Measuring the Impact of Packet Reordering on Internet Protocol Networks

Colin Michael Arthur

A thesis submitted for the degree of Doctor of Philosophy to
the Department of Electronic & Electrical Engineering
University of Strathclyde

December 2008

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has lead to the award of a degree.

Signed:  Date: 30/12/08

Abstract

Packet Reordering in IP networks is a phenomenon which is becoming increasingly important in network performance analysis. Reordering is a consequence of network equipment manufacturers increasing switch and link level parallelism within networks, in the quest for performance, reliability and fiscal gains. Wireless technologies are also expected to increase the amount of packet reordering observable in an end-to-end path.

This thesis addresses the issue of measuring the impact of packet reordering on Internet traffic, by proposing a number of measurement methodologies and metrics. Previous techniques assume that packet reordering does not often occur, or make assumptions which severely limit the results obtained. This thesis proposes a two-point passive measurement technique, which improves on previous methods by allowing lightweight measurement of the amount and extent of reordering observed in a TCP flow, and classification of the cause of each reordering-induced packet retransmission. A large testbed measurement study performed using this technique indicated that TCP is tolerant to large percentages of reordered packets, providing that the delay of these packets is maintained below a threshold relative to Round-Trip-Time. This study further indicated that the effects of TCP packet reordering are not always negative. In specific scenarios reverse-path reordering can increase the overall throughput of a flow. This thesis further proposes a mid-point passive Measurement Technique and Visualisation Metric of TCP packet reordering, designed to classify out of sequence packets for many thousands of concurrent TCP flows. This technique is lightweight to implement and does not require symmetric TCP connections to operate. Finally, this thesis argues that future packet reordering metrics must correlate reordering observed at the network layer, with the resulting impacts observed at the application layer. An example of an application-specific metric is developed for MPEG-4 video over UDP traffic, and this metric is used to describe the effects of packet reordering on streamed video traffic.

Acknowledgements

I would like to sincerely thank David Harle, without whose constant support and encouragement throughout, I would not have been able to complete this work.

I also am greatly indebted to Andrew Lehane – for his mentorship, his critique and his energy. Andrew has helped me in many ways throughout my short research career, and I can only hope that in the future, I can help others as he has helped me.

I am indebted to many people at Strathclyde University throughout my period as a PhD student and a Research Assistant – all were very generous with their time, their ideas and their encouragement; Demessie Girma, Robert Atkinson, Alisdair McDiarmid, John Bush, Ian Robertson, Ian Armstrong, Kurian Oommen, Omar Tayan, Christos Tachtatzis, Kostas Sasloglou, Joan Cortes, Stéfan Martin and Gordon Morison.

I would also like to thank my new colleagues within the Measurement Research Laboratory at Agilent Labs for their support, and for always allowing me to learn from them; Frankie Garcia, Alex Tudor, Martin Curran-Gray, Kevin Mitchell, Tony Kirkham, Kathy Graham and Lance Tatman.

I am indebted to my friends, whose constant motivation and support was greatly appreciated during the unrelenting write-up months; Elizabeth Watson, Alastair Davis, Brian Turnbull, Euan Robertson, Catalina Aguirre, Kevin McClenaghan and Ryan Tumilty.

Swee deserves a special mention; her consistent help and encouragement, throughout the highs and the lows, was very important to me in the latter stages of writing up.

Finally, I would like to thank my Mum, my Dad, my Gran and my Sis Laura. I am lucky to have a family that supports me in everything that I do, and I am sure that this PhD has been as traumatic for them as it has been for me. My Gran has a greater belief in my abilities than I ever will, and for that, I am truly grateful.

List of Publications

C. M. Arthur, D. Girma, "Unified Method for Video Traffic Modelling on IP Networks". IEE Electronics Letters, Vol. 38 No. 10, pp 492-494, May 2002.

C. M. Arthur, D. Girma, "An Experimental Platform for Video Traffic Analysis over Lossy IP Networks". Fifth IEE European Personal Mobile Communications Conference (EPMCC), April 2003, Glasgow.

C. M. Arthur, D. Girma, D. Harle, A. Lehane, "The Effects of Packet Reordering in a Wireless Multimedia Environment". First International Symposium on Wireless Communication Systems, September 2004, Mauritius.

C. M. Arthur, A. Lehane, M. Curran-Gray, D. Girma, "Real Time Monitoring of TCP Flows", UK Patent Application GB2430577, Filed September 2005, Published March 2007. US Patent Office Application 20070070916, Published March 2007.

C. M. Arthur, A. Lehane, D. Harle, "Keeping Order: Determining the Effect of TCP Packet Reordering", Second International Workshop on Internet Packet Dynamics, IPDy 2007, June 2007, Greece. Winner Best Paper Award.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

Ack	TCP Acknowledgement Number
ADSL	Asymmetric Digital Subscriber Line
AS	Autonomous System
ASF	Microsoft Advanced Systems Format
ATM	Asynchronous Transfer Mode
awnd	TCP Receiver's Advertised Window
BDP	Bandwidth Delay Product
BER	Bit Error Rate
CPU	Central Processing Unit
cwnd	TCP Congestion Window
DARPA	Defence Advanced Research Projects Agency
DEC	Digital Equipment Corporation
DiffServ	Differentiated Services
dupthresh	TCP Fast Retransmission Duplicate Acks Threshold
ECN	Explicit Congestion Notification
EP	Expected Position
FDDI	Fibre Distributed Digital Interface
FIFO	First In First Out
FIN	TCP Finish Flag
FSM	Finite State Machine
FTP	File Transfer Protocol
GPS	Global Positioning System
GPU	General Purpose Processing Unit
HDTV	High-definition television
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IHL	Internet Header Length
IOS	Cisco Internetwork Operating System
IP	Internet Protocol version 4

IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPFIX	Internet Protocol Flow Information Export
IPID	IP Identification Field
IPMON	Sprint IP Monitoring Project
IPPM	IP Performance Metrics Working Group
IPTV	Internet Protocol Television
ISN	Initial Sequence Number
ISP	Internet Service Provider
L3 VPN	Layer 3 Virtual Private Networks
LAN	Local Area Network
MANET	Mobile Ad-hoc Network
MIB	Management Information Base
MMAP	Memory Map Extensions
MMS	Microsoft Media Services
MOS	Mean Opinion Score
MPEG	Moving Pictures Experts Group
MPLS	Multiprotocol Label Switching
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NACK	Negative Acknowledgement
NEM	Network Equipment Manufacturer
NESN	Next Expected Sequence Number
NIST	National Institute of Standards and Technology
NMS	Network Management System
NS-2	Network Simulator version 2
NTP	Network Time Protocol
OOS	Out Of Sequence
OP	Observation Position
OSI	Open Systems Interconnection
PSTN	Public Switched Telephone Network
QCIF	Quarter Common Intermediate Format
QoE	Quality of Experience
QoS	Quality of Service
RBD	Reorder Buffer Density
RD	Reorder Density
RED	Random Early Detection
RFC	Request For Comments
RIPE	Réseaux IP Européens
RSH	Remote Shell
RST	TCP Reset Flag
RSVP	Resource ReSerVation Protocol

RTO	Retransmission Timeout
RTP	Real-time Transport Protocol
RTT	Round Trip Time
rwnd	TCP Receiver's Advertised Window
SACK	TCP Selective Acknowledgement
SDK	Software Development Kit
Seq	TCP Sequence Number
SLA	Service Level Agreement
SLS	Service Level Specification
SMSS	Sender Maximum Segment Size
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Networking
SPAN	Switched Port Analyser
ssthresh	TCP Slow Start Threshold
Syn	TCP Synchronise Flag
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TTL	Time To Live
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WM	Microsoft Windows Media

Chapter 1

Introduction

In December 1999 the IEEE/ACM Transactions on Networking published a paper entitled "Packet Reordering is Not Pathological Network Behavior" [Benn99]. Bennett et al. had intended to prove the hypothesis that the reordering of packets in the Internet is an ever increasing phenomenon. The results of their study, performed in January 1998, indicate that the probability of a session, running through the US MAE-East exchange, experiencing packet reordering was over 90%. Intuitively, Bennett et al. cited that the reason for the large proportion of flows experiencing reordering was the presence of parallelism on the routes taken by the packets flowing through the network.

However, the reason for this reordering, and the parallelism which caused it, was not immediately obvious. Bennett discovered that much of the packet reordering observed is not, as was first expected, due to multi-path routing or broken network equipment causing packets to traverse different logical paths, but occurred as a result of switch and link-level parallelism. This included link-level striping and switches that allow packets

travelling between the same source and destination to take different paths through the internal switch hardware.

Bennett's work represented a significant contribution to the field of network science since packet reordering can have a measurable impact on both network and application performance. For example, out-of-order arrival of packets can cause apparent loss of data in real time flows, such as voice over packet and video streams. Reordering is also detrimental to Transmission Control Protocol (TCP), causing it to use available capacity less effectively, and lose the TCP self-clocking property, resulting in irregular data transmission.

1.1 The Increase of Internet Parallelism

It is clear from the literature that the level of parallelism in network paths is on the increase, although what overall impact this will have on packet reordering is less clear. Load balancing in network switches introduces local parallelism, which can allow packets flowing between the same source and destination to take different paths within the switch. Simple economics also has a bearing; it is often more cost effective to put two components in parallel than to use one component that has twice the speed[Benn99]. For example, when purchasing long-haul serial links many tariffs offer link bandwidths that are multiples of each other. Parallel links are also a very useful way to improve reliability; if the parallel links follow different physical paths, the virtual link they implement is generally less vulnerable to single-point failures. Large businesses, Internet Service Providers (ISPs) and their vendors are therefore aggressively promoting parallel links. In a survey of 38 major ISPs conducted in mid-1997 [Gare97], only two of the smaller ISPs did not have parallel uplink paths between nodes.

Bennett's paper explains how packet reordering can impact network performance, by exemplifying TCP during packet reordering. In the presence of forward path reordering, TCP has great difficulty opening its congestion window and makes inefficient use of available link capacity through unnecessary retransmissions. During reverse path reordering events (reordering of acknowledgments), TCP loses self-clocking and data

transmission becomes very irregular, with a large quantity of short duration data bursts instead of more evenly loaded flows. However, the impact of packet reordering is not limited to TCP. Any protocol that is reliant on the ordered arrival of packets can be affected by this phenomenon. For example, RTP flows, based upon UDP, will not be immune to packet reordering. The impact on TCP is to slow traffic and reduce throughput. However, the impact upon real-time flows is often far more severe. Even low levels of reordering increase the buffer memory requirements at the receiver as well as increasing processing related latencies. However, as reordering becomes even more prevalent buffering becomes ineffective; the result is degradation in the quality of the delivered service. For example, in applications such as voice over packet there is no time to retransmit data, and so the supposedly missing, but in reality late information, has to be replaced at the application level by 'white noise' thus causing loss of intelligibility.

Bennett et al. also found that one of the challenges of understanding this form of reordering is that this type of parallelism is not easily measured. During Paxson's measurement experiments with reordering in end-to-end routing, the different network paths taken by packets were clearly indicated by the different addresses of the routers they traversed [Paxs96]. However, in link and local parallelism, the only indication of the existence of parallel links, may be that a particular hop exhibits varying levels of delay.

1.2 Characterising Packet Reordering

There have been several proposals to create protocols that can either adapt, or are robust, to packet reordering. However, evaluating their effectiveness requires a good understanding of the dynamics of the reordering processes prevalent in the Internet. Unfortunately, Internet packet sequencing is still a poorly characterised and under-studied behaviour. Measurement studies in this field are, to some extent, contradictory, including two papers presented at the Internet Measurement Workshop 2002 "Measuring Packet Reordering" [Bell02] and "Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone" [Jais02]. Both papers describe in detail, how packet reordering can impact network performance and attempt to measure the problem using simple active and passive measurements techniques. Unfortunately, their measurement results

on live Internet traffic do not correlate with Bennett [Benn99]. Bellardo and Savage [Bell02] used a single active probe based in UCSD testing 40 different destinations over a 20 day period. Bellardo's results show some level of reordering on over 40% of paths tested and 15% of individual measurements with out-of-sequence packets. Bellardo's study indicates that the amount of packet reordering varies upon a daily basis and can range from 5% to 25% of packets appearing out-of-sequence.

Conversely Jaiswal et al.[Jais02], who passively observed TCP flows at a single point in the middle of the Sprint backbone, measured only 13.6% of flows with some form of reordering present and 5% out-of-sequence packets. These conflicting findings might be due to different network topologies, switch architectures, underlying link layer protocols or the measurement techniques used, but both studies are sufficiently different in nature that drawing conclusions without further work is difficult, thus providing an ideal stimulus for further research into measuring and understanding this phenomena.

In November 2006, the IETF IPPM Working Group published "Packet Reordering Metrics" RFC 4737 [Mort06], after thirteen drafts of the metric has been proposed and discussed. This metric was strongly contested by Jayasumana et al., who published "Improved Packet Reordering Metrics' [Jaya08] in June 2008, thus indicating that there remains disagreement in the research community on defining a metric which meaningfully, accurately and unambiguously characterises packet reordering.

Clearly, further exploration of link and local level parallelism, how it drives packet reordering and impacts on network performance is important. Previously, there has been little work published approaching this problem from first principles, investigating how parallelism drives packet reordering, and then correlating this with the resulting performance impacts, on reasonably large scale networks.

1.3 Thesis Organisation

This thesis is organised as follows. Chapter 2 discusses the Internet Protocol Suite and the various options and additions to the protocol and current implementations. The effects of Packet Reordering on TCP are discussed, and an overview of Internet Measurement techniques is presented.

Chapter 3 presents the prior art in this area, by presenting a taxonomy of Metrics and Methods used to characterise Packet Reordering in the Internet, and the results obtained by using these methods. The taxonomy classifies these metrics and methods as active or passive techniques, and discusses the advantages and limitations of each technique.

Chapter 4 presents a methodology for simulating Packet Reordering, and the development of a testbed and experimental network to empirically measure packet reordering. A two-point passive measurement technique is designed and prototyped, which improves on previous methods by allowing lightweight measurement of the amount and extent of reordering observed in a TCP flow, and classification of the cause of each reordering-induced packet retransmission. A large testbed study of over 30,000 TCP flows is performed to investigate and measure the behaviour of TCP during reordering.

Chapter 5 presents an investigation and the development of a mid-point passive measurement technique of TCP Packet Reordering, which allow improved classification of out of sequence packets, an improved measure of TCP Goodput, and a Visualisation Metric for indicating the performance of TCP throughout the lifetime of a flow.

Chapter 6 presents a case-study of non-TCP traffic and how it is affected by Packet Reordering; an example of an application-specific packet reordering metric is developed for MPEG-4 video over UDP traffic, and this metric is used to describe the effects of packet reordering on streamed video traffic.

Chapter 7 presents conclusions and proposals for future work.

Chapter 2

The Internet Protocol Suite

2.1 Introduction

The growth of the Internet has been well documented [Hobb97], from the very first networking research carried out by the US Advanced Research Projects Agency in 1957, through to the exponential growth of connected hosts experienced and measured in the last decade. [Hobb06]

A significant part of this exceptional growth rate is due to the research conducted in the 1970s into the first host-to-host protocols, which resulted in the development of the Transmission Control Protocol [Post81b] over Internet Protocol [Post81a] (TCP/IP) Suite. This has allowed a multitude of heterogeneously interconnected systems, all with

diverse characteristics, vendors and operating systems, to communicate seamlessly with each other, over various communications channels.

This chapter provides an overview of the TCP/IP protocol suite, discussion of the various enhancements which have been added to TCP since its initial development, an overview of Network Measurement Science, and discussion of the effects of Packet Reordering on TCP flows.

2.2 The Internet Protocol Suite

The 'Internet Protocol Suite', often generically referred to as 'TCP/IP' is considered to be a 4-layer system [Stev94] [Bra89] as illustrated in Figure 1, with each layer responsible for a particular aspect of the transmission system:-

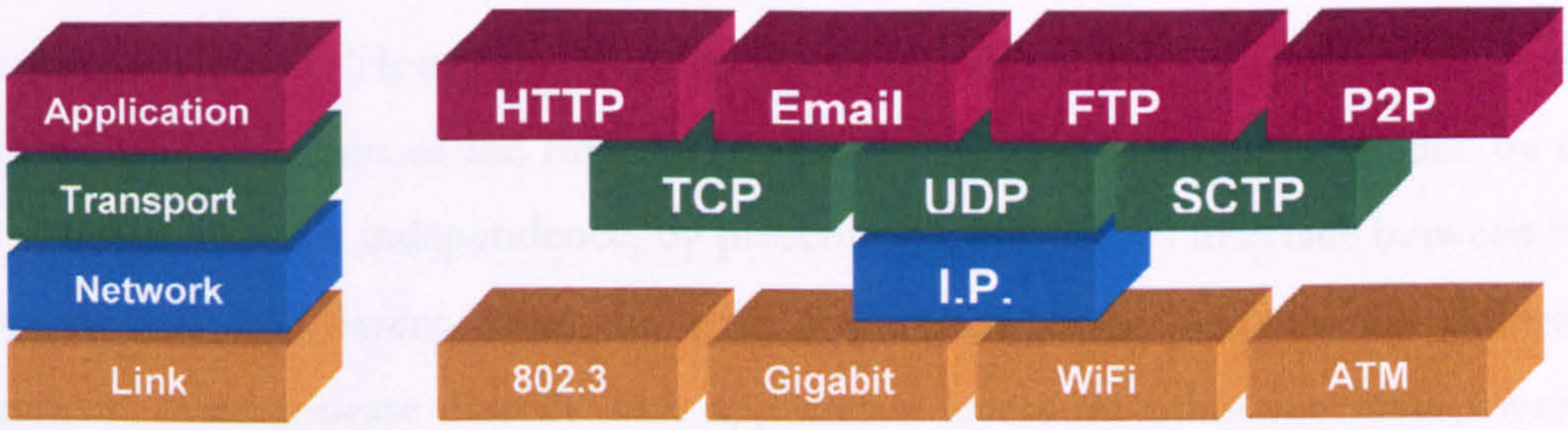

Figure 1 - The Internet Protocol Suite

The **Application Layer** is responsible for providing service to particular applications on an end host, such as Web, Email, and file transfer systems, through application layer 'Messages'.

The **Transport Layer** provides additional functionality above the Network Layer, to provide a particular type of service between two hosts. For the vast majority of Internet traffic, the two main transport protocols in use are TCP (Transmission Control Protocol) [Post81b], and UDP (User Datagram Protocol) [Post80], discussed in greater detail later in this chapter. Protocol Units at this layer of the suite are termed as 'Segments' when discussing TCP, and 'Datagrams' when discussing UDP [Soco91].

The **Network Layer** is responsible for the routing of packets around the network, fragmentation of packets if required, and the structures for addressing in the Internet. The network layer must [Brad89] implement both Internet Protocol (IP) [Post81a] and the Internet Control Message Protocol [Post81c], which provides the routing, diagnostic and error capabilities in the IP suite. Protocol units in this layer are described as "Datagrams' [Soco91] when referring to the end-to-end unit of data passed from network layer to link layer, and IP 'Packets' when referring to mid-point forwarding. Due to fragmentation, an IP Datagram may be transmitted as a single, or multiple, IP Packets.

The **Link Layer** is the network interface layer, and is normally considered to be the device driver in the operating system and the corresponding network interface cards in the end hosts. End hosts handle the details of physically interfacing with the relevant transmission media. Protocol units at this layer of the stack are usually termed 'Frames'.

Figure 1 illustrates the 'Hourglass Analogy' [Deer01] which has been argued to be the main factor in the success of the Internet Protocol. The 'thin waistline' provided by IP, allows physical network independence, by presenting a ubiquitous interface between the application and link layers, thus allowing application layer services on different machines, to communicate directly with application layers on other machines, over a multitude of per-hop link media, creating one 'end-to-end' [Salt81] path.

2.2.1 Internet Standardisation

Internet Standardisation is a loosely defined process, driven by volunteers from academia and industry, in four main organisations; the Internet Society, the Internet Architecture Board, the Internet Engineering Task Force (IETF), and the Internet Research Task Force. The majority of standardisation work is carried out by the IETF and published in the form of incremental documents called 'Request for Comments' (RFC) [Malk93]. RFCs can describe protocols standards, describe best practice, or be informational. When describing protocol standards, an RFC will describe a protocol as either Standard, Draft Standard, Proposed Standard, Experimental, Informational or Historical, with various aspects and features of each protocol, marked with various

requirement levels[Brad97a].

It should be noted though, that standardisation itself is a lengthy process – the Transmission Control Protocol is planned to move from 'Proposed Standard' to 'Standard' in October 2008 [Allm07], over 20 years since original conception. Indeed, it is worth noting that once a standard is documented in an RFC, there are no formal methods to enforce compliance, and misbehaving implementations [Chen07] [Sava99] are commonly observed [Medi05] in the Internet.

2.2.2 Internet Protocol version 4

The Internet Protocol version 4 (IP) [Post81a], commonly referred to as 'RFC 791', was created in September 1981 and describes the DARPA Internet Protocol Specification, for transmitting a packet across a packet-switched communications network.

RFC 791 is specifically limited in discussion so as to only describe the two basic functions of addressing and fragmentation. Therefore there are no mechanisms to allow end-to-end data reliability, flow control nor sequencing, and it is assumed that IP will be used in conjunction with other higher-layer protocols in the Suite to provide these additional functionalities. IP is considered to be a 'best effort' unreliable, datagram delivery service. Sources and Destinations are identified by fixed length addresses, with mid-point hosts given the ability to perform fragmentation and reassembly of packets over intermediate networks with varying Maximum Transmission Units (MTU).

The standard describes how packets are to be moved by passing from one 'Internet Module' in a host, to another, until the final destination address is reached[Post81b]. A host which is implemented and designed for the specific task of forwarding IP packets is called an IP 'router' [Soco91] [Bake95]. Each datagram must be considered independently of all others; there are no connections or logical circuits, and there are specifically no guarantees of reliability, flow control, or datagram sequencing. This allows a light-weight and simple implementation in mid-point routers; a router does not record state information to maintain a connection, and it is acceptable for a router to randomly drop packets from its input queue should congestion occur. A router is also

allowed to output packets travelling between the same source destination pair through different output paths, and therefore no guarantees can be made on packet sequencing at the destination host[Benn99].

The basic functionalities of Addressing and Fragmentation performed by the IP layer are now discussed.

2.2.2.1 Addressing

The purpose of addressing is to provide an interface between the local network addressing structure, and Internet-wide routing. Addressing avoids the complexity of naming, separately carried out by Domain Names [Mock87], or an end user being required to specify routes between nodes.

Upon receipt of a packet [Bake95], the router will validate the IP header, process any relevant IP options specified in the header, and then examine the IP Destination Address in order to make a forwarding decision. An IP address can be partitioned into two constituent parts; a Network Prefix, and a host number. The Network Prefix is compared with the router's routing table, and the next hop IP address for the packet and relevant output interface are determined. This continues until reaching a router capable of mapping the Destination IP address to a local network address, whereupon the packet is delivered to the end host.

IPv4 addresses, are of a standardised [Post81a] fixed length of 32 bits, with a convention [Soco91] of writing each of the 4 bytes in decimal, separated by a period. The original specification has undergone significant developments, through Classless Inter Domain Routing [Full93], and Network Address Translation [Sris01]. Although the IPv4 address space is limited, thus motivating the development of IPv6 [Deer98], a number of challenges [Wadd02] have slowed actual deployment of IPv6, and it is expected that IPv4 will remain the predominant addressing technique for the foreseeable future.

2.2.2.2 Fragmentation

The purpose of Fragmentation is to allow the transmission of IP packets across an end-to-end path, through constituent intermediate networks, with varying sizes of MTU. This is particularly common when a source host is located on an 802.3 Ethernet network, where the size of each packet could be as large as 1500 bytes. All IP compliant hosts must be able to forward a packet of 68 bytes without performing further fragmentation (60 bytes maximum header size, and 8 byte minimum fragment size)[Post81a]. Additionally, every host must have the capability of receiving at least a 576 byte IP packet, either in one single packet, or in multiple packet fragments.

Fragmentation is performed transparently to higher layers, and re-assembly of fragments is only performed at the destination host in a connection – individual fragments of packets are each routed individually, and therefore may transit differing disjoint paths prior to arrival at the destination.

Fragments are reassembled using the Identification Field (IPID) in the IP header, in conjunction with the Fragment Offset Field, Length Field and More Fragments Field. The IPID uniquely identifies each packet sent by a host, and is used together with the source and destination addresses and protocol fields, to identify datagram fragments for reassembly. The sending host must therefore ensure that the IPID is unique for each source/destination pair, and protocol, for the time that IP packet or its fragments, are alive in the Internet. Most TCP/IP Linux implementations increment a Kernel variable each time an IP datagram is sent [Stev94]. The IPID, therefore, normally increments predictably each time a datagram is sent, and is often exploited in IP header compression techniques[West06].

2.2.3 IPv4 Header Format

Figure 2 illustrates the format of the IPv4 Datagram Header applied to packets upon leaving an Internet host. Bit positions are illustrated along the top of the header and it should be noted that IP packets will always be 32 bit aligned for optimum performance

on commodity hardware. Each field of the header is populated as follows:-

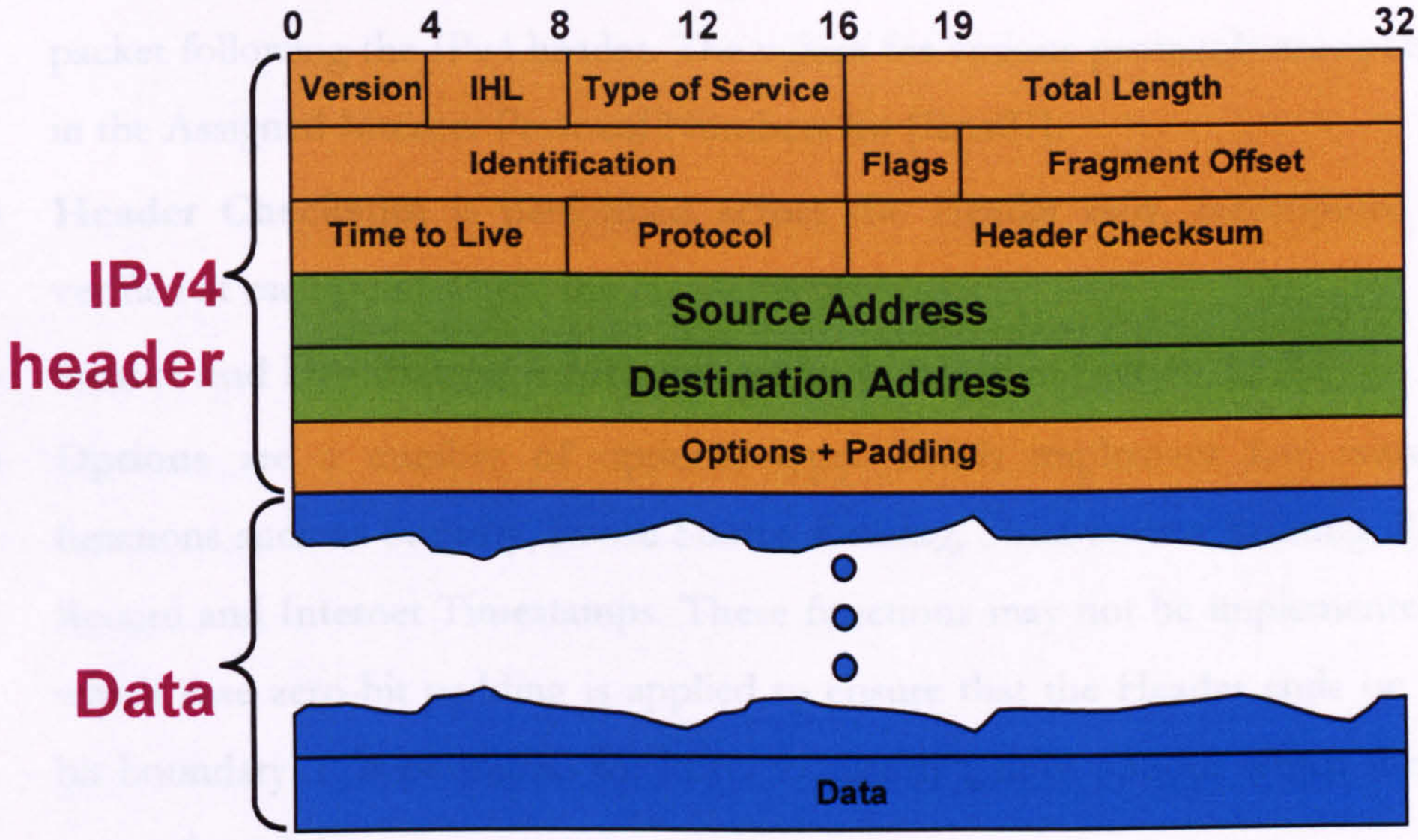

Figure 2 - IPv4 Header Format

- **Version** indicates the version of the internet header, which for RFC 791 - IPv4, is the value 4, thereby specifying the format of the following fields.
- **IHL** is the length of the internet header in 32 bit words, and thus a pointer to the beginning of the data. A correctly formed IPv4 header will have a minimum value of 5, corresponding to the minimum header length of 160 bits.
- The **Type of Service** field historically provided an indication of the Quality of Service desired [Alqm92], which was examined by mid-point routers when determining an onward path. Currently, it is used as either the Differentiated Services (DiffServ) Field [Nich98] in DiffServ networks [Blak98], or Explicit Congestion Notification (ECN) Field [Rama01] on compatible hosts.
- **Total Length** is the total length of the datagram, measured in octets, including the packet header. A 16 bit field allows a datagram of up to 65,535 octets.
- **Identification, Flags and Fragment Offset** are variables set by the sender to allow packet fragmentation, as discussed in Section 2.2.2.2.
- **TTL** indicates the maximum time that the datagram is allowed to remain in the

Internet system, and is decremented by each host processing that packet. The current recommended default value is 64 [Brad89]

- **Protocol** indicates the next type of protocol header which will appear in the packet following the IPv4 header. The values for various protocols are specified in the Assigned Internet Protocol Numbers list [Iana07].
- **Header Checksum** is performed across the Header only, recomputed and verified at each point where the Header is processed.
- **Source and Destination Addresses** are as discussed in Section 2.2.2.1
- **Options** are a number of optional types which implement less common functions such as Security, Loose Source Routing, Strict Source Routing, Route Record and Internet Timestamps. These functions may not be implemented, in which case zero-bit padding is applied to ensure that the Header ends on a 32 bit boundary. It is permitted for hosts to silently ignore options which they do not understand.

2.3 User Datagram Protocol

UDP is a connectionless transport protocol [Post80], and provides a simple interface to IP when a connection-oriented guaranteed delivery service is not required. A UDP header consists of a Source and Destination Port Numbers, to allow multiplexing of packet flows between hosts, a Length header and a Checksum header.

UDP provides a very simple service with no congestion or flow control, and no method of retransmitting lost packets. For non-real-time applications where reliable transport is important, TCP would be the protocol of choice.

2.4 Transmission Control Protocol

Transmission Control Protocol (TCP) [Post81b] is the predominant transport-layer protocol operating in the Internet [Medi05], and provides a reliable full-duplex connection, across an end-to-end path, between two Internet hosts. TCP is used in

applications where accuracy and completeness of data take precedence over latency, and is therefore suited to applications such as E-mail, Web traffic and file transfer.

2.4.1 Reliable Transmission

TCP operates a 'sliding window' over a continuous byte stream of data from the application layer, packetising this data into Segments. A Sequence Number is conceptually assigned to every byte transmitted, and positive Acknowledgements (Acks) are required from the receiving host, thus indicating that each byte has been successfully received. Expiration of a timer at the sending host, before an Ack is received, indicates that the segment has been lost and that a retransmission should be scheduled.

Upon arrival at the receiving host, TCP specifies that the Sequence Numbers should be used to correctly reorder segments that may have arrived out of order and to reveal duplicates. TCP is therefore able to accommodate the loss, damage, duplication or reordering of packets that may be caused by any of the underlying networks along the end-to-end path.

Flow control is performed in TCP, allowing the receiving TCP to control the rate at which the sender transmits data, by returning a Window with every Ack, indicating available space in the receiver buffer. This 'Receiver Advertised Window' (*rwnd*) is flow control governed by the Receiving TCP based on the Receiver's buffering and processing capabilities. Flow control performed at the Sending TCP is discussed in Section 2.4.6.

Multiplexing is achieved in TCP through the use of Port Numbers on each host which, when concatenated with the host address, are termed as a 'Socket'. A pair of 'Sockets' uniquely identify each 'Connection', and thereby allows multiple TCP connections to terminate on any host.

2.4.2 TCP Header Format

Figure 3 illustrates the TCP header, with fields defined as follows:

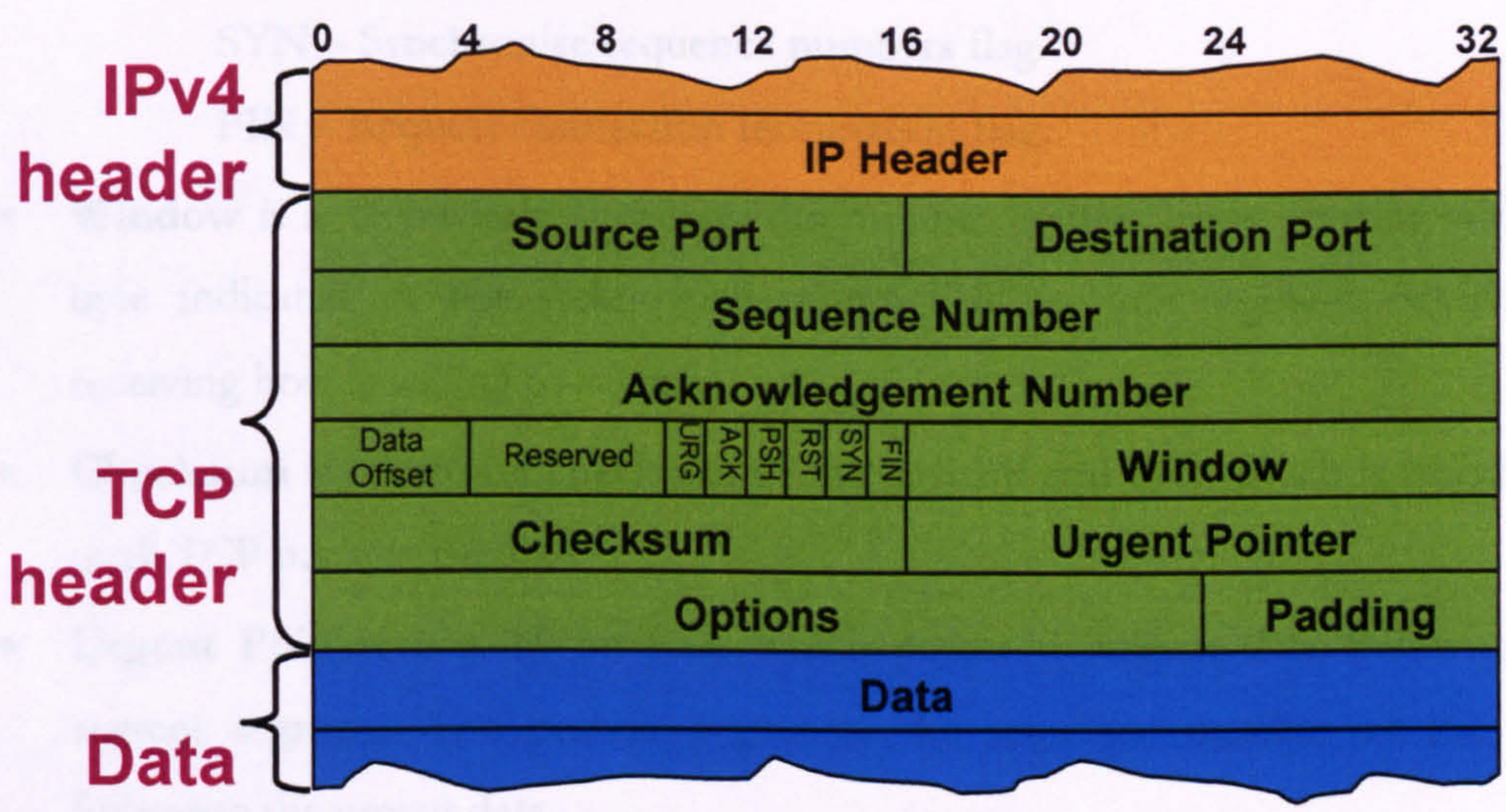

Figure 3 - TCP Header Format

- **Source Port** is the 16 bit source port on the source host.
- **Destination Port** is the 16 bit destination port on the destination host.
- **Sequence Number** is the 32 bit Sequence Number of the first data octet in the segment's payload (except when the SYN flag is present). If SYN is present the sequence number is the Initial Sequence Number (ISN) of the connection, and the first data octet is ISN+1.
- **Acknowledgment Number** field is valid only if the ACK control bit is set, and contains the 32 bit Next Sequence Number that is expected at the destination host.
- **Data Offset** is the 4 bit indicator of the number of 32 bit words in the TCP header. The TCP header is always 32 bit aligned.
- **Reserved** is 6 bits reserved and must be zero, or, if Explicit Congestion Notification [Rama01] is enabled, are used as described in RFC3168 [Rama01]

and RFC3540 [Spri03].

- **Control Bits** are 6 single-bit control flags as follow:-

  URG – Urgent Pointer flag.

  ACK – Acknowledgement flag.

  PSH – Push function flag.

  RST – Reset connection flag.

  SYN – Synchronise sequence numbers flag.

  FIN – Request connection termination flag.

- **Window** is a 16 bit field signalling the number of data bytes, starting with the byte indicated in the Acknowledgement field in this segment, which the receiving host is willing to accept.
- **Checksum** is the 16 bit checksum of the payload and data, which is mandatory in all TCP packets[Brad89]
- **Urgent Pointer** is a 16 bit field which points to urgent data following the current segment. The pointer points to the sequence number of the octet following the urgent data.
- **Options** may appear at the end of the TCP header and are multiples of 8 bits in length, with **Padding** to ensure that the packet is 32 bit aligned. Options may indicate Timestamp options [Post81b], Maximum Receive Segment Size (MSS), Selective Acknowledgements (SACK) [Math96], or other options. It is specified that a TCP must be able to receive an option in any segment, and ignore without error any option not implemented [Brad89].

2.4.3 Sequence Numbers and Acknowledgements

The idea of Sequence Numbers is important in TCP as, conceptually, each octet of data is assigned a sequence number. Once the 'sliding window' has isolated a stream of octets to form into a segment, it is the sequence number of the first octet of data in a segment that is used as the sequence number for the complete packet; termed the 'Segment Sequence Number'. The Segment Sequence Number is placed in the Sequence Number field of the TCP header.

In the reverse direction, segments carry an Acknowledgement number, placed in the Acknowledgement Number field of the TCP header, with the ACK flag set, so as to mark the field as being valid. An Acknowledgement number in TCP is the sequence number of the octet that the receiver is next expecting to receive. Therefore, in a simple scenario, the Acknowledgement of a packet would be the (Sequence Number + Packet Length) of the most recent packet received.

Acknowledgements in TCP are cumulative as illustrated in Figure 5. A Receiver may operate the 'Delayed Ack' algorithm [Clar82], in which case a Sender, receiving acknowledgement of sequence number *x*, should interpret this to mean that the Receiver has correctly received all bytes up to but not including *x*. Cumulative Acks can substantially reduce protocol overhead [Brad89], but excessive delays can disturb the round-trip sampling and packet 'clocking' algorithms[Jaco88].

2.4.4 Establishing a Connection

A connection is established through a three-way handshake mechanism, as illustrated in Figure 4, where the TCP modules of the Sender and Receiver synchronise on each other's Initial Sequence Numbers (ISN). Each TCP selects their ISN through an implementation dependent mechanism, and transmit a packet with the SYN (Synchronise) flag enabled. Once each TCP has positively acknowledged the ISN of the other TCP, by transmitting Acks, the connection is established and data transmission can begin.

For sequence number purposes, the SYN packet sent to establish a connection, is considered to occur before the first actual data octet of the segment in which it occurs, while the FIN packet, sent to signal the end of a connection, is considered to occur after the last actual data octet in a segment in which it occurs.

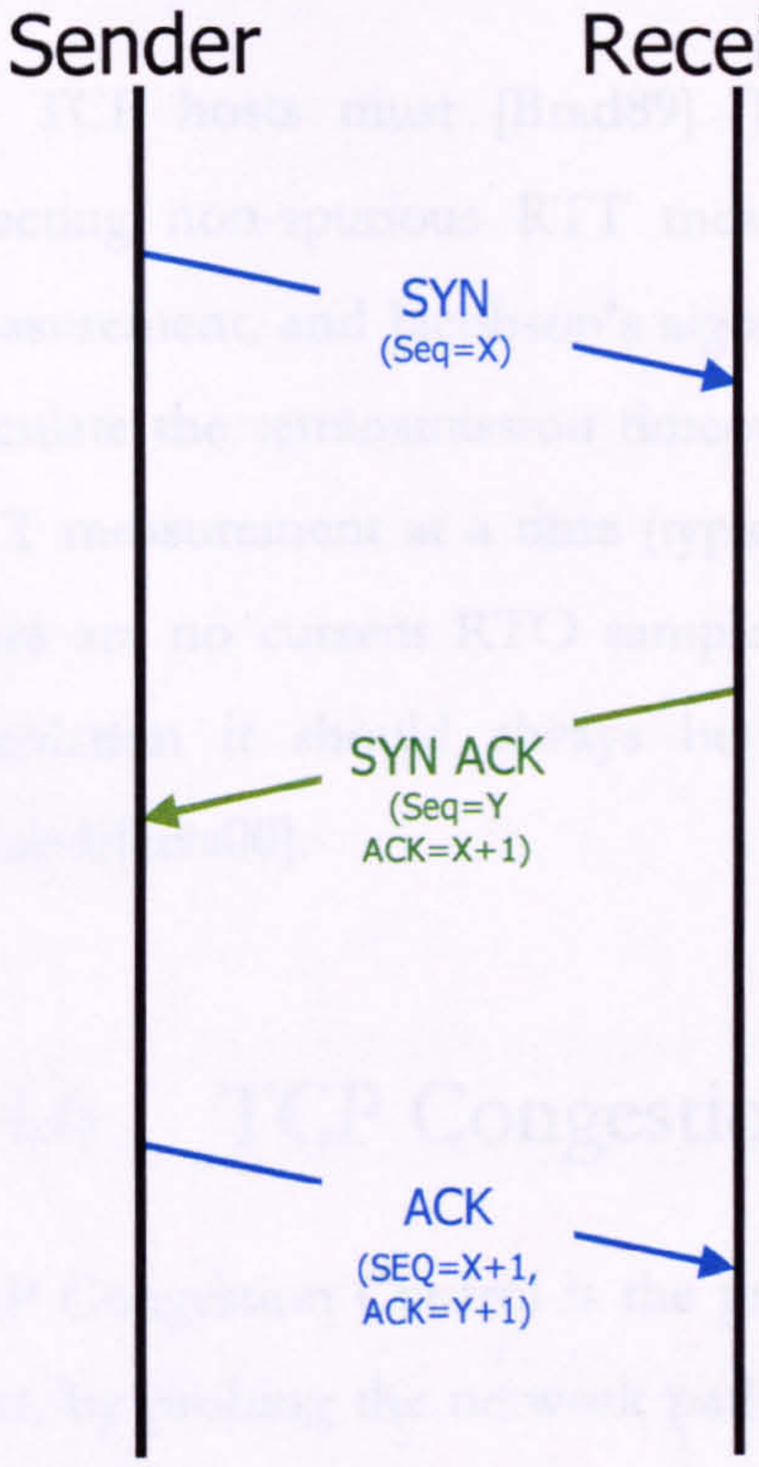

Figure 4 - TCP 3-way Handshake

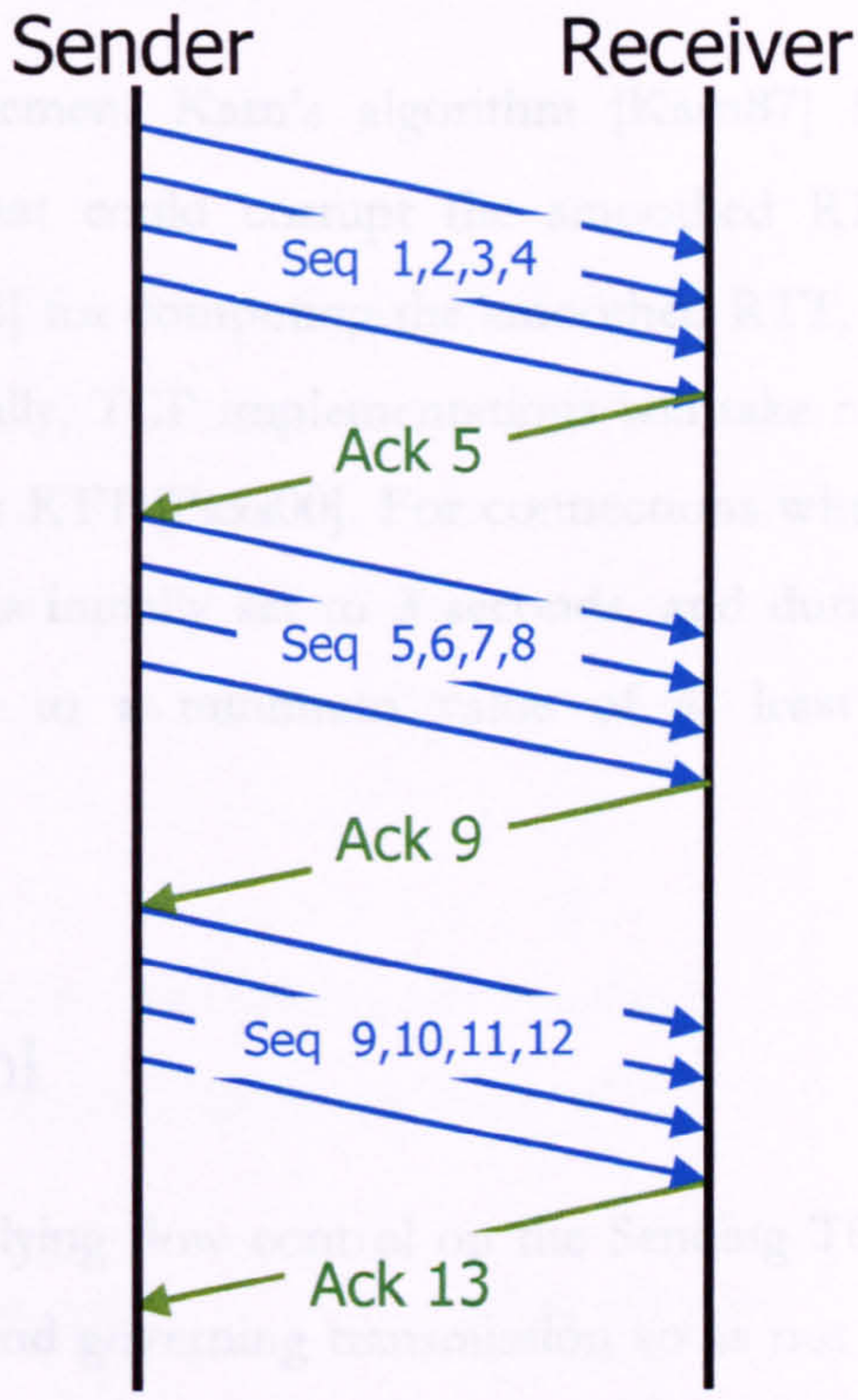

Figure 5 - TCP Cumulative Acknowledgements

Closing a connection can happen in two ways [Brad89] – either party can signal they wish to close by sending a FIN handshake, or an 'abort' can be sent when a RST segment is sent, and the connection is discarded by both parties.

2.4.5 Retransmission Timeout

TCP reliability is implemented through the use of retransmissions, should loss be detected in the network. A TCP sender will maintain a copy of each transmitted segment, and a timer is initialised which will count until an acknowledgement is received which encompasses the sequence number of that segment. Should an acknowledgement not be received before a 'Retransmission Timeout' (RTO) value is reached, the Sender will assume that the segment has been lost and will initiate the retransmission process.

To enable this loss detection, the TCP sender requires a method of calculating the Round Trip Time of connections, which must be calculated on a per source-destination basis, and must be dynamically updated to ensure that any time-varying effects of the end-to-end path are considered.

All TCP hosts must [Brad89] [Paxs00] implement Karn's algorithm [Karn87] for selecting non-spurious RTT measurements that could corrupt the smoothed RTT measurement, and Jacobson's algorithm [Jaco88] for computing the smoothed RTT, to calculate the retransmission timeout. Traditionally, TCP implementations will take one RTT measurement at a time (typically once per RTT)[Paxs00]. For connections where there are no current RTO samples, the RTT is initially set to 3 seconds, and during calculation it should always be rounded up to a minimum value of at least 1 second[Paxs00].

2.4.6 TCP Congestion Control

TCP Congestion Control is the process of applying flow control on the Sending TCP host, by probing the network path capability, and governing transmission so as not to overwhelm the intermediate nodes. Congestion Control [Allm99] [Allm07] is specified by four closely related algorithms; Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery [Jaco88][Jaco90]. [Brad89] mandates that a TCP Sender must implement Slow Start and Congestion Avoidance, with Fast Retransmit and Fast Recovery later optionally introduced in [Allm99].

TCP maintains three variables per connection at the Sending TCP host:

- The **Congestion Window** (*cwnd*) is a sliding window, which limits the amount of data a Sending TCP can transmit into a network, before receiving an Acknowledgement.
- The **Receiver's Advertised Window** (*rwnd*) is flow control from the Receiving TCP, indicating a window size of data the receiver is willing to accept.
- The **Slow Start Threshold** (*ssthresh*) is a value used to decide whether the Sending TCP is transmitting packets using the Slow Start or Congestion Avoidance algorithm, and if the *cwnd* variable should be adjusted.

The minimum of *cwnd* and *rwnd* controls the amount of data a TCP sender can transmit into a network, before an Ack is received from the recipient.

The aim of TCP Congestion Control is that transmission should be 'self-clocking' [Jaco90], where the Sending TCP uses feedback in the form of Ack packets, to strobe packets into the network, as other packets leave the network. Before this equilibrium can be reached, termed the Congestion Avoidance phase, the Sender must aggressively probe the network in order to find the end-to-end capacity of the path – termed the Slow Start phase.

The Slow Start algorithm is used to govern transmission when *cwnd* < *ssthresh*, and the Congestion Avoidance algorithm is used to govern transmission when *cwnd* > *ssthresh*.

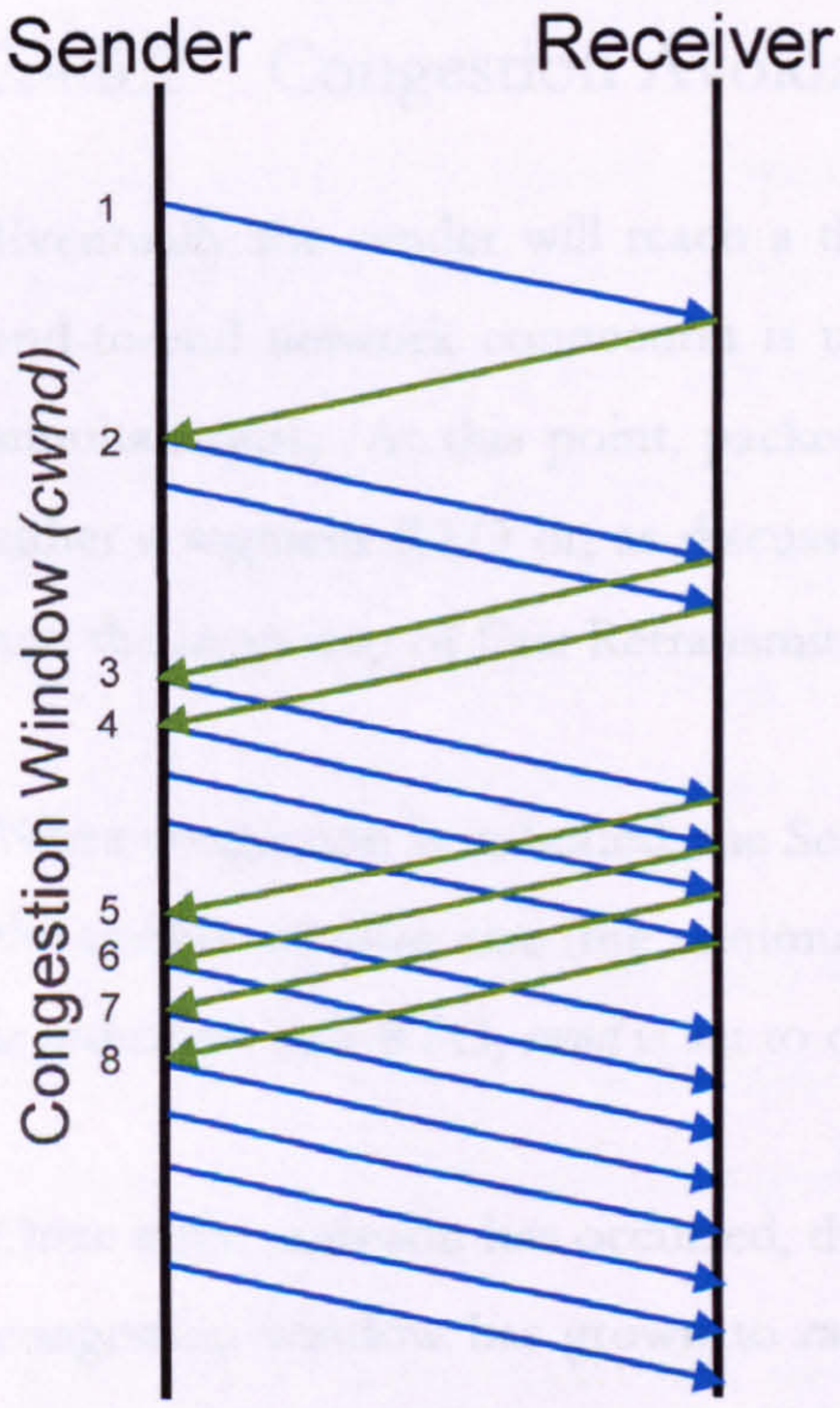

Figure 6 - Slow Start

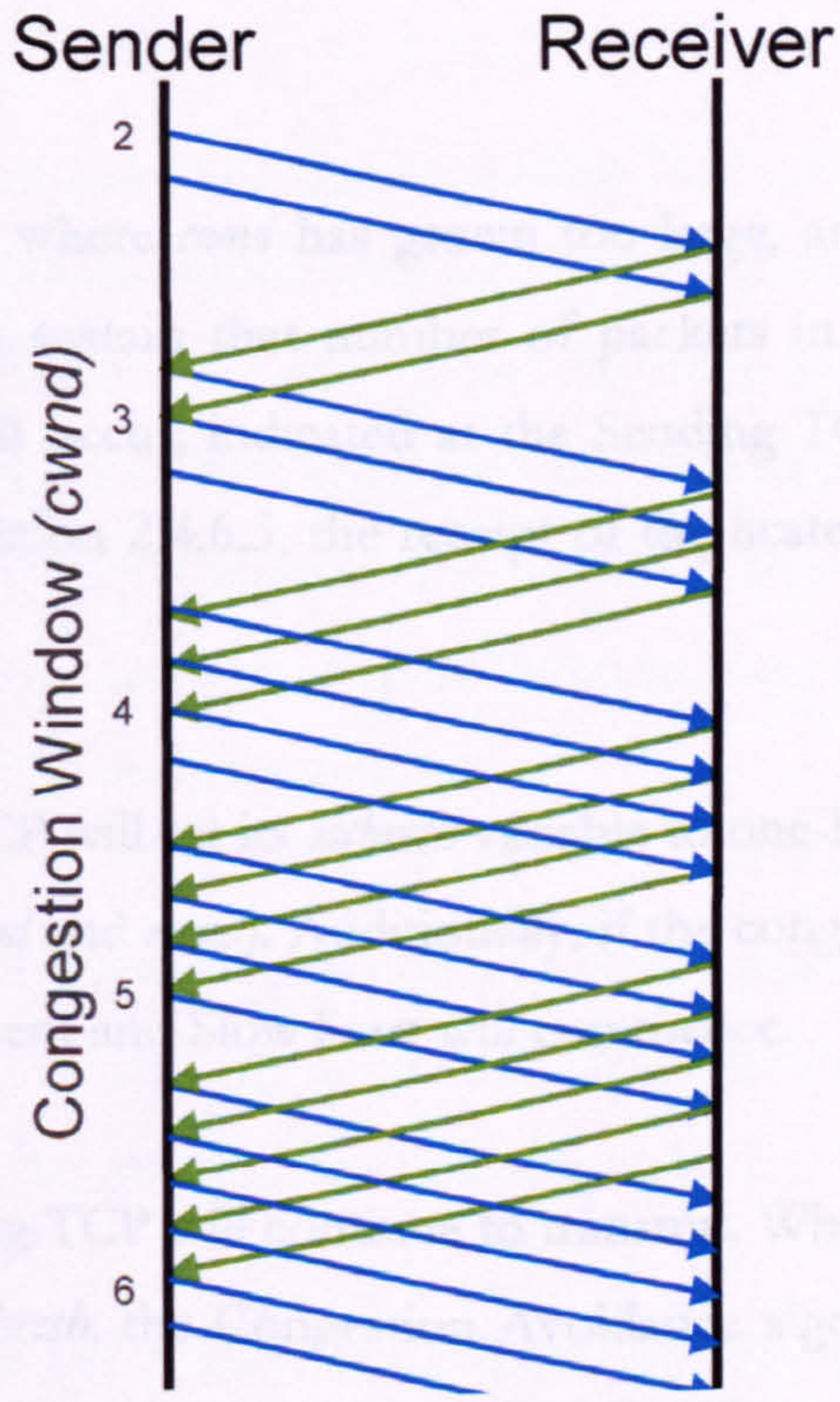

Figure 7 - Congestion Avoidance

2.4.6.1 Slow Start

During Slow Start, the Sending TCP will increment *cwnd* by one 'sender maximum segment size' (SMSS), each time an Ack is received as illustrated in Figure 6. This results in an exponential rise of segments injected into the network, and *cwnd* growing rapidly. Slow Start ends when *cwnd* is greater than *ssthresh*, or congestion is observed. RFC 2581 defines the slow start algorithm in terms of segments rather than bytes, but there are many TCP implementations which increase *cwnd* by exactly SMSS bytes whenever an Ack covering any new data, whatever size, is received,. This is known as 'Ack Division' [Sava99]. This, and other 'mis-behaving' TCP implementations [Sava99] are widespread in the Internet today.

2.4.6.2 Congestion Avoidance

Eventually the sender will reach a threshold where *cwnd* has grown too large, and the end-to-end network connection is unable to sustain that number of packets in flight simultaneously. At this point, packet loss will occur; indicated at the Sending TCP by either a segment RTO or, as discussed in Section 2.4.6.3, the receipt of duplicate Acks and the triggering of Fast Retransmit.

When congestion is indicated, the Sending TCP will set its *ssthresh* variable to one-half of the current window size (the minimum of *cwnd* and *rwnd*). Additionally, if the congestion is indicated by a RTO, *cwnd* is set to one segment and Slow Start will commence.

Once retransmission has occurred, the sending TCP will continue to transmit. When the congestion window has grown to *cwnd* > *ssthresh*, the Congestion Avoidance algorithm will regulate transmission, and will increment *cwnd* by approximately 1 SMSS per RTT, no matter how many Acks are received in that period. This results in a continued growth of *cwnd*, but in a linear fashion, as illustrated in Figure 7. The process of probing the path and the continuous adjustment of *ssthresh* is illustrated in Figure 8, where three losses each require a halving of the *ssthresh* value, and transmission restarting using the Slow Start algorithm.

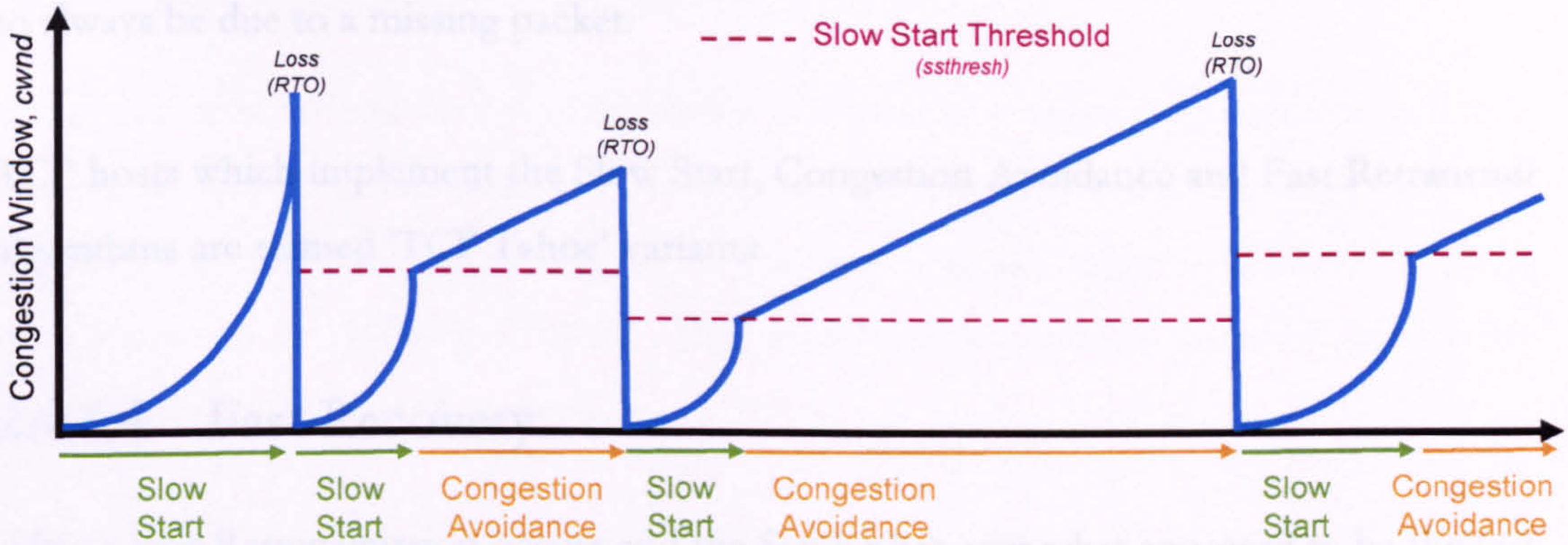

Figure 8 - Adjusting Slow Start Threshold

2.4.6.3 Fast Retransmit

The Fast Retransmit algorithm operates on the principle that Receiving TCP hosts are required [Brad89] to generate an immediate Ack each time they receive an out-of-order TCP segment. This Ack is termed a 'Duplicate Ack' of the last in-order segment successfully received, and signals to the Sending host that segments are still successfully arriving at the end host, but are arriving out of order, indicating that there is loss in the segment sequence at the receiving end. It is acknowledged [Allm07] that the reaction to the arrival of duplicate Acks varies widely in TCP implementations.

A TCP sender *should* [Allm07] use the Fast Retransmit algorithm to detect and repair loss, by using the arrival of 3 duplicate Acks, by default, as an indication that loss has occurred – that is, a total of four packets with the same acknowledgement field.

The arrival of 3 duplicate Acks should cause the *ssthresh* to be set to one-half of the current *cwnd*. Once the lost segment has been retransmitted, *cwnd* is inflated to *(ssthresh + 3xSMSS)* to account for the segments that have left the network.

The value of 3 was chosen since the Sending TCP does not know if the duplicate Acks are caused by packet loss or packet reordering over the path and, when the algorithm was developed in 1990[Jaco90], it was assumed that if 'the consecutive duplicates

threshold is set high enough, we can reasonably assume that duplicate Acks mean dropped packets', as the usual cause of out-of-order packets at the receiver was assumed to always be due to a missing packet.

TCP hosts which implement the Slow Start, Congestion Avoidance and Fast Retransmit algorithms are termed 'TCP Tahoe' variants.

2.4.6.4 Fast Recovery

After a Fast Retransmission occurs and the Sender has sent what appeared to be the lost segment, the Fast Recovery algorithm controls the Sending TCP until new data is successfully Acknowledged by performing Congestion Avoidance, rather than entering Slow Start. When the next Ack arrives acknowledging new data, *cwnd* is set to *ssthresh,* and transmission of new data continues, at half the rate at which packet loss occurred.

The Fast Recovery algorithm assumes that, although packet loss has occurred, the duplicate Acks indicate that other packets were successfully leaving the network and so there is no need to abruptly close *cwnd* and re-start the connection with Slow Start. This allows the 'Ack Clock' to be preserved [Jaco88], and the TCP algorithm to remain stable.

TCP hosts which implement the Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery algorithms are termed 'TCP Reno' variants.

2.4.6.5 Limited Retransmit

The fast retransmit and fast recovery algorithms can be implemented using the Limited Retransmit algorithm [Allm01], where on the first and second duplicate Acks, a TCP should send a segment of previously unsent data, as the duplicate Acks indicate that data is leaving the network.

2.4.7 Loss Recovery Mechanisms

Congestion control loss recovery mechanisms represent a popular research area, where investigations attempt to build on the mandatory Fast Retransmit and Fast Recovery algorithms to illustrate improvements in performance, although comparison of these techniques can be difficult [Floy07]. The predominant methods [Medi05] live in the current Internet are now discussed.

2.4.7.1 Partial Acknowledgements

The NewReno modification to TCP's Fast Recovery algorithm [Floy04] is motivated by the fact that simulations illustrate TCP Reno performs poorly [Fall96] when multiple packets are lost in a single packet 'flight'. Multiple packet losses will trigger a Fast Retransmit, but this will only result in a 'Partial Acknowledgements' from the Receiver – an Ack which does cover previously unacknowledged data, but not all the data outstanding when loss was detected, thus revealing that more than one loss has occurred.

During the Fast Retransmission phase, NewReno examines Acks received after the Fast Retransmit has sent the retransmitted packet. If the ack acknowledges all data, up to the highest sequence number transmitted by the Sender, then the Fast Retransmit is assumed to have successfully completed loss recovery. If the Ack does not acknowledge all data sent by the Sender, up to the highest sequence number transmitted, the Ack is a Partial-Ack. On identification of a Partial-Ack, the segment indicated by that Ack is retransmitted – without waiting for any more duplicates, and hopefully before an RTO occurs.

2.4.7.2 Selective Acknowledgements

With the limited information available from cumulative acknowledgements, a Sender can only learn about one lost packet per RTT. The TCP Selective Acknowledgements option [Math96] allows the Receiving TCP to inform the sender what segments have

arrived successfully, so that only the actually lost segments need to be retransmitted.

The Receiving TCP uses the TCP Options header to inform the Sender of the non-contiguous blocks of data that have been successfully received, but are queued until all sequence gaps have been filled. As retransmission from the Sender fills gaps in the holes, the Ack field is increased in the usual way, to Acknowledge successful receipt of the data.

RFC 2883 [Floy00] extends the use of SACK by specifying that when duplicated packets are received, the SACK options header can be used to report the Sequence number of the duplicated packet, in order to allow the TCP sender to infer the order of packets received, and infer when unnecessary retransmissions have been sent. This could be useful in environments where reordering, Ack loss, duplication or early retransmit timeouts occur frequently.

2.5 The Problem of Reordering

As illustrated by the development of the Fast Retransmit algorithm, later additions to TCP were made, based on the assumption that packet reordering on the Internet was a pathological behaviour – a phenomenon that was very unlikely to occur. Discussion of the degree of reordering measured in the Internet is presented in Chapter 3, and a measurement study of the true effects of packet reordering is presented in Chapter 4.

As discussed in Chapter 1, a significant contribution of Bennett's original paper in 1999, was to question the assumptions made by protocol designers that packets will traverse the Internet in-order. The following section presents the effects of reordering on TCP as hypothesised by Bennett, and the resulting effects that Bennett argued would be measurable on TCP performance [Benn99]. These assumptions have been assumed to be correct in many of the later studies of packet reordering [Bell02][Blan02].

Bennett hypothesised that, due to the asymmetric nature of the Internet, connections will frequently only experience reordering in one direction, and therefore there are three

types of packet reordering that can be considered; forward-path reordering or data reordering, reverse-path reordering or Ack reordering, and a combination of both forward and reverse path. Each type of reordering was argued to have very different effects on the overall TCP connection.

2.5.1 Forward path reordering

In forward path reordering, TCP data segments arrive out-of-sequence at the receiver as shown in Figure 9. Bennett hypothesised that this would result in the five effects of unnecessary retransmissions, difficulty growing *cwnd* and *ssthresh*, actual losses being obscured, poor RTT estimation, and reduced efficiency at the receiving TCP.

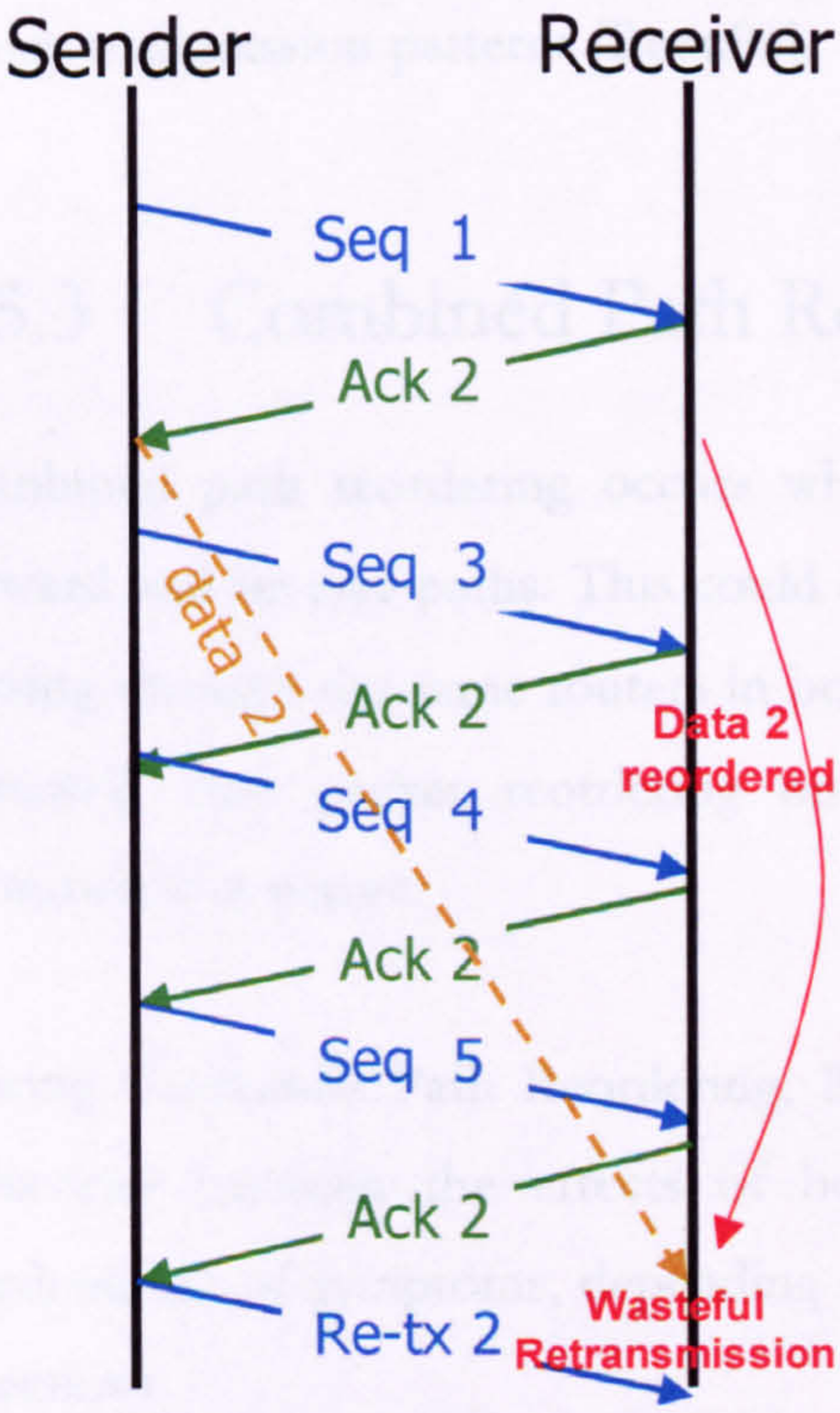

Figure 9 – Forward Path Reordering

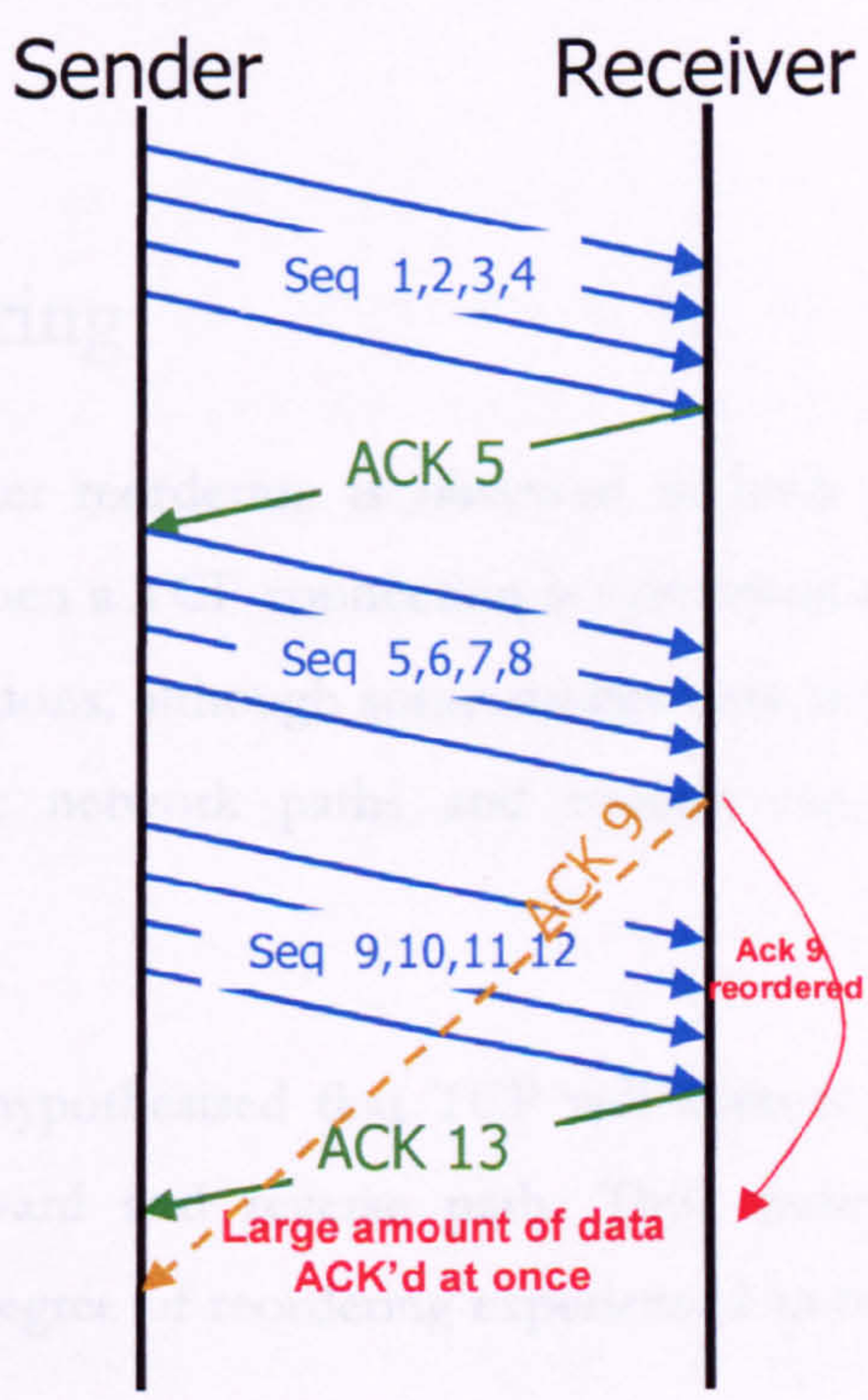

Figure 10 – Reverse Path Reordering

When data arrives out of order, the receiving TCP sends a duplicate acknowledgement of the last in-order byte received or, if SACK is implemented, the ack will acknowledge both the last in-order byte and the new out-of-order data.

Upon receiving an out-of-sequence packet, through either loss or reordering, the TCP

receiver's request for retransmission will require the sender to infer that the path is congested. This will result in an unnecessary re-transmission, but additionally will initiate unnecessary congestion avoidance, thereby further reducing the potential throughput of the link.

2.5.2 Reverse Path Reordering

In reverse path reordering, the acknowledgments travelling back to the sender are reordered, as shown in Figure 10. Data travelling in the forward path may be arriving in sequence, but the asymmetric nature of the Internet may cause the receiver's cumulative Acks to appear out of sequence. Bennett hypothesised that reverse reordering would cause significant problems with the self-clocking property of TCP, leading to highly bursty transmission patterns [Benn99].

2.5.3 Combined Path Reordering

Combined path reordering occurs when packet reordering is observed in both the forward and reverse paths. This could occur when a TCP connection is symmetric and passing through the same routers in both directions, although some studies have noted [Ghar04], that packet reordering on specific network paths and routers can be asymmetric in nature.

During Combined Path Reordering, Bennett hypothesised that TCP will alternate in behaviour between the effects of both forward and reverse path. This causes a combination of symptoms, depending on the degree of reordering experienced in both directions.

Packet Reordering therefore has an instinctively negative effect on the performance of TCP, and therefore the majority of the current metrics and measurement studies have focussed on characterising the movement of packets within a flow, rather than measuring the resulting performance of a connection during reordering. Few papers in the literature [Laor02] have actually measured the performance of TCP, or questioned

Bennett's arguments of TCP's behaviour when undergoing reordering. A measurement study of the true effects of reordering on TCP application performance is therefore carried out in Chapter 4.

2.6 Internet Measurement

Network measurement science is motivated by a number of factors in order to develop the tools and techniques to allow accurate characterisation and modelling of live network traffic currently transported in the Internet today. Measurement is important for provider operations, such as capacity planning, billing, and fulfilling local Lawful Intercept requirements, and also from a scientific perspective of evaluating current protocols and architectures, in order to develop new technologies and standards.

As the Internet evolves and new technologies such as voice, video and data 'Triple-Play'[Hens08] are deployed, there will be an increasing requirement to develop and implement Service Quality support in IP networks. To perform the traffic engineering required for this support, protocols such as RSVP [Brad97b] require more sophisticated characterisation of traffic flows, in order to allow network carriers to make provision for perhaps millions of concurrent connections, with diverse traffic characteristics and requirements[Ragh07], while at all times maintaining a guaranteed level of Quality of Service (QoS) [Info07].

2.6.1 Quality of Service

Quality of Service and Traffic Engineering [Awdu02] are becoming dominant in Internet access technologies[Info07], as business users move from traditional ATM leased line installations to Virtual Private Networks (VPN)[Info07]. This new technology provides many cost benefits to the user, as there is no longer the requirement to install expensive point-to-point leased lines but, as a result, the level of service and traffic characteristics are no longer guaranteed. There is, therefore, an increasing requirement for Service Level Agreements with accurate methods, to police and measure compliance of the end user service experience, and to ensure that the

network is providing the Quality of Service that is expected.

2.6.2 Service Level Agreements

A Service Level Agreement (SLA) is a contract which documents the level of service between a user and a network provider [Mart02]. The SLA describes the business terms of the agreement, the network provider and the users' responsibilities, and the penalties to be imposed should the agreement be broken. Examples of SLAs are widely available [Pipe07][Att07], as they are used by providers as key differentiators to attract new customers. The Service Level Specification (SLS) [Nich01] of the SLA is a set of parameters and their values, defining quantitative characteristics, and the bounds on these characteristics, that the provider is committing to deliver. A wide variety of characteristics may be included, such as delay, loss, Delay Variation and availability; these characteristics, and the methods used to measure these characteristics, are documented and agreed upon before a provider will make provision for each new customer.

2.7 Metrics and Measurements

The term 'Metric' is used to describe the computation of a measurement, and results in some quantifiable value that characterises a feature of the network.

The term 'Measurement' is used to describe the process by which the metric is obtained or retrieved and, therefore, could be one of several methods to perform a measurement that will result in the same metric. Performing a Measurement requires 'Instrumentation' of the network, at an 'Observation Point' (OP), for the given time-base of the metric.

The following section discusses the common measurements and metrics that are used in current Internet monitoring[Brow01]. A number of metrics are in common use, but their definitions are not necessarily standardised, resulting in problems when comparing one set of metrics with others.

2.7.1 Packet Latency

An accurate measure of latency is important as many applications, such as voice and video, do not perform well over network paths with high levels of delay. TCP will suffer degraded performance, as the round-trip-time estimator will measure a very high value, resulting in the poor performance of loss recovery and congestion control algorithms.

The 'One Way Delay' measurement as defined by the IETF IPPM Working Group, in Standards Track RFC 2679 [Alme99a], is the time measured between a host transmitting a packet, and the destination host receiving it. 'Network Latency' is the term used to describe the round-trip delay[Alme99b], and is a function of the time taken to travel along the physical links (transport time), the time to pass through routers (queuing and transmission time), and the time for the receiving host to process the packet and generate and Acknowledgement (server response time). When measuring network latency, the transmission time component of the measurement may be asymmetric, as forward and reverse measurements may travel over disjoint paths.

Latency can be measured using simple ICMP Echo Request 'ping' messages, or TCP resets, although using ICMP may not always result in accurate network measurements[Wenw07]. Measuring forward and reverse path delays separately, requires instrumentation at both ends of a connection, with appropriate hardware and software agents. RFC 2679 [Alme99a] describes a One-Way Delay metric, where both hosts have highly-synchronised clocks, and specially constructed packets with timestamps are transmitted to allow delay calculation in each direction.

2.7.2 Packet Loss

Network loss measurements are important because many applications do not perform well when end-to-end packet loss is high. Many services, such as voice over IP, are designed to tolerate some packet loss but, over a threshold value the service quality decays quickly. TCP actually requires packet loss in networks, in order to probe network capacity and adjust its transmission rate accordingly, and so an accurate measure of one way loss [Alme99c] and round trip loss are important network characteristics. Packet

loss may vary due to network load and time, and may exhibit bursty behaviour. Therefore, loss patterns [Koo02] and loss distribution are key parameters which determine the performance observed by the users for certain real-time applications.

A definition for Packet Loss is provided by the IETF IPPM Working Group, in Standards Track RFC 2680 [Alme99c], 'A One-Way Packet Loss Metric for IPPM'. Loss is defined simply as the number of packets transmitted from sender to receiver, which have been lost in transit. This can be expressed as a percentage over a set period of time. If a packet arrives, but any part of it is corrupted, RFC 2680 defines that packet to be counted as lost. If a packet arrives very late, a method is required to differentiate between packet loss and those very late packets; RFC 2680 suggests an upper bound of 255 seconds, as defined by the theoretical TTL lifetime of an IP packet. Packets which arrive later than 255 seconds are to be counted as lost. If a packet is duplicated along the path, so that multiple non-corrupt copies arrive at the destination, RFC 2680 defines the packet to be counted as received. If the packet is fragmented, and for whatever reason, it is not reassembled correctly at the destination, RFC 2680 defines that packet to be counted as lost.

The main difficulty with current loss measurement techniques, is that it is difficult to isolate an end-to-end loss measurement to a particular network node or path. Additionally, RFC 2680 comments that packet loss may occur asymmetrically across a network, and therefore loss measurements should be considered 'one-way'. Loss measurements would be more useful for providers if they isolated a particular malfunctioning node or path. Loss measurements are also based on active probing techniques, which may result in packet probes traversing different intermediate routes between source and destination, thus measuring different paths. Therefore, routing and switching anomalies must be considered when performing these measurements, especially if it is impossible to gain knowledge of the interior gateway routing protocols in use.

2.7.3 Packet Jitter and Delay Variation

The IETF IPPM Standards Track RFC 3393 [Demi02], 'IP Packet Delay Variation Metric for IP Performance Metrics', defines a metric for the variation in delay of packets across Internet paths. This metric is based on the variance of the one-way-delay, as defined by RFC 2680, of two or more selected packets.

RFC 3393 establishes that terminology in this area of measurement lacks standardisation, and that the variation in packet delay is sometimes called 'Jitter'. 'Jitter' commonly has two meanings; the first being the variation of a signal with respect to a clock signal, where the arrival of a specified signal is expected to coincide with the arrival of that clock signal. This definition of 'Jitter' is similar to a metric called 'Wander' [Demi02] and is specific to networks such as ATM. The second meaning of 'Jitter', as defined in RFC 3393, is related to the variation of a metric (e.g. delay) with respect to some reference metric (e.g. the average delay). RFC 3393 recommends discontinuation of the word 'Jitter', and use of the more specific term 'IP Packet Delay Variation.'

'Delay Variation' is defined [Pore06] as the absolute value of the difference between the arrival delay variation of two consecutive packets belonging to the same stream, and is therefore indicated by packets exhibiting a differential delay, positive or negative, compared to the other packets in the stream. Delay Variation can be caused by multi-path routing, route fluttering, or packets of the same stream traversing different queues inside a router. Delay Variation is an important metric for determining queuing and buffering capacities at mid-points and end-points in a network, as applications such as video require a constant flow of packets. Therefore Delay Variation must be smoothed by appropriately sized buffers.

2.7.4 Packet Throughput

Throughput is defined [Brad91] as the maximum rate, measured in bits or packets per second, at which none of the offered packets are dropped by a network device.

Throughput therefore describes the number or rate of delivered packets to a network device or end host. Throughput as a metric has many applications in terms of capacity planning and traffic engineering, but is also extended to discuss 'Goodput'. Goodput is the number or rate of *useful* packets delivered, and therefore is a function of the throughput with respect to loss and retransmissions. Goodput is discussed further in Chapter 5.

2.7.5 Packet Ordering

In-sequence delivery is a good indicator of the health of a connection, as it indicates that there are no large variations in transmission time or Delay Variation, and that the receiving host is receiving data in the order by which it was intended. In protocols such as TCP, extremely late packets may result in the Receiver assuming that a loss has occurred, and that the Sender has chosen to retransmit that packet due to RTO. These extremely late packets may also result in the Receiver signalling to the Sender for Fast Retransmission, perhaps unnecessarily resulting in packet retransmission, and perhaps resulting in unnecessary Sender congestion avoidance. In Negative-Acknowledgement based protocols, such as the RFC 3940 NACK-Oriented Reliable Multicast Protocol [Adam04], extremely late packets would result in the Receiver signalling to the Sender for unnecessary retransmission.

Packet Ordering will also have an overhead, associated with the re-sequencing packets at the Receiver, before presentation of these packets to higher network layers.

A review of both Metrics and Measurement Techniques for IP packet ordering is the subject of Chapter 3.

2.8 Measurement Bases

Measurements can be classified on the basis of where and at which level of traffic aggregation the traffic data is gathered, and a number of Measurement Bases are defined[Lai03]. The observation point may be a probe on a line card of a router, or a software probe in the IP stack of a particular host. The exact location of the observation point may also have an effect on the observation measurement time. The concept of 'wire time' and 'host time' are therefore important, and are affected by the specific instance of probe. Wire time is defined [Alme99a] as the moment when a test packet leaves the network interface of the source, and the moment when it arrives completely at the destination. Host time is the timestamp taken from the sending/receiving hosts kernel, when the probe requests a timestamp.

2.8.1 Flow-based Measurements

Flow-based measurements are primarily used on interfaces at routers, and are used to collect detailed information about a particular, or group of, IP flows. A flow is identified by the source and destination IP addresses, port numbers, and protocol numbers, and once a flow has been identified, a variety of the measurements described in Section 2.7 can be performed. On core network routers, measurement of every flow through a router interface is extremely challenging due to the number of source/destination pairs which could be identified, and the large number of measurement records that are created and then associated with each flow. This can result in large amounts of measurement data that may be difficult to process in real-time, without adversely affecting the network node's performance.

2.8.2 Interface, Link and Node-based Measurements

The purpose of these measurements is to characterise the behaviour of the particular network element or interface, and is typically performed by a single-point passive

technique as discussed in Section 2.9.1. This type of measurement may be used to describe the traffic aggregation across a particular router interface, such as the maximum packet throughput that a card will sustain, without giving attention to particular flows or their source and destinations.

The Simple Network Management Protocol (SNMP) [Case90] is an example of an interface, link or node based measurement protocol, which uses passive monitoring to collect data, typically in the form of packet counters, byte counters, packet discards and errors. These metrics are organised in hierarchies and stored in Management Information Bases (MIB)s, and are reported by the SNMP agent to the Network Management System as required. These measurements would typically not be able to reconstruct a flow of packets between sender and receiver and, as discussed with flow based measurements, may require monitoring of thousands of concurrent flows and the generation of large amounts of measurement data, which would require significant post-processing to generate practical results.

2.8.3 Node-pair-based

Node-pair measurements concern the measurement made between two predefined network elements. This is usually the case for active measurements, such as Ping or Traceroute as discussed in Section 2.9.2. Alternatively, it could be derived from the records generated from a flow-based measurement through post-processing of source/destination pairs. Due to multi-path routing and asymmetric paths, node-pair measurements can be difficult to perform in IP networks.

2.8.4 Path-based

Path-based measurements are performed over a path between a number of network nodes. This may be a strictly defined MPLS path, or simply a multi-hop route across several nodes. These measurements describe the characteristics of the aggregate traffic of the path and could be retrieved, based on one observation point to passively monitor throughput, or based on multiple observation points to actively monitor the delay.

2.8.5 Local and End-to-End Measurements

The observation point required to perform a measurement must be defined when reporting the results of any Internet measurement, and can be described as being either a Local Measurement or an End-to-End Measurements.

- **Local Measurements** describe measurements which can be made from a single observation point, such as interface-based and node-based measurements. Path and flow based measurements can also be made locally if the correct observation point is chosen, although as discussed in previous sections, these measurements are more challenging to implement.

- **End-to-End Measurements** are those which require more than one observation point. This facilitates path-based and node-pair based measurements, over multiple hops across an end-to-end Internet path.

In addition, Internet measurements may be described as being absolute or derived:-

- **Absolute Measurements** provide an accurate representation of a metric without calculation of any statistics. An example of an Absolute Measurement is a count of the number of packets discarded by a router.

- **Derived Measurements** are computed from simple measurements, such as the delay of a single packet in a stream and, from a series of several measurements, derive a subjective measurement result – such as the Mean Opinion Score of a video playback.

2.9 Measurement Methodologies

There are two primary techniques when performing both local and end-to-end measurements; Passive measurement methods and Active measurement methods. These passive or active methods can both be applied across local or end-to-end points, to compute a number of metrics.

2.9.1 Passive Measurements

Passive measurements are performed on live user traffic by monitoring the normal operational packet load from an observation point. The observation point applies a mask on the packets travelling through the instrumented device, and copies these packets of interest into the measurement probe, to perform metric calculation and possible storage.

Passive measurements may be applied across a single or multiple observation points. A single-point measurement allows monitoring of traffic load and protocol statistics or, if the packet flows contain timestamps such as RTP, then metrics such as delay, Delay Variation and packet loss can be calculated. With multiple observation points, a passive measurement approach can perform delay measurements without timestamps, provided that both observation points are synchronised, as discussed in Section 2.10.

Passive measurements do not add further traffic to the network but, in general, require more computation than active measurements and can result in very large datasets requiring to be processed in order to obtain relatively simple metrics.

2.9.1.1 Passive Measurement Examples

Cisco IOS NetFlow [Clai04] was a proprietary passive measurement technique to perform flow measurement on Cisco routers. For each interface on the router, NetFlow maintains tables of 'Flow Records' of each connection passing through the interface,

based on source and destination IP addresses and ports. At specified intervals, NetFlow data records would be sent by UDP from the router, to a NMS for processing.

NetFlow Version 9 [Lein04] is the basis for the emerging IETF standard 'Internet Protocol Flow Information eXport' (IPFIX), to create a common format for IP flow information, to be used in accounting, billing and network management systems.

There are several methods and tools to collect and analyse NetFlow data, such as Flow-tools[Full07]. Flow-tools is an open-source collection of tools to process and generate reports from NetFlow-enabled devices. Other router manufacturers provide similar functionality to Cisco NetFlow. Juniper Networks provide Jflow statistics collection[Juni07], and Huawei Technologies provide a similar feature called NetStream[Huaw07].

2.9.2 Active Measurements

Active measurements operate by injecting controlled test packets through the network, in order to observe how these packets, with their characteristics known a-priori, are affected by the network. Active measurements therefore require two elements in the network; the sending host must generate the test packets, and the receiving host must either perform the measurement, or return specific information to the sending host.

Active test traffic may be simple ICMP Echo Request probes, such as those sent by Ping or Traceroute, or may be synthetic packets, typically carrying timestamp and sequence numbers to allow delay and Delay Variation measurements to be calculated. Active measurements are intrusive on the network and will add to the overall load that the network must process, but are commonly used as the method for monitoring SLAs [Bt07] and for basic site connectivity testing. There is also the possibility that the synthetic traffic may be treated differently from other live traffic on the network[Wenw07], or that the traffic may be processed by a router's 'slow path' control-plane, thus generating non-representative results.

2.9.2.1 Active Measurement Examples

'Ping', the Packet Internet Groper [Mill83] was the first single-point active measurement tool, which transmits ICMP echo request packets to a specified host and measures the round-trip time of the response. The advantage of Ping is that ICMP packets are handled inherently by a hosts' IP kernel, and so no additional endpoint equipment is required. ICMP is used less often now for internet measurement, for the reasons discussed above, and because of the increased likelihood of firewalls blocking these messages.

The IPPM Periodic Stream Measurement [Rais02] transmits equally sized packets at regular intervals to simulate a constant bit-rate multimedia stream, and to quantify the delay and Delay Variation experienced. This method provides a way to perform measurements irrespective of the QoS mechanisms employed by the IP network.

'Cisco IOS IP SLAs' is a tool which runs as embedded software within Cisco routers and network entities, to provide active monitoring of delay, loss and Delay Variation between the router device and other devices. Service Assurance Agents are installed on each Cisco device, which is polled by test traffic (simulating a particular IP network traffic type), and the response measured. As the name suggests, this is a key tool for network operators to ensure SLA compliance.

2.10 Limitations of Current Techniques

Many of the limitations of current measurement techniques have been highlighted throughout the previous discussion. The greatest difficulty with passive measurements is the positioning of the observation point, to ensure that all of the traffic of interest is measured. Additionally, passive measurements are likely to create large amounts of logging data, which require storage, transmission from the observation point to a NMS, and significant computation to produce the required results. Active measurement techniques require instrumentation of more than one host, and ensuring that the

receiving host is capable of processing the synthetic traffic as required. Active measurements consume valuable network resources as a part of their procedure and, if ICMP is used, they may also be handled differently from normal traffic – thus generating spurious results.

Positioning of network probes is another difficulty with current techniques. Should the technique require specialist logging or processing capabilities, it may not be possible to place this on a particular piece of network hardware. The use of SPAN ports on switches [Cisc07] can only monitor low numbers of links and low utilisation levels. The use of fibre taps can have a detrimental effect on the distance the fibre would normally be able to span.

Time Synchronisation is critical in many of these metrics, as many techniques, both active and passive, require the ability to timestamp measurements at geographically separate points in the network. It is widely accepted [Paxs98] that Network Time Protocol (NTP) is not sufficiently accurate for network measurements, as the clock accuracy is affected by the delays of the paths used by the NTP peers. Currently, the most common method of time synchronisation for network measurement is Global Positioning System (GPS). GPS systems are in wide use in telecoms applications to provide a reference site clock to a local site, that is then shared within that site using NTP.

IEEE-1588 [Ieee07] is an emerging standard that defines a protocol which can synchronise heterogeneous systems, across a LAN, to nanosecond levels of resolution and accuracy on a Gigabit Ethernet. This protocol performs measurement of time offsets between devices on a LAN, and then the synchronisation of those devices to a master device – which would normally be connected to a GPS receiver. Standardisation work has now been completed on the protocol, but at the present time, there are very few devices available with 1588 capability.

2.11 Summary

This chapter has provided an introduction to the Internet Protocol Suite, and specifically the operation of the Transmission Control Protocol. The various loss recovery mechanisms and congestion control algorithms have been presented, with the motivation and operational assumptions of these improvements discussed.

An overview of network measurement science has discussed the common metrics and measurements that are used on the Internet today, and the two main methodologies of active and passive techniques have been shown. Examples of these techniques, and their limitations have been presented.

The effects of packet reordering on TCP flows has been discussed, and it has been argued that packet reordering will have a significantly detrimental effect on overall performance. In-sequence packet delivery is a good indicator of the health of a connection, as it indicates that there are no large variations in transmission time or Delay Variation, and that the receiving host is receiving data in the order it was intended. Chapter 3 will continue this theme by presenting the current state of the art in packet reordering measurement research, by discussing the various proposed measurement techniques, and the results of recent measurements performed on live Internet networks.

Chapter 3

Measuring Packet Reordering

3.1 Introduction

It has recently become clear to the networking community that the traditional metrics and measurements used to characterise an IP flow, namely latency, loss, Delay Variation and throughput, do not convey sufficient information to fully describe flow performance across an entire end-to-end path. Recent work [Benn99] has indicated that TCP's design assumption, that packet reordering occurs infrequently, may be invalid and

may be subject to increase further as the Internet grows. This may be further exacerbated by the increase of parallel links and the predominance of other new technologies, such as IP multihoming, and the increasing use of wireless technology.

Since Paxson's first work in 1995[Paxs96] to characterise the degree and severity of packet reordering in the Internet, there have been several attempts to develop a metric and measurement methodology to describe IP packet reordering. These studies are so diverse in their techniques and assumptions, that it is very difficult to compare results across the literature. The lack of a standard experimental measurement methodology, and the lack of a standard reordering metric, has been argued [Benn99] to be a significantly limiting factor in understanding the effect, impact and prevalence of packet reordering in today's Internet.

It is also important to note, that TCP itself is not the only protocol within the IP suite which is susceptible to packet reordering [Ghar04]. Any protocol which mimics 'TCP-Friendly' behaviour, where the packet transmission rate is divided over the square root of the packet loss rate, or, has tight constraints on packet arrival times, may be susceptible to the effects of reordering, and thus this lack of understanding could affect a large proportion of the traffic on the Internet.

This chapter presents the prior art of the area, by presenting a taxonomy of metrics and measurement methodologies which have been developed to characterise packet reordering in the Internet, and the results obtained by using these methods.

Firstly, a survey of active and passive reordering measurement techniques are discussed and evaluated, which have been proposed to measure the degree of packet reordering occurring on an end-to-end path.

Secondly, a number of packet reordering metrics are discussed, which have been proposed in order to numerically describe the amount of packet reordering that is occurring on a flow.

Thirdly, a comparison of these techniques is presented, and a comparison of the

measurement results obtained using these techniques. This Chapter concludes with discussion of the difficulties of measuring both the impact and degree of packet reordering, and the drivers that should motivate further research in this area.

3.2 Active Packet Reordering Measurements

As previously discussed in Chapter 2, active measurement techniques are commonly used for SLA compliance monitoring [Somm07], and operate by injecting synthetic traffic into the network in order to emulate the performance characteristics endured by a real traffic flow. Passive measurement techniques are unable to characterise the end-to-end performance of a packet from an arbitrary single point, and therefore active packet probes are required. Active probes are, of course, not without their own problems, and it has been argued [Scho04] that results obtained by active probes are low in accuracy and high in packet probing overhead, and often do not correlate [Barf04] with router-based passive measurements.

Nevertheless, due to the complexity associated with designing a passive measurement technique for packet reordering, as discussed in Section 3.3, the majority of packet reordering measurement techniques are based on active methods. This section reviews these techniques, and summaries the results obtained when tests have been performed on live Internet networks.

3.2.1 Limitations of Active Reordering Measurements

Designing active packet probes which can be injected into a network and generate meaningful performance metrics, which are representative of real network behaviour, is a challenging process, and has been shown to produce results [Barf04], which do not correlate with other passive measurements made by protocols such as SNMP. In many circumstances, intrusive measurements using active probe packets is the only option available [Scho04]. There are many practical issues when conducting large scale Internet measurements that must be addressed [Paxs04], such as including relevant meta-data with results, dealing with large amounts of data, ensuring results are reproducible and

accurate, and making datasets publicly available.

Additionally, there are many limitations imposed by the operation of Internet nodes which may affect measurement. Not all implementations interpret standards consistently [Fang03], and this must be considered when designing measurement experiments which operate on live networks. Individual link measurements may not correlate with a user's end-to-end path experience, and it can be difficult to know what to measure. SNMP can provide a great deal of data about the status of each management network element, but this can be difficult to correlate with overall user experience[Hust03].

Round-trip probes such as Ping and Traceroute are useful active measurements to measure a total network path, but these cannot measure the characteristics of a single component in that path. One-way measurement packet probing techniques [Luck01] are being developed to perform these measurements, but require strict clock synchronisation between sender and receiver in order to calculate accurate results.

The Internet does not lend itself well to being measured. 'Middleboxes' [Allm03b] are intermediate network devices which do not follow the standard partitioning of functionality as defined by the OSI model. Therefore 'multilayer switches', 'layer 4 routers', 'layer 4-7 switches' or 'content switches' are all devices which provide the basic functionality of packet switching but, in addition, may inspect higher OSI layers to provide additional functionality, such as firewalling, intrusion detection services, web-server load balancing or network address translation. These additional network elements, which may appear transparent to a normal data connection, can have adverse affects on active packet probes, as many violate traditional networking assumptions that packets flow from source to destination essentially unchanged[Medi05].

Many publications of active Internet measurements rely on unusual or uncommon parts of specifications, in order to construct measurement packets. Approximately 40% of hosts do not operate SACK correctly, Explicit Congestion Notification has been measured in only 2.1% of connections, and less than 36% of end hosts support IP Options such as Timestamps[Medi05]. It is common for Middleboxes to simply drop packets with unknown IP options, and indeed, some Middleboxes deploy

countermeasures termed 'fingerprint scrubbers' [Smar00][Medi05] to manipulate TCP options, thus preventing identification of TCP end hosts. Any active packet probe risks being identified as a Denial of Service attack, and should at least expect additional latency due to further analysis by intermediate IDS entities.

With these limitations considered, a review of active packet reordering measurements is now presented.

3.2.2 Paxson

Between December 1994 and December 1995, Vern Paxson performed the first large scale studies of Internet packet behaviour from an end-to-end perspective[Paxs97], with the aim of investigating how routing dynamics translate into perceived quality by the end user[Paxs97a]. Paxson investigated 'pathological conditions', such as routing behaviour and routing asymmetry, and discovered that both packet reordering and route asymmetry were much more common than was previously assumed.

Paxson's work was important due to the sheer scale of the measurement methodology involved. 37 end hosts around the world [Paxs96] were instrumented as active probes, and 40,000 end-to-end ICMP 'Traceroutes' were performed and post-processed for analysis. The results from these Traceroutes questioned many of the commonly held IP networking assumptions, such as in-order packet delivery, FIFO queueing, and path symmetries.

Paxson extended this work [Paxs97b] [Paxs99] by performing 20,000 100-KByte TCP bulk transfers between 35 probe sites and, using Tcpdump [Tcpd08] to record the output at both ends, performed the largest one-way measurement [Hust03] to date, using TCP rather than UDP or ICMP. In subsequent offline post-processing, each tcpdump file was traced using 'Tcpanaly' [Paxs97c], a tool to parse Sequence Numbers, follow the congestion control specifics of each TCP implementation, and to generate statistics.

Paxson developed his own metric for packet reordering. As each packet arrives, it is

checked against the last 'non-reordered packet'. If the sequence number is greater than the last non-reordered packet, then that packet is marked as being in-order, and becomes the new non-reordered packet. Therefore, in a sequence of arriving packets 1, 6, 2, 3, 4, 5, packets 1 and 6 are marked 'in-order', while the other 4 packets are marked 'reordered'. This simple metric highlights packets which arrive 'late', rather than marking 'early' packets as those which have undergone reordering.

The results from these measurements concluded that packet reordering is highly prevalent in the Internet. During the two measurement periods, over 36% and over 12% of the TCP sessions included at least one packet which was delivered out of order, with the fraction of packets that were reordered measured as 2% and 0.3% in the forward direction, and 0.6% and 0.1% in the reverse. Paxson argued that the larger number of data packets being reordered in the forward direction results from the cumulative-ack function, resulting in data packets being sent closer together and, thus, requiring a smaller difference in transit time to cause reordering. Paxson questioned further assumptions on network behaviour by observing that Network Replication of packets was extremely rare, and that packet corruption was also negligible, measured in 0.02% of data packets, and 1 in 1.6 million ack packets.

Packet reordering was also found to be both highly site-dependent and asymmetric; one particular site exhibited 15% packet reordering, which was significantly higher than the average 2%. Of this 15% reordering, only 1.5% of data packets sent forward to that site were reordered; the majority of reordering was measured on data packets travelling away from the site.

The major shortcoming of this, and of other active techniques, is the requirement to run code on end hosts; therefore this measurement methodology cannot scale to multitudes of arbitrary hosts. Furthermore, the use of 100 kbyte TCP transfers may not be sufficient data to allow the sender's congestion window to fully open, thus making comparison with other bulk transfer measurements difficult [Laor02] [Feng07].

The ability to instrument the end-points across so many diverse hosts, to measure live TCP traffic rather than packet probes, to perform one-way unidirectional measurements

between each source-destination pair, and to perform offline post-processing of the results are compelling advocates for the use of Paxson's measurement methodology. The scale of these measurements, performed nearly 10 years ago, generated results which questioned many protocol design assumptions on packet reordering and duplication, and stimulated further attempts to measure and characterise these features. Paxson himself notes that these measurements may not be representative numbers for the whole Internet, but surmises that specific Internet paths may be subject to a high incidence of reordering, and that this incidence is site dependent and correlated with route fluttering.

3.2.3 Bennett

In 1999, Bennett [Benn99] questioned Paxson's conclusion that packet reordering was a 'pathological behaviour', mainly caused by route fluttering, router update events and incorrect or malfunctioning networking components. The fact that packet reordering did occur on the Internet was not a revelation from Paxson's work, as this had been a design assumption of the Fast-Retransmit algorithm, but Bennett argued that a recent significant increase in 'local parallelism' within Internet components, was causing packet reordering under normal operation. Bennett asserted that packet reordering was no longer a pathological behaviour, and that the incidence of packet reordering was substantially higher than had been previously reported.

Bennett performed a number of active measurements at MAE-East[Mae08]; the largest Internet Exchange Point in the world in terms of bits per second of traffic, equipped with a core DEC Gigaswitch multiport FDDI crossbar switch. Between December 1997 and January 1998, Bennett chose 140 hosts that were topologically close to the Exchange, and sent a burst of 5 ICMP Ping packets, to prime for route cache misses, followed by a back-to-back burst of 50 ICMP Ping packets each of 56 bytes. The number of hosts to successfully receive the first 10 packets was recorded, and then the number to receive these first 10 packets in order was also recorded. This measurement was then repeated for the first 20 packets.

Results from these initial tests indicated that the probability of a session experiencing

reordering was over 90%. To better understand the characteristics of reordering, a second site test was performed on a specific host which had exhibited high degrees of reordering, in order to investigate the effect of traffic load and reordering. This host was sent a 100-packet burst of 512-byte packets every minute for four days, and the degree of reordering was measured by calculating the average number of SACK blocks required to cover the out-of-order Ping replies received if the session had been a TCP connection. From publicly-available traffic statistics of MAE-East, Bennett was able to plot his SACK-block metric against the load of the Gigaswitch, and conclude that packet reordering, specifically at MAE-East, was a function of core network load.

The concept of 'local parallelism' introduced by Bennett, was exemplified by discussion of the DEC Gigaswitch and its feature of 'Hunt Groups', whereby multiple FDDI ports operate as a single virtual link, thus allowing that switch port to "load stripe" across dual parallel physical links for increased bandwidth capacity. Bennett argued that prior to mid-1997, the Gigaswitch operated at loads where the "Hunt Groups" features would not cause packet reordering, but, due to the explosive growth in traffic by 1998, packet reordering was now a common feature. Bennett notes, though, that local parallelism is not a problem with just this particular switch. In order to achieve the multi-gigabit performance which users demand, the use of load balancing, link striping, and local parallelism within nodes will also have to increase. Packet reordering is therefore a complex phenomenon, and Bennett concludes that it is a function of the existence of parallel links between nodes on a path, of the exact configuration of the hardware and software in nodes on the path, and of the traffic load of the nodes on the path.

Bennett's packet reordering technique is important as the use of ICMP Pings allows measurement against arbitrary hosts, thus negating the need for instrumentation of the endpoints. There are, though, some limitations to this technique. It is known that network operators often filter or rate-limit ICMP traffic [Bell02] [Wenw07] and, if the measurement end-point is a switch or router, the packet will be processed on the 'slow path' by the router's GPU. Secondly, it is not possible to infer if the ICMP packets were reordered on either the forward or reverse path. The use of the 'SACK blocks' metric is dependent on the end-host supporting this TCP extension. It has also been argued [Jais07] that the use of back to back ICMP packets may exacerbate the amount of

reordering observed, as the inter-packet gaps are small, and unlike TCP, the send rates is not reduced upon detecting congestion.

3.2.4 Loguinov

Loguinov's active measurement study of video traffic [Logu02] in 2002, across 16 thousand ten-minute MPEG-4 sessions over seven months, is a significant large scale real-time measurement study of loss, delay and packet reordering, and provides an insight into the behaviour of low-bitrate streaming sessions. Based on connections to commercial dial-up ISPs, the experiments consisted of streaming video sequences to unicast home users, using UDP as the transport mechanism, and a simple NACK-based retransmission scheme to recover lost packets before their decoding deadlines.

Two video streams were encoded at 14 kb/s and 25 kb/s, and split into 576 byte IP packets of roughly 5000 each. In the first set, three clients performed 16783 connection attempts by long distance PSTN modem calls to ISPs and completed 8429 successful streaming sessions. For the second set, 17465 calls were placed, resulting in 8423 successful streaming sessions. Results unexpectedly indicated that, despite the very low bitrates of the streams downloaded, certain paths experienced consistent reordering although at a very small degree.

The percentage of reordered packets was calculated relative to the total number of missing packets. The average reordering rate was measured to be 6.5% of the number of missing packets, or 0.04% of the number of sent packets, which although only 10% of that measured by Paxson, is explained by the authors due to the lower bit rates. Of the total number of transfers, 9.5% experienced at least one reordering, although specific paths exhibited up to 35% of connections (and 0.2% of sent packets) experienced reordering.

Loguinov defined two metrics to describe reordering. The packet reordering delay, D_n is the delay from the time when a reordered packet was declared as missing to the time when the reordered packet arrived at the client. Packet reordering distance d_n is the number of packets (including the very first out-of-sequence packet, but not the

reordered packet itself), received by the client during reordering delay D_r

Across the two sets of experiments, the largest reordering distance d_r was measured as 10 packets, and the largest reordering delay D_r was 20 seconds, although this was seen on only one packet. 90% of d_r measurements were below 150ms, 97% below 300ms, and 99% below 500ms.

The Reordering Distance was used in order to measure the effectiveness of TCP's Fast Retransmit mechanism. By plotting the pdf of D_r, 91.1% of reordered packets were seen to have moved by less than 3 packets, and 95.7% of packets were reordered less than 4 packets.

3.2.5 Bellardo

In 2002, Bellardo [Bell02] developed a suite of active measurement tools named 'Sting' [Bell03] to measure one-way end-to-end packet reordering rates; aiming to improve on Bennett's 'SACK block' measurement and Paxson's ICMP measurements by negating the need to instrument end-point nodes. Bellardo uses a 'packet-pair approach' [Hust03], where a packet 'train' is sent to an arbitrary TCP endpoint, and the response to this packet-pair allows a one-way packet reordering measurement to be performed.

Figure 11 illustrates the 'Single Connection Test', whereupon after the normal 3-way handshake has been completed, the first data packet to be sent, Seq 2, is exactly one segment size higher in sequence number than that expected by the receiver. The end host will acknowledge this packet by sending Ack 1, indicating that the first data packet appears to have been lost. The measurement probe responds by sending the first and third data packets, Seq 1 and Seq 3. By priming a 'hole' in the receiver's window of acknowledged packets, and then by measuring the response to this packet-pair, it is possible to differentiate from the resulting Acks if packet reordering is occurring on the forward path, reverse path or in both directions.

The Single Connection Test provides a simple method of identifying reordering, but will fail if the end node implements the delayed acknowledgement algorithm, whereby a

receiving host will delay acknowledgements for a period before sending a cumulative acknowledgement covering several segments of data[Clar82]. On many TCP implementations, the arrival of Seq 1 and Seq 3 in close succession will result in a single Ack 4 being sent.

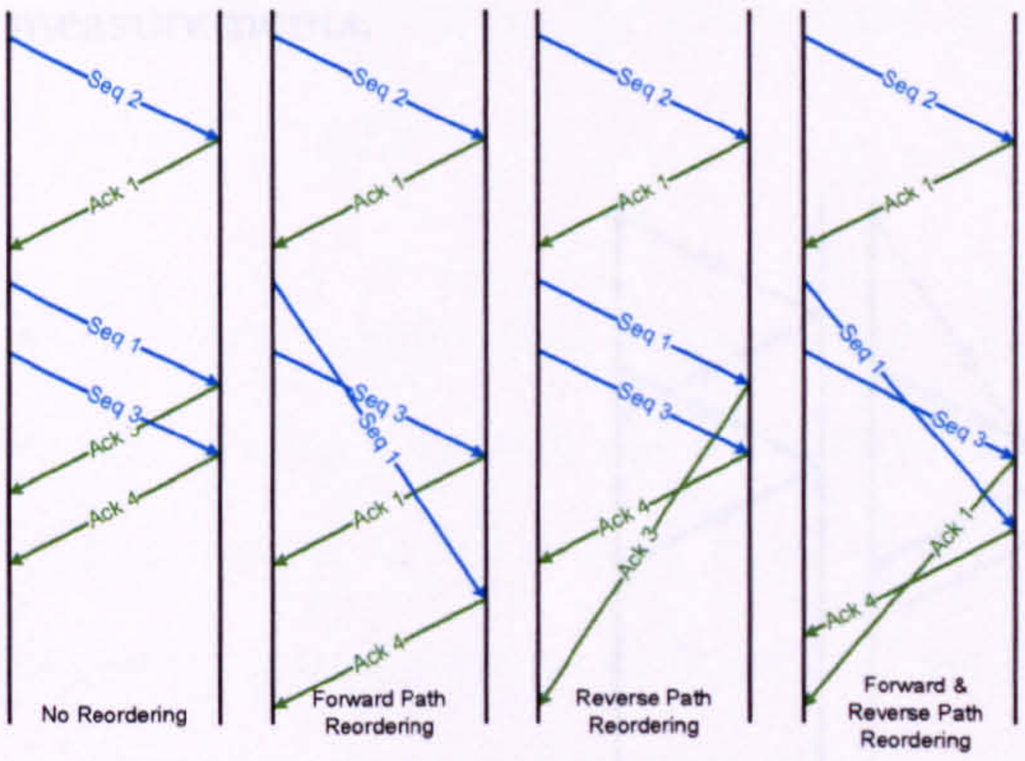

Figure 11 - Bellardo Single Connection Test

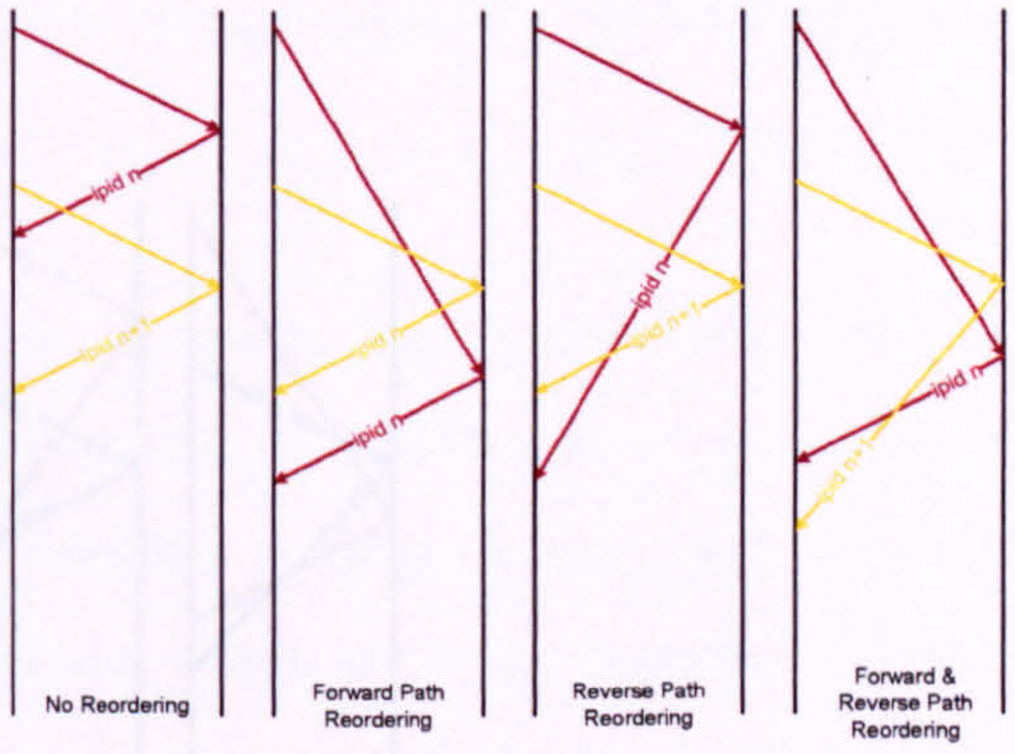

Figure 12 - Bellardo Dual Connection Test

To address this limitation, Bellardo's 'Dual Connection Test' establishes two simultaneous TCP connections from the measurement probe to the end host, as illustrated in Figure 12. Each TCP connection primes the end point in a fashion similar to the Single Connection Test, by sending a Sequence Number exactly one segment size greater than that expected. The Dual Connection Test assumes that the IPID field will increase monotonically across TCP connections to the same end host destination, and therefore the IPID can be examined in the returning Acks, to indicate the presence of reordering in the forward or reverse paths. The authors acknowledge that this assumption is not without its own problems, and that this test will fail completely should the end host be hidden behind a Middlebox, such as a transparent load balancer.

The TCP SYN test, illustrated in Figure 13, assumes that any Middlebox will perform load balancing by hashing the four-tuple addresses and ports viewed in the IP header, and that by sending a packet pair of identical Syns that differ only slightly in the starting sequence number, the end host will reply with a Syn-Ack to the first Syn, and a Reset (Rst) to the second Syn. As with the previous tests, evaluation of the replies from the end host allows the probe to infer the presence of reordering in either direction. As with the Dual Connection Test, this measurement requires the end host's TCP

implementation to respond in a specific way, to a part of the TCP specification which may not be consistently implemented, and therefore cannot be assumed to be reliable in all cases. Additionally, many Middleboxes assume multiple roles, including that of Intrusion Detection Systems (IDS), and therefore this measurement technique may wrongly be identified as a 'SYN Flood' Denial-of-Service attack, resulting in unreliable measurements.

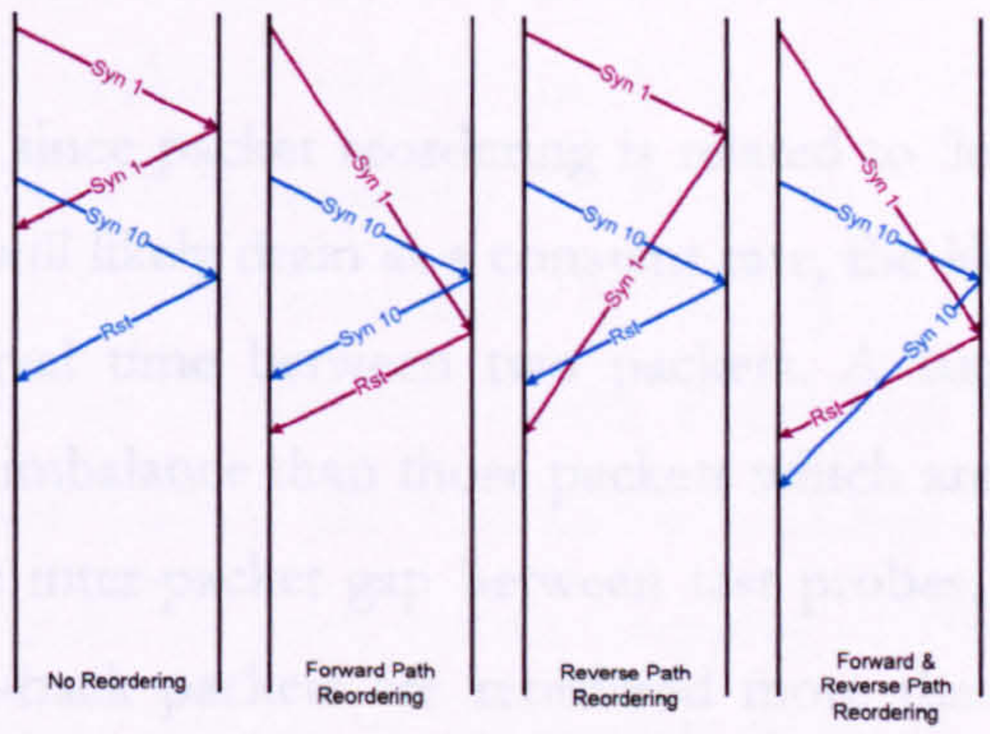

Figure 13 - Bellardo Syn Test

As a fourth test, Bellardo described the 'TCP Data Transfer test', a simple HTTP GET request to a web server. By generating Acks for the largest Seq number recorded, even when data is lost, and also by advertising a small Maximum Segment Size (MSS), it is possible to generate enough data to fill at least two packets and thereby measure the reverse path reordering.

Bellardo chose 50 random hosts across the Internet and, from a single probe machine located at University of California San Diego, cycled through all four tests on each host over 20 days, resulting in 850 measurements per host, where each individual measurement consisted of 15 samples. Bellardo observed that over 40% of the hosts measured experienced some reordering during the 20 day period, with more than 15% of measurements having at least one reordered sample, and with forward path reordering significantly more prevalent than reverse path reordering.

Bellardo found that during periods of significant reordering, the TCP data transfer tests produced significantly lower estimates of reordering than the other techniques. From all the metrics available in their experiments, they chose to report reordering as the

probability that a pair of back-to-back packets are reordered over a given time interval. Bellardo suggested that this inaccuracy was most likely due to the fact that in the TCP test, the size of data packets would be 1500 bytes, whereas the other tests would consist of 40 byte packets. Therefore, the extra delay required to serialise the data for transmission, results in a much larger delay between the leading edges of the data packets, thus reducing the probability of the packets being reordered if they are assigned to different queues.

Bellardo concludes that since packet reordering is related to 'local parallelism' and that queues within a switch will likely drain at a constant rate, the likelihood of reordering is related to the inter-arrival time between two packets. A large inter-packet gap can tolerate a greater queue imbalance than those packets which are closer together. During experiments to vary the inter-packet gap between test probes, Bellardo measured that minimum-sized back-to-back packets are reordered more than 10% of the time, but with an additional 50 μsec delay between packets, reordering decreases to less than 2%, and approaches 0% after 250 μsec. From this relationship, Bellardo proposes that it is therefore possible to infer an application's behaviour when undergoing reordering. For example, during bulk data transfer, full-sized data packets are less likely to be reordered than acknowledgement packets.

Bellardo's conclusions contribute to the discussion on packet reordering prevalence, but the measurement technique makes assumptions on the characteristics of end host TCP implementation, and assumes it will respond in certain ways. This limitation, and the difficulties that would be experienced when determining which method to use, and determining if Middlebox interaction is affecting performance, indicate deficiencies with this particular metric.

3.2.6 Tsinghua

In 2004, Wang [Wang04] presented an active end-point technique, to analyse TCP streams arriving at a measurement probe, and to correlate measurements of packet reordering with network topology. Using traceroute to map the routes to various web server endpoints, and Wget [Wget08] to initiate downloads from those servers, the arriving data packets are analysed using a simple decision algorithm which classifies packets as Normal, Duplicated, Retransmitted or Reordered, as illustrated in Figure 14.
A packet is determined to be out-of-sequence if the Sequence number is less than that of a previous packet; similar to Paxson's metric where 'early' packets are considered 'Normal'. Out-of-sequence packets are classified as Duplicates if they share the same Seq and IPID fields. In order to guard against wrapping of the 4 bit IPID field, a time lag threshold of 300 msec is set, to distinguish between reordered packets, and retransmitted packets with wrapped IPID.

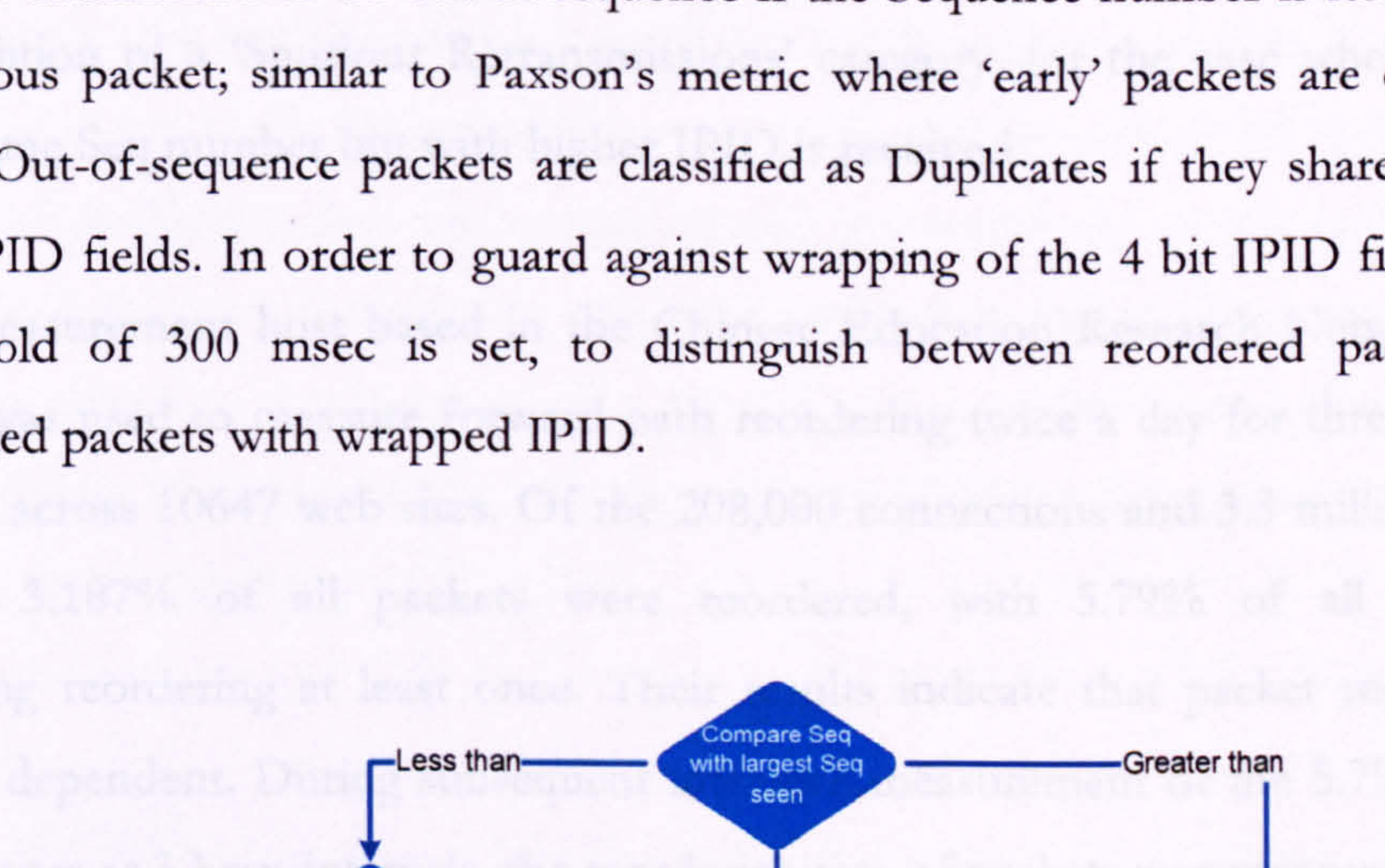

Figure 14 - Tsinghua reorder-judging algorithm

This algorithm classifies late packets as reordered, but some deficiencies can be identified upon inspection of Figure 14. Consider the sequence of packets 1, 2, 5, 3, 4, 6. Packets 1 and 2 arrive in-order and therefore increase the 'largest Seq seen' variable in each case, and are marked as 'Normal Packets'. Packet 5 subsequently arrives, and is seen to be the highest Sequence number seen, thus being marked as a Normal packet and increasing the 'largest Seq seen' variable to 5. Upon arrival of packets 3 and 4, these packets are seen to be lower in Sequence number than 5, and as they exhibit different IPID fields, are wrongly marked as Retransmissions. This wrong classification of 'late' reordered packets may also hold true during IPID wrap, as packets 3 and 4 may arrive within the 300 msec threshold. Additionally, this algorithm could be improved by the simple addition of a 'Spurious Retransmissions' category, for the case when a packet with the same Seq number but with higher IPID is received.

From a measurement host based in the Chinese Education Research Network, Wget [Wget08] was used to measure forward path reordering twice a day for three weeks in May 2003, across 10647 web sites. Of the 208,000 connections and 3.3 million packets measured, 3.187% of all packets were reordered, with 5.79% of all web sites experiencing reordering at least once. Their results indicate that packet reordering is highly site dependent. During subsequent intensive measurement of the 5.79% reorder exhibiting sites at 3 hour intervals, the reordering rate of packets was measured between 2.9% and 3.6% with a mean of 3.187%, compared to a random sample of non-reordering sites, which were constantly measured below 0.04% with a mean of 0.06%. Surprisingly, 20% of the 5.79% reordering sites, exhibited a reordering frequency higher than 80% ! Based on the TTL values, reorder-exhibiting sites are typically those located further away, with average hop count of 13.8, compared to those with less reordering and average hop count of 12.9.

In order to distinguish between a reordered and a retransmitted packet, the authors studied the time lag of the packet arrival. 90% of reordered packets arrived at the receiver with a time lag of less than 5.1 msec, whereas only 3.5% of retransmitted packets arrive within this interval. Within 22.1 msec, 50% of retransmitted packets and 99.6% reordered packets have arrived. Empirical measurements suggest that 12.8 msec is a useful threshold for determining between reordering and retransmissions, where

95% of reordered and only 8.3% retransmitted packets will have arrived.

The authors further investigated the degree of reordering places that packets will move when undergoing reordering. 86.5% of reordered packets were lagging by 1 place, and 95.3% of packets were within 2 places late. Approximately 78.8% of retransmitted packets appeared 3 or more packets late. The conclusion drawn was that there is a small probability of reordered packets triggering the fast retransmit algorithm, and that this 3 position boundary provides a useful method for differentiating reordering and loss.

In order to infer the locations where packet reordering is occurring, Traceroute was used to build a tree of forward-paths from the 10,647 websites to their measurement host, assuming that both forward and reverse paths will be symmetric. Based on this tree, a metric is defined for each router, termed the *reorder ratio* which is the ratio of reordering-websites to total-websites passing through that router. This simple method may help to pinpoint reorder generating routers in some cases, although the authors acknowledge that this approach is extremely limited if the reorder generating router is close to the root of the tree.

3.2.7 Delft

In 2004, Zhou [Zhou04], analysed end-to-end UDP traces between 12 hosts in the RIPE Test Traffic Measurement project [Ripe08]. 50 100-byte UDP 'probe streams' were continuously transmitted, interspersed with 30 second gaps, resulting in approximately 360 probe-streams in 3 hours. The experiments were later repeated with 100 packets per probe-stream. To limit the effects of packet loss, only probe-streams which received at least 90% packets were analysed. This study is distinctive from other measurements it its use of UDP to generate probe packets.

The authors define a number of metrics in order to explain their results. The Reordered Probe-Stream ratio defines the total number of streams having at least one reordered packet, against the total number of streams received. In the first experiment of 50 packets per probe-stream, approximately 56% of the probe-streams included at least one packet delivered out of sequence, equivalent to 6% of the total packets received.

This increased in the 100 packet per probe-stream measurements to 66% of the probe-streams, and 5.6% of the total packets being received out of sequence. This metric indicated that packet reordering is highly site-dependent. Two specific hosts, in Australia and the UK, were measured to exhibit reordering in over 70% of streams in the first test, and 80% of streams in the second test. This suggests that those sites, or intermediate networks towards those sites, were inducing high degrees of reordering.

The Reordered Packet Lag, P_L and the Reordered Time Lag T_L were defined to predict whether a reordered packet would arrive in time at a receiver, in order to be useful to the application, or so as not to expire a finite buffer. In the sequence 1, 2, 4, 5, 3, $P_L = 2$ for the third packet, and T_L is the time difference between the delay of the reordered packet and its expected delay without reordering. Plotting the probability density function of P_L indicated a heavy tail, suggesting that most reordered packets move only a small number of positions, and that each packet in a sequence had the same probability of being reordered, indicating reordering is a Poisson process. Further analysis of the probability of a reordered stream affecting the probability of the *next* stream being reordered, suggested that with 30 second gaps between probe-streams, there was no conditional probability and that reordering is a Poisson process which affects bursts at random. The pdf of the normalised T_L , showed that the 90th percentile was 5% of the one-way delay, suggesting that the time lag induced by packet reordering is very small in most cases.

A final investigation was to measure the degree of asymmetry in reordered probe-streams, by comparing the Reordered Probe Stream Ratio in each direction. The authors observed that asymmetry is present on all measurements, but varies significantly from host to host, and conclude that reordering may be caused by routing policies of the nodes on a path.

3.2.8 Hong Kong Pointer

In 2005, Luo [Luo05] developed a tool call Pointer (Packet reOrderINg tesTER), which implements three methods to measure forward and reverse path end-to-end packet reordering. The measurement is based on TCP data packets; avoiding the use of ICMP

and TCP Syn packets, and therefore decreasing the likelihood that these measurements will be affected by Middleboxes. The three methods differ in the mechanisms used to trigger responses from specific end host TCP implementations, as different implementations respond to unacceptable Sequence Numbers and unacceptable Acknowledgement Numbers in different ways.

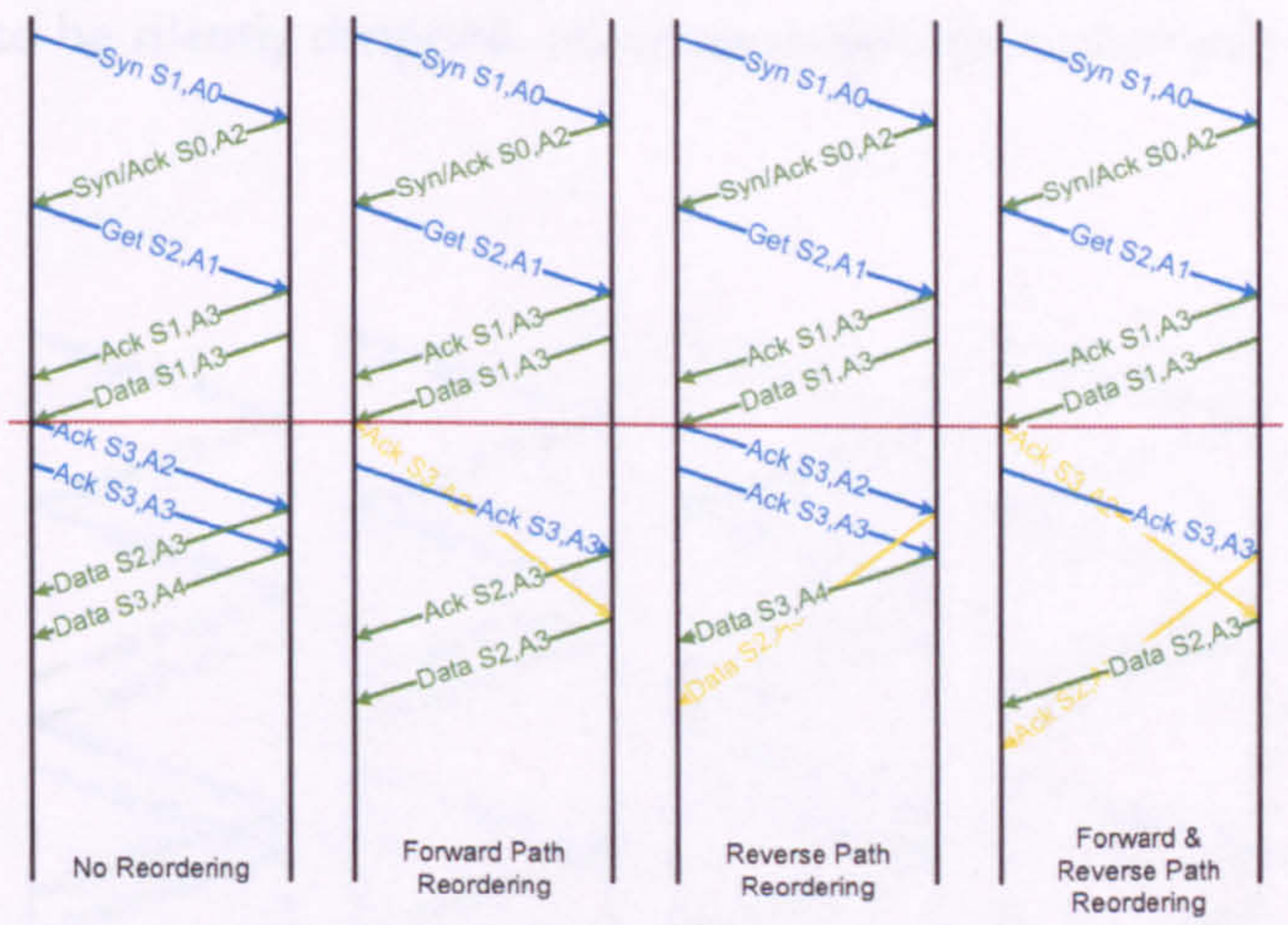

Figure 15 - Pointer ACM Test

The ACM (ACknowledgment based Measurement) Test sends a probing message pair to test a server's response to an unacceptable Ack message. As illustrated in Figure 15, the measurement host establishes a connection with the remote host, by sending a SYN indicating a small advertised TCP window *rwnd*, thus ensuring that the remote server will adopt the same small window, and making future returning packet sizes from the server predictable.

The measurement host then sends an HTTP GET request. This request is acknowledged with a Pure Ack (which does not acknowledge any new data), and then responded to with the first data packet. The data packet is acknowledged by the measurement host but, due to the small *rwnd* and the therefore predictable size of the next data packet, the measurement host is able to construct and send a 'yet-to-receive' acknowledgement packet, Pure Ack S3,A3. This pair of Acks forms the test probing

pair, as indicated below the dashed red line. Comparison of the resulting pair of messages to be received at the measurement host, can then be used as shown in Figure 15, to differentiate between forward, reverse, and combined path reordering.

This method relies on the fact that a TCP host, receiving an unacceptable Ack number, as shown in the case for Forward Path Reordering, will generate a Pure Ack, and as noted by the authors, this is the case for Windows, Mac BSD and Solaris operating systems. For other operating systems, where the receipt of an unacceptable Ack may cause the packet to be silently dropped, other measurement techniques are required.

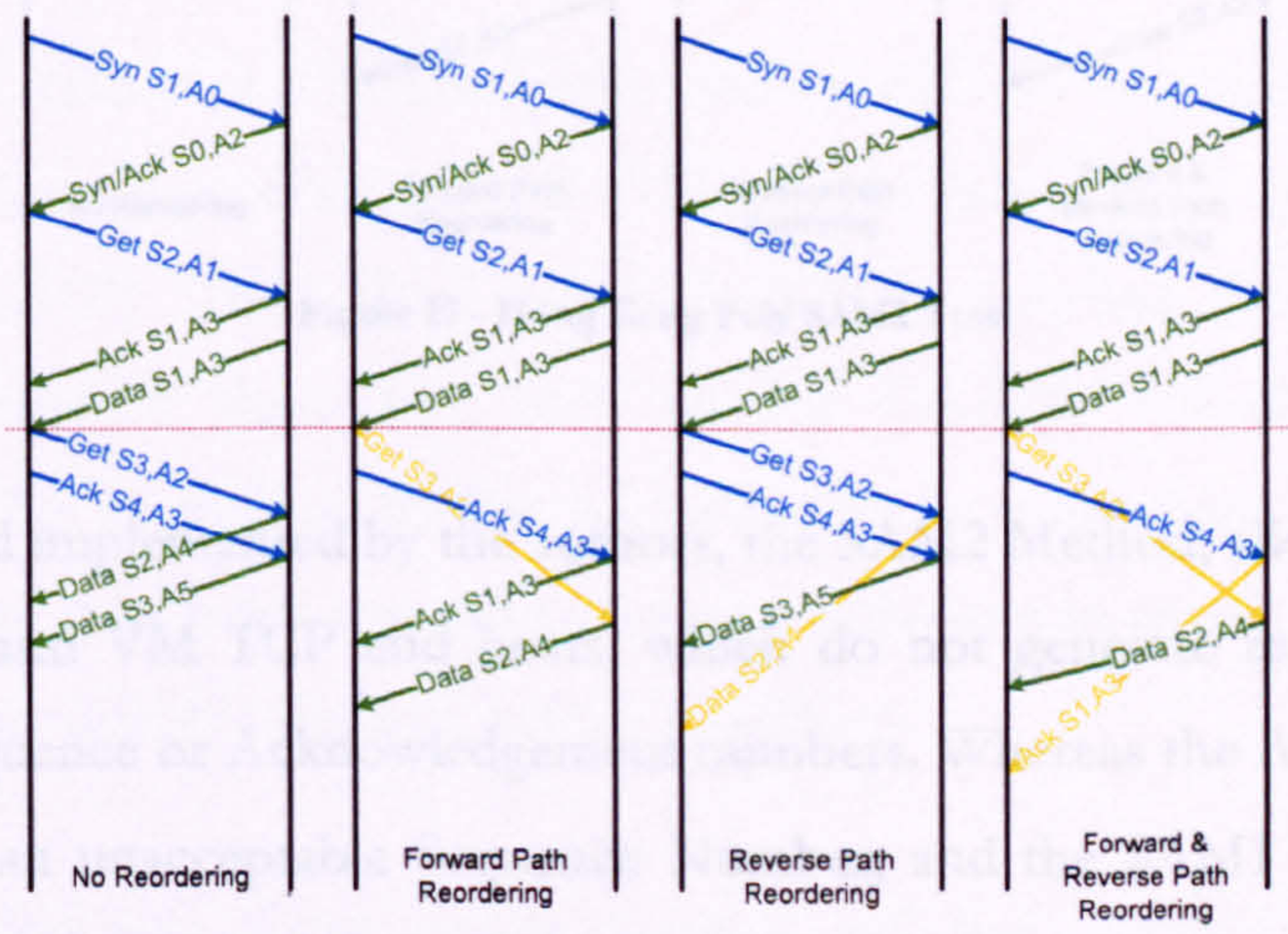

Figure 16 - Hong Kong Poly SAM1 Test

The SAM1 (Seq number and Ack based Measurement) tests the response of an end-host to an unacceptable Seq Number, and is designed for use on Linux systems which will drop the segment and respond with a Pure Ack.

As with the previous method, the probing message pair, sent after the red dashed line in Figure 16, consists of a second HTTP GET request message, and a Pure Ack, which acknowledges a 'yet-to-receive' data segment. Should no reordering occur on the forward path, the probe pair will arrive in order and result in two data packets being sent by the end host. Should reordering occur, the Pure Ack will fail the Sequence Number Check at the receiving side and cause the end host to generate a pure Ack in

response. Differentiation of the resulting Data and Ack messages received at the measurement host, as shown in Figure 16, allows tests for forward, reverse and combined path reordering to be considered.

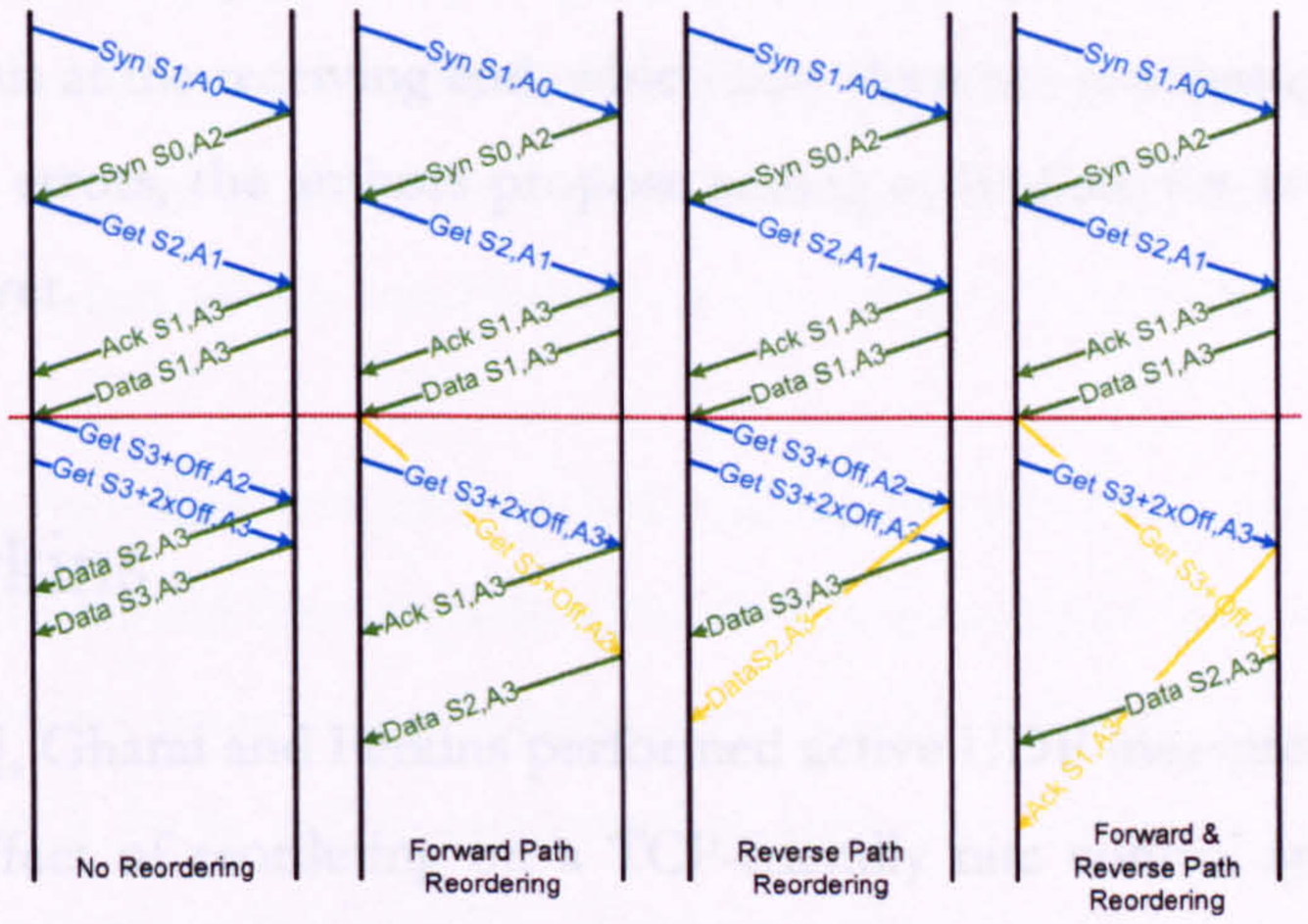

Figure 17 - Hong Kong Poly SAM2 Test

The third method implemented by the authors, the SAM2 Method, allows measurement against HP-UX and VM TCP end hosts, which do not generate responses to either unacceptable Sequence or Acknowledgement numbers. Whereas the ACM method tests the response to an unacceptable Sequence Number, and the SAM1 method tests the response to an unacceptable Acknowledgement, the SAM2 test is constructed to test the end host response to an out-of-order segment. Figure 17 illustrates the two HTTP GET commands sent as the probing message pair. In this case, the Sequence numbers of both packets are offset by a small value, which results in both messages containing unexpected but acceptable Sequence Numbers. The packets are deemed by the receiving host to have been received out-of-order. This method exploits the fact that a TCP should send an immediate Duplicate-Ack upon receiving what it perceives to be an out-of-order segment.

In live Internet trials, the authors performed tests on 100 randomly selected web servers, and sent 500 measurement tests to each server. Their measurements indicated that more than 35% of the paths measured experienced forward-path reordering at least once and 10% of the paths exhibited reverse path reordering. The forward-path

reordering rate was also more prevalent in terms of the percentage of reordering events.

The clear benefits of these techniques are that any end host web server can be used as a measurement host and that the measurements themselves are carried out using TCP data packets. Unfortunately if a measurement packet is lost, it may be extremely difficult to differentiate this at the receiving end, which may skew the reordering results. In order to mitigate these errors, the authors propose setting a deadline for receiving responses from the end server.

3.2.9 Perkins

In 2004 [Ghar04], Gharai and Perkins performed active UDP measurements in order to determine the effect of reordering on a TCP-friendly rate control system for HDTV RTP streams [Perk02]. Using IPerf v1.1.1 [Tiru05] to generate forward and reverse measurements between three test sites on the DARPA SuperNet, the authors investigated whether an increase in data rate, achieved by maintaining a constant packet size and increasing the packet rate, thus causing a reduction in packet inter-arrival time, would result in measurable increased packet reordering.

Each UDP test flow was of one minute duration, with rates varied between 1 and 900 Mbps, and packet sizes of 500, 1500 and 4500 bytes. An optical splitter at each traffic generation host would copy all packets using a modified version of tcpdump [Tcpd08], allowing for further offline analysis of the complete packet traces; thereby allowing loss to be distinguished from reordering.

Perkins defined two metrics to describe packet reordering. The Monotonic Increasing Sequence Metric is similar to previous methods [Jais02] [Paxs99] where packets should monotonically increase in sequence number. Otherwise, packets are classified reordered until a packet arrives with a sequence number larger than the last classified 'in-order' packet received. The number of reordering events is recorded as a percentage of the total number of packets in a flow.

Perkins second metric, the TCP-like Packet Reordering Metric, counts the number of

reordering events which would likely cause a triple-duplicate Ack and probably cause the Fast Retransmission algorithm to activate. This simple algorithm compares a series of three consecutive packets, and establishes whether all three have arrived after a reordered 'late' packet, thus signalling triple-duplicate Acks.

Traceroute was used to confirm that flows were taking the same paths between source-destination pairs, but the authors note that forward and reverse paths exhibit highly asymmetrical properties, which was blamed on the effects of cross-traffic.

Of the 155 flows that were analysed and 60 million packets sent, only 22 packets were measured to be lost during their experiments. The authors argue that the absence of loss indicates that network capacity is available, and that TCP should be able to maintain a high throughput. 73 flows (47%) contained at least one out of order packet and, of those, 48 flows saw more than 0.01% of packets reordered by the monotonic metric. Exhibiting the largest amount of reordering was the Pittsburg to Los Angeles route which experienced 1.65% of reordered packets.

To investigate the correlation between reordering, packet size and inter-arrival time, Iperf [Tiru05] was used to vary the sides of transmitted packets. The majority of flows with packet size 500 octets experienced reordering at rates of 200Mbps and higher, while flows of 1500 octet packets experience reordering at 600 Mbps and higher. None of the flows with packet sizes of 4500 octets, and therefore inter-arrival times greater than 0.04 msec, experienced any packet reordering. The authors argue that there is a threshold, coinciding with a inter-arrival rate of approximately 0.02 msec, beyond which packet reordering will increasingly occur.

The authors summarise that the relative frequency of packet reordering increases as the inter-packet arrival time in the network core is reduced. Therefore, flows with small inter-arrival rates, or flows with high packet rates, will be more seriously affected by reordering than low-rate flows, as illustrated in Figure 18; Future protocol designs should be wary of these implications.

It is acknowledged that the two metrics proposed are non-linear, and that it is not

possible to predict the behaviour of a TCP flow based on the percentage of packets reordered, unless that metric also describes the effects of the pattern of reordering. In one particular example, from Los Angeles to Pittsburgh, the monotonic metric measured 0.04% on both forward and reverse paths, but the TCP-like metric measured 0 in the forward and 30 in the reverse paths. It was found that the monotonic metric would indicate consistent results in both directions, but highly asymmetric measurements using the TCP-like measurement in the reverse direction.

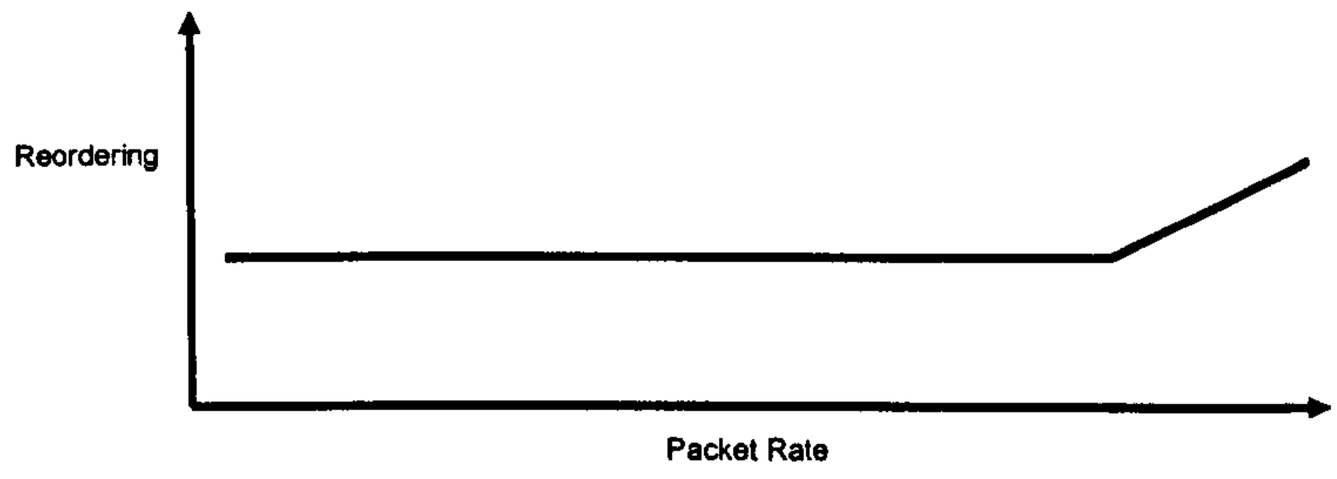

Figure 18 - Perkins relation of reordering and packet rate [Ghar04]

3.2.10 Summary

Section 3.2 has discussed the current field in active measurements of packet reordering, and illustrated the range of numerical results which have been obtained by using these measurements. Comparison and analysis of these measurements is presented in Sections 3.5 and 3.6, where they are presented in context with similar passive measurement results.

3.3 Passive Packet Reordering Measurements

As discussed in Chapter 2, passive measurement techniques provide many advantages over their active equivalents, and allow for characterisation of live, rather than synthetic traffic. The main advantage of passive techniques is that there is no requirement to instrument both endpoints of every connection and measurements obtained are based on real user data, thus providing results more representative of user experience. Positioning of a passive probe is important too; a mid-point probe may have the ability to monitor many thousands of concurrent connections simultaneously, across a diverse array of endpoints.

This section reviews the prior art of passive packet reordering measurements and discusses and comments upon their results.

3.3.1 Limitations of Passive Reordering Measurements

The advantages of a passive approach to TCP monitoring are well documented and the literature includes many examples of these measurement methodologies. However, a large number of these methodologies and metrics have been designed on the same assumption as the Fast Retransmission algorithm, that packet reordering does not often occur. There are, therefore, many examples where passive measurements simply ignore the effects of packet reordering, which can lead to errors in the results obtained by these methods.

In work by Benk [Benk02][Benk04], a passive mid-point method for estimating end-to-end TCP packet loss is presented which, by observing sequence numbers and using heuristics, attempts to infer the state machines of the sending and receiving TCPs. A packet loss before the measurement point is determined by observation of an out-of-order TCP segment that 'fills a hole' in the data sequence. Similarly, a repeated sequence number suggests that a retransmission has been sent due to a loss after the measurement point. When packet reordering occurs, the algorithm erroneously assumes that the holes

in sequence numbers indicate packet loss before the measurement point. In order to mitigate this problem, the authors suggest using the IPID field in order to detect when packet reordering is occurring.

A method for calculating TCP 'Goodput' is presented by Love [Love06], where Goodput is defined as the ratio of useful data divided by total data, thus presenting a ratio of the degree of retransmissions a network is exhibiting. In this passive mid-point algorithm, the Sequence number of each arriving packet is examined and assumed to be monotonically increasing. When a packet arrives with a lower sequence number than those seen previously, it is immediately assumed to be a TCP retransmission, resulting in a retransmission counter being incremented and the goodput calculation being adversely affected.

Implementing a passive measurement technique in itself is not particularly challenging; the difficulty is trying to explain observed events, based on the limited information available at that observation point. Visibility of out of sequence packets at an observation point can infer a number of possible scenarios, but without knowledge of the TCP state machines at either the sender or receiver, the challenge is to determine the root cause of the observed event.

The following section reviews the state-of-the-art in passive packet reordering measurements and presents results obtained using these methods.

3.3.2 Mid-point Passive Measurements

In 2003, Jaiswal [Jais07] performed a large scale study of packet reordering, by instrumenting a mid-point of a Tier-1 IP backbone, as part of the Sprint IP Monitoring (IPMON) [Fra03] project. A passive mid-point measurement has the advantage of allowing very large scale measurement studies to be carried out, without requiring instrumentation of sender and receiver hosts. The sheer scale of this measurement study, namely several-hour packet-level traces from a set of OC-12 and OC-48 links for 29 million TCP connections generated in nearly 7600 unique ASes, and the fact that the study is performed passively on live TCP traffic, make this paper an important

contribution in the measurement of packet reordering.

Performing passive TCP monitoring at a mid-point is not without its challenges. A packet can easily be identified as being out-of-sequence when it is observed as having a sequence number smaller than or equal to that of a previously observed packet at that measurement point. Explanation of the *cause* of the packet appearing out-of-sequence is challenging, as many variables, such as the state machines at the sending and receiving hosts, can only be *inferred* from the packets observed at the mid-point, and therefore a set of heuristics are required in order to examine the packet events observed.

Additionally, several of these inferences will require an estimation of the sending host's congestion control, RTO and RTT values, for every source-destination pair, throughout the lifetime of every connection. These may also vary significantly over time. Tracking the RTT calculation and congestion control mechanisms of a sending host, from a mid-point position, is extremely challenging and techniques to perform this are discussed in Section 3.3.3.1. The major contribution of mid-point measurement algorithms such as Jaiswal, is explaining why these out-of-sequence events have occurred based purely on the analysis of the previously and subsequently observed packets.

3.3.3 Jaiswal TCPFlows

Jaiswal's TCPFlows algorithm, as illustrated in Figure 19, allows the classification of out-of-sequence packets as either Sender Retransmissions, Network Duplicates or Packet Reordering. During evaluation of the algorithm [Jais07], Jaiswal argues that it was possible to classify almost all observed packets using this algorithm, with between 1% and 4% of packets being classified in the Unknown category.

At the measurement point on each of the links monitored, two probes are used to capture the first 44 bytes of IP and TCP packet headers in both the forward and reverse directions. Sequence Numbers, Acknowledgement Numbers, IPID field and observation time are used in the algorithm. Post-processing of the traces is carried out offline where, firstly, the traces are filtered to consider only the TCP connections where both the forward data path and reverse Ack path have passed through the same

measurement point and have been logged entirely. Out-of-sequence packets, those which have sequence numbers less than or equal to a previously observed packet in that connection, are then classified using the rules in Figure 19.

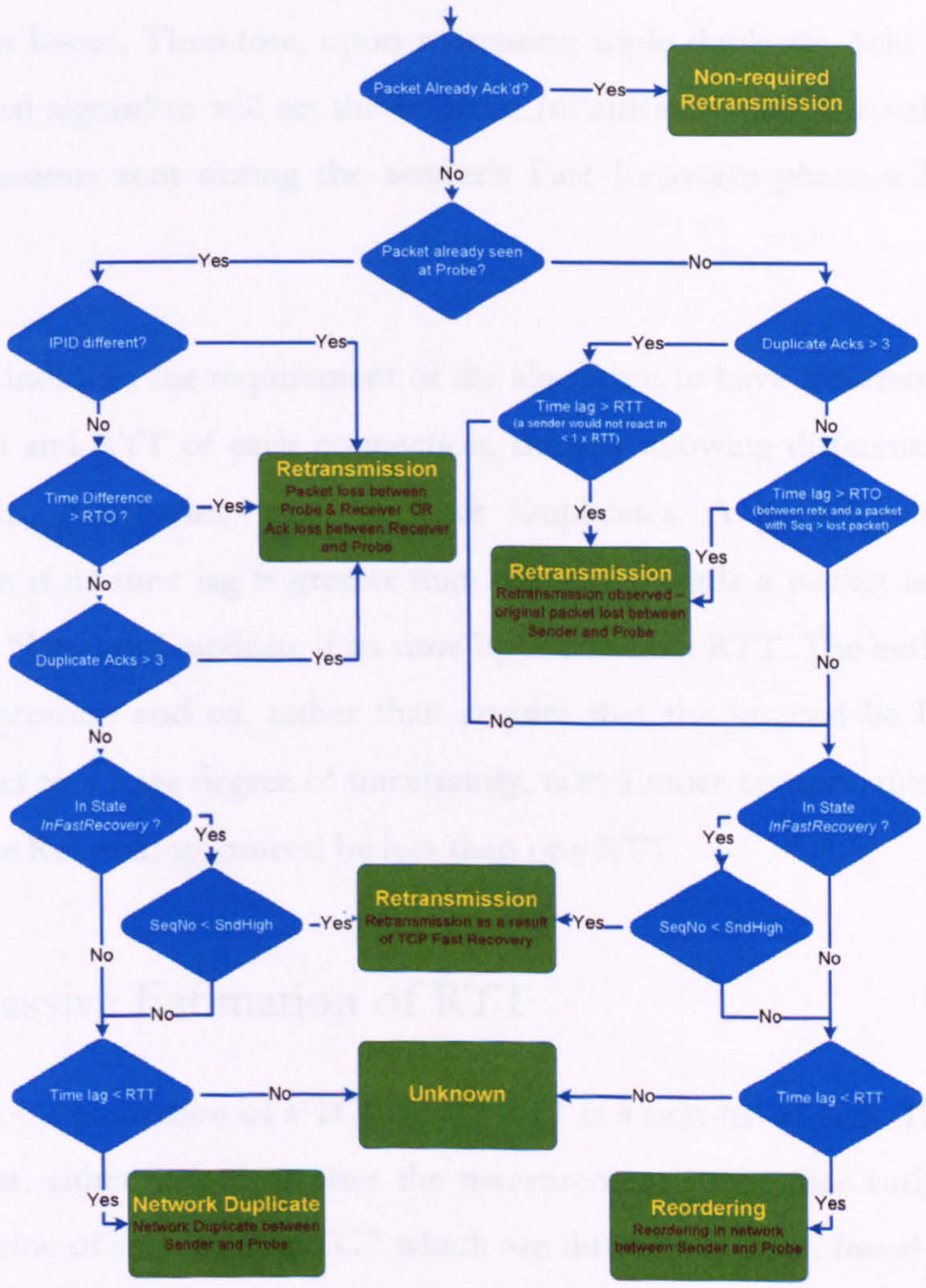

Figure 19 - Jaiswal's Out of Sequence Classification Algorithm

As Figure 19 illustrates, a number of variables are used in the classification algorithm which must be inferred from the data and Ack packets as seen from the mid-point of each connection. In order to classify those retransmissions which do not have distinct IPIDs, nor occur within the RTO period, the Jaiswal algorithm sets two variables *InFastRecovery* and *SndHigh*, in order to cater for retransmissions sent during a sender's Fast Recovery phase. When recovering from multiple packet losses in one flight of

packets, a TCP NewReno sender will, upon receiving three duplicate-Acks, trigger Fast Recovery, and set the Sequence Number of the most recently sent data packet in the variable *sndHigh*. It will then retransmit the lost packet but, in addition, will immediately retransmit any packet for which any partial-Acks are received between the triple duplicate-Ack and the *sndHigh* sequence number, thus allowing faster recovery from multiple packet losses. Therefore, upon measuring triple duplicate-Acks in the reverse path, the Jaiswal algorithm will set the boolean *InFastRecovery* and the value of *sndHigh*; thus retransmissions sent during the sender's Fast Recovery phase will be classified appropriately.

Figure 19 also indicates the requirement of the algorithm to have a current estimation of both the RTO and RTT of each connection, thereby allowing differentiation between Retransmissions, Reorderings and Network Duplicates. A packet is classified as a Retransmission if its time lag is greater than the RTO, while a packet is classified as a Reordering or Network Duplicate if its time lag is less than RTT. The authors argue that this is a compromise and so, rather than require that the interval be less than RTO which is subject to a large degree of uncertainty, take a more conservative approach and require that the Reordering interval be less than one RTT.

3.3.3.1 Passive Estimation of RTT

Passive mid-point estimation of a TCP flow's RTT is a non-trivial task. The loss of Data or Ack packets, either before or after the measurement probe, has various actions on the state machine of the Sending TCP which are difficult to infer, based on the sub-set of packets actually observed at any intermediate point. Previous techniques [Jian02] have estimated RTT during the initial SYN-ACK handshake at the start of a connection or on the time difference between subsequent window transmissions at the start of a connection [Mart00].

3.3.3.2 Jaiswal Running RTT Estimation Technique

The 'Running RTT Estimation Technique' [Jais04], used by the Jaiswal Out of Sequence Classification Algorithm, is illustrated in Figure 20. The addition of observation times *t1* and *t2*, allow for inference of the TCP Sender's RTT estimation. The argued benefit of this technique compared to previous methods, is that it allows calculation of RTT throughout the lifetime of the connection rather than on just one sample at the start. Under some circumstances, this method will compute one RTT sample for every 'round' of packets sent by the sender which will equal the number of samples used by the Sender itself to calculate RTT.

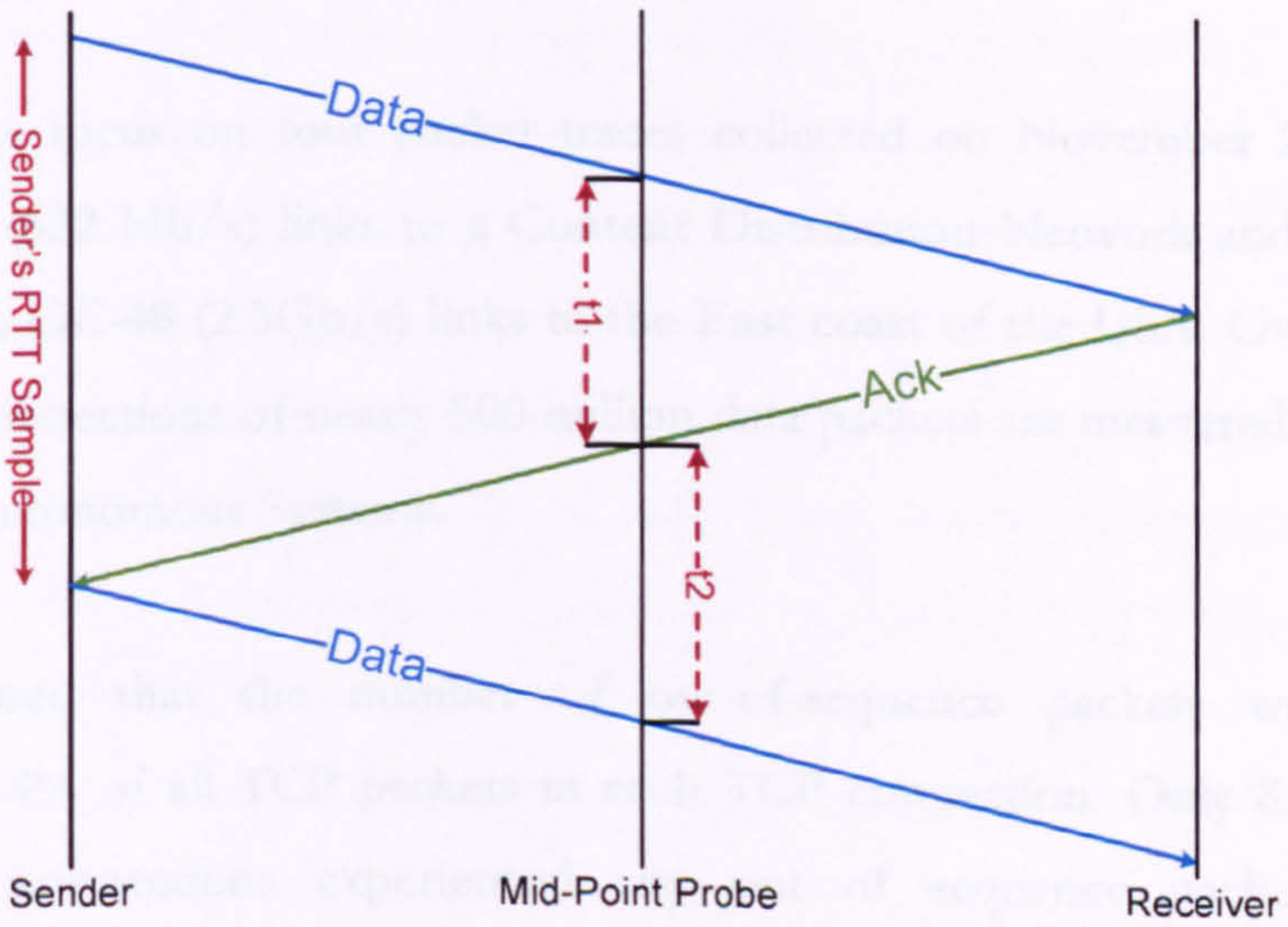

Figure 20 – Jaiswal Running RTT Estimation Technique

The problem is not as simple as it first appears. In order to operate, this technique requires the ability to correlate from a mid-point position, the particular Ack packet that has triggered the transmission of each data packet observed. This, in turn, requires knowledge of the sender's current *cwnd* size which is a function of the particular TCP congestion control algorithm operating on the Sending host. The RTT estimation technique must be able to stop the RTT estimation during loss recovery, in order to emulate a transmitting TCP session which also does not compute RTT during loss recovery. In order to do this, knowledge of the state of the Sending TCP and an estimation of that TCP's *cwnd* is required.

The Jaiswal RTT estimation technique estimates the sender's *cwnd* by constructing a replica TCP Finite State Machine (FSM) for each connection and, based on mid-point observations of Acknowledgements and inferred Sender Timeouts [Jais03], the FSM is progressed through its various states. The algorithm performs TCP fingerprinting by observing the packets sent after a loss and Fast Retransmit has occurred, thus relating this inferred *cwnd* to the behaviour of either a Tahoe, Reno or NewReno host. Later in their work, the authors acknowledge that 97.05% of all TCP Senders and hence 67.51% of data packets sent, were indistinguishable using this fingerprinting technique, as it will only operate under a specific simple loss pattern.

3.3.3.3 Jaiswal Classification Results

Jaiswal's results focus on four packet traces collected on November 21st 2002; two across OC-12 (622 Mb/s) links to a Content Distribution Network and a Tier-1 ISP, and two across OC-48 (2.5Gb/s) links to the East coast of the USA. Overall nearly 30 million TCP connections of nearly 500 million data packets are measured, originating in 7664 unique Autonomous Systems.

Jaiswal measured that the number of out-of-sequence packets was limited to approximately 4% of all TCP packets in each TCP connection. Only 8.8.% of all the studied TCP connections experienced any out of sequence packets, but these connections consist of a significant fraction of all the data packets (48%). Longer connections would be expected to experience more out-of-sequence packets.

The majority of out-of-sequence packets were due to Retransmissions. Across the four links monitored, this was identified as the cause between 64% and 79% of the time. Note that a source may send more than one packet to repair a loss and thus this metric does not correlate directly with packet loss. Unneeded Retransmissions, varying between 11 and 15% are the second highest cause of out-of-sequence packets.

Network Duplicates were found to be negligible and the cause of approximately 0.1% of out-of-sequence packets. Packet Reordering was measured as the cause of 7.04%,

25.89%, 16.06% and 16.57% of out-of-sequence packets across the four monitored links. This indicates that reordering affects between 0.17 and 0.96% of all data packets, and that between 0.6 and 5.1% of TCP connections experience packet reordering.

In the four traces, approximately 93% of the reordered packets have a packet lag of less than 3 and, therefore, Packet Reordering will have a minimal impact on a connection's performance. 92% of all reordered packets have a time lag of less than 50 msec, and 88% of all reordered packets have a time lag less than 50 msec and a packet lag of less than 1. The authors argue that, when compared to the delayed acknowledgement timeout of between 50-100 msec, packet reordering will have a minimal effect on end-user performance.

From further examinations of Acks, it was possible to suggest, from the measurements, when reordering or duplication may be occurring between the measurement probe and the receiver. Assuming that duplicate-Acks from previously classified events and unneeded retransmissions have been filtered out, it is possible to infer that any remaining duplicate Acks are indications of reordering between probe and receiver. This ignores the possibility that duplicate Acks may be sent to communicate updates of *rwnd*, or that delayed acknowledgements from the receiver would prevent duplicate Acks from being sent. By applying this analysis to the four monitored links, the computed estimate for end-to-end reordering and duplication was estimated to be 1.13%, 1.75%, 1.02% and 1.29% respectively.

3.3.3.4 Evaluation

There are clearly a number of issues with this measurement methodology which must be considered when discussing the results obtained. Firstly, the requirement that both the forward and reverse paths pass through the measurement point is highly unlikely due to the asymmetric nature of the Internet, resulting in some experiments where only 9.2% of the TCP connections could be analysed. Secondly, the authors acknowledge that if a SYN is lost before the measurement point, or an entire window of packets is lost before the measurement point, it will go completely unnoticed by the classification algorithm.

There are also clear deficiencies with the classification algorithm itself. Reverse-path Ack reordering and cumulative Acks have not been considered, which could cause the algorithm to wrongly classify packets as Unneeded Retransmissions. The algorithm does also not acknowledge that these packets could also have been delayed Network Duplicates. The Retransmission category could easily be extended to identify the cause of retransmissions; either as a result of Fast Retransmission, or correlated with those retransmissions resulting from Fast Recovery.

Finally, the RTT Estimation algorithm also has clear deficiencies. Since RTO calculation is subject to some uncertainty, the authors use RTT as their method of differentiating original packets from reordered packets. This requires a degree of accuracy and, although the authors claim [Jais04] that connections do not experience large RTT variations and that for 80-85% of connections the ratio between the 95th percentile and 5th percentile RTT value is less than 3, other recent work [Aika03] has measured that RTT values can vary widely over the lifetime of a connection. Furthermore, the value of *t1* can be affected by delayed Acknowledgements, while the value of *t2* can be affected if the sender receives an Ack, but does not have any data immediately ready to send.

RTT estimation is a problem which affects most mid-point measurement techniques, as it is accepted that a single estimate will not characterise network variability across the lifetime of a connection. An extension of Jaiswal's work, proposed by But [But05], simply estimates *t1* in each direction, thus negating the need to estimate *t2* and avoiding errors induced by delayed Acks. This technique does, however, require observation of full duplex TCP connections. Although all TCP connections are full duplex, many operate in a half duplex fashion, and are highly asymmetric in the volumes of data carried. For example, in a typical Active FTP session, a full duplex control connection is established on server port 21 for infrequent control commands, and individual half-duplex TCP connections are established for each PUT or GET command. But's technique therefore requires every observed TCP connection to operate in full duplex, and to each have sufficient data ready to be sent, from both sides of every connection. Should there be a lack of data to send from either side of a connection, the Delayed Ack algorithm may interfere with the RTT estimate observed. Other techniques [Veal05] [Yan04] have used TCP Timestamps in order to increase accuracy of mid-point RTT

estimation, although some have argued [Medi05] that this option is not in common use and so is not appropriate on the majority of passively monitored connections.

Although Jaiswal's work represents the most powerful mid-point packet reordering methodology to date, the limitation that only 9.2% of paths were symmetric and that 97% of TCP flavours were unidentifiable by the fingerprinting method, question how representative of the whole Internet these results actually are. Therefore, although Jaiswal concludes that a relatively constant 4% of packets in the Internet are out of sequence and that the majority are retransmissions with only a small percentage due to reordering, these results must be considered in the context of the limitations identified above.

3.3.4 Rewaskar

In 2006, Rewaskar [Rewa06a][Rewa06b] developed a passive mid-point probe classification tool for out-of-sequence packets. This extends the work of Jaiswal by considering variations across TCP implementations and explicit analysis of the cause of each retransmission. The authors argue that there are diverse and undocumented features of stack implementations which affect TCP behaviour and that significant numbers of retransmissions on the Internet are unnecessary.

The main purpose of this algorithm is to classify the TCP mechanism which caused each packet retransmission, and to indicate if that retransmission was required or unnecessary. Data and Ack streams from each connection are parsed by replica partial TCP state machines augmented with extra state and logic about *all* previously transmitted packets; each retransmission is then classified as *needed* or *unneeded* and further classified as caused by RTO, Triple Duplicate Ack, Partial Ack, Selective Ack or Implicit. The replica state machines include the implementation details of four prominent TCP stacks (Windows XP, Linux 2.4.2, FreeBSD 4.10, and Solaris), such as initial RTO, the RTO estimation algorithm, the number of duplicate ACKs that trigger Fast Retransmit and the responses to partial ACKs and SACKs. Many of these features are specific to the implementation or, in the case of SACK responses, non-standardised.

Packets are classified as Reordered if they appear within 0.75 of that connection's minimum RTT after the segment with the next higher sequence number or, in the IP ID field of the packets seen from that source. Duplication is also identified using IP ID. Each potential indicator of packet loss, as identified from the Ack stream, will only trigger tentative changes in each state machine until a retransmission is confirmed in the data stream.

To process results from experiments, each connection is parsed using all four state machines; the state machine which can explain and classify the most packets in that trace is selected and the results are stored.

3.3.4.1 SYN/ACK RTT Estimation

RTT estimation for each flow is partially based on the SYN SYN/ACK handshake at the initiation of each connection. During the initial three-way handshake, the measurement probe to Sender RTT is estimated. This estimate is added to repeated samples, from the data and ack flows, of the measurement probe to Receiver RTT estimate. The authors argue that the initial value obtained from the three way handshake is a good approximation of the minimum probe to Sender RTT [Aika03] and, that if subsequent delays vary significantly, this would not greatly affect results. RTO is used as a minimum threshold for the differentiation between the original packet and a retransmission. Therefore, a value lower than RTO would simply lower the threshold, but would still be able to correctly identify retransmissions that occur due to timeouts.

The authors do not discuss how variations in their RTT calculation could affect the differentiation of reordered packets.

3.3.4.2 Rewaskar Classification Results

Seven tracesets were analysed; three of which were compared with Jaiswal's algorithm. In two tracesets, between 13% and 14% of out-of-sequence packets were classified as the result of network reordering between the sender and monitoring probe. The majority of reordering events were measured to be within 5ms. The small fraction of out

of sequence packets with large delays was said to occur in connections with large minimum RTTs. A significant number of retransmissions, between 3% and 19%, were shown to be unneeded, suggesting that retransmissions should not always be considered as indicators of packet loss.

Comparison with results from Jaiswal's algorithm was favourable. Of the three tracesets where both algorithms were applied, Jaiswal reported 0.8, 13.8 and 0.27% of out of sequence packets were due to reordering, while Rewaskar reported 0.2, 12.9 and 0.2% of events due to reordering. This suggests that, from a reordering classification perspective, there is little additional merit in considering TCP implementation specific features.

For each of the seven tracesets, between 25 - 35% of connections exhibited at least one packet which could not be classified correctly. The authors acknowledge that in more than 50% of these traces, this was due to more than one state machine claiming to be able to explain every packet in the trace; resulting in these traces having to be discarded.

As with the Jaiswal algorithm, Rewaskar has a requirement that all packets, from both forward and reverse paths, are fully recorded at the measurement probe. Rewaskar has the additional requirement that each connection must be analysed and recorded directly from initialisation, in order to sample the three-way handshake and generate a measure of the RTT value from probe to sender. The results presented using the Rewaskar algorithm are based on pre-recorded publicly available traces. Therefore, application of this algorithm on a real probe on a live network, in order to illustrate the percentage of flows which might not be asymmetric and, therefore, unclassifiable, has not been tested.

3.3.5 Tstat Torino Algorithm

In 2006, the Jaiswal classification algorithm was extended by Mellia [Mell06] to allow the classification and root cause analysis of additional modes of out of sequence packets.

As with Jaiswal, the algorithm is designed as a mid-point technique and therefore assumes visibility of both data and Ack paths. 'Anomalous events', such as either duplicates or out-of-sequence packets, are classified using the algorithm shown in Figure

21. The algorithm maintains a number of variables in order to parse each packet. RTT_{MIN} is the minimum RTT since the flow commenced. *RT Recovery Time* is the time elapsed between the time the current anomalous segment has been observed and the time the segment with the largest sequence number has been received. ΔT is the *inverted-packet gap* - the difference between the observation time of the current anomalous event and of the previously received segment. RTO is the sender Retransmission Timer value, computed by observation from the midpoint, as defined in RFC 2988.

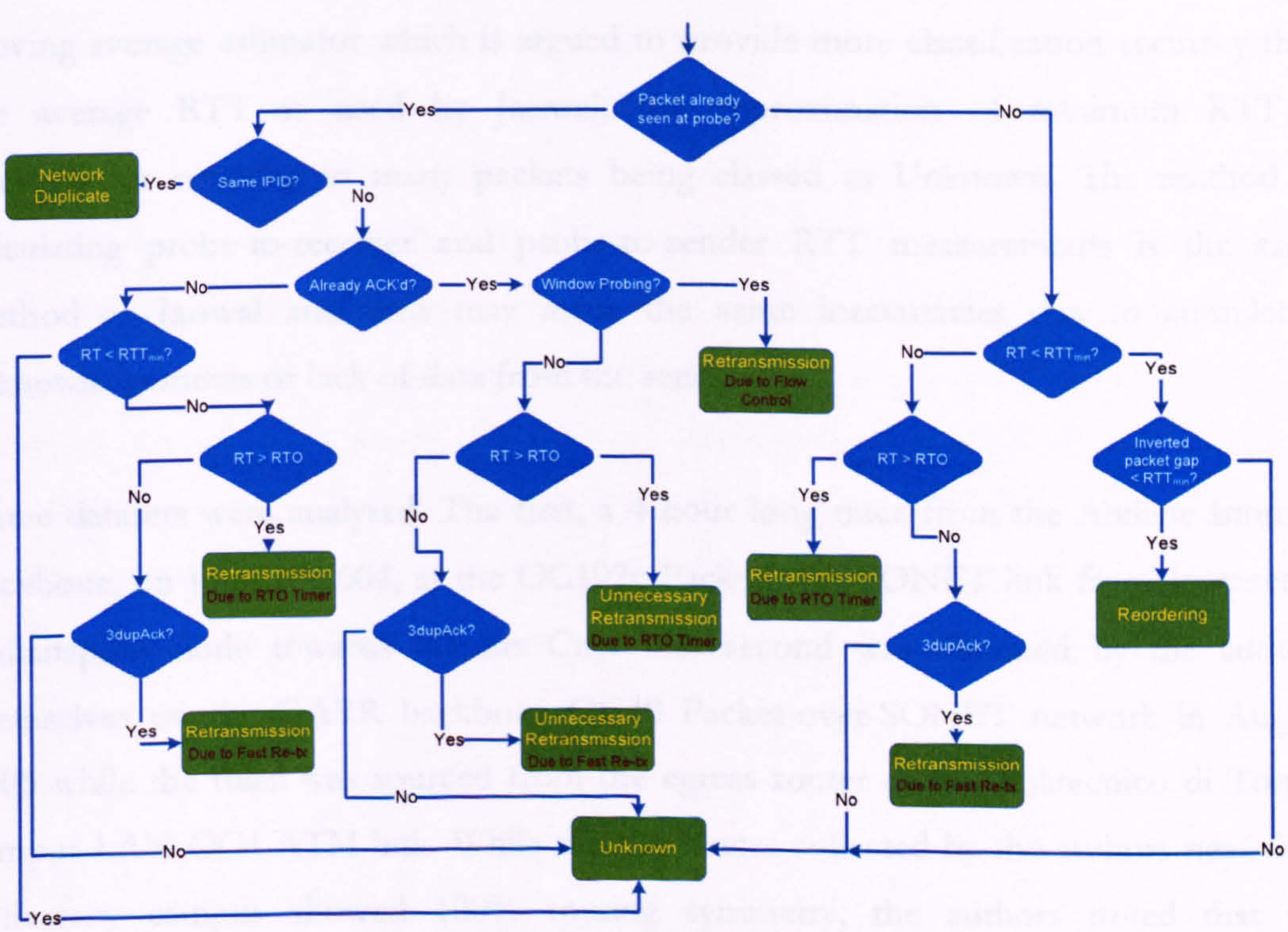

Figure 21 - Torino Algorithm

The algorithm extends Jaiswal in a number of ways. Firstly, all packets with the same IPID are immediately defined as Network Duplicates, regardless of arrival time. Secondly, the algorithm accounts for the 'Window Probing' feature of Flow Control, whereby a TCP sender will retransmit its last segment, in order to force transmission of an Ack, thus measuring if the *rwnd* has increased from zero. These window probe packets are easily identifiable; the Sequence number appears to repeat the last byte of the previous segment, but payload is empty (zero).

The algorithm does not attempt to model the sender side state machine as other

classification methods have done. Therefore, retransmissions sent during the Fast Recovery phase are, unlike Jaiswal, not explicitly identified as such. Retransmissions are classified as those which occur if RT is greater than RTT_{MIN} and, by comparing the RT with the estimated RTO, retransmissions are distinguished between RTO, Fast Recovery or Unknown. If RT is less than RTT_{MIN} and the *inverted-packet gap* is less than RTT_{MIN}, the packet is classified as Reordered.

RTT is calculated from the mid-point throughout the lifetime of the connection using a moving average estimator which is argued to provide more classification accuracy than the average RTT as used by Jaiswal; the approximation of minimum RTT is conservative resulting in many packets being classed as Unknown. The method of calculating probe-to-receiver and probe-to-sender RTT measurements is the same method as Jaiswal and thus may incur the same inaccuracies due to cumulative acknowledgements or lack of data from the sender.

Three datasets were analysed. The first, a 4 hour long trace from the Abilene Internet backbone, on June 1st 2004, at the OC192c Packet-over-SONET link from Internet2's Indianapolis node towards Kansas City. The second was obtained by the authors themselves on the GARR backbone OC48 Packet-over-SONET network in August 2005 while the third was sourced from the egress router of the Politecnico di Tornio campus LAN OC4 ATM link. While the two traces collected by the authors near their University campus showed 100% routing symmetry, the authors noted that the backbone Abilene trace indicated that only 46% of paths on a large Internet backbone may be symmetric. Where both directions of the traffic have not been captured, the authors discard the trace.

Of the backbone packets analysed, between 5% and 8% of all packets were measured to be out of sequence. Of these out of sequence packets, 10.5% were classified as packet reordering, 19% as Unknown, and nearly 70% as retransmissions triggered by RTO. The remaining < 1% was accounted for by the other classifications.

By normalising the breakdown of out-of-sequence packets against traffic load and by analysing the results over a month, the authors argue that the percentage of anomalies

appeared to be independent of the load. This contrasts with other findings which suggest loss and reordering as a function of applied load on a router [Benn99]. The authors argue that, due to the greedy nature of TCP even during off-peak periods when the average offered load is lower, TCP will grow to consume as much bandwidth as possible and so reordering rates will appear constant. On certain connections, including some very long connections, no reordering would occur at all, suggesting that reordering is path dependent.

The Torino algorithm has clear comparisons with Jaiswal's classification algorithm, but also shares many of the same issues. The requirement for symmetric paths, the deficiencies in calculating an estimate of RTT and the very large number of classifications which resulted in Unknowns, question the value of the results. The additional complexity in the algorithm to classify less than 1% of the packets is difficult to justify and, therefore, the contribution beyond Jaiswal's work is questionable. The conclusions from the authors suggest that, although the absolute amount of out-of-sequence events is highly dependent on the link load, the relative amount compared to the total traffic and the classification breakdown are independent of the current load. This suggestion, that packet reordering could be independent of offered load requires further consideration.

3.3.6 Summary

Section 3.3 has discussed the current state of the art in passive packet reordering measurement techniques, and the range of numerical results obtained. Clearly mid-point observation affords many benefits in the number of connections which can be monitored, but results in increased complexity and the requirement to calculate an estimate of RTT per flow. Further comparison of these techniques is presented in Section 3.6.

3.4 Packet Reordering Metrics

As discussed in Chapter 2, Internet standardisation is a loosely defined process, defined by RFC documents which can be produced by IETF working groups, or individual parties. The IP Performance Metrics (IPPM) working group produced RFC 4737 in November 2006, which defines whether a network has maintained packet order, on a packet-by-packet basis and the context information required for all metrics. In June 2008, an individual contribution has resulted in RFC 5236, 'Improved Metrics for Packet Reordering'.

The authors of both RFCs agree on the requirements for packet reordering metrics. Packet reordering metrics must have relevance to an application, be computable 'on the fly', be relevant to TCP and real-time performance and allow for the concatenation of separate segments to estimate the reordering of an entire path. This section discusses and compares the proposed metrics and illustrates some results from the limited number of deployments documented in the literature.

3.4.1 IP Performance Metrics Standardisation

The IP Performance Metrics Working Group is charted by the IETF to define metrics and measurement methodologies for network performance evaluation. IPPM has defined metrics for measuring connectivity RFC2678, one-way delay RFC2679, one-way packet loss RFC2680, round-trip delay RFC2681, bulk transfer capacity measurements RFC3148, one-way loss patterns RFC3357, IP packet delay variation RFC3393, a one-way active measurement protocol RFC4656, Network Capacity RFC5136 and recently packet reordering metrics RFC4737.

Current research involves producing standards for a Two-way Active Measurement Protocol and a One-way Packet Duplication Metric.

3.4.2 RFC 4737

RFC 4737 highlights that packet reordering may be present on a steady-state basis, which is easily detectable by minimising spacing between test packets, or the reordering may be on a transient basis as a result of network instability. The standard therefore defines a method to determine a 'reordered singleton' – an atomic metric to indicate whether or not packet order has been maintained. It then defines multiple sample metrics to quantify the degree of reordering in terms of frequency and distance between events, since one metric that quantifies a key aspect of one receiver's behaviour may be completely irrelevant to another.

As with other IPPM metrics, RFC4737 is an active measurement protocol. In order to provide the context in which the measurement was made, RFC2330 defines a packet 'of type-p', which ensures that the constructed probe packets are designed so as not to receive any different packet treatment from any other data packets on the Internet and that their specific construction is reported in the context of the measurement. Additionally, the sending stream parameters must be reported with the metric, so as to document if the probe stream is periodic as in RFC3432, TCP-like as in RFC3148, or Poissonian as in RFC2330.

3.4.2.1 A Reordered Packet Singleton Metric, Type-P-Reordered

The metric to identify if packets are arriving in sequence requires the implementation of a sender which produces a series of monotonically increasing identifiers at the source on each packet in order to establish the original order of transmission.

At the destination, a method is required to examine the 'Next Expected' packet number, which may either be computed by the destination, or transmitted to the destination offline for correlation. In a TCP scenario, the value of Next Expected would be the sequence number of the previous packet plus payload size.

If Sequence Number of received packet is >= Next Expected, then Type-P-Reordered

= False, else Type-P-Reordered = True. Packets with a Sequence Number > Next Expected are considered as a special case of in-order delivery, caused by packet loss or reordering. A SequenceDiscontinuity metric is calculated for these gaps in sequence numbers, using either packets, bytes or time.

3.4.2.2 Sample Metrics

A number of sample metrics are defined to assess the degree to which a packet is reordered with respect to other packets in the flow. These are illustrated in Table 1.

Name	Unit	Description
Reordered Packet Ratio	Percentage	Count of packets with Type-P-Reordered=True/Total # of packets
Reordering Extent	Packets	The maximum distance, in packets, from a reordered packet to the earliest packet received that has a larger sequence number.
Reordering Late Time Offset	Time	Indication of lateness in terms of the buffer time that a receiver requires to accommodate a reordered packet.
Reordering Byte Offset	Bytes	Indication of lateness in terms of the storage bytes required that a receiver must possess to accommodate a reordered packet.
Gaps between multiple Reordering Discontinuities	Packets or Time	The distance between successive reordering discontinuities.
Reordering-Free Runs	Packets	The count of consecutive in-order packets between reordered packets.
TCP-Relevant Metric	Percentage	The percentage of packets which are reordered by a distance $\geq n$ packets, where, if n=3, a NewReno sender would consider this packet lost for purposes of congestion control. 3 is the default threshold for Stream Control Transport Protocol RFC2960, and the Datagram Congestion Control Protocol RFC4340 when used with Control ID 2: TCP-like Congestion Control RFC4341.

Table 1 – RFC4737 Sample Reordering Metrics

3.4.2.3 Evaluation

The sample metrics provided by RFC4737 are relatively simplistic and do not attempt to explain the cause of packet reordering. The Type-P-Reordered non-reversing order criterion means that packet losses alone do not cause subsequent packets to be classified as reordered and the criterion results in only 'late' arriving packets being classified. It is noted by the authors that determining reordering extents and gaps will be exceptionally

difficult when there are overlapping or nested reordering events occurring.

In scenarios where a user wishes to apply these metrics to a normal TCP data stream, the authors suggest that, since the sequence numbers are based on the byte stream with varying packet sizes, care must be taken to not declare retransmissions as reordered and that the TCP timestamp field [RFC1323] should be used.

3.4.2.4 Results

There are few reports in the literature of successful measurement studies using RFC4737. Ciavattone and Morton [Ciav03] utilised a pre-RFC4737 Internet Draft of packet reordering, using a Poissonian probe, with Reordering Extent characterised in units of time, position and octets. The measurement was carried out on a Tier 1 ISP backbone by AT&T Laboratories. The metrics were used to analyse a 'Blender' – transient routing loops that occur when a router does not have correct forwarding information and sends packets on a path that loop back to that router; a loop that continues until a routing update corrects the problem. This results in short bursts of reordered packets with varying RTTs. For this particular event, they measured 79 reordered packets, with maximum reordering extent 85 and maximum late time offset 64ms, over seven separate sequence discontinuities.

3.4.3 RFC 5236

In June 2008, Jayasumana published RFC 5236, the culmination of several packet reordering metric publications [Pira08]. RFC 5236 argues that the metrics described in RFC 4737 are difficult to implement and interpret and suffer complexity, lack of robustness and issues when attempting to evaluate in real-time. A packet arriving early can be classified as reordered only if receiving packets are not lost. Similarly a late arriving packet may not be reordered if there are earlier copies of the same packet [Pira08]. Two metrics for packet reordering are defined; Reorder Density and Reorder Buffer Density. Reorder Density aims to capture the characteristics of reordering. Reorder Buffer Density aims to evaluate the packet sequence from the recovery perspective. A threshold is set in order to bound when a packet is designated as lost, so

as to bound the number of packets in a trace that are required for comparison and thus simplify storage and calculation requirements.

3.4.3.1 Reorder Density

Reorder Density (RD) is defined as the distribution of displacements of packets from their original positions, normalised with respect to the number of packets. An early packet corresponds to a negative displacement while a late packet corresponds to a positive displacement. RD measured on individual subnets can be combined by convolution, in order to predict the end-to-end reordering of the network.

3.4.3.2 Reorder Buffer Density

Reorder Buffer Density (RBD) is the normalised histogram of the occupancy of a hypothetical buffer, that would allow the recovery from out-of-order delivery of packets. As packets are analysed, they are added to this hypothetical buffer until sufficient other packets arrive, such that all can be released in order to the receiving application. The occupancy of this buffer at any given time is used to describe the reordering.

3.4.3.3 Results

As with RFC4737, there are few results published in the literature that make use of the metrics defined in RFC5236 on a live internet network. Ye [Ye06] used RD to passively measure 5 tcpdump traces of HTTP traffic downloaded to an observation point at Colorado State University in August 2005, each over 2 megabytes in size. The threshold defined, beyond which an early or late packet is deemed to be lost, was set to 25. The histogram plotted of displacement, between -25 and +25, and RD, shows that the majority of packets are grouped between -15 and +10, with RD approximately 0.030. No particular dominating features are present and it is difficult to evaluate from the histogram if such behaviour exhibited by the network is either good or bad.

3.5 Comparison of Techniques

Figure 22 presents a classification taxonomy of the current methodologies and metrics which can be used to measure packet reordering. Previous classifications [Luo05] have classified active measurements as bulk-transport measurements and packet-train measurements; this method will not highlight the potential for adverse Middlebox interaction. Figure 22 classifies active measurements as Control-Plane packets and Data-Plane packets, in order to highlight the potential additional latencies which control-plane measurement techniques might endure. The Hong-Kong Pointer method is classified as a control-plane technique because, although the measurements themselves are performed on data packets, the method relies on less well defined parts of the TCP specification where the processing of these data packets may lead to some uncertainty. The Paxson and Tsinghua techniques are highlighted as techniques which may lead to the most representative techniques for inferring TCP behaviour during reordering. The UDP-based measurements may not fully characterise TCP congestion window-like behaviour when dealing with the effects of packet loss and other cross traffic.

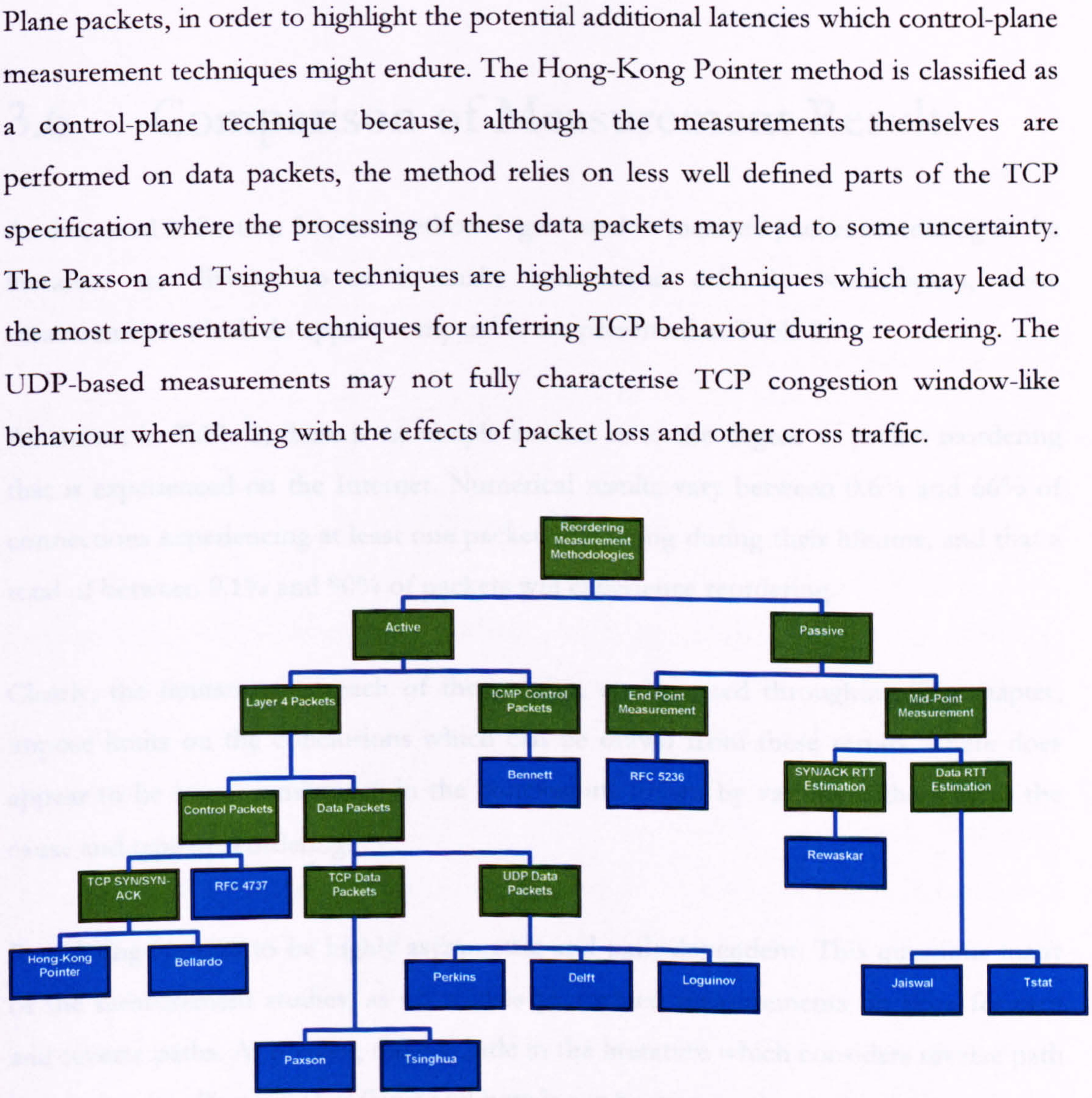

Figure 22 - Packet Reordering Measurement Taxonomy

Inspecting the passive measurements indicates the necessity for all mid-point measurements to obtain an accurate estimation of the sender's calculated RTT, for every TCP connection, throughout the entire lifetime of that connection. Clearly this is a challenging task, which will introduce large degrees of uncertainty in each of the classification algorithms. RFC 5236 avoids this uncertainty, but is primarily an end-point measurement and does not attempt to classify the cause of each packet reordering event. Clearly, some notion of the cause of packet reordering in a network, in addition to a numerical value of the degree and scale of reordering taking place, would be important to network operators and providers.

3.6 Comparison of Measurement Results

As discussed in Section 3.1, the methodologies used to measure packet reordering in the literature are diverse, so as to make comparison difficult. Nevertheless, those measurements which do appear comparable are presented in Table 2.

As shown in Table 2, there is no simple answer as to the degree of packet reordering that is experienced on the Internet. Numerical results vary between 0.6% and 66% of connections experiencing at least one packet reordering during their lifetime, and that a total of between 0.1% and 90% of packets will experience reordering.

Clearly, the limitations of each of the metrics, as discussed throughout this chapter, impose limits on the conclusions which can be drawn from these results. There does appear to be some consistency in the conclusions drawn by various authors as to the cause and type of reordering.

Reordering appears to be highly asymmetric and path dependent. This questions many of the measurement studies, as no studies performed measurements on both forward and reverse paths. At present, there is little in the literature which considers reverse path reordering, its effect on TCP flows and how it can be measured.

Name	% Connections Experiencing Reordering	Amount of Reordering	Conclusions as to the Cause
Paxson	Over 36% and Over 12%	2% and 0.3% forward direction, 0.6 and 0.1% reverse	The large number of data reorderings instead of ack, is due to the cumulative-ack function Highly-site dependent, Highly asymmetric, Highly path dependent
Bennett	Not Comparable	Over 90%	Reordering is a function of local parallelism and network load
Loguinov	9.5%	0.04% of all packets	
Bellardo	Over 15%		Forward path reordering significantly more prevalent than reverse path. Packet sizes (1500 bytes) gives different measurements to 40 bytes Reordering is related to inter-packet gap (with experimental proof)
Tsinghua	5.79%	3.187% of all packets	Strongly site dependent.
Delft	56% and 66%	6% and 5.6%	Highly site dependent, and asymmetric Reordering is a Poisson process which affects bursts at random
Hong Kong Pointer	35% of forward paths, 10% reverse paths	Not Comparable	
Perkins	47%	0.01%	Highly asymmetric The relative frequency of packet reordering increases as the inter-packet arrival time is reduced.
Jaiswal	Between 0.6 and 5.1%	Between 7 and 26%	
Rewaskar	Not Comparable	Between 0.2 and 12.9%	
Tstat	Not Comparable	10.5%	

Table 2 - Comparison of Measurement Results

Reordering appears to be related to the inter-packet gap of packets in a connection. This intuitively appears logical; packets which are spaced close together, will require less delay in order to be moved out of sequence. Packets with larger inter-packet gaps, will have to be significantly delayed in order to be moved behind the next packet in the connection.

Comparison of the measurement results also serves to highlight that, although some metrics claim to emulate TCP-like behaviour in their assessment of reordering, this is often simplified to mean that they evaluate the number of packets appearing more than 3 packets out of sequence. This simplistic notion is unlikely to be sufficient to characterise the complex effects that reordering will have on a sending TCP congestion window and, thus, the performance of the flow. Further assessment of the impact of reordering is considered in Chapter 4.

3.7 Conclusions

Chapter 3 has presented a taxonomy of the current state of the art in packet reordering measurement research and has discussed a number of key issues which affect both passive and active techniques in this field. In addition, a number of measurement studies of packet reordering have been surveyed and those results which are comparable have been presented and discussed.

A number of conclusions can be drawn from the generated taxonomy. There are few active measurement techniques, which perform their measurements by using TCP–like data packets. Section 3.2.1 has discussed the effects of Middleboxes on reordering measurements. The use of such boxes is likely to increase, and their presence is difficult to detect. This will make it difficult to perform measurements, or will interfere with the results that are produced.

The Paxson and Tsinghua measurements are highlighted in the taxonomy as the methodologies which may generate the most representative results of real network traffic. Results obtained using these two techniques, as shown in Table 2, differ in their report of the percentage of connections experiencing reordering by over 30%, but direct comparison of these approaches is not possible, as they were not applied over the same datasets.

Passive measurements have also been classified in the taxonomy and differentiated by their observation point and their ability to generate RTT estimates. The clear benefits of passive monitoring techniques have been discussed in Section 3.3.1, the most important features being their ability to measure real network traffic and the ability to potentially monitor thousands of concurrent flows. Although two of the passive methods discussed enjoy these benefits, the difficulties in generating an accurate mid-point estimate of RTT coupled with the unrealistic expectation that both forward and reverse paths will flow symmetrically, highlight serious limitations with these methods. The resulting measurements obtained by these methods vary by over 20% in their estimation of the

total amount of packets undergoing reordering in the Internet.

The drivers of packet reordering have been discussed during comparison of the measurement results. There is agreement in the literature that reordering appears to be highly asymmetric and path dependent, which is consistent with Bennett's hypothesis that reordering is an effect of local parallelism within nodes. Clearly, as multipath routes, the use of end-to-end wireless technologies such as WiFi and WiMax and local parallelism at multiple layers all increase, increased packet reordering becomes much more likely.

Finally, although many metrics have claimed to emulate TCP-like behaviour in their assessment of packet reordering, their simplistic approach focuses on identifying the number of packets which have been reordered by 3 or more positions. None of the proposed metrics for packet reordering have been correlated with a measure of real TCP Goodput, thus providing a metric that truly is characteristic of receiver behaviour. Indeed, the literature is extremely limited [Laor02] when discussing the actual effect of packet reordering on TCP performance and such assumptions as the behaviour of TCP during reverse path reordering, have been hypothesised [Benn99], but never investigated by experiment or simulation. This would suggest that much of the recent work [Leun07] focussed on simulating TCP and providing patches to the congestion control algorithms to mitigate reordering, is premature and much of the recent work lacks an understanding of the problem.

Chapter 4 continues by investigating the effects of packet reordering, and presents results illustrating the true behaviour of TCP during reordering, which any effective reordering metric should attempt to characterise.

Chapter 4

A Two-Point Passive Packet Reordering Measurement Technique

4.1 Introduction

It is clear that, given recent efforts to design both metrics and methodologies to characterise the degree of reordering along an end-to-end path, packet reordering is a phenomenon which is becoming increasingly important in network performance

measurement and analysis.

It has been proposed [Benn99][Ligh01][Przy05] that packet reordering is a consequence of Network Equipment Manufacturers (NEMs) increasing switch and link level parallelism on the Internet, whilst seeking performance, reliability and economy improvements. It is therefore likely that the degree of packet reordering prevalent on the Internet is on the increase. Although many methods to both measure and mitigate reordering have recently been proposed, limited consideration has been given to measuring and understanding the true drivers of packet reordering, and correlating these measurements with the effect that they actually have on a user's application. It is only through this correlation of measurements that it will be possible to ascertain if packet reordering will affect the users perceived Quality of Service, and then allow for the design of appropriate metrics and mitigations.

In this chapter, an investigation of the drivers of packet reordering is presented; a methodology for emulating and measuring TCP reordering is described, that allows empirical measurement of the true performance of TCP and provides an insight into the complex behaviours of the congestion and retransmission algorithms. A novel two-point packet reordering measurement methodology is presented, followed by a description of algorithms developed which measure and demonstrate the effect that reordering may have on application performance. Results of these measurements are presented and then methods to mitigate the effects of packet reordering are discussed.

4.1.1 Drivers of Packet Reordering

Bennett [Benn99] hypothesised that much of the reordering observed during his experimentation was, not as previously expected due to multi-path routing or broken equipment, but as a result of 'Switch and Link-level parallelism'.

Multi-path routing suggests that there is already an inherent degree of parallelism in existence in the Internet; it is well known that packets travelling between the same source-destination pair may experience 'route flutter' due to transient effects at the intermediate routers along each path. Figure 23 illustrates the additional concepts of

Link-level Parallelism and Switch or Local Parallelism, all of which have been argued to play an even greater role as drivers of future packet sequencing issues.

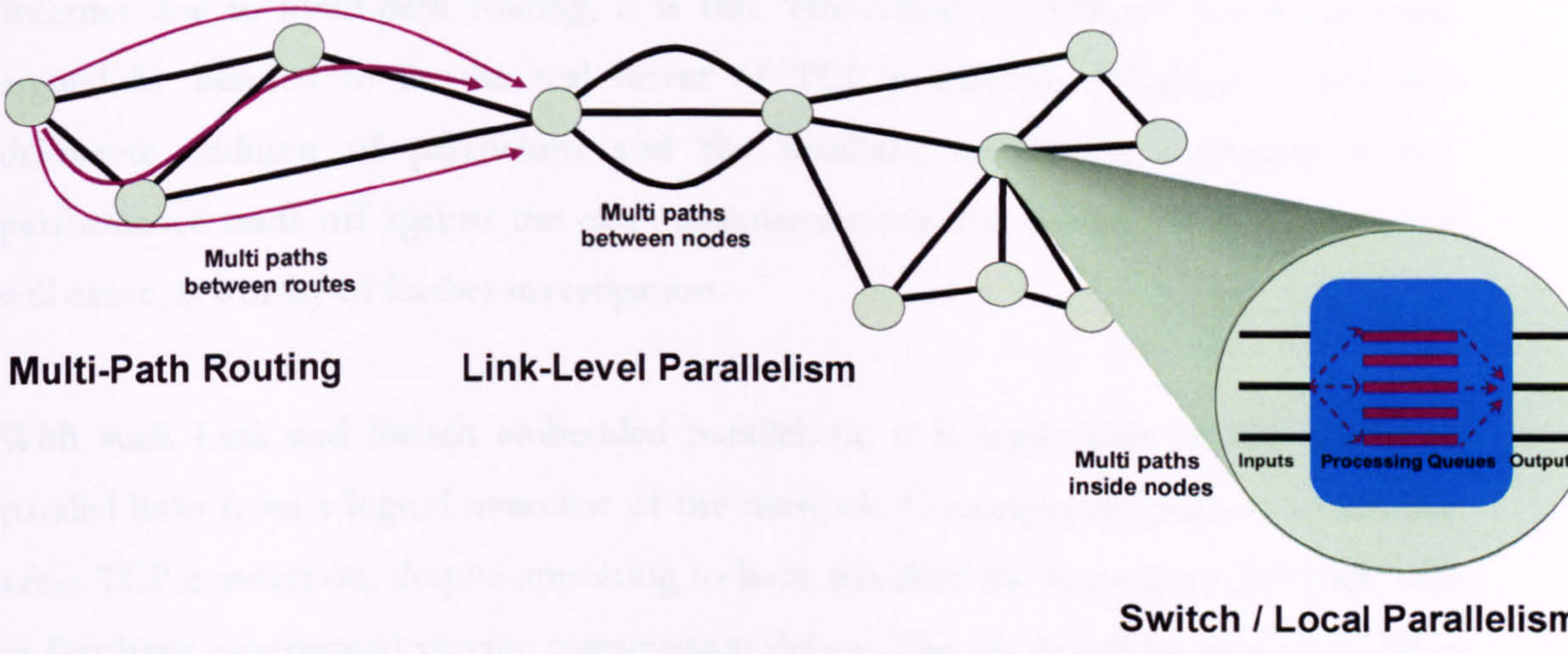

Figure 23 – Link-Level and Switch / Local Parallelism

Link-level Parallelism, includes mechanisms such as link striping and load balancing. It is often more cost effective for network providers to aggregate the throughput of several smaller links, rather than leasing a single high throughput connection. Such parallelism may provide additional benefits in terms of increased flexibility, capacity planning and redundancy. Many NEMs implement L2 link striping on a per-packet basis [Bell02] and, therefore, since queues drain at a constant rate, packet flows with small inter-arrival times are likely to suffer reordering.

Switch Parallelism describes elements of network hardware which allow packets, potentially from the same end-to-end connection, to take different paths within the internal hardware architecture. Switch Parallelism is therefore an intentional design feature and is a result of decisions made by NEMs in order to attain multi-Gigabit switching speeds. In such architectures, several parallel processing queues may have been implemented, potentially at multiple stages throughout the switch, in order to aggregate high overall throughput [Thom02]. Empirical observations have shown that these design compromises can result in packet ordering issues [Ligh01].

Consider a high speed router design that makes use of parallel processors and queues to

switch packets. Upon entering the router, packets are dispersed over these multiple FIFO queues of varying occupancies and it is very difficult to guarantee that sequencing will be maintained. Despite there having always been some degree of parallelism on the Internet due to multi-path routing, it is this 'embedded parallelism' which has been argued by Bennett to be the real driver of 'TCP' reordering. Whether or not this deliberate addition of parallelism and the resultant reordering, represents a fair performance trade-off against the extra retransmissions that current 'TCP' mechanisms will cause, is worthy of further investigation.

With such Link and Switch embedded parallelism, it is impossible to identify these parallel links from a logical overview of the network. Consequently, packets within the same TCP connection, despite appearing to have travelled the same layer 2/3 path, will in fact have experienced varying transmission delays. The result will be increased Delay Variation and, in the worst cases, loss of packet sequence. External identification of the source of packet reordering is therefore difficult; the only indication may be that a particular hop exhibits varying levels of delay. While the majority of routers will count packet losses and export these values through an SNMP MIB, reordering is invisible to routers and is not recorded or measured [Laor02]. Detecting and isolating routers that cause packet reordering [Mizr06], is therefore likely to become an important feature in future internet architectures. It is worthwhile to note that new flow-oriented routing technologies, such as MPLS and L3 VPN, are not necessarily immune to this phenomenon since the parallelism occurs inside the nodes.

Recent papers have suggested that the continued drive to increase router performance for multi-gigabit throughput will lead to increased link and local level parallelism. While CPU computing speed doubles every 18 months, recent trends indicate that network link speeds will double every nine months[Barc07]. This, combined with the ever-increasing sizes of routing tables, will result in an increase of the amount of processing to be performed within routers, thus causing a bottleneck unless highly parallelised architectures are developed. Although approaches have been developed to mitigate switch reordering [Pira06][Kand07], both input sorting and output re-sequencing result in design complexities with severe processing overheads, resulting in increased latency at intermediate nodes.

The effects of Packet Reordering may be cumulative [Laor02]. Across an entire end-to-end path, there may be multiple links and switches with individual degrees of parallelism which, each in turn, contribute to the overall level of packet reordering. These cumulative effects are not currently understood and are difficult to predict. Cumulative reordering at multiple intermediate points may either cancel out the effects, or exacerbate the problem further.

The degree of multiplexing on a backbone switch may also be a factor in packet reordering. Given that the types of switches which implement highly parallelised architectures, such as the Juniper M160 [Juni08], will be core network devices multiplexing many millions of concurrent flows, the probability of an individual flow experiencing packet reordering is low[Laor02]. It is unclear though how future traffic demands will affect the degree of reordering and other work [Benn99] has measured a correlation between packet reordering and switch load.

The various measurement studies presented in Chapter 3 indicate that the only conclusion to which all agree is that packet reordering is highly site or link dependent. Trends indicate that the drivers of packet reordering are likely to become more prevalent in future devices and links across the Internet. At present, it is unclear how much there is, or what its effects will be.

Further exploration of local level parallelism, packet reordering and its impact on the performance of TCP/IP is therefore important for future design considerations of Internet switching equipment architectures.

4.1.2 Measuring the Impact of Reordering

Packet Reordering has an instinctively negative effect on the performance of TCP and therefore the majority of the current metrics and measurement studies have focussed on characterising the movement of packets within a flow, rather than measuring the resulting overall performance of a connection during reordering.

On closer examination, it is clear that many of the assumptions regarding the performance of TCP during reordering have not been tested, but rely on the assumptions of Bennett's seminal paper in 1999.

Bennett hypothesised that, due to the asymmetric nature of the Internet, connections will frequently only experience reordering in one direction, and that the following characteristics will be observed. During forward path reordering, the five effects of unnecessary retransmissions, difficulty growing *cwnd* and *ssthresh*, actual losses being obscured, poor RTT estimation, and reduced efficiency at the receiving TCP will all be prevalent. On the reverse path, a loss of self-clocking and a highly bursty transmission pattern will be observed.

The literature indicates that many of Bennett's assertions have not actually been measured or observed on real networks, nor has there been any correlation between a metric for packet reordering and the resulting effects on application performance. There is, therefore, a need to carry out this study in order to evaluate the relevance of packet reordering metrics, and the accuracy of proposed reordering solutions.

In 'Is TCP Packet Reordering Always Harmful?' [Negl04], Neglia performs a series of NS-2 Simulations that indicate that a limited amount of reordering can actually improve network performance in terms of throughput and delay. In wireless links, where the steady-state dropping probability is independent of link congestion, TCP performance has been observed to improve due to random losses preventing link saturation. In simulations, 8.4% reordering was measured equivalent to 0.18% loss, and 14.2% reordering to 0.69% loss; all were found to result in increased link utilisation and decreased queuing delay. At very high reordering levels, loss was found to outperform reordering, as the unnecessary retransmissions due to reordering were beginning to consume a larger proportion of the link utilisation.

The main reason for Neglia's measured improvement is due to a better operation of the RED algorithm in the simulated mid-point routers. In simulations, a specific equilibrium was found where the sending TCP algorithm and the mid-point RED algorithm were interacting so as to increase overall throughput.

Whilst this is a specific degenerative case, this paper does serve as the only one in the literature to propose that reordering is not necessarily harmful. Neglia notes that the improvement measured is highly dependent on having a uniform reordering (or dropping) probability, that the improvement would not happen for short-lived or reverse-path reordering, and that the amount of 'helpful' reordering depends on the specific network scenario – the same probability may be extremely harmful for a different network configuration.

A further study on the effect of reordering and dropping TCP packets over a slow wireless link is carried out by Nehme[Nehm03]. Using NS-2, a 9.6 kbps GSM link is simulated and buffer exhaustion induced at the Basestation. Only one packet is reordered at a time, and it is found that if reordering occurs at the beginning of a connection, an RTO is more likely to happen, because RTT estimates are not yet inflated by a large cwnd. The effects of reordering are negligible when loss rates are high, as the cwnd is already extremely small.

A study on the effects of packet reordering on the subjective quality of broadband digital television [Spir06], measured that the subjective quality becomes unacceptable when more than 0.12% of packets are reordered on an IPTV network between a video server and a set-top-box. Although a metric similar to the Type-P-Reordered-Ratio-Stream is used, there is no detail as to how packets were reordered or how late a reordered packet would arrive. The effects of packet reordering and subjective video quality are further discussed in Chapter 6.

In 'Shall we worry about Packet Reordering' [Przy05], Przybylski used a UDP traffic generator to send test streams across the GÉANT network [Gean08] between seven hosts across Europe. It was found that the streams of packets of the same size were not affected by reordering. The tolerance of specific applications to reordering will be affected by transmission rate, packet size, transport protocol used and receive buffers. The most vulnerable applications are those that generate small packets followed closely by large packets in a single stream. During experiments, Przybylski regularly observed reordering exceeding three packets over many European links.

Laor [Laor02] investigated the effects of reordering on application performance over a backbone link, where multiple TCP flows were multiplexed onto a single link, to investigate various operating systems, delay values and flow mixes. Laor argues that, with higher throughput in backbone links and higher throughput in technologies such as ADSL, along with larger delay values, there will be a significant increase in the bandwidth-delay product of many TCP connections and, therefore, also in TCP window size.

Using an Agilent QA Robot, Laor was able to reorder one packet at a time (the amount currently supported by a QA-Robot), and measure the throughput of several operating systems, such as Windows 2000, Windows NT, Solaris and Linux. Laor found that applying a small percentage of reordering resulted in a drop in application throughput; between 0 and 1% (depending on the delay) can start to have effects. At around 8-10% reordering, the throughput of affected applications approaches minimum utilisation.

Laor discovered that long flows are affected most by packet reordering, as they have sufficient time to open their congestion window. In fact, packet reordering can be advantageous for short flows, as it causes longer flows to behave in a more TCP-friendly manner.

The main limitation of Laor's experiment is that in each case where packet reordering is applied, it is done so by a fixed number of packet positions. This does not allow for a range of experiments to be carried out, where the timing of such parameters as the RTT can be compared with the additional latency applied to a reordered packet.

There is some discussion that packet reordering will have a measurable impact on high-speed TCP variants [Feng07], which modify the *cwnd* algorithm to be more aggressive, yet still rely on Fast Retransmit and Fast Recovery for packet loss indication. Feng discusses how the current packet reordering models in simulators such as NS-2, are extremely limited in that they cause a block of packets to be reordered at the same time, or make it difficult to relate a specific property of packet reordering to observed TCP performance. Feng finds that when the reordering interval is very small, high-speed TCP

variants suffer significantly from packet reordering even with very small reordering delay times and block sizes.

4.1.3 Fixing Packet Reordering

A significant number of recent publications [Leun07] have suggested methods to mitigate the effects of packet reordering on TCP. As Bellardo comments [Bell02], despite the limitations of existing measurement studies, several researchers have used them to justify modifications to TCP, designed to better tolerate packet reordering. Bellardo comments that all projects would benefit from access to more empirical data, since additional patches to TCP cannot be validated without understanding the prevalence of reordering in the current Internet.

A significant number of the methods to mitigate the effects of packet reordering on TCP, require adjusting the *tcp_reordering* variable in Linux, which controls the number of duplicate Acks allowed before a packet is declared as lost. Lee [Lee02], Blanton [Blan02], Zhang [Zhan03], Ma [Ma04a][Ma04b], Bhandarkar [Bhan03][Bhan06], all propose different methods of dynamically altering the *dupthresh* during the lifetime of a TCP connection. Other methods, such as TCP-PR [Boha03], perform retransmission by timeout, rather than by *dupthresh*, based on RTT estimation in order to trigger Fast Retransmission and Fast Recovery. Reordering Notifying TCP, RN-TCP [Sath05] and Robust TCP, TCP-R, [Sath05b], require the involvement of intermediate routers along the path to distinguish reordering from loss.

It has been argued [Pira06] that little attention has been paid to understanding the nature of reordering and its cause-and-effect relationships. Substantial quantitative results have been produced to show that packet reordering does occur in the Internet but, as others have argued [Leun07], there are few studies which have measured the impact of packet reordering and provided a quantitative assessment of the effects of reordering on a connection.

The effects of reverse path reordering have been particularly poorly investigated in the past; much of Bennett's hypothesis on the impact of reordering remains untested but is

assumed to be correct. It has been argued that there has been a rush within the networking research community [Bell02][Leun07] to provide yet more patches for TCP congestion control algorithms, without actually understanding how TCP will behave during reordering. Although adjusting the *tcp_reordering* variable will allow for fewer spurious fast retransmissions, it is not known if fast retransmissions are the cause of performance degradation, nor what the effects will be for these algorithms for interactive applications or during normal packet loss.

4.1.4 The Motivation for Measuring the Effects of Reordering

The literature clearly indicates that there is a lack of understanding of the effects of packet reordering. The majority of work has focussed on simulations, where the methods to induce reordering are non-standardised, resulting in work which is difficult to evaluate and compare. Simulations often suffer from lack of credibility and there is now a clear trend in network research to move towards the development of testbeds, to allow simulations to be based on credible empirical measurements.

Several papers have explicitly said that there is a lack of appreciation of the cause and effects of reordering, with the majority of Bennett's early assumptions about reordering, not actually having been measured in the real world.

There is, therefore, a need to build a testbed to emulate packet reordering, and to determine the parameters which have the greatest influence on the performance degradation of a TCP stream. A testbed allows emulation of large amounts of network traffic, where all variables can be tightly controlled and the real effects of packet reordering can be instrumented and characterised. Although performance measurements of real world networks, such as those discussed in Chapter 3, are useful to gauge the amount of reordering in occurrence, they do not allow for the controlled environment in which the effects of reordering on individual flows can be measured and analysed.

The investigation presented in this Chapter details the construction and methodology of

an experimental measure of the effects of packet reordering on a typical FTP application. Previous measurements of TCP Reordering have been based on simplistic simulation models [Blan02], or on large scale production networks [Laor02][Jais07][Bell02]. The only measurement study of reordering has been based on moving packets by position, rather than in time [Laor02]. They do not allow for measurement of TCP algorithms in a controlled environment, where all nodes can be instrumented, and aspects such as the cause of retransmissions (for example Fast Retransmit requests with respect to RTT timeout), can be investigated.

The use of FTP provides a simple method for illustrating TCP behaviour during single long-lived connections and is not intended to be representative of all Internet traffic. However, such an application provides an excellent first approximation to gain a better understanding of the protocol's characteristics under a variety of conditions. This Chapter discusses the effects of reordering as perceived by the user and illustrates the degree of service quality degradation that a user could expect in situations of severe reordering.

This chapter discusses work which was presented at the Second International Workshop on Internet Packets Dynamics, IPDy 2007, and has been invited to appear in the IARIA International Journal on Advances in Internet Technology.

4.2 Experimental Methodology

From the previous discussion, it is clear that the effects of packet reordering could be cumulative across an end-to-end path as a flow traverses several reordering-inducing components and links. Figure 24 illustrates a network equivalence diagram upon which the testbed network is designed.

4.2.1 Core Transit Network Reordering Equivalence

It is assumed that a flow is traversing from the cloud on the left to the cloud on the right, passing through edge devices such as low capacity routers, through a core transit network, towards the destination. It is the cumulative effects of reordering within the core transit network that the testbed will emulate.

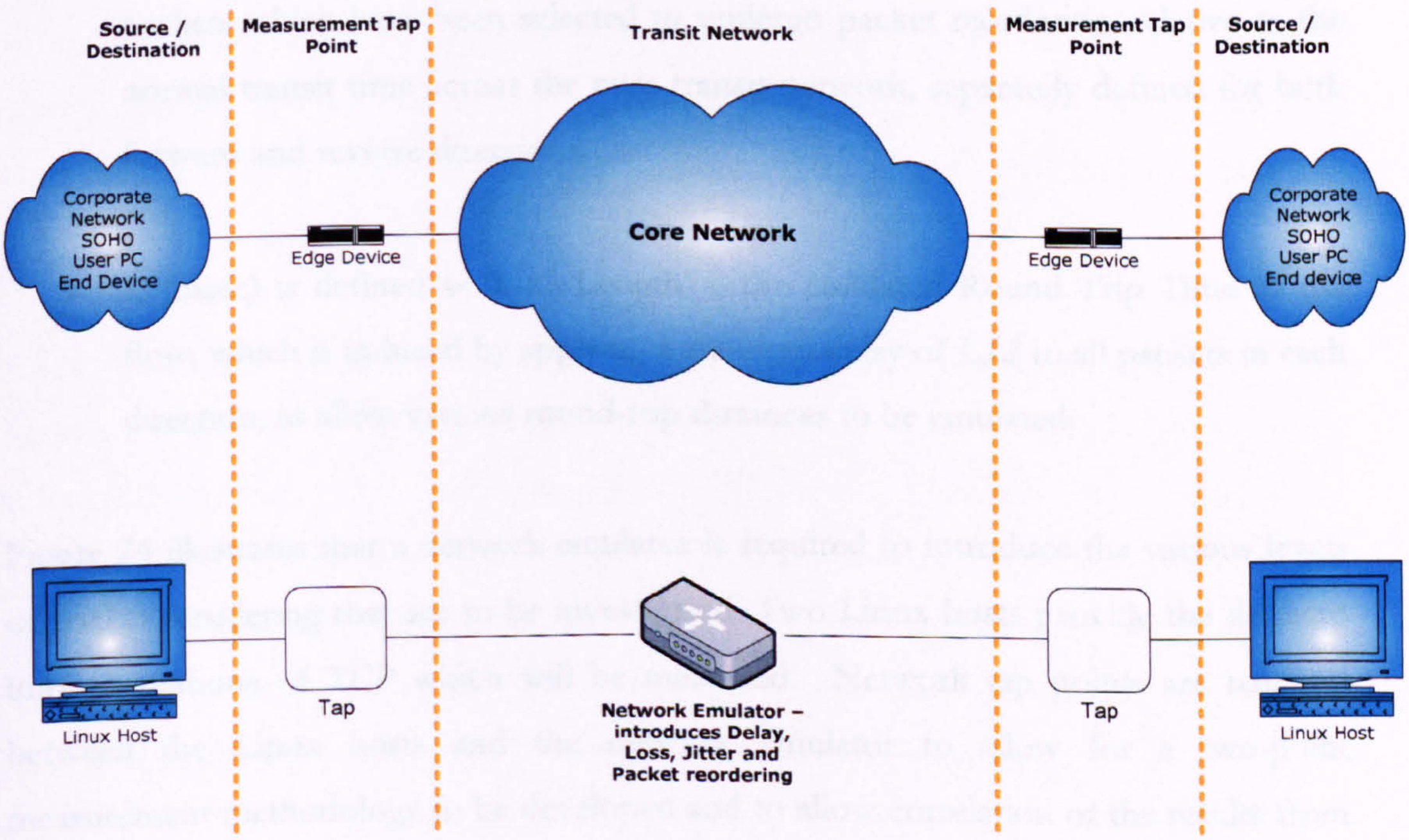

Figure 24 - Network Equivalence Diagram

The purpose of this study is to emulate these cumulative effects of reordering on a single TCP stream and measure the resulting performance characteristics. Previous

studies to measure specific router architectures [Ligh01] do not allow for consideration of multiple reordering-inducing components, and thus do not accommodate the range of reordering measurements supported by this testbed.

It is worth noting further, that the cumulative effects of reordering could be positive or negative at each device. That is to say that packet reordering occurring at a first device, may be 'undone' by a second device. This phenomenon and its effects on a single TCP stream are not well understood or documented.

The problem of reordering can therefore be refined into the following components.

- P_r is defined as 'Reordering Probability' – the percentage probability of a packet undergoing reordering as it traverses the core transit network, separately defined for both forward and reverse directions.

- d (msec) is defined as 'Reordering Delay' - the additional time delay applied to packets which have been selected to undergo packet reordering, relative to the normal transit time across the core transit network, separately defined for both forward and reverse directions.

- L (msec) is defined as 'Line Length' – the emulated Round Trip Time of the flow, which is induced by applying a standard delay of $L/2$ to all packets in each direction, to allow various round-trip distances to be emulated.

Figure 24 illustrates that a network emulator is required to introduce the various levels of packet reordering that are to be investigated. Two Linux hosts provide the de-facto implementations of TCP which will be measured. Network tap points are required between the Linux hosts and the network emulator to allow for a two-point measurement methodology to be developed and to allow correlation of the results from both sides of the network, before and after reordering has been induced.

4.2.2 An Open Extensible Router

Router NEMs are moving towards building products based on standardised Linux kernels[Info08], thus allowing multiple services such as firewalls, border controllers and deep packet inspection, to run on a single 'Network Virtualised' device.

At present, commercial routers are difficult to modify or to extend due to specialised hardware and some proprietary code. There is therefore a requirement to find an open source extensible software router, which can run on standardised Intel server hardware, to allow the programmability required for network emulation of packet reordering.

4.2.2.1 The Click Modular Router

The Click Modular Router Project [Clic08] at M.I.T. and U.C.L.A, aims to develop a software architecture for building flexible and configurable routers. A Click software router [Kohl00] employs a simple declarative language to describe a router's configuration, allowing full control of packet processing within the router, such as packet modification, queuing, dropping and scheduling, and providing the flexibility required for the testbed packet reordering experimentation.

Click routers are assembled from packet processing modules called 'Elements'. An Element represents a basic unit of processing that would occur inside any router – example Elements include decrementing a TTL counter, checking the value of an IP checksum or counting packets as they pass a point in the configuration. Click schedules the router's CPU with a task queue, one element at a time. Each task in the task queue is an Element requiring access to CPU time, and so each Click Element represents both Click's unit of packet processing as well as its unit of CPU scheduling.

Each Element belongs to one Element Class, which specifies the piece of C++ code which should be executed when a packet is passed to that Element. Each Element specifies a number of ports; packets are passed from the output port of one Element to the input port of another.

A user can then define a configuration for the router, by using the Click declarative language to describe a directed graph, with Elements at the vertices, and packets flowing across the edges of the graph. Declarations are used to instantiate Elements, while Connections described how each of the Elements should be connected together. Depending on the endpoint ports of each graph edge, a particular Connection may be push or pull. Push processing is used when a packet arrives from a device, such as a packet arriving and being loading into a Queue Element. Pull connections are used when an Element is controlling the time of packet processing, such as a Scheduling Element loading packets from the Queue Element.

4.2.2.2 Installing a Click Router

Once a Configuration has been described, the router configuration is run in the context of a Linux driver, either at user level or in the Linux kernel. The user level driver operates on the Linux networking stack using Berkeley packet filters, whereas the in-kernel driver offers much increased performance. The kernel thread runs the router driver, which loops over the Click task queue and runs each task in turn. The Click language file is passed to the kernel driver, checked for errors, each Element is initialized, and the router is put online.

The idea of modular routers is not new. However, finding the right level of abstraction to achieve high performance and flexibility is difficult. If the building blocks are very fine-grained, with expensive packet transfers between blocks, then performance will be poor. Creating monolithic blocks can reclaim this performance, but at the expense of flexibility. Click has explored a particular region within this spectrum and has demonstrated impressive packet forwarding speeds whilst retaining a degree of modularity. On conventional PC hardware, a Click router has been measured to achieve a maximum loss-free forwarding rate of 333,000 64-byte packets per second [Kohl00]. An otherwise idle Click IP router has been measured to forward 64-byte packets with a one-way latency of 29 microseconds. Minimum sized packets stress the router greater than larger packets, as the CPU and other processing resources are consumed in proportion to the number of packets forwarded, not in proportion to bandwidth. Each

Element has a processing cost associated with it, although even substantial Elements such as CheckIPHeader and DecIPTTL have been measured to have very low processing times of 457 nanoseconds and 119 nanoseconds respectively.

For the experimentation carried out on the testbed, it will be shown that any extra processing latency induced by Click is negligible in proportion to the reordering delays and round trip times induced.

4.2.2.3 ElementClass 'Reorder'

A new Element and corresponding ElementClass were created called 'Reorder', to allow for the selective delay and reordering of packets passing through the Click router. Reorder uses two queues to simulate packet reordering. Packets traversing the first queue will pass with no additional delay, whereas packets randomly selected to traverse the second queue will have an additional delay applied to them.

The Reorder Element is passed two variables from the Click configuration language. P_r, the Reordering Probability, is a value between 0 and 1, where 0.2 is equivalent to 20% of the packets being selected for reordering. d (msec) Reordering Delay, is the delay applied to the first queue within the Reorder Element. The random is self-weighting, so that every packet in the direction of interest has equal opportunity of being selected to be reordered.

4.2.2.4 Click Language Configuration

A Click language script was implemented to make use of the new Reorder ElementClass as illustrated in Figure 25. The figure illustrates two separate directed graphs; reordering occurs asymmetrically, both forward and reverse paths must be treated independently in the testbed and so the Click language script must instantiate two separate graphs.

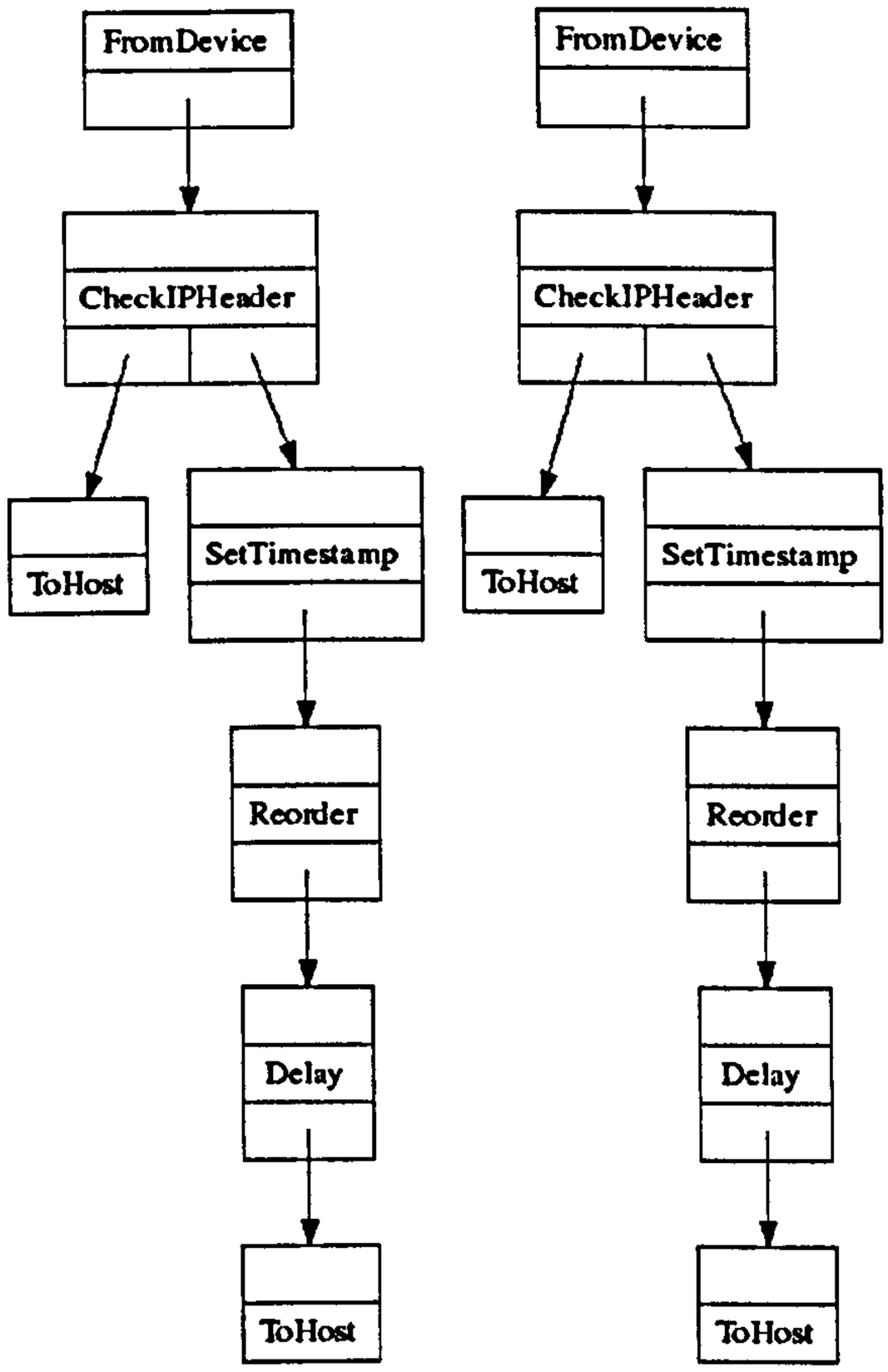

Figure 25 - Click Element Configuration

Upon arriving at a network interface card, a packet is pushed from the FromDevice Element to the CheckIPHeader Element. In this particular configuration, CheckIPHeader initially checks if a packet is IP or not; non-IP packets are passed to the ToHost Element and they are permitted to traverse the test network freely; IP packets are checked for valid IP length, address and checksum fields, and if correct, are passed to SetTimestamp. SetTimestamp records locally the time at which each packet arrived at the Click router. The packet is then pushed to the Reorder Element and processed as discussed in the previous section. Finally, each packet is pushed to the Delay element, where the additional delay $L/2$ (msec) is applied to each packet, thus allowing emulation of multiple Round Trip Times. $L/2$ is applied rather than L, to emulate the router as a mid-point device, thus ensuring that, for non-reordered packets, both forward and reverse paths induce the same overall path delay.

4.2.3 Gigabit Network Testbed

A network testbed was constructed as illustrated in Figure 26 using fibre-based gigabit Ethernet components. The use of full-duplex fibre avoids local Ethernet contention or other transient network effects which may influence the measurements made.

The testbed consists of a number of servers and networking components, which were rack mounted and configured in a single unit.

- Quoyloo is an HP NetServer LPr, installed with two Netgear GA620 fibre-based Gigabit networking cards. Quoyloo was installed with Fedora Linux 2.4.18, with a recompiled Kernel to support Click kernel extensions and to allow Click to run as a Kernel module for improved performance. A version of the Click script described in Section 4.2.2.4 was installed on Quoyloo for each experiment.

- Raasay and Stroma are two HP Kayak XU workstations, each with Netgear GA620 fibre Ethernet cards, and Fedora Linux 2.4 installed. During experimentation, Raasay performed 10 Megabyte FTP upload sessions to Stroma.

- Yell is a Datum TymServe GPS NTP Server, which provided synchronisation between all the machines on the testbed. Yell itself was connected via coaxial cable to a GPS Antenna located on the roof of the building.

- Isay is a Dell Poweredge server, used for offline processing of results, as discussed in Section 4.2.7.

- Missouri was an additional Netserver LPr, which acted as the testbed commander, providing real time indication of the progress of experiments.

- Hoy was a HP Netserver LPr with two Netgear GA620 cards, used to probe the

Gigabit network using the two Agilent Passive Optical Taps connected at the two points shown in Figure 26.

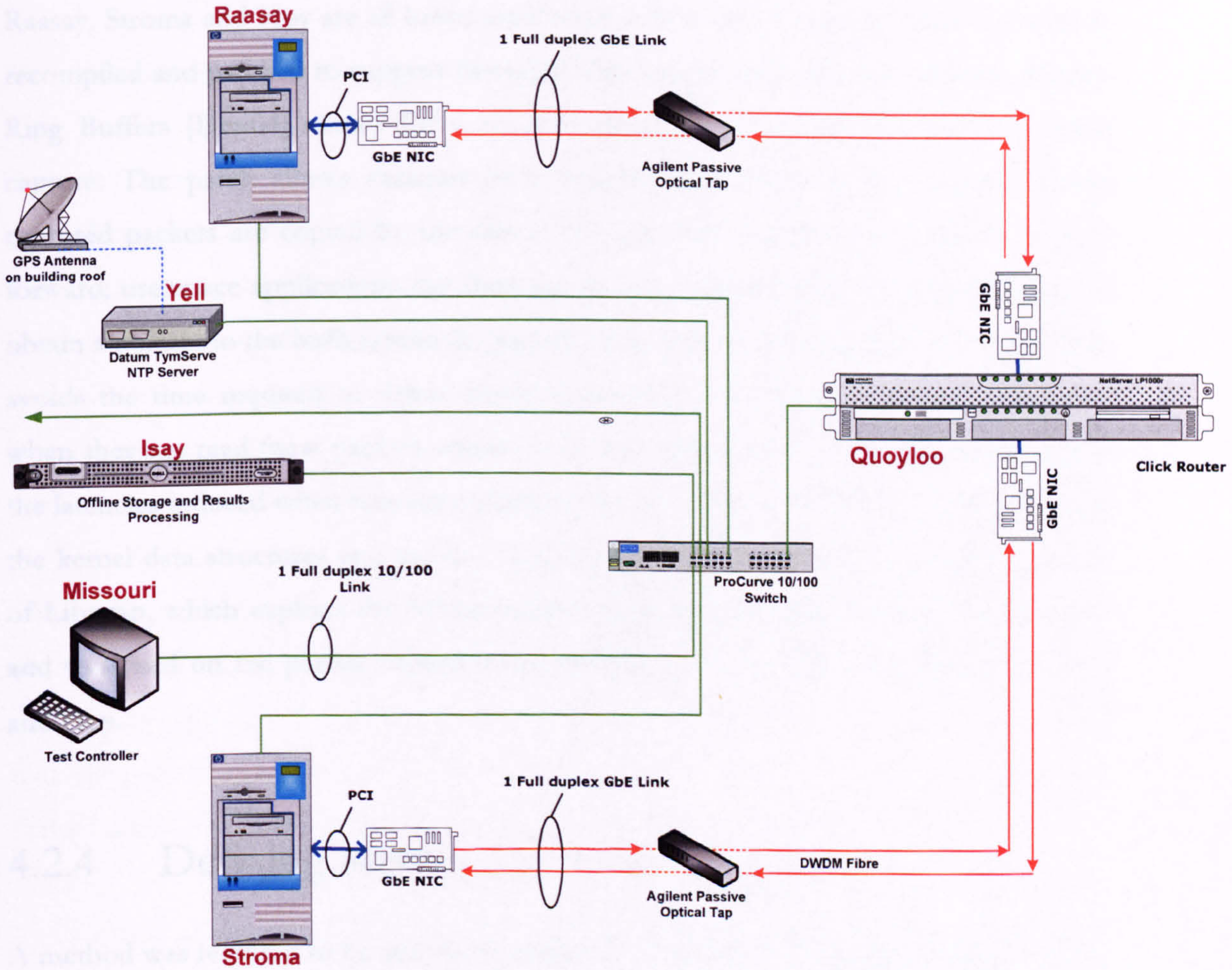

Figure 26 - Gigabit Network Testbed

Figure 26 indicates that, in addition to the Full duplex Gigabit Ethernet link between Raasay, Quoyloo and Stroma, each element in the testbed was connected via a 10/100 electrical Ethernet switch to allow for setup and control of each experiment.

During experimentation on the Gigabit Ethernet testbed, the network maximum transfer unit on each workstation was set to 1500 bytes, thereby disabling Gigabit Jumbo Frame support, and allowing simulation of the standard packet size observed in the Internet[Medi05].

4.2.3.1 MMap Extensions

Raasay, Stroma and Hoy are all based on Fedora Linux 2.4.18 kernels, which have been recompiled and patched to support Memory Map Extensions. The use of Linux Kernel Ring Buffers [Deri04] allows for a much improved performance in passive packet capture. The patch allows creation of a circular buffer for packet capture, where captured packets are copied by the driver into the ring and the write pointer moved forward; userspace applications can then access the circular buffer by calling MMap to obtain a pointer to the buffer, read the packet, then move the read pointer forward. This avoids the time required to delete captured packets from Kernel memory structures when they are read (new packets simply overwrite old packets in the ring), and avoids the latencies induced when moving a packet from the adaptor to the user space through the kernel data structures and queues. Libpcap-MMap [Wood08] is a modified version of Libpcap, which exploits the MMap system calls for passing packets into user space and was used on the packet capture code, developed and installed on Raasay, Stroma and Hoy.

4.2.4 Defining Metrics for Packet Reordering

A method was required to be able to measure the amount of packet reordering that each packet had undergone, when observed at either of the two tap points in the testbed. A C++ packet probe was developed using Libpcap-MMap, which was designed with the ability to record the arrival sequence of packets as observed at the probe, and additionally monitor and log all packets in text files, of both forward and reverse directions, for later parsing and analysis.

In order to calculate the distance by which a packet has been reordered, a method is required to differentiate the order in which the packets were sent, compared with the order in which they were observed by the probe. The IPID field within the IP packet header was used to provide a sending sequence number. IP is designed to support fragmentation and re-assembly of packets when traversing various layers of the OSI that may have differing maximum transmission unit (MTU) sizes. The IPID field is inserted

by the sender into each packet as it is transmitted. The IPID is then copied into each segment at fragmentation and can then be used by routers for re-assembly of the datagrams back into the originally transmitted packet.

The traditional method of implementing IPID is using a simple counter incremented on a per packet basis on each Ethernet interface. On the testbed setup only one concurrent flow was active on each gigabit interface; each of the hosts on the test network were controlled using separate interfaces on the 100baseT electrical network. Using this setup the IPID increased by only one on each packet sent on each of the test streams. The IPID therefore provided a simple and effective 'packet counter' from both the sender and receiver and provided a means of identifying the outbound sequence in which the packets had been transmitted. Each flow's initial packet's IPID was used to normalise this 'sending sequence' number as recorded by the packet probes.

The packet probes were deployed either side of the Click router, depending on their direction, forward or reverse, to capture packets after they had been reordered. As each packet was captured, the sniffer would record the arrival order of packets within the flow using an integer counter, which would indicate the arrival sequence of the packets at their destinations. This was the 'receiving sequence' number.

Comparison of the 'receiving sequence' number with the 'sending sequence' number provided a very effective real-time method of measuring the "Absolute Reordering" of the packet. Previous methods have relied upon calculating the next IP sequence number based on the previous sequence number plus payload length. This method becomes extremely complex under high degrees of reordering and loss, as it can be difficult to calculate the next expected sequence numbers and the distance a packet has been reordered.

Therefore, a simple metric can be defined that can be applied at probes on either side of the Click router, to define if packet reordering has occurred and the severity of the reordering observed.

For packet P_i, where i is the index position of that packet in the arrival stream of packets, j is the observed IPID of the packet, and k is the first IPID observed in that stream of packets, the packet can be defined as out of order if Equation 1 is satisfied.

$$\forall i, j, k: \; i < (j-k) \; or \; i > (j-k) \; \Rightarrow \; P_i \text{ is out of order}$$

Equation 1 - Reordered Packet Metric

For a packet P_i which has been defined as out of order, the extent of the reordering can be defined by Equation 2, where the result is the number of packet positions that a Reordered packet has arrived, early or late.

$$\forall i, j, k : if \; P_i \Rightarrow \text{is out of order}, i - (j - k) \Rightarrow \text{Reordering Extent}$$

Equation 2 - Reordering Extent Metric

When implementing these metrics, it is important to note that, as discussed in Chapter 2, the IPID field in the IP header is limited to two bytes. Therefore, the counter will recycle after 65,535 iterations. Code implementation of Equations 1 and 2 requires a check in order to test if counter recycling has occurred, and if so, to normalise the value of j to zero on the packet at which this occurs.

4.2.5 Packet Probe 'Out of Sequence' Code

The C++ packet probe operated as illustrated in Figure 27. A filter is specified at run time; in this case, the filter was to capture all packets between IP addresses 10.0.0.2 (Raasay) and 10.0.0.6 (Stroma), which did not appear on port 21. Port 21 is the control port for FTP sessions and, through the use of Active rather than Passive FTP, it is known in advance that a separate data session will be created from 10.0.0.2 to 10.0.0.6, which forms the data transfer and the basis for the measurement.

Figure 27 shows that as each packet which matches the filter criteria arrives, it is passed to a callback function for processing. This continues until the code receives a SIGINT signal from the Linux kernel, indicating that the packet capture should cease, and the results should be flushed to disk.

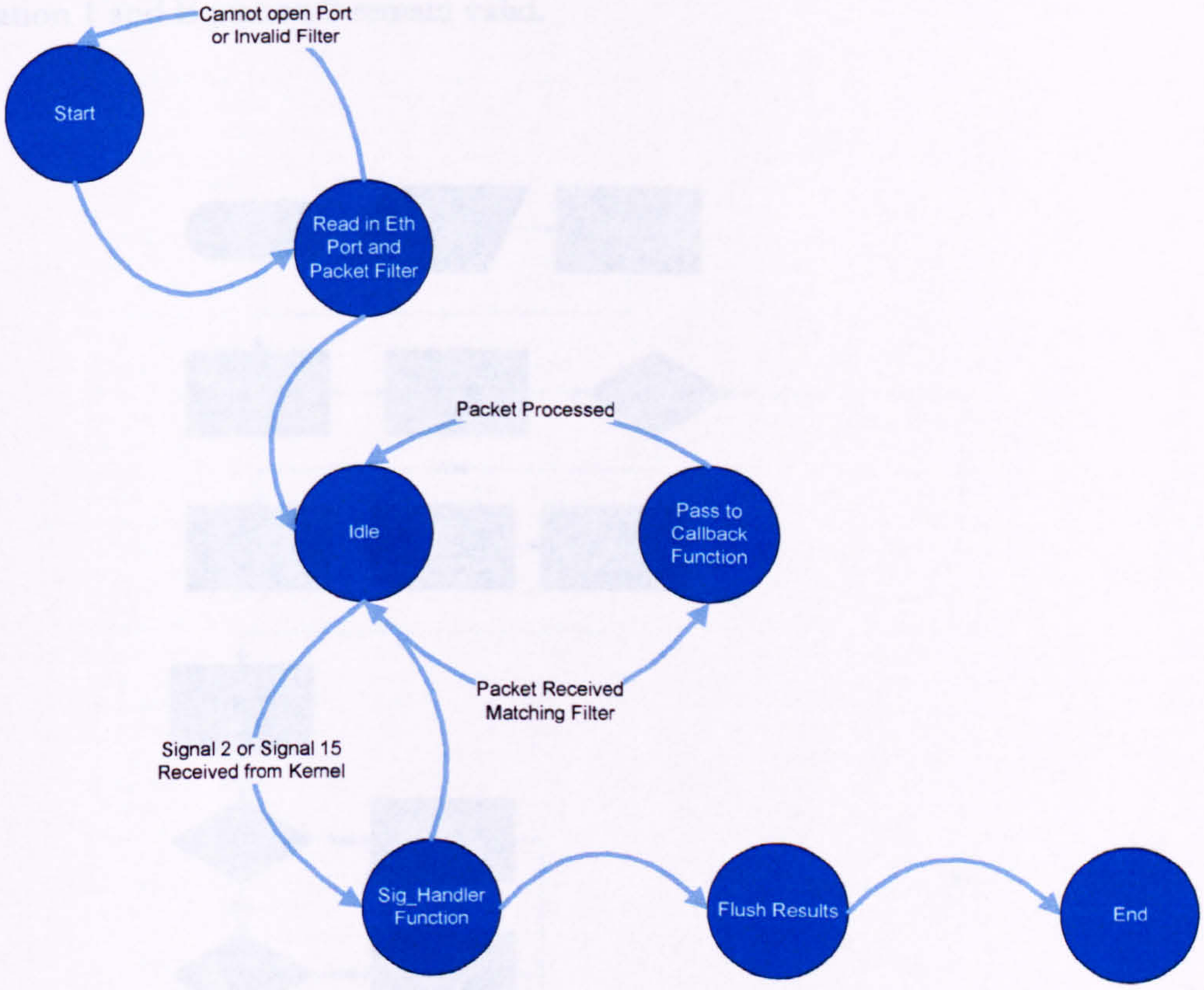

Figure 27 – Out of Sequence FSM

Figure 28 illustrates the algorithm implemented by the called function as each new packet arrives at the probe. The packet is read in network byte order and byte masks, in the form of structs, are applied in order to delimit the Ethernet Header and IP Header. If the packet is not IPv4, as indicated in the Version field, it is silently dropped.

The 4-tuple of IP source and destination addresses and ports, are then used to define a TCP connection. If the header has the TCP SYN flag set, the current IPID is used as the normaliser applied to successive packet's IPID in order to calculate the 'Sending Sequence' of the packets.

The Initial IPID value may be reset during the lifetime of a connection if it is detected that the IPID value has recycled from 65535 to 0. In this instance, it is assumed that the IPID counter on the sending Network Interface Card has recycled and therefore the Out of Sequence code follows suit. This occurrence will not have any adverse effect on the calculation of packet reordering, as the normalising IPID will reset to zero, and Equation 1 and Equation 2 remain valid.

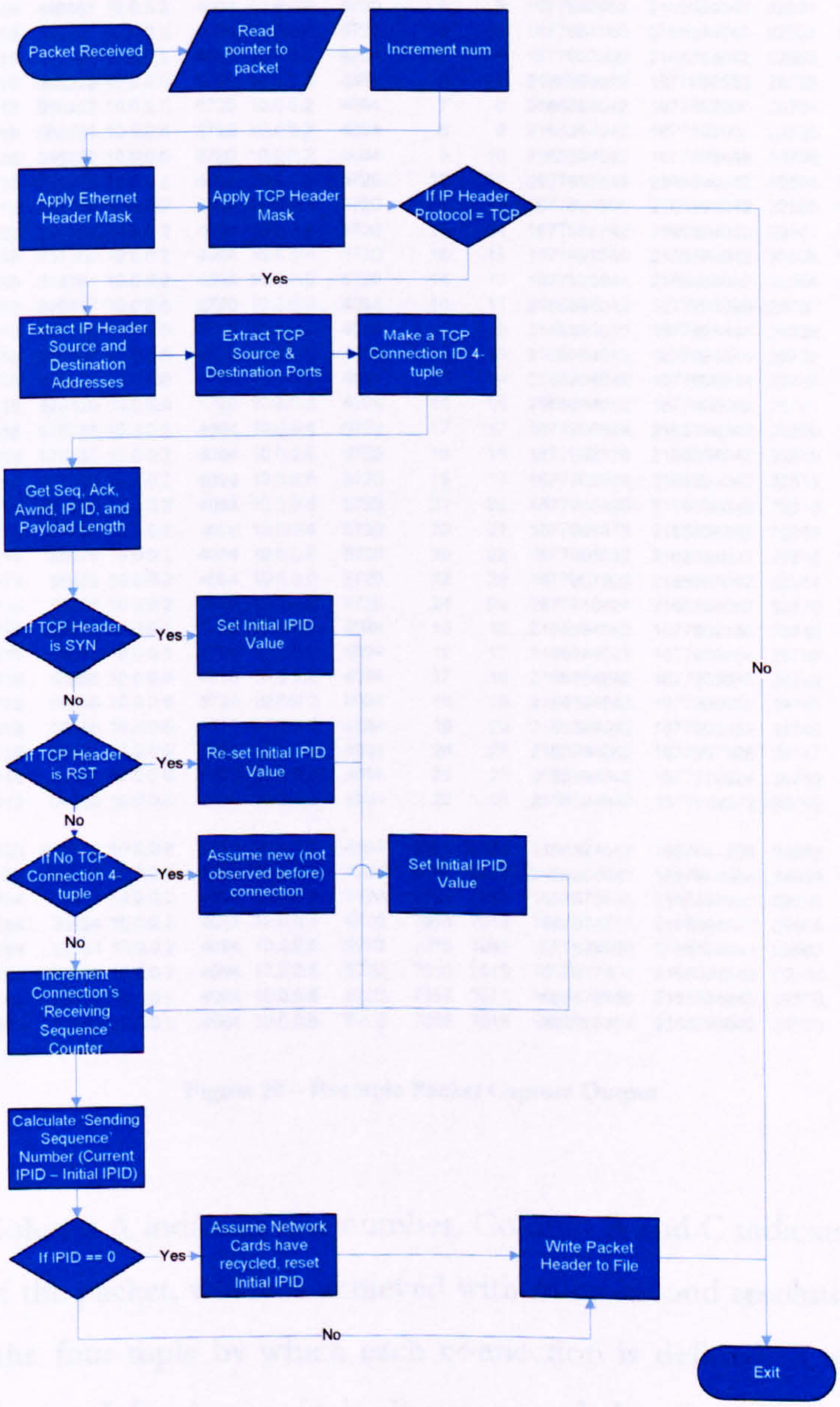

Figure 28 - OOS Packet Callback Algorithm

An example capture output from the probe software is illustrated in Figure 29.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1218632117	649377	10.0.0.2	4094	10.0.0.6	5720	0	0	1677880416	0	22492	0	SYN	5840
2	1218632117	649441	10.0.0.6	5720	10.0.0.2	4094	0	0	2165394042	1677880416	0	0	SYN	5792
3	1218632117	799834	10.0.0.2	4094	10.0.0.6	5720	1	1	1677880416	2165394042	22493	0		5840
4	1218632118	18821	10.0.0.2	4094	10.0.0.6	5720	2	2	1677880416	2165394042	22494	1448		5840
5	1218632118	19273	10.0.0.2	4094	10.0.0.6	5720	3	3	1677881864	2165394042	22495	1448		5840
6	1218632118	106245	10.0.0.6	5720	10.0.0.2	4094	28727	1	2165394042	1677881864	28727	0		8688
7	1218632118	106277	10.0.0.6	5720	10.0.0.2	4094	1	2	2165394042	1677883312	28728	0		11584
8	1218632118	256771	10.0.0.2	4094	10.0.0.6	5720	4	4	1677883312	2165394042	22496	1448		5840
9	1218632118	257169	10.0.0.2	4094	10.0.0.6	5720	6	5	1677886208	2165394042	22498	1448		5840
10	1218632118	257170	10.0.0.2	4094	10.0.0.6	5720	7	6	1677887656	2165394042	22499	1448		5840
11	1218632118	334777	10.0.0.2	4094	10.0.0.6	5720	5	7	1677884760	2165394042	22497	1448		5840
12	1218632118	346230	10.0.0.6	5720	10.0.0.2	4094	2	3	2165394042	1677884760	28729	0		14480
13	1218632118	346257	10.0.0.6	5720	10.0.0.2	4094	3	4	2165394042	1677884760	28730	0		14480
14	1218632118	346270	10.0.0.6	5720	10.0.0.2	4094	4	5	2165394042	1677884760	28731	0		14480
15	1218632118	346300	10.0.0.6	5720	10.0.0.2	4094	5	6	2165394042	1677889104	28732	0		17376
16	1218632118	496600	10.0.0.2	4094	10.0.0.6	5720	8	8	1677889104	2165394042	22500	1448		5840
17	1218632118	496965	10.0.0.2	4094	10.0.0.6	5720	9	9	1677890552	2165394042	22501	1448		5840
18	1218632118	496967	10.0.0.2	4094	10.0.0.6	5720	10	10	1677884760	2165394042	22502	1448		5840
19	1218632118	497365	10.0.0.2	4094	10.0.0.6	5720	11	11	1677892000	2165394042	22503	1448		5840
20	1218632118	586223	10.0.0.6	5720	10.0.0.2	4094	6	7	2165394042	1677890552	28733	0		20272
21	1218632118	586252	10.0.0.6	5720	10.0.0.2	4094	7	8	2165394042	1677892000	28734	0		23168
22	1218632118	586266	10.0.0.6	5720	10.0.0.2	4094	8	9	2165394042	1677892000	28735	0		23168
23	1218632118	586290	10.0.0.6	5720	10.0.0.2	4094	9	10	2165394042	1677893448	28736	0		26064
24	1218632118	736775	10.0.0.2	4094	10.0.0.6	5720	12	12	1677893448	2165394042	22504	1448		5840
25	1218632118	737158	10.0.0.2	4094	10.0.0.6	5720	13	13	1677894896	2165394042	22505	1448		5840
26	1218632118	737160	10.0.0.2	4094	10.0.0.6	5720	15	14	1677897792	2165394042	22507	1448		5840
27	1218632118	737162	10.0.0.2	4094	10.0.0.6	5720	16	15	1677899240	2165394042	22508	1448		5840
28	1218632118	814781	10.0.0.2	4094	10.0.0.6	5720	14	16	1677896344	2165394042	22506	1448		5840
29	1218632118	826217	10.0.0.6	5720	10.0.0.2	4094	10	11	2165394042	1677894896	28737	0		28960
30	1218632118	826245	10.0.0.6	5720	10.0.0.2	4094	11	12	2165394042	1677896344	28738	0		31856
31	1218632118	826262	10.0.0.6	5720	10.0.0.2	4094	12	13	2165394042	1677896344	28739	0		31856
32	1218632118	826274	10.0.0.6	5720	10.0.0.2	4094	13	14	2165394042	1677896344	28740	0		31856
33	1218632118	826320	10.0.0.6	5720	10.0.0.2	4094	14	15	2165394042	1677900688	28741	0		34752
34	1218632118	976731	10.0.0.2	4094	10.0.0.6	5720	17	17	1677900688	2165394042	22509	1448		5840
35	1218632118	977055	10.0.0.2	4094	10.0.0.6	5720	18	18	1677902136	2165394042	22510	1448		5840
36	1218632118	977056	10.0.0.2	4094	10.0.0.6	5720	19	19	1677903584	2165394042	22511	1448		5840
37	1218632118	977058	10.0.0.2	4094	10.0.0.6	5720	21	20	1677906480	2165394042	22513	1448		5840
38	1218632118	977454	10.0.0.2	4094	10.0.0.6	5720	23	21	1677909376	2165394042	22515	1448		5840
39	1218632119	55079	10.0.0.2	4094	10.0.0.6	5720	20	22	1677905032	2165394042	22512	1448		5840
40	1218632119	55453	10.0.0.2	4094	10.0.0.6	5720	22	23	1677907928	2165394042	22514	1448		5840
41	1218632119	55455	10.0.0.2	4094	10.0.0.6	5720	24	24	1677910824	2165394042	22516	1448		5840
42	1218632119	66215	10.0.0.6	5720	10.0.0.2	4094	15	16	2165394042	1677902136	28742	0		37648
43	1218632119	66245	10.0.0.6	5720	10.0.0.2	4094	16	17	2165394042	1677903584	28743	0		40544
44	1218632119	66268	10.0.0.6	5720	10.0.0.2	4094	17	18	2165394042	1677905032	28744	0		43440
45	1218632119	66285	10.0.0.6	5720	10.0.0.2	4094	18	19	2165394042	1677905032	28745	0		43440
46	1218632119	66298	10.0.0.6	5720	10.0.0.2	4094	19	20	2165394042	1677905032	28746	0		43440
47	1218632119	66327	10.0.0.6	5720	10.0.0.2	4094	20	21	2165394042	1677907928	28747	0		46336
48	1218632119	66340	10.0.0.6	5720	10.0.0.2	4094	21	22	2165394042	1677910824	28748	0		49232
49	1218632119	66353	10.0.0.6	5720	10.0.0.2	4094	22	23	2165394042	1677912272	28749	0		52128
13176	1218632183	885123	10.0.0.6	5720	10.0.0.2	4094	6161	6162	2165394042	1687841208	34888	0		65160
13177	1218632183	885141	10.0.0.6	5720	10.0.0.2	4094	6162	6163	2165394042	1687873064	34889	0		65160
13178	1218632184	35411	10.0.0.2	4094	10.0.0.6	5720	7013	7013	1687873064	2165394042	29505	1448		5840
13179	1218632184	35424	10.0.0.2	4094	10.0.0.6	5720	7014	7014	1687874512	2165394042	29506	1448		5840
13180	1218632184	35803	10.0.0.2	4094	10.0.0.6	5720	7015	7015	1687875960	2165394042	29507	1448		5840
13181	1218632184	35805	10.0.0.2	4094	10.0.0.6	5720	7016	7016	1687877408	2165394042	29508	1448		5840
13182	1218632184	35807	10.0.0.2	4094	10.0.0.6	5720	7017	7017	1687878856	2165394042	29509	1448		5840
13183	1218632184	35809	10.0.0.2	4094	10.0.0.6	5720	7018	7018	1687880304	2165394042	29510	112	FIN	5840

Figure 29 – Example Packet Capture Output

In Figure 29, Column A indicates line number. Column B and C indicate the timestamp of the arrival of the packet, which is achieved with microsecond resolution. Columns D to G indicate the four-tuple by which each connection is defined. It should be noted that a connection is defined as a single direction, and therefore the packet probe will treat data and ack paths separately for the purposes of computing the reordering.

Column I indicates the 'Receiving Sequence' number, which can be seen to increment monotonically for each of the 10.0.0.2 to 10.0.0.6 and 10.0.0.6 to 10.0.0.2 connections. Column H illustrates the 'Sending Sequence' number which, in this particular example, with little reordering, is seen on the forward path to match the 'Receiving Sequence' until row nine, where the packet with Sending Sequence 5 is reordered by two positions and arrives late at position 7. Column J indicates the Seq number, column K indicates the Ack, Column L the IPID and Column M the payload size. Finally, set flags are indicated in Column N, and Column O indicates the *rwnd* advertised by the packet.

4.2.6 Automated Distributed Measurement System

In order to perform the substantial number of experiments and measurements that would be required to accurately measure the effects of packet reordering on the testbed, a distributed system of code components and actions was implemented in order to parse test plans, control and schedule measurement runs, configure the programmable router and initiate the FTP sessions to be measured. These code components were written in a variety of languages and controlled from the test commander Missouri.

Each experiment can be defined by 6 variables:

- The Reordering Probability in both the forward and reverse directions.
- The Reordering Delay in both the forward and reverse directions.
- The Line Length emulated round-trip distance of the experiment.
- Finally, the iteration number of the experiment.

Each experiment consisted of a 10 Megabyte FTP transfer, typically each experiment capturing approximately 13000 packets, equating to a 1.9 Megabyte text capture file. In order to manage the large amounts of data that were created, each capture file was named with the format *Forward%-ForwardDelay-Reverse%-ReverseDelay—LineLength—Iterations-Direction*.txt, where the Direction variable is enumerated F or R, thus indicating if the capture is from the TCP Forward path (between Quoyloo and Stroma), or the TCP Reverse path (between Quoyloo and Raasay).

4.2.6.1 Distributed Measurement System State Machine

The state machine of the distributed Measurement System is illustrated in Figure 30. The state machine executes on the test command station Missouri. A perl script, called MakeMyClick was developed, in order to automate the process of generating a Click configuration for each experiment. An experimental run is initialised by loading the 6 run variables, and MakeMyClick is called on Quoyloo to configure the Reorder ElementClass and Delay elements appropriately in a Click language file. The Click language file is checked for correctness, and the click router on Quoyloo is configured with the file.

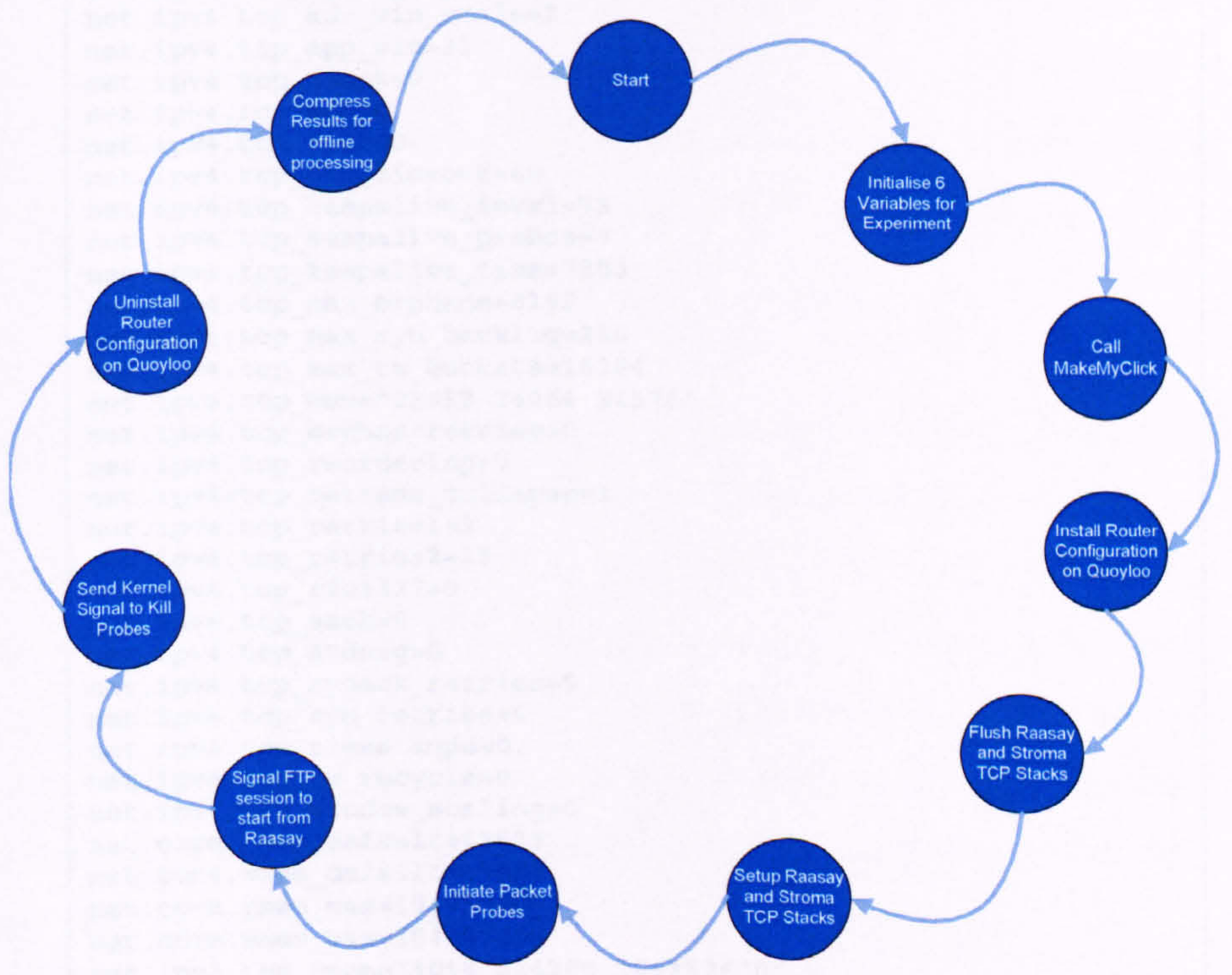

Figure 30 - Measurement System State Machine

On Raasay and Stroma, an *rsh* call is made to restart the TCP/IP networking stack, and to flush all variables from previous experiments using the net.ipv4.route.flush *sysctl* command. Upon restarting the TCP/IP stacks, a number of variables are loaded into the stacks from bash scripts files, thus ensuring that the kernel settings are consistent across experimental runs. Table 3 illustrates some of these variables, which are loaded through Kernel *SysCtl* calls.

The values in Table 3 illustrate the TCP stack tuning applied in order to maximise throughput from the sending and receiving robots, thus stressing the reorder inducing router as much as possible. The TCP tuning parameters are based on values used in the NASA Engineering and Research Network [Nasa08], and involve setting 100 Megabyte networking send and receive buffers, disabling timestamps for performance, and allowing the TCP advertised window to grow to its full size of 65535 bytes. The use of *tcp_no_metrics_save* ensures that the TCP congestion state variables, such as *ssthresh*, are flushed between experiments.

```
net.ipv4.ip_forward=0
net.ipv4.ipfrag_high_thresh=262144
net.ipv4.ipfrag_low_thresh=196608
net.ipv4.ipfrag_time=30
net.ipv4.ip_local_port_range="1024 4999"
net.ipv4.tcp_abort_on_overflow=0
net.ipv4.tcp_adv_win_scale=2
net.ipv4.tcp_app_win=31
net.ipv4.tcp_dsack=0
net.ipv4.tcp_ecn=0
net.ipv4.tcp_fack=0
net.ipv4.tcp_fin_timeout=60
net.ipv4.tcp_keepalive_intvl=75
net.ipv4.tcp_keepalive_probes=9
net.ipv4.tcp_keepalive_time=7200
net.ipv4.tcp_max_orphans=8192
net.ipv4.tcp_max_syn_backlog=256
net.ipv4.tcp_max_tw_buckets=16384
net.ipv4.tcp_mem="23552 24064 24576"
net.ipv4.tcp_orphan_retries=0
net.ipv4.tcp_reordering=0
net.ipv4.tcp_retrans_collapse=1
net.ipv4.tcp_retries1=3
net.ipv4.tcp_retries2=15
net.ipv4.tcp_rfc1337=0
net.ipv4.tcp_sack=0
net.ipv4.tcp_stdurg=0
net.ipv4.tcp_synack_retries=5
net.ipv4.tcp_syn_retries=5
net.ipv4.tcp_timestamps=0
net.ipv4.tcp_tw_recycle=0
net.ipv4.tcp_window_scaling=0
net.core.rmem_default=65535
net.core.wmem_default=65535
net.core.rmem_max=104857600
net.core.wmem_max=104857600
net.ipv4.tcp_rmem="4096 524288 104857600"
net.ipv4.tcp_wmem="4096 524288 104857600"
net.core.netdev_max_backlog=30000
net.ipv4.tcp_no_metrics_save=1
```

Table 3 - Linux Kernel Variables

The majority of experiments involve setting *tcp_sack*, *tcp_dsack*, *tcp_fack*, and *tcp_reordering* off, although experiments with these options activated are discussed in Section 0.

Upon checking that the correct kernel variables have been set in sending and receiving

hosts, the test commander initialises the packet probes as discussed in Section 4.2.5. Missouri then calls a bash script on Raasay to commence the active FTP session, and upload a standard 10 Megabyte file. Raasay signals completion of this upload to Missouri, which then sends a Linux kernel kill signal to the packet probes, forcing them to flush their contents to disk. Missouri then calls Quoyloo to uninstall the Click router configuration, compress the captured packet traces using a zip algorithm, and then prepare to initiate the next experiment run.

4.2.7 Post-processing of Results

Upon completion of a batch of experiments, a Perl program called 'Oos Parser' was developed to parse the traces of out-of-sequence packets, and generate statistics on what was observed. The main algorithm implemented in 'Oos Parser' is illustrated in Figure 31, which illustrates the unique ability of this measurement to be able to correlate packet captures at two points, thus improving the ability to differentiate the cause of retransmissions as a result of packet reordering.

The algorithm initially parses the packet traces recorded from the Forward probe; that is, those packets which have already undergone reordering on the Forward path and are observed at the probe between Quoyloo and Stroma. Observation of packets at this position facilitates the assumption that this is the order at which they will appear at the receiving host of the FTP session and that this is the order and relative timing at which the packets will be received. As each data packet in the trace is analysed, a data hash for that connection is created, keyed by the Sequence Numbers observed, and a count maintained for the number of occurrences of that Sequence Number.

The packet trace recorded from the Reverse probe is then parsed and, in a similar technique, a data hash is created keyed by Acknowledgement Numbers observed and a count maintained for the number of occurrences of each Acknowledgement Number.

Using Equation 1, packets that are defined to be Reordered, have their relevant Reordering Extent calculated as defined by Equation 2.

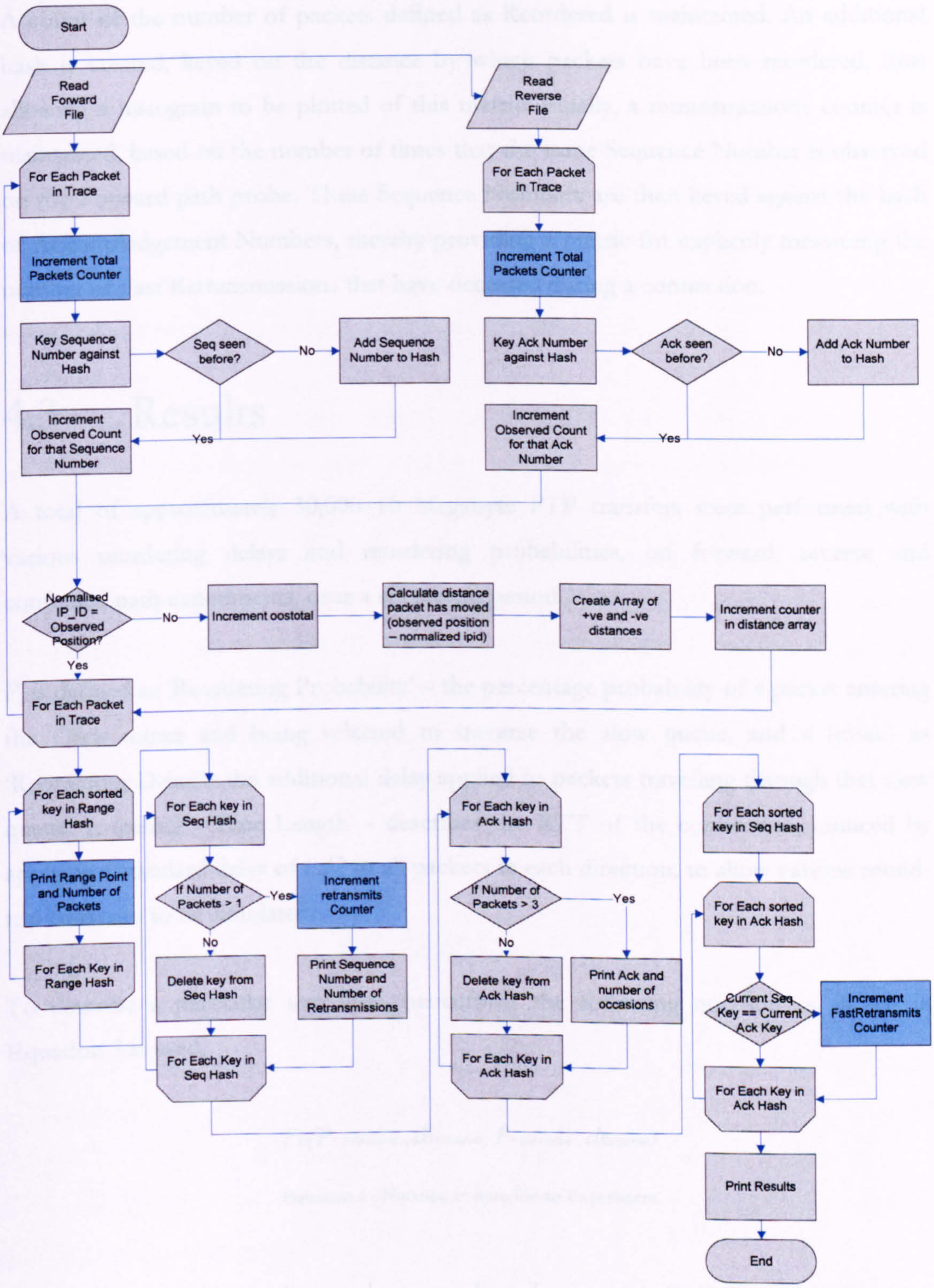

Figure 31 - OOS Parser Algorithm

A count of the number of packets defined as Reordered is maintained. An additional hash is created, keyed on the distance by which packets have been reordered, thus allowing a histogram to be plotted of this metric. Finally, a retransmissions counter is maintained, based on the number of times that the same Sequence Number is observed on the Forward path probe. These Sequence Numbers are then keyed against the hash of Acknowledgement Numbers, thereby providing a metric for explicitly measuring the number of Fast Retransmissions that have occurred during a connection.

4.3 Results

A total of approximately 30,000 10 Megabyte FTP transfers were performed with various reordering delays and reordering probabilities, on forward, reverse and combined path experiments, over a six month period.

P_r is defined as 'Reordering Probability' – the percentage probability of a packet entering the Click router and being selected to traverse the slow queue, and d (msec) as 'Reordering Delay' - the additional delay applied to packets travelling through that slow queue. L (msec) - 'Line Length' - describes the RTT of the connection, induced by applying a standard delay of $L/2$ to all packets in each direction, to allow various round-trip distances to be simulated.

To describe a particular test flow, henceforth the following notation as shown in Equation 3 is used.

$$F_L(P_{r\,Forward}, d_{Forward}, P_{r\,Reverse}, d_{Reverse})$$

Equation 3 - Notation to describe an Experiment

Each of the experiments discussed was conducted using a single FTP over TCP, which although it does not account for competing cross traffic and varying traffic patterns, makes it possible to gauge the ideal performance of TCP algorithms in a controlled environment.

Finally, the experiments were performed such that no packet loss would occur. Therefore, all retransmissions measured were caused by packet reordering.

Multiple tests were carried out under varying conditions; the aim was to demonstrate the effects of forward path, reverse path, and combined forward and reverse reordering. In each experimental run, a fixed reordering delay was applied. These were increased at intervals between 0 and up to 5 x RTT, with the likelihood of a packet being selected varied from random samples with probability 5%, 10%, 15%, and 25%.

The results illustrate typical data obtained for Reno TCP source and destinations, with TCP maximum window size of 65 kilobytes and with the results plotted as the mean over five experimental runs with 90% Confidence Intervals. The choice of 90% and five experimental runs is important, as the purpose of these experiments is to measure TCP over a wide range of conditions in order to gain an understanding of the complexities of the protocol, within a reasonable timescale. Testbed emulation requires significantly longer time periods and requires significant storage for captured data, as when compared to other simulation studies of TCP. The results reported in this chapter represent six months of experimental data measured on a single testbed, and over 30,000 individual FTP transfers

4.3.1 Experiment Validation

In such large scale experiments, there are clearly many opportunities for external influences which may affect the validity of the results obtained. As the main contribution of this study compared with previous studies [Laor02] is that the method of packet reordering is induced via time periods, rather than by packet position movements, it is important to consider all parts of the experiment which may influence time measurements in any way.

Secondly, as discussed in Chapter 2, all two-point measurement techniques require a highly accurate notion of time at all probes in an Internet. In this particular experiment, all machines were synchronised to the NTP server at the start of each experiment, but in this particular set up where both network taps reside in Hoy, synchronisation between

the tap points is therefore known to be exact. If this technique was applied in a wider context where one machine could not operate as both taps simultaneously, a more accurate method of synchronisation between taps would be required.

The use of the dedicated GPS receiver Yell, as a NTP Stratum 1 server, and the immediate proximity of each machine to this server, allows the assumption that all machines in the testbed are synchronised with an accuracy better than 1 microsecond. 1 microsecond is the limit to which LibPcap can perform packet time-stamping on the particular hardware architectures of the machines in the testbed, and therefore clock synchronisation is not considered to be a limitation of this experiment.

The clock resolution of the software-configurable-router is another area which could introduce errors, as it is this timer which introduces both the RTT delay and Reordering Delay applied in each experiment. With the particular 64-bit hardware architecture used in these experiments, and the version of Click installed, Click has the ability to operate at 1 microsecond resolutions and, therefore, all Line Lengths are Reordering Delays are specified in units of microseconds. To confirm that these time resolutions were as expected, a simple independent test was performed as detailed in Table 4.

Table 4 illustrates initial experiments that were carried out using 200 ICMP packets to measure the additional delay placed on traffic due to the use of the Click router, in both directions, for a variety of emulated RTT paths. In each experiment, the standard Click Language installation file was used with all Elements instantiated in the router, but with Reordering Probability set to zero. Although Click was shown to cause a slight increase in RTT of approximately 0.15ms in each direction, this is insignificant in comparison to the additional RTT that would be intentionally introduced by the click script. This additional latency is shown in Table 4 to be relatively consistent across all RTT paths, and therefore will have a consistent effect on the measurements made.

		Round-Trip-Time Measurement (msec)				
		Minimum	Average	Maximum	Standard Deviation	Packet Loss (%)
(L/2) = 0	Raasay-> Stroma	0.206	0.319	0.519	0.057	0
(L/2) = 0	Stroma-> Raasay	0.210	0.320	0.475	0.053	0
(L/2) = 0.025	Raasay-> Stroma	50.262	50.327	50.417	0.132	0
(L/2) = 0.025	Stroma-> Raasay	50.207	50.316	50.485	0.247	0
(L/2) = 0.05	Raasay-> Stroma	100.216	100.321	100.425	0.426	0
(L/2) = 0.05	Stroma-> Raasay	100.214	100.318	100.430	0.328	0
(L/2) = 0.075	Raasay-> Stroma	150.215	150.323	150.431	0.088	0
(L/2) = 0.075	Stroma-> Raasay	150.208	150.315	150.498	0.351	0
(L/2) = 0.1	Raasay-> Stroma	200.239	200.339	200.466	0.490	0
(L/2) = 0.1	Stroma- >Raasay	200.256	200.312	200.392	0.574	0
(L/2) = 0.15	Raasay-> Stroma	300.216	300.298	300.406	0.598	0
(L/2) = 0.15	Stroma-> Raasay	300.267	300.353	300.598	0.631	0

Table 4 – Latency Test of Click Router

Competing flows may also account for inaccuracies in the measurements obtained. In each case, all initialisation, setup and processing of results took place over the electrical network; only the FTP session to be measured was carried out on the fibre gigabit network. Network adaptors on the electrical network were prefixed with Class C subnet 156.141.122.x, while adaptors on the gigabit network were prefixed with Class C subnet 10.0.0.x. Each experiment consisted of a single active FTP session, with Raasay as client and Stroma as FTP server. Once the FTP session had been established on port 21, Stroma would issue a single Put command to upload the 10 Megabyte file. This would result in a second TCP connection being established on random high port numbers. By adding a filter to each packet probe to ignore port 21 traffic, it was only this second flow of FTP traffic which would be captured.

The throughput of the entire end-to-end path is another factor which can be used to validate the correctness of this experiment. The Bandwidth Delay Product (BDP), illustrated in Equation 4, describes the amount of data that can be in transit in the network. BDP is important for windowing protocols such as TCP, as BDP describes the amount of 'un-acknowledged' data which can be in-flight in the network at any one time and, therefore, the maximum throughput that a TCP host can achieve. The results reported in this Chapter allow the receiving host to advertise the full 65kbyte *rwnd* using the 2 byte TCP header; the optional TCP Window Scaling feature [Jaco92] is not considered, as it is not enabled as standard in current modern operating systems[Micr08].

$$\text{BDP (bytes)} = \text{total available link capacity (kbytes/sec)} \times \text{RTT (msec)}$$

Equation 4 - Bandwidth Delay Product

For a RTT of 50ms, where the *rwnd* and, therefore, the BDP are limited to 65160, the total available capacity of the link can be calculated as 1311 kbytes/sec.

The limitations of TCP throughput have been described by Mathis et al [Math97]; for loss rates less than 1%, the maximum achievable throughput of a TCP connection is limited by Equation 5, where *rwnd* is the receiver host's advertised window, RTT is the sending host's evaluation of RTT, and p is the probability of packet loss.

$$Throughput \leq \frac{\text{Max(rwnd)}}{\text{RTT}} \times \frac{1}{\sqrt{p}}$$

Equation 5 - Mathis TCP Throughput Limitation

In the experiments reported in this chapter, no packet loss is induced and it is assumed that all packets will eventually reach the destination. Equation 5 illustrates that the Mathis Throughput limitation does not account for this case, as the $1/\sqrt{p}$ term would be expected to tend towards ∞. However, it should be noted that although no packet loss was specifically induced in the network, the effects of extremely late packets are equivalent to those of lost packets; a factor that the Mathis formula also does not

account for. For experiments where no packet loss was induced, and no packet reordering is induced, the $1/\sqrt{p}$ term can be simplified to 1, and the maximum throughput rate during a 50ms RTT experiment can be calculated to be 65160 bytes / 0.05 seconds, equalling 1311 kbytes/second.

For a file size of exactly 10 Megabytes, it can therefore be calculated that the ideal transmission time, assuming no loss and no packet reordering, over a 50 msec RTT is 7.6 seconds. Clearly, this is the ideal throughput and this simplified formula does not consider the slow start algorithm, which for short-lived flows will dominate a large proportion of the transmission time. The packet capture trace illustrated in Figure 29 is illustrated with more detail in the Appendix. The Appendix clearly shows the operation of the flow control and congestion control algorithms at the start of a TCP connection. Column O indicates the receiver's *rwnd*, which does not grow to the full 65 kbytes until packet 64 in the trace. Column M indicates the payload size of each packet; analysis of the bursts of data packets indicates operation of the sender congestion control algorithm and linear growth as larger bursts of packets are injected into the network. In this capture trace, the sending TCP does not launch a full volley of its 45 packet maximum cwnd until 680 packets into the trace, 6 seconds after the data transfer has commenced. This 6 second period of initial window growth, combined with the 7.6 second maximum throughput limit, combine favourably with the initial result plotted in Figure 32 indicating that the mean average to transfer 10 Mbytes was 19.49 seconds.

Finally, all experiments are performed on the basis that no packet loss will be induced by the network. A ping flood was performed from Raasay to Stroma at the maximum sending rate possible by the machine, which indicated 0% packet loss. It can therefore be concluded that all three of the network nodes, Quoyloo, Raasay and Stroma have the capacity to process the maximum sending rate which can be generated by Raasay and that packet loss will not occur during experiments.

4.3.2 Measuring Forward Path Packet Reordering

The primary purpose of these experiments is to characterise the effect of packet reordering, and to correlate these effects with a metric which describes the amount of reordering occurring in a network. In order to describe the impact that packet reordering has on a TCP flow and on the assumption that in every experiment, exactly the same amount of data was to be transferred, the overall throughput of each experiment is described in the form of the 'Transmission Time' of 10 Megabytes.

This metric was chosen for a number of reasons. Previous methods of articulating the throughput of a link are inaccurate when the effects of packet reordering are considered. As will be discussed in Chapter 5, accepted methods to calculate TCP Goodput are incorrect when packet reordering occurs. Traditional Goodput and Throughput measurements often confuse reordered packets with retransmissions, or are unable to differentiate those retransmissions which have occurred as a result of packet reordering.

Transmission Time serves as a simple metric which fully describes the Quality of Experience that a user will experience during packet reordering. Coupled with the packet capture functionalities described in the testbed, it is possible to measure Transmission Time to microsecond resolution. Transmission Time can then be correlated with the other metrics of packet reordering, described in Equation 1 and Equation 2, to correlate these metrics with their effects.

4.3.2.1 50 msec Round Trip Time

Figure 32 plots a number of experiments performed on an emulated 50 msec RTT, with forward path Reordering Delay plotted on the y axis, and the percentage of forward path Reordering Probabilities plotted as the coloured series as indicated by the legend. The range of Reordering Delays, between 0 to 5 x RTT illustrate the surprisingly large range over which TCP can operate, without serious degradation in Transmission Time. Although the range of Reordering Delays up to 0.25 seconds may appear unlikely, it is important to note that a wide range of Reordering Delays may be observed in a

production network and, therefore, this wide range of values allows full characterisation of the behaviour of TCP under a variety of conditions.

Figure 32 indicates that there are three distinct regions of the graph, which appear as a function of Reordering Delay, regardless of Reordering Probability. This is also prevalent in Figure 39, which illustrates the same features, but for a slightly longer RTT of 150 msec. Figure 32 clearly indicates that between 0 and 0.117 msec Reordering Delay, the effects of Packet Reordering are marginal, with only a very small increase in Transmission Time observed between these two points. At 0.126 msec, there is a step change in performance, where the mean Transmission Time for all connections rises to over 75 seconds. This defines the start of the second period of the graph, between 0.126 msec and 0.198 msec. Here, the effects of Reordering Probability are more obvious, as the 5% series maintains the lowest Transmission Time. However, the difference in performance between the various Reordering Probabilities is not significant and the confidence intervals widen for all, indicating that the performance of all flows has become less predictable. The third region of Figure 32 is evident beyond 0.2 msec, where confidence intervals widen further and the overall Transmission Time for all Reordering Probabilities rises steeply. Within this region, the effects of reordering are extremely difficult to predict, although it is important to consider that the width of the confidence intervals indicates a wildly varying behaviour, with flows alternating drastically between high and low performance. This can be explained by the fact that under these higher degrees of reordering, it would be reasonable to expect large blocks of packets to all be chosen to be delayed by the Reorder Element in the router (whilst retaining relative sequence), therefore being unlikely to cause further spurious reordering retransmissions and also allowing time for congestion control mechanisms to recover from the last reordering event.

The confidence intervals on Figure 32 illustrate the unusual behaviour of packet reordering on TCP flows, when compared to previously well documented anomalies such as loss. As the reordering delay is increased, the mean transmission time tends also to increase, but with a significantly larger error range. This indicates that, unlike percentage loss which guarantees retransmissions and invocation of congestion control, packet reordering is a complicated function of buffer sizes, control algorithms and

Delay Variation, which may or may not have a significant effect on a TCP flow. This variance is significant as it indicates that the effects of reordering are much more difficult to predict than the effects of loss.

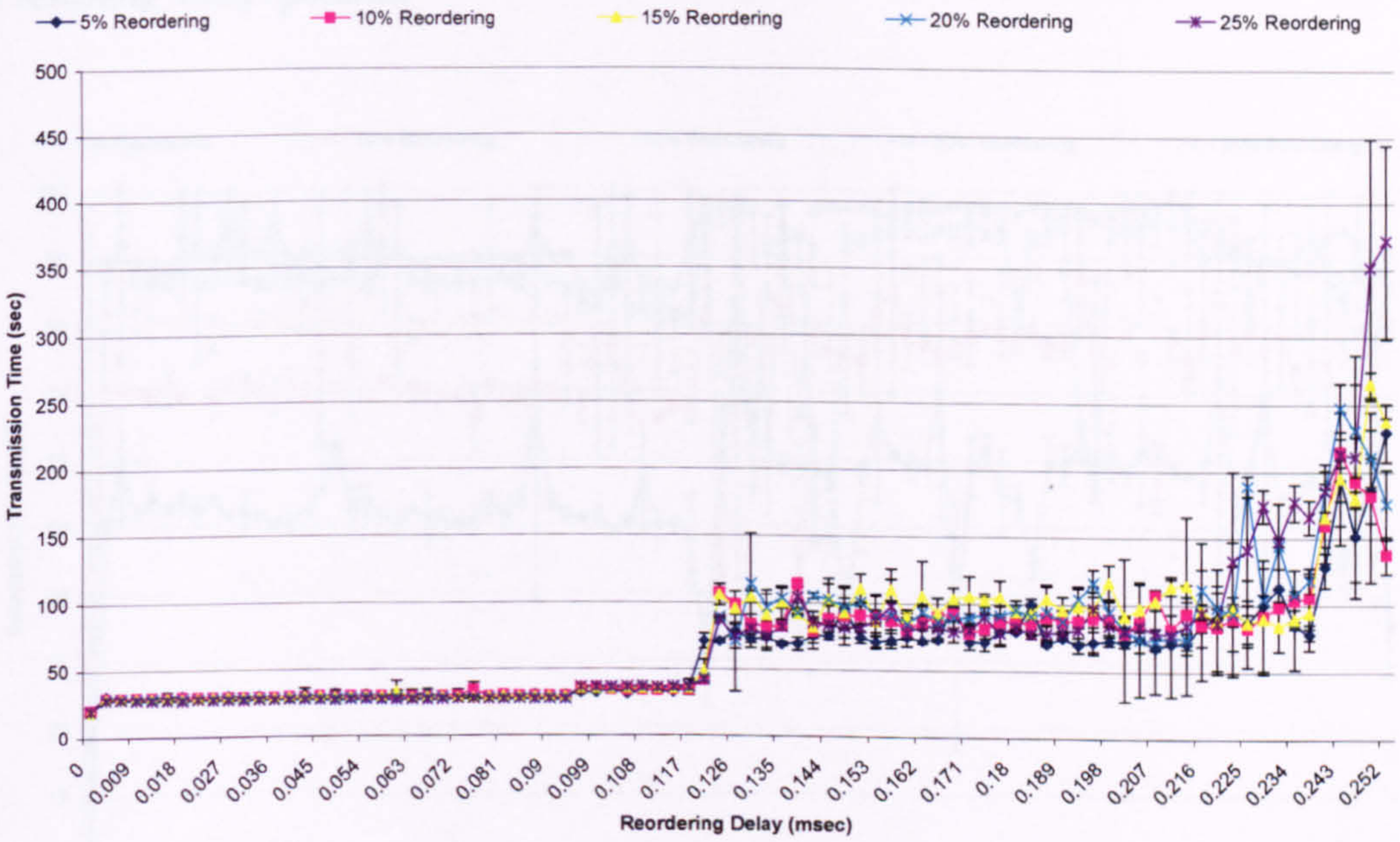

Figure 32 - Mean transmission time of 10 Megabytes, 90% C.I., F_{50}(various, various, 0, 0)

The three distinct regions are an important illustration of TCP's behaviour when undergoing reordering. The transition from the first to the second region, indicates that there is a point where reordering will always cause an unnecessary fast retransmission and therefore resulting a halving of the congestion window. Once this spurious fast retransmission has been triggered, the sending rate is reduced in a uniform fashion and, after this point within the second region, reordering has little additional effect. The important observation which can be made on this transition from the first to second regions, is that Figure 32 suggests that it occurs independently from the Reordering Probability. The step change is observed to occur at the same Reordering Delay point for all Reordering Probabilities. This observation is of significance, as it suggests that large amounts of packet reordering can be permitted – up to 25% of packets – but by ensuring that Reordering Delay is maintained below the step change, the effects of reordering on QoE will be negligible.

Figure 33 illustrates the percentage of packets which have appeared out of sequence, for the same experiments as plotted in Figure 32. 'Out of Sequence' is termed as any packet where 'receiving sequence number' and 'sending sequence number' do not match, as defined in Equation 1. Therefore, this percentage metric includes both 'late' packets and the resulting 'early' packets.

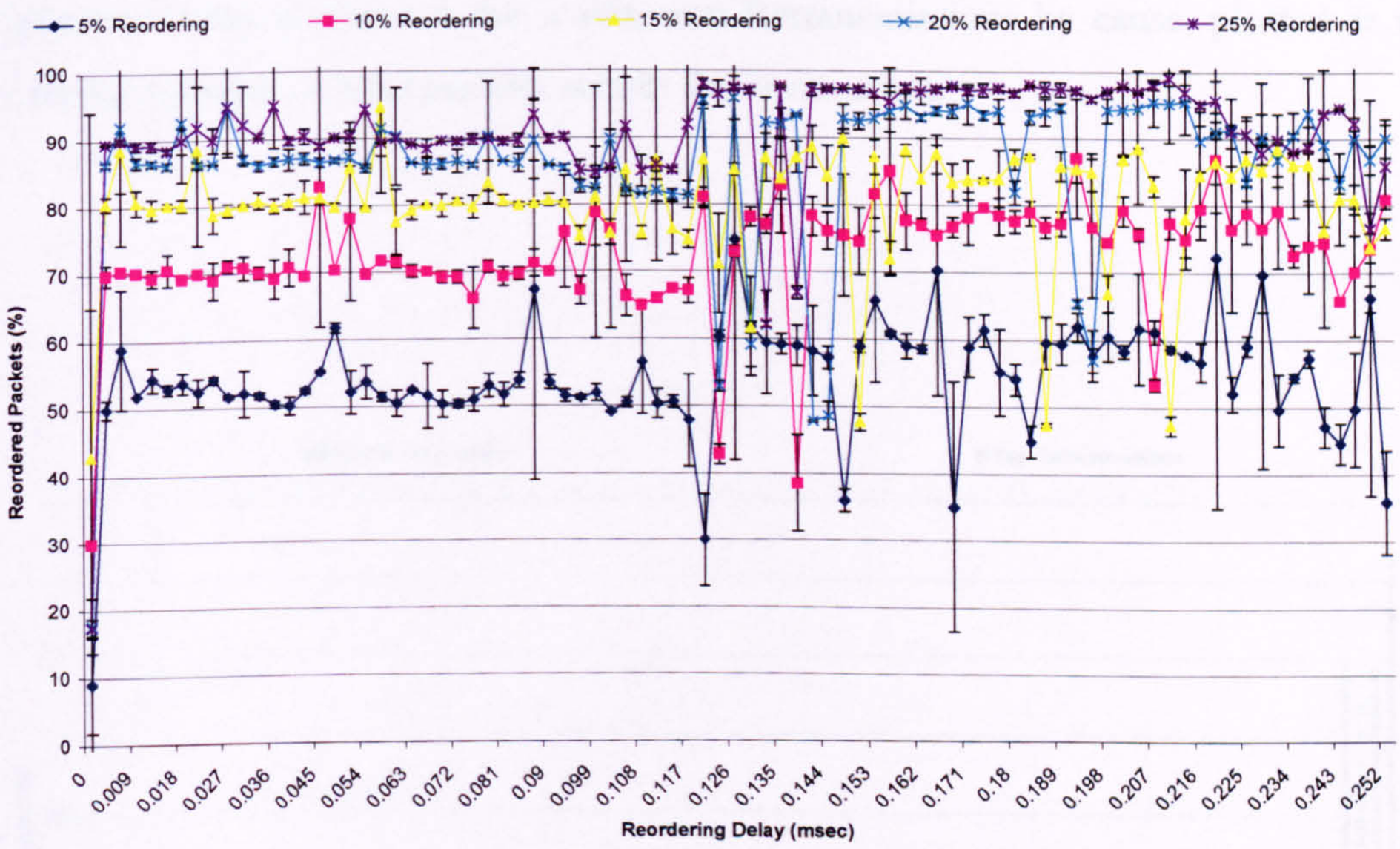

Figure 33 - Percentage Reordered Packets, 90% C.I., F_{50}(various, various, 0, 0)

Correlation of Figure 32 with Figure 33 allows evaluation of this metric, and other percentage metrics of packet reordering which have been proposed, such as Type-P-Reordering in RFC4737, Paxson, Bennett, Loguinov, Bellardo, Tsinghua and Perkins. Although each method relies on a slightly different approach for calculating the percentage of reordered packets, each method relies on reporting their results in the same way.

Figure 33 suggests that it is impossible to predict the effects of packet reordering, based on a percentage reordering metric. The key transition in Figure 32 from the first to second regions of the graph at 0.126 msec, is in no way differentiable from the series plotted in Figure 33. For each Reordering Probability, the percentage of Out-of-Sequence packets rises steeply and is relatively consistent throughout all Reordering Delays. As discussed for Figure 32, it is Reordering Delay which has the most dramatic

effect on TCP behaviour, leading to the conclusion that percentage reordering metrics are not particularly useful in describing the QoS of a connection.

Figure 34 to Figure 38 illustrate results obtained during the same 50 msec RTT experiments, which analyse the cause of packet retransmissions observed and classified by the Oos Parser code. Each Reordering Probability is plotted separately, with Reordering Delay plotted on the x axis, and Retransmissions by cause, plotted as the percentage number of total packets sent in the connection.

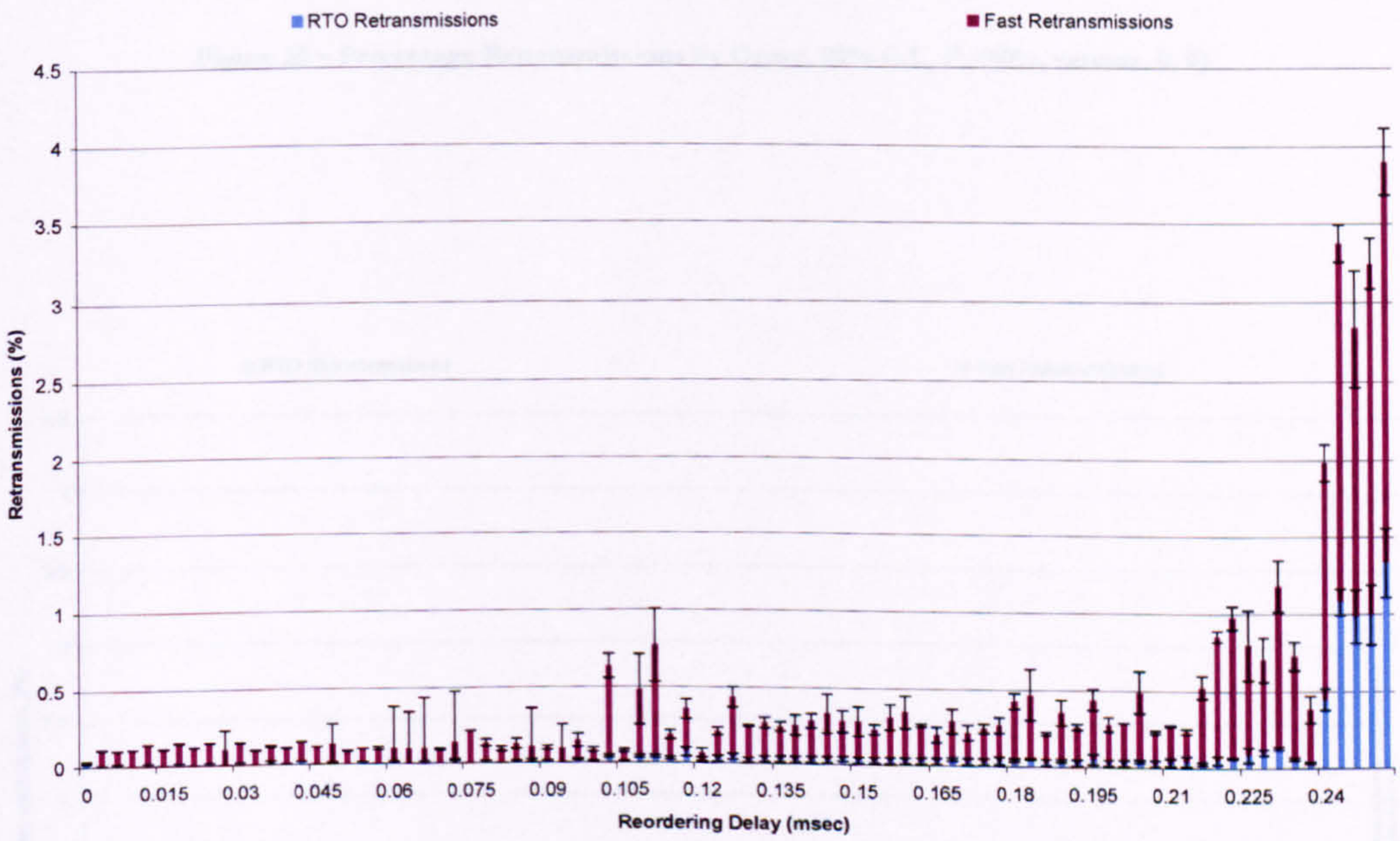

Figure 34 – Percentage Retransmissions by Cause, 90% C.I., F_{50}(5%, various, 0, 0)

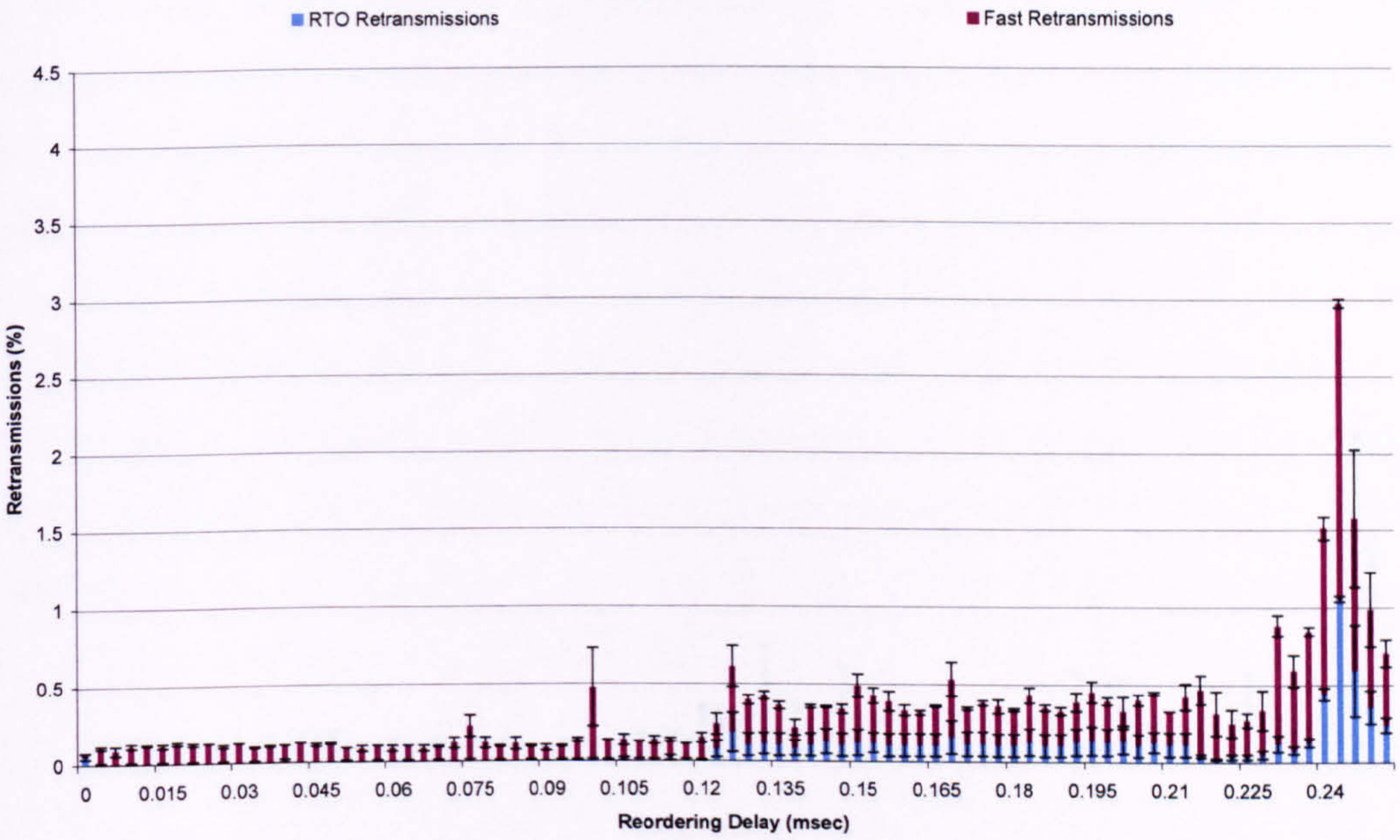

Figure 35 – Percentage Retransmissions by Cause, 90% C.I., F_{50}(10%, various, 0, 0)

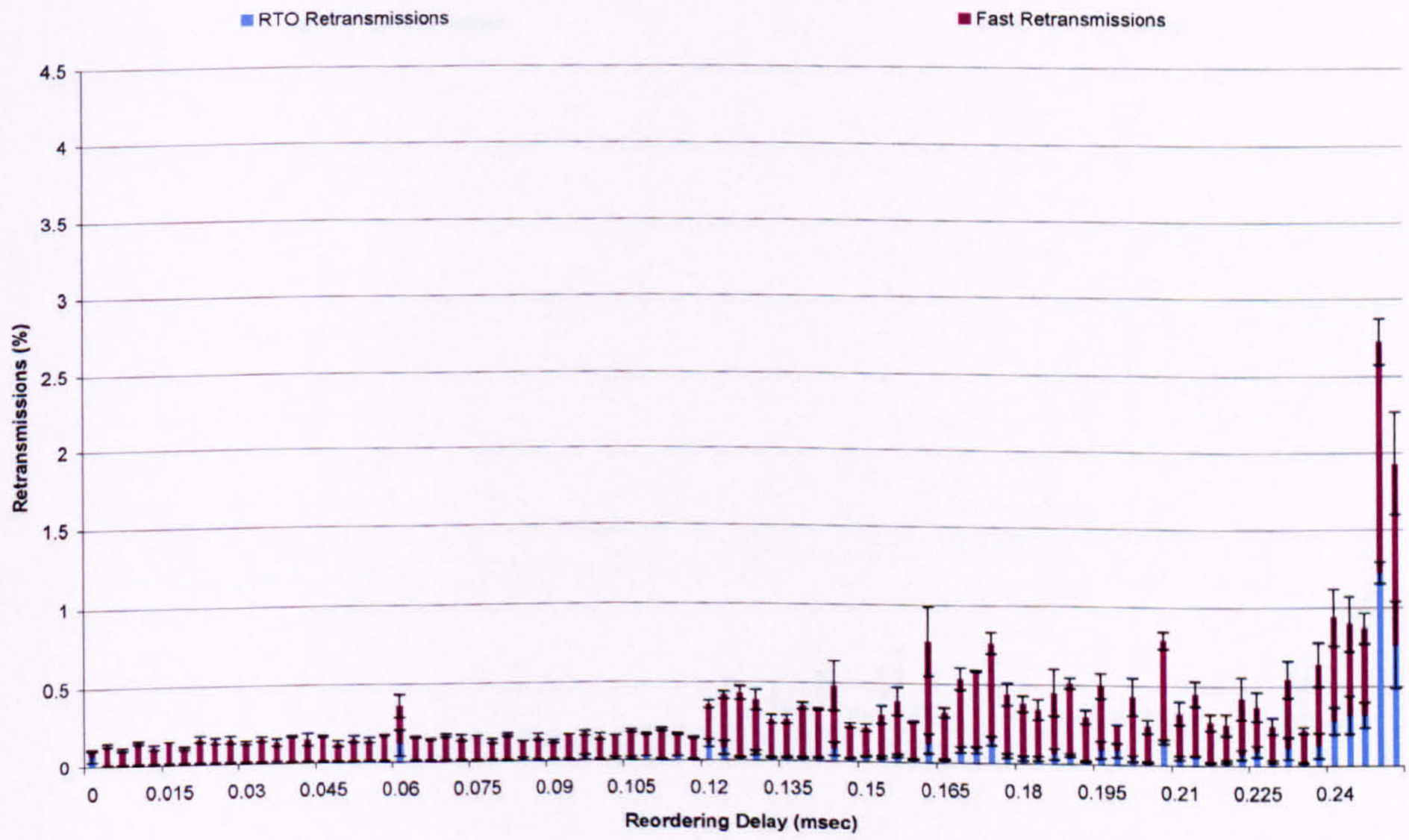

Figure 36 - Percentage Retransmissions by Cause, 90% C.I., F_{50}(15%, various, 0, 0)

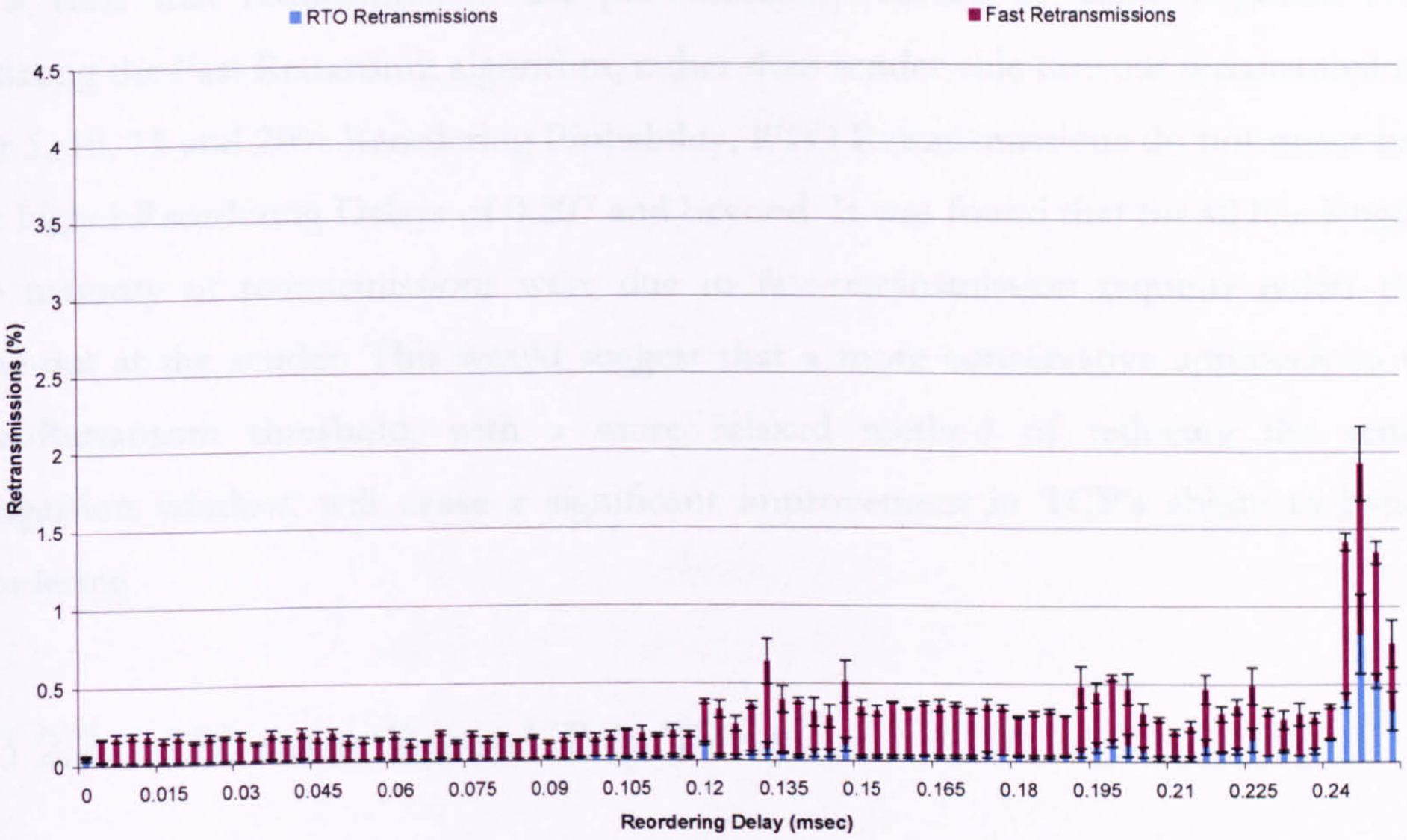

Figure 37 - Percentage Retransmissions by Cause, 90% C.I., F_{50}(20%, various, 0, 0)

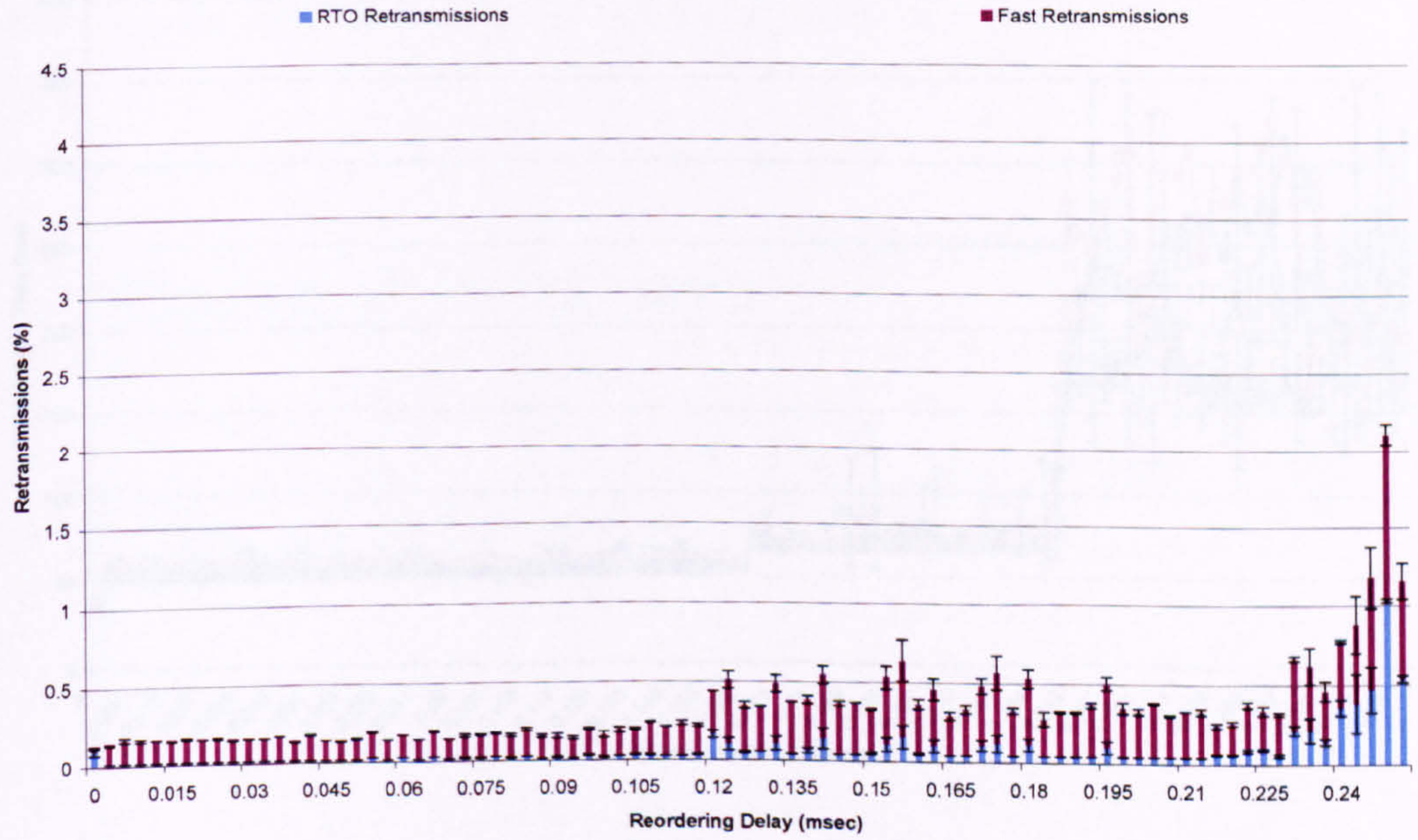

Figure 38 - Percentage Retransmissions by Cause, 90% C.I., F_{50}(25%, various, 0, 0)

It is clear that retransmissions are predominantly caused by triple duplicate Acks initiating the Fast Retransmit algorithm, rather than sender-side timeout retransmissions. For 5, 10, 15 and 20% Reordering Probability, RTO Retransmissions do not occur until the higher Reordering Delays of 0.207 and beyond. It was found that for all line lengths, the majority of retransmissions were due to fast-retransmission requests rather than time-out at the sender. This would suggest that a more conservative approach to the Fast-Retransmit threshold, with a more relaxed method of reducing the sender congestion window, will cause a significant improvement in TCP's ability to handle reordering.

4.3.2.2 150 msec Round Trip Time

To allow investigation of the effects of forward path packet reordering, with respect to varying RTT values, experiments were performed on an emulated 150 msec RTT and illustrated in Figure 39 - Figure 45.

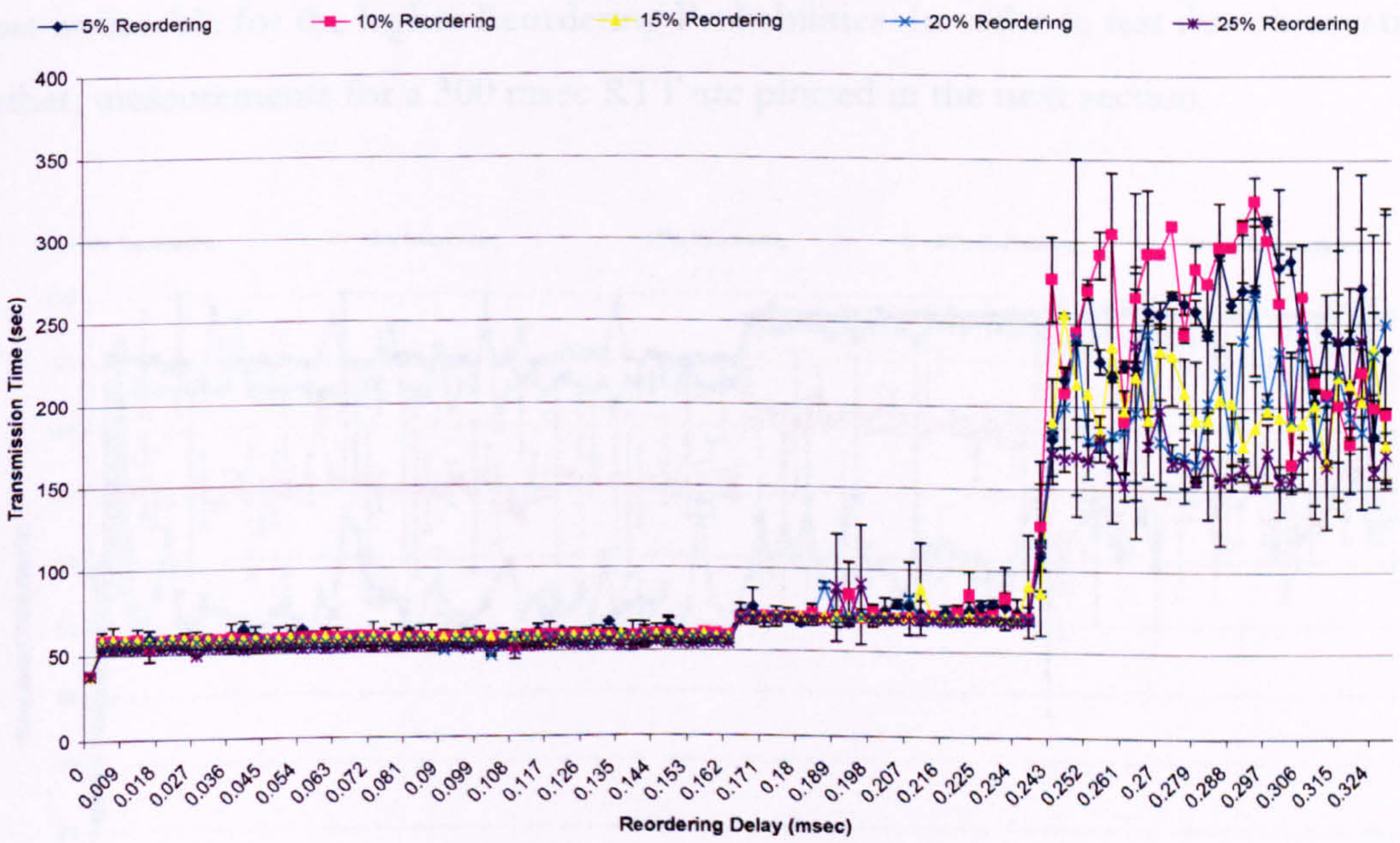

Figure 39 - Mean transmission time of 10 Megabytes, 90% C.I., F_{150}(various, various, 0, 0)

In Figure 39, the mean transmission time for 10 Megabytes is plotted for various Reordering Probabilities and Delays, over a set of experiments with 150 msec RTT. Figure 39 indicates similarities with Figure 32, where three distinct regions are visible as

discussed in the previous section. The move from the first to second regions of the graph, where the Reordering Delay has caused an unnecessary Fast Retransmission, occurs at 0.171 msec and therefore the increase in RTT has increased the point at which this transition occurs.

Figure 40 illustrates the Percentage Reordered Packets for the 150 msec RTT, which as discussed in the previous section, can be difficult to relate to the effects of reordering. Each Reordering Probability results in a relatively constant percentage number of reordered packets – 5% Reordering Probability results in 55% of packets being measured as reordered, whereas 25% Reordering Probability results in between 90 and 100% of packets being measured as reordered.

Correlating Figure 39 and Figure 40 is more successful at this RTT of 150 msec than in the previous discussion of 50 msec RTT. The transition from the first to second region occurs at 0.171 msec on Figure 39; this coincides with a very slight rise in the percentage of reordered packets in Figure 40, where at 0.171 msec a step increase can be observed, most noticeable for the higher Reordering Probabilities. In order to test this observation further, measurements for a 300 msec RTT are plotted in the next section.

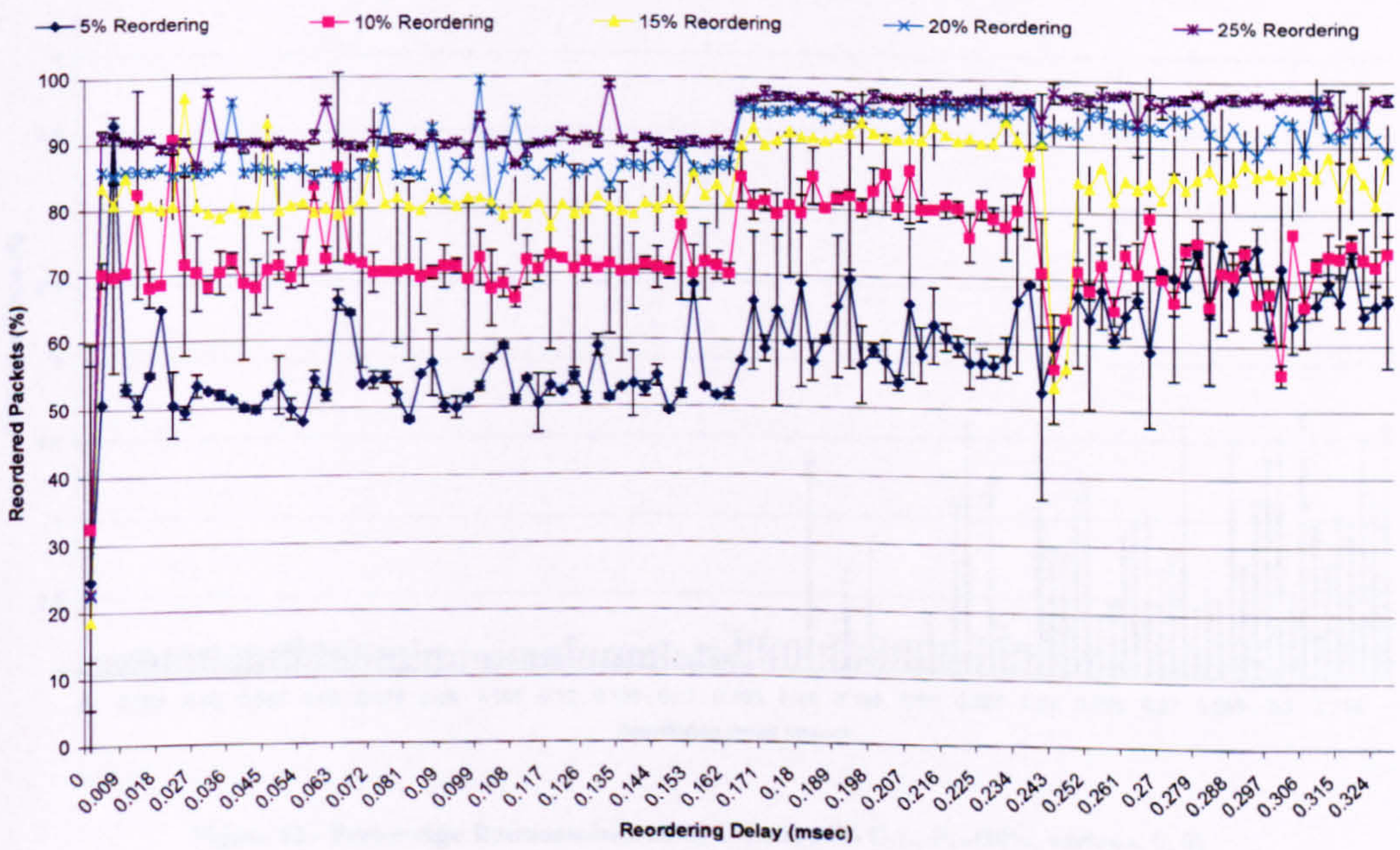

Figure 40 - Percentage Reordered Packets, 90% C.I., F_{150}(various, various, 0, 0)

Figure 41 to Figure 45 perform retransmission analysis on the various Reordering Probabilities for a 150 msec RTT.

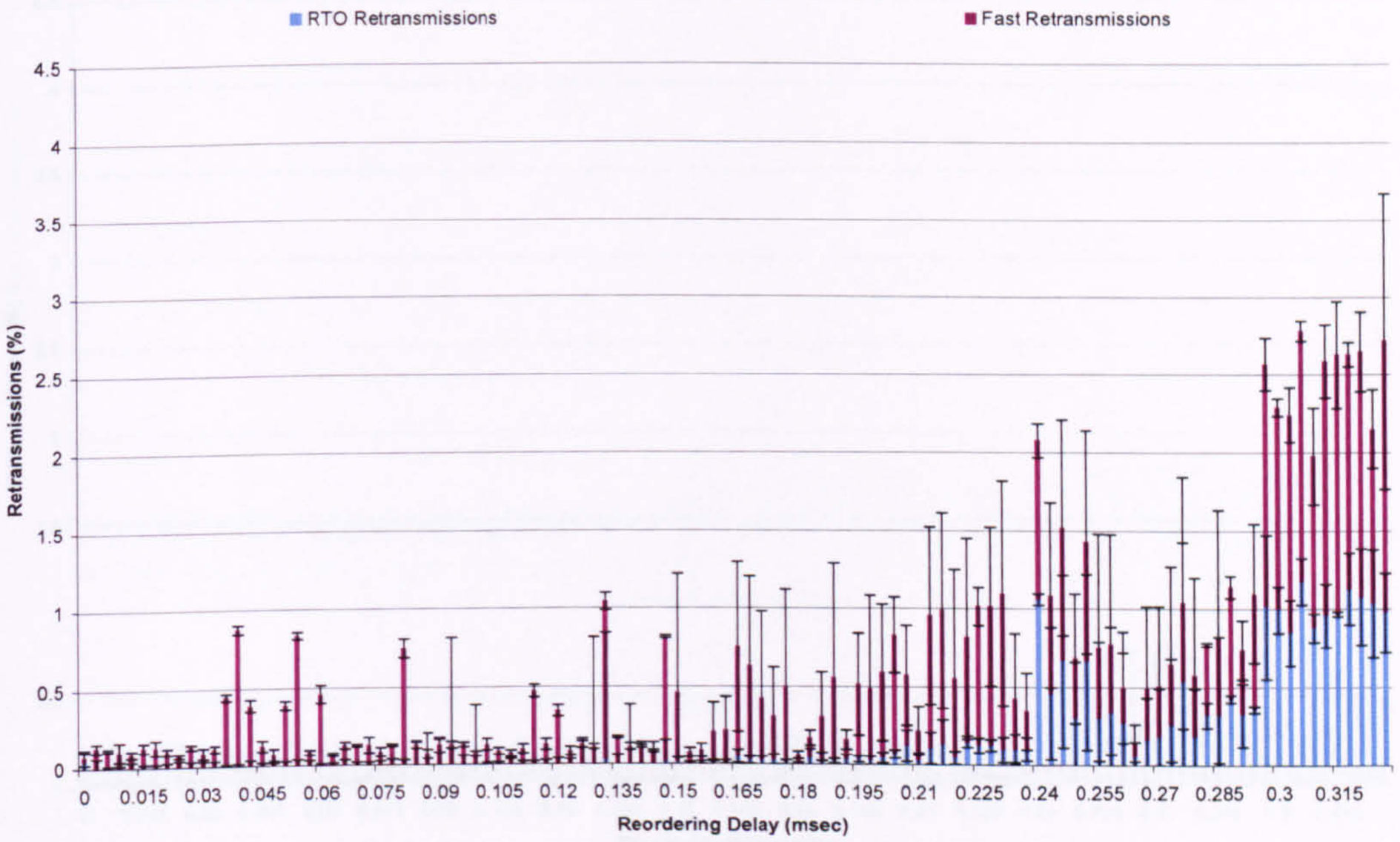

Figure 41 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(5%, various, 0, 0)

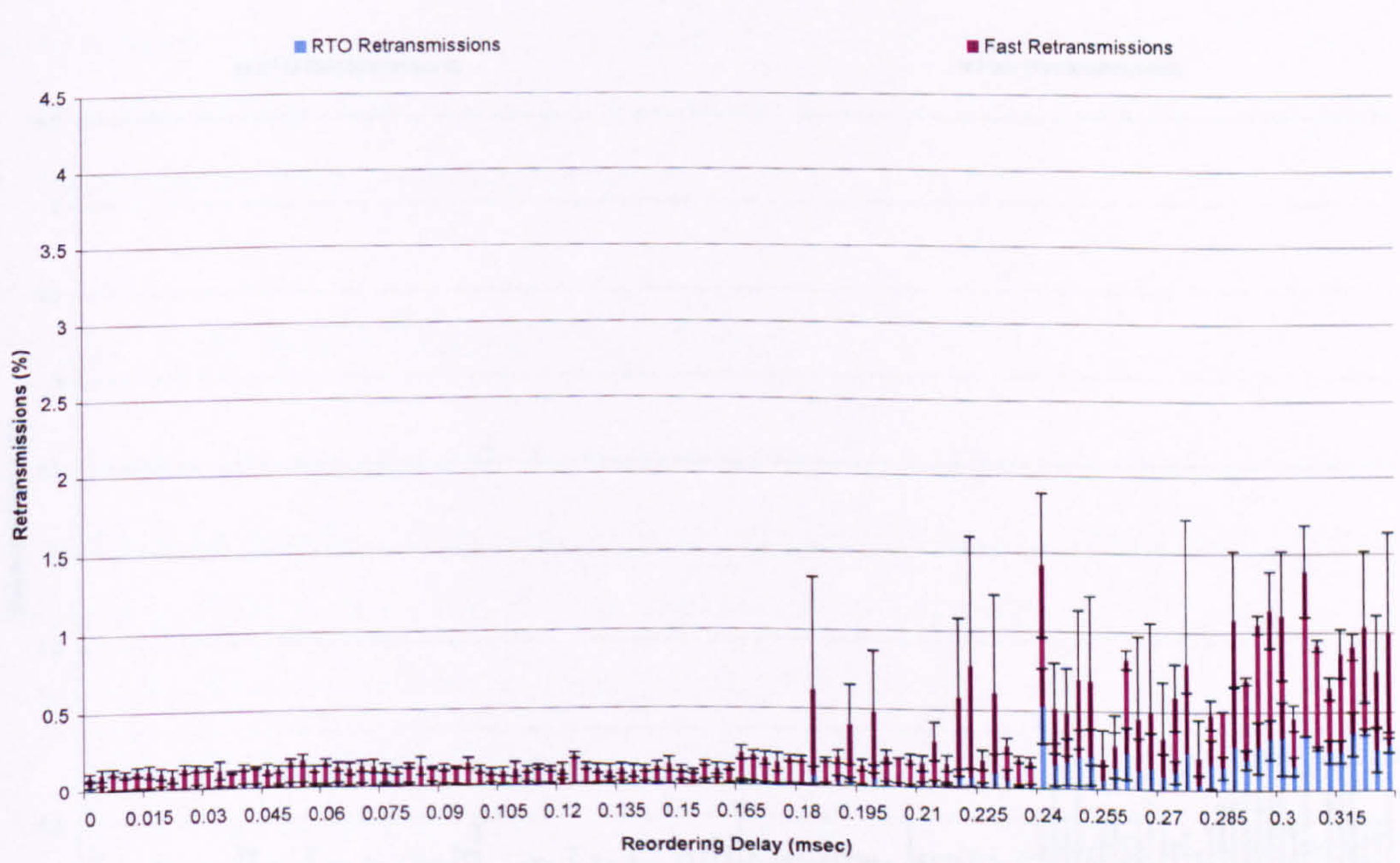

Figure 42 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(10%, various, 0, 0)

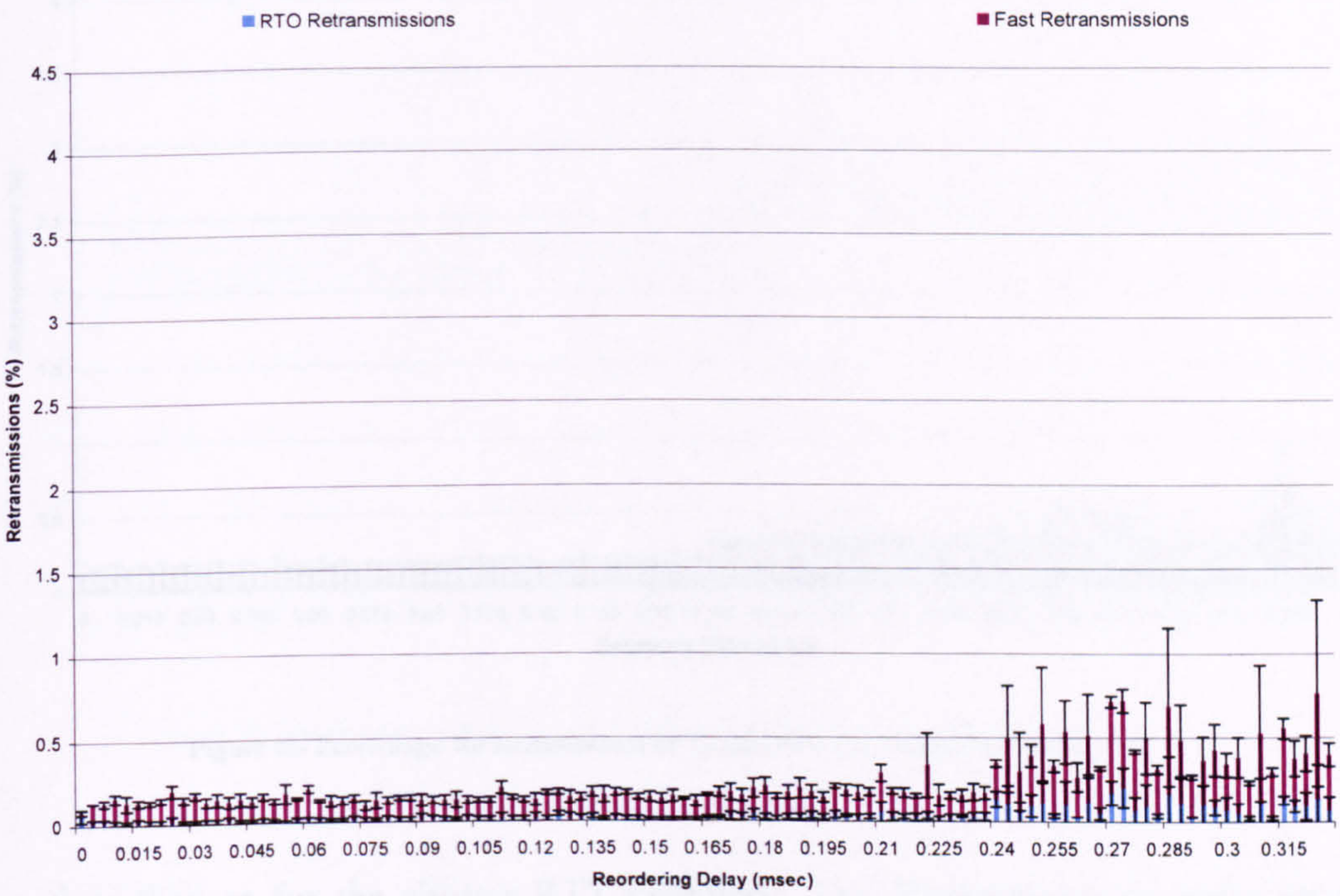

Figure 43 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(15%, various, 0, 0)

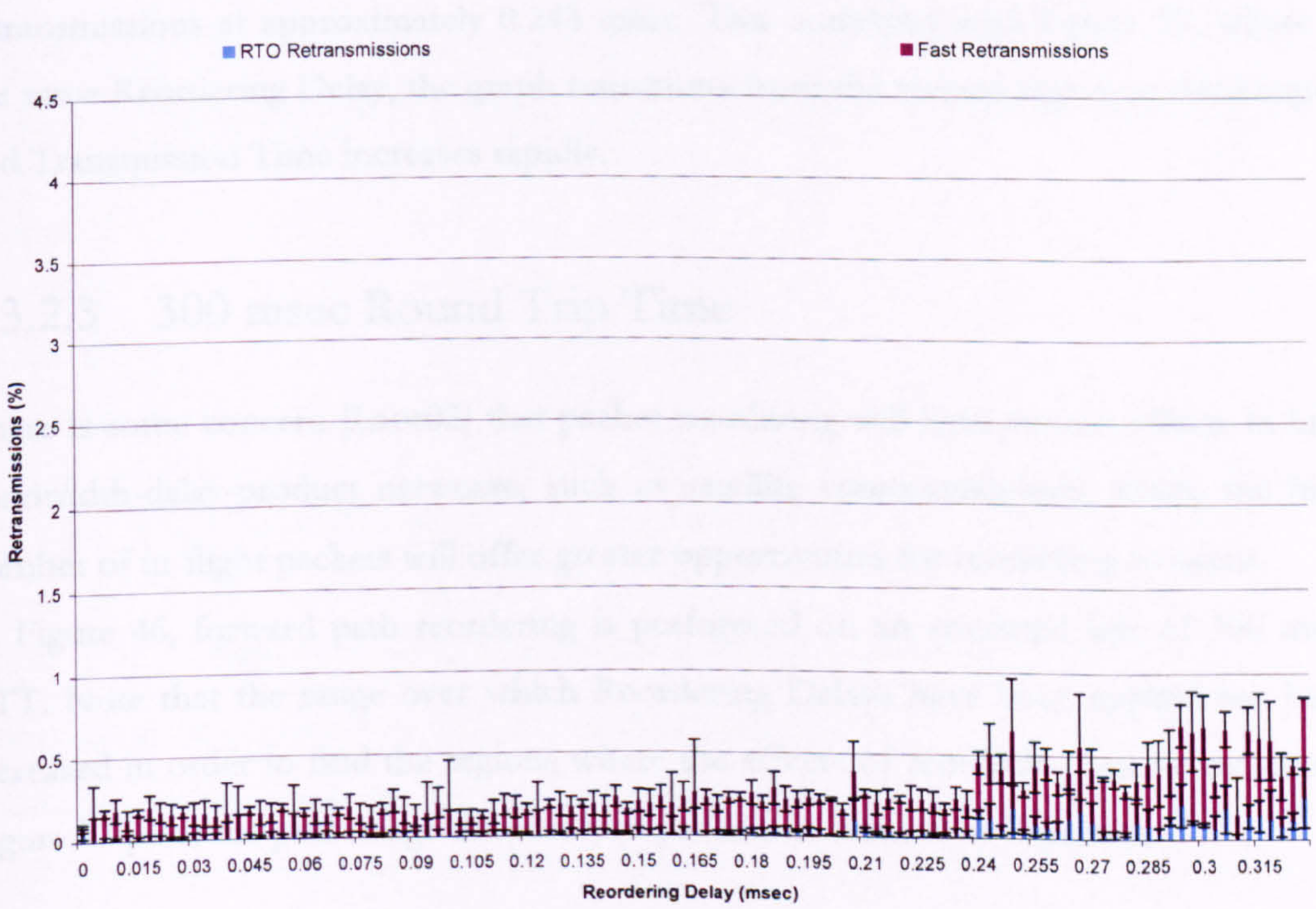

Figure 44 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(20%, various, 0, 0)

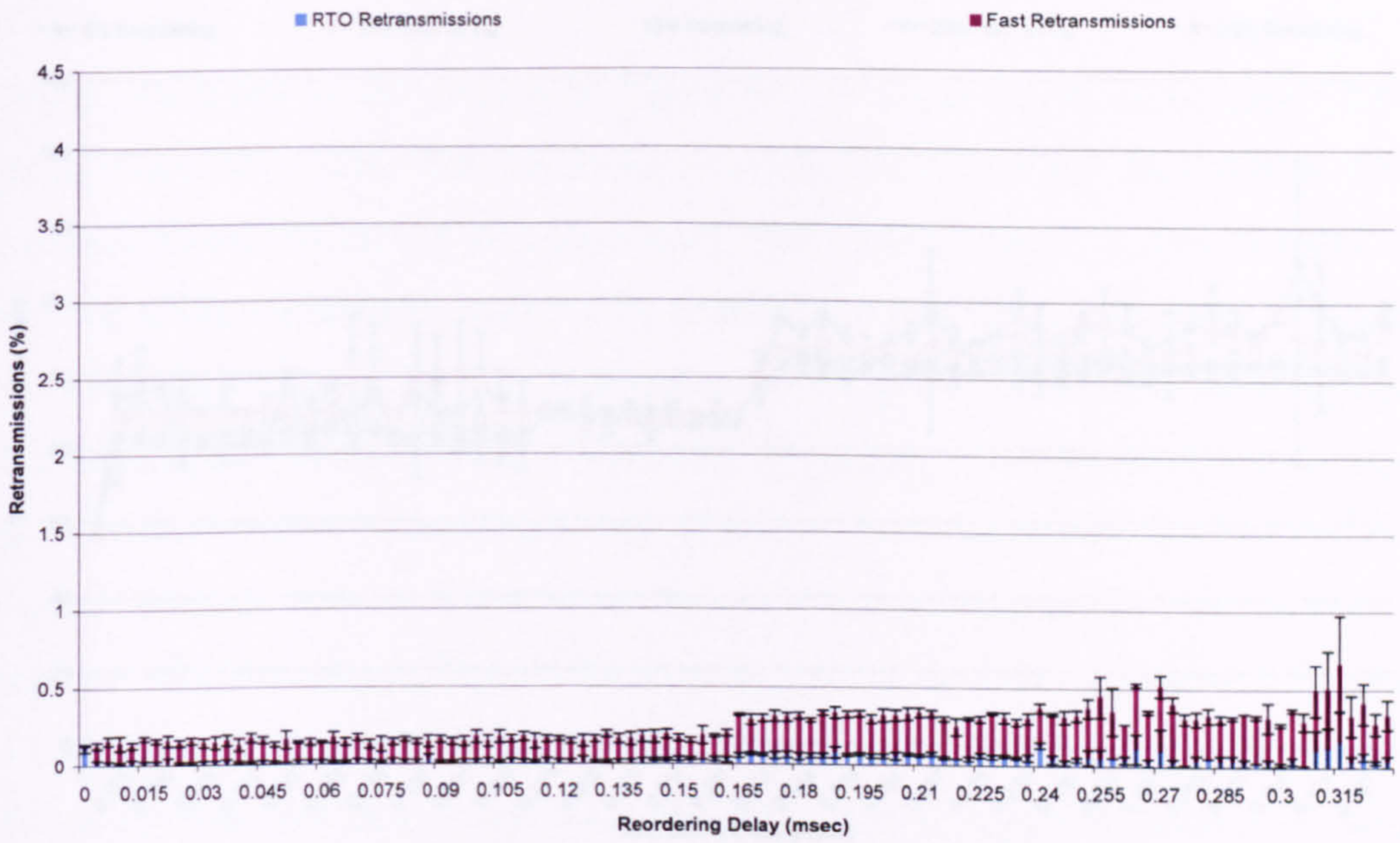

Figure 45- Percentage Retransmissions by Cause, 90% C.I., F_{150}(25%, various, 0, 0)

It is clear that, as for the shorter RTT variations, Fast Retransmissions make up the predominant number of retransmissions observed. For the higher percentage Reordering Probabilities, there is a clear indication of an increase in the number of retransmissions at approximately 0.243 msec. This correlates with Figure 39, where at the same Reordering Delay, the graph transitions from the second region to third region and Transmission Time increases rapidly.

4.3.2.3 300 msec Round Trip Time

There is some concern [Laor02] that packet reordering will have serious effects in high bandwidth-delay-product networks, such as satellite communications, where the high number of in-flight packets will offer greater opportunities for reordering to occur.

In Figure 46, forward path reordering is performed on an emulated link of 300 msec RTT. Note that the range over which Reordering Delays have been applied has been increased in order to find the regions where the effects of reordering can be measured. Figure 47 plots the percentage of reordered packets as defined by Equation 1.

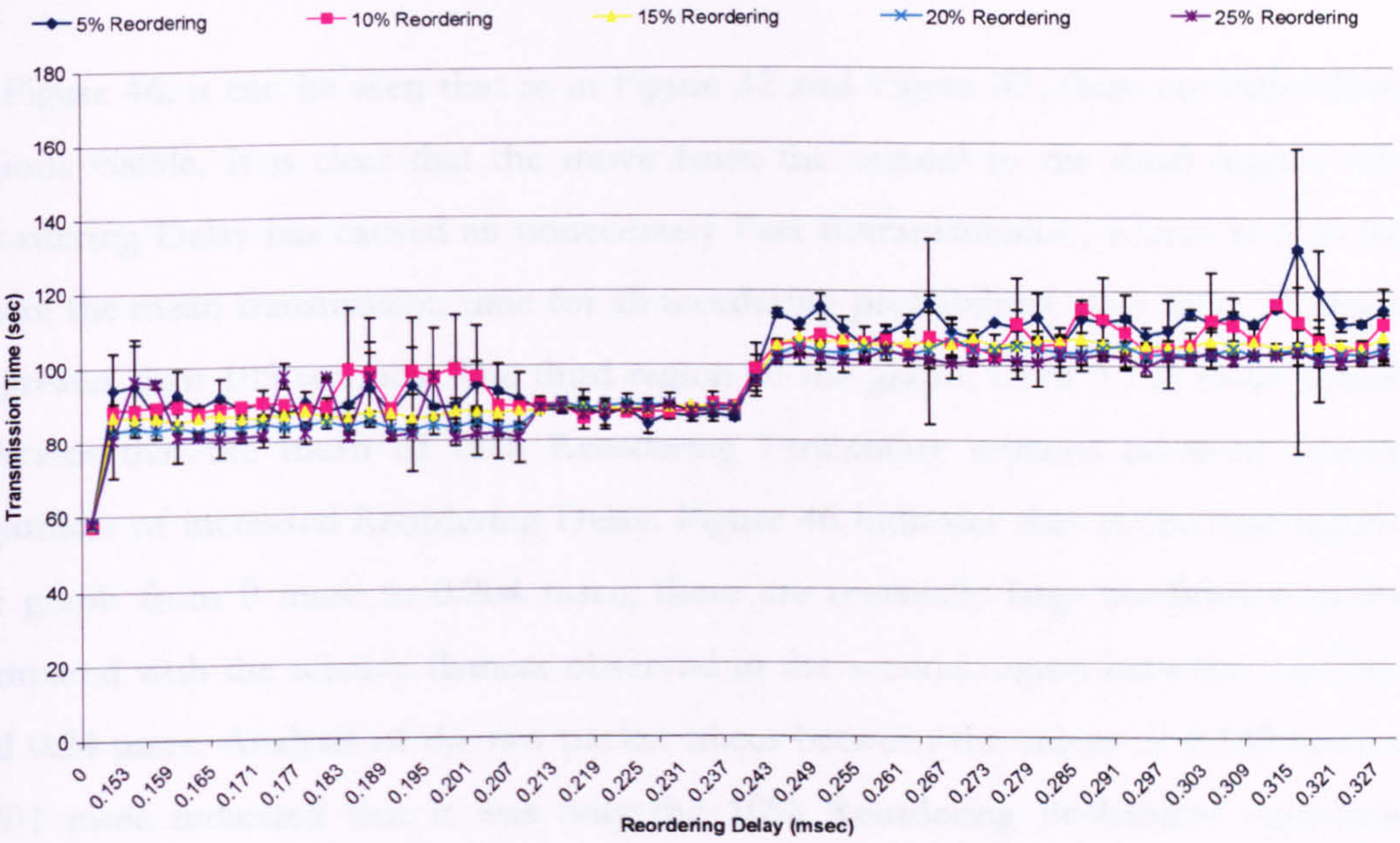

Figure 46 - Mean transmission time of 10 Megabytes, 90% C.I., F_{300}(various, various, 0, 0)

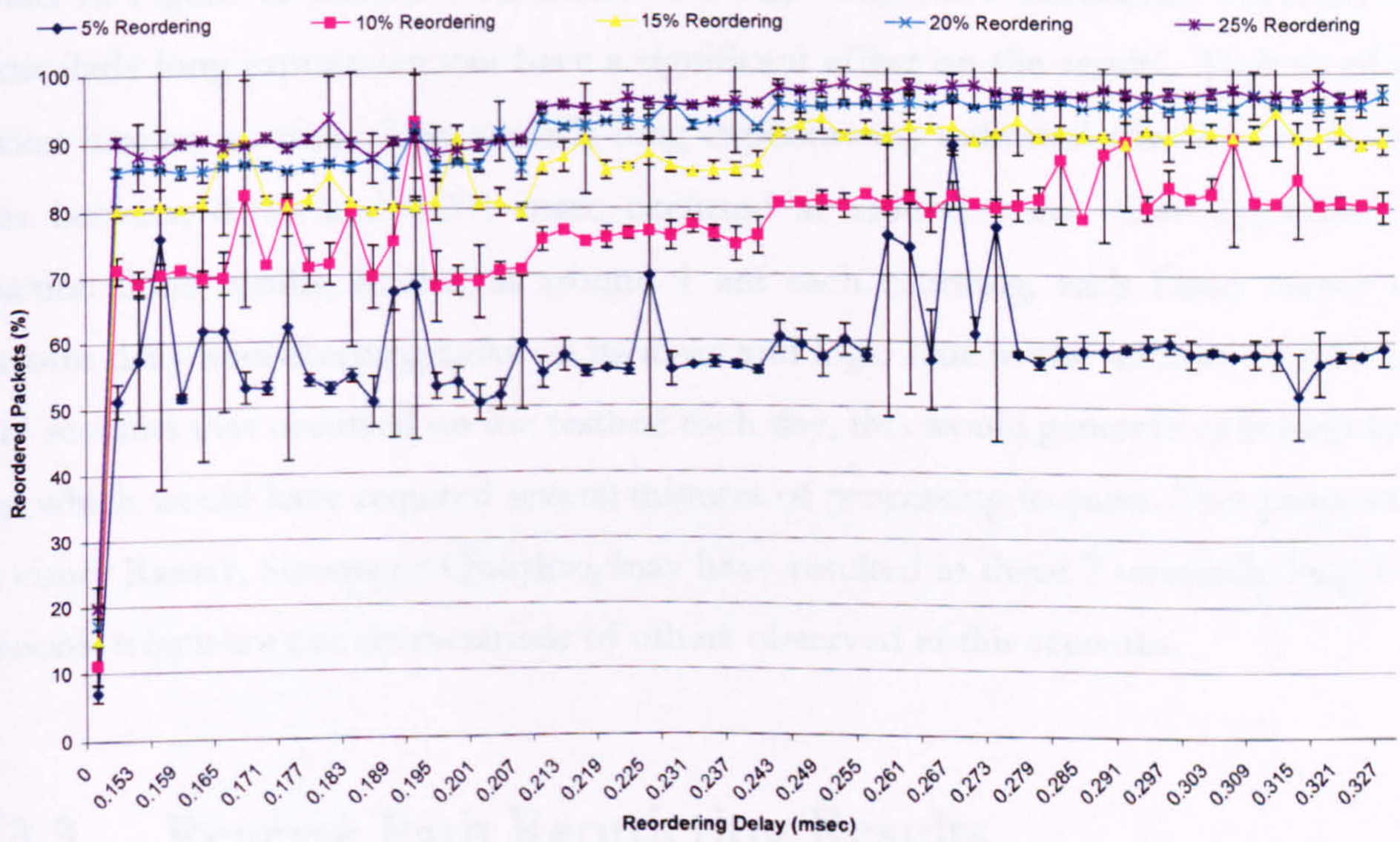

Figure 47 - Percentage Reordered Packets, 90% C.I., F_{300}(various, various, 0, 0)

Comparison of Figure 46 with previous Transmission Time graphs over shorter RTT distances indicates that the behaviour is more predictable, with significantly smaller confidence intervals. As previously discussed, the degree of Reordering Probability has little further effect on the degradation of the Transmission Time, despite Figure 47 indicating that for 25% packet reordering, over 90% of packets are always reordered.

In Figure 46, it can be seen that as in Figure 32 and Figure 39, there are three distinct regions visible. It is clear that the move from the second to the third region, where Reordering Delay has caused an unnecessary Fast Retransmission, occurs at 0.24 msec where the mean transmission time for all reordering probabilities rises from 90 seconds to greater than 105 seconds. The third region of the graph, from 0.243 msec upwards, indicates that the mean of each Reordering Probability remains relatively constant, regardless of increased Reordering Delay. Figure 46 indicates that in the first region of the graph from 0 msec to 0.204 msec, there are unusually large confidence intervals compared with the relative flatness observed in the second region between 0.204 msec and 0.24 msec. Analysis of the raw packet traces between the values of 0.183 msec and 0.201 msec indicated that it was only the 10% Reordering Probability experiments which were exhibiting these wide confidence intervals, due to one set of unusually long experiments, where each run would take approximately 115 seconds to complete. As the results in Figure 46 illustrate the mean of 5 runs with 90% confidence intervals, one particularly long experiment can have a significant effect on the results. Analysis of the packet headers of these 7 particularly long experiments, indicated that the set of 10% runs between 0.183 and 0.201 msec occurred at around 1 am. One hypothesis to describe these results, in that at around 1 am each morning, each Linux server will perform daily housekeeping tasks on its disks and logs. Due to the number of FTP and SSH sessions that occurred on the testbed each day, this would generate extremely large logs which would have required several minutes of processing to parse. This processing, on either Raasay, Stroma or Quoyloo, may have resulted in these 7 unusually long FTP sessions, which are not characteristic of others observed in this scenario.

4.3.3 Reverse Path Reordering Results

Reverse path TCP packet reordering, or Acknowledgement reordering, has received very little attention in the literature. Previous studies have concentrated on the data path as it is expected that this is where the majority of packet reordering effects will be observed. It has been hypothesised by Bennett that the major effect of reverse path reordering is an increase in burstiness, although this has never been confirmed in the literature by measurement or simulation.

Continuing the methodology developed for the Forward Path experiments discussed in the previous section, multiple experiments were performed to measure the effects of Reverse Path reordering over a number of emulated RTT, for various reverse path Reordering Probabilities and Reordering Delays.

For RTT emulations of 50 msec and 100 msec, the effects of reverse path reordering for a wide range of Reordering Delays, were found to be negligible. In these experiments, the 10 Megabyte TCP connection would quickly grow to fill the bandwidth delay product of the link, and the Transmission Time would complete in close to the Matthis ideal throughput for that particular RTT. For short RTT connections, it can be assumed that the effects of reverse path TCP packet reordering are negligible.

4.3.3.1 150 msec Round Trip Time

Figure 48 illustrates the Transmission Time for 150 msec RTT connections emulated with various degrees of reverse path Reordering Delays, and Reordering Probabilities. 150 msec is chosen to allow comparison with the equivalent forward path reordering results illustrated in Section 4.3.2.2.

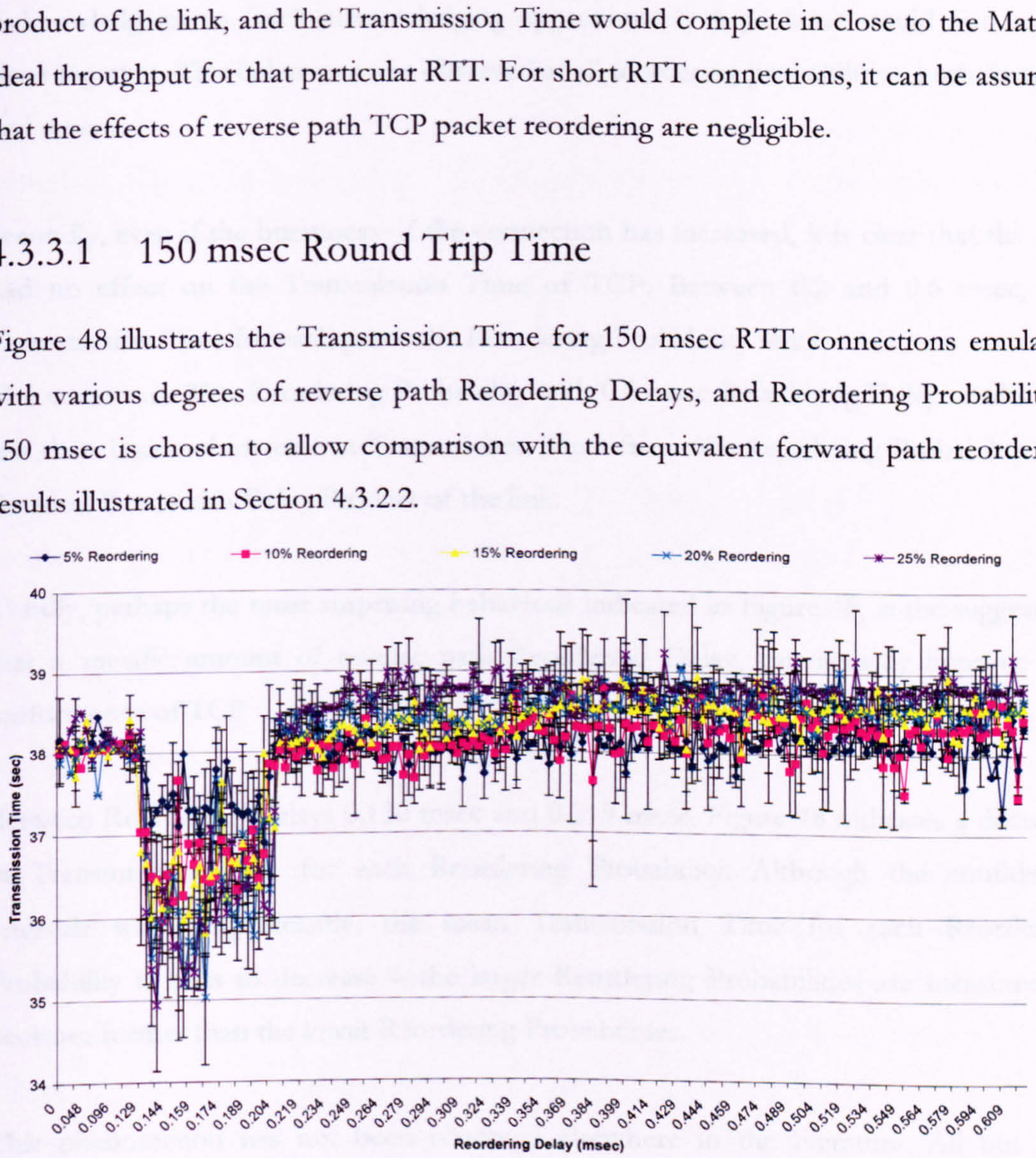

Figure 48 - Mean transmission time of 10 Megabytes, 90% C.I., F_{150}(0, 0, various, various)

This graph illustrates a number of surprising results which have not been measured or observed previously.

Firstly, for this emulated experiment of RTT 150 msec, it can be observed that Transmission Time is relatively constant between 0.2 and 0.6 msec Reordering Delays. Although Bennett hypothesised that reverse path reordering would cause highly bursty behaviour, it should be noted that TCP is already a bursty protocol - evaluation of packet traces indicates that volleys of packets are launched into the network by a sending TCP, sometimes as many as 45 at a time, and the responding Acknowledgements (each acknowledging approximately 6 packets) would also arrive close together. This behaviour was observed at all reordering probabilities, both forward and reverse.

Secondly, even if the burstiness of the connection has increased, it is clear that this has had no effect on the Transmission Time of TCP. Between 0.2 and 0.6 msec, the Transmission Time for each particular Reordering Probability remain relatively constant. The worst case 25% Reordering Probability with 0.6 msec Reordering Delay results in a less than 1 second increase in Transmission Time from 0% Reordering Probability and the ideal Bandwidth Delay Product of the link.

Thirdly, perhaps the most surprising behaviour indicated in Figure 48, is the suggestion that a specific amount of reverse path Reordering Delay, can actually improve the performance of TCP.

Between Reordering Delays 0.130 msec and 0.219 msec, Figure 48 indicates a decrease in Transmission Time for each Reordering Probability. Although the confidence intervals widen significantly, the mean Transmission Time for each Reordering Probability is seen to decrease – the larger Reordering Probabilities are measured to decrease further than the lower Reordering Probabilities.

This phenomenon has not been observed elsewhere in the literature. All but one publication [Negl04] have been based on the assumption that either forward or reverse path TCP packet reordering will result in a degradation of service quality, and that it is

something which should be avoided when possible. Neglia's [Negl04] NS-2 simulation demonstrated that specific degrees of forward path reordering resulted in improved performance, but these were due to the specific RED configuration simulated within the mid-point routers, thus correlating reordering with the effects of loss and resulting in an improved performance of the RED algorithm.

The reason for this improvement in performance can be explained by examination of the packet traces in these experiments. These indicate that when Acknowledgement reordering occurs, the self-clocking control loop of TCP quickly breaks down. At the points in Figure 48 where the Transmission Time has decreased, this is because the Acknowledgement Reordering has allowed a larger Acknowledgement to arrive before the previous smaller Acknowledgement, signalling to the sending TCP that it is permitted to launch a large volley of packets into the network. The result of this is that the Congestion Window at the sending TCP is effectively allowed to grow much faster than it otherwise would, and so the sending TCP is able to grow to the Bandwidth Delay Product of the link much faster.

As discussed in Section 4.3.1, it can take as long as seven seconds for the sending TCP to probe the Bandwidth Delay Product of the link during normal operation. Reordered Acknowledgements resulted in the sending TCP growing its congestion window to 45 packets in under 3 seconds, during the periods of improved performance. In this particular set of experiments, with a RTT of 150 msec and the mean Transmission Time in normal circumstances of approximately 38 seconds, this time saving in achieving the maximum link throughput results in a measurable performance improvement of TCP.

Figure 49 to Figure 53 perform retransmission analysis on the 150 msec RTT reverse path reordering experiments, indicating that Retransmissions are caused exclusively by timeout at the Sender, rather than Fast Retransmissions. This is also an interesting observation and assists in explaining why the Transmission Times illustrated in Figure 48 remain constant over a high range of Reordering Probabilities and Delays. As only reverse path reordering is being applied, packets are arriving at the receiving TCP in order. Therefore, all Acknowledgements are being generated in order too. As Acknowledgements pass through the Click router they are being reordered, some to a

very large delay, but not to the extent that they are moved more than 3 positions out of sequence thus causing a Fast Retransmission. The fact that no Fast Retransmissions are being signalled results in the Sending TCP assuming that there is loss, but no congestion, on the end-to-end path. Therefore, the sending TCP does not implement the Fast Recovery algorithm and scale back the congestion window and steady state threshold. The sending TCP continues transmission of packets at its full congestion window bandwidth delay product.

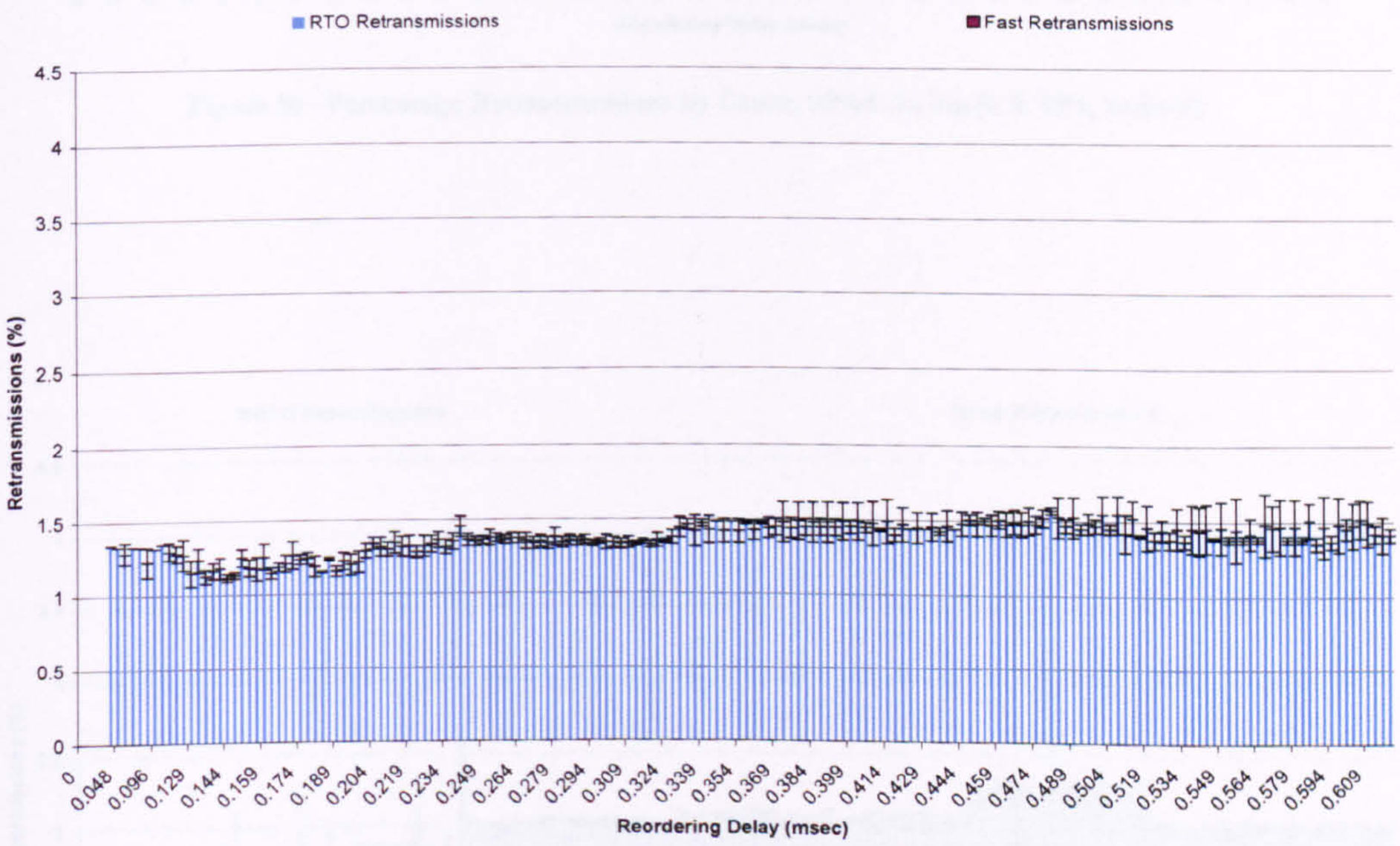

Figure 49 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(0, 0, 5%, various)

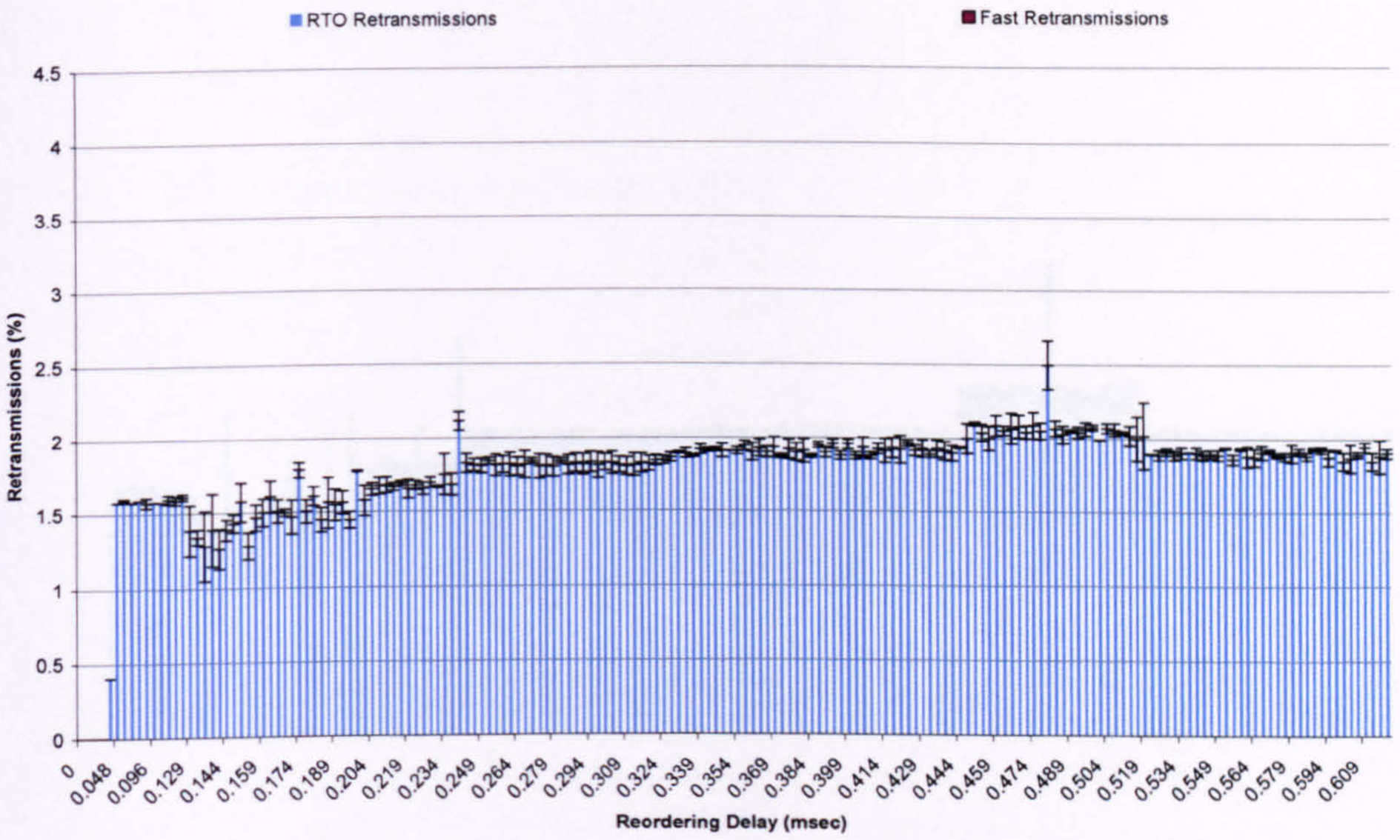

Figure 50 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(0, 0, 10%, various)

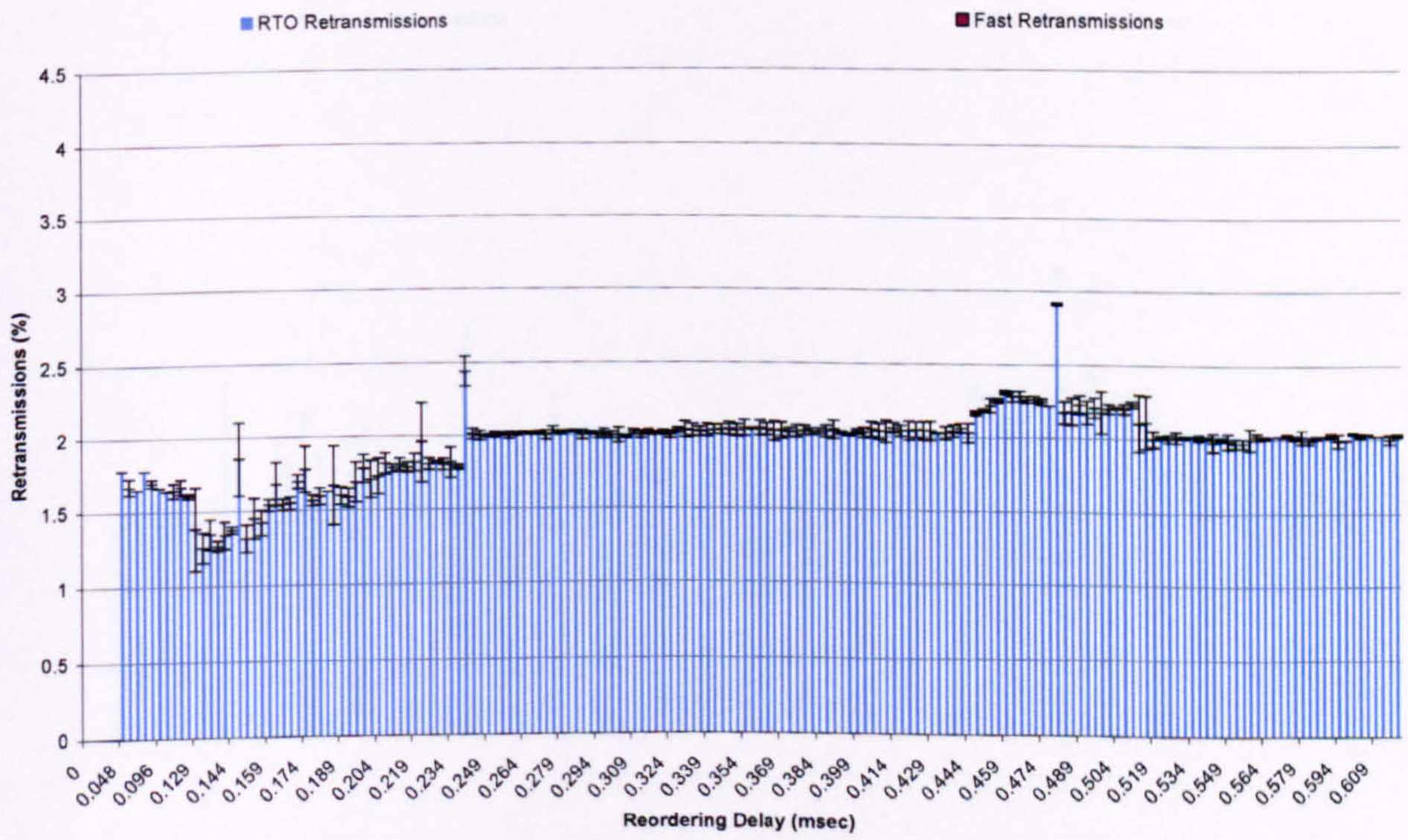

Figure 51 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(0, 0, 15%, various)

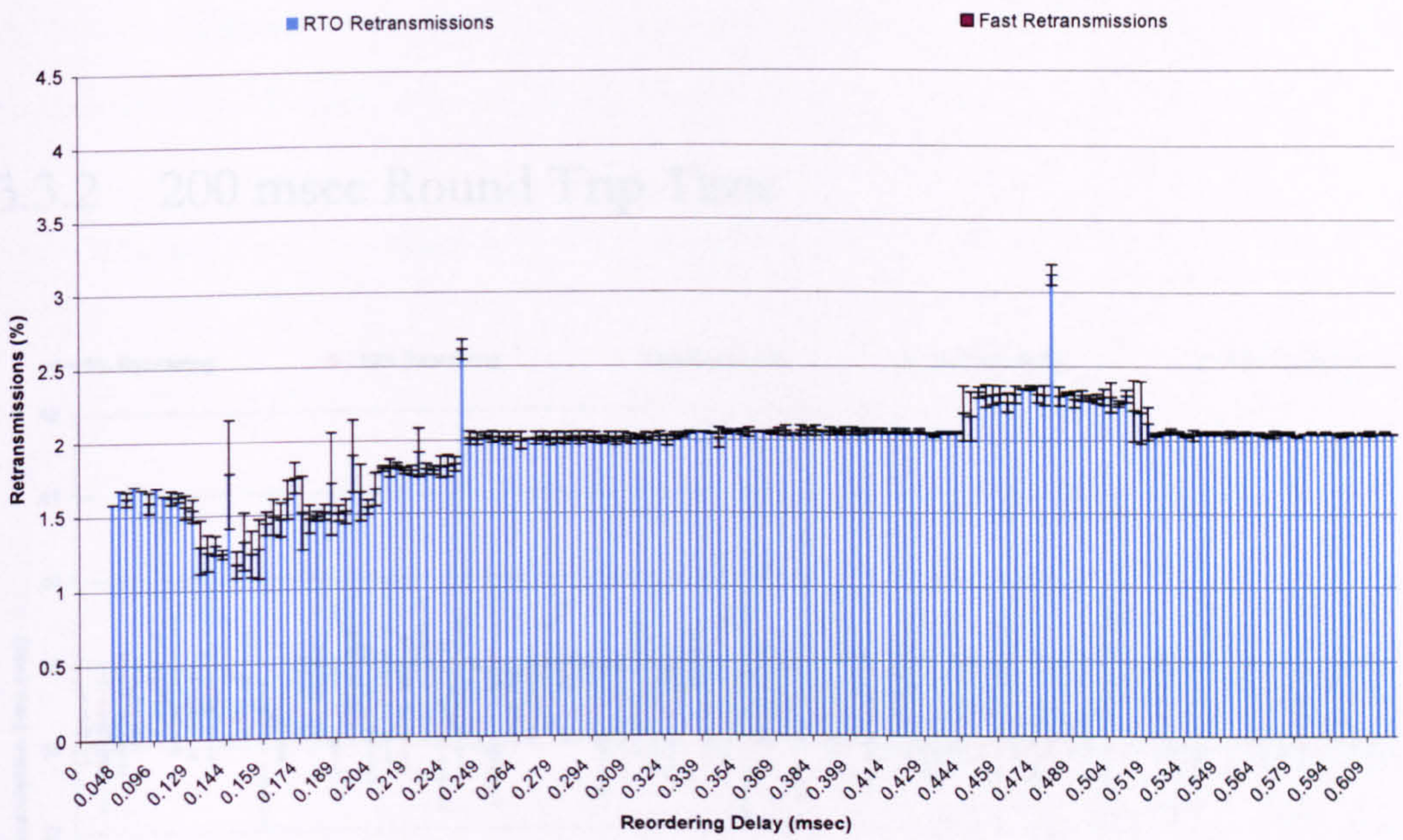

Figure 52 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(0, 0, 20%, various)

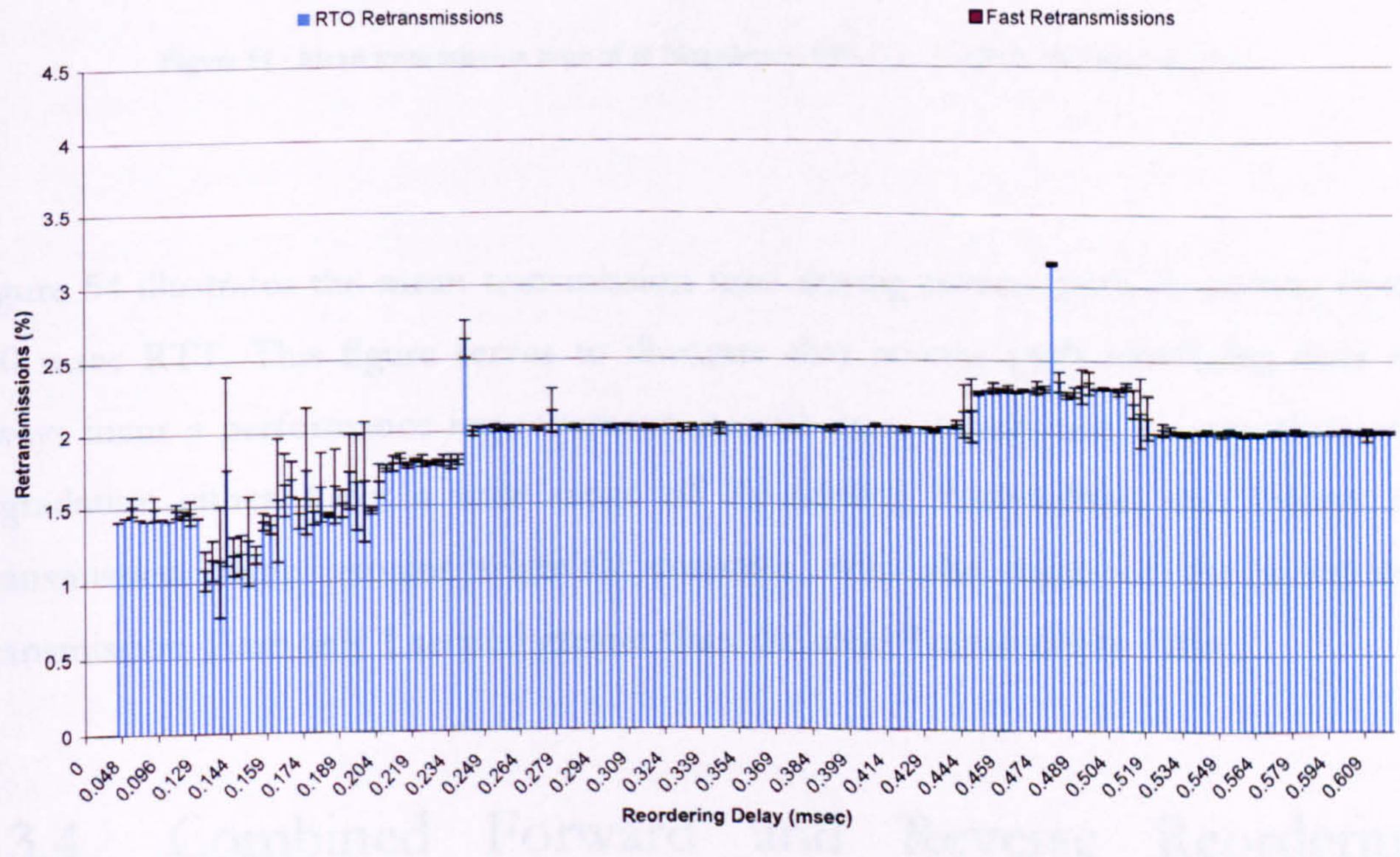

Figure 53 - Percentage Retransmissions by Cause, 90% C.I., F_{150}(0, 0, 25%, various)

4.3.3.2 200 msec Round Trip Time

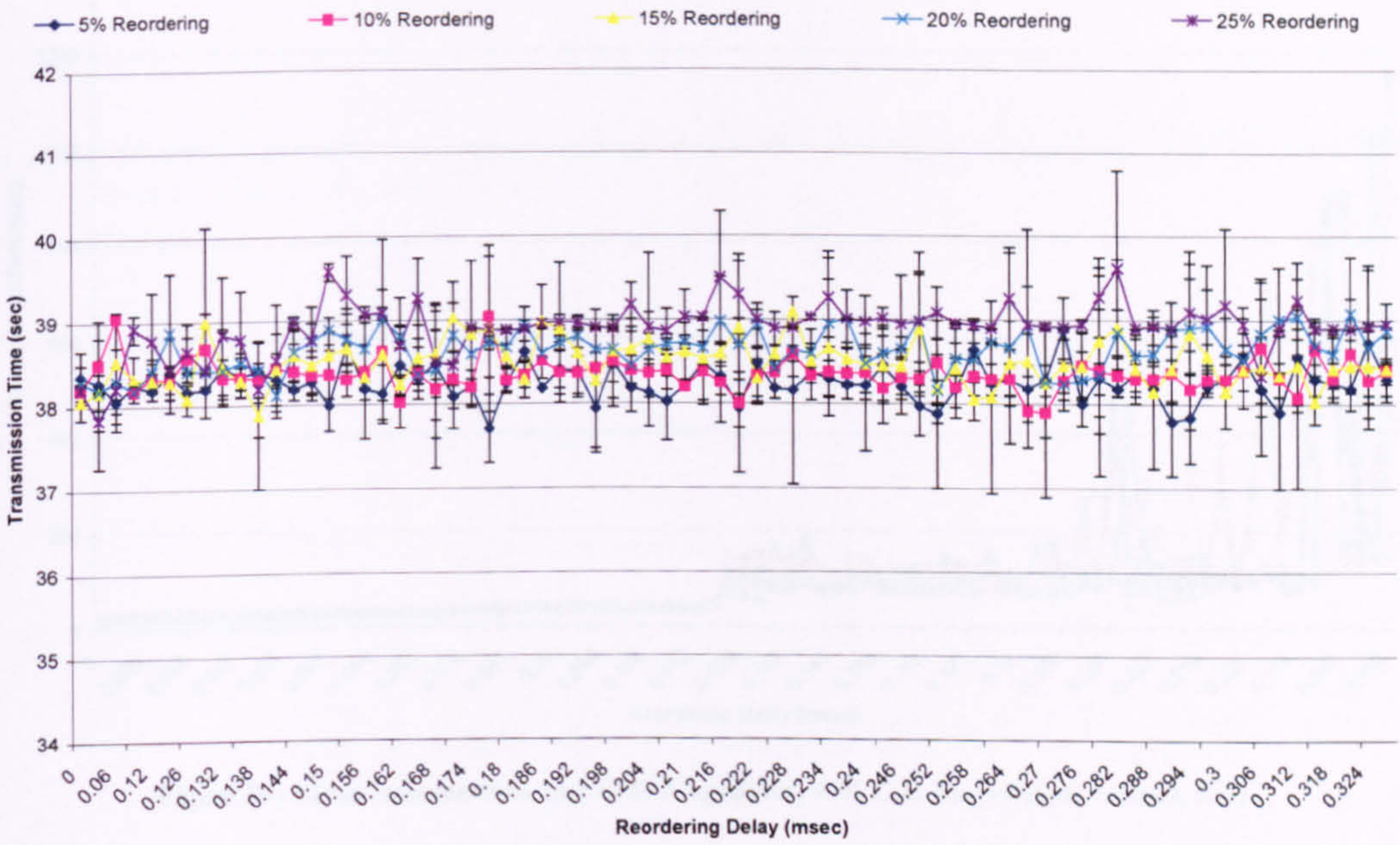

Figure 54 - Mean transmission time of 10 Megabytes, 90% C.I., F_{200}(0, 0, various, various)

Figure 54 illustrates the mean transmission time during reverse path reordering over a 200 msec RTT. This figure serves to illustrate that reverse path reordering does not always incur a performance improvement; nonetheless, it does not incur performance degradation either. Over a wide range of Reordering Probabilities and Delays, the Transmission Time remains relatively constant, with the mean of the worst case Transmission Time only 1 second greater than the ideal Transmission Time.

4.3.4 Combined Forward and Reverse Reordering, 100ms RTT

The effects of combined forward and reverse path reordering were investigated. Figure 55 illustrates Transmission Time for a 100 msec RTT, where a constant reverse path reordering of 15% Reordering Probability at 0.34 msec Reordering Delay was applied.

100 msec is illustrated as it demonstrates the mid-range performance, allowing for comparison with 50 msec and 150 msec forward path reordering results.

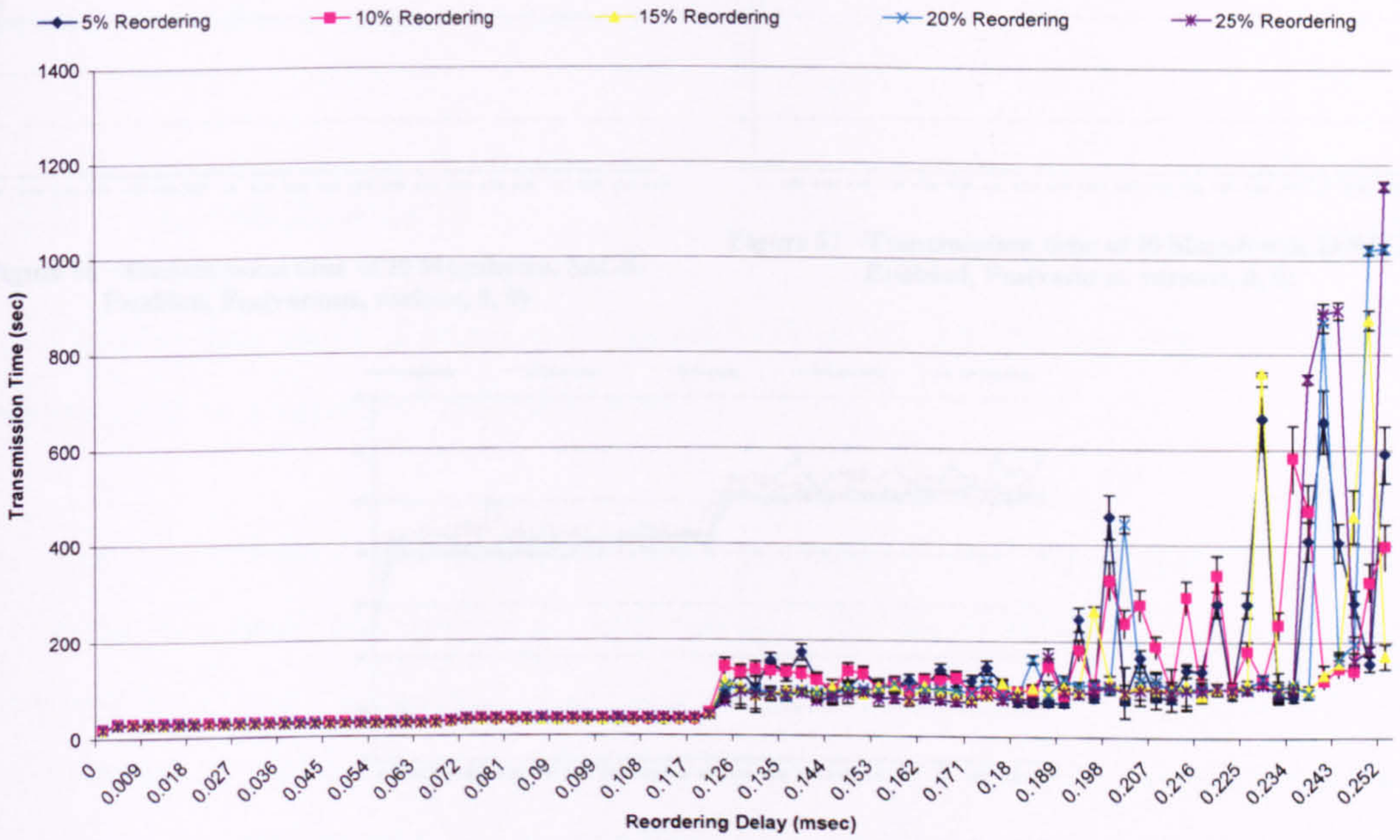

Figure 55 - Mean transmission time of 10 Megabytes, 90% C.I., F_{100}(various, various, 0, 0)

As can be seen, the effects of forward path reordering dominate the behaviour of the TCP transmission. This suggests that when considering the effects of packet reordering, it is the reordering on the forward path which has the potential for severe degradation of service quality. As illustrated in the previous Section, reverse path reordering, whether in isolation or combined with forward path, will not result in the significant throughput losses that forward path reordering could incur.

4.3.5 Comparison of Methods to Combat Reordering

A number of extensions to TCP have been proposed in order to mitigate the effects of packet reordering. In this section, a brief investigation is carried out on three of these potential methods.

Figure 56 to Figure 58 illustrate the Transmission Time for forward path packet reordering over a 150 msec RTT path.

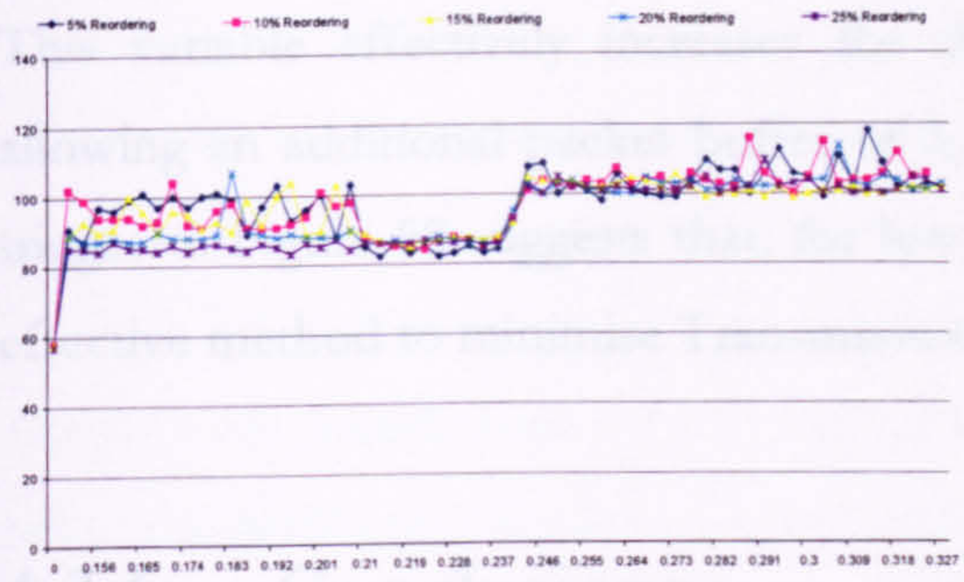

Figure 56 –Transmission time of 10 Megabytes, SACK-Enabled, F_{150}(various, various, 0, 0)

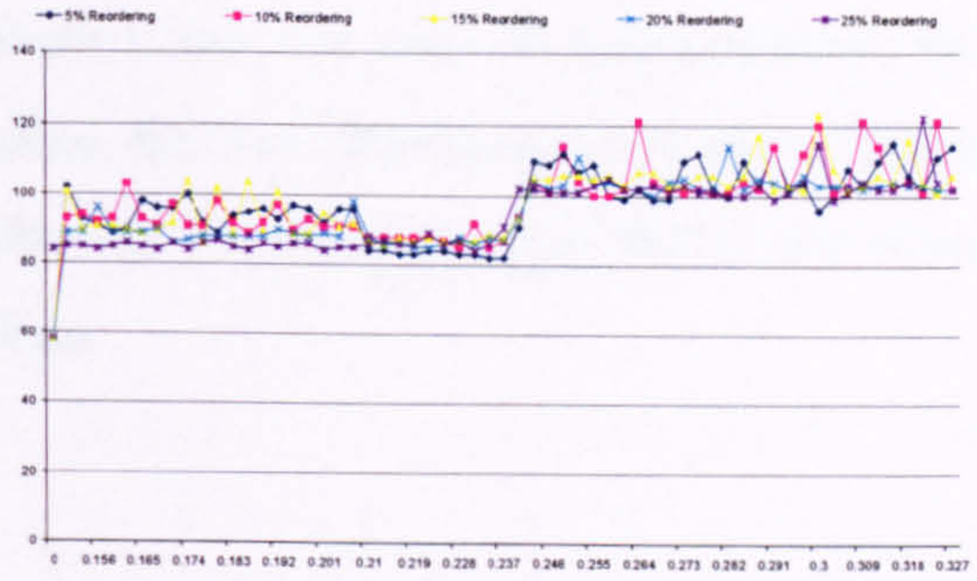

Figure 57 –Transmission time of 10 Megabytes, D-SACK-Enabled, F_{150}(various, various, 0, 0)

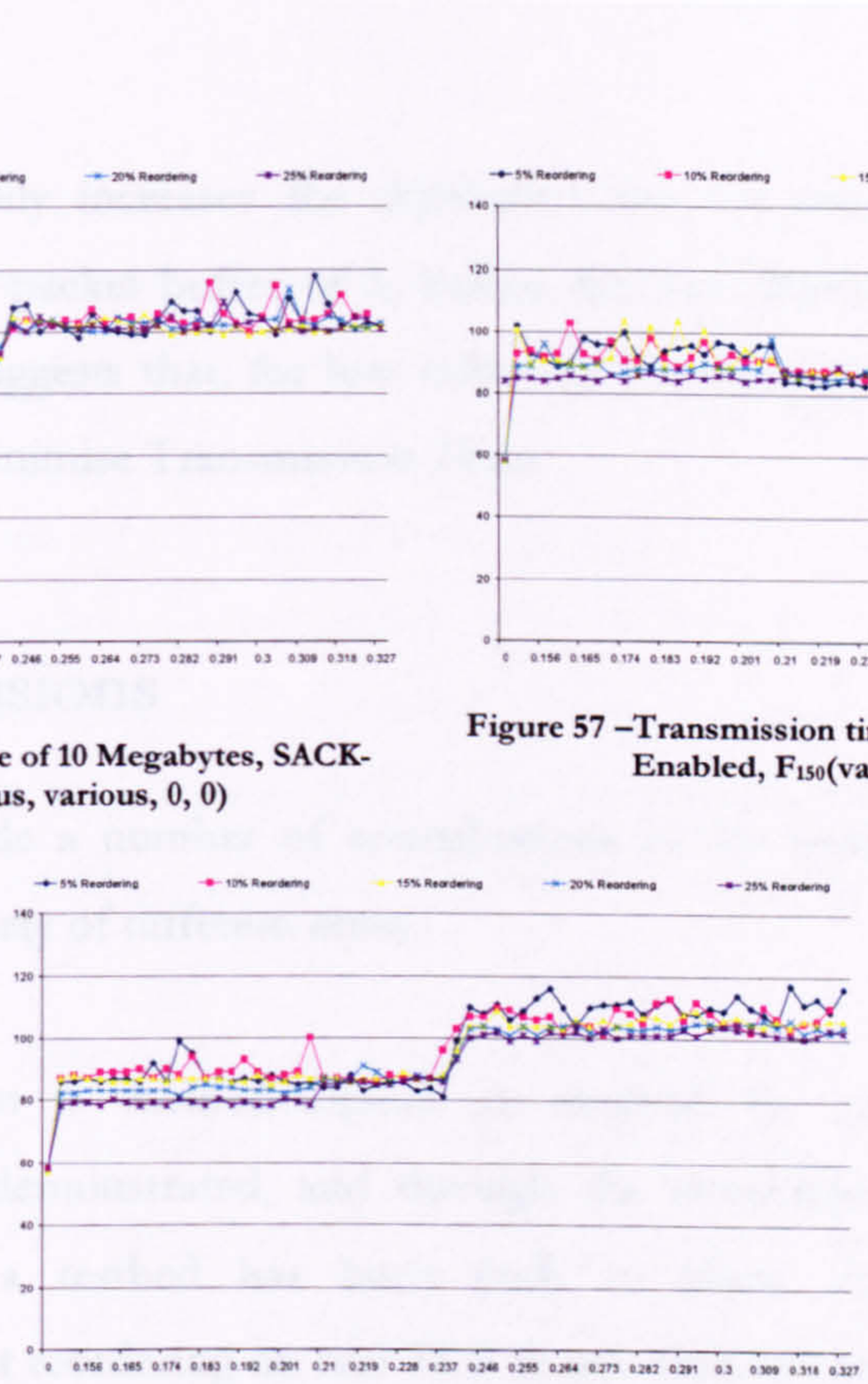

Figure 58 - Transmission time of 10 Megabytes, tcp_reordering = 3 Enabled, F_{150}(various, various, 0, 0)

Figure 56 illustrates performance when Selective Acknowledgements are enabled. This may be compared with Figure 39 in order for comparison of a similar RTT, but when SACK is disabled. Figure 56 suggests that there is little improvement in performance for all Reordering Probabilities at low Reordering Delays, until the threshold where a Fast Retransmission will always be invoked at 0.243 msec. Both graphs display the same step change in Transmission Time, suggesting that SACK is not able to perform any better than Reno TCP during the extreme reordering tests applied in this study.

Figure 57 illustrates a 150 msec RTT with D-SACK [DSAC08] enabled. D-SACK allows a receiver to communicate to a sender when a retransmission was unnecessary and, therefore, tune the TCP *dupthresh* variable to avoid false fast retransmits. Figure 57 suggests that there is little improvement achieved by D-SACK compared with standard SACK, under extreme reordering conditions.

Figure 58 illustrates a 150 msec RTT with the Linux kernel variable *tcp_reordering* set to 3.

This variable effectively increases the *dupthresh* value for Fast Retransmissions, by allowing an additional packet buffer of 3, before the Fast Retransmission algorithm is triggered. Figure 58 suggests that, for low values of Reordering Delay, this is the most effective method to minimise Transmission Time.

4.3.6 Conclusions

This Chapter has made a number of contributions in the field of packet reordering measurement, in a variety of different areas.

The first contribution is methodological. A method for emulating TCP packet reordering has been demonstrated, and through the development of software for a configurable router, a testbed has been built to allow the demonstration and measurement of packet reordering on real TCP flows. This has been done to provide an insight into the behaviours of the congestion and retransmission algorithms, thereby demonstrating that reordering can have both positive and negative effects on TCP performance.

A two point passive measurement technique has been developed, which has allowed more accurate measurement than previous studies of packet reordering, by exploiting the use of the IPID field as a method to determine the sending sequence of a TCP connection. Simple metrics have been developed that exploit this IPID field, thereby allowing determination if a packet has been reordered and the extent by which that packet has moved. Under the high degrees of reordering measured in the testbed, this method has provided a lightweight and simple method for determining the Absolute Reordering of a packet and avoids the calculation of future Sequence Number based on current payload lengths. Although this method may not be applicable in the wider Internet where fragmentation may occur, it does allow a method for reordering in a controlled environment and could have future applications in the testing of specific reorder-inducing routers or paths.

The two point measurement technique has allowed determination of the cause of retransmissions, which are the by-product of packet reordering effects on TCP. By

correlating the packet traces obtained at two points, it has been possible to investigate and classify each retransmission, thus providing a more complete analysis of the effects of packet reordering, compared to previous measurement studies [Laor02].

The second contribution of this chapter is the measurements obtained using this two point methodology. Separate investigations have been performed on the effects of Forward Path, Reverse Path and combined forward and reverse path reordering. Transmission Time has been chosen to be the metric used to describe the Quality of Experience that a user could expect under each packet reordering environment.

The study of forward path packet reordering has indicated that the effects of packet reordering are negligible with respect to Reordering Probability. Reordering Delay is the dominant factor in determining the effect of packet reordering on a particular TCP connection. The sudden drop in throughput measured during these experiments is an important feature of TCP's tolerance to reordering. The drop in throughput is continuous and sustained for all reordering probabilities above a threshold reordering delay, regardless of additional reordering applied, providing an insight into the behaviour of TCP's control algorithms. As the reordering delay is increased, the mean transmission time tends also to increase, but with a significantly larger error range. This indicates that, unlike percentage loss which guarantees retransmissions and invocation of congestion control, packet reordering is a complicated function of buffer sizes, control algorithms and Delay Variation, which may or may not have a significant effect on a TCP flow.

These results compare favourably with previous reported investigations into packet reordering [Blan02], where a command was implemented in NS-2 to swap two elements of a router's input queue at a given time. Their results illustrated a similarly initial steep decline in throughput as a function of queue swaps, with little additional impact caused by increasing the frequency of reordering events, and an eventual flattening out of throughput. Forward path reordering experiments in this chapter also indicate this behaviour, which can be attributed to there being a point where reordering always causes a needless fast retransmit and a halving of the congestion window. Once a spurious fast retransmit is triggered, the sending rate is reduced in a uniform fashion

and, after this point, reordering has little additional effect.

In the only other measurement study of TCP reordering [Laor02], the authors used an Agilent QA Robot to randomly delay packets by 3 positions. This does not allow investigation of the Reordering Delay, nor investigation of the Reordering Delay with respect to RTT.

For every RTT, it has been found that there is a forward path maximum reordering delay threshold which can be applied to packets, regardless of percentage reordering, below which reordering has negligible effects. Determination of this threshold, on a specific path, is key to ensuring that a specific switch or router does not introduce reordering to such an extent that it causes unnecessary retransmissions and an associated reduction in throughput.

This chapter has performed the first measurement study of reverse path packet reordering and has demonstrated surprising results. Contrary to assumptions in the literature, it has been measured that reverse path reordering has little additional negative effect on the throughput of a connection. Indeed, it has been measured that in specific circumstances, as a function of the RTT, a function of the amount of data to be transmitted in the flow and a function of reordering delay, reverse path reordering can actually be beneficial for a connection. This phenomenon was explained due to the loss of self-clocking during Acknowledgement resequencing, thus allowing the sending TCP *cwnd* to grow faster than normal.

The first measurement study of combined path reordering has also been performed, which has illustrated that the effects of forward path reordering dominate the behaviour of the connection.

The use of percentage reordered packets as a metric, has been shown to be difficult to correlate with the actual performance of a TCP connection. This suggests that many of the metrics proposed in the literature, such as RFC 4737, are difficult to apply in the context of Quality of Experience. Percentage Retransmissions, as a function of Reordering Delay, has been demonstrated as a more effective technique of representing

the performance of a connection. This, combined with Transmission Time, both illustrate the Reordering Delay at which a measurable effect in performance will occur.

In forward path reordering measurements, it was found that for all RTT, the majority of retransmissions were due to Fast Retransmission requests sent by the receiver. This suggests that a more conservative approach to adjusting the *dupthresh* Fast Retransmit threshold, would result in significant improvements during reordering. This hypothesis was confirmed by investigation of the *tcp_reordering* variable, which was demonstrated as shown in Figure 58, to provide a near 20% performance improvement when compared to SACK and D-SACK, at low levels of reordering.

This Chapter has performed one of the largest studies of TCP packet reordering to date, emulating over 30,000 FTP sessions over a six month period. It has presented the need to develop an autonomous measurement system to perform such a large study and demonstrated the methods to perform data management and processing of such large amounts of packet captures.

Clearly there is motivation for NEMs and Operators to increase the amount of parallelism prevalent on the internet. This study has shown that percentage reordering itself does not cause problems on a TCP connection; Reordering Delay is the method which should be used to measure if a particular piece of equipment is likely to cause problems with the efficient operation of TCP.

Chapter 5

Mid-Point Passive Monitoring of TCP Flows

5.1 Introduction

The classification taxonomy presented in Chapter 3 describing the current methodologies and metrics which can be used to measure packet reordering, has illustrated the range of techniques that have been developed by previous work and described in the literature. The diversity of these techniques has made it difficult to

compare the various measurement studies, and this has then led to an incomplete picture of the extent of packet reordering observable in the Internet today.

Chapter 4 is a study on the effects of packet reordering on a TCP connection and has argued that, for any metric of packet reordering to be useful, it must be able to characterise the effects that will be experienced by a real flow. Chapter 4 has further demonstrated that it is difficult to correlate percentage reordering metrics with the measurable effects of packet reordering.

The taxonomy in Chapter 3 indicates the two main classifications of all Internet measurements; active and passive techniques. Chapter 2 has argued the benefits of passive measurements while Chapter 3 has presented examples of these. Clearly the significant benefit of using a passive measurement methodology is that it can be used to measure the behaviour of many thousands of concurrent flows; in order to achieve this and to allow observation of large volumes of traffic, the passive measurement must be a mid-point technique.

Jaiswal, Rewaskar and Tstat have each presented mid-point passive measurement techniques, which have been evaluated in Chapter 3. Both Rewaskar and Tstat are subsets of the work of Jaiswal and design classification algorithms which each classify out-of-sequence packets based on inferred knowledge observed around that packet.

Each of the mid-point passive packet reordering techniques proposed in the literature, makes several assumptions that could significantly affect their operation. Each assumes the ability to be able to calculate the RTT of every concurrent flow observed. Additionally, each also assumes the ability to observe and correlate both Data and Acknowledgement packets, in order to be able to explain each out-of-sequence event. This requirement for symmetric data and acknowledgement paths, has been acknowledged by the authors themselves as a significant limitation of their research[Jais07].

This chapter discusses the challenge of developing a mid-point passive real time monitor of TCP flows which does not require Data and Ack symmetry, nor estimation

of RTT. A lightweight, mid-point methodology and classification algorithm is developed and applied to live Internet traffic in order to gauge performance when compared with Jaiswal. Finally, a technique for the visualisation of a TCP flow's performance is presented. This proposed technique is superior to others; it allows simple evaluation of the degree of resequencing occurring within a TCP connection over time, improving on the metrics presented in RFC 5237.

5.2 Large Scale Monitoring of TCP Flows

The Internet and its millions of users now depend upon the reliable operation of TCP. Clearly, some mechanism for large-scale monitoring of TCP would be extremely valuable in ensuring that performance is optimised, faults are easily identifiable, service level agreements are maintained by Internet Service Providers and end users can expect a guaranteed high Quality of Service.

There are, however, several difficulties when performing large scale mid-point monitoring of TCP flows. Firstly, the majority of flows are often short-lived. Secondly, there will be many millions of concurrent active flows observable from a mid-point position. Thus, any monitoring technique should be simple and lightweight to implement, should not consume costly processing and memory resources and should also be scalable, so that it can be applied to many simultaneous concurrent flows and deployed in many places around the Internet, without causing additional overhead itself.

5.2.1 Single Point Measurement Techniques

A single point measurement technique is desirable from a network monitoring perspective. Multi-path routing and link parallelism will cause an asymmetry in TCP flows, where the outgoing data packets may choose a different route from the resulting Acknowledgement packets, as illustrated in Figure 59. Therefore, a single point measurement technique should allow measurement of either Data or Ack packets at geographically separate points, without the need to correlate these measurements for processing and analysis. Furthermore, single point measurements should have no

requirement for time synchronisation between measurement probes and, therefore, are advantageous as they avoid the computationally expensive and bandwidth intensive real-time correlation of measurement data between probes. Time synchronising a large number of measurement devices over a very wide area and, to a high degree of accuracy, is a significant burden that should be avoided.

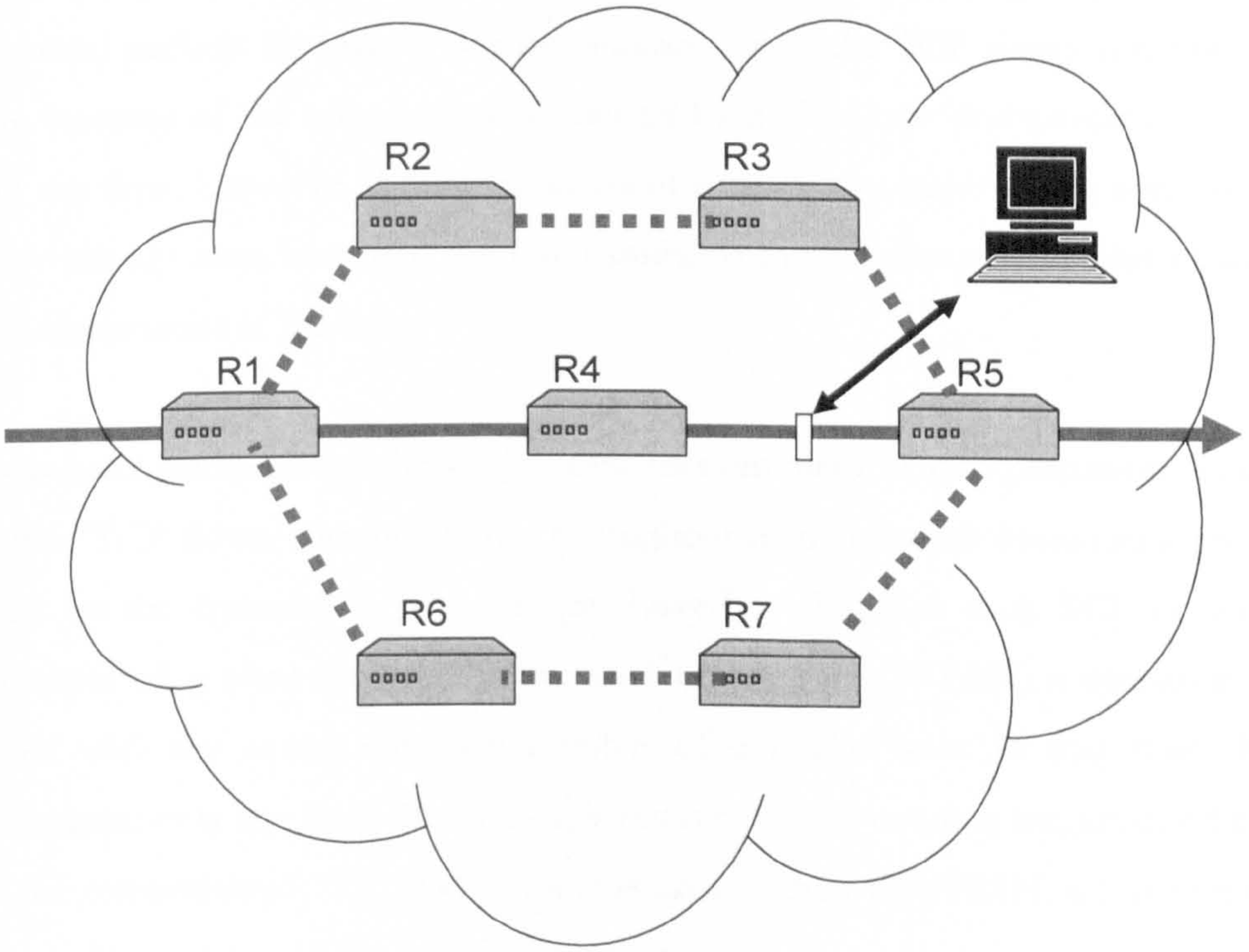

Figure 59 - Mid Point Network Monitoring

5.2.2 Goodput

The traditional measure used to monitor TCP performance is "Goodput", which is defined as the amount of data received versus the amount transmitted; in this work, the amount transmitted includes retransmissions caused by losses in the network. Goodput provides a simple method of indicating the health of a TCP connection, as retransmissions are an excess overhead on the network that should be avoided.

Goodput is recorded by measuring the volume of all the transmitted TCP payloads in a given flow by recording the first and last sequence numbers for a given direction in a flow. This provides the amount of traffic successfully transmitted and received during

the lifetime of that flow. The calculation is adjusted to make allowance for the SYN and FIN signalling packets that signal the start and end of the flow and increase the sequence number without transmitting any payload.

The conventional measure of TCP retransmission takes the sum of all TCP packet lengths actually transmitted and subtracts the goodput figure, giving a value for retransmitted packets. By making this measurement near the TCP source it produces an accurate measure of the retransmissions caused by packet loss 'downstream' from that point in the flow. However, if the measurement is made at a point where some packets may have already been lost, then the retransmission measurement will under-report the value by the amount of the loss.

A standardised methodology for single point measurement of retransmission, loss, and goodput of TCP flows, currently in use throughout many network monitoring products deployed on the Internet, is proposed by Love[Love06]. For each TCP connection being monitored, a Next Expected Sequence Number value (NESN) is maintained and compared with the actual sequence number of a packet seen in that flow. If the sequence number is less than the NESN, a retransmission count is incremented by the size of the retransmitted TCP payload; if it is greater then the NESN, a loss counter is incremented by the size of the lost TCP payload.

This technique, using analysis of the Sequence numbers, enables an observer at an arbitrary monitoring point on a TCP connection to estimate the traffic that was originally sent by the transmitting node, even though some of this traffic may have already been lost. However, this thesis has demonstrated that TCP traffic is not guaranteed to arrive at the observation point in the same order as which it was transmitted. Therefore out-of-sequence packets can be caused by a loss followed by retransmission or packet reordering. Clearly, when taking measurements at the transmission source, before the packets traverse a switching device, no packet re-ordering can have occurred and the traditional Love calculation for retransmission based upon subtracting goodput from throughput will suffice. However, any measurements using this method at subsequent later points in the network, after the packets have passed through multiple routers and switches, are likely to over-estimate

the amount of loss and retransmission by the amount of packet reordering that has occurred.

5.2.3 Jaiswal

Jaiswal's algorithm, as reviewed in Section 3.3.3, classifies out-of-sequence packets by observing properties of the forward path packets carrying the TCP segments observed, such as time of observance, the packet's IP ID field, the existence of the segments reverse path ACK packets, and some derived measures, such as the time difference between two occurrences of the same TCP segment. Each out-of-sequence packet is classified into one of five types; Retransmission, Unneeded Retransmission, Network Duplicate, Reordering and Unknown.

However, as part of the algorithm, Jaiswal relies on two important properties that can severely limit the number of flows which can be analysed at any particular mid-point. They must first observe the return path Ack packets, and secondly they require an accurate estimation of the senders RTO and RTT for every single observed flow. There are simply too many complex heuristics used in this method to make a simple, lightweight and reliable measurement. Jaiswal acknowledges that only approximately 13% of the monitored flows on the Sprint Tier-1 backbone were found to be symmetrical and therefore 87% of packets were not processed by the Jaiswal algorithm.

5.2.4 Summary

There is clear motivation for the development of accurate single point measurements of TCP loss, goodput, retransmission and reordering. The technique described by Love is clearly lightweight, simple to implement, and can perform measurement entirely on the data path of a connection, but will grossly overestimate retransmission count during periods of packet reordering. The Jaiswal algorithm will provide a much improved determination of loss and retransmissions, but the requirement to observe both Data and Acknowledgement packets and perform RTT estimation for every concurrent flow, makes this technique infeasible for the majority of network monitoring applications.

This chapter presents the development of a lightweight method for real time monitoring of TCP flows, which seeks to improve on the issues associated with the work by Jaiswal and Love. The work presented in this chapter has been filed for patent, "Real Time Monitoring of TCP Flows", UK Patent Application GB2430577, Filed September 2005, Published March 2007, and US Patent Application 20070070916, Published March 2007.

5.3 A Passive Mid-Point Monitoring Technique

This section describes the development of a passive mid-point monitoring technique and out-of-sequence packet classification algorithm, which were developed in order to classify TCP data packets from an arbitrary point in a network, without the requirement of observing symmetrical Acknowledgement packets, nor performing estimation of RTT.

5.3.1 Development of a Passive Mid-Point Software Probe

A capture device and software probe are clearly necessary to collect the relevant information required, in order to passively monitor and collect enough relevant information, to then infer why a packet has been reordered and how it should be classified.

At any point in the network where TCP flows can be observed, the probe monitors the packets that pass by. Whenever a packet is observed that matches specific criteria relating to the measurements of interest, the time that the packet is observed is noted and the packet is copied to the packet capture buffer for analysis.

Measurement is performed on a per-flow basis. For each flow, a "Flow Trace" is

created; the trace consists of a series of "Packet Records" where each record consists of information about each packet in the flow.

Each TCP connection is uniquely identifiable using the IP addresses of the source and destination nodes and the port numbers on those nodes between which the connection has been established. Figure 60 illustrates part of a TCP packet header that provides the identification by which each Packet Record is stored and defines a Flow Trace. For each Flow Trace, packet records are formed for all packets that relate to that flow trace and the packet records are stored in the memory of the probe, although the memory could be located elsewhere if required.

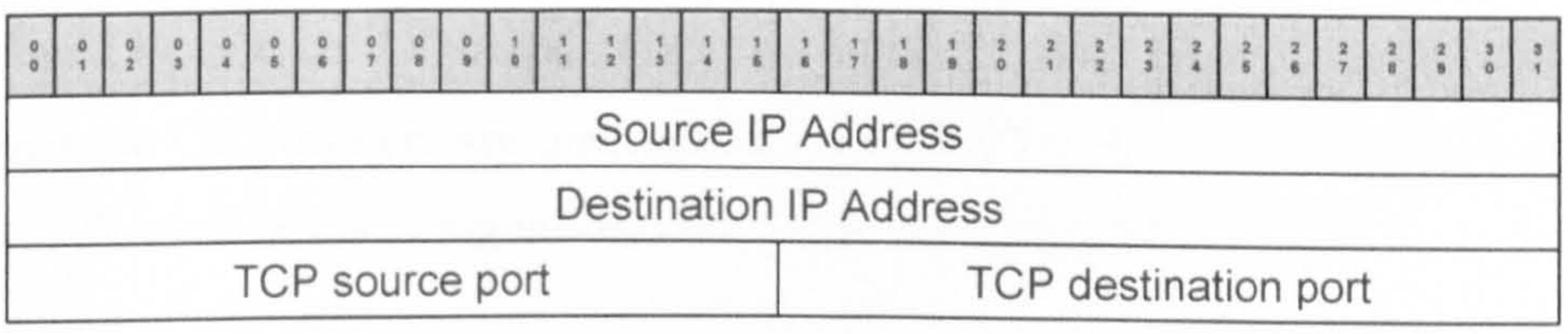

Figure 60 – Example Flow Trace

For each TCP flow, starting with the SYN packet in forward direction flows, and the SYN-ACK in reverse direction flows, the packet's headers are examined and a packet record is added to the flow trace. On each packet arrival, a new packet record is added to the flow trace for that connection.

Figure 61 illustrates the contents of a TCP packet record. The packet record consists of 14 bytes and comprises a sequence number, the IP ID of the packet, the timestamp of its arrival at the network probe, an incrementing sequence number, called the 'Observation Position' (OP) and the packet length.

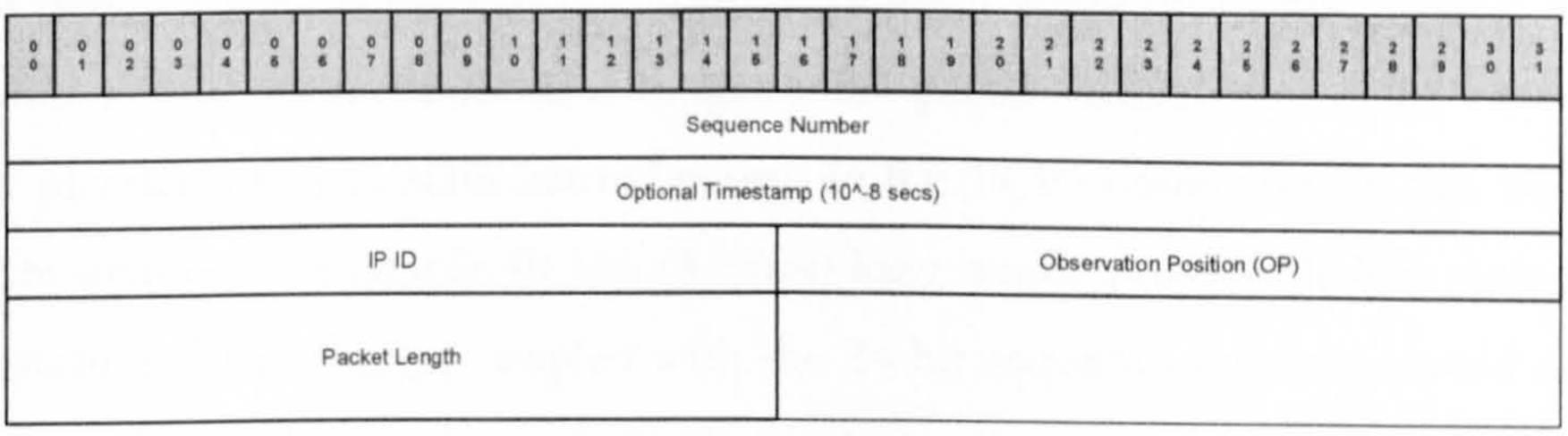

Figure 61 – Example Packet Record

The OP is provided by an integer counter within the probe device, which is used to record the arrival position of each packet observed in the flow at the measurement position. The counter is initialised to zero when a new Flow Trace is created and is then incremented by one on the arrival of each new packet for a given direction within that flow. Separate OP counters are maintained for each Flow Trace and for each direction within the flow currently being monitored by the network probe.

The optional timestamp measures the time elapsed between the current packet's arrival, and the arrival of the previous packet. For example, if this time was recorded in 10's of nanoseconds, an inter-packet gap of just under 43 seconds could be accommodated. The OP counter, IP ID and Packet Length may be 16 bit numbers with a range 0 to 65535. The IP ID can be arbitrarily set by the sender, but should change on each packet transmitted, thereby providing a method for identifying network duplicates of packets with identical Sequence and Ack numbers. The OP counter may recycle back to 0 after reaching 65535. Since this number is used to determine packet sequencing and packets are unlikely to arrive at more than a few tens of positions out of sequence, the 65535 limit is adequate for most connections. If measurements over a very long flow are required, a larger counter could be used.

Each TCP packet sequence number is a 32 bit number. Storage could be optionally further reduced by minimising the memory storage requirements for each flow record

by normalising the sequence numbers, with respect to the start of the sequence numbers for that flow for a given direction. Monitoring certain applications may require a longer timestamp period to accommodate longer inter packet arrival times, for example, to support protocols that contain natural pauses in the TCP connection. In this instance, a larger timestamp, for example 40 bits (5 bytes) long would provide slightly over 3 hours of timestamps. This change, coupled with the 24 bit sequence number, would align the packet record on a 4 byte storage boundary, which can be useful on certain commodity hardware. The length of the various elements in the packet record can be adapted to suit the type of protocols being monitored, the type of hardware resources available on the probe, and the requirements to minimise the processing footprint on the hardware device.

5.3.2 Insertion of Packet Records into Flow Traces

On arrival each packet record is inserted into a flow trace. However, the insertion does not occur in arrival order. Insertion occurs in a sorted order keyed against the Sequence Number and IP ID. The sort is characterised in pseudo code as the function shown in Table 5, where each Packet Record has a structure of the type previously described in Figure 61. Using this sort function, a packet record will be placed in the Flow Trace, ordered from the lowest Sequence Number to the largest Sequence Number. For packets with the same Sequence Number, the IP ID is used for differentiation, although packets could just as easily be sorted against the Sequence Number, only. Using both numbers gives an increased level of differentiation between those packets that are retransmissions and those that were caused by network duplication.

As each packet is inserted into the Flow Trace, it is assigned an OP. The OP indicates the actual arrival order of the packets within the flow and the Sequence Number provides the order that the packets were actually transmitted.

```
boolean sort(Record a, Record b) {
    if (a.sequenceNumber == b.sequenceNumber) {
        return a.ipid < b.ipid;
    } else {
        return a.sequenceNumber < b.sequenceNumber;
    }
}
```

Table 5 - Pseudo code of Packet Record Sorting

When either a FIN or a RST is sent, or in case these are missed, after a suitable timeout period, the whole of the captured Flow Trace can be analysed. If the processing resources are not available at the probe, the Flow Trace could be sent to a network management station for analysis. Any results generated are either stored and displayed via the probe, or via the network management station.

The analysis of the flow trace (when finished, reset or after the timeout) starts with a calculation of an "Expected Position" (EP) number for each packet record within the flow trace. The calculation of EP for each packet record is described in further detail in Figure 62.

5.3.3 Calculation of Expected Position

Figure 62 illustrates the calculation of an "Expected Position" (EP) number for each packet record within the Flow Trace. Upon completion of the packet capture, all packet records within the flow trace have been entered in sorted order keyed against "Sequence Number" and "IP ID". The steps to calculate EP are as follows:

The first data packet record in the Flow Trace is found. This is identified as having a Sequence Number identical to the original flow SYN packet, but with different IP ID fields and a packet length not equal to zero.

The first data packet is assigned the initial EP number. For example, the initial EP could be set to the same as the OP of the first data packet, thus allowing simple comparison between the two values, and initially, before any retransmissions occur, thereby allowing EP==OP as an indication of perfect sequencing.

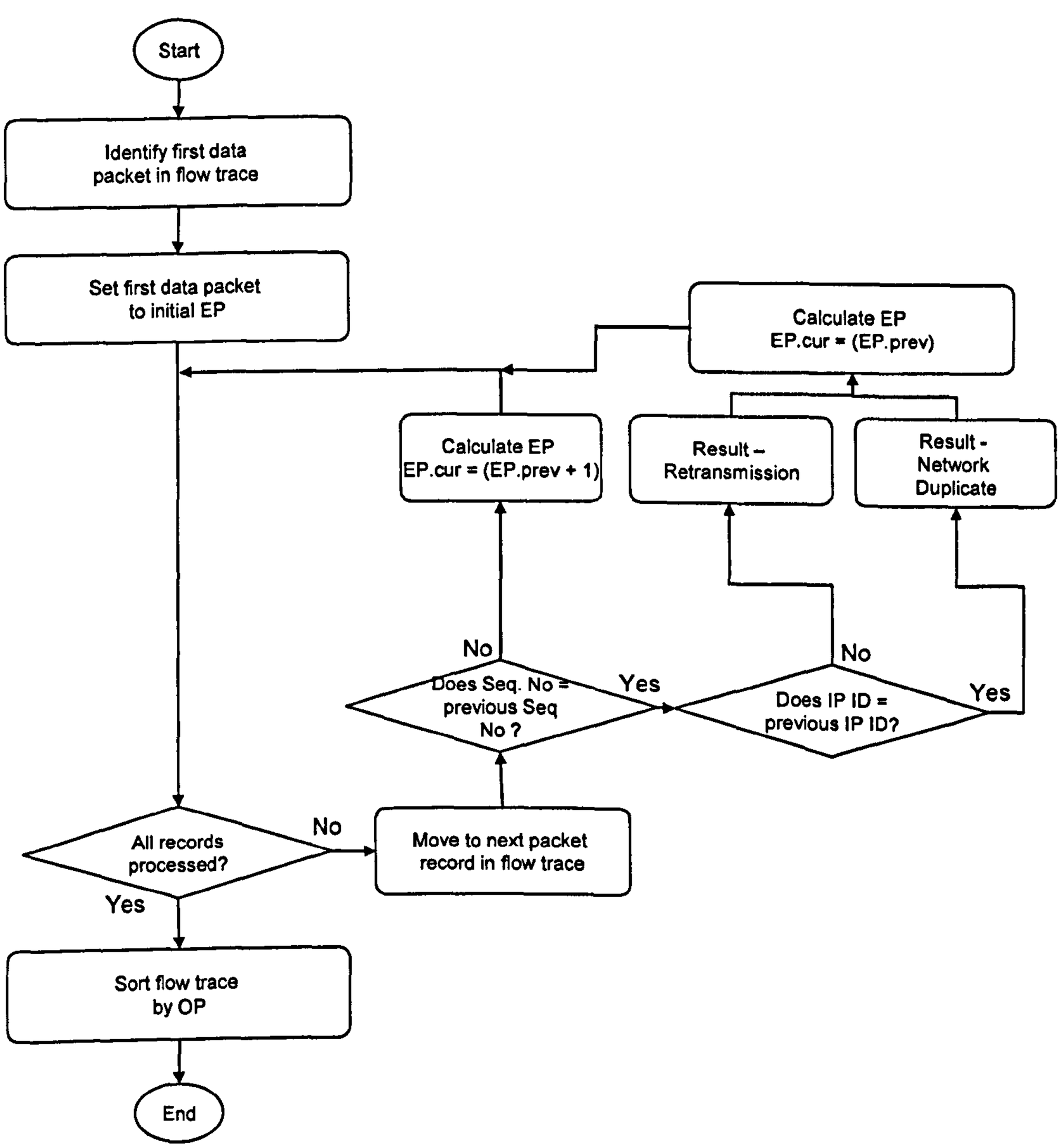

Figure 62 - Expected Position Calculation

The algorithm then loops around all records in the flow trace until all have been processed.

Each packet record is then tested by Sequence Number, to compare it with the Sequence Numbers of previous packet record. If the Sequence Numbers are different, the EP value of the current packet record is calculated as the EP value of the previous packet record plus 1.

If two packet records have the same Sequence Number, the IP ID value of the current packet record is tested with respect to the IP ID value of the previous packet record. This allows differentiation between a Retransmission and a Network Duplicate, allowing a simple count to be maintained of each, which provides a more accurate measure of Goodput than the method described by Love. The EP value of the current packet record is then set to the same value as the EP value of the previous packet record.

Once all packets reordered within the Flow Trace have been processed, they are then sorted are re-inserted into the Flow Trace by OP.

The benefit of recording an entire Flow Trace and then sorting by Sequence Number and IPID, is that it does not require calculation of an NESN for each packet, as described in the Love algorithm. However, if the probe is located at a point where data packets from a single TCP connection may traverse different paths, resulting in some packets not passing through the measurement probe, the Flow Trace will be incomplete for that connection, and neither the sorting method nor NESN would be able to evaluate the correct EP values for the Flow Trace.

5.3.4 Calculation of Relative Sequencing

Figure 63 illustrates the processing carried out in order to define the amount of re-sequencing which has occurred within a flow trace. Once each packet record has been assigned an OP and EP value, the algorithm steps through all OP numbers and performs a series of calculations, on the EP (EP.cur) and OP (OP.cur) values of the current packet record, with respect to the EP (EP.prev) and OP (OP.prev) values of the previous packet record.

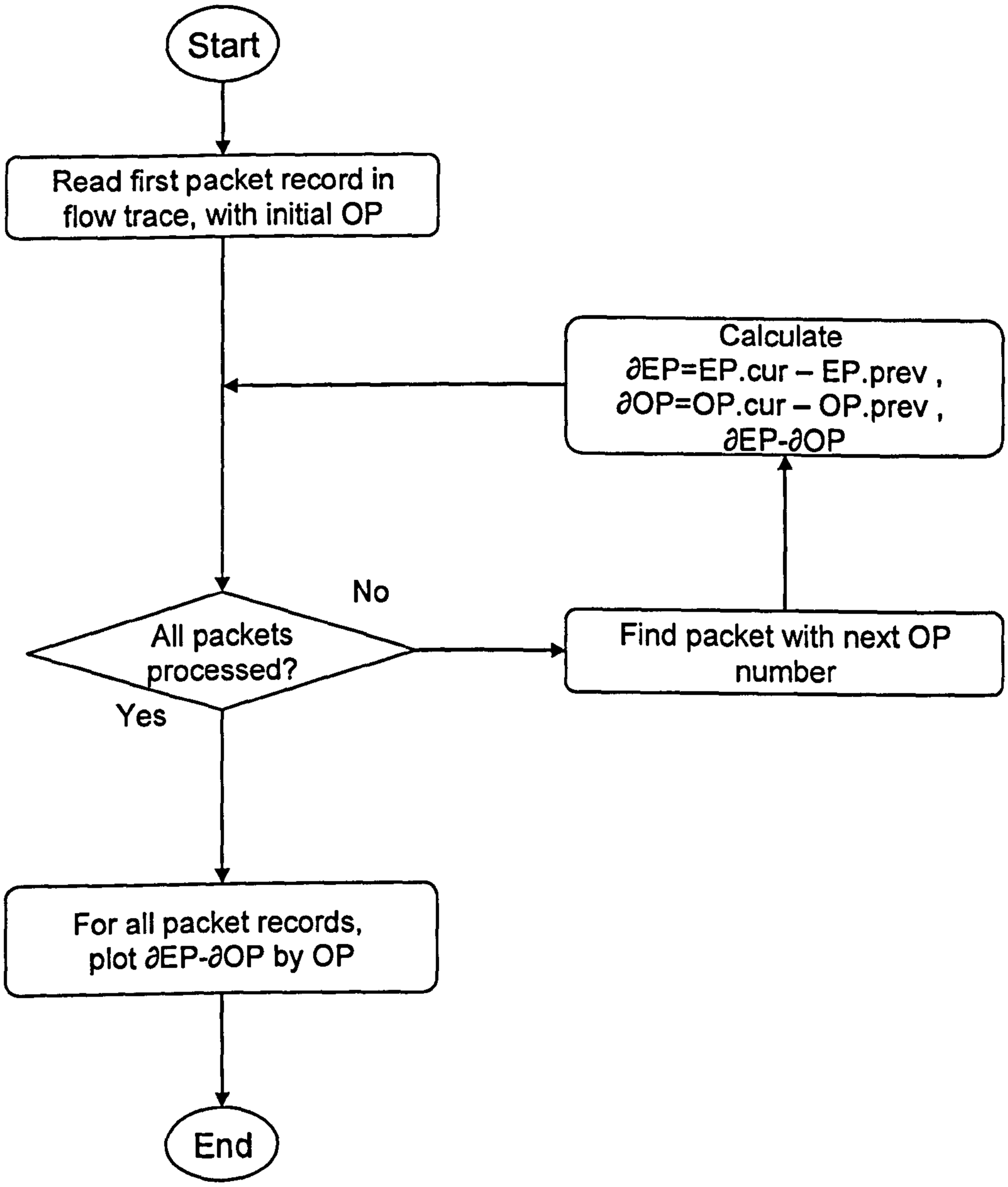

Figure 63 - Post Processing Flowchart

Equation 6 and Equation 7 calculate the rate of change between the current packet record and the previous packet record, where in an ideal scenario with perfect network sequencing, both would increment monotonically each time. Equation 8 performs rate of change analysis, by allowing definition of a simple metric to test if a packet is out of sequence. If Equation 8 yields zero, then perfect network behaviour is being observed. If Equation 8 gives any other value, then the packet is flagged as Out-of-Sequence and will be classified as such by the algorithm.

$$\partial EP = EP.cur - EP.prev$$

Equation 6 - Expected Position Rate of Change

$$\partial OP = OP.cur - OP.prev$$

Equation 7 - Observed Position Rate of Change

$$\partial EP - \partial OP$$

Equation 8 - Rate of Change Analysis

5.3.5 The Arthur "Out of Sequence" Classification Algorithm

With the packet records reinserted in the flow trace by OP and calculation of all EP numbers complete, it is then possible to parse the flow trace and evaluate each packet record using the algorithm presented in Figure 64.

Figure 64 illustrates the steps of capturing the TCP stream, inserting the packets into the flow trace as discussed in Section 5.3.2, calculating EP as discussed in 5.3.3, and then comparing EP and OP for each packet as illustrated by Equations 6, 7 and 8.

Each packet is then classified using the algorithm as illustrated in Figure 64 and the following steps are used in the algorithm to allow interpretation of the results:-

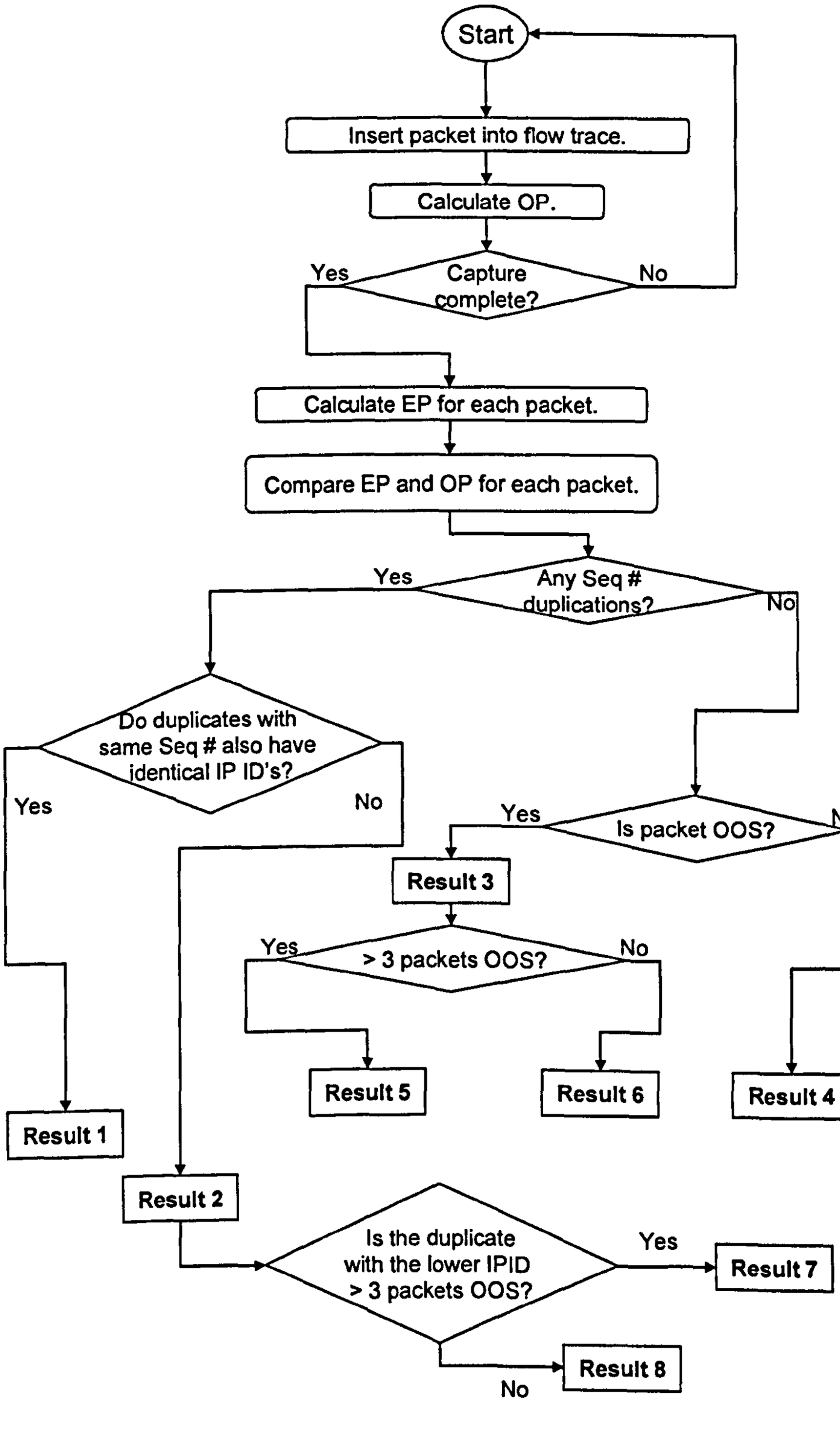

Figure 64 – Arthur Out of Sequence Classification Algorithm

5.3.5.1 Result 1

Result 1 - Network Duplicate Packets.

Packets that are duplicated in the flow record and share the same Sequence Number and IP ID are marked as network duplicates; a count of the number of network duplicates is maintained.

5.3.5.2 Result 2

Result 2 - Retransmitted Packet, due to Packet Reordering Upstream or Downstream, or Packet Loss & Retransmission Downstream from Measurement Point.

Packets that are duplicated in the Flow Trace, sharing the same Sequence Number, but which are not network duplicates, as determined by differing IP ID fields, are retransmissions of packets and a count of such packets is maintained. The cause of this retransmission is a result of

- Packet reordering either upstream or downstream from the measurement point.

  or

- Packet loss downstream from the measurement point due to network congestion or malfunction of the network.

5.3.5.3 Result 3

Result 3 - Packet Reordering, or Packet Loss & Retransmission Upstream from Measurement Point

Packets that are Out-Of-sequence and are not duplicated in the flow record will be due either to:

- Packet reordering upstream from the measurement point.

  or

- Packet loss upstream from the measurement point due to network congestion or malfunction, which has resulted in a subsequent retransmission of the lost packet.

5.3.5.4 Result 4

Result 4 – Perfect Network Behaviour

If no flow record Sequence Number duplicates are detected and the packet is not measured to be out of sequence by Equation 8, then Perfect Network Behaviour is observed.

5.3.5.5 Result 5

Result 5 - Packet Loss Upstream & Retransmission, or Packet Reordering

Packets that are categorised as being out-of-sequence and are significantly out-of-sequence but do not share a duplicate packet in the flow record, can be assumed to be caused by:

- Packet loss upstream from the measurement point, as a significantly late arrival of the packet will have triggered either a Fast-Retransmit from the receiver, or a Round-Trip-Timeout at the sender. This would be the case for packets that are significantly more than 3 positions Out-of-Sequence, as a large delay would be incurred whilst a retransmission was transmitted.

  or

- Packet reordering upstream and downstream from the measurement point,

which has not been to a large enough degree to cause a retransmission through RTT or Fast-Retransmit, but indicates significant disruption in network behaviour and should be measured as such.

The degree to which packets are being measured out-of-sequence should be maintained, as significant numbers of packets arriving more than 3 positions out-of-sequence, is an indicator of unhealthy network performance.

5.3.5.6 Result 6

Result 6 – Packet Reordering Upstream from the Measurement Point

Packets that are categorised as being Out-Of-Sequence, but are less than three packets Out-Of-Sequence are a result of packet reordering upstream from the measurement point. This is indicative of unhealthy network performance and a count of this measure should be maintained, but these packets do not indicate problems with network performance as serious as those categorised under Result 5, as they are unlikely to cause unnecessary retransmissions.

5.3.5.7 Result 7

Result 7 – Packet Retransmission, caused by Fast Retransmission

Packets that are categorised as Result 2 and have duplicate packet records, but where the packet record with the lower IPID value is more than three packets out-of-sequence, are retransmissions that are likely to be the result of the TCP receiver receiving three duplicate Acks and determining that the packet must be retransmitted. This is the amount of fast retransmission occurring in the flow and a count of such events should be maintained.

5.3.5.8 Result 8

Result 8 – Packet Retransmission, caused by RTO or Reverse Path Reordering or Reverse Path Loss

Packets that are categorised as sharing the same Sequence Number, but not being network duplicates, but where the packet record with the lower IP ID value is less than three packets Out-of-Sequence, are retransmissions that could have been caused either by:

- The TCP transmitter's RTO (retransmit timer) firing. This is the amount of sender RTO retransmission occurring in the flow and a count of this events should be maintained

  or

- Reverse-Path reordering or loss, resulting in Acks not arriving at the transmitter as expected, and causing retransmissions.

5.4 Out of Sequence Classification Example

Table 6 illustrates an example of both directions of a single TCP flow capture, with the respective OP measurement calculated. Analysis of this sequence of packets, would be carried out using two separate flow traces, one for each direction:

- Flow Trace 1 - 10.0.0.2:1789 to 10.0.0.6:35427
- Flow Trace 2 - 10.0.0.6:35427 to 10.0.0.2:1789

Upon termination of the connection, when Flow Trace 1 is observed transmitting a RST or FIN, the results are processed and the EP numbers are calculated (as demonstrated in Figure 62).

Timestamp		Source		Destination		OP	Seq	Ack	IPID	Packet Length
Secs	Usecs	Address	Port	Address	Port					
1125264298	160003	10.0.0.2	1789	10.0.0.6	35427	1	3816962545	3811057447	43099	1448
1125264298	160007	10.0.0.2	1789	10.0.0.6	35427	2	3816963993	3811057447	43100	1448
1125264298	275604	10.0.0.6	35427	10.0.0.2	1789	1	3811057447	3816963993	36800	0
1125264298	275636	10.0.0.6	35427	10.0.0.2	1789	2	3811057447	3816965441	36801	0
1125264298	277983	10.0.0.2	1789	10.0.0.6	35427	3	3816965441	3811057447	43101	1448
1125264298	278330	10.0.0.2	1789	10.0.0.6	35427	4	3816966889	3811057447	43102	1448
1125264298	278332	10.0.0.2	1789	10.0.0.6	35427	5	3816969785	3811057447	43104	1448
1125264298	278334	10.0.0.2	1789	10.0.0.6	35427	6	3816968337	3811057447	43103	1448
1125264298	395590	10.0.0.6	35427	10.0.0.2	1789	3	3811057447	3816966889	36802	0
1125264298	395619	10.0.0.6	35427	10.0.0.2	1789	4	3811057447	3816968337	36803	0
1125264298	395641	10.0.0.6	35427	10.0.0.2	1789	5	3811057447	3816968337	36804	0
1125264298	395671	10.0.0.6	35427	10.0.0.2	1789	6	3811057447	3816971233	36805	0
1125264298	398070	10.0.0.2	1789	10.0.0.6	35427	7	3816971233	3811057447	43105	1448
1125264298	398427	10.0.0.2	1789	10.0.0.6	35427	8	3816972681	3811057447	43106	1448
1125264298	398428	10.0.0.2	1789	10.0.0.6	35427	9	3816974129	3811057447	43107	1448
1125264298	398430	10.0.0.2	1789	10.0.0.6	35427	10	3816975577	3811057447	43108	1448
1125264298	398432	10.0.0.2	1789	10.0.0.6	35427	11	3816968337	3811057447	43109	1448
1125264298	398434	10.0.0.2	1789	10.0.0.6	35427	12	3816977025	3811057447	43110	1448
1125264298	515588	10.0.0.6	35427	10.0.0.2	1789	7	3811057447	3816972681	36806	0
1125264298	515616	10.0.0.6	35427	10.0.0.2	1789	8	3811057447	3816974129	36807	0
1125264298	515638	10.0.0.6	35427	10.0.0.2	1789	9	3811057447	3816975577	36808	0
1125264298	515661	10.0.0.6	35427	10.0.0.2	1789	10	3811057447	3816977025	36809	0
1125264298	515673	10.0.0.6	35427	10.0.0.2	1789	11	3811057447	3816977025	36810	0
1125264298	515700	10.0.0.6	35427	10.0.0.2	1789	12	3811057447	3816978473	36811	0
1125264298	517947	10.0.0.2	1789	10.0.0.6	35427	13	3816978473	3811057447	43111	1448
1125264298	518325	10.0.0.2	1789	10.0.0.6	35427	14	3816979921	3811057447	43112	1448
1125264298	518327	10.0.0.2	1789	10.0.0.6	35427	15	3816981369	3811057447	43113	1448
1125264298	518329	10.0.0.2	1789	10.0.0.6	35427	16	3816982817	3811057447	43114	1448
1125264298	518330	10.0.0.2	1789	10.0.0.6	35427	17	3816984265	3811057447	43115	1448
1125264298	518725	10.0.0.2	1789	10.0.0.6	35427	18	3816985713	3811057447	43116	1408
1125264298	635583	10.0.0.6	35427	10.0.0.2	1789	13	3811057447	3816979921	36812	0
1125264298	635611	10.0.0.6	35427	10.0.0.2	1789	14	3811057447	3816981369	36813	0
1125264298	635634	10.0.0.6	35427	10.0.0.2	1789	15	3811057447	3816982817	36814	0
1125264298	635657	10.0.0.6	35427	10.0.0.2	1789	16	3811057447	3816984265	36815	0
1125264298	635685	10.0.0.6	35427	10.0.0.2	1789	17	3811057447	3816987121	36816	0
1125264298	637902	10.0.0.2	1789	10.0.0.6	35427	19	3816987121	3811057447	43117	1448
1125264298	638254	10.0.0.2	1789	10.0.0.6	35427	20	3816990017	3811057447	43119	1448
1125264298	638622	10.0.0.2	1789	10.0.0.6	35427	21	3816988569	3811057447	43118	1448
1125264298	638624	10.0.0.2	1789	10.0.0.6	35427	22	3816991465	3811057447	43120	1448
1125264298	638626	10.0.0.2	1789	10.0.0.6	35427	23	3816992913	3811057447	43121	1448
1125264298	638628	10.0.0.2	1789	10.0.0.6	35427	24	3816994361	3811057447	43122	952
1125264298	638630	10.0.0.2	1789	10.0.0.6	35427	25	3816995313	3811057447	43123	1448
1125264298	638631	10.0.0.2	1789	10.0.0.6	35427	26	3816998209	3811057447	43125	1448
1125264298	638633	10.0.0.2	1789	10.0.0.6	35427	27	3816996761	3811057447	43124	1448

Table 6 - Example TCP Stream Capture

Table 7 and Table 8 illustrate the TCP packet records that would be generated by the packet sequencing analysis using the TCP stream capture data illustrated in Table 6.

Timestamp (secs)	Timestamp (usecs)	OP	Seq	IPID	Packet Length
1125264298	160003	1	3816962545	43099	1448
1125264298	160007	2	3816963993	43100	1448
1125264298	277983	3	3816965441	43101	1448
1125264298	278330	4	3816966889	43102	1448
1125264298	278332	5	3816969785	43104	1448
1125264298	278334	6	3816968337	43103	1448
1125264298	398070	7	3816971233	43105	1448
1125264298	398427	8	3816972681	43106	1448
1125264298	398428	9	3816974129	43107	1448
1125264298	398430	10	3816975577	43108	1448
1125264298	398432	11	3816968337	43109	1448
1125264298	398434	12	3816977025	43110	1448
1125264298	517947	13	3816978473	43111	1448
1125264298	518325	14	3816979921	43112	1448
1125264298	518327	15	3816981369	43113	1448
1125264298	518329	16	3816982817	43114	1448
1125264298	518330	17	3816984265	43115	1448
1125264298	518725	18	3816985713	43116	1408
1125264298	637902	19	3816987121	43117	1448
1125264298	638254	20	3816990017	43119	1448
1125264298	638622	21	3816988569	43118	1448
1125264298	638624	22	3816991465	43120	1448
1125264298	638626	23	3816992913	43121	1448
1125264298	638628	24	3816994361	43122	952
1125264298	638630	25	3816995313	43123	1448
1125264298	638631	26	3816998209	43125	1448
1125264298	638633	27	3816996761	43124	1448

Table 7 - Flow Trace 1 - 10.0.0.2:1789 to 10.0.0.6:35427

Table 9 illustrates the results of performing EP calculation using Flow Trace 1. It can be seen that the packet with Expected Position 6 was reordered during transmission and was observed arriving at position 7. The degree of sequence loss was sufficient to cause a retransmission; this can be observed as the packet arriving at Observed Position 12.

Timestamp (secs)	Timestamp (usecs)	OP	Ack	IPID	Packet Length
1125264298	275604	1	3816963993	36800	0
1125264298	275636	2	3816965441	36801	0
1125264298	395590	3	3816966889	36802	0
1125264298	395619	4	3816968337	36803	0
1125264298	395641	5	3816968337	36804	0
1125264298	395671	6	3816971233	36805	0
1125264298	515588	7	3816972681	36806	0
1125264298	515616	8	3816974129	36807	0
1125264298	515638	9	3816975577	36808	0
1125264298	515661	10	3816977025	36809	0
1125264298	515673	11	3816977025	36810	0
1125264298	515700	12	3816978473	36811	0
1125264298	635583	13	3816979921	36812	0
1125264298	635611	14	3816981369	36813	0
1125264298	635634	15	3816982817	36814	0
1125264298	635657	16	3816984265	36815	0
1125264298	635685	17	3816987121	36816	0

Table 8 - Flow Trace 2 - 10.0.0.6:35427 to 10.0.0.2:1789

Timestamp (secs)	Timestamp (usecs)	OP	Seq	IPID	Packet Length	EP
1125264298	159999	1	3816961097	43098	1448	1
1125264298	160003	2	3816962545	43099	1448	2
1125264298	160007	3	3816963993	43100	1448	3
1125264298	277983	4	3816965441	43101	1448	4
1125264298	278330	5	3816966889	43102	1448	5
1125264298	278332	6	3816969785	43104	1448	7
1125264298	278334	7	3816968337	43103	1448	6
1125264298	398070	8	3816971233	43105	1448	8
1125264298	398427	9	3816972681	43106	1448	9
1125264298	398428	10	3816974129	43107	1448	10
1125264298	398430	11	3816975577	43108	1448	11
1125264298	398432	12	3816968337	43109	1448	6
1125264298	398434	13	3816977025	43110	1448	12
1125264298	517947	14	3816978473	43111	1448	13
1125264298	518325	15	3816979921	43112	1448	14
1125264298	518327	16	3816981369	43113	1448	15
1125264298	518329	17	3816982817	43114	1448	16
1125264298	518330	18	3816984265	43115	1448	17
1125264298	518725	19	3816985713	43116	1408	18
1125264298	637902	20	3816987121	43117	1448	19
1125264298	638254	21	3816990017	43119	1448	21
1125264298	638622	22	3816988569	43118	1448	20
1125264298	638624	23	3816991465	43120	1448	22
1125264298	638626	24	3816992913	43121	1448	23
1125264298	638628	25	3816994361	43122	952	24
1125264298	638630	26	3816995313	43123	1448	25
1125264298	638631	27	3816998209	43125	1448	27
1125264298	638633	28	3816996761	43124	1448	26

Table 9 - Packet Sequencing Analysis

Table 10 illustrates the results for a Packet Sequence Rate of Change Analysis as applied to the packets from Table 7 and the resulting classifications as applied by the Arthur classification algorithm. Empirical observation has indicated that a good indicator of the health of a TCP connection is the rate of change of sequence numbers. TCP receivers expect to receive packets from a flow in order and so an out-of-sequence packet, whatever the cause, indicates a breakdown which causes additional overhead to the connection.

OP	EP	∂OP	∂EP	∂EP - ∂OP	Timestamp (secs)	Timestamp (usecs)	Seq	IPID	Packet Length	Arthur Algorithm (Figure 64) Classification Results
1	1	1	1	0	1125264298	159999	3816961097	43098	1448	4
2	2	1	1	0	1125264298	160003	3816962545	43099	1448	4
3	3	1	1	0	1125264298	160007	3816963993	43100	1448	4
4	4	1	1	0	1125264298	277983	3816965441	43101	1448	4
5	5	1	1	0	1125264298	278330	3816966889	43102	1448	4
6	7	1	2	1	1125264298	278332	3816969785	43104	1448	3, 6
7	6	1	-1	-2	1125264298	278334	3816968337	43103	1448	2, 8
8	8	1	2	1	1125264298	398070	3816971233	43105	1448	3, 6
9	9	1	1	0	1125264298	398427	3816972681	43106	1448	4
10	10	1	1	0	1125264298	398428	3816974129	43107	1448	4
11	11	1	1	0	1125264298	398430	3816975577	43108	1448	4
12	6	1	-5	-6	1125264298	398432	3816968337	43109	1448	2, 8
13	12	1	6	5	1125264298	398434	3816977025	43110	1448	3, 6
14	13	1	1	0	1125264298	517947	3816978473	43111	1448	4
15	14	1	1	0	1125264298	518325	3816979921	43112	1448	4
16	15	1	1	0	1125264298	518327	3816981369	43113	1448	4
17	16	1	1	0	1125264298	518329	3816982817	43114	1448	4
18	17	1	1	0	1125264298	518330	3816984265	43115	1448	4
19	18	1	1	0	1125264298	518725	3816985713	43116	1408	4
20	19	1	1	0	1125264298	637902	3816987121	43117	1448	4
21	21	1	2	1	1125264298	638254	3816990017	43119	1448	3, 6
22	20	1	-1	-2	1125264298	638622	3816988569	43118	1448	3, 6
23	22	1	2	1	1125264298	638624	3816991465	43120	1448	3, 6
24	23	1	1	0	1125264298	638626	3816992913	43121	1448	4
25	24	1	1	0	1125264298	638628	3816994361	43122	952	4
26	25	1	1	0	1125264298	638630	3816995313	43123	1448	4
27	27	1	2	1	1125264298	638631	3816998209	43125	1448	3, 6
28	26	1	-1	-2	1125264298	638633	3816996761	43124	1448	3, 6

Table 10 – Rate of Change Analysis

By performing this analysis, in an ideal case EP should increase by exactly 1 per packet - every packet received should be that predicted from the previous packet. Therefore, if applied across a sequence of more than two packets sorted by their observed position, if

Equation 8 does not equal zero, the connection is shown to have suffered sequencing problems and is, therefore, unhealthy. This Packet Sequence Rate of Change Analysis can measure the effects of packet reordering, by highlighting the points at which the observed sequence changes, rather than identifying packets as being either in or out of sequence. Even minor loss of sequence would suggest that a connection is unhealthy as the change in sequence could be misinterpreted as loss by the receiver, in which case a retransmission would be caused and congestion avoidance would begin.

From the timestamp data within each packet record, the rate of change of inter-packet arrival times can also be calculated. The difference in inter-arrival times, differentiated over a distance of packets, selected in a method similar to that described for rate of sequence change, would allow for an indication of the burstiness of the data. If both forward and reverse flow traces were obtained, comparison of these would allow for measurements of reverse path reordering and how it had affected the forward path's burstiness.

5.4.1 Dealing with Duplicates

Table 10 further illustrates the use of the ∂EP, ∂OP, and ∂EP-∂OP when retransmissions occur in a flow trace. It is important to note that there are situations when multiple packet records in the flow trace will be assigned identical EP numbers. This will occur when retransmitted packets are allocated EP numbers that are identical to the original 'lost' packet. This will also occur when there are multiple retransmissions of a lost packet, or where reordering occurs to such an extent that the receiver has assumed the packet to be lost, and thus requested retransmission. Meanwhile, the OP counter will increase linearly for each packet arrival, which after a retransmission event when the flow has returned to correct sequencing, will have resulted in the OP counter being ahead of the EP counter by the number of packet retransmissions that have occurred.

Therefore, calculating the change in EP and OP between packets provides a measure of sequence change that is relevant in situations where retransmissions will affect the expected sequence of arriving packets. Calculation of ∂OP will always result in a value

of 1 when calculated over 2 packet records, as OP increases linearly during packet capture. Therefore, when calculating ∂EP - ∂OP, a value of anything other than 0 is an indication of sequence breakdown. For example, in Table 10 a sequence gap is detected at OP 6 where EP 6 has undergone packet reordering, and arrives in OP 7. In this example, the reordering was sufficient to cause a retransmission of the packet at EP 6 which is observed again at OP 12. In the intervening time between OP 9 and OP 11, packet sequencing is maintained as expected, with EP==OP, and so ∂EP - ∂OP would indicate good network behaviour for this part of the flow.

After the anomaly of the retransmission at OP 12, ∂EP - ∂OP from OP 14 onwards indicates that the flow trace is back in perfect sequence - despite OP and EP becoming 'out of step' due to the additional retransmission. Calculation of ∂EP-∂OP therefore mitigates this effect in addition to providing a simple measure of how late or early a packet has been measured as arriving with respect to the expected sequence of arrival. Positive numbers indicate early packets, while negative numbers indicate late packets.

For example, in Table 10, OP 6 is calculated as having ∂EP-∂OP=1, indicating that the packet has arrived 1 position earlier than what it should be expected to be. Calculation of ∂EP-∂OP on OP 7=-2, indicating that the packet has arrived 2 positions late. Calculation of ∂EP-∂OP on OP 12=-6, indicating that the packet is 6 positions late and with an EP of 6(which has also been assigned to the packet with OP 7) indicates that this is a retransmission, caused by either downstream packet loss or packet reordering.

5.5 Implementation of Algorithm

In order to compare this mid-point classification algorithm with Jaiswal, a software prototype of the algorithm was implemented in C++ and Perl. Using the Libpcap-MMap extensions on a Linux platform, the C++ code performs passive sniffing of a Gigabit Ethernet Interface Card which, when used with a passive optical tap, can be used to perform mid-point monitoring of a fibre-based gigabit Ethernet connection.

For each packet received, the C++ code evaluates the 4-tuple ports and addresses of the

source and destination, to define a Connection object. This Connection object is then used as a key into a hash of MeasuredSequence objects. Each MeasuredSequence object is a sorted 'Packet Record' of the format described in Figure 61.

The length of a packet capture can be specified at run time; in this case, the C++ code executes until a defined number of flows have been captured and then, prior to exit, flushes each MeasuredSequence object to output files. Example MeasuredSequence objects, for each flow, are illustrated in Table 7 and Table 8.

At this point, further processing must be performed in Perl. A single txt capture file is generated by the C++ application, but with all MeasuredSequence objects, for every flow observed, written into that file. A Perl script was developed to parse these text files and perform the Expected Position calculation as defined in Figure 62, and the Rate of Change Analysis as defined in Table 10. This results in a separate text file output, for each flow trace, with summary statistics generated of the number of packets classified by each rule in Figure 64.

5.5.1 Comparison with Jaiswal

One of the significant contributions of Jaiswal's work was the ability to access packet captures from the Sprint IP Monitoring project and, therefore, perform real-time analysis of Tier-1 trans-continental 2.5 Gb/s network links.

As is common with many ISPs, for security and privacy concerns, access to such real data is not publicly available and therefore the packet captures used by Jaiswal are not available in the form of packet headers that can be processed by the Arthur classification algorithm. Therefore, in order to provide comparison between this mid-point classification algorithm and the Jaiswal algorithm, a set of experiments were devised to generate new data in the formats which both algorithms could analyse and classify, thus allowing comparison of results from both techniques.

The source code for the Jaiswal Out-of-Sequence Classification algorithm was obtained from Sharad Jaiswal and was compiled and tested. Jaiswal's classification tool,

'TCPFlows', can read packet traces either by live capture through a DAG card or by reading packets that have been captured and saved in the form of tcpdump files.

5.5.2 Experimental Setup

In order to capture packet traces of live Internet traffic across a reasonably large Internet backbone connection, an experiment was performed using a HP NetServer LPr. The NetServer was compiled with the C++ code developed in this chapter, which performed its own capture and logging of Flow Traces into text files. Additionally, tcpdump was simultaneously run on the same gigabit interface, in order to capture the same packets which were logged by the C++ code and written in the tcpdump file format for later processing by Jaiswal TCPFlows. Both the C++ code and tcpdump perform packet capture using Libpcap and were found to operate on the same interface on the same machine with no performance overheads. Tcpdump was configured to capture only the first 96 bytes of each packet to avoid disk space exhaustion.

The NetServer was located beside an egress router at the Agilent Technologies South Queensferry facility. This router, a Cisco Catalyst 3620, is one of two used to route a variety of different IP traffic, from general office web and email, to experimental IP test traffic from the Agilent Laboratories Test Network. This router was chosen because it operates in parallel with a second identical router to connect, via 1000-Base-SX over fibre Gigabit, to one of the site core routers for 45 Mb/sec uplink to the Agilent Technologies MPLS cloud.

5.5.3 Results and Comparison

A total of 331649 flows were measured over a 6 hour period, and were classified using the Jaiswal algorithm as illustrated in Table 11, and the Arthur algorithm in Table 12. It should be noted that a flow can traverse in either direction and, for the purposes of the Arthur algorithm, describes the Data path alone, while for the Jaiswal algorithm defines both Data and Acknowledgements.

Total Data Packets	Flows	Symmetric Flows	Packets within Symmetric Flows	Total OOS Packets	OOS Unclassified	Network Dups	Unneeded Retx (Ack already Observed)	Reordered Before Probe
51040802	331649	271952 (81.99%)	40556286 (79.46%)	1262395 (2.47%)	0	0	43531 (3.45%)	232165 (18.39%)

OOS due to Retransmission – 986699 (78.16%)				
Retx due to RTO	Retx due to Fast Retx	Retx during Fast RTO Recovery	Retx during Fast Recovery	Retx unknown
72551 (7.35%)	536880 (54.41%)	0	377268 (38.24%)	0

Table 11 - Jaiswal Classification Results

Total Data Packets	Flows	OOS Packets & Sequence Number Duplications
51040802	331649	2728100

Result 1– Network Duplicate	Result 2 – Packet Retransmission due to Reordering, or Packet Loss & Retransmission Downstream		Result 3 – Packet Reordering, or Packet Loss & Retransmission Upstream		Result 4 – Perfect Network Behaviour
0	1873658 (68.68%)		854442 (31.32%)		48312702 (94.67%)
	Result 7 – Retransmission likely caused by Fast Retransmission	Result 8 – Retransmission likely caused by RTO or Reverse Path Reordering or Reverse Path Loss	Result 5 – Packet Loss Upstream & Retransmission, or Packet Reordering	Result 6 – Packet Reordering Upstream	
	1566747 (57.43%)	306911 (11.25%)	594998 (21.81%)	259444 (9.51%)	

Table 12 - Arthur Classification Results

The first comparison that can be made between the two techniques is that the flows observed at this observation point were only slightly asymmetrical, resulting in only 82% of the flows being parsed by the Jaiswal algorithm. This discrepancy can make correlation of results across the techniques difficult as the Jaiswal algorithm is operating over a smaller set of data.

The number of packets reported to be Out-of-Sequence is also shown to be different.

This is partially due to the different definitions for this term and the methods by which both terms are calculated. In Jaiswal, a packet is termed to be Out-of-Sequence if the Sequence number observed is less than or equal to that of a previously observed Sequence number in that connection. In the Arthur algorithm, all packets where ∂EP-∂OP does not equal zero are defined as Out of Sequence. It is important to note that, as a by-product of this test, ∂EP-∂OP will not equal zero when a packet loss occurs (and a normal in-order packet then appears to arrive early) and also immediately after a retransmitted packet has arrived (when again, the immediately successive packet will appear to have arrived late). It is therefore expected that the Arthur algorithm will apparently over-classify the number of Out of Sequence packets in a measurement.

The number of measured Retransmissions due to losses downstream from the probe will be equal in both algorithms; in the Arthur algorithm, a Retransmission is counted as two packets with the same Sequence number while in the Jaiswal algorithm, it is two packets with the same Sequence number and different IP ID fields.

From the data packets alone, Retransmissions that result from a loss prior to the measurement point, are much more difficult to identify. Classifications R2 and R3 of the Jaiswal algorithm, where a packet has not been observed previously, but either has a Time Lag > RTO, Duplicate Acks > 3, or is in state *InFastRecovery* are all straightforward to identify from the Acknowledgement stream, but are difficult to infer using the Data stream alone. In these particular scenarios, the Arthur algorithm will classify the packet as not having observed this Sequence number previously, but also being Out of Sequence, and therefore Result 3. Result 3 defines packets which are either Reordered, or Lost and Retransmitted upstream. Without Acknowledgements, it is extremely difficult to differentiate between these categories and, therefore, the simple rule > 3 positions OOS is used to differentiate between Retransmission and Reordering. It is assumed that if a data packet is more than 3 positions late, a Fast Retransmit will have been initiated by the receiver; this may be inaccurate if the receiving TCP is using SACK or the Linux Kernel *tcp_reordering* variable. Therefore, although the confidence of correctly identifying Result 3 is high, the further differentiation between Result 5 and Result 6 may involve some error, dependent upon the TCP implementations in the connections measured.

Another similar factor that must be considered in the differences between Arthur and Jaiswal Retransmission measurements, is the ability of the Jaiswal algorithm to determine the *inStateFastRecovery* state and, therefore count retransmissions which are occurring as a function of the Sending TCP entering the Fast Recovery algorithm. These retransmissions, due to Fast Recovery, will be classified by the Arthur algorithm as Result 8, as they will be observed as Retransmissions appearing to be more than 3 positions out of sequence and the result will be in an over-estimation of RTO Retransmissions.

It is unusual that the Jaiswal algorithm classifies Network Duplicates in such a complicated fashion. Network Duplicates are classified as such if two packets with the same Sequence Number share the same IP ID, do not have Triple Duplicate Acks, are not in state *InFastRecovery* and have a Time Lag < RTT. In the Arthur algorithm, any packet with the same Sequence Number and same IP ID are immediately flagged as Network Duplicates. It is uncertain why the Jaiswal algorithm requires the evaluation of Time lag < RTT to discern a Network Duplicate, as a Duplicate could appear in the network at any time and for any time duration (especially if such a duplicate became stuck in a routing loop). Unfortunately, during this measurement study, no Network Duplicates were observed, suggesting that these are uncommon phenomena on modern networks.

5.5.4 Conclusions

There are many challenges encountered when designing a mid-point measurement algorithm of TCP as such an algorithm must infer the behaviour and state machine of both TCP end hosts, based only on a small subset of information available in the middle of the connection. There is, therefore, a degree of error in all classification algorithms and these potential errors may adversely affect the results obtained.

A similar problem when designing mid-point measurement algorithms, is the validation of these algorithms to ensure that they classify behaviour as expected. On a large scale experiment, it is impossible to instrument all end hosts and later correlate the actual

behaviour with the mid-point inferred behaviour and, therefore, *packet reordering* studies such as Jaiswal may also include inherent errors due to untested features of the algorithm.

Based on the large number of uncertainties which are encountered during this type of measurement, such as type of end host TCP implementation, the possibility of non-standard compliant hosts and the many TCP features that could be implemented on each TCP stream such as SACK, D-SACK, *tcp_reordering* etc, it is important to err on the side of caution when designing a mid-point algorithm. The Arthur algorithm requires only to view data path and adopts simple classification rules, which makes this algorithm attractive as a tool to analyse the performance of a large number of flows. The methods to calculate loss and retransmission are significantly more advanced than those proposed by Love and will therefore not be adversely affected by packet reordering. Finally, the lightweight nature of the algorithm, in that there is no need to calculate RTT or infer state machine behaviour, would allow this algorithm to be deployed on a large number of mid-point nodes, with little additional processing overhead.

5.6 Network Measurement Visualisation

Measurement Visualisation is an important emerging area in the field of network measurement science and considers techniques to display large multivariate datasets in ways which reveal insights into the data. Information Visualisation is a form of data mining and is increasingly important for network monitoring and management; it allows the characterisation of the overall performance of a vast amount of measurement data.

Chapter 4 has presented the issues encountered when performing any large scale measurement of network traffic. In order to calculate metrics that are representative of the traffic's performance, a significant amount of packet captures must be performed. Chapter 4 has also illustrated that, even when these large amounts of data have been captured and processed, the resulting metrics that are calculated, such as the percentage reordered packet metric, may be meaningless in their ability to describe end-user

performance. Simple numerical value cannot always describe the complex effects that the observed traffic may have on end system congestion control algorithms.

Network Visualisation is therefore the process of applying Information Visualisation techniques to network measurement data, in order to develop methods of displaying this data in a more intuitive format which is more meaningful to a network manager or network operator. It is therefore likely to play an important role in future network performance and diagnosis methods to convey information about the system which simple numerical metrics alone cannot.

5.6.1 Visualisation of TCP

The most common method of visualising the performance of a TCP connection is through the use of a TCP Time-Sequence graph. Figure 65 illustrates a Stevens' [Stev94] TCP Time-Sequence graph, plotted using experimental data obtained from the testbed in Chapter 4, for a 150 msec RTT link with 15% 0.201 sec forward path packet reordering. The Stevens' graph plots observed sequence numbers against time and, therefore, allows simple analysis of the rate of change of sequence numbers. During periods of throttled congestion control, or retransmission, the gradient of the graph will decrease, indicating a drop in TCP throughput.

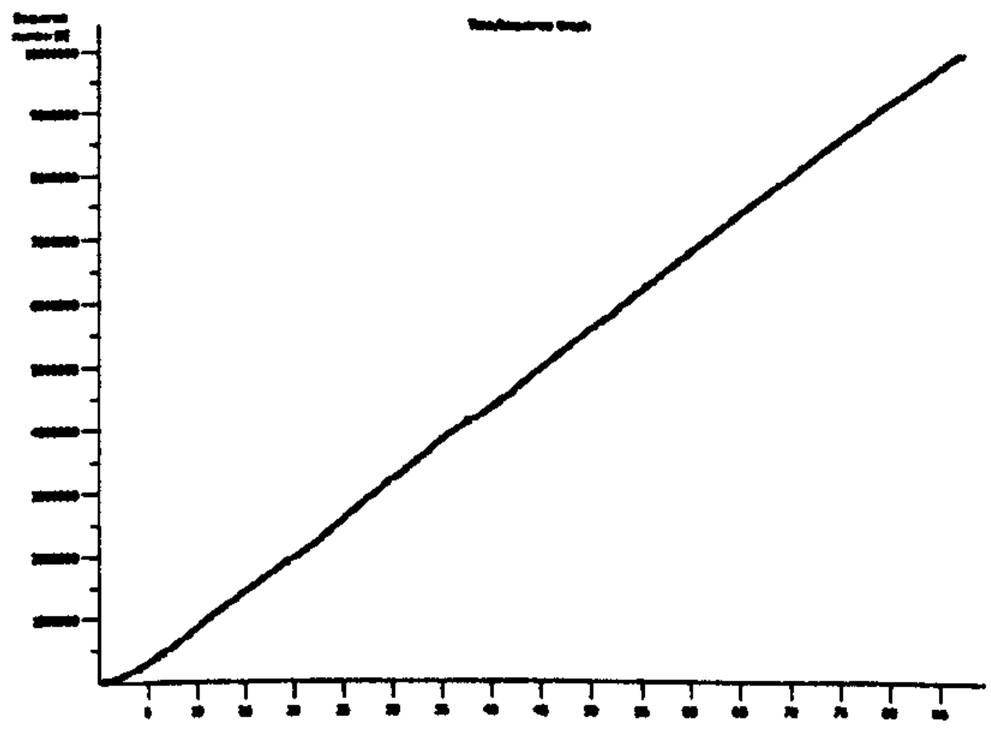

Figure 65 - Stevens' TCP Time-Sequence Graph

A TCP Time-Sequence graph may also be plotted using the Ostermann technique [Park98], as performed by the TCPTrace tool [Oste08]. An Ostermann Time-Sequence graph, as illustrated in Figure 66 and Figure 67, improves on the Stevens' technique by

plotting both Acknowledgement Numbers received from the Receiving TCP and also the *rwnd* advertised by the Receiving TCP. The latter is drawn at the Sequence Number value corresponding to the sum of the acknowledgement number and the *rwnd* advertised from the last Ack packet received. Tick lines in Figure 67 represent segments sent and, therefore, vertical groups of tick lines indicate a volley of packets launched into the network at once.

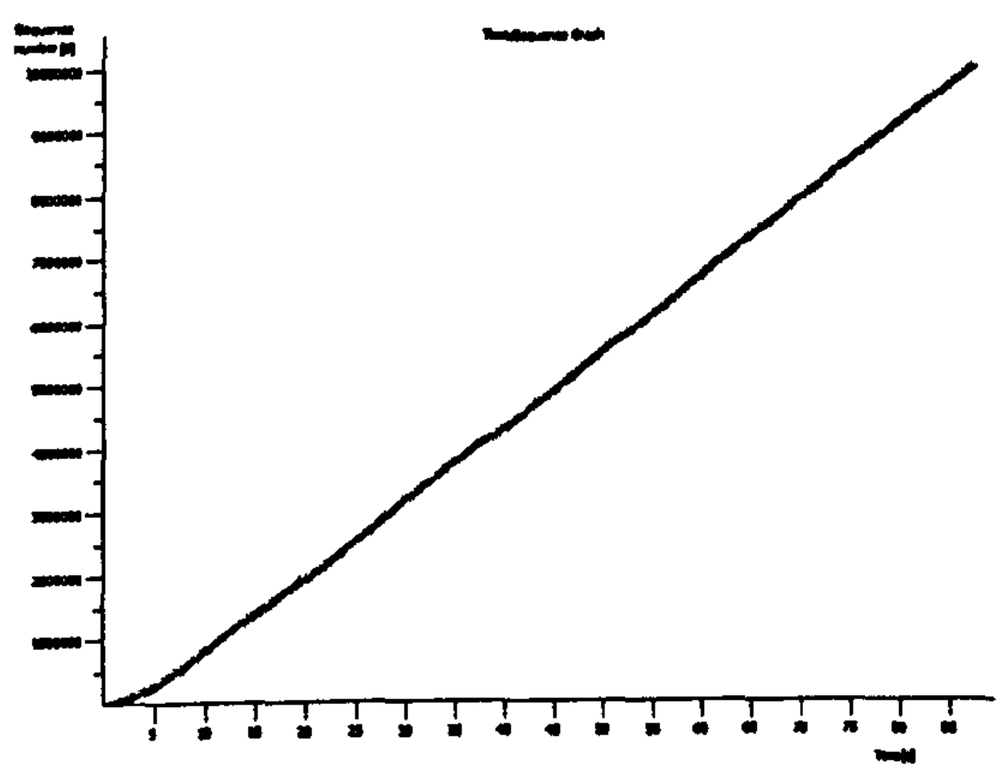

Figure 66 - Ostermann Time-Sequence Graph

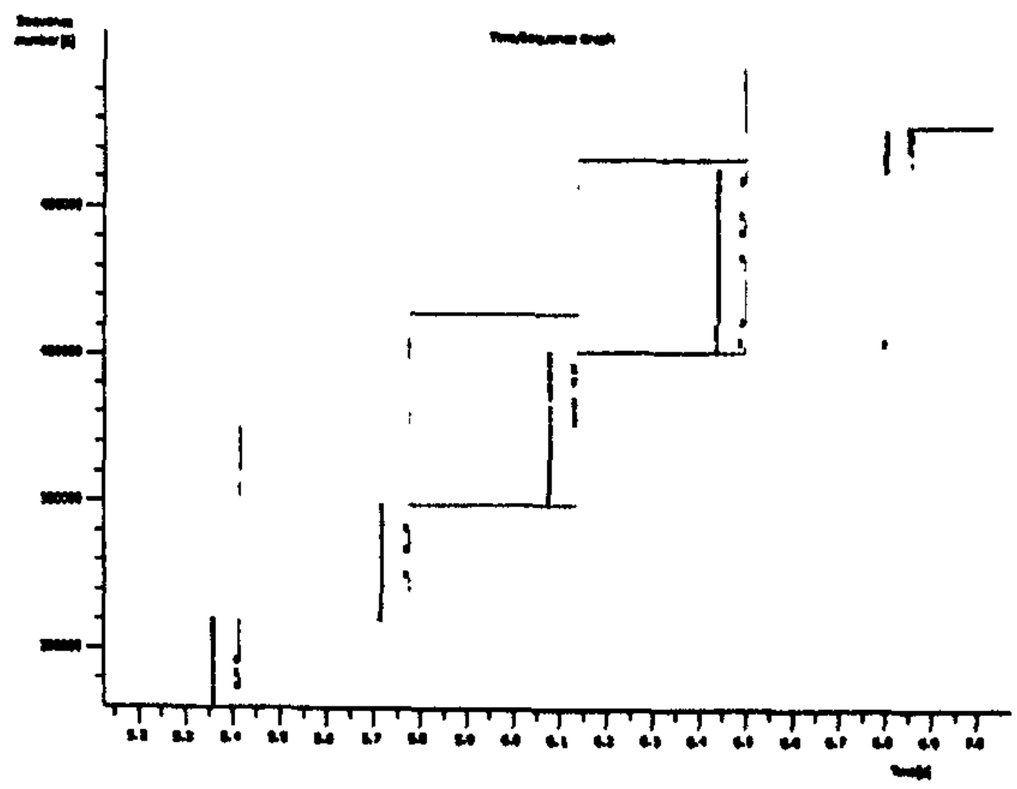

Figure 67 - Zoomed Ostermann Time-Sequence Graph

Neither of these techniques are appropriate for analysing TCP during packet reordering. In Figure 67 at time 6.8 seconds, a segment can be seen to arrive 0.3 seconds later than the other segments within that congestion window, therefore indicating that packet reordering has occurred on this segment. This is difficult to observe, unless the graph is plotted with a very fine resolution and does not intuitively highlight to the network operator that reordering is occurring in the connection.

5.6.2 Visualisation of TCP Packet Reordering

RFC 5236 [Jaya08] defines two metrics to calculate and visualise the degree of packet reordering which a TCP stream has undergone. As discussed in Section 3.4.3, 'Reorder Density' shows the distribution of displacement of packets from their original positions, and 'Reorder Buffer-Occupancy Density' displays the histogram of the occupancy of a hypothetical buffer used to re-sequence early arriving packets.

The two metrics proposed in RFC 5236 differ significantly from the packet reordering metrics standardised in RFC 4737 [Mort06]. The reordering metrics in RFC 4737 are as listed in Chapter 3, Table 1. Each RFC 4737 metric is a 'lateness' based metric; that is packets which arrive 'late' are highlighted by this metric, rather than packets which appear to arrive 'early'. This is similar to the Paxson metric for packet reordering discussed in Section 3.2.2. As each packet arrives, it is checked against the last 'non-reordered packet'. If the sequence number is greater than the last non-reordered packet, then that packet is marked as being in-order, and becomes the new non-reordered packet. Consider the sequence of arriving packets 1, 6, 2, 3, 4, 5, 7, 8, 9, 10. Packets 1, 6, 7, 8, 9, 10 are marked 'in-order', while packets 2, 3, 4, 5 are marked 'reordered'. Simply calculating a percentage of packets by marking them as 'in-order' has been illustrated in Chapter 4 as not sophisticated enough to describe the effects of reordering on TCP. Therefore, the extent or movement of packets, from their original positions, is calculated by many of the metrics proposed in Chapter 3. Metrics which quantify this 'offset' in terms of packets, by which a packet is reordered, are useful for determining the portion of reordered packets that can or cannot be restored to order by the receiving host's buffer[Mort06].

'Reordering Extent', is defined in RFC 4737 as the maximum distance in packets, from a late packet until the earliest packet received that has a larger sequence number. It is therefore similar to the Paxson metric, but additionally measures the number of positions by which a packet is reordered. Table 13 illustrates the Reordering Extent calculation for a series of 10 packets. As is shown, if a packet is in order or, if it arrives early, the Reordering Extent is undefined.

Sent Order	1	2	3	4	5	6	7	8	9	10
Received Order	1	6	2	3	4	5	7	8	9	10
RFC 4737 Reordering Extent	Un-defined	Un-defined	4	3	2	1	Un-defined	Un-defined	Un-defined	Un-defined

Table 13 - RFC 4737 Reordering Extent

The 'TCP-Relevant' metric defined in RFC 4737, known as the 'n-reordering' metric,

defines the percentage of packets which are reordered by a distance ≥ n packets where, if n = 3, a NewReno sender would consider the packet lost for the purposes of congestion control. N-reordering defines the extent as the maximum distance in packets, from the reordered packet to the earliest packet that has a larger sequence number. This metric has been argued to be ambiguous [Pira08], as packets are considered in sets. If two or more consecutive packets are late, but maintain position with respect to each other, then only the first packet is marked as reordered.

A packet is late if one of, or a consecutive set of, its immediate preceding packets have higher sequence numbers. However, as shown in Table 14, although packets 2, 3, 4 and 5 are all defined as 'Reordered' by the Type-P-Reordered metric, only packet 3 is defined as n-reordered. Therefore the n-reordering metric is inconsistent with other metrics in RFC 4737, as it does not correctly identify all late reordered packets.

Sent Order	1	2	3	4	5	6	7	8	9	10
Received Order	1	6	2	3	4	5	7	8	9	10
RFC 4737 Reordered?	N	N	Y	Y	Y	Y	N	N	N	N
RFC 4737 n-reordering	0	0	4	0	0	0	0	0	0	0

Table 14 - RFC 4737 n-Reordering

It has been argued [Pira08] that lateness-based packet reordering metrics, as shown in Table 13 and Table 14, are not appropriate, as early packets should also be identified as reordered. Consider the arrival sequence in Table 14. If a metric is based on early arrivals, then packet 6 is identified as reordered. Alternatively, a lateness-based metric will mark all packets from 2 to 5 to be out of order, even although the most likely cause of the phenomenon is that the single packet arrived early. The metrics in RFC 4737 are lateness based metrics, but even these, such as n-reordering, will only capture a subset of late arriving packets, as sequences of several consecutive late packets will be marked in-order.

5.6.2.1 RFC 5236 – Improved Packet Reordering Metrics

RFC 5236, Improved Packet Reordering Metrics, defines two methods to describe both late and early reordered packets in a TCP connection; Reorder Density and Reorder Buffer-Occupancy Density.

5.6.2.2 Reorder Density

Reorder Density (RD) describes the distribution of displacement of packets from their original positions. As a sequence of packets arrives at a measurement point, a *receive_index* is assigned to each non-duplicate packet. The *receive_index* is an integer number assigned to each packet, which is calculated to describe the original order in which the packets were transmitted. For TCP, this would be calculated using the Sequence Numbers.

receive_index	1	2	3	4	5	6	7	8	9	10
Received Order	1	6	2	3	4	5	7	8	9	10
Displacement	0	-4	1	1	1	1	0	0	0	0

Table 15 - Reorder Density Example

If the *receive_index* assigned to packet m is $(m + d_m)$, with $d_m != 0$, then a 'reorder event' has occurred and this event is denoted by $r(m,d_m)$. Packet m is late if this offset $d_m>0$, early if $d_m<0$ and, in order if $d_m= 0$. Therefore, packet reordering in a sequence of packets is completely represented by the union of reorder events, *R*, referred to as the 'reorder set':-

$$R = \bigcup_m \{r(m, d_m) \mid d_m \neq 0\}$$

Equation 9 - Reorder Density, Reorder Set

Therefore, the Reorder Set for Table 15 is defined by Equation 9 to be $R = \{(6, -4), (2, 1), (3, 1), (4, 1), (5, 1)\}$.

Reorder Density (RD) is defined as the histogram of the Reorder Set R, normalised with respect to the total number of packets, adjusted for losses and duplicates. Therefore, RD for Table 15 can be calculated to be RD[-4] = 1/10 , RD[0] = 5/10, RD[1] = 4/10.

5.6.2.3 Assigning *receive_index* Values

One complexity with this technique is the calculation of the *receive_index* values as packets arrive within a flow. If a flow is short in duration, it is possible to assign *receive_index* once the flow has completed. However, for real-time monitoring of packet reordering, *receive_index* values must be assigned to packets as they arrive at the probe. This is an important consideration when network duplication or retransmissions occur, as they appear as additional packets in the 'Received Order' counter, thus making this counter out-of-step with the *receive_index* values.

RD uses a threshold, D_T to decide when to declare a packet as lost, as in many applications such as TCP or VoIP, a packet reordered beyond a certain displacement is considered lost anyway. This same threshold is used to maintain a buffer of early-arriving packets, to allow identification of network duplicates. If a packet is not received within D_T packets, it is considered to be lost.

Two methods are proposed to assign the D_T threshold for monitoring real-time flows[Jaya08]. The 'Go-Back' D_T method applies the rules at each packet arrival, and if a packet that was supposed to arrive D_T places ago does not arrive, then the sequence number is removed from the *receive_index* and RD is recomputed for the previous D_T steps. The 'Stay-Back' D_T method performs RD calculation by lagging the computation of arrival packets by D_T positions, therefore not requiring any further adjustments, as a missing packet can be immediately declared as lost.

5.6.2.4 Reorder Buffer-Occupancy Density

Reorder Buffer-Occupancy Density (RBD) is defined in RFC 5236 as the normalised form of the occupancy of a hypothetical buffer, which could be used by a receiving TCP to recover from the out-of-order arrival of packets. Packets which appear to arrive 'early', are placed in this buffer until intermediate packets arrive to fill any resulting gaps. The occupancy of this buffer, measured in packets, is used as a packet reordering metric.

For the sequence of arrivals illustrated in Table 16, when packet sequence number 6 arrives at position 2, the packet is stored in the hypothetical buffer until packet sequence number 5 arrives, which then allows the release of packet sequence number 6 to the application. Therefore, the density of the buffer is calculated as 1, for four packet arrivals and 0 for all other arrivals: RBD[0]=6/10, RBD[1]=4/10.

Sent Order	1	2	3	4	5	6	7	8	9	10
Received Order	1	6	2	3	4	5	7	8	9	10
RBD Buffer Occupancy	0	1	1	1	1	0	0	0	0	0

Table 16 - Reorder Buffer-Occupancy Density Example

RBD requires a similar threshold to RD to define a maximum extent, both late and early, that a packet can appear before it is assumed to be lost. This is a requirement for this metric to operate, and to perform calculation in real-time. This threshold B_T, describes the maximum number of packets that can be stored in the hypothetical buffer. If the buffer is already filled to B_T, the packet is considered to be delayed more than the threshold and is considered to be lost. As each packet arrives, the sequence number is compared with a NESN and the existing contents of the hypothetical buffer. If the Sequence Number is lower than the NESN, or is identified as a duplicate or retransmission of a packet in the buffer, it is classified as such, and not considered in the calculation of RBD.

5.6.3 The Arthur Visualisation Technique

Calculation of ∂EP-∂OP during the Arthur Classification Algorithm, provides a useful method of visualising TCP packet reordering, as demonstrated in Figure 68. The effect of calculating ∂EP-∂OP is to effectively 'cancel out' the packets that have arrived in perfect sequence, as these will have ∂EP-∂OP=0. This results in a graph which indicates only the positions in the Flow Trace where sequence breakdown was apparent and the magnitude at which these breakdowns have occurred.

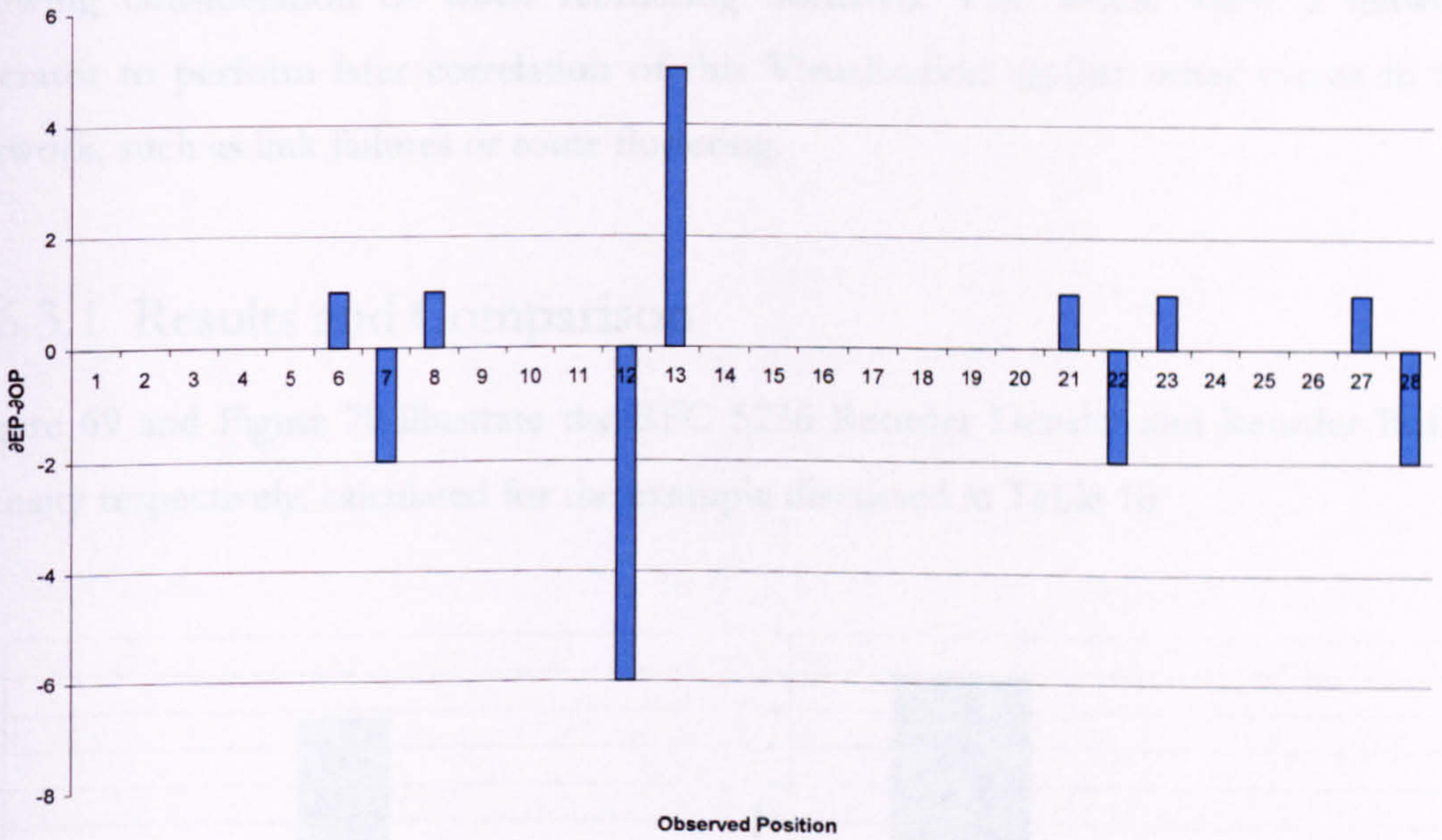

Figure 68 – Arthur Visualisation of TCP Reordering

Figure 68 plots the Arthur Visualisation for the Flow Trace example discussed in Table 10, over the range of observed OP values. Positions on the graph below the *x* axis indicate packets that were considered to have arrived with observed positions 'later' than their expected positions, as is the case for reordered packets or those that have been retransmitted. Positions on the graph above the *x* axis indicate packets that were considered to have arrived with observed positions 'earlier' than their expected positions.

As discussed in Section 5.6.2, it is important to consider how both early and late packets

are reordered; both will have an effect on the receiver's ability to recover from reordering. The Arthur Visualisation allows this ability. In common with the two metrics proposed in RFC 5236, it allows identification of both early and late packets, but additionally allows indication of when the packet reordering occurred during the lifetime of the connection.

Graphs with many peaks are an indication of poor packet sequencing and hence poor link quality. Graphs with the majority of points on the 0 line of the x axis, are an indication of good packet sequencing and high link quality. The Arthur Visualisation allows plotting ∂EP-∂OP against either OP, or against a measure of time, thereby allowing consideration of when reordering occurred. This would allow a network operator to perform later correlation of this Visualisation against other events in the network, such as link failures or route fluttering.

5.6.3.1 Results and Comparison

Figure 69 and Figure 70 illustrate the RFC 5236 Reorder Density and Reorder Buffer Density respectively, calculated for the example discussed in Table 10.

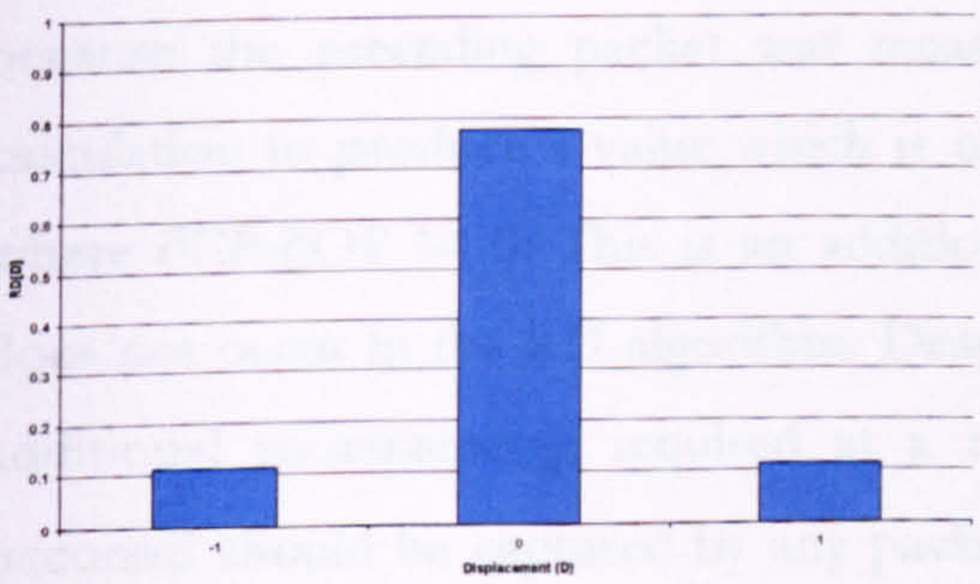

Figure 69 - Reorder Density

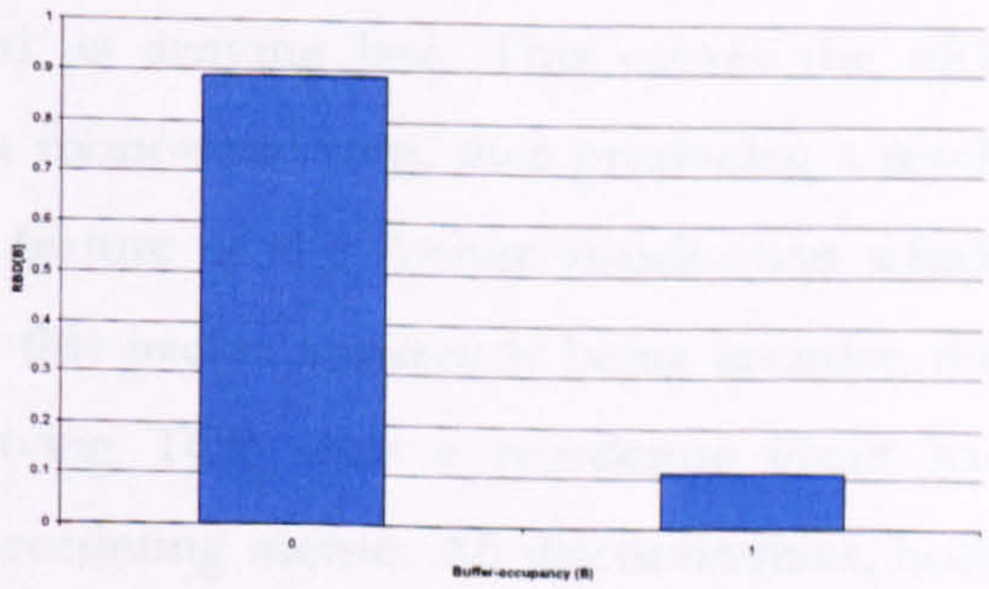

Figure 70 - Reorder Buffer Density

Figure 69 indicates that 21 packets were measured with RD[0], 3 packets with RD[-1], and 3 packets with RD[1]. Clearly this accounts for only 27 packets out of the 28 illustrated in the same Flow Trace illustrated in Figure 68. The reason being that the packet at OP 12 in Figure 68 is a retransmission of packet OP 7 (EP 6), which was reordered to such an extent so as to cause a retransmission at OP 12. The RD algorithm

discards all retransmissions and duplicates and thus does not consider them when producing a visualisation of the TCP connection. The RD algorithm identifies RD[-1] for packets 7, 21 and 27, and RD[1] for packets at positions 6, 20 and 26. When comparing these results with Figure 69, it should be noted that the RD algorithm has discarded packet 12, and therefore RD packet numbering above 12 is one less than that shown in the Arthur visualisation in Figure 68.

Figure 70 plots the Reorder Buffer Density for the example discussed in Table 10, indicating that RBD[0] was measured for 24 packets, and RBD[1] was measured for 3 packets. Similarly with RD, the retransmitted packet 12 has been discarded. RBD[1] was measured for packets 6, 20, and 26. Again consideration must be given to the change in numbering after the retransmission at 12 and, therefore, 20 and 26 correspond to 21 and 27 in Figure 68.

Comparison of these results suggest that the RBD measure corresponds naturally with the peaks shown in Figure 68. The peaks in the Arthur Visualisation, identifying early packets, correspond with the buffered packets identified in the RBD algorithm, but in addition, three extra 'early' packets are identified. The packets at positions 5, 13 and 23 are each shown to be arriving early due to ∂EP-∂OP analysis, but the reason for this is because the preceding packet was measured as arriving late. This causes the ∂EP calculation to produce a value which is not a monotonic step, thus producing a result where ∂EP-∂OP != 0. This is an additional feature of the Arthur visualisation which does not occur in the RD algorithm. Despite this packet apparently being in-order, the additional re-sequencing required at a receiving TCP after a reordering event has occurred should be captured by any packet reordering metric. All discontinuities, both early and late, must be considered in order to fully convey the breakdown in sequence and the possible extra processing required at the receiver, in order to repair this breakdown. By allowing an immediately succeeding packet to be marked as 'early' after a late packet has arrived, the Arthur algorithm is significantly less computationally intensive than RD and RBD. Both RD and RBD require the ability to identify and remove retransmissions or duplicates which, as discussed in Section 5.6.2.2, greatly increases the complexity when performing measurements in real time.

RD fails to convey the number of retransmissions in a TCP connection visually, which

is an important aspect of the health of a connection. RD also relies on the user to determine the value of D_T, the value required to differentiate between an upstream reordered or retransmitted packet. Should the user choose an incorrect value for a particular TCP connection, the RD algorithm will confuse upstream reordering and retransmission and will not accurately report either event.

Figure 71 illustrates a further example of ∂EP-∂OP analysis, employed on a Flow Trace recorded from the midpoint of the experimental testbed described in Chapter 4. The Flow Trace illustrates transfer of a 10 Megabyte file, over an RTT of 20 msec, with 10% reordering applied at 1 msec reordering delay.

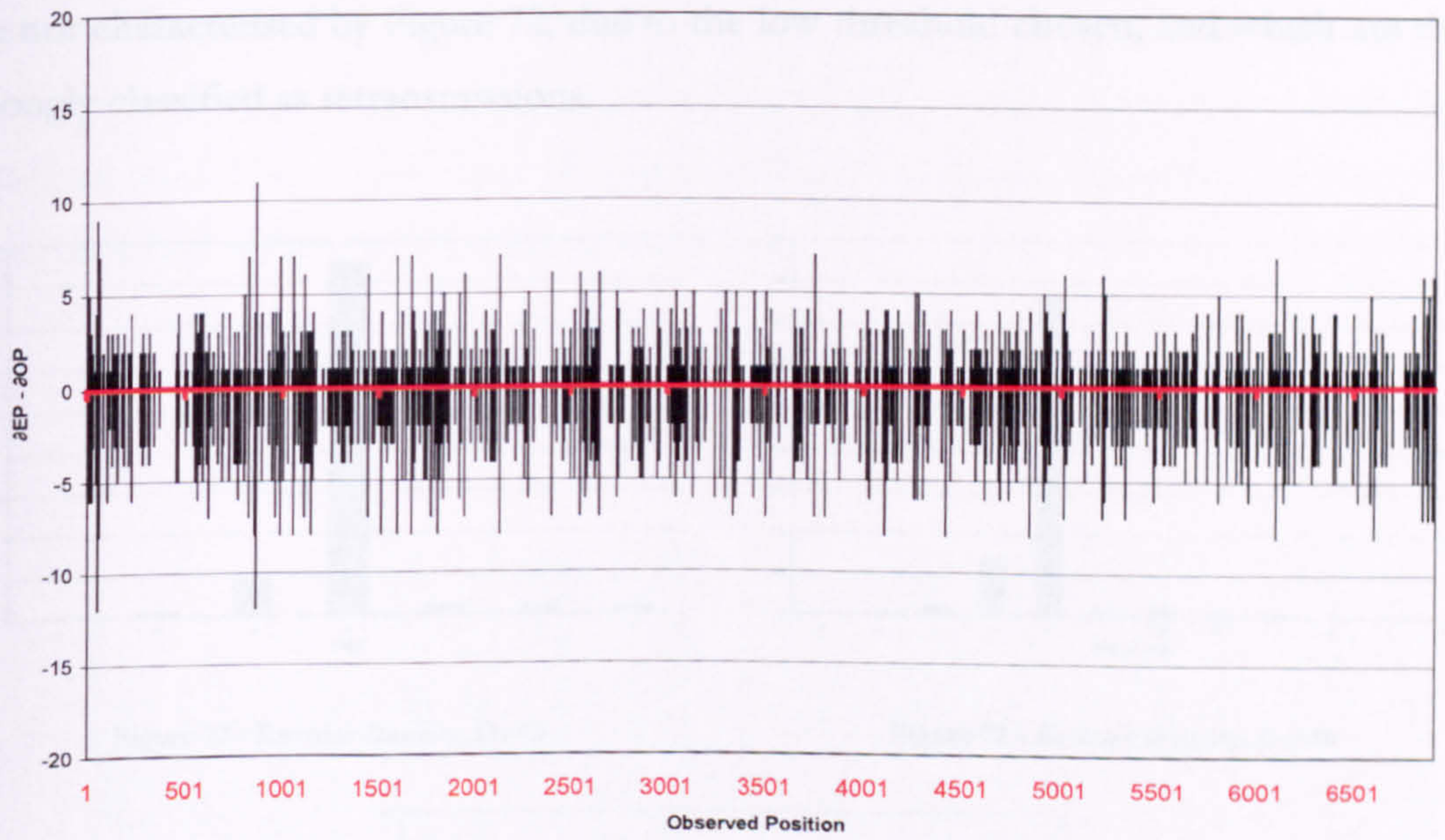

Figure 71 – 20 msec RTT, 10% Reordering, 1 msec Reordering Delay

By applying the Arthur Classification Algorithm, a total of 6 packets were classified as Result 2 (Retransmission due to Reordering), with 3 packets each classified under Result 7 (Retransmission due to Fast Retransmission) and Result 8 (Retransmission due to RTO). Visual inspection of Figure 71 indicates that the majority of packets appear less than 3 positions late or early, with significant periods of the flow where perfect sequencing has been achieved and ∂EP-∂OP = 0.

The calculation of ∂EP-∂OP also provides an excellent indication of the time periods during the transmission of the 10 megabyte file, when sequence breakdown occurred,

indicating that the reordering appears bursty; there are both periods where sequence breakdown occurs and other periods where perfect sequencing has been re-established and maintained.

Comparison of Figure 71 with Figure 72, Figure 73 and Figure 74, allows evaluation of the Arthur Visualisation, with both RD and RBD calculation for the same TCP flow. Figure 72 and Figure 73 plot RD for various values of D_T in order to demonstrate the importance of choosing a suitable threshold for determining loss and reordering when using the RD metric. It is suggested in the literature [Pira08] that $D_T = 3$ should be used in order to differentiate packets which will cause a New Reno receiver to signal Fast Retransmit. Figure 73 indicates that there are a significant number of late packets which are not characterised by Figure 72, due to the low threshold chosen, and which are then wrongly classified as retransmissions.

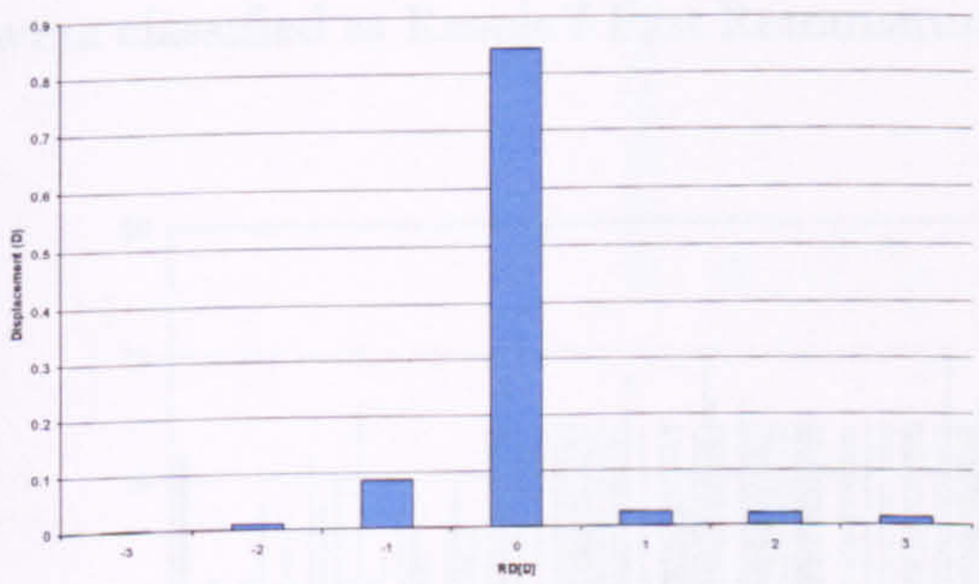

Figure 72 - Reorder Density, D_T=3

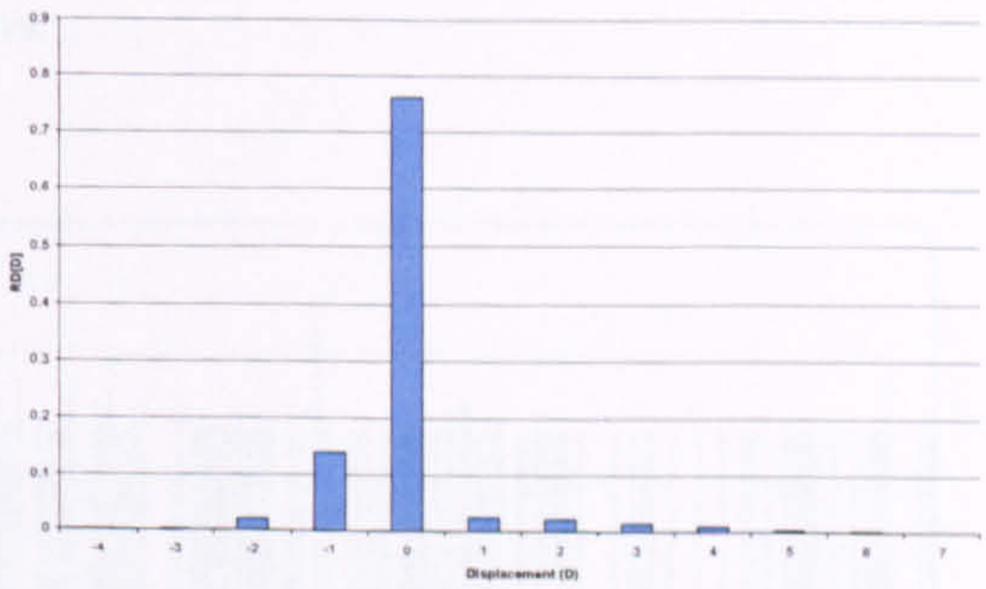

Figure 73 - Reorder Density, D_T=10

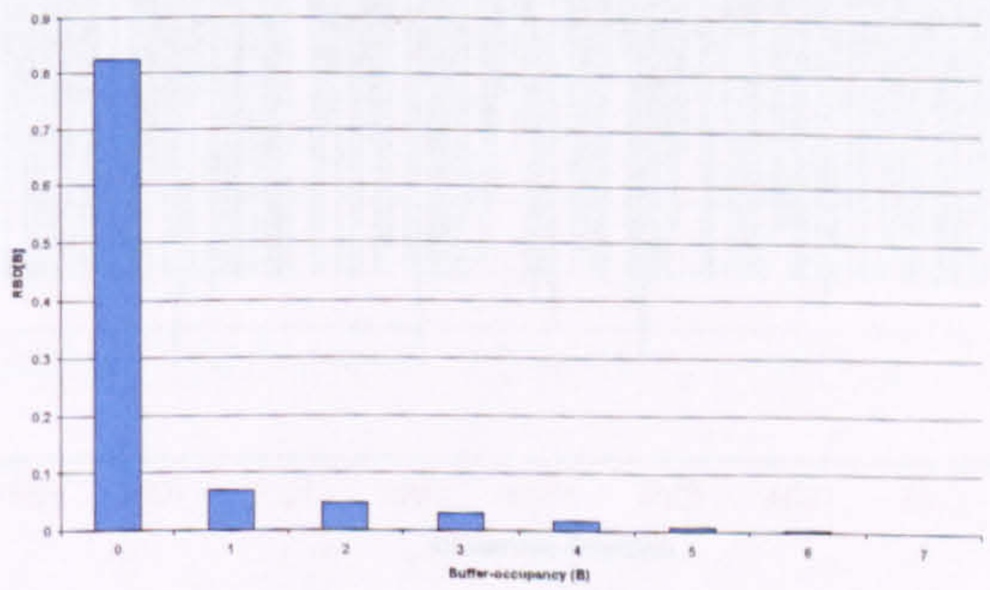

Figure 74 - Reorder Buffer Density

Figure 74 plots the RBD metric for the data shown in Figure 71, where the RBD buffer limit was set to 10. The graph indicates that the maximum buffer required to store an early packet was 7 packet positions, which corresponds with Figure 71, where the

majority of early and late packets are under 5 packet positions out-of-sequence. As discussed previously, RBD discards retransmitted packets, and therefore, unlike Figure 71, does not provide indication of the number of retransmissions or how late these retransmissions were with respect to their originally intended positions. It should be noted that in Figure 74, RBD was calculated to have a Buffer Occupancy of 7 for only 4 packets out of the total of 6911 packets analysed, and therefore RBD[7]=0.000579. This highlights another deficiency with the RBD metric, as when plotted in comparison to RBD[0]=0.8256 with 5706 packets, it is very difficult to observe the few outlying highly reordered packets which could have highly negative effects on a TCP connection.

Figure 75 illustrates a third example of the Arthur Visualisation ∂EP-∂OP analysis, applied to another TCP stream obtained from the testbed, where Reordering Delay has been increased by a factor of 10. Application of the Arthur Classification Algorithm, indicated that 172 packets were classified as Result 2 Retransmissions, of which 162 were classified as Result 7 Fast Retransmissions.

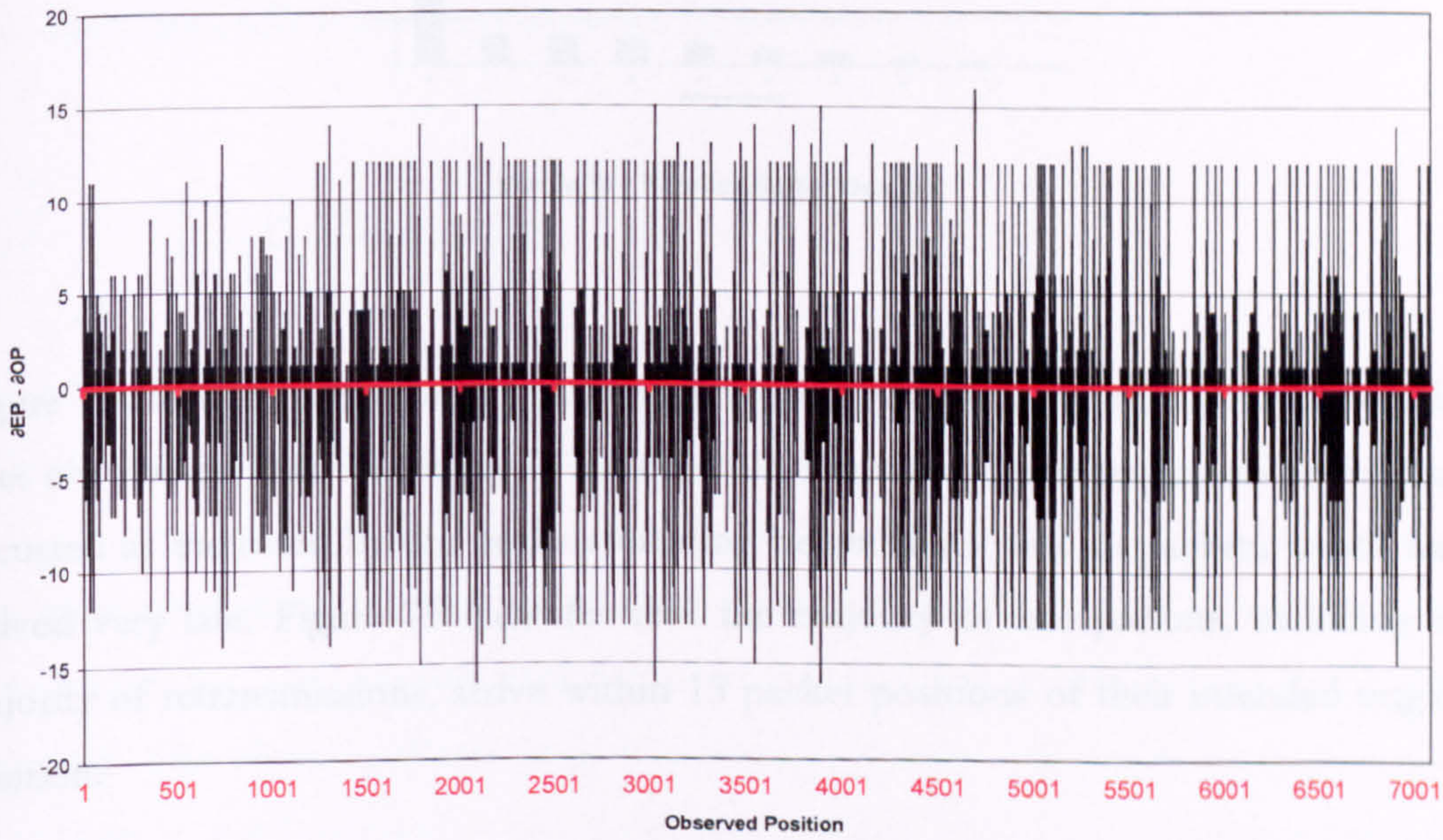

Figure 75 - 20 msec RTT, 10% Reordering, 10 msec Reordering Delay

Figure 75 illustrates that the majority of packets in this connection are out of sequence, with many packets distributed over a large range of ∂EP-∂OP values. This is reflected in the range of displacements indicated in the equivalent RD metric, as shown in Figure 77

and the equivalent RBD metric, as shown in Figure 78. Figure 76 indicates that the choice of $D_T = 3$ is too conservative to describe the range of reordering events which has been induced in this test stream.

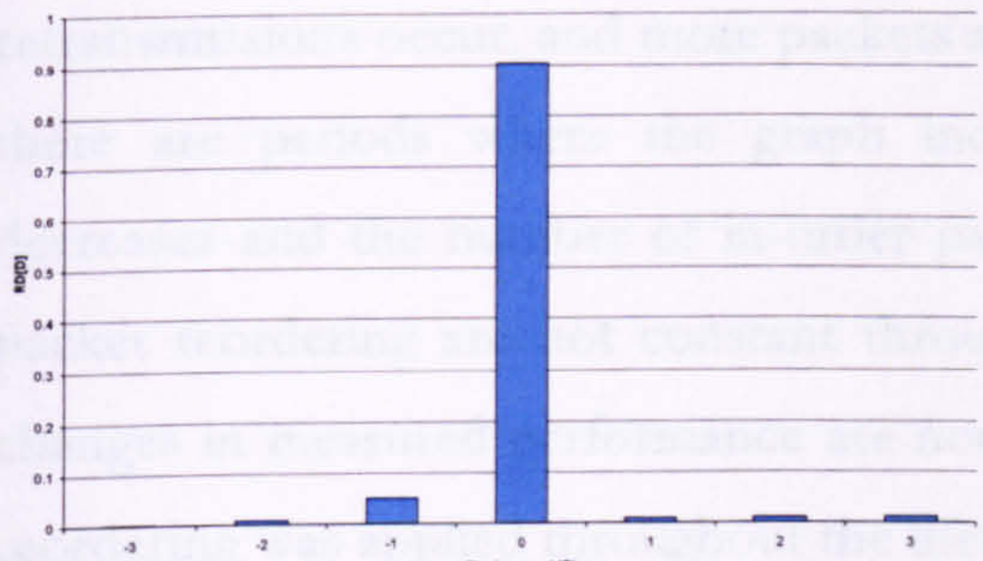

Figure 76 - Reorder Density, D_T=3

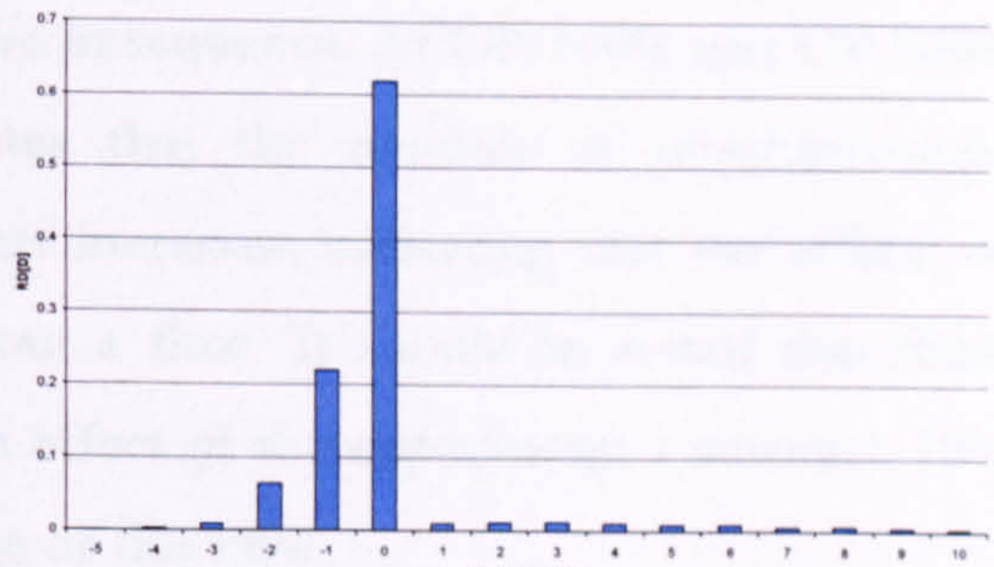

Figure 77 - Reorder Density, D_T=10

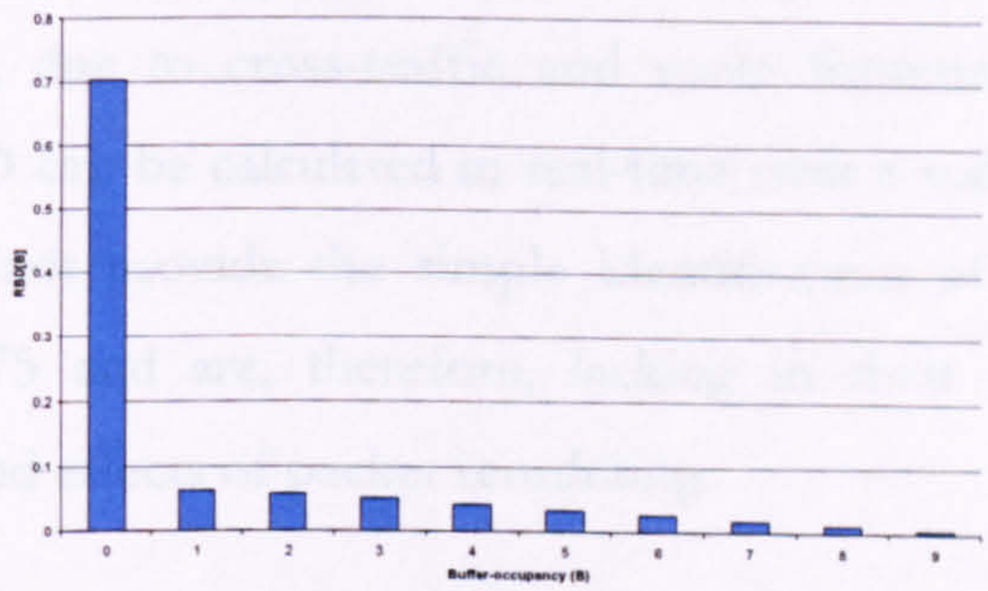

Figure 78 - Reorder Buffer Density

Figure 75 illustrates a number of properties of the Arthur Visualisation which RFC 5236 does not possess. Firstly, Figure 75 indicates the number of retransmissions which have occurred as indicated by the peaks appearing below the x axis as packets which have arrived very late. Figure 75 indicates that the majority of late packets, including the majority of retransmissions, arrive within 13 packet positions of their intended original position.

Figure 75 also illustrates a second important property of the Arthur Visualisation technique. From discussion in Chapter 4, it is known that packet reordering is a phenomenon which is path dependent and which can vary rapidly over time. Figure 75 indicates that, even when a fixed amount of reordering is applied, a connection's performance can degrade, over time as can be seen by the distribution of ∂EP-∂OP over

the first third of the graph. During this period, fewer packets are shown to be extremely late or early, thus providing in an insight into the degenerative nature of packet reordering on a connection. It should be noted that there are also periods where there is a relative improvement in the connection's performance, but where fewer retransmissions occur, and more packets arrive in sequence. At OP 3400, and OP 5600, there are periods where the graph indicates that the number of retransmissions decreases and the number of in-order packets increases, indicating that the effects of packet reordering are not constant throughout a flow. It should be noted that these changes in measured performance are not an effect of the experiment; a constant 10% reordering was applied throughout the lifetime of this flow.

On a live Internet, the effects of packet reordering are likely to be even more changeable over time, due to cross-traffic and route fluttering. In these situations, although RD and RBD can be calculated in real-time over a rolling window of packets in a stream, they do not provide the simple identification of changes over time as illustrated in Figure 75 and are, therefore, lacking in their usefulness as tools to investigate the cause and effects of packet reordering.

5.7 Conclusions

This Chapter has presented a number of contributions in the field of mid-point passive TCP measurement.

The taxonomy of packet reordering metrics and methods has been reviewed and the motivations for the development of passive mid-point measurements have been discussed. The clear significant benefit of using a passive mid-point measurement, is that it can be used to measure the behaviour of many thousands of concurrent flows from a single point in the network. Unfortunately, each of the current proposed mid-point measurements in the literature make several assumptions which could significantly affect their operation and the accuracy of results that they produce.

The Love mid-point Goodput measurement has been argued to overestimate the

amount of loss and retransmission experienced by a TCP connection, as the algorithm does not consider the effects of packet reordering. The Jaiswal Out-of-Sequence Classification algorithm is a significantly more advanced mid-point measurement of TCP, but has the requirements of symmetrical Data and Ack packets passing through a single mid-point measurement device, and also the requirement to estimate the RTT and RTO for every single observed flow. There is clearly a need for accurate, lightweight single-point measurements of TCP loss, goodput, retransmission and reordering; the current Love technique is too simplistic, whereas the Jaiswal technique's requirements result in only a small fraction of connections being analysed.

A passive mid-point monitoring technique has been developed, which is lightweight in both its storage requirements and processing overhead at the mid-point measurement probe. The passive mid-point technique has described the capture and storage of Flow Traces, which can then be analysed to provide an improved measure of Goodput, Loss and Retransmission.

The Arthur Classification Algorithm has been developed and performs analysis of the packet Flow Traces captured using the passive mid-point technique, and classifies packets into eight different results. These results identify Network Duplicate packets, Retransmitted packets, Lost Packets and Reordered Packets, and attempt to explain the cause of each event, based entirely on knowledge inferred from the Data path of the TCP connection. Clearly the ability to perform classification on only the Data path is a significant contribution beyond the Jaiswal algorithm, as it avoids the need of capturing and processing Acknowledgement packets at the same network tap. However, working on such a smaller subset of data limits the number of classifications that can be performed, as it is impossible to infer certain TCP end host states based only on the Data packets.

The mid-point measurement technique and classification algorithm have been implemented in order to perform comparison with Jaiswal during a live Internet traffic experiment. The difficulties of validating such mid-point algorithms has been discussed; on a large scale experiment, it is impossible to instrument all end hosts and later correlate the actual behaviour with the mid-point inferred behaviour. Experiments

indicated that although the Arthur Classification Algorithm tended to over-estimate the number of out-of-sequence packets, this is due to the algorithm reporting packets immediately after a retransmission as 'late' and after a loss as 'early'. The number of Retransmissions measured were found to be similar to Jaiswal, but the ability of the Jaiswal algorithm to identify retransmission due to Fast Recovery is a feature which the Arthur algorithm cannot provide. In these instances, the Arthur algorithm will overestimate the number of retransmissions due to Fast Recovery.

This chapter has argued that there is clearly a trade-off to be made in mid-point measurements between accuracy and complexity, and that due to the variations in end host TCP implementation, non standard compliant hosts and many additional TCP features, building a complex mid-point TCP measurement algorithm that can cater for these many variables is extremely complex. Therefore, the simplicity of the Arthur algorithm, the requirement for only Data packets without the necessity for Acknowledgements, the simple classification rules and the lightweight processing required at the measurement host, make this an attractive algorithm compared to both Love and Jaiswal.

Network Visualisation Techniques have been discussed as an important research area which can describe the overall performance of a network in a more intuitive method than numerical metrics, and examples of TCP visualisations have been illustrated. The packet reordering metrics proposed in RFC 5236 have been discussed and compared with those in RFC 4737; it has been argued that a metric for packet reordering should describe both late and early packets.

The Arthur Packet Reordering Visualisation Technique has been demonstrated and compared with both Reorder Density and Reorder Buffer Density. The Arthur technique improves on the techniques in RFC 5236 by first indicating the number of retransmissions which have occurred in a connection; retransmissions are the ultimate indicator of the health of a TCP connection and cannot be ignored. Secondly, the Arthur technique has illustrated the ability to show packet reordering during the lifetime of a connection. Through examples it has been shown that despite a constant amount of reordering applied, the effect on a TCP stream can vary over time. It is therefore very

important to be able to correlate a metric of packet reordering with time, in order to perform investigation of the causes and effects of packet reordering.

Clearly there is a need for future work and development in the area of mid-point passive packet reordering measurements. As with the development of TCP and the Fast Retransmit algorithm, network measurement techniques have been developed under the assumption that TCP packet reordering is a phenomenon which does not often occur. Although Jaiswal and subsets of his work attempt to address this issue, a passive mid-point technique must be lightweight and simple in order to be of use to a Network Operator. This chapter has developed novel patented techniques for the classification and visualisation of packet reordering, and has progressed the state of the art to address these issues.

Chapter 6

Measuring the Impact of Packet Reordering

6.1 Introduction

It is clear from Chapter 4 and Chapter 5 that packet reordering on the Internet can have a measurable effect on the performance of a TCP flow, and that there are many varying methods proposed in order to measure and describe the amount of reordering occurring within an end to end path.

Chapter 4 has highlighted one area which has received little attention to date. While many different methods have been proposed in order to measure and classify the

amount of reordering occurring at the packet level as classified in the taxonomy presented in Chapter 3, very little work in the literature has attempted to correlate across layers to measure this, and the resulting effects on the end user application. Although it is argued that many studies are indeed relevant to TCP, such as the RFC 4737 'TCP-Relevant Metric', the simple notion of regarding a packet greater than three positions out of sequence as lost, is over-simplistic and requires more detailed investigation. Chapter 4 has made some advances in understanding the real impact of packet reordering on a single TCP connection; in this Chapter, a less prevalent, but significantly more complicated type of network traffic is now considered.

Chapter 4 has also highlighted that the next main driver of packet reordering may be end-to-end wireless technologies such as WiFi and WiMax, where parallelism at all layers is likely to increase. Wireless networks are especially prone to problems due to the higher levels of link layer retransmissions found in noisy wireless environments. Wireless links are very different from traditional wired links; the steady-state dropping and reordering probability are independent from link congestion and, so traditional assumptions that loss indicates congestion are invalid.

An investigation into the behaviour of video traffic over UDP in situations of high packet reordering is now considered. Video traffic is significantly more complicated that a simple TCP session, due to the temporal inter-packet dependencies introduced by the MPEG video encoding structure.

An experimental investigation into the effects of video packet reordering using the Windows Media streaming system is presented. A method for invoking packet reordering is introduced along with a tool for client-side measurements of video quality is presented. Typical measurements of video performance undergoing reordering are shown, with a study of buffering occupancy at the client and the potential impact this could have on video packet reordering is demonstrated.

This chapter discusses a measurement study which was presented at the International Symposium on Wireless Communications Systems (ISWCS), September 2004, and development of a testbed which has been published in IEE Electronics Letters, May

2002, and European Personal Mobile Communications Conference (EPMCC), April 2003.

6.1.1 Wireless as a Driver for Packet Reordering

In the near future, multimedia traffic is expected to represent a substantial percentage of the data carried on both mobile and fixed communications systems. Due to the predictive coding structures employed, video over IP has its own critical timing characteristics that make it highly susceptible to loss and varying delay. At any point where inter-packet latency cannot be guaranteed, it will prove exceedingly difficult to deliver real-time video streaming with guaranteed Quality of Service.

Packet reordering is a symptom not only of fixed networks; the effects of reordering must also be addressed when considering other types of IP network, for example wireless. TCP is known to suffer performance degradation in mobile wireless environments[Xylo99], which typically have high bit error rates (BERs) and mobility induced disconnections. Moreover, wireless channels are afflicted with significant delay variations, due to factors such as link-layer retransmissions in radio access networks. In a wireless environment, a non-optimised TCP or UDP streaming implementation would mis-interpret a re-transmitted datalink frame as a reordered network layer packet. Without appropriate cross-layer feedback [Rais02b], TCP may interpret this apparently reordered packet as lost and would then invoke unnecessary congestion avoidance; UDP streams could be similarly affected.

Trends towards Mobile Ad-hoc Networks (MANETs) may also prove to be the next driver of packet reordering. Such networks, with frequent route re-computations and the absence of a central base station [Oliv02], could cause current TCP and UDP implementations to misinterpret packet reordering as congestion, resulting in severe performance degradation. MANETs are also characterised as having high BERs and, as a result, exhibit frequent packet retransmission behaviours. MANETs also suffer from long pauses in transmission during the frequent route re-computations and exhibit regular network re-partitioning, causing packets to be dropped. Furthermore, some routing algorithms, (e.g. TORA [Oliv02]), maintain multiple routes between source-

destination pairs and so include actual Layer Three routing parallelism. Mobility Management protocols such as Mobile-IP, and multi-homed Mobility protocols, such as NEMO, have been measured [Park08] [Tsan08] to cause severe packet reordering during handovers. Each of these mechanisms, in isolation, can result in extreme loss of sequence and, therefore, when presented in combination provide motivation to investigate wireless and packet reordering further.

The literature has focused on the separation of fixed networks from wireless networks[Kopp02], to seek gains in IP performance by optimising TCP for use in such error-prone environments. The wireless link is assumed to be the last hop of the connection, where most of the loss and delay occurs and where reordering would be most prevalent. It has been shown, though, that packet reordering can affect all parts of an IP network, and therefore it is important to consider the behaviour of TCP along the whole connection length. Obviously, the proposed modifications to TCP, do not take into account the effects of reordering on UDP traffic, which itself may be both time and sequence sensitive, depending on the type of data being carried.

6.1.2 The Effects of Reordering on Video

Little attention has been performed in the literature to investigate the effects of packet reordering on other non-TCP traffic. Although TCP is the predominant protocol used in the Internet today [Medi05], the trends in measurement studies of TCP would suggest that packet reordering is occurring on many other types of traffic too.

A single study on the effects of packet reordering on the subjective quality of broadband digital television has been carried out [Spir06]. This study involved several assessment sessions where human observers were shown the output of broadband digital television, to examine how users would perceive the audiovisual subjective quality. The study concluded that current set top box receivers suffer unacceptable quality when more than 0.12% of packets are reordered, on an IPTV network between a video server and a set-top-box. The study investigated the use of several different types of IPTV set top box, in order to gauge the performance of each box when a fixed amount of network reordering was applied using the NIST Net emulator. Test streams in these experiments

focussed on very high bit-rates, with video greater than 3 Mbit/sec, and audio at 192 kilobit/sec.

6.1.3 Video over UDP

UDP alone does not provide retransmission or congestion control and, with simple datagrams, it would be largely unaffected by packet reordering. However, UDP is now often used to deliver stream-oriented traffic and, with no in-built method of presenting feedback to the transmitting node, it must be assumed that packets will arrive on time and in order. Unfortunately when stream-based data, such as encoded video, is transmitted over UDP, the predictive coding strategies employed in techniques, such as MPEG-4, place a new set of constraints on traffic sequencing. For example, predictive coding introduces temporal dependencies into the video data that improve compression ratios, but can result in greater error propagation in the event of packet loss or late arrival.

The degree of error introduced by a lost MPEG encoded frame (or one which is so delayed by reordering as to be assumed to be lost), is governed by the specific temporal coding dependencies of that affected frame. Codecs such as MPEG-4 make use of *I*, *P* and *B* frames; each frame having its own characteristics of prediction dependence and delay constraints. *I* frames are required for the decoding of subsequent *P* and *B* frames. Therefore all frames must arrive by their 'playback time' at the client; additionally some frames may also have secondary deadlines. For example, an *I* frame that has been delayed and missed its individual 'playback time', will still be useful when decoding the subsequent *P* frames, that are based upon that *I* frame. Receiving a frame 'late' is, therefore, considerably more useful than not receiving it at all. Hence, the effects of frame reordering are significantly different from the effects of frame loss.

An improved streaming strategy has been suggested [Wee02] that attempts to reorder video frames before network transmission. The technique minimises perceptual errors by exploiting the fact that different late frames result in different degrees of video error. Work on this and other 'Rate-Distortion' optimisation techniques are based on the principle that each frame is transmitted separately. In modern transmission systems, i.e.

those supporting high bit-rate video and many frames-per-second, it is much more likely that multiple frames will be transmitted in each packet, thereby requiring transmission scheduling on a per-packet and not a per-frame basis. The Microsoft Advanced Systems Format (ASF) [Micr08b] is the most popular video streaming format used on the Internet today. ASF video packets are of fixed length, and can contain multiple video frames within each packet. When transmitted over proprietary protocols such as Microsoft Media Services (MMS), which further package several ASF video packets per IP MMS packet, it becomes almost impossible to design a scheduling algorithm that can effectively combat packet reordering.

6.2 Experimental Methodology

The aim of this investigation is to experimentally measure the behaviour of low bit rate Video over UDP traffic during various degrees of packet reordering in an end-to-end connection. This will allow for both a better understanding of the mechanisms employed in video streaming and for better overall QoS prediction.

An experimental set-up was devised to transmit video traffic as UDP IP packets across an Ethernet network encoded using an ISO-compliant MPEG-4 codec. Clearly, an Ethernet based system will not exhibit the same behaviour as a wireless network but instead offers a controlled environment where the effects of reordering can be measured, and then correlated with the behaviour expected, without the additional complications associated with wireless networks. A software tool was then developed to selectively add delay to packets to simulate reordering, and the results obtained from these experiments are discussed.

6.2.1 Microsoft Windows Media

The Microsoft Windows Media (WM) [Micr08b] suite of tools is designed for the authoring and distribution of multimedia content over the Internet; it provides an excellent controlled testbed for the experimentation of streaming video delivery. The purpose of this experiment is to inflict varying delays upon video packets streamed over

UDP/IP between a client and server, while monitoring the perceptual effects of reordering on the video display and using an instrumented receiver to correlate quantitative results on packet arrivals.

6.2.2 Video Traffic Generation

A typical videoconference scenario was simulated via a USB desktop camera. Using Microsoft WM Encoder v7, a simple profile with single unicast video stream was set-up to encode packets using Microsoft's ISO-compliant MPEG-4 codec at 15 frames per second. QCIF resolution, using average audience bit rates of 300kbps, was employed. The encoded video is passed to the Windows 2000 server running WM Services 4.1, which was configured for MMS streaming over UDP as illustrated in Figure 79

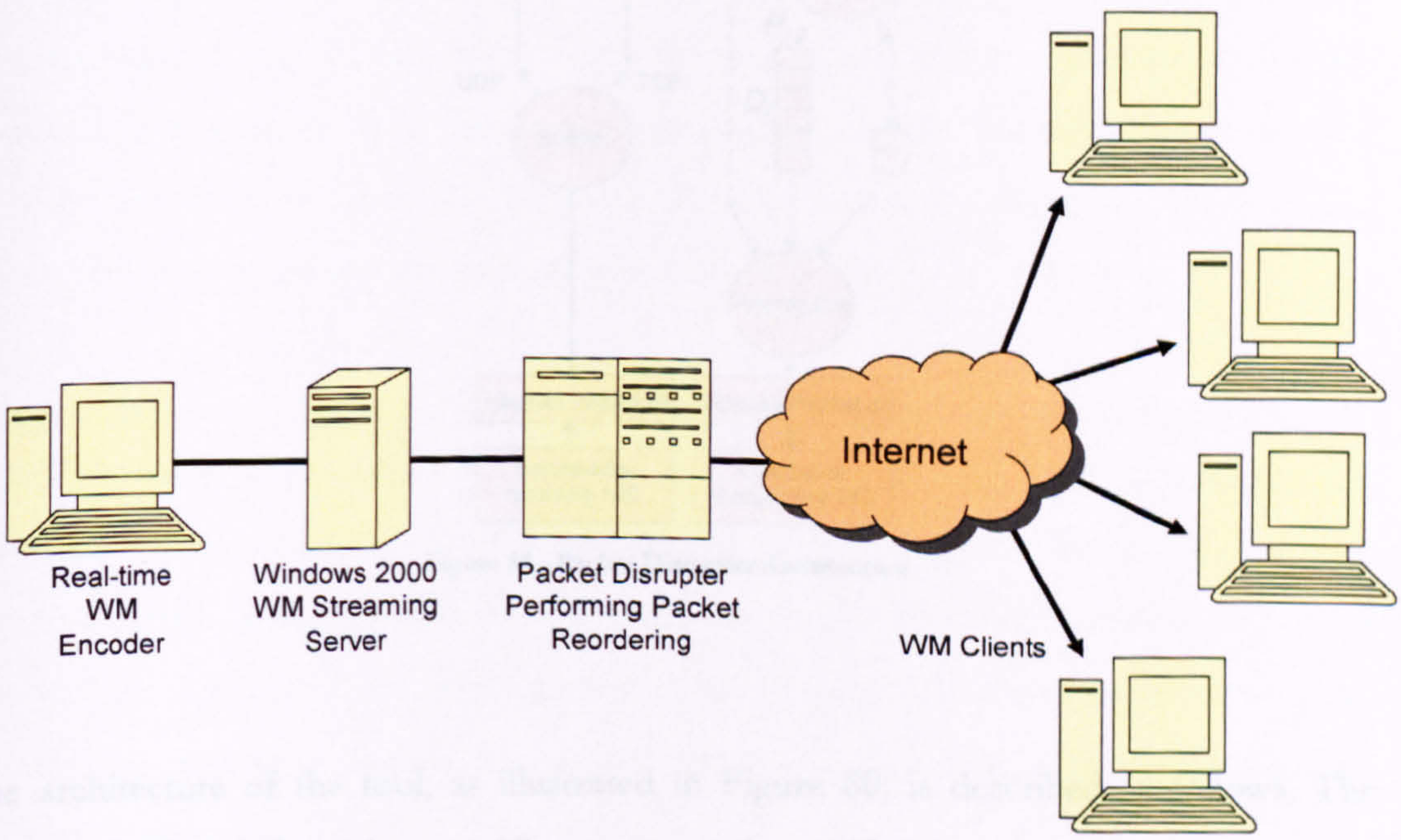

Figure 79 – Video Reordering Experimental Testbed

6.2.3 Reordering of Video Packets

The Packet Disrupter tool is a C++ application developed for the application of noise on video packets, transmitted across an IP network. Interfacing through two network cards, the tool acts as a software router performing Network Address Translation and allows the addition of noise and errors, as well as changing scheduling, queuing and dropping behaviours.

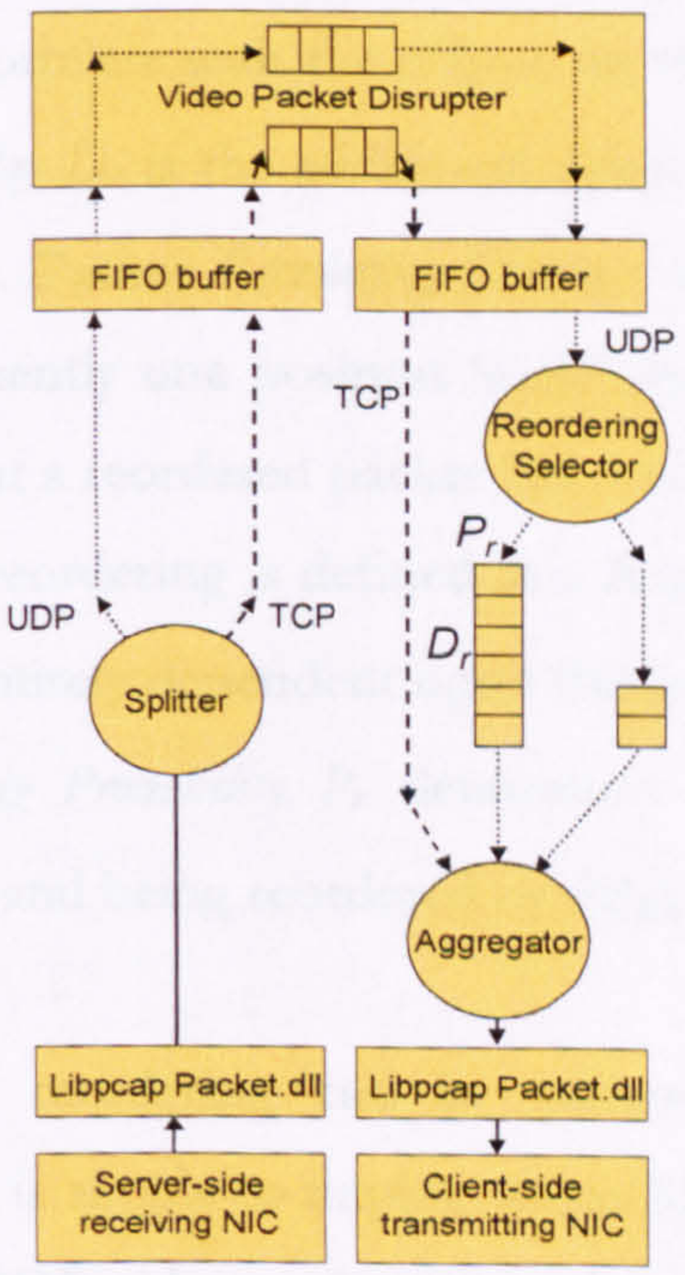

Figure 80 - Packet Disrupter Architecture

The architecture of the tool, as illustrated in Figure 80, is described as follows. The Disrupter was configured as an IP gateway and provided the default route from the LAN segment containing the Encoder and Streaming Server. Frames containing IP packets with a destination IP address of one of the WM Clients were copied into the application using the Libpcap packet capture driver. Packets were initially filtered according to their protocol type, with UDP packets buffered while all others were passed immediately up to the application level. In all experiments, video was streamed

over UDP and it was therefore assumed that all other packets were control protocols, required for the start-up and maintenance of the streaming session. The Packet Disrupter application monitored the number of packets processed, maintained a register of inter-packet spacing and monitored the overall performance of the tool to ensure that all packets were serviced in a timely fashion without the risk of dropping. The application provided a post-processing logging functionality of all packets' arrival and departure times, which provided a means to measure the degree of reordering that had taken place.

During experimentation, three metrics are used to characterise the extent of reordering. The aim is that these should correlate with the effects on video quality perceived by the end-user. Packet *Reordering Delay* D_r is the additional delay applied to a packet that has undergone a reordering event. Packet *Reordering Distance* d_r, is the number of packets (including the very first, apparently one position 'early' packet, and also the reordered multi-position 'late' packet) that a reordered packet has traversed, after it has undergone a reordering event. Note that reordering is defined as a function of time and not packet positions, and therefore d_r is entirely dependent upon the instantaneous bit-rate of video at that time. Packet *Reordering Probability* P_r determines the likelihood of a packet undergoing a reordering event and being reordered by delay D_r.

Using these metrics, packet reordering can be reduced to a simple dual-queue scheduling architecture, which is simple to implement within the Packet Disrupter tool. When leaving the Disrupter, UDP video packets pass through the 'Reordering Selector' which randomly selects packets around a uniform distribution with probability P_r; those selected for reordering pass down a slow queue of length D_r seconds, therefore being reordered a distance d_r packets. Queue outputs are aggregated and then transmitted on the client-side network card.

6.2.4 Instrumentation of Receiver

To allow accurate quantitative analysis of video received at the decoder, a client-side estimator of WM QoS was required; the aim was to compare the measured quantitative results of packet reordering with the qualitative results displayed after decoding.

An instrumented version of WM Player v7 was developed, as illustrated in Figure 81, using the WM Software Development Kit[Micr08b]. The additional metrics collected from the player included the current bandwidth, the number of times buffering occurred during playback, the current video frame rate, the total number of frames skipped during playback, the number of packets lost and the number of recovered packets.

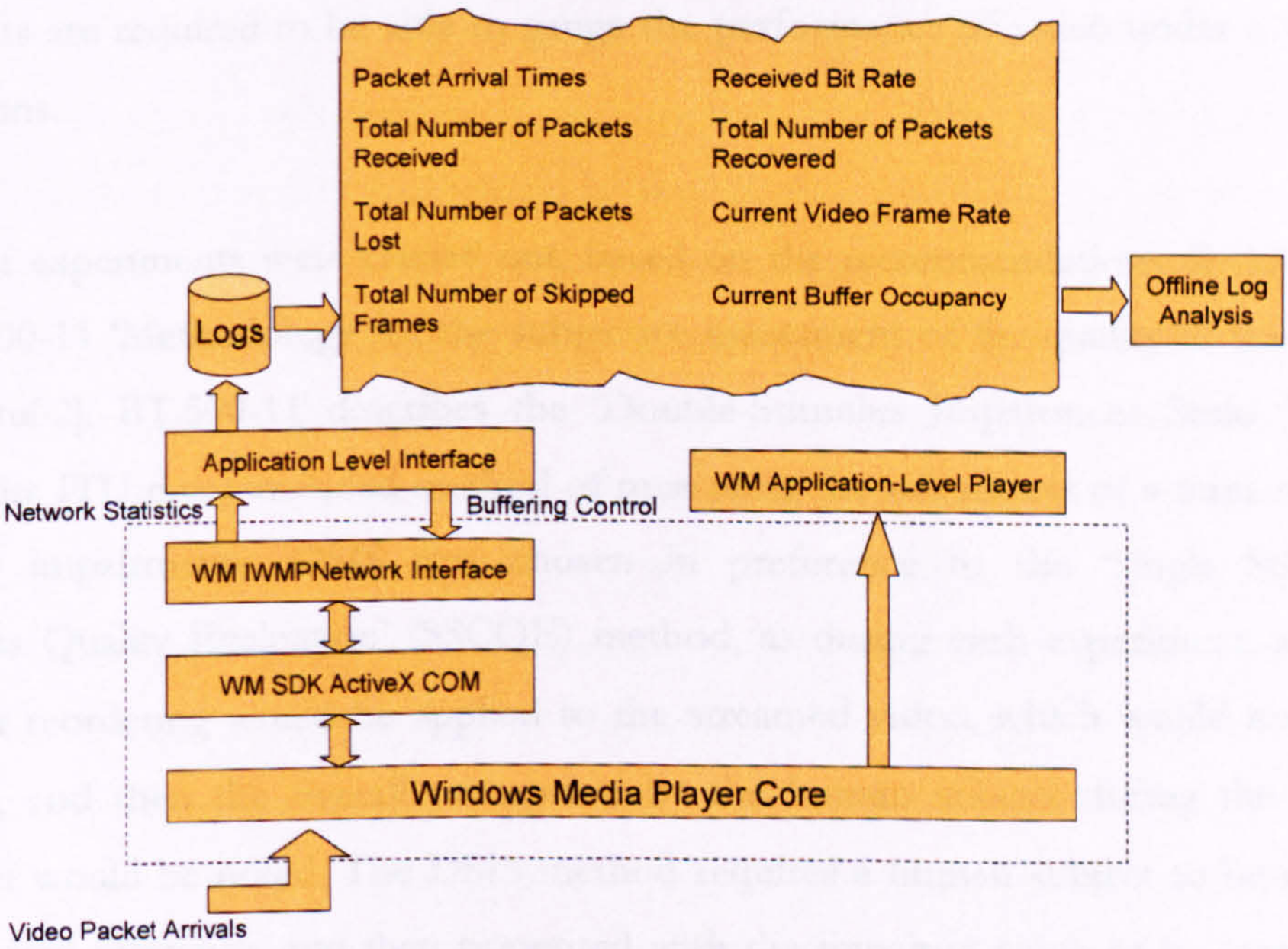

Figure 81 - WM Player Instrumentation

In addition, logging was also enabled at the WM Server allowing for measurements of the amount of time that the client spent rendering the stream, the average bandwidth of the connection, the number of bytes received by the client, the number of packets that were not delivered to the client and the number of times the client buffered the stream.

6.3 Results and Perceptual Quality

The purpose of these experiments was to measure the effects of packet reordering as perceived by an end-user. The perceptual performance of digital video systems can be assessed in either an Objective or Subjective basis. Objective methods attempt to measure video quality by mathematically comparing a reference video signal, with a second video signal which has been transmitted over an impaired transmission network[Spir06]. Objective methods of assessing video are still in their infancy[Vqeg08], as they require a reference signal to perform computation of results, and require initial 'training' using a subjective method in order to correlate the mathematical assessment with a human's perceived performance. It is accepted [Itu02] that it is impossible to fully characterise a system using entirely Objective means, and therefore Subjective quality assessment is required to supplement all Objective results. A cross-layer application-aware metric of video packet reordering is developed in Section 6.3.2. To be able to correlate this Objective method of video quality measurement, a series of Subjective experiments are required to be able to gauge the performance of video under a variety of conditions.

A series of experiments were carried out, based on the recommendations discussed in ITU BT.500-11 'Methodology for the subjective assessment of the quality of television pictures'[Itu02]. BT.500-11 describes the 'Double-Stimulus Impairment Scale (DSIS) method', the ITU recommended method of measuring the robustness of a transmission system to impairments. DSIS was chosen in preference to the 'Single Stimulus Continuous Quality Evaluation' (SSCQE) method, as during each experiment, a fixed amount of reordering would be applied to the streamed video, which would not vary over time, and then the overall perception by the human subject during the entire experiment would be noted. The DSIS method requires a human subject to be shown an unimpaired reference, and then presented with the impaired video to be measured. Following this, the subject is asked to vote on the second, based on the reference. The impaired video to be measured is randomly displayed with various impairments, interspersed with non-impaired copies of the reference video at frequent points.

The DSIS method recommends grading video on a five-grade measurement scale.

Experiments were therefore performed, and each subject was asked to score the quality of the entire 120 second video clip based on the descriptions in Table 17.

Grading Scale	Description
5	Errors Imperceptible
4	Errors Perceptible, but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying
0	Video imperceptible

Table 17 - Subjective Grading Descriptions

A 120 second 'head-and-shoulders' typical videoconferencing scenario was recorded using Windows Media Encoder. The same video recording was used in every experiment for consistency of results, and was placed on the Windows Media server for download during each assessment session.

Five human observers volunteered to take part in the series of experiments. The observers were not experts in technology or multimedia, and varied in age from 19 to 79. The assessment sessions were held in a room conforming to the 'Home Viewing Environment Specifications' and test procedure documented in ITU BT.500-11 [Itu02]. Before each assessment began, each observer was given introductory training about the purpose of the assessment, and the types of video impairments that they may expect to observe, such as video blockiness or colour artefacts, excessive pauses or freezing of the screen, and complete 'blacking-out' of the screen. The observers were trained in the 0 to 5 scoring system, and understood that the score chosen at the end of each 120 second recording, was to describe the user's experience throughout the duration of the entire recording. Before commencing the assessment, a recording was shown with no reordering induced, a recording with 10% 1 second reordering was shown, and finally a recording shown with 25% 2 second reordering, thus illustrating to each observer, the range of qualities that they could expect to observe throughout the experiments.

The assessments lasted approximately 2.5 hours, during which, 60 experiments were

shown to each of the five human observers. The 60 experiments introduced reordering at rates P_r = [1%, 10%, 25%], with D_r varied between 0 and 2 seconds in 0.1 second steps. These were conducted in a random order for each observer. Earlier experimentation had shown that for a default WM buffer of 5 seconds, the range over which reordering could be applied was between 0 and 2 seconds. The Reordering Probabilities of 1%, 10% and 25% were chosen as they emulate a wide range of results, thus providing evaluation of video under a variety of conditions. Figure 82 illustrates the average score of the 5 observers obtained from each experiment, to attempt to describe the visual disturbance caused by packet reordering.

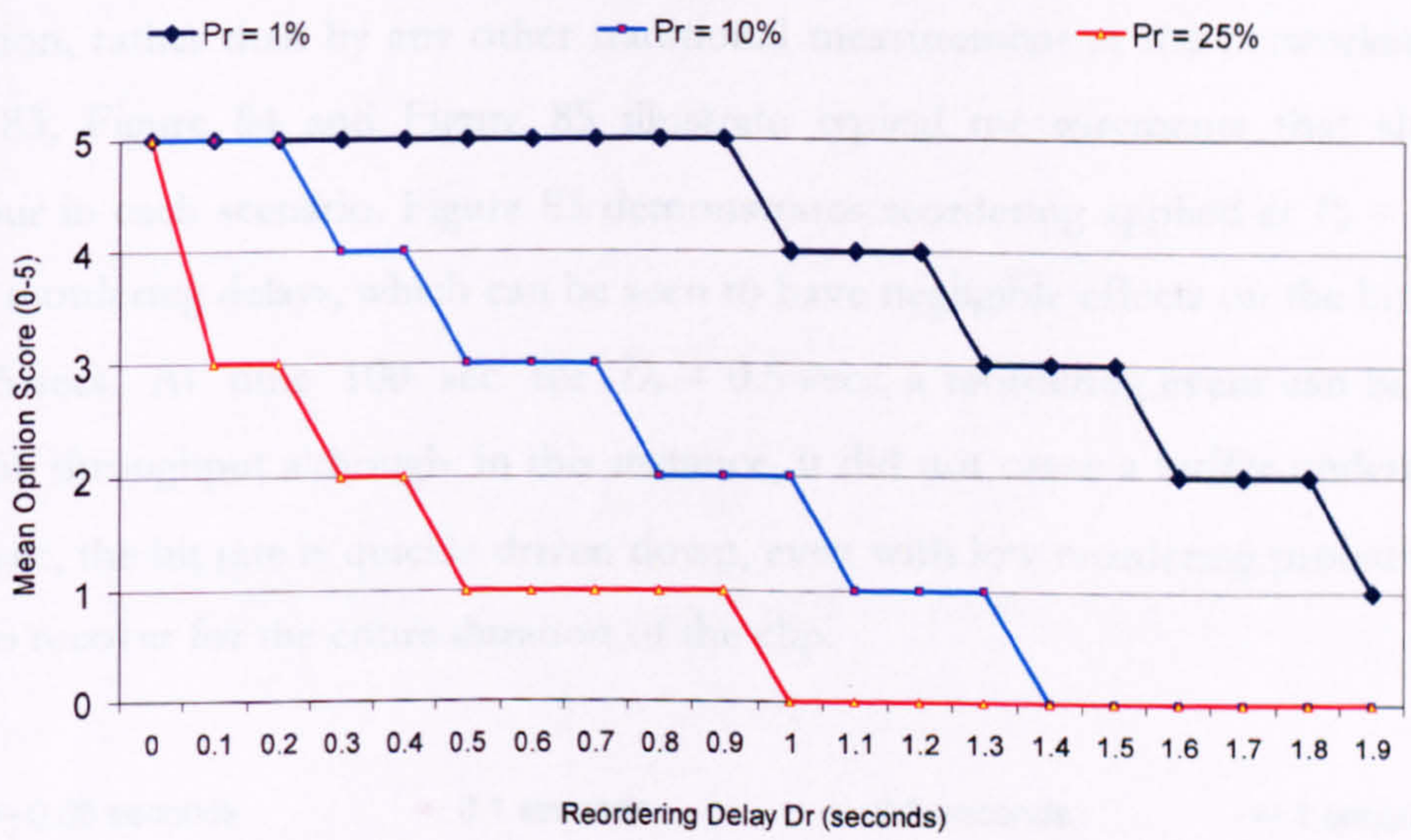

Figure 1.

Figure 82 – Average Mean Opinion Scores

Initial results indicated that MMS was surprisingly tolerant to varying delays, contrasting with previous measurements of TCP undergoing similar reordering events. This was due, in part, to the large pre-roll buffer that was generated at the client player before streaming commenced which, by default, was set to 5 seconds of real-time audio/video. During periods when the server maintained its constant encoding bit-rate and did not attempt to 'thin' the stream, visual output remained stable throughout the duration of a clip. Perceived quality would not alter significantly, with little loss of sharpness, even during scenes of high motion, and no artefacts appearing as a result of corrupted visual data. This quality was maintained in all experiments up to a threshold level, where 'frame freezing' would occur due to an underrun at the client input buffer and playback

would pause on the last correctly displayed frame. Playback would not resume until the WM client had successfully re-filled the entire contents of the buffer.

6.3.1 Packet Arrival Bit Rates

The instantaneous bit delivery rates were measured over 120-second clips at the 300kbps target encoding rate and varying reordering degrees applied. It should be noted that this metric is the instantaneous bit-rate as reported by the WM API, and therefore is an application-layer measurement as reported by the player itself. This cross-layer metric allows characterisation of the effects of packet reordering as observed by an application, rather than by any other traditional measurement in the networking stack. Figure 83, Figure 84 and Figure 85 illustrate typical measurements that show the behaviour in each scenario. Figure 83 demonstrates reordering applied at $P_r = 1\%$, for varying reordering delays, which can be seen to have negligible effects on the bit rate for $D_r < 0.5$ secs. At time 100 sec for $D_r = 0.5$ secs, a reordering event can be seen to affect the throughput although, in this instance, it did not cause a buffer underrun. For $D_r = 1$ sec, the bit rate is quickly driven down, even with low reordering probability and it fails to recover for the entire duration of the clip.

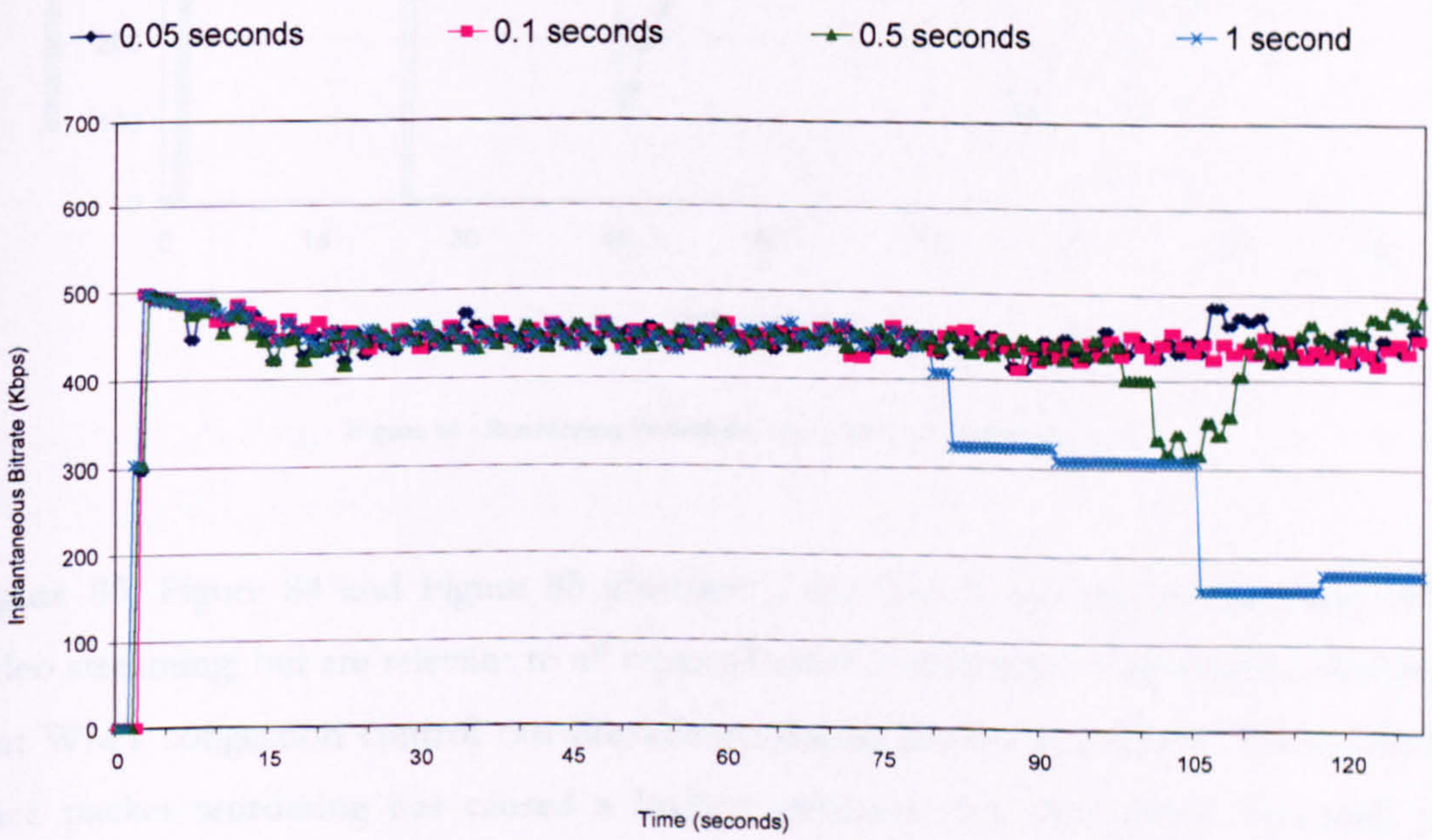

Figure 83 – Reordering Probability $P_R = 1\%$, for varying D_r

Figure 84 illustrates the behaviour of WM video under degrees of heavy reordering. For example, at time 15 sec where $D_r = 1$ sec the bit rate can be seen to decay rapidly; this was due to a severe reordering event, which then resulted in buffer underrun and, as a consequence, freeze frame was observed at the player. At time 24 sec, the player initiates re-filling of the buffer; this request is achieved by the server sending a burst of high bit rate UDP video packets – peaking at 700kbps – considerably higher than the standard 300kbps stream. Upon filling the 5 second buffer, playing was resumed at time 30 sec, and continued successfully until another significant reordering event at 95 sec. Figure 85 demonstrates this behaviour further and illustrates multiple significant reordering events. Each event had a severe impact on buffer underruns, followed by a period of recovery where the delivery of high bit rate traffic occurs in an attempt to re-fill the buffer as quickly as possible.

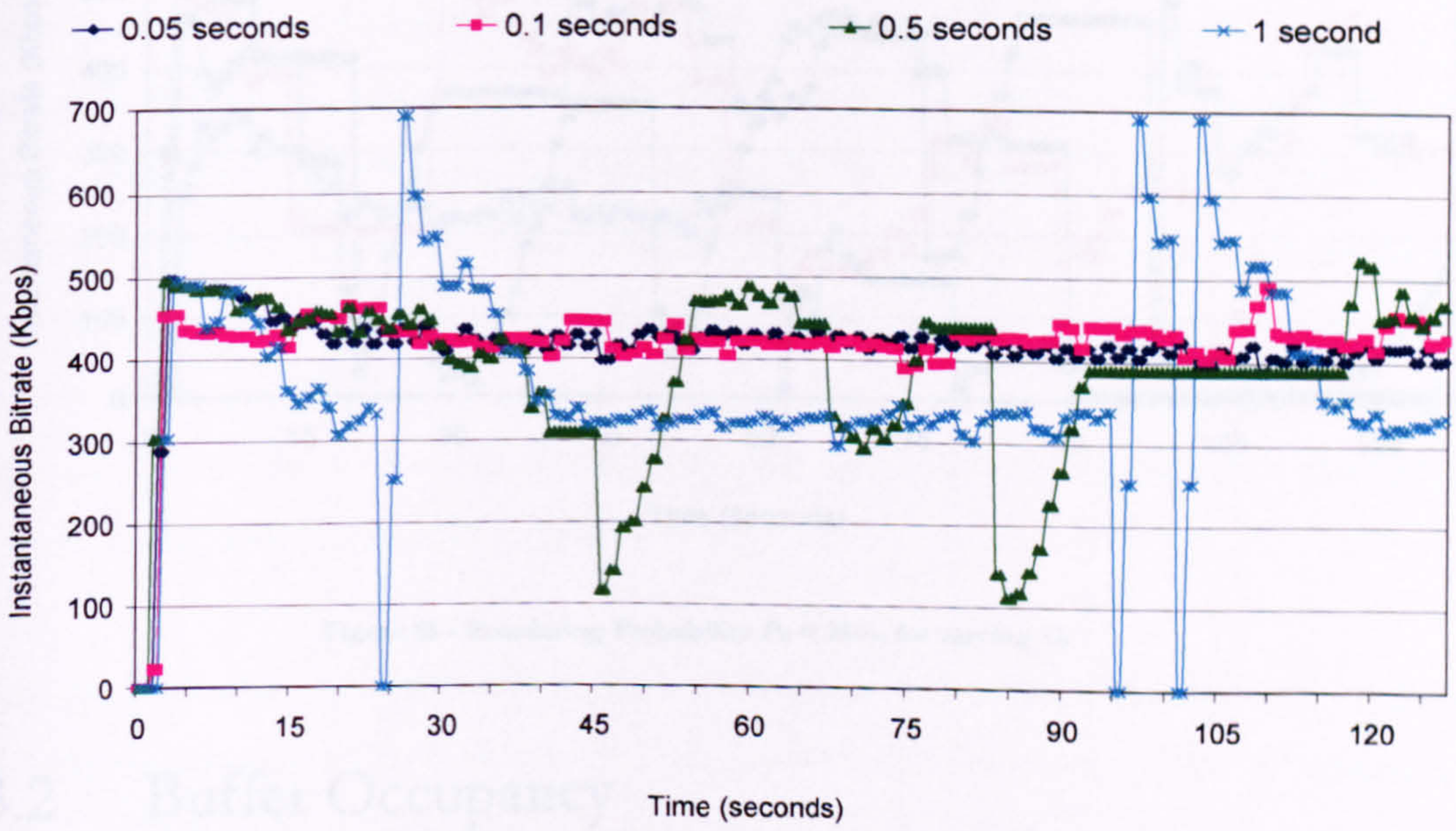

Figure 84 - Reordering Probability P_R = 10%, for varying D_r

Figure 83, Figure 84 and Figure 85 illustrate a number of key points regarding WM video streaming, but are relevant to all types of video transmission. The figures illustrate that WM's congestion control can breakdown during packet reordering. For example, once packet reordering has caused a buffer underrun, the WM Player responds by requesting re-transmissions through N-ACK packets that instruct the WM server to launch a large amount of traffic into the network in an attempt to fill the 5 second

player buffer as quickly as possible. This resultant high bit-rate and large number of packets, also undergo significant reordering which is often exacerbated due to its greater share of the bandwidth consumed when compared to other applications on the network. The subsequent storm of traffic also increases the burstiness of the stream and is grossly unfair to competing packet flows. Furthermore, this traffic storm on a heavily loaded network may also increase the reordering probability. Such a congestion control mechanism could prove costly in a wireless environment, where both high transmission tariffs and power consumptions must be taken into account.

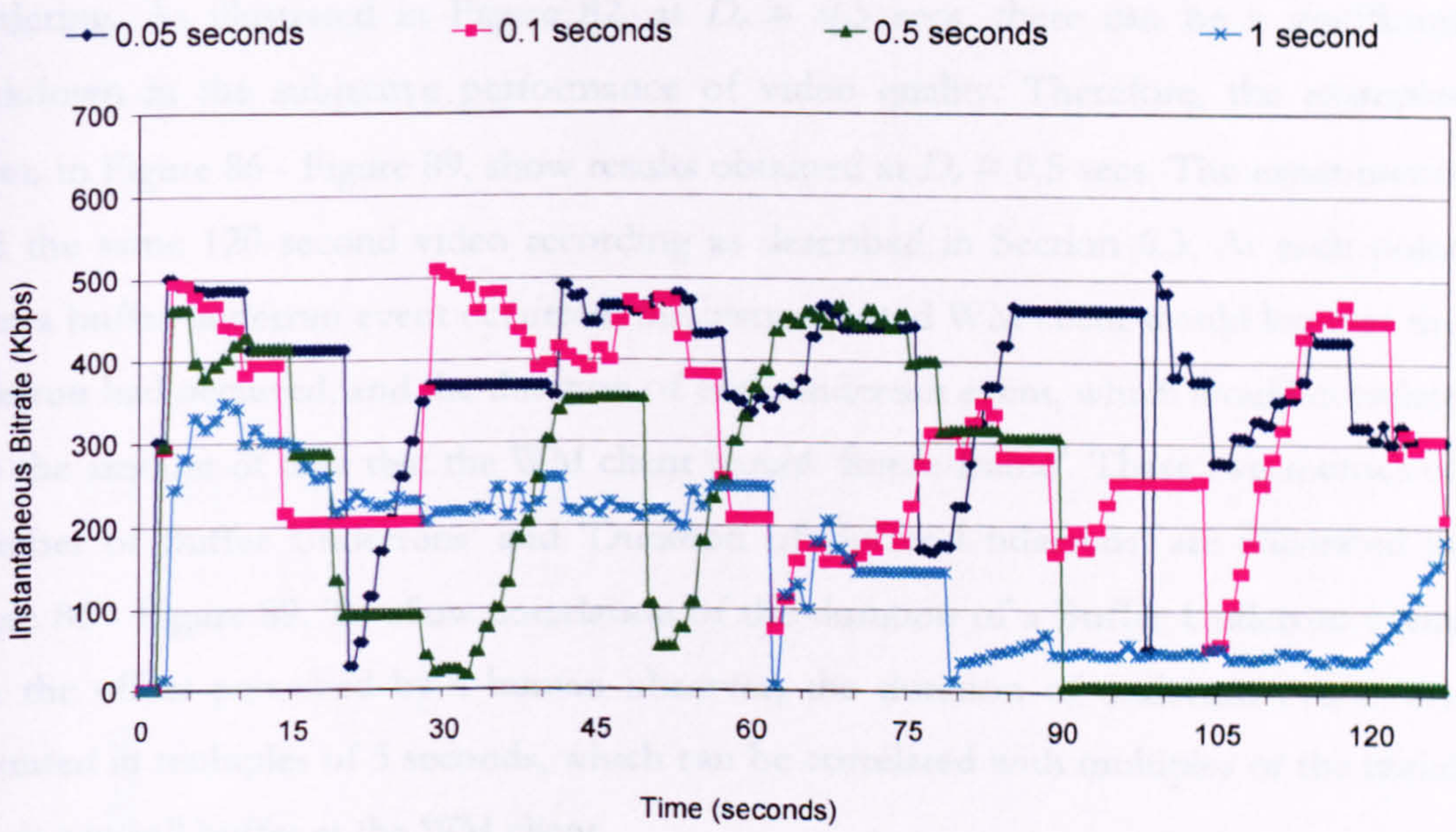

Figure 85 - Reordering Probability P_R = 25%, for varying D_r

6.3.2 Buffer Occupancy

Buffer underrun has been previously measured as the most important factor when determining the performance of WM across an end-to-end network [Dala03]. Buffering is important as it allows an inherently bursty medium, such as video, to be transmitted over a communications channel at a constant rate. With large enough buffers, it would be possible to re-sequence any packet (providing it has arrived before the playback time), but large buffers take a significant amount of time to initially fill, which is perceived badly by the user. Moreover, large buffers could be costly to implement on low powered portable devices. Therefore, it is important to decide on an appropriate

buffer size that will act effectively to smooth traffic in the event of loss or reordering, but will also initially fill quickly to provide an effective user experience.

The client was further adapted to measure the occupancy rate of the player's buffer. This allowed for monitoring of any period where packet arrival was disrupted to the extent that a freeze frame resulted, so that the time taken to re-fill the buffer to 100% and resume normal playback could be measured.

A total of 40 experiments were completed for each percentage of 1%, 10% and 25% reordering. As illustrated in Figure 82, at D_r = 0.5 secs, there can be a significant breakdown in the subjective performance of video quality. Therefore, the examples shown in Figure 86 - Figure 89, show results obtained at D_r = 0.5 secs. The experiments used the same 120 second video recording as described in Section 6.3. At each point when a buffer underrun event occurred, the instrumented WM client would log that the underrun had occurred, and the duration of each underrun event, which would correlate with the amount of time that the WM client would 'freeze-frame'. These two metrics of 'Number of Buffer Underruns' and 'Duration of Buffer Underruns' are illustrated in Figure 86 - Figure 89. To allow correlation of the duration of a Buffer Underrun event with the effect perceived by a human observer, the duration of underrun events are illustrated in multiples of 5 seconds, which can be correlated with multiples of the initial default pre-roll buffer at the WM client.

Figure 86 illustrates the probability density function of buffer underruns that a user could expect to experience during a clip of 120 seconds duration. Figure 87 plots the durations of these buffer underrun events and of the durations of the resulting freeze frames. As can be seen, for P_r = 1%, all streams will encounter at least one recorded underrun event during the initial pre-roll buffering before streaming commences. After this for P_r = 1%, it is over 70% likely that no further buffer underruns will occur at all. Conversely, at 25% reordering, at least 2 underruns will occur during the same 120 sec period.

As can be seen in Figure 87, the majority of underrun events last approximately 5 seconds – equivalent to the default buffer size in WM Player. Buffer underruns lasting longer than 5 seconds indicate severe congestion.

Using the SDK, it was possible to alter the amount of buffering time apportioned by the WM client. Figure 88 and Figure 89 show the results of doubling the buffer to 10 seconds and the resulting effects. As can be seen, the buffer underrun is shifted towards the middle. Hence the number of smaller pauses has been reduced, but more importantly, the probability of excessively long pauses is also reduced and the behaviour of the client is more predictable. The improvement is, in part, due to the extra time available for packet re-sequencing.

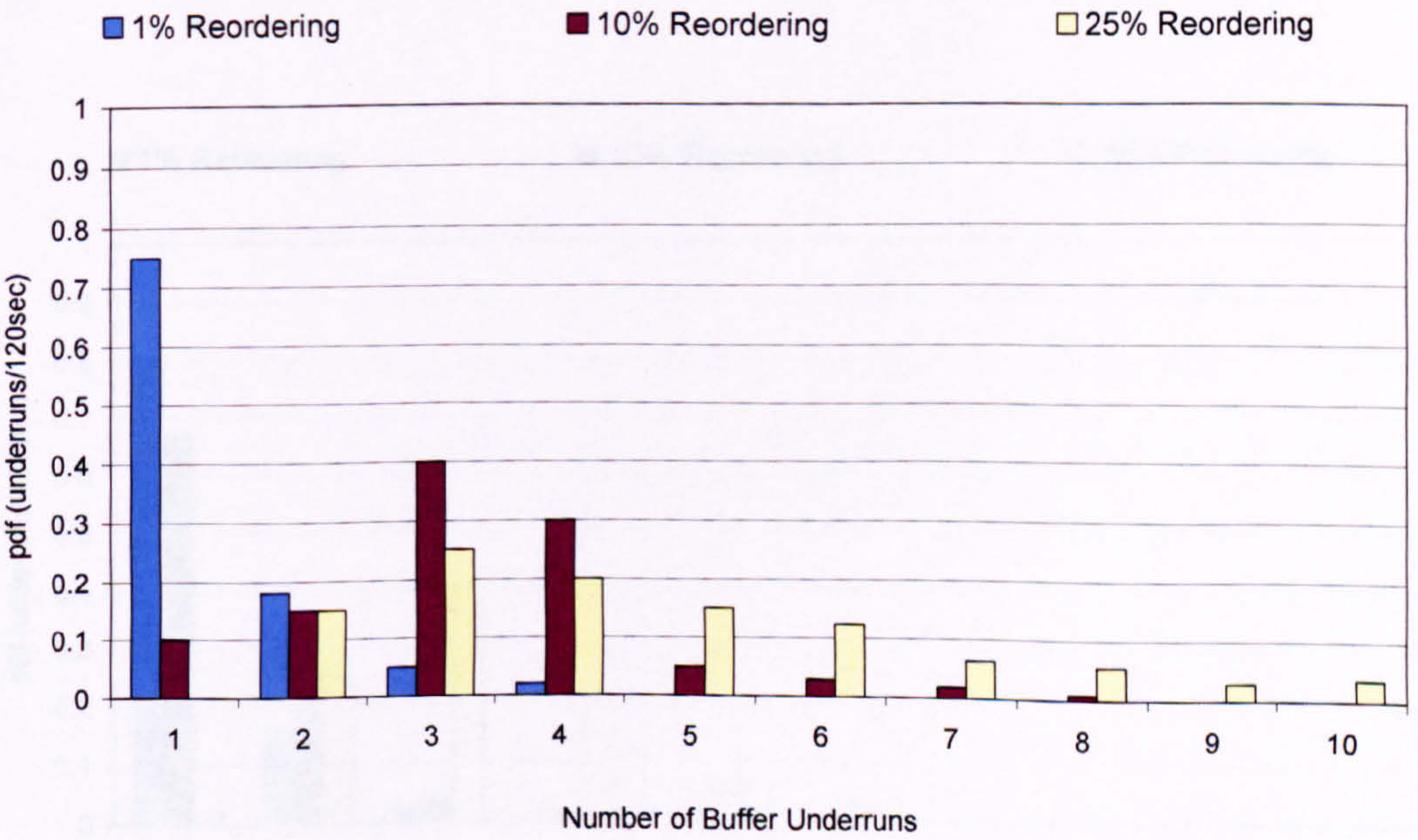

Figure 86 - pdf Under-Run Number

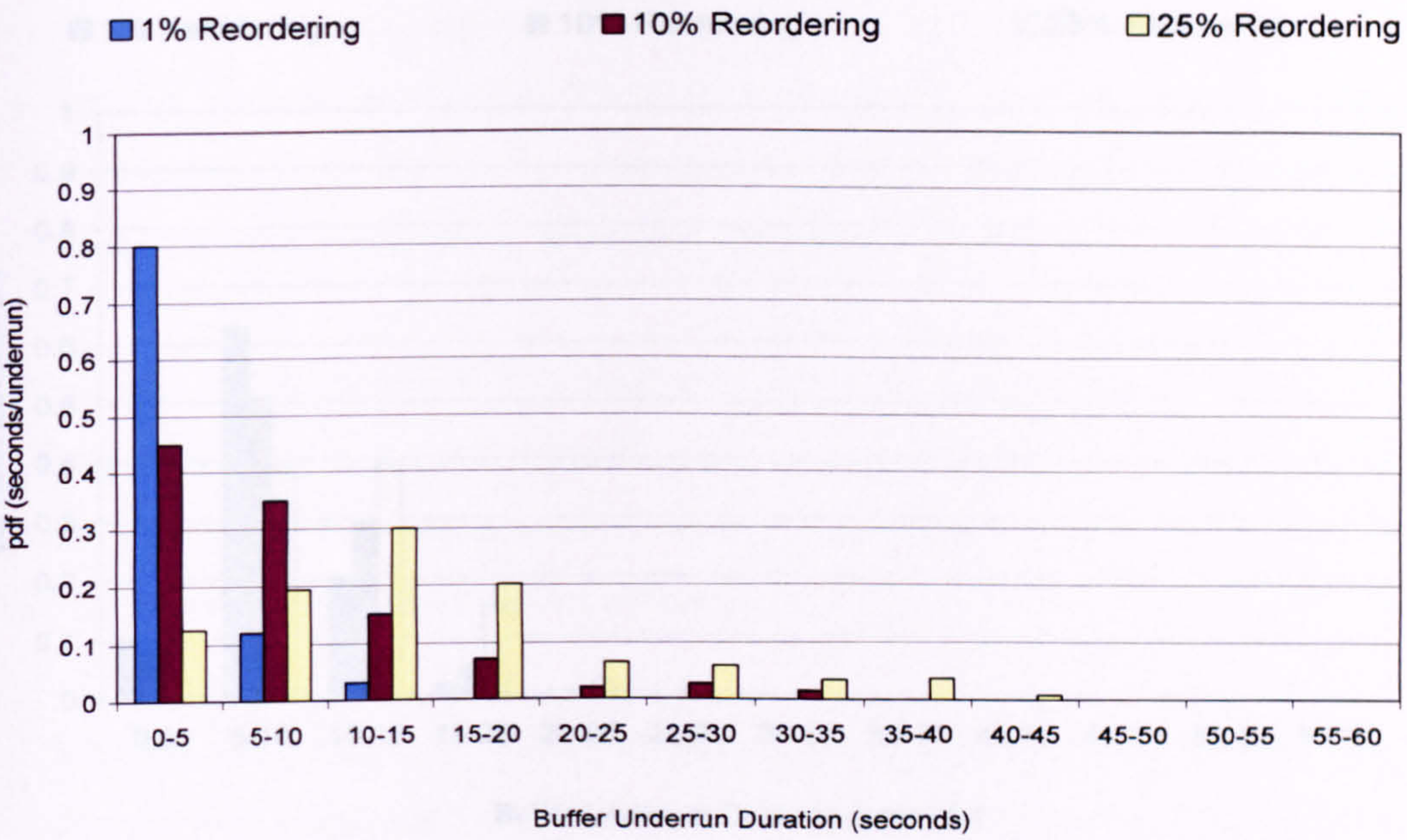

Figure 87 – pdf Under-Run Time

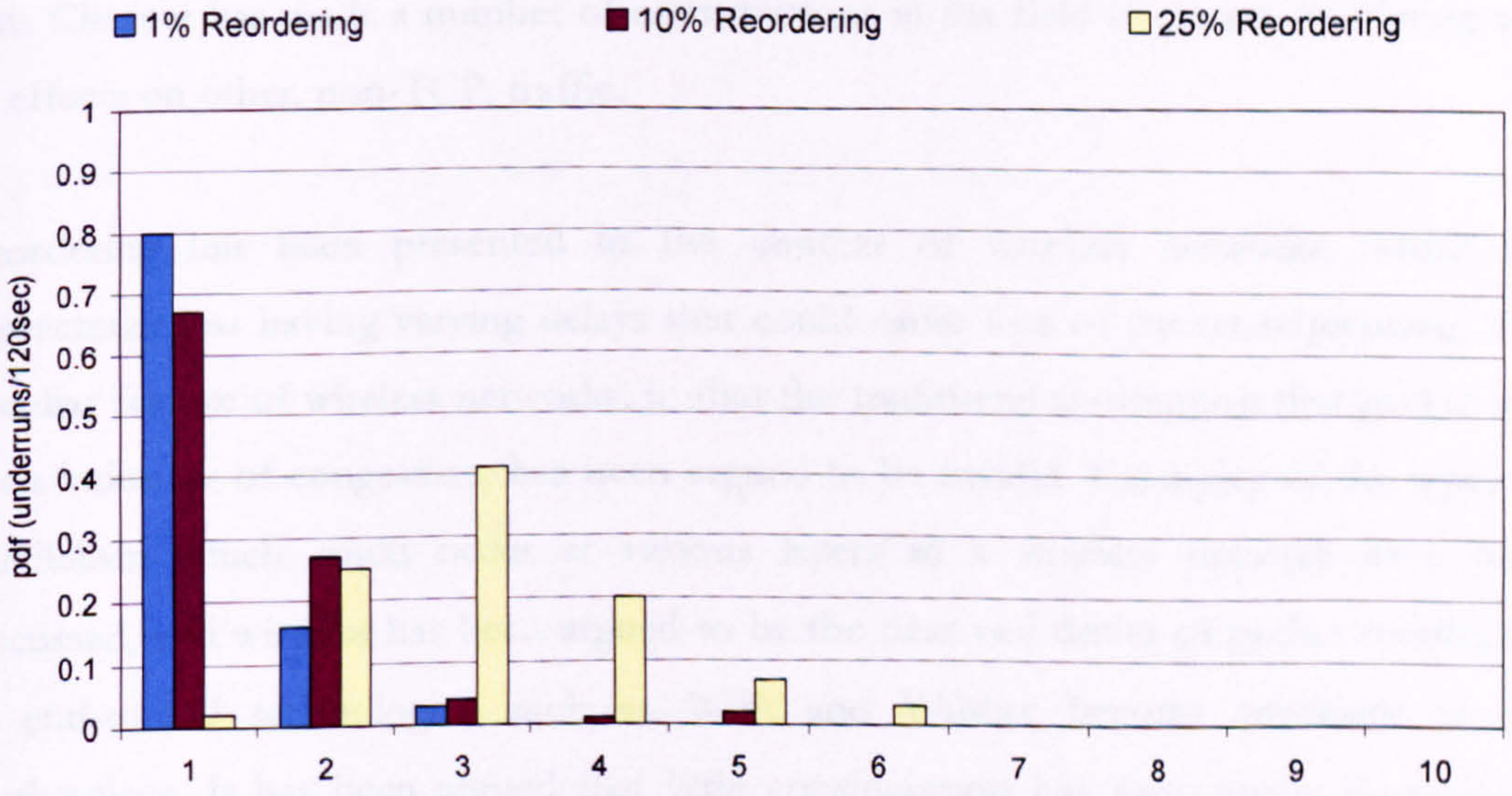

Figure 88 – pdf Under-Run Number

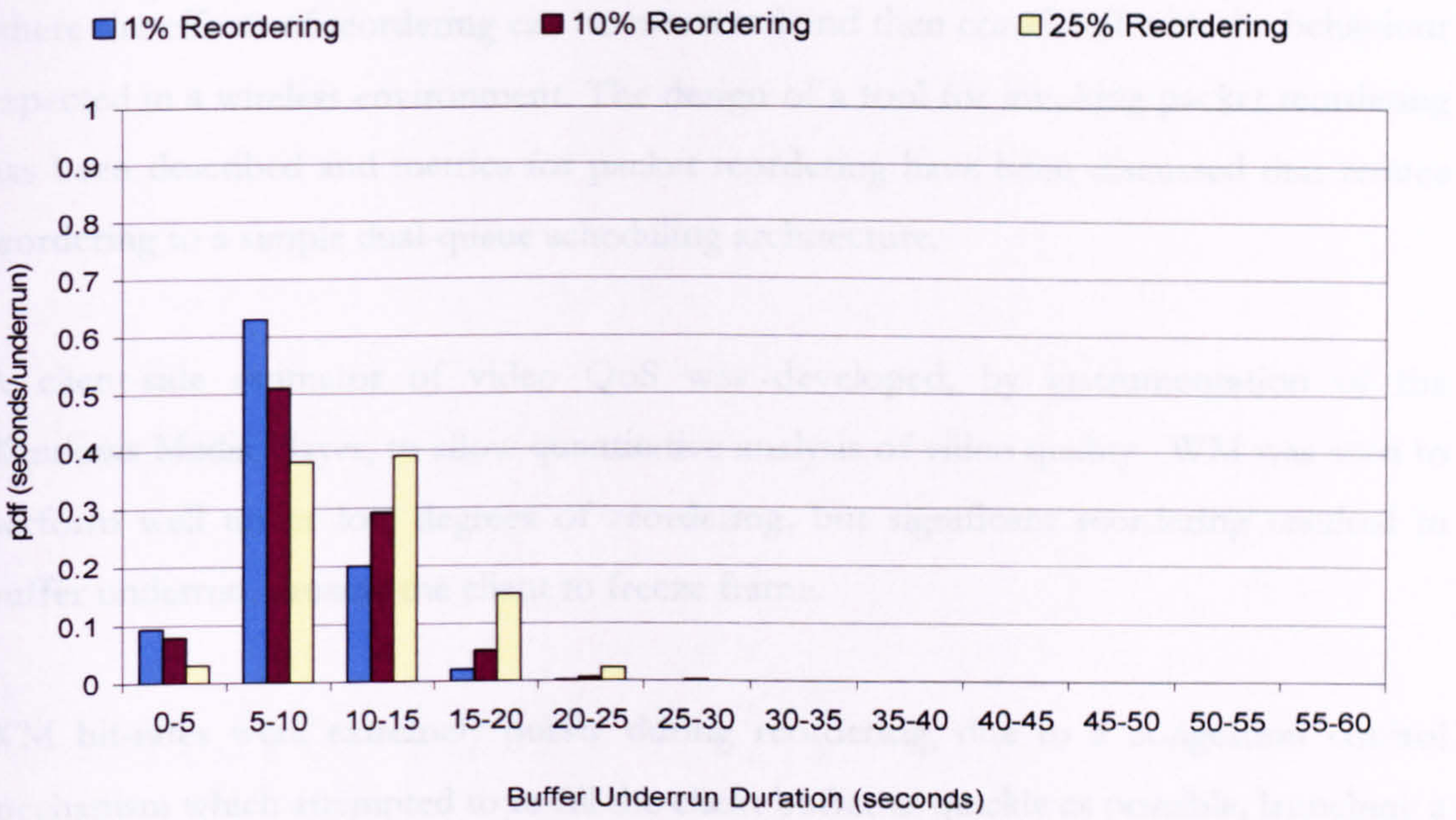

Figure 89 - pdf Under-Run Time

6.4 Conclusions

This Chapter has made a number of contributions in the field of packet reordering and its effects on other, non-TCP, traffic.

Reordering has been presented in the context of wireless networks, which are characterised as having varying delays that could cause loss of packet sequencing. The peculiar feature of wireless networks, in that the traditional assumption that packet loss is an indicator of congestion, has been argued to be invalid. Examples of the types of parallelism which could occur at various layers in a wireless network have been discussed, and wireless has been argued to be the next real driver of packet reordering, as end-to-end technologies such as WiFi and WiMax become prevalent in the marketplace. It has been argued that little consideration has been given, thus far, to other time-critical types of network traffic, such as Video over UDP. Indeed, the effects of packet reordering are even more difficult to predict on such traffic, due to the temporal inter-packet dependencies introduced by the MPEG video encoding structure.

An experimental testbed has been presented, that offers a controlled environment

where the effects of reordering can be measured and then correlated with the behaviour expected in a wireless environment. The design of a tool for invoking packet reordering has been described and metrics for packet reordering have been discussed that reduce reordering to a simple dual-queue scheduling architecture.

A client-side estimator of video QoS was developed, by instrumentation of the Windows Media Player, to allow quantitative analysis of video quality. WM was seen to perform well under low degrees of reordering, but significant reordering resulted in buffer underrun, causing the client to freeze frame.

WM bit-rates were extremely bursty during reordering, due to a congestion control mechanism which attempted to re-fill the client buffer as quickly as possible, launching a packet storm that was grossly unfair to competing packet flows. Buffer occupancy measurements were made to describe the number of underruns that a user would expect to experience during reordering, thereby providing a more advanced metric of packet reordering, specific to this particular traffic. Adjustment of the buffer size was found to enhance the client's ability to perform under significant reordering, by controlling the number of excessively long pauses a user will experience.

It is clear that the diversity of traffic types prevalent in the Internet today, will require a range of advanced metrics and measurement methodologies, in order to correlate measurements made at various layers in the OSI, with the behaviours experienced by the end user. This thesis has illustrated that, although several attempts have been made to measure TCP during reordering, a metric is only truly useful if it is designed for a specific application and specific scenario. The method described in this Chapter, where the WM SDK was modified in order to expose Buffer Occupancy as a metric of reordering, is an example of the type of metric required to truly measure the effects of packet reordering. Clearly as packet reordering becomes more prevalent, and drivers such as wireless begin to make an impact, there will be clear motivation for network operators and application developers to build such measurement tools.

Chapter 7

Conclusions and Future Work

7.1 Introduction

This thesis has argued that Packet Reordering in IP networks is an increasingly common phenomenon, which will require a range of sophisticated measurement methodologies and metrics in order to characterise the performance impact on various network traffic types in the future Internet. Packet Reordering has been argued to be the result of increasing parallelism within networks. The trend towards future end-to-end wireless links, has further been identified as a key driver of the reordering observable in a path.

This thesis has addressed the issue of measuring the impact of packet reordering on Internet traffic, by proposing a number of measurement methodologies and metrics. The effects of packet reordering as proposed by Bennett have been discussed. A two-point passive measurement technique has then been developed, which improves on previous methods by allowing the lightweight classification of the cause of each reordering-induced packet retransmission. This technique has been applied to a large-scale testbed measurement study of packet reordering, which has indicated that TCP is tolerant to large percentages of reordered packets, providing that the delay of these packets is maintained below a specific threshold relative to Round-Trip-Time. Packet reordering metrics, which report percentage reordered packets, have been shown to be poorly correlated to the measured effects of reordering on TCP. Traditional assumptions that TCP packet reordering is an intuitively negative phenomenon have been questioned. Empirical measurements have shown that, in specific scenarios, packet reordering can actually increase the overall throughput of a flow.

A classification taxonomy of active and passive packet reordering measurement techniques has been presented. This has identified the key limitations of each technique, and the range of disparate measurements on the amount of reordering occurring in the Internet today. A mid-point passive Measurement Technique and Visualisation Metric of TCP packet reordering has been proposed, designed to classify out-of-sequence packets for many thousands of concurrent TCP flows. The proposed technique is lightweight to implement and does not require symmetric TCP connections to operate, thereby allowing an improved measure of TCP Goodput and simpler classification of the cause of each out-of-sequence packet. The proposed Visualisation Metric has been shown to offer an improved method of characterising packet reordering, by being simple to compute, and by indicating the packet reordering performance throughout the lifetime of a TCP flow.

Finally, this thesis has argued that future packet reordering metrics must correlate reordering observed at the network layer with the resulting impacts observed at the application layer. An example of an application-specific metric is developed for MPEG4 video over UDP traffic, and this metric is used to describe the effects of packet reordering on streamed video traffic. It has been argued, by this thesis, that this is the

type of measurement technique which will be required in the future, in order to describe the complex cross-layer effects of packet reordering on future complex traffic types.

This chapter provides a summary of the work documented in this thesis, thereby highlighting the contributions to knowledge which have been achieved. Motivated by the themes of the work performed, a number of areas of future work are then presented and discussed, and the motivations for performing these works are documented.

Finally, concluding remarks are presented, thereby completing this chapter and this thesis.

7.2 Thesis Summary

This thesis has proposed that the Internet is 'out-of-order'. That is, there has been an inherent assumption in the literature that, in general, packet sequencing is maintained across an end-to-end IP network. The motivation for investigation of packet reordering in the Internet has been presented, and the pioneering work of Bennett [Benn99] has been discussed. Bennett argued that packet reordering is not a pathological problem; it is a naturally occurring phenomenon; it is on the increase, and it is a result of an increased presence in the degree of parallelism apparent in the Internet. Although it has been known that there is Internet parallelism due to multi-path routing and broken network equipment, it was argued by Bennett that switch and link-level parallelism are the real drivers of reordering and are on the increase. This includes link-level striping and switches that allow packets travelling between the same source-destination pair to take different paths through the switch hardware. Parallelism of network paths is on the increase, due to simple economics and an increase in redundancy. Chapter 1 has highlighted the importance of Bennett's work and its contribution to the field of network science, since packet reordering can have a significant impact on both network and application performance. For example, out-of-order arrival of packets can cause apparent loss of data in real time flows, such as voice-over-packet and video streams. Any protocol that is reliant on the ordered arrival of packets can be affected by this phenomenon, e.g. RTP flows based on UDP. Reordering is also detrimental to TCP,

causing it to use available capacity less effectively, and to lose the TCP self-clocking property, thus resulting in irregular data transmission. Bennett hypothesised that, due to the asymmetric nature of the internet, connections will frequently experience reordering in one direction only, and therefore there are three types of packet reordering that must be considered; forward-path reordering or data reordering, reverse-path reordering or Ack reordering, and a combination of both forward and reverse path. Each type of reordering was argued to have different overall effects on a TCP connection.

IPv4 and TCP have allowed a multitude of heterogeneously interconnected systems, all with diverse characteristics, vendors and operating systems, to communicate seamlessly with each other over various communications channels. These protocols have developed over time with various loss recovery and congestion control enhancements added and have been slowly adopted by Internet users. The Fast Retransmit algorithm is an important mandatory algorithm which Chapter 2 has highlighted and has been developed on the assumption that packet reordering does not often occur. Therefore any packet delayed by more than three positions, can be assumed to be lost.

There is clearly a need to measure the amount of packet reordering occurring in the Internet, as the traditional measurement techniques and metrics used in network performance analysis and presented in Chapter 2, are not capable of describing the complex effects of packet reordering. Recent measurement work in the field of Internet packet sequencing is, to some extent, contradictory, as there is a significant variation in the reported numerical measurements. These conflicting findings may be due to different network topologies, switch architectures, underlying link protocols, or the measurement techniques used. However the various studies are sufficiently different in nature that drawing conclusions without further work is difficult, thereby providing an ideal stimulus for further research into measuring and understanding these phenomena.

A wide range of both active and passive measurement techniques has been developed in the literature; some to provide a metric for the amount of reordering that a packet will undergo; others specifically to describe the performance of reordering on TCP. These studies are so diverse in their techniques and assumptions, that it is very difficult to compare results across the literature. The lack of a standard experimental measurement

methodology, and the lack of a standard reordering metric, has been argued to be a significantly limiting factor in understanding the effect, impact and prevalence of packet reordering in today's Internet. A classification taxonomy of metrics and measurement methodologies of packet reordering has been presented in Chapter 3. This taxonomy has classified active measurement Layer 4 techniques as using either Control-Plane packets or Data-Plane packets in order to highlight the potential Middlebox interactions which Control-Plane packets may endure. Middleboxes are difficult to detect and will either prevent active measurements from taking place, or may interfere in the metrics generated from these measurements. The Paxson and Tsinghua active measurements are highlighted in the taxonomy as the methodologies which may generate the most representative results of real network traffic. Unfortunately, their results do not correlate well, differing in the percentage of reordered connections by over 30%. The taxonomy has classified passive measurements by their observation point, and their method of generating RTT estimates. The passive techniques in the literature suffer from difficulties in generating an accurate mid-point estimate of RTT, and from the unrealistic expectation that both forward and reverse paths will flow symmetrically. The Jaiswal and Tstat techniques have been argued to provide the most representative results, but measurements of the total amount of packets undergoing reordering in the Internet vary by over 20% in their estimation.

Limited consideration has been given to measuring and understanding the true drivers of packet reordering, and to correlating these measurements with the effect that they will actually have on a user's application. It is only through this correlation of measurements, that it will be possible to ascertain if packet reordering will affect the user's perceived Quality of Service, and then allow for the design of appropriate metrics and mitigations. Indeed, the literature is sparse when discussing the actual effects of packet reordering on TCP performance, and such assumptions as the behaviour of TCP during reverse-path reordering, have been hypothesised but have not been investigated and measured. A two-point passive measurement technique has been developed in Chapter 4, which has allowed more accurate measurement than previous studies of packet reordering, by exploiting the use of the IPID field as a method to determine the sending sequence of a TCP connection. Simple metrics have been developed that exploit this IPID field, thereby allowing determination of whether a packet has been

reordered, and of the extent by which that packet has moved. The two-point measurement technique has allowed determination of the cause of retransmissions, which are the by-product of packet reordering effects on TCP. By correlating the packet traces obtained at two points, it has been possible to investigate and classify each retransmission, thus providing a more complete analysis of the effects of packet reordering, compared with previous measurement studies. A significant study of over 30,000 FTP sessions during a six month period, of forward, reverse and combined path reordering has been performed. The study of forward path packet reordering has indicated that the effects of packet reordering are negligible with respect to the percentage of reordered packets. But, for every RTT, it has been found that there is a forward-path maximum reordering delay threshold which can be applied to packets, regardless of percentage reordering, and below which reordering has negligible effects. Determination of the value of this threshold, on a specific path, is key to ensuring that a specific switch or router does not introduce reordering to such an extent that it causes unnecessary retransmissions and an associated reduction in throughput. The study of reverse-path packet reordering has demonstrated results, contrary to previous assumptions in the literature, that reverse path reordering has little additional negative effect on the throughput of a connection. Indeed, it has been measured that, in specific circumstances, as a function of the RTT, of the amount of data to be transmitted, and of the reordering delay, reverse path reordering can actually increase throughput of a connection. This phenomenon was explained by the loss of self-clocking during Acknowledgement resequencing, thus allowing the sending TCP *cwnd* to grow faster than normal. A measurement study of combined path reordering has also been performed, and this has illustrated that the effects of forward-path reordering dominate the behaviour of the connection.

The use of percentage-reordered packets as a metric, has been shown to be difficult to correlate with the actual performance of a TCP connection. This suggests that many of the metrics proposed in the literature, such as RFC 4737, are difficult to apply in a way that meaningfully describes the user's Quality of Experience. More sophisticated techniques are therefore required, and the classification taxonomy has highlighted the benefits of using passive techniques which can characterise many thousands of concurrent flows. Performing passive TCP monitoring at a mid-point, though, is not

without its challenges. A packet can easily be identified as being out-of-sequence when it is observed as having a sequence number smaller than or equal to that of a previously observed packet at that measurement point. Explanation of the *cause* of the packet appearing out-of-sequence is challenging as many variables, such as the state machines at the sending and receiving hosts, can only be *inferred* from the packets observed at the mid-point, and therefore a set of heuristics are required in order to examine the packet events observed. A passive mid-point monitoring technique has been developed in Chapter 5. This is lightweight in both its storage requirements and its processing overhead at the mid-point measurement probe. The passive mid-point technique has described the capture and storage of Flow Traces, which can then be analysed to provide an improved measure of Goodput, Loss and Retransmission. A lightweight, mid-point methodology and classification algorithm is developed, and this was applied to live Internet traffic in order to gauge performance when compared with Jaiswal. This mid-point technique improves on previous techniques in the literature, as it does not require calculation of RTT of every concurrent flow observed, nor does it assume visibility of symmetric connections. Finally, a technique for the visualisation of a TCP flow's performance is presented. This technique is superior to others, in that it allows simple evaluation of the degree of resequencing occurring within a TCP connection over time, thereby improving on the metrics presented in RFC 5237.

The literature has indicated that packet reordering and parallelism in fixed networks is on the increase, due to large businesses, ISPs and their vendors aggressively promoting parallel links. This thesis has argued that wireless and mobility will be the next drivers of packet reordering in the future Internet. End-to-end wireless technologies such as WiFi and WiMax, and the use of protocols such as Mobile IP and IP multi-homing, will result in an increase in parallelism at all layers. Wireless links are very different from traditional wired links; the steady-state dropping and reordering probability are independent from link congestion, and so traditional assumptions that loss indicates congestion, are invalid.

It is clear that the predominant focus in measurement research to date, has been to attempt to characterise the performance of packet reordering on TCP. The effects of packet reordering are still in their infancy and, therefore, the literature has concentrated

on describing the most predominant type of traffic in the Internet today. This thesis has proposed that a range of sophisticated measurement techniques will be required in order to characterise the diverse network traffic types occurring in today's Internet, each of which must be *relevant* to the particular application that they are trying to describe. This thesis has argued that work in the literature has not attempted to correlate measurements across layers, to measure reordering at the packet layer, and the resulting effects on the end user application.

The effects of packet reordering are extremely difficult to predict on new complex traffic, such as video over UDP, due to the temporal inter-packet dependencies introduced by the MPEG video encoding structure. An experimental investigation into the effects of video packet reordering using the Windows Media streaming system was presented. A client-side estimator of video QoS was developed, by instrumentation of the Windows Media Player, to allow quantitative analysis of video quality. WM was seen to perform well under low degrees of reordering. Significant reordering resulted in buffer underrun, and in extremely bursty traffic patterns, due to a poorly designed congestion control mechanism. Buffer occupancy measurements were made to describe the number of underruns that a user would expect to experience during reordering, thereby providing a more advanced metric of packet reordering, specific to this particular traffic.

It is clear that the diversity of traffic types prevalent in the Internet today, will require a range of advanced metrics and measurement methodologies, in order to correlate measurements made at various layers in the OSI with the behaviours experienced by the end user. This thesis has illustrated that, although several attempts have been made to measure TCP during reordering, a metric is only truly useful if it is designed for a specific application and specific scenario. The method developed used Buffer Occupancy as a metric of reordering, and is an example of the type of metric required to accurately measure the effects of packet reordering. Clearly, as packet reordering becomes more prevalent, and drivers such as wireless begin to make an impact, there will be further motivation for network operators and application developers to develop similar sophisticated measurement tools.

7.3 Main Contributions

The primary contributions of this thesis relate to

- The various measurement techniques and metrics which have been developed to characterise packet reordering.
- The measurement results that have been obtained using these proposed techniques.
- The comparisons indicating improvements over previous measurement techniques.

An additional contribution is the taxonomy of packet reordering measurements and a review of previous measurement studies performed in the literature.

7.3.1 A Two-Point Passive Measurement Technique

A two-point passive measurement technique has been described in Chapter 4 and prototyped in software. This has allowed more accurate measurement than previous studies of packet reordering, by exploiting the use of the IPID field as a method to determine the sending sequence of a TCP connection. Simple metrics have been developed that exploit this IPID field, thereby allowing determination of whether a packet has been reordered, and the extent by which that packet has moved. Under the high degrees of reordering measured on the testbed, this method has provided a lightweight and simple method for determining the Absolute Reordering of a packet, and avoids the calculation of future Sequence Numbers based on current payload lengths. The two-point measurement technique has allowed determination of the cause of retransmissions, which are the by-product of packet reordering effects on TCP. By correlating the packet traces obtained at two points, it has been possible to investigate and classify each retransmission, thus providing a more complete analysis of the effects of packet reordering, compared with previous measurement studies.

Although this method may not be applicable in the wider Internet, where fragmentation may occur, it does allow a method for highly accurate measurement of reordering in a controlled environment, and could have future applications in the testing of specific reorder-inducing routers or paths.

7.3.2 Development of Testbeds

A number of network testbeds were designed, and implemented to allow for the measurement of packet reordering in a controlled environment, and to allow the development and testing of software prototypes of the measurement methodologies and metrics proposed.

In Chapter 4, a method for emulating TCP packet reordering was demonstrated through the development of software for a configurable router, the development of software probes, and the development of a distributed automated measurement architecture, to perform a large-scale measurement of TCP reordering. This testbed allowed the testing of the developed software probes and metrics, the actual measurement study to be performed, and the validation and development of the algorithms presented in Chapter 5. In Chapter 6, a second testbed was built and configured, and this has allowed measurement of the performance of video traffic during packet reordering. A video disruptor tool was developed to allow testing of video transmission with varying reordering and dropping probabilities, thereby allowing development and testing of the client-side estimator of video QoS.

7.3.3 Large scale measurement studies of packet reordering

A number of measurement studies were performed during this thesis to evaluate the impact of packet reordering on TCP, and to develop and validate improved metrics of TCP packet reordering. In Chapter 4, the passive two-point methodology was used to measure the effects of Forward Path, Reverse Path and Combined Forward and Reverse Path reordering. It has performed one of the largest studies of TCP packet reordering to

date, emulating over 30,000 FTP sessions over a six month period. It has demonstrated the need to develop an autonomous measurement system to perform such a large study, and the methods to perform data management and processing of such large amounts of packet captures. It has improved on the only other measurement study of TCP reordering [Laor02], where the authors used an Agilent QA Robot to randomly delay packets by three positions, but this does not allow investigation of the Reordering Delay, nor investigation of the Reordering Delay with respect to RTT.

The results obtained have questioned the assumptions that packet reordering is an intuitively negative phenomenon. The study of forward-path packet reordering has indicated that, for every RTT, it has been found that there is a forward-path maximum reordering delay threshold which can be applied to packets, regardless of percentage reordering, below which reordering has negligible effects The study of reverse-path reordering has revealed little negative effect on the throughput of a connection. Indeed, it has been measured that, in specific circumstances, as a function of the RTT, and amount of data to be transmitted, and reordering delay, reverse path reordering can actually be beneficial for a connection. The first measurement study of combined path reordering has also been performed, and this has illustrated that the effects of forward path reordering dominate the behaviour of the connection.

The use of percentage reordered packets as a metric has been shown in Chapter 4 to be difficult to correlate with the actual performance of a TCP connection, thus stimulating the development of the algorithms developed in Chapter 5. The Arthur mid-point Classification Algorithm developed in Chapter 5 has been prototyped and applied in a live network environment, thus allowing comparison with the Jaiswal mid-point technique, and with a measurement study of the out-of-order packets in a live network.

7.3.4 A Passive Mid-Point Classification Algorithm of TCP Reordering

A passive mid-point monitoring technique has been developed in Chapter 5, which has clear improvements over previous techniques presented in the literature. The Arthur passive mid-point technique has described the capture and storage of Flow Traces, which can then be analysed to provide an improved measure of Goodput, Loss and Retransmission. The classification algorithm developed is lightweight and improves on the passive mid-point measurement techniques presented in Chapter 3, because it does not require calculation of RTT for every concurrent flow observed, nor does it assume visibility of symmetric connections.

7.3.5 An Improved Visualisation Technique and Metric of TCP Packet Reordering

The passive mid-point monitoring technique has been extended in Chapter 5 to provide an improved technique for the visualisation of a TCP flow's performance. This technique has been demonstrated to be superior to those proposed in RFC 5237, as it allows simple evaluation of the degree of resequencing occurring within a TCP connection over time. The Arthur technique has been developed as a software prototype, and its lightweight real-time abilities have been demonstrated. It builds on the benefits discussed in Section 7.3.4, and its usefulness and relevance as a diagnostic tool for network managers and researchers has been discussed.

7.3.6 A client-side estimator of video QoS

An application-specific metric of packet reordering was developed as an example of the types of cross-layer techniques that will be required in future communications networks. In Chapter 6, a client-side estimator of video QoS was prototyped by instrumentation of the Windows Media Player, to allow quantitative analysis of video quality playback during packet reordering. A simple metric of packet reordering was defined and

measured by using the WM SDK to expose Buffer Occupancy within the player, and this application-relevant technique was contrasted with packet level reordering measurements. These measurements have been correlated, thus substantiating the argument that future metrics of packet reordering should correlate measurements made at various layers in the OSI, to explain behaviours experienced by the user.

7.3.7 Packet Reordering Measurement Taxonomy

A review and taxonomy of packet reordering measurement techniques and metrics has been performed, classifying these techniques as Active and Passive, and identifying the motivation for each, and assumptions made in each. Active techniques were further classified by Control-Plane and Data-Plane packets, while passive techniques were classified by their Observation Position, and their method of estimating RTT. This taxonomy has allowed a survey of the range of packet reordering techniques in the literature, the advantages and limitations of each one, and has highlighted the true novelty and value of the work presented in this thesis.

7.4 Future Directions

A number of areas of future work can be identified during this thesis; some motivated as a direct continuation of the work performed thus far, others based along themes and trends which have been identified.

7.4.1 Packet Reordering as a tool for SLA Compliance

The testbed designed and implemented in Chapter 4 has demonstrated the ability to perform large numbers of experiments, in an automated fashion, over long periods of time. There is therefore the possibility of performing many more experiments. These experiments could emulate reordering over very long satellite-like RTTs, investigate various congestion control algorithms, or investigate the TCP Window Scaling option. It can be speculated though, that this work is unlikely to generate more significant results than those that have already been illustrated. The mandatory Fast Retransmit algorithm has been shown in this thesis to dominate the effects measured on a TCP stream.

An interesting application of results from the testbed, would be to investigate the use of Packet Reordering as a method of mid-point traffic throttling, and therefore SLA compliance across a network. Chapter 4 has illustrated that packet reordering can have both positive and negative effects on the throughput of a TCP connection. Forward-path packet reordering can be used to throttle a connection passing through a mid-point, by selectively delaying packets and forcing a Fast Retransmit. Reverse-path packet reordering has been shown to have the ability to grow a Sender's *cwnd* at a faster rate than normal. This allows the unusual feature that, in theory, from a mid-point position, a Middlebox would be able to both speed-up and slow-down TCP connections.

Rate-limiting Middleboxes, such as the Packeteer PacketShaper [Pack08], perform TCP rate limiting by artificially altering the *rwnd* in the Ack packets from a TCP Receiver, as they pass the PacketShaper at a mid-point in the network. Clearly the processing

required to monitor each concurrent flow, to perform deep packet inspection, to delay each Ack while the header is re-built, and then to re-compute checksums, is highly significant. Packet Reordering could offer the same ability to rate-limit each concurrent flow, but with significantly less processing requirement, as packets do not have to be altered. The additional possibility, that the same Middlebox could also be used to increase the throughput of specific TCP flows, indicates that this potential technique has significant commercial opportunities if it could be developed in a lightweight and robust manner.

Clearly, this proposal would require a number of investigations. Firstly, a rigorous study on the testbed would have to be performed, to allow modelling of the scenarios where reverse-path packet reordering can cause an improvement in performance, in terms of various RTT and congestion control mechanisms. Secondly, a mid-point measurement technique, similar to that discussed in Chapter 5, would be developed to allow measurement of the performance of each flow, and a selection algorithm would be developed, to decide when packet reordering would be applied to a flow, in which direction and in what amount.

7.4.2 Software Routers as Measurement Instruments

The use of Click in Chapter 4 as a configurable router, has allowed a large-scale measurement of packet reordering to take place. It is worth noting, though, that software routers such as Click, are no longer in use only as research tools. Trends indicate that future routers will be built upon flexible hardware, with open source Operating Systems. This is expected to be the driver of 'Router Virtualisation' research[Egi07], where a single hardware platform can simultaneously perform the roles of multiple independent routers. Previously, instrumentation *inside* a router was impossible due to the commodity hardware and closed software. This motivated projects such as NetFPGA[Mcke07], to build open router platforms where measurements such as the effects of mid-point buffering on TCP [Appe04] could be investigated. The authors of Click have already acknowledged [Kohl06] that their software language could be extended to design mid-point measurement systems and, therefore, as software routers become more prevalent in production networks, this

affords the opportunity for large scale passive mid-point measurements to be made.

The classification algorithm and visualisation technique presented in Chapter 5 could be implemented within a software router, to test the scalability and performance of the algorithm, in a variety of traffic conditions, thus producing a large scale measurement of empirical traffic. Software routers and virtualisation will allow for mid-point passive monitoring techniques to be deployed for many different traffic types, and therefore there is a great deal of motivation to extend the work presented in Chapter 5, to classify out-of-sequence packets passively, for a variety of traffic types.

The method presented in Chapter 4, whereby the IPID of a TCP flow was used as a method for measuring the absolute packet reordering within a flow, could also be useful as a method for router testing. In both hardware and software routers, the Chapter 4 technique provides a lightweight and highly accurate method for measuring packet displacement. In a software router, this method could be used as a method to monitor, in real-time, the amount of reordering which that router was inducing. Software probes on each line-card could correlate IPID values between input and output ports, thus measuring the degree of reordering induced on each flow.

The use of the IPID field has only recently been proposed as a method to infer network measurements [Chen05]. This thesis has suggested that the IPID measurement technique proposed in Chapter 4 could only operate within a testbed network, due to the variety of methods to increment the IPID field. However, work by Chen [Chen05] has suggested that packet order in the Internet can be measured in this way. Further investigation and deployment of the proposed technique on real network traffic, would be a useful further study.

7.4.3 Extending the Arthur Classification and Visualisation Algorithm

A benefit of mid-point passive measurement techniques is that they can be applied as multiple probes throughout a network. Assuming that the probes can be time-synchronised to a high degree of accuracy, packet captures can be correlated between probes thereby allowing end-to-end measurements to be made.

An interesting area of future work would be the development of a Distributed Measurement Architecture, whereby multiple Arthur out-of-sequence classification probes would be deployed across a network. Correlation of the Flow Traces created at various points around the network, would allow development of further heuristics in the algorithm, to identify the location of reorder-inducing paths and nodes.

A second area of interesting research would be to investigate the design of an out-of-sequence classification algorithm, based entirely on Flow Traces created by Acknowledgements observed. This classification algorithm would provide mid-point analysis of TCP flows, based entirely on information inferred from the Acknowledgement packets. This is a challenging endeavour. With packets measured in the forward path, it is possible to calculate the NESN for each Seq observed, thus allowing simple identification of packet loss. Based entirely on Acknowledgements alone, it is not possible to predict the next expected packet, thereby making loss identification difficult. This is complicated further by the use of SACK and Partial-Acks. Nevertheless, a number of useful metrics could be defined, such as a Fast Retransmit counter based on observing triple duplicate-Acks.

A third possible interesting extension to the Arthur algorithm, would be to extend the algorithm to accommodate non-TCP traffic, such as UDP or RTP. A method of determining the sending sequence of packets would be required, such as the IPID field

if it is seen to increase monotonically from a host. This would also allow extension of the Arthur Visualisation Technique, which would indicate the lateness or earliness of packets, and thresholds could be defined which were relevant to the application-traffic being carried.

7.4.4 Cross-layer Correlation of Packet Reordering Metrics

This thesis has argued that, for a packet reordering metric to be relevant, it must be specific to the user application traffic. This thesis has also argued that wireless is likely to be the next driver of packet reordering, due to parallelism occurring at multiple layers in a wireless network. Recent work has suggested that TCP can be used effectively on Multihop Wireless [Fu05] and Wireless Mesh [Liu07] networks. Worryingly, other work has argued that IPv6 [Blan06] and TCP [Dunk04] are both viable for use on Wireless Sensor Networks ! TCP is already in use on Wi-Fi and WiMAX networks, on which it is likely that Mobile-IP will be used to provide session continuity during handover. Recent research in Mobility Management protocols for multiple interfaces, has measured [Park08] [Tsan08] that wireless multihoming causes severe packet reordering, significantly degrading the handover performance. There are a range of research topics in this area; it is clear that packet reordering has not previously been considered by Mobility Management researchers, and therefore a range of performance studies could be performed on the various mobility protocols proposed

However, an interesting research approach to this area, would be determining the cross-layer impacts that wireless packet reordering will have on each upper layer. It is clear from this thesis that a packet can be defined as in-sequence or out-of-sequence, without significant difficulty. The difficulty occurs when relating this low-level movement of packets in a flow, to a measurable event in the user's application-level experience. Chapter 6 has highlighted this for a specific type of video traffic. In a recent SIGCOMM paper, Cheng [Chen07b] attempted to correlate 802.11b Layer 2 metrics of loss caused by queuing, back-offs and contention, with the Layer 4 TCP performance

observed. The result of this work is a graph of TCP Goodput, plotted against various Layer 2 loss metrics, which indicates the causes of each loss, thereby providing root-cause analysis of TCP performance.

A similar study is required for packet reordering. Simple metrics can define the amount of reordering occurring at each Layer of the OSI. But, for a measured drop in TCP throughput to be *explained*, this event must be correlated with the packet reordering metrics obtainable at the various lower layers, to identify the cause-and-effect of reordering on a particular flow.

7.5 Concluding Remarks

The explosive growth of the Internet has allowed a multitude of heterogeneously interconnected systems to communicate seamlessly with each other, and to carry a diverse mix of traffic and applications. Simple byte-windowing protocols are being driven in diverse ways for which they were never intended. The Internet is becoming more complex and highly parallelised; new wireless technologies, transport protocols and new applications present a variety of challenges to network designers and operators. The effects of packet reordering have only recently become apparent; they will get worse! Many questions will arise around the implications of packet reordering for re-engineering the future internet. Is a new TCP needed or new router designs? Should future networks be reordering-free, or should future protocols be reordering-tolerant? Never before has network measurement science been so important for the future of the Internet, to ensure that the high expectations of users can be fulfilled.

It is clear that the diversity of traffic types prevalent in the future Internet, will require a range of sophisticated metrics and measurement methodologies, in order to explain the effects of packet reordering on a user's Quality of Experience. This thesis has addressed the issue of measuring the impact of packet reordering on Internet traffic, by proposing a number of measurement methodologies and metrics that will be required in order to describe the complex cross-layer effects of packet reordering on future complex traffic types.

Bibliography

[Adam04] Adamson, B., Bormann, C., Handley, M., Macker, J., Negative-acknowledgement (NACK)-Oriented Reliable Multicast (NORM) Protocol, IETF Network Working Group, Request For Comments RFC 3940, November 2004.

[Aika03] Aikat, J., Kaur, J., Smith, F., Jeffay, K., Variability in TCP round-trip times, In Proceedings ACM SIGCOMM Internet Measurement Conference, Miami Beach, Florida, Oct. 2003, pp. 279-284.

[Appe04] Appenzeller, G., Keslassy, I., McKeown, N., Sizing Router Buffers, in Proceedings of ACM SIGCOMM 2004, Portland, August 2004.

[Att07] AT&T Managed Internet Service, Accessed December 2007, http://new.serviceguide.att.com/mis.htm .

[Awdu02] Awduche, D., Overview and Principles of Traffic Engineering, IETF Network Working Group, Request For Comments RFC 3272, May 2002.

[Allm99] Allman, M., Paxson, V., IETF Network Working Group, Request For Comments RFC 2581, TCP Congestion Control, April 1999.

[Allm01] Allman, M., Enhancing TCP's Loss Recovery Using Limited Transmit IETF Network Working Group, Request For Comments RFC 3042, January 2001.

[Allm03] Allman, M., TCP Congestion Control with Appropriate Byte Counting (ABC), IETF Network Working Group, Request For Comments RFC 3465, February 2003.

[Allm03b] Allman, M., On the Performance of Middleboxes, In Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, Florida, USA, 2003, pp. 307-312.

[Allm07] Allman, M., TCP Congestion Control, IETF TCP Maintenance Working Group, Internet Draft http://tools.ietf.org/id/draft-ietf-tcpm-rfc2581bis-03.txt, September 2007.

[Alme99a] Almes, G., A One-way Delay Metric for IPPM, IETF Network Working Group, Request For Comments RFC 2679, September 1999.

[Alme99b] Almes, G., A Round-trip Delay Metric for IPPM, IETF Network Working Group, Request For Comments RFC 2681, September 1999.

[Alme99c] Almes, G., A One-way Packet Loss Metric for IPPM, IETF Network Working Group, Request For Comments RFC 2680, September 1999.

[Alqm92] Almquist, P., Type of Service in the Internet Protocol Suite, IETF Network Working Group, Request For Comments RFC 1349, July 1992.

[Bake95] Baker, F., Requirements for IP Version 4 Routers, IETF Network Working Group, Request For Comments RFC 1812, June 1995.

[Bare07] Bare, A.A., Jayasumana, A.P., Piratla, N.M., On Growth of Parallelism within Routers and Its Impact on Packet Reordering, 15th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), 10-13 June 2007, pp. 145-150.

[Barf04] Barford, P., Sommers, J., Comparing Probe-and Router-Based Packet-Loss Measurement, IEEE Internet Computing, vol.8, no.5, Sept.-Oct. 2004, pp. 50-56.

[Bhan03] Bhandarkar, S., Narasimha Reddy, A.L., TCP-DCR: Making TCP Robust to Non-Congestion Events, in Proceedings of Networking Conference, Athens, Greece, 9-14 May 2004.

[Bhan06] Bhandarkar, S., Narasimha Reddy, A.L., Allman, M., Blanton, E., Improving the Robustness of TCP to Non-Congestion Events, IETF Network Working Group, Request For Comments RFC4653, August 2006.

[Blak98] Blake, S., An Architecture for Differentiated Services, IETF Network Working Group, Request For Comments RFC2475, December 1998

[Blan02] Blanton, E., Allman, M., On Making TCP More Robust to Packet Reordering, SIGCOMM Computer Communications Review, Vol. 32, Issue , pp 20-30, January 2002.

[Boha03] Bohacek, S., Hespanha, J.P., Lee, J.S., Lim, C.S., Obraczka, K., TCP-PR: TCP for Persistent Packet Reordering, In Proceedings of 23rd International Conference on Distributed Computing Systems, 19-22 May 2003, pp. 222-231.

[Blan06] Blanchet, M., IPv6 Primer for Sensor Networks, 2006, www.viagenie.ca/publications/2006-05-31-sensometworks-ipv6primer.pdf Accessed 24th September 2008.

[Bell02] Bellardo, J., Savage, S., Measuring Packet Reordering, In Proceedings of the 2nd ACM SIGCOMM Workshop on internet Measurement (Marseille, France, November 06 - 08, 2002). IMW '02. ACM, New York, NY, 97-105.

[Bell03] Sting Tool, version 0.8.1, March 2003 http://sysnet.ucsd.edu/reordering/index.html

[Benk02] Benko, P., Veres, A., A Passive Method for Estimating End-to-End TCP Packet Loss, in Proceedings IEEE Global Telecommunications Conference (GLOBECOM), vol.3, pp. 2609-2613 vol.3, 17-21 Nov. 2002.

[Benk04] Benko, P., Malicsko, G., Veres, A., A Large-Scale, Passive Analysis of End-to-End TCP Performance over GPRS, in Proceedings of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2004, 7-11 March 2004, pp. 1882-1892

[Benn99] Bennett, J., Partridge, C., Shectman, N., Packet Reordering is Not Pathological Network Behaviour, IEEE/ACM Transactions on Networking, Vol. 7, Issue 6, December 1999, pp. 789-798.

[Brad89] Braden, R., Requirements for Internet Hosts -Communication Layers, IETF Network Working Group, Request For Comments RFC1122, October 1989.

[Brad91] Bradner, S., Benchmarking Terminology for Network Interconnection Devices, IETF Network Working Group, Request For Comments RFC1242, July 1991.

[Brad97a] Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, IETF Network Working Group, Request For Comments RFC2119, March 1997.

[Brad97b] Braden, R., Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification, IETF Network Working Group, Request For Comments RFC2205, September 1997.

[Brow01] Brownlee, N., Loosley, C., Fundamentals of Internet Measurement - A Tutorial, CMG Journal of Computer Resource Management, April 2001

[Bt07] BT Global Services, IP Performance Hourly Network Summary, Round-Trip Delay and Loss Measurements, http://ippm.bt.net/, Accessed 25th September 2007.

[But05] But, J., Keller, U., Kennedy, D., Armitage, G., Passive TCP Stream Estimation of RTT and Jitter Parameters, in Proceedings of The IEEE Conference on Local Computer Networks, 30th Anniversary, 15-17 November 2005.

[Case90] Case, J., A Simple Network Management Protocol (SNMP), IETF Network Working Group, Request For Comments RFC1157, May 1990.

[Chen05] Chen, W., Huang, y., Ribeiro, b., Suh, K., Zhang, H., Souza e Silva, E., Kurose, J., Towsley, D., Exploiting the IPID field to infer network path and end-system characteristics, in Proceedings of Passive and Active Measurement 2005 (PAM2005), Boston.

[Chen07] Cheng, R., Lin, H., Protecting TCP from A Misbehaving Receiver, International Journal of Network Management, vol.17, no.3, June 2007.

[Chen07b] Cheng, Y., Afanasyev, M., Verkaik, P., Benkö, P., Chiang, J., Snoeren, A. C., Savage, S., and Voelker, G. M. 2007. Automating cross-layer diagnosis of enterprise wireless networks. ACM SIGCOMM Computer Communications Review, vol. 37, no, 4, pp 25 – 36, October 2007.

[Ciav03] Ciavattone, L., Morton, A., Ramachandran, G., Standardized Active Measurements on A Tier 1 IP Backbone, IEEE Communications Magazine, vol.41, no.6, June 2003, pp. 90-97.

[Cisc07] Cisco Catalyst Switched Port Analyser Configuration Example, December 2007, http://www.cisco.com/warp/public/473/41.html.

[Clai04] Claise, B., Cisco Systems NetFlow Services Export Version 9, IETF Network Working Group, Request For Comments RFC3954, October 2004.

[Clar82] Clark, D., Window and Acknowledgement Strategy in TCP, IETF Network Working Group, Request For Comments RFC813, July 1982.

[Clic08] The Click Modular Router Project, http://www.pdos.lcs.mit.edu/click/ , Accessed 25th September 2008.

[Dala03] Dalal, A. C., Perry, E., A New Architecture for Measuring and Assessing Streaming Media Quality, in Proceedings of the Passive and Active Measurement Workshop (PAM 2003), La Jolla, California, 6-8 April 2003.

[Deer98] Deering, S., Hinden, R., Internet Protocol, Version 6 (IPv6) Specification, IETF Network Working Group, Request For Comments RFC 2460, December 1998.

[Deer01] Deering, S., Watching the Waist of the Protocol Hourglass, IETF Meeting 51, Plenary Presentation, London, 30th August 2001.

[Demi02] Demichelis, C., IP Packet Delay Variation Metric for IP Performance Metrics (IPPM), IETF Network Working Group, Request For Comments RFC3393, November 2002.

[Deri04] Deri, L., Improving Passive Packet Capture: Beyond Device Polling, In Proceedings of the 4th International System Administration and Network Engineering Conference, Amsterdam, September 2004.

[Dsac08] Reordering Robust DSACK TCP, (RR-TCP), http://www.icir.org/bkarp/RR-TCP/index.html , Accessed 25th September 2008.

[Dunk04] Dunkels, A, Voigt, T., Alonso, J., Making TCP/IP Viable for Wireless Sensor Networks, In Proceedings of First European Workshop on Wireless Sensor Networks (EWSN 2004), January 2004, Berlin, Germany.

[Egi07] Egi, N.; Greenhalgh, A.; Handley, M.; Hoerdt, M.; Mathy, L.; Schooley, T., "Evaluating Xen for Router Virtualization," In Proceedings of 16th International Conference on Computer Communications and Networks, 2007. ICCCN 2007. , pp.1256-1261, 13-16 Aug. 2007

[Fall96] Fall, K., Floyd, S., Simulation-Based Comparisons of Tahoe, Reno, and SACK TCP, ACM Computer Communication Review, vol.26, no.3, July 1996, pp. 5-21.

[Fang03] Fang, F., DeDourek, J., Verifying TCP Implementation, In Proceedings of Communication Networks and Services Conference 2003, Moncton, Canada, May 15-16 2003.

[Feng07] Feng, J., Ouyang, Z., Xu, L., Ramamurthy, B., Packet Reordering in High-Speed Networks and its Impact on High-Speec Variants, in Proceedings of 5th International Workshop on Protocols for Fast Long-Distance Networks, Los Angeles, California, USA, 7-9 Feb. 2007.

[Full07] Fullmer, M, Flow-tools – Tool Set for Working with NetFlow Data, December 2007, http://www.splintered.net/sw/flow-tools/docs/flow-tools.html.

[Floy00] Floyd, S., Mahdavi, J.,Mathis, M., Podolsky, M., An Extension to the Selective Acknowledgement (SACK) Option for TCP, IETF Network Working Group, Request For Comments RFC2883, July 2000.

[Floy04] Floyd, S., Henderson, T., Gurtov, A., The NewReno Modification to TCP's Fast Recovery Algorithm, IETF Network Working Group Request For Comments RFC3782, April 2004.

[Floy07] Floyd, S., Metrics for the Evaluation of Congestion Control Mechanisms, IETF Internet Draft, http://tools.ietf.org/html/draft-irtf-tmrg-metrics-11, October 2007.

[Fra03] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, S.C., Packet-level traffic measurements from the Sprint IP backbone, IEEE Network, vol.17, no.6, pp. 6-16, Nov.-Dec. 2003.

[Fu05] Fu, Z., Luo, H., Zerfos, P., Lu, S., Zhang, L., Gerla, M., The impact of multihop wireless channel on TCP performance, IEEE Transactions on Mobile Computing, vol.4, no.2, pp. 209-221, March-April 2005.

[Full93] Fuller, V., Yu, J., Varadhan, K., Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy, IETF Network Working Group, Request For Comments RFC1519, September 1993.

[Gare97] Gareiss, R., Is The Internet in Trouble?, Data Communications Magazine, September 1997, pp. 36–50.

[Ghar04] Gharai, L. and Perkins, C. and Lehman, T., Packet Reordering, High Speed Networks and Transport Protocol Performance, In 13th International Conference on Computer Communications and Networks, Chicago, 11-13 October 2004, pages pp. 73-78.

[Gean08] The GÉANT project, http://www.geant.net/, , Accessed 25th September 2008.

[Hens08] Hens, F., Caballero, J., Triple Play: Building the Converged Network for IP, VoIP and IPTV, Published by John Wiley & Sons, 2008, ISBN 0470753676.

[Hobb97] Zakon, R., Hobbes' Internet Timeline, IETF Network Working Group, Request For Comments RFC2235, November 1997.

[Hobb06] Zakon, R., Hobbes' Internet Timeline v8.2, http://www.zakon.org/robert/internet/timeline/, Accessed 25th September 2008.

[Huaw07] Huawei Technologies Co. Ltd, Technical White Paper for NetStream, http://www.huawei.com/products/datacomm/pdf/view.do?f=65. , Accessed 25th September 2008.

[Hust03] Huston, G., Measuring IP Network Performance, The Internet Protocol Journal, vol.6, Issue 1, March 2003.

[Iana07] IANA Assigned Protocol Numbers, 12 February 2007, http://www.iana.org/assignments/protocol-numbers.

[Ieee07] IEEE 1588TM-2002 Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, http://ieee1588.nist.gov/, Accessed 25th September 2008.

[Info07] Howard, M., Ethernet and IP MPLS VPN Growth Continues, Infonetics Research Report, http://searchnetworkingchannel.techtarget.com/generic/0,295582,sid100_gci1256434,00.html, 25th May 2007.

[Info08] Dornan, A., Does Cisco's Switch to Linux Make IOS More Open?', Information Week, March 2008, http://www.informationweek.com/blog/main/archives/2008/03/does_ciscos_swi.html, Accessed 20th August 2008.

[Itu02] ITU-R Recommendation BT.500-11, 'Methodology for the subjective assessment of the quality of television pictures', 2002.

[Jaco88] Jacobson, V., Congestion Avoidance and Control, ACM Computer Communication Review, vol.18, no.4, Aug. 1988, pp. 314-329.

[Jaco90] Jacobson, V., Modified TCP Congestion Avoidance Algorithm, end2end-interest mailing list, April 30, 1990, ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[Jaco92] Jacobson, V., TCP Extensions for High Performance, IETF Network Working Group, Request for Comments 1323, May 1992.

[Jais02] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D., Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone, In Proceedings of the 2nd Internet Measurement Workshop, ACM Press, Marseille, France, November 6 - 8, 2002.

[Jais03] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D., Inferring TCP Connection Characteristics Through Passive Measurements (extended version – technical report), Sprint Labs Technical Report RR03-ATL-070121, Sprint ATL, July 2003.

[Jais04] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D., Inferring TCP Connection Characteristics Through Passive Measurements, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2004, vol.3, 7-11 March 2004, pp. 1582-1592.

[Jais07] Jaiswal, S., Iannaccone, G., Diot, C., Kurose, J., Towsley, D., Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone, IEEE/ACM Transactions on Networking, vol.15, no.1, Feb. 2007, pp. 54-66.

[Jaya03] Jayasumana, A., Piratla, N. M., Bare, A. A., Banka, T., Internet Draft: Reorder Density Function - Metric for Packet Reordering Measurement. Tech. rep., IETF, February 2003, draft-jayasumana-reorder-density-00.txt.

[Jaya08] Jayasumana, A., Piratla, N., Banka, T., Bare, A., Whitner, R., Improved Packet Reordering Metrics, IETF Network Working Group, Request For Comments RFC 5236, June 2008.

[Jian02] Jiang, H., Dovrolis, C., Passive Estimation of TCP Round-Trip Times, ACM SIGCOMM Computer Communications Review, vol.32, no.3, 2002, pp. 75-88.

[Juni07] Juniper Networks, Configuring Flow-Based Statistics Collection, http://www.juniper.net/techpubs/software/erx/junose60/swconfig-routing-vol1/html/ip-jflow-stats-config4.html#560916 , Accessed 25th September 2008.

[Juni08] Juniper Networks, M160 Internet Router Overview, Accessed 19th August 2008, http://www.juniper.net/techpubs/software/nog/nog-hardware/html/m160-router.html.

[Kand07] Kandula, S., Katabi, D., Sinha, S., and Berger, A., Dynamic Load Balancing Without Packet Reordering, In ACM SIGCOMM Computer Communication Review, vol. 37, no. 2, Mar. 2007, pp. 51-62.

[Kam87] Kam, P., Partridge, C., Round Trip Time Estimation, ACM SIGCOMM-87, August 1987.

[Kohl00] Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M. F., The Click Modular Router, ACM Transactions on Computer Systems vol.18, no.3, August 2000, pp. 263-297.

[Kohl06] Kohler, E., Click for Measurement, UCLA Computer Science Department Technical Report TR060010, February 2006.

[Koo02] Koodli, R., One-way Loss Pattern Sample Metrics, IETF Network Working Group, Request For Comments RFC3357, August 2002.

[Kopp02] Kopparty, S., Krishnamurthy, S.V., Faloutsos, M., Tripathi, S.K., Split TCP for Mobile Ad-hoc Networks, in Proceedings IEEE Global Telecommunications Conference, GLOBECOM '02, vol.1, pp. 138-142, 17-21 Nov. 2002.

[Laor02] Laor, M., Gendel, L., The Effect of Packet Reordering in A Backbone Link on Application Throughput, IEEE Network, vol.16, no.5, pp. 28-36, Sep/Oct 2002.

[Lai03] Lai, W., Requirements for Internet Traffic Engineering Measurement, IETF Internet Draft, http://tools.ietf.org/html/draft-ietf-tewg-measure-06 July 2003.

[Lee02] Lee, Y., Park, I., Choi, Y., Improving TCP Performance in Multipath Packet Forwarding Networks, Journal of Communications and Networks, vol.4, no.2, June 2002, pp. 148-157.

[Lein04] Leinen, S., Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX), IETF Network Working Group, Request For Comments RFC 3955, October 2004.

[Leun07] Leung, K., Li, V., Yang, D., An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges, IEEE Transactions on Parallel and Distributed Systems, vol.18, no.4, April 2007, pp. 522-535.

[Ligh01] Light Reading (2001), Internet Core Router Test, Accessed 19 August 2008, http://www.lightreading.com /document.asp?doc_id=4009&page_number=8.

[Liu07] Liu, C., Shen, F., Sun, M., A Unified TCP Enhancement for Wireless Mesh Networks, in Proceedings of International Conference on Parallel Processing Workshops, 2007. ICPPW 2007, pp.71-71, 10-14 Sept. 2007

[Logu02] Loguinov, D., Radha, H., End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis, In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2002, vol.2, 2002, pp. 723-732.

[Love06] Love, S., Archibald, D. M., Measuring Efficiency of Data Transmission, United States Patent 7075887, Application Number 09/735607, 07 November 2006.

[Luck01] Luckie, M., McGregor, A., Braun,. H., Towards Improving Packet Probing Techniques, In Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, San Francisco, California, November 1 – 2, 2001, pp.145-150.

[Luo05] Luo, X., Chang, R., Novel Approaches to End-to-End Packet Reordering Measurement, In Proceedings of ACM/SIGCOMM Internet Measurement Conference, USA, October 2005, pp. 227-238.

[Ma04a] Ma, C.M., Leung, K.C., Improving TCP Reordering Robustness in Multipath Networks, 29th Annual IEEE International Conference on Local Computer Networks, 16-18 Nov. 2004, pp. 409-410.

[Ma04b] Ma, C.M., Leung, K.C., Improving TCP Robustness Under Reordering Network Environment, IEEE Global Telecommunications Conference, GLOBECOM '04, vol.2, no., 29 Nov.-3 Dec. 2004, pp. 828-832.

[Mae08] Verizon MAE East, Internet Exchange Point, Accessed 25th September 2008, http://www.mae.net/fac/mae-east.htm

[Math97] Mathis, M., Semke, J., and Mahdavi, J., The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, ACM SIGCOMM Computer Communications Review, vol. 27, no. 3, July 1997, pp. 67-82.

[Mart00] Martin, H.S., McGregor, A., Cleary, J.G., Analysis of Internet Delay Times, In Proceedings of Passive and Active Measurement Workshop (PAM 2000), April 2000.

[Mart02] Martin, J., Nilsson, A., On Service Level Agreements for IP Networks, IEEE INFOCOM, 2002.

[Malk93] Malkin, G., The Tao of IETF - A Guide for New Attendees of the Internet Engineering Task Force, IETF Network Working Group, Request For Comments RFC1391, January 1993.

[Math96] Mathis, M., TCP Selective Acknowledgment Options, IETF Network Working Group, Request For Comments RFC2018, October 1996.

[Mcke07] McKeown, N., Lockwood, J., Naous, J., Gibb, G., Covington, A., Hands-on with the NetFPGA to build a Gigabit-rate Router, in Proceedings of the 15th Annual IEEE Symposium on High-Performance Interconnects, 2007. HOTI 2007, pp.7-10, 22-24 Aug. 2007

[Medi05] Medina, A., Allman, M., and Floyd, S., Measuring the Evolution of Transport Protocols in The Internet, ACM SIGCOMM Computer Communication Review, vol.35, no.2, April 2005.

[Mell06] Mellia, M., Meo, M., Muscariello, L., Rossi, D., Passive Identification and Analysis of TCP Anomalies, in Proceedings of the IEEE International Conference on Communications, ICC '06., vol. 2, June 2006, pp.723-728.

[Micr08] Microsoft Corporation, Description of Windows 2000 and Windows Server 2003 TCP Features, Accessed 23rd August 2008, http://support.microsoft.com/kb/224829.

[Micr08b] Microsoft Windows Media, Accessed 12th September 2008, http://www.microsoft.com/windows/windowsmedia/.

[Mill83] Mills, D. L., Internet Delay Experiments, IETF Network Working Group, Request For Comments RFC 889, December 1983.

[Mizr06] Mizrak, A.T., Yu-Chung Cheng, Marzullo, K., Savage, S., Detecting and Isolating Malicious Routers, IEEE Transactions on Dependable and Secure Computing, vol.3, no.3, July-Sept. 2006, pp. 230-244.

[Mock87] Mockapetris, P., Domain Names – Implementation and Specification, IETF Network Working Group, Request For Comments RFC1035, November 1987.

[Mort06] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., Perser, J., Packet Reordering Metrics, IETF Network Working Group, Request For Comments RFC4737, November 2006.

[Nasa08] Nasa Engineering and Research Network, Accessed 22nd August 2008, http://www.nren.nasa.gov/tcp_tuning.html.

[Negl04] Neglia, G., Falletta, V., Is TCP Packet Reordering Always Harmful?, Modeling, In Proceedings of IEEE Computer Society's 12th Annual International Symposium on Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004), 4-8 Oct. 2004, pp. 87-94.

[Nehm03] Nehme, A., Phillips, W., Robertson, W., The effect of Reordering and Dropping Packets on TCP Over A Slow Wireless Link, IEEE Canadian Conference on Electrical and Computer Engineering, IEEE CCECE 2003, vol.3, 4-7 May 2003, pp. 1555-1558.

[Nich98] Nichols, K., Definition of the Differentiated Services Field (DS Field) in The IPv4 and IPv6 Headers, IETF Network Working Group, Request For Comments RFC2474, December 1998.

[Nich01] Nichols, K., Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification, IETF Network Working Group, Request For Comments RFC3086, April 2001.

[Oliv02] Oliveira, R., Braun, T., TCP in Wireless Mobile Ad Hoc Networks, Technical Report, IAM-02-003, University of Bern, July 2002.

[Oste08] Ostermann, S., TCP Trace Website, Retrieved 10th September 2008, http://tcptrace.org/index.html.

[Pack08] Packeteer PacketShaper, http://www.packeteer.com/products/ Accessed 24th September 2008.

[Park98] Parker, S., Schmechel, C., Some Testing Tools for TCP Implementers, IETF Network Working Group, Request for Comments RFC 2398, August 1998.

[Park08] Park, M., Lee, J., Kim, B., Kim, D., Design of fast handover mechanism for multiple interfaces mobile IPv6, in Proceedings of 3rd International Symposium on Wireless Pervasive Computing, 2008. ISWPC 2008., pp.697-701, 7-9 May 2008

[Paxs96] Paxson, V., End-to-End Routing Behavior in The Internet, In Proceedings ACM SIGCOMM, ACM Press, Stanford, CA, pp. 25–39, 1996.

[Paxs97] Paxson, V., Measurements and Analysis of End-to-End Internet Dynamics, University of California, Berkeley, Ph.D. dissertation, April 1997, ftp://ftp.ee.lbl.gov/papers/vp-thesis/dis.ps.gz

[Paxs97a] Paxson, V., End-to-End Routing Behavior in The Internet, IEEE/ACM Transactions on Networking, vol.5, no.5, Oct 1997, pp. 601-615.

[Paxs97b] Paxson, V., End-to-End Internet Packet Dynamics, In Proceedings ACM SIGCOMM, Cannes, France, September 1997.

[Paxs97c] Paxson, V., Automated Packet Trace Analysis of TCP Implementations, In Proceedings ACM SIGCOMM, Cannes, France, September 1997.

[Paxs99] Paxson, V., End-to-End Internet Packet Dynamics, IEEE/ACM Transactions on Networking, vol.7, no.3, Jun 1999, pp. 277-292.

[Paxs00] Paxson, V., Computing TCP's Retransmission Timer, IETF Network Working Group, Request For Comments RFC2988, November 2000.

[Paxs98] Paxson, V., Framework for IP Performance Metrics, IETF Network Working Group, Request For Comments RFC2330, May 1998.

[Paxs04] Paxson, V., Strategies for Sound Internet Measurement, In Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement, Taormina, Siciliy, Italy, 2004, pp. 263-271.

[Perk02] Perkins, C. S., Gharai, L., Lehman, T., Mankin, A., Experiments with Delivery of HDTV Over IP Networks, In Proceedings of 12th International Packet Video Workshop, Pittsburgh, April 2002.

[Pipe07] Pipex Pipeline Service Level Agreement, December 2007, http://www.dial.pipex.com/legal/sla/pipeline.shtml.

[Pira06] Piratla, N.M., Jayasumana, A.P., Reordering of Packets Due to Multipath Forwarding - An Analysis, IEEE International Conference on Communications, ICC '06., vol.2, June 2006, pp.829-834.

[Pira08] Piratla, N., Jayasumana, A., Metrics for Packet Reordering – A Comparative Analysis, International Journal of Communication Systems, vol. 21, issue 1, January 2008, pp. 99-113.

[Pore06] Poretsky, S., Terminology for Benchmarking Network-layer Traffic Control Mechanisms, IETF Network Working Group, Request For Comments RFC4689, October 2006.

[Post80] Postel, J., User Datagram Protocol, IETF Network Working Group, Request For Comments RFC768, August 1980.

[Post81] Postel, J., Internet Control Message Protocol, IETF Network Working Group, Request For Comments RFC792, September 1981.

[Post81a] Postel, J., Internet Protocol, IETF Standard, Network Working Group, Request For Comments RFC791, September 1981.

[Post81b] Postel, J., Transmission Control Protocol, IETF Network Working Group, Request For Comments RFC793, September 1981.

[Post81c] Postel, J., Internet Control Message Protocol, IETF Network Working Group, Request For Comments RFC792, September 1981.

[Przy05] Przybylski, M., Belter, B., Binczewski, A., Shall We Worry About Packet Reordering?, Computational Methods in Science and Technology, vol. 11, no. 2, 2005, pp. 141-146.

[Ragh07] Raghunath, S., Ramakrishnan, K. K., and Kalyanaraman, S. 2007. Measurement-based characterization of IP VPNs. IEEE/ACM Transactions on Networking, vol. 15, no 6, December 2007, pp1428-1441.

[Rama01] Ramakrishnan, K., The Addition of Explicit Congestion Notification (ECN) to IP, IETF Network Working Group, Request For Comments RFC3168, September 2001.

[Rais02] Räisänen, V., Network Performance Measures with Periodic Streams, IETF Network Working Group, Request For Comments RFC 3432, November 2002..

[Rais02b] Raisinghani, V.T., Singh, A.K., Iyer, S., Improving TCP Performance Over Mobile Wireless Environments Using Cross Layer Feedback, IEEE International Conference on Personal Wireless Communications 2002, New Delhi, India, 15-17 December 2002, pp. 81-85.

[Rais03] Räisänen, V., Implementing Service Quality in IP Networks, Addison Wesley, 2003, ISBN 047084793X

[Rewa06a] Rewaskar, S., Kaur, J., Smith, F.D., A Passive State-Machine Based Approach for Reliable Estimation of TCP Losses, In Proceedings of the 7th Passive and Active Measurements Conference (PAM'06), Adelaide, Australia, March 2006.

[Rewa06b] Rewaskar, S., Kaur, J., Smith, F.D., A Passive State-Machine Approach for Accurate Analysis of TCP Out-of-Sequence Segments, In Proceedings of ACM Computer Communication Review, vol.36, issue 3, Jul. 2006, pp. 51-64.

[Ripe08] RIPE Test Traffic Measurements, Accessed 25th September 2008, http://www.ripe.net/tt,

[Sath05a] Sathiaseelan, A., Radzik, T., Reorder Notifying TCP (RN-TCP) with Explicit Packet Drop Notification (EPDN), International Journal of Communication Systems, Wiley, vol.19, issue 6, 2005, pp. 659 - 678.

[Sath05b] Sathiaseelan, A., Radzik, T., Robust TCP (TCP-R) with Explicit Packet Drop Notification (EPDN) for Satellite Networks, In Proceedings of the 4th International Conference on Networking (ICN '05), Reunion Island, France, LNCS 3421, April 2005, pp. 250-257.

[Sava99] Savage, S., Cardwell, N., Wetherall, D., and Anderson, T., TCP Congestion Control with A Misbehaving Receiver, ACM SIGCOMM Computer Communications Review, vol.29, no.5, October 1999.

[Scam08] IST Scampi Project, A Scalable Monitoring Platform for the Internet, Accessed 25th September 2008, http://www.ist-scampi.org/.

[Scho04] Schormans, J.A., Pitts, J.M., So You Think You Can Measure IP QoS?, IEE Telecommunications Quality of Services: The Business of Success, 2004. QoS 2004, 2-3 March 2004, pp. 151-155.

[Soco91] Socolofsky, T., Kale, C., IETF Network Working Group, Request For Comments RFC1180: A TCP/IP Tutorial, January 1991.

[Salt81] Saltzer, J. H., Reed, D.P., Clark, D.G., End-to-End Arguments in System Design, ACM Transactions on Computer Systems, vol.2, no. 4, November 1984.

[Shen97] Shenker, S., Specification of Guaranteed Quality of Service, IETF Network Working Group, Request For Comments RFC 2212, September 1997.

[Smar00] Smart, M., Malan, R., Jahanian, F., Defeating TCP/IP Stack Fingerprinting, In Proceedings of 9th USENIX Security Symposium, 2000, pp. 229-240.

[Somm07] Sommers, J., Barford, P., Duffield, N., Ron, A., Accurate and Efficient SLA Compliance Monitoring, ACM SIGCOMM Computer Communication Review, vol.37, issue 4, October 2007, pp. 109-120.

[Spir06] S. Spirou, Packet Reordering Effects on the Subjective Quality of Broadband Digital Television, IEEE 10th International Symposium on Consumer Electronics, ISCE '06, 2006, pp. 1-6.

[Spri03] Spring, N., Robust Explicit Congestion Notification (ECN) Signaling with Nonces, IETF Network Working Group, Request For Comments RFC3540, June 2003.

[Sris01] Srisuresh, P., Traditional IP Network Address Translator (Traditional NAT), IETF Network Working Group, Request For Comments RFC3022, January 2001.

[Stev94] Stevens, W. R., TCP/IP Illustrated - Volume 1 - The Protocols, Addison Wesley, 1994, ISBN 0201633469.

[Tcpd08] Lawrence Berkeley National Laboratory, Network Research Group, Accessed 25th September 2008, TCPdump/LibPcap programs, http://www.tcpdump.org/

[Thom02] Thomas, T. M., Pavlichek, D., Dwyer, L. H., Chowbay, R., Downing, W. W., Sonderegger, J., Juniper Networks Reference Guide: JUNOS Routing, Configuration and Architecture, Addison-Wesley Professional, October 27, 2002, ISBN 0201775921.

[Tiru05] Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K., The IPerf Project, Accessed 25th September 2008, http://dast.nlanr.net/Projects/Iperf/

[Tsan08] Tsang, K., Wang, C., Lau, F.C.M., Handoff Performance Comparison of Mobile IP, Fast Handoff and mSCTP in Mobile Wireless Networks, in Proceedings of . International Symposium on Parallel Architectures, Algorithms, and Networks, 2008. I-SPAN 2008, pp.45-52, 7-9 May 2008

[Veal05] Veal, B., Kang, L, Lowenthal, D., New Methods for Passive Estimation of TCP Round-Trip Times, In Proceedings of Passive and Active Measurement, Lecture Notes in Computer Science 3431, Springer-Verlag, March 2005.

[Vqeg08] Video Quality Expert Group, Final report on the Validation of Objective Models of Multimedia Quality Assessment, Phase 1, Accessed 21st December 2008, http://www.its.bldrdoc.gov/vqeg/ .

[Wadd02] Waddington, D.G., Fangzhe Chang, Realizing The Transition to IPv6, IEEE Communications Magazine, vol.40, no.6, Jun. 2002, pp.138-147.

[Wang04] Wang, Y., Guohan, L., Xing, L., A Study of Internet Packet Reordering, In Proceedings of International Conference on Information Networking (ICOIN2004), Lecture Notes in Computer Science 3090, Springer-Verlag, 2004, pp. 350-359.

[Wee02] Wee, S., Tan, W-T, Apostolopoulos, J., Etoh, M.,"Optimised Video Streaming For Networks With Varying Delay," in Proceedings of the IEEE International Conference on Multimedia and Expo 2002, Lausanne, Switzerland, August 2002.

[Wenw07] Wenwei, L., Dafang, Z., Jinmin, Y., Gaogang, X., On Evaluating the Differences of TCP and ICMP in Network Measurement, ACM Computer Communications, vol.30, no.2, January 2007.

[West06] West, M., McCann, S., TCP/IP Field Behaviour, IETF Network Working Group, Request For Comments RFC4413, March 2006.

[Wood08] Wood, P., A Libpcap Version Which Supports MMAP Mode on Linux Kernels 2.[46], Accessed 25th September 2008, http://public.lanl.gov/cpw/

[Xylo99] Xylomenos, G., Polyzos, G. C., Internet Protocol Performance Over Networks With Wireless Links, IEEE Network, vol.13, issue 4, July - August 1999, pp. 55-63.

[Yan04] Yan, H., Li, K., Watterson, S., Lowenthal, D., Improving Passive Estimation of TCP Round-Trip Times Using TCP Timestamps, In Proceedings of IEEE Workshop on IP Operations and Management, 11-13 Oct. 2004, pp. 181-185.

[Ye06] Ye, Y., Jayasumana, A.P., Piratla, N.M., On Monitoring of End-to-End Packet Reordering Over the Internet, International conference on Networking and Services, ICNS '06., 2006.

[Zhan03] Zhang, M., Karp, B., Floyd, S., Peterson, L., RR-TCP: A Reordering-Robust TCP with DSACK, In Proceedings of the 11th IEEE International Conference on Networking Protocols (ICNP 2003), Atlanta, GA, November 2003.

[Zhou04] Zhou, X., Mieghem, R. V., Reordering of IP Packets in Internet, In Proceedings of Passive and Active Measurement, Lecture Notes in Computer Science 3015, Springer-Verlag, April 2004, pp. 237-246.

Appendix

Figure 29, Example Packet Capture Output, extended to illustrate the first 750 packets of a TCP Flow, thus illustrating the growth of *cwnd*.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1218632117	649377	10 0.0.2	4094	10.0 0 6	5720	0	0	1677880416	0	22492	0	SYN	5840
2	1218632117	649441	10 0.0 6	5720	10.0.0 2	4094	0	0	2165394042	1677880416	0	0	SYN	5792
3	1218632117	799834	10 0.0.2	4094	10.0.0 6	5720	1	1	1677880416	2165394042	22493	0		5840
4	1218632118	18821	10.0.0.2	4094	10.0 0.6	5720	2	2	1677880416	2165394042	22494	1448		5840
5	1218632118	19273	10.0.0.2	4094	10.0.0 6	5720	3	3	1677881864	2165394042	22495	1448		5840
6	1218632118	106245	10 0.0.6	5720	10 0.0 2	4094	28727	1	2165394042	1677881864	28727	0		8688
7	1218632118	106277	10 0 0.6	5720	10.0 0 2	4094	1	2	2165394042	1677883312	28728	0		11584
8	1218632118	256771	10.0.0 2	4094	10.0.0.6	5720	4	4	1677883312	2165394042	22496	1448		5840
9	1218632118	257169	10.0.0 2	4094	10.0 0 6	5720	6	5	1677886208	2165394042	22498	1448		5840
10	1218632118	257170	10.0 0 2	4094	10 0 0.6	5720	7	6	1677887656	2165394042	22499	1448		5840
11	1218632118	334777	10.0.0.2	4094	10.0 0 6	5720	5	7	1677884760	2165394042	22497	1448		5840
12	1218632118	346230	10.0.0 6	5720	10.0 0 2	4094	2	3	2165394042	1677884760	28729	0		14480
13	1218632118	346257	10 0 0 6	5720	10.0.0.2	4094	3	4	2165394042	1677884760	28730	0		14480
14	1218632118	346270	10.0.0 6	5720	10 0 0.2	4094	4	5	2165394042	1677884760	28731	0		14480
15	1218632118	346300	10.0 0.6	5720	10.0.0.2	4094	5	6	2165394042	1677889104	28732	0		17376
16	1218632118	496600	10 0.0.2	4094	10 0 0.6	5720	8	8	1677889104	2165394042	22500	1448		5840
17	1218632118	496965	10.0.0.2	4094	10 0.0 6	5720	9	9	1677890552	2165394042	22501	1448		5840
18	1218632118	496967	10 0.0.2	4094	10.0.0 6	5720	10	10	1677884760	2165394042	22502	1448		5840
19	1218632118	497365	10 0.0 2	4094	10.0.0 6	5720	11	11	1677892000	2165394042	22503	1448		5840
20	1218632118	586223	10 0.0.6	5720	10.0 0.2	4094	6	7	2165394042	1677890552	28733	0		20272
21	1218632118	586252	10.0.0 6	5720	10 0.0.2	4094	7	8	2165394042	1677892000	28734	0		23168
22	1218632118	586266	10 0.0 6	5720	10 0.0 2	4094	8	9	2165394042	1677892000	28735	0		23168
23	1218632118	586290	10.0.0 6	5720	10.0.0.2	4094	9	10	2165394042	1677893448	28736	0		26064
24	1218632118	736775	10.0.0 2	4094	10 0.0 6	5720	12	12	1677893448	2165394042	22504	1448		5840
25	1218632118	737158	10.0 0.2	4094	10.0.0 6	5720	13	13	1677894896	2165394042	22505	1448		5840
26	1218632118	737160	10 0 0.2	4094	10.0 0.6	5720	15	14	1677897792	2165394042	22507	1448		5840
27	1218632118	737162	10 0.0.2	4094	10.0.0 6	5720	16	15	1677899240	2165394042	22508	1448		5840
28	1218632118	814781	10 0.0.2	4094	10.0.0.6	5720	14	16	1677896344	2165394042	22506	1448		5840
29	1218632118	826217	10 0.0.6	5720	10 0.0.2	4094	10	11	2165394042	1677894896	28737	0		28960
30	1218632118	826245	10 0.0 6	5720	10 0 0.2	4094	11	12	2165394042	1677896344	28738	0		31856
31	1218632118	826262	10.0.0 6	5720	10.0.0.2	4094	12	13	2165394042	1677896344	28739	0		31856
32	1218632118	826274	10.0.0.6	5720	10.0.0 2	4094	13	14	2165394042	1677896344	28740	0		31856
33	1218632118	826320	10 0.0 6	5720	10 0.0 2	4094	14	15	2165394042	1677900688	28741	0		34752
34	1218632118	976731	10 0.0.2	4094	10.0.0 6	5720	17	17	1677900688	2165394042	22509	1448		5840
35	1218632118	977055	10 0.0 2	4094	10.0 0.6	5720	18	18	1677902136	2165394042	22510	1448		5840
36	1218632118	977056	10.0 0 2	4094	10.0.0 6	5720	19	19	1677903584	2165394042	22511	1448		5840
37	1218632118	977058	10.0.0.2	4094	10.0 0.6	5720	21	20	1677906480	2165394042	22513	1448		5840
38	1218632118	977454	10.0 0.2	4094	10.0 0.6	5720	23	21	1677909376	2165394042	22515	1448		5840
39	1218632119	55079	10.0.0.2	4094	10.0 0.6	5720	20	22	1677905032	2165394042	22512	1448		5840
40	1218632119	55453	10.0 0.2	4094	10.0.0.6	5720	22	23	1677907928	2165394042	22514	1448		5840
41	1218632119	55455	10 0.0.2	4094	10 0 0 6	5720	24	24	1677910824	2165394042	22516	1448		5840
42	1218632119	66215	10.0 0 6	5720	10.0.0.2	4094	15	16	2165394042	1677902136	28742	0		37648
43	1218632119	66245	10 0 0 6	5720	10.0 0 2	4094	16	17	2165394042	1677903584	28743	0		40544
44	1218632119	66268	10.0.0.6	5720	10.0 0 2	4094	17	18	2165394042	1677905032	28744	0		43440
45	1218632119	66285	10 0.0.6	5720	10 0.0.2	4094	18	19	2165394042	1677905032	28745	0		43440
46	1218632119	66298	10.0 0 6	5720	10.0.0.2	4094	19	20	2165394042	1677905032	28746	0		43440
47	1218632119	66327	10 0 0 6	5720	10.0 0 2	4094	20	21	2165394042	1677907928	28747	0		46336
48	1218632119	66340	10 0 0.6	5720	10.0 0 2	4094	21	22	2165394042	1677910824	28748	0		49232
49	1218632119	66353	10 0.0.6	5720	10.0 0.2	4094	22	23	2165394042	1677912272	28749	0		52128
50	1218632119	216798	10 0 0 2	4094	10 0 0 6	5720	25	25	1677912272	2165394042	22517	1448		5840
51	1218632119	216807	10.0.0 2	4094	10.0.0.6	5720	26	26	1677913720	2165394042	22518	1448		5840
52	1218632119	217149	10 0.0 2	4094	10 0 0 6	5720	27	27	1677915168	2165394042	22519	1448		5840
53	1218632119	217151	10 0 0.2	4094	10.0 0.6	5720	28	28	1677916616	2165394042	22520	1448		5840
54	1218632119	217152	10 0.0.2	4094	10 0 0.6	5720	29	29	1677918064	2165394042	22521	1448		5840
55	1218632119	217154	10.0 0 2	4094	10 0.0 6	5720	30	30	1677919512	2165394042	22522	1448		5840
56	1218632119	217156	10.0 0 2	4094	10 0.0 6	5720	31	31	1677920960	2165394042	22523	1448		5840
57	1218632119	217158	10.0.0 2	4094	10.0.0 6	5720	32	32	1677905032	2165394042	22524	1448		5840
58	1218632119	217549	10 0 0 2	4094	10 0 0 6	5720	33	33	1677922408	2165394042	22525	1448		5840
59	1218632119	217551	10.0 0.2	4094	10 0 0.6	5720	34	34	1677923856	2165394042	22526	1448		5840
60	1218632119	306211	10.0.0 6	5720	10 0.0 2	4094	23	24	2165394042	1677913720	28750	0		55024
61	1218632119	306241	10 0 0 6	5720	10 0 0.2	4094	24	25	2165394042	1677915168	28751	0		57920
62	1218632119	306264	10.0 0.6	5720	10 0 0.2	4094	25	26	2165394042	1677916616	28752	0		60816
63	1218632119	306287	10.0 0.6	5720	10.0 0 2	4094	26	27	2165394042	1677918064	28753	0		63712
64	1218632119	306311	10 0.0 6	5720	10 0.0.2	4094	27	28	2165394042	1677919512	28754	0		65160
65	1218632119	306335	10.0.0.6	5720	10.0.0.2	4094	28	29	2165394042	1677920960	28755	0		65160
66	1218632119	306347	10.0 0 6	5720	10 0.0.2	4094	29	30	2165394042	1677922408	28756	0		65160
67	1218632119	306360	10.0.0 6	5720	10 0.0.2	4094	30	31	2165394042	1677922408	28757	0		65160
68	1218632119	306373	10 0 0 6	5720	10 0.0.2	4094	31	32	2165394042	1677923856	28758	0		65160
69	1218632119	306384	10 0.0 6	5720	10.0.0.2	4094	32	33	2165394042	1677925304	28759	0		65160
70	1218632119	456629	10 0 0 2	4094	10 0.0.6	5720	35	35	1677925304	2165394042	22527	1448		5840
71	1218632119	457044	10.0.0.2	4094	10 0 0 6	5720	36	36	1677926752	2165394042	22528	1448		5840
72	1218632119	457046	10 0.0.2	4094	10.0.0 6	5720	38	37	1677929648	2165394042	22530	1448		5840
73	1218632119	457444	10 0.0.2	4094	10.0.0 6	5720	39	38	1677931096	2165394042	22531	1448		5840
74	1218632119	457446	10.0.0 2	4094	10.0.0 6	5720	40	39	1677932544	2165394042	22532	1448		5840
75	1218632119	457448	10.0.0.2	4094	10.0.0 6	5720	41	40	1677933992	2165394042	22533	1448		5840
76	1218632119	457450	10.0 0.2	4094	10.0 0 6	5720	42	41	1677935440	2165394042	22534	1448		5840
77	1218632119	457452	10.0.0.2	4094	10.0.0 6	5720	43	42	1677936888	2165394042	22535	1448		5840
78	1218632119	457454	10 0.0.2	4094	10.0.0 6	5720	44	43	1677938336	2165394042	22536	1448		5840
79	1218632119	457456	10.0 0.2	4094	10.0.0 6	5720	45	44	1677939784	2165394042	22537	1448		5840
80	1218632119	457844	10.0.0.2	4094	10.0.0 6	5720	46	45	1677941232	2165394042	22538	1448		5840
81	1218632119	534869	10.0.0.2	4094	10 0.0 6	5720	37	46	1677928200	2165394042	22529	1448		5840
82	1218632119	546202	10.0.0 6	5720	10.0.0 2	4094	33	34	2165394042	1677926752	28760	0		65160
83	1218632119	546231	10.0 0 6	5720	10 0 0 2	4094	34	35	2165394042	1677928200	28761	0		65160
84	1218632119	546248	10.0 0.6	5720	10.0.0.2	4094	35	36	2165394042	1677928200	28762	0		65160
85	1218632119	546260	10 0.0 6	5720	10.0.0 2	4094	36	37	2165394042	1677928200	28763	0		65160
86	1218632119	546270	10.0.0.6	5720	10 0.0.2	4094	37	38	2165394042	1677928200	28764	0		65160
87	1218632119	546281	10.0 0 6	5720	10.0.0.2	4094	38	39	2165394042	1677928200	28765	0		65160
88	1218632119	546292	10.0.0 6	5720	10.0.0.2	4094	39	40	2165394042	1677928200	28766	0		65160

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
89	1218632119	546303	10.0 0.6	5720	10 0 0.2	4094	40	41	2165394042	1677928200	28767	0		65160
90	1218632119	546314	10.0.0 6	5720	10.0.0 2	4094	41	42	2165394042	1677928200	28768	0		65160
91	1218632119	546324	10.0.0 6	5720	10.0 0 2	4094	42	43	2165394042	1677928200	28769	0		65160
92	1218632119	546335	10.0.0 6	5720	10 0 0 2	4094	43	44	2165394042	1677928200	28770	0		65160
93	1218632119	546363	10 0 0.6	5720	10 0 0 2	4094	44	45	2165394042	1677942680	28771	0		65160
94	1218632119	696601	10.0 0.2	4094	10.0 0 6	5720	47	47	1677942680	2165394042	22539	1448		5840
95	1218632119	696939	10 0.0.2	4094	10 0 0 6	5720	48	48	1677944128	2165394042	22540	1448		5840
96	1218632119	696941	10.0.0 2	4094	10.0 0 6	5720	49	49	1677945576	2165394042	22541	1448		5840
97	1218632119	696942	10.0.0 2	4094	10.0.0 6	5720	50	50	1677947024	2165394042	22542	1448		5840
98	1218632119	697338	10.0 0.2	4094	10 0 0 6	5720	51	51	1677948472	2165394042	22543	1448		5840
99	1218632119	697339	10 0 0 2	4094	10 0 0 6	5720	52	52	1677949920	2165394042	22544	1448		5840
100	1218632119	697341	10.0 0 2	4094	10 0.0 6	5720	53	53	1677951368	2165394042	22545	1448		5840
101	1218632119	697343	10.0 0.2	4094	10 0 0 6	5720	54	54	1677928200	2165394042	22546	1448		5840
102	1218632119	697345	10.0.0.2	4094	10 0 0 6	5720	55	55	1677952816	2165394042	22547	1448		5840
103	1218632119	697738	10 0 0 2	4094	10.0.0.6	5720	56	56	1677954264	2165394042	22548	1448		5840
104	1218632119	697740	10 0.0 2	4094	10 0.0 6	5720	57	57	1677955712	2165394042	22549	1448		5840
105	1218632119	786196	10.0 0 6	5720	10.0.0.2	4094	45	46	2165394042	1677944128	28772	0		65160
106	1218632119	786226	10.0 0.6	5720	10 0 0.2	4094	46	47	2165394042	1677945576	28773	0		65160
107	1218632119	786248	10.0 0.6	5720	10.0.0.2	4094	47	48	2165394042	1677947024	28774	0		65160
108	1218632119	786271	10.0.0.6	5720	10.0.0.2	4094	48	49	2165394042	1677948472	28775	0		65160
109	1218632119	786294	10 0 0.6	5720	10.0.0 2	4094	49	50	2165394042	1677949920	28776	0		65160
110	1218632119	786317	10 0.0 6	5720	10.0.0 2	4094	50	51	2165394042	1677951368	28777	0		65160
111	1218632119	786328	10.0.0.6	5720	10.0.0.2	4094	51	52	2165394042	1677952816	28778	0		65160
112	1218632119	786341	10 0.0 6	5720	10.0 0 2	4094	52	53	2165394042	1677952816	28779	0		65160
113	1218632119	786355	10.0 0 6	5720	10.0.0 2	4094	53	54	2165394042	1677954264	28780	0		65160
114	1218632119	786367	10.0 0.6	5720	10.0.0.2	4094	54	55	2165394042	1677955712	28781	0		65160
115	1218632119	786378	10.0.0.6	5720	10.0.0.2	4094	55	56	2165394042	1677957160	28782	0		65160
116	1218632119	936656	10.0.0.2	4094	10 0.0.6	5720	58	58	1677957160	2165394042	22550	1448		5840
117	1218632119	937033	10 0.0 2	4094	10 0 0.6	5720	59	59	1677958608	2165394042	22551	1448		5840
118	1218632119	937035	10.0.0.2	4094	10 0 0.6	5720	60	60	1677960056	2165394042	22552	1448		5840
119	1218632119	937037	10 0.0.2	4094	10 0 0.6	5720	61	61	1677961504	2165394042	22553	1448		5840
120	1218632119	937039	10.0.0 2	4094	10 0.0 6	5720	62	62	1677962952	2165394042	22554	1448		5840
121	1218632119	937433	10.0.0.2	4094	10.0.0 6	5720	63	63	1677964400	2165394042	22555	1448		5840
122	1218632119	937435	10 0.0 2	4094	10.0.0.6	5720	64	64	1677965848	2165394042	22556	1448		5840
123	1218632119	937437	10.0 0.2	4094	10.0.0.6	5720	65	65	1677967296	2165394042	22557	1448		5840
124	1218632119	937438	10 0.0.2	4094	10 0.0 6	5720	66	66	1677968744	2165394042	22558	1448		5840
125	1218632119	937527	10.0 0.2	4094	10.0.0 6	5720	67	67	1677970192	2165394042	22559	1448		5840
126	1218632119	937529	10 0.0.2	4094	10.0.0 6	5720	68	68	1677971640	2165394042	22560	1448		5840
127	1218632119	937531	10 0.0.2	4094	10.0 0 6	5720	69	69	1677973088	2165394042	22561	1448		5840
128	1218632119	937935	10 0 0.2	4094	10.0.0 6	5720	70	70	1677974536	2165394042	22562	1448		5840
129	1218632119	937937	10.0 0.2	4094	10 0.0.6	5720	71	71	1677975984	2165394042	22563	1448		5840
130	1218632119	937939	10.0 0.2	4094	10 0.0.6	5720	72	72	1677977432	2165394042	22564	1448		5840
131	1218632120	26191	10.0 0 6	5720	10 0.0.2	4094	56	57	2165394042	1677958608	28783	0		65160
132	1218632120	26219	10.0.0 6	5720	10.0.0.2	4094	57	58	2165394042	1677960056	28784	0		65160
133	1218632120	26242	10.0 0 6	5720	10 0.0 2	4094	58	59	2165394042	1677961504	28785	0		65160
134	1218632120	26265	10.0.0.6	5720	10.0.0.2	4094	59	60	2165394042	1677962952	28786	0		65160
135	1218632120	26288	10.0.0.6	5720	10.0.0 2	4094	60	61	2165394042	1677964400	28787	0		65160
136	1218632120	26309	10.0.0.6	5720	10.0.0.2	4094	61	62	2165394042	1677965848	28788	0		65160
137	1218632120	26321	10 0.0 6	5720	10.0.0.2	4094	62	63	2165394042	1677967296	28789	0		65160
138	1218632120	26333	10 0 0 6	5720	10 0.0.2	4094	63	64	2165394042	1677968744	28790	0		65160
139	1218632120	26344	10 0.0 6	5720	10.0 0.2	4094	64	65	2165394042	1677970192	28791	0		65160
140	1218632120	26356	10 0.0 6	5720	10.0.0.2	4094	65	66	2165394042	1677971640	28792	0		65160
141	1218632120	26368	10 0.0 6	5720	10 0.0.2	4094	66	67	2165394042	1677973088	28793	0		65160
142	1218632120	26379	10 0 0.6	5720	10.0.0.2	4094	67	68	2165394042	1677974536	28794	0		65160
143	1218632120	26393	10 0 0 6	5720	10 0 0.2	4094	68	69	2165394042	1677977432	28795	0		65160
144	1218632120	62742	10.0 0 6	5720	10 0.0.2	4094	69	70	2165394042	1677978880	28796	0		65160
145	1218632120	176632	10.0 0 2	4094	10.0 0.6	5720	73	73	1677978880	2165394042	22565	1448		5840
146	1218632120	177029	10 0 0 2	4094	10 0.0 6	5720	74	74	1677980328	2165394042	22566	1448		5840
147	1218632120	177030	10 0 0 2	4094	10 0 0 6	5720	75	75	1677981776	2165394042	22567	1448		5840
148	1218632120	177032	10 0.0 2	4094	10 0 0.6	5720	76	76	1677983224	2165394042	22568	1448		5840
149	1218632120	177034	10 0.0 2	4094	10 0 0.6	5720	77	77	1677984672	2165394042	22569	1448		5840
150	1218632120	177036	10.0.0.2	4094	10.0.0 6	5720	78	78	1677986120	2165394042	22570	1448		5840
151	1218632120	177038	10 0 0 2	4094	10.0.0 6	5720	79	79	1677987568	2165394042	22571	1448		5840
152	1218632120	177429	10.0.0.2	4094	10 0 0 6	5720	80	80	1677989016	2165394042	22572	1448		5840
153	1218632120	177431	10.0 0.2	4094	10.0.0.6	5720	81	81	1677990464	2165394042	22573	1448		5840
154	1218632120	177433	10 0 0.2	4094	10 0.0.6	5720	82	82	1677991912	2165394042	22574	1448		5840
155	1218632120	177434	10 0 0 2	4094	10.0 0 6	5720	84	83	1677994808	2165394042	22576	1448		5840
156	1218632120	177436	10 0.0.2	4094	10.0.0 6	5720	85	84	1677996256	2165394042	22577	1448		5840
157	1218632120	177438	10.0 0.2	4094	10.0.0.6	5720	86	85	1677997704	2165394042	22578	1448		5840
158	1218632120	177440	10.0.0.2	4094	10.0.0.6	5720	87	86	1677999152	2165394042	22579	1448		5840
159	1218632120	177828	10.0.0 2	4094	10.0.0 6	5720	88	87	1678000600	2165394042	22580	1448		5840
160	1218632120	177830	10.0.0 2	4094	10 0.0.6	5720	89	88	1678002048	2165394042	22581	1448		5840
161	1218632120	177832	10.0 0 2	4094	10.0.0.6	5720	90	89	1678003496	2165394042	22582	1448		5840
162	1218632120	177833	10.0 0 2	4094	10.0.0.6	5720	91	90	1678004944	2165394042	22583	1448		5840
163	1218632120	177835	10.0 0 2	4094	10.0 0 6	5720	93	91	1678007840	2165394042	22585	1448		5840
164	1218632120	177837	10.0.0.2	4094	10.0.0 6	5720	94	92	1678009288	2165394042	22586	1448		5840
165	1218632120	177839	10.0.0.2	4094	10.0.0.6	5720	95	93	1678010736	2165394042	22587	1448		5840
166	1218632120	177841	10.0 0.2	4094	10 0.0 6	5720	96	94	1678012184	2165394042	22588	1448		5840
167	1218632120	178229	10.0.0.2	4094	10.0.0.6	5720	97	95	1678013632	2165394042	22589	1448		5840
168	1218632120	178232	10 0.0.2	4094	10 0 0 6	5720	98	96	1678015080	2165394042	22590	1448		5840
169	1218632120	178233	10.0.0.2	4094	10.0.0.6	5720	99	97	1678016528	2165394042	22591	1448		5840
170	1218632120	213222	10.0 0.2	4094	10.0.0.6	5720	100	98	1678017976	2165394042	22592	1448		5840
171	1218632120	213629	10.0.0.2	4094	10.0.0.6	5720	101	99	1678019424	2165394042	22593	1448		5840
172	1218632120	255385	10.0.0.2	4094	10.0.0.6	5720	83	100	1677993360	2165394042	22575	1448		5840
173	1218632120	255727	10.0.0.2	4094	10.0.0.6	5720	92	101	1678006392	2165394042	22584	1448		5840
174	1218632120	266257	10 0.0.6	5720	10.0.0.2	4094	70	71	2165394042	1677987568	28797	0		65160
175	1218632120	266275	10.0 0.6	5720	10.0.0.2	4094	71	72	2165394042	1677990464	28798	0		65160
176	1218632120	266289	10 0.0 6	5720	10.0.0.2	4094	72	73	2165394042	1677993360	28799	0		65160
177	1218632120	266303	10.0.0.6	5720	10.0.0.2	4094	73	74	2165394042	1677993360	28800	0		65160
178	1218632120	266315	10 0.0.6	5720	10.0 0.2	4094	74	75	2165394042	1677993360	28801	0		65160
179	1218632120	266326	10 0.0 6	5720	10.0.0.2	4094	75	76	2165394042	1677993360	28802	0		65160
180	1218632120	266337	10.0.0.6	5720	10.0.0.2	4094	76	77	2165394042	1677993360	28803	0		65160
181	1218632120	266348	10.0.0.6	5720	10.0.0 2	4094	77	78	2165394042	1677993360	28804	0		65160
182	1218632120	266359	10.0.0 6	5720	10.0.0.2	4094	78	79	2165394042	1677993360	28805	0		65160
183	1218632120	266370	10 0.0.6	5720	10.0 0 2	4094	79	80	2165394042	1677993360	28806	0		65160
184	1218632120	266381	10.0.0.6	5720	10 0.0.2	4094	80	81	2165394042	1677993360	28807	0		65160

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
185	1218632120	266394	10 0 0.6	5720	10.0 0.2	4094	81	82	2165394042	1677993360	28808	0		65160
186	1218632120	266406	10 0 0 6	5720	10 0 0.2	4094	82	83	2165394042	1677993360	28809	0		65160
187	1218632120	266416	10.0 0.6	5720	10 0.0.2	4094	83	84	2165394042	1677993360	28810	0		65160
188	1218632120	266427	10 0 0.6	5720	10.0.0.2	4094	84	85	2165394042	1677993360	28811	0		65160
189	1218632120	266437	10 0.0.6	5720	10.0.0.2	4094	85	86	2165394042	1677993360	28812	0		65160
190	1218632120	266448	10 0.0.6	5720	10 0.0.2	4094	86	87	2165394042	1677993360	28813	0		65160
191	1218632120	266459	10.0 0 6	5720	10.0 0.2	4094	87	88	2165394042	1677993360	28814	0		65160
192	1218632120	266470	10 0.0 6	5720	10.0 0.2	4094	88	89	2165394042	1677993360	28815	0		65160
193	1218632120	266481	10 0.0 6	5720	10.0.0.2	4094	89	90	2165394042	1677993360	28816	0		65160
194	1218632120	266496	10.0.0.6	5720	10.0 0.2	4094	90	91	2165394042	1678006392	28817	0		65160
195	1218632120	266509	10 0.0 6	5720	10.0 0.2	4094	91	92	2165394042	1678020872	28818	0		65160
196	1218632120	416704	10.0.0.2	4094	10.0.0.6	5720	102	102	1678020872	2165394042	22594	1448		5840
197	1218632120	417124	10.0.0.2	4094	10.0 0.6	5720	103	103	1678022320	2165394042	22595	1448		5840
198	1218632120	417126	10.0 0.2	4094	10.0.0.6	5720	104	104	1678023768	2165394042	22596	1448		5840
199	1218632120	417128	10 0.0.2	4094	10.0.0.6	5720	105	105	1678025216	2165394042	22597	1448		5840
200	1218632120	417130	10.0.0.2	4094	10.0.0.6	5720	106	106	1678026664	2165394042	22598	1448		5840
201	1218632120	417131	10.0.0.2	4094	10.0.0.6	5720	107	107	1678028112	2165394042	22599	1448		5840
202	1218632120	417133	10.0.0 2	4094	10.0 0 6	5720	108	108	1678029560	2165394042	22600	1448		5840
203	1218632120	417135	10 0.0 2	4094	10.0 0 6	5720	110	109	1678032456	2165394042	22602	1448		5840
204	1218632120	417525	10 0.0 2	4094	10.0.0.6	5720	111	110	1678033904	2165394042	22603	1448		5840
205	1218632120	417527	10 0.0 2	4094	10.0.0.6	5720	112	111	1678035352	2165394042	22604	1448		5840
206	1218632120	417528	10 0.0.2	4094	10.0.0.6	5720	113	112	1678036800	2165394042	22605	1448		5840
207	1218632120	417530	10.0 0.2	4094	10.0 0.6	5720	114	113	1678038248	2165394042	22606	1448		5840
208	1218632120	417532	10.0 0.2	4094	10.0.0.6	5720	115	114	1678039696	2165394042	22607	1448		5840
209	1218632120	417534	10 0 0.2	4094	10 0.0.6	5720	116	115	1678041144	2165394042	22608	1448		5840
210	1218632120	417536	10 0.0.2	4094	10.0.0 6	5720	117	116	1678042592	2165394042	22609	1448		5840
211	1218632120	417538	10 0.0.2	4094	10.0.0 6	5720	118	117	1678044040	2165394042	22610	1448		5840
212	1218632120	417924	10.0.0.2	4094	10.0.0 6	5720	119	118	1678045488	2165394042	22611	1448		5840
213	1218632120	417927	10.0.0.2	4094	10.0.0 6	5720	120	119	1678046936	2165394042	22612	1448		5840
214	1218632120	417929	10.0.0.2	4094	10.0 0.6	5720	121	120	1678048384	2165394042	22613	1448		5840
215	1218632120	417930	10.0.0.2	4094	10 0 0 6	5720	122	121	1678049832	2165394042	22614	1448		5840
216	1218632120	417932	10.0.0.2	4094	10 0 0.6	5720	123	122	1677993360	2165394042	22615	1448		5840
217	1218632120	418323	10 0.0.2	4094	10 0.0 6	5720	124	123	1678051280	2165394042	22616	1448		5840
218	1218632120	418324	10.0.0.2	4094	10.0.0.6	5720	125	124	1678052728	2165394042	22617	1448		5840
219	1218632120	418326	10.0.0.2	4094	10.0 0 6	5720	126	125	1678054176	2165394042	22618	1448		5840
220	1218632120	418723	10.0 0.2	4094	10.0.0 6	5720	127	126	1678055624	2165394042	22619	1448		5840
221	1218632120	418724	10 0.0.2	4094	10.0 0.6	5720	128	127	1678057072	2165394042	22620	1448		5840
222	1218632120	418726	10 0.0.2	4094	10.0.0 6	5720	129	128	1678058520	2165394042	22621	1448		5840
223	1218632120	495077	10.0.0.2	4094	10.0 0.6	5720	109	129	1678031008	2165394042	22601	1448		5840
224	1218632120	506183	10.0.0 6	5720	10.0.0.2	4094	92	93	2165394042	1678022320	28819	0		65160
225	1218632120	506214	10.0.0.6	5720	10.0 0.2	4094	93	94	2165394042	1678023768	28820	0		65160
226	1218632120	506237	10.0 0.6	5720	10.0 0.2	4094	94	95	2165394042	1678025216	28821	0		65160
227	1218632120	506260	10.0.0.6	5720	10.0 0.2	4094	95	96	2165394042	1678026664	28822	0		65160
228	1218632120	506284	10 0.0.6	5720	10.0.0.2	4094	96	97	2165394042	1678028112	28823	0		65160
229	1218632120	506308	10.0.0 6	5720	10.0.0.2	4094	97	98	2165394042	1678029560	28824	0		65160
230	1218632120	506320	10.0.0.6	5720	10.0.0.2	4094	98	99	2165394042	1678031008	28825	0		65160
231	1218632120	506335	10.0.0.6	5720	10.0.0.2	4094	99	100	2165394042	1678031008	28826	0		65160
232	1218632120	506346	10.0.0.6	5720	10.0.0.2	4094	100	101	2165394042	1678031008	28827	0		65160
233	1218632120	506358	10.0 0.6	5720	10.0.0.2	4094	101	102	2165394042	1678031008	28828	0		65160
234	1218632120	506369	10.0.0 6	5720	10.0.0.2	4094	102	103	2165394042	1678031008	28829	0		65160
235	1218632120	506381	10 0.0 6	5720	10 0 0 2	4094	103	104	2165394042	1678031008	28830	0		65160
236	1218632120	506391	10.0 0 6	5720	10.0.0.2	4094	104	105	2165394042	1678031008	28831	0		65160
237	1218632120	506403	10 0 0.6	5720	10.0 0.2	4094	105	106	2165394042	1678031008	28832	0		65160
238	1218632120	506414	10.0.0 6	5720	10.0 0.2	4094	106	107	2165394042	1678031008	28833	0		65160
239	1218632120	506425	10.0.0.6	5720	10.0 0.2	4094	107	108	2165394042	1678031008	28834	0		65160
240	1218632120	506436	10 0 0 6	5720	10 0.0.2	4094	108	109	2165394042	1678031008	28835	0		65160
241	1218632120	506447	10.0.0 6	5720	10 0 0.2	4094	109	110	2165394042	1678031008	28836	0		65160
242	1218632120	506460	10.0.0.6	5720	10.0.0 2	4094	110	111	2165394042	1678031008	28837	0		65160
243	1218632120	506472	10.0 0.6	5720	10 0 0.2	4094	111	112	2165394042	1678031008	28838	0		65160
244	1218632120	506483	10 0 0 6	5720	10 0.0 2	4094	112	113	2165394042	1678031008	28839	0		65160
245	1218632120	506496	10 0 0 6	5720	10.0.0.2	4094	113	114	2165394042	1678031008	28840	0		65160
246	1218632120	506507	10.0 0.6	5720	10 0 0.2	4094	114	115	2165394042	1678031008	28841	0		65160
247	1218632120	506518	10 0 0 6	5720	10 0.0.2	4094	115	116	2165394042	1678031008	28842	0		65160
248	1218632120	506529	10.0.0.6	5720	10.0.0 2	4094	116	117	2165394042	1678031008	28843	0		65160
249	1218632120	506541	10 0.0 6	5720	10.0.0.2	4094	117	118	2165394042	1678031008	28844	0		65160
250	1218632120	506552	10.0 0 6	5720	10 0.0.2	4094	118	119	2165394042	1678031008	28845	0		65160
251	1218632120	506568	10 0.0 6	5720	10 0.0 2	4094	119	120	2165394042	1678059968	28846	0		65160
252	1218632120	656660	10 0.0.2	4094	10.0 0 6	5720	130	130	1678059968	2165394042	22622	1448		5840
253	1218632120	657019	10.0.0.2	4094	10.0.0.6	5720	131	131	1678061416	2165394042	22623	1448		5840
254	1218632120	657021	10 0.0 2	4094	10 0.0.6	5720	132	132	1678062864	2165394042	22624	1448		5840
255	1218632120	657023	10.0.0.2	4094	10 0.0 6	5720	133	133	1678064312	2165394042	22625	1448		5840
256	1218632120	657418	10.0 0.2	4094	10.0.0.6	5720	134	134	1678065760	2165394042	22626	1448		5840
257	1218632120	657420	10 0.0 2	4094	10.0 0 6	5720	135	135	1678067208	2165394042	22627	1448		5840
258	1218632120	657818	10.0 0 2	4094	10.0 0 6	5720	137	136	1678031008	2165394042	22629	1448		5840
259	1218632120	657820	10 0.0.2	4094	10.0.0 6	5720	138	137	1678070104	2165394042	22630	1448		5840
260	1218632120	657822	10.0 0.2	4094	10.0 0.6	5720	139	138	1678071552	2165394042	22631	1448		5840
261	1218632120	657824	10.0.0 2	4094	10 0 0.6	5720	140	139	1678073000	2165394042	22632	1448		5840
262	1218632120	658218	10.0 0.2	4094	10.0 0 6	5720	141	140	1678074448	2165394042	22633	1448		5840
263	1218632120	658220	10.0 0.2	4094	10.0.0 6	5720	142	141	1678075896	2165394042	22634	1448		5840
264	1218632120	658222	10.0.0.2	4094	10.0.0 6	5720	143	142	1678077344	2165394042	22635	1448		5840
265	1218632120	658224	10.0 0.2	4094	10 0 0.6	5720	144	143	1678078792	2165394042	22636	1448		5840
266	1218632120	658619	10.0.0.2	4094	10.0.0 6	5720	145	144	1678080240	2165394042	22637	1448		5840
267	1218632120	658620	10.0.0.2	4094	10.0.0.6	5720	146	145	1678081688	2165394042	22638	1448		5840
268	1218632120	658622	10.0.0.2	4094	10 0.0.6	5720	147	146	1678083136	2165394042	22639	1448		5840
269	1218632120	658624	10.0.0.2	4094	10.0.0.6	5720	148	147	1678084584	2165394042	22640	1448		5840
270	1218632120	735375	10.0.0.2	4094	10 0.0.6	5720	136	148	1678068656	2165394042	22628	1448		5840
271	1218632120	746174	10.0 0.6	5720	10.0.0.2	4094	120	121	2165394042	1678061416	28847	0		65160
272	1218632120	746202	10 0.0.6	5720	10.0 0.2	4094	121	122	2165394042	1678062864	28848	0		65160
273	1218632120	746225	10.0.0.6	5720	10.0.0.2	4094	122	123	2165394042	1678064312	28849	0		65160
274	1218632120	746247	10.0.0.6	5720	10 0.0.2	4094	123	124	2165394042	1678065760	28850	0		65160
275	1218632120	746270	10.0.0.6	5720	10.0.0 2	4094	124	125	2165394042	1678067208	28851	0		65160
276	1218632120	746293	10 0.0 6	5720	10.0.0.2	4094	125	126	2165394042	1678068656	28852	0		65160
277	1218632120	746305	10.0.0.6	5720	10 0.0 2	4094	126	127	2165394042	1678068656	28853	0		65160
278	1218632120	746320	10.0.0 6	5720	10.0.0.2	4094	127	128	2165394042	1678068656	28854	0		65160
279	1218632120	746331	10.0.0 6	5720	10.0.0.2	4094	128	129	2165394042	1678068656	28855	0		65160
280	1218632120	746342	10 0.0.6	5720	10.0.0.2	4094	129	130	2165394042	1678068656	28856	0		65160

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
281	1218632120	746354	10.0.0 6	5720	10 0.0.2	4094	130	131	2165394042	1678068656	28857	0		65160
282	1218632120	746365	10.0 0 6	5720	10.0.0.2	4094	131	132	2165394042	1678068656	28858	0		65160
283	1218632120	746376	10 0 0 6	5720	10 0 0 2	4094	132	133	2165394042	1678068656	28859	0		65160
284	1218632120	746387	10.0.0 6	5720	10 0.0 2	4094	133	134	2165394042	1678068656	28860	0		65160
285	1218632120	746399	10.0 0 6	5720	10 0.0.2	4094	134	135	2165394042	1678068656	28861	0		65160
286	1218632120	746410	10.0 0 6	5720	10.0.0.2	4094	135	136	2165394042	1678068656	28862	0		65160
287	1218632120	746421	10 0 0.6	5720	10.0.0.2	4094	136	137	2165394042	1678068656	28863	0		65160
288	1218632120	746432	10 0.0 6	5720	10.0.0.2	4094	137	138	2165394042	1678068656	28864	0		65160
289	1218632120	746447	10.0 0 6	5720	10.0.0 2	4094	138	139	2165394042	1678086032	28865	0		65160
290	1218632120	896630	10.0.0.2	4094	10 0.0 6	5720	149	149	1678086032	2165394042	22641	1448		5840
291	1218632120	897014	10.0 0.2	4094	10.0.0 6	5720	150	150	1678087480	2165394042	22642	1448		5840
292	1218632120	897016	10.0 0.2	4094	10.0.0.6	5720	151	151	1678088928	2165394042	22643	1448		5840
293	1218632120	897017	10 0.0.2	4094	10.0.0.6	5720	152	152	1678090376	2165394042	22644	1448		5840
294	1218632120	897019	10.0.0.2	4094	10.0.0.6	5720	153	153	1678091824	2165394042	22645	1448		5840
295	1218632120	897021	10 0.0 2	4094	10.0.0 6	5720	154	154	1678093272	2165394042	22646	1448		5840
296	1218632120	897023	10 0 0 2	4094	10.0 0.6	5720	155	155	1678094720	2165394042	22647	1448		5840
297	1218632120	897413	10 0.0.2	4094	10.0.0.6	5720	156	156	1678096168	2165394042	22648	1448		5840
298	1218632120	897415	10 0 0.2	4094	10.0 0 6	5720	157	157	1678097616	2165394042	22649	1448		5840
299	1218632120	897416	10.0 0.2	4094	10.0.0 6	5720	158	158	1678099064	2165394042	22650	1448		5840
300	1218632120	897418	10.0.0.2	4094	10.0.0.6	5720	159	159	1678100512	2165394042	22651	1448		5840
301	1218632120	897420	10 0.0.2	4094	10 0.0 6	5720	160	160	1678101960	2165394042	22652	1448		5840
302	1218632120	897422	10.0.0.2	4094	10 0 0 6	5720	161	161	1678103408	2165394042	22653	1448		5840
303	1218632120	897815	10.0.0.2	4094	10 0 0.6	5720	162	162	1678104856	2165394042	22654	1448		5840
304	1218632120	897817	10 0 0.2	4094	10.0.0.6	5720	163	163	1678106304	2165394042	22655	1448		5840
305	1218632120	897819	10.0 0.2	4094	10.0.0 6	5720	165	164	1678109200	2165394042	22657	1448		5840
306	1218632120	897821	10.0 0.2	4094	10.0 0.6	5720	166	165	1678110648	2165394042	22658	1448		5840
307	1218632120	897822	10 0 0.2	4094	10.0 0.6	5720	167	166	1678112096	2165394042	22659	1448		5840
308	1218632120	897824	10.0 0.2	4094	10.0.0.6	5720	168	167	1678113544	2165394042	22660	1448		5840
309	1218632120	898213	10 0 0.2	4094	10.0 0 6	5720	169	168	1678114992	2165394042	22661	1448		5840
310	1218632120	898214	10.0 0 2	4094	10.0.0.6	5720	170	169	1678116440	2165394042	22662	1448		5840
311	1218632120	898216	10 0 0.2	4094	10.0.0.6	5720	171	170	1678117888	2165394042	22663	1448		5840
312	1218632120	898218	10 0 0.2	4094	10.0 0.6	5720	172	171	1678119336	2165394042	22664	1448		5840
313	1218632120	898220	10 0.0 2	4094	10.0 0.6	5720	173	172	1678120784	2165394042	22665	1448		5840
314	1218632120	975584	10.0.0.2	4094	10.0.0.6	5720	164	173	1678107752	2165394042	22656	1448		5840
315	1218632120	986171	10.0 0.6	5720	10 0.0.2	4094	139	140	2165394042	1678087480	28866	0		65160
316	1218632120	986201	10.0.0.6	5720	10.0 0.2	4094	140	141	2165394042	1678088928	28867	0		65160
317	1218632120	986224	10.0.0.6	5720	10.0 0.2	4094	141	142	2165394042	1678090376	28868	0		65160
318	1218632120	986264	10.0 0.6	5720	10.0 0.2	4094	142	143	2165394042	1678091824	28869	0		65160
319	1218632120	986289	10.0.0 6	5720	10.0.0 2	4094	143	144	2165394042	1678093272	28870	0		65160
320	1218632120	986312	10.0 0 6	5720	10.0 0 2	4094	144	145	2165394042	1678094720	28871	0		65160
321	1218632120	986325	10 0.0 6	5720	10.0 0 2	4094	145	146	2165394042	1678096168	28872	0		65160
322	1218632120	986337	10.0.0 6	5720	10 0.0 2	4094	146	147	2165394042	1678097616	28873	0		65160
323	1218632120	986348	10.0.0 6	5720	10.0.0 2	4094	147	148	2165394042	1678099064	28874	0		65160
324	1218632120	986360	10 0.0.6	5720	10.0 0.2	4094	148	149	2165394042	1678100512	28875	0		65160
325	1218632120	986371	10 0 0 6	5720	10 0.0.2	4094	149	150	2165394042	1678101960	28876	0		65160
326	1218632120	986382	10 0.0 6	5720	10.0.0.2	4094	150	151	2165394042	1678103408	28877	0		65160
327	1218632120	986393	10 0.0.6	5720	10 0 0.2	4094	151	152	2165394042	1678104856	28878	0		65160
328	1218632120	986404	10.0 0.6	5720	10.0.0.2	4094	152	153	2165394042	1678106304	28879	0		65160
329	1218632120	986422	10 0 0 6	5720	10.0 0.2	4094	153	154	2165394042	1678107752	28880	0		65160
330	1218632120	986434	10.0 0.6	5720	10.0.0 2	4094	154	155	2165394042	1678107752	28881	0		65160
331	1218632120	986446	10 0 0 6	5720	10.0.0 2	4094	155	156	2165394042	1678107752	28882	0		65160
332	1218632120	986456	10 0 0 6	5720	10.0.0.2	4094	156	157	2165394042	1678107752	28883	0		65160
333	1218632120	986467	10 0 0 6	5720	10.0.0 2	4094	157	158	2165394042	1678107752	28884	0		65160
334	1218632120	986478	10.0 0.6	5720	10 0 0.2	4094	158	159	2165394042	1678107752	28885	0		65160
335	1218632120	986489	10 0 0 6	5720	10.0 0 2	4094	159	160	2165394042	1678107752	28886	0		65160
336	1218632120	986500	10 0.0 6	5720	10.0 0 2	4094	160	161	2165394042	1678107752	28887	0		65160
337	1218632120	986511	10.0.0 6	5720	10.0 0 2	4094	161	162	2165394042	1678107752	28888	0		65160
338	1218632120	986526	10.0.0.6	5720	10.0 0.2	4094	162	163	2165394042	1678122232	28889	0		65160
339	1218632121	136587	10.0.0.2	4094	10 0 0.6	5720	174	174	1678122232	2165394042	22666	1448		5840
340	1218632121	136945	10.0 0 2	4094	10 0 0 6	5720	176	175	1678125128	2165394042	22668	1448		5840
341	1218632121	137310	10 0.0.2	4094	10 0 0 6	5720	177	176	1678126576	2165394042	22669	1448		5840
342	1218632121	137312	10 0.0.2	4094	10.0.0 6	5720	178	177	1678128024	2165394042	22670	1448		5840
343	1218632121	137314	10 0 0.2	4094	10.0.0.6	5720	179	178	1678129472	2165394042	22671	1448		5840
344	1218632121	137316	10.0.0.2	4094	10.0.0.6	5720	180	179	1678130920	2165394042	22672	1448		5840
345	1218632121	137318	10.0.0.2	4094	10.0.0.6	5720	181	180	1678132368	2165394042	22673	1448		5840
346	1218632121	137320	10 0 0.2	4094	10 0 0 6	5720	182	181	1678133816	2165394042	22674	1448		5840
347	1218632121	137322	10 0.0 2	4094	10.0 0 6	5720	183	182	1678135264	2165394042	22675	1448		5840
348	1218632121	137324	10.0.0.2	4094	10.0.0.6	5720	184	183	1678136712	2165394042	22676	1448		5840
349	1218632121	137708	10.0 0.2	4094	10 0 0 6	5720	185	184	1678138160	2165394042	22677	1448		5840
350	1218632121	137710	10 0 0.2	4094	10.0.0 6	5720	186	185	1678139608	2165394042	22678	1448		5840
351	1218632121	137712	10.0.0.2	4094	10 0.0.6	5720	187	186	1678141056	2165394042	22679	1448		5840
352	1218632121	137714	10.0.0.2	4094	10.0.0.6	5720	188	187	1678142504	2165394042	22680	1448		5840
353	1218632121	137716	10.0.0.2	4094	10.0.0 6	5720	189	188	1678143952	2165394042	22681	1448		5840
354	1218632121	137718	10.0.0.2	4094	10.0 0.6	5720	190	189	1678145400	2165394042	22682	1448		5840
355	1218632121	137720	10.0.0.2	4094	10.0.0.6	5720	191	190	1678146848	2165394042	22683	1448		5840
356	1218632121	137802	10.0.0.2	4094	10.0.0 6	5720	192	191	1678148296	2165394042	22684	1448		5840
357	1218632121	137805	10 0.0.2	4094	10.0.0.6	5720	193	192	1678149744	2165394042	22685	1448		5840
358	1218632121	138208	10.0 0.2	4094	10.0 0 6	5720	195	193	1678152640	2165394042	22687	1448		5840
359	1218632121	138210	10.0.0.2	4094	10.0.0.6	5720	196	194	1678154088	2165394042	22688	1448		5840
360	1218632121	138212	10.0.0.2	4094	10.0.0.6	5720	197	195	1678155536	2165394042	22689	1448		5840
361	1218632121	138214	10.0.0.2	4094	10.0.0.6	5720	198	196	1678156984	2165394042	22690	1448		5840
362	1218632121	138216	10.0.0.2	4094	10 0.0 6	5720	199	197	1678158432	2165394042	22691	1448		5840
363	1218632121	138218	10.0 0.2	4094	10.0.0 6	5720	200	198	1678159880	2165394042	22692	1448		5840
364	1218632121	138219	10 0.0.2	4094	10 0 0 6	5720	201	199	1678161328	2165394042	22693	1448		5840
365	1218632121	138610	10.0.0 2	4094	10.0.0.6	5720	202	200	1678162776	2165394042	22694	1448		5840
366	1218632121	138612	10.0.0.2	4094	10.0 0 6	5720	203	201	1678164224	2165394042	22695	1448		5840
367	1218632121	138614	10.0.0.2	4094	10.0.0.6	5720	204	202	1678165672	2165394042	22696	1448		5840
368	1218632121	138616	10 0.0.2	4094	10.0.0.6	5720	205	203	1678167120	2165394042	22697	1448		5840
369	1218632121	138618	10.0.0.2	4094	10.0.0 6	5720	206	204	1678168568	2165394042	22698	1448		5840
370	1218632121	138620	10.0.0.2	4094	10.0.0.6	5720	207	205	1678170016	2165394042	22699	1448		5840
371	1218632121	138622	10 0.0.2	4094	10.0.0.6	5720	208	206	1678171464	2165394042	22700	1448		5840
372	1218632121	139009	10 0.0 2	4094	10.0.0.6	5720	209	207	1678172912	2165394042	22701	1448		5840
373	1218632121	139012	10.0.0.2	4094	10.0.0.6	5720	210	208	1678174360	2165394042	22702	1448		5840
374	1218632121	139014	10.0.0.2	4094	10.0.0 6	5720	211	209	1678175808	2165394042	22703	1448		5840
375	1218632121	214756	10 0.0.2	4094	10.0.0.6	5720	175	210	1678123680	2165394042	22667	1448		5840
376	1218632121	215895	10.0.0.2	4094	10.0.0.6	5720	194	211	1678151192	2165394042	22686	1448		5840

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
377	1218632121	226168	10 0 0.6	5720	10.0.0.2	4094	163	164	2165394042	1678123680	28890	0		65160
378	1218632121	226191	10 0 0 6	5720	10.0.0.2	4094	164	165	2165394042	1678123680	28891	0		65160
379	1218632121	226203	10 0 0 6	5720	10.0 0.2	4094	165	166	2165394042	1678123680	28892	0		65160
380	1218632121	226214	10 0.0 6	5720	10.0.0.2	4094	166	167	2165394042	1678123680	28893	0		65160
381	1218632121	226225	10 0 0.6	5720	10 0.0.2	4094	167	168	2165394042	1678123680	28894	0		65160
382	1218632121	226237	10.0.0.6	5720	10.0 0.2	4094	168	169	2165394042	1678123680	28895	0		65160
383	1218632121	226248	10 0.0 6	5720	10.0 0.2	4094	169	170	2165394042	1678123680	28896	0		65160
384	1218632121	226259	10 0.0 6	5720	10.0.0.2	4094	170	171	2165394042	1678123680	28897	0		65160
385	1218632121	226270	10 0 0.6	5720	10 0 0.2	4094	171	172	2165394042	1678123680	28898	0		65160
386	1218632121	226281	10 0.0 6	5720	10.0 0.2	4094	172	173	2165394042	1678123680	28899	0		65160
387	1218632121	226292	10.0.0 6	5720	10.0.0.2	4094	173	174	2165394042	1678123680	28900	0		65160
388	1218632121	226303	10.0 0 6	5720	10.0.0.2	4094	174	175	2165394042	1678123680	28901	0		65160
389	1218632121	226314	10.0 0 6	5720	10.0.0.2	4094	175	176	2165394042	1678123680	28902	0		65160
390	1218632121	226326	10.0 0 6	5720	10.0.0.2	4094	176	177	2165394042	1678123680	28903	0		65160
391	1218632121	226339	10.0.0 6	5720	10 0.0.2	4094	177	178	2165394042	1678123680	28904	0		65160
392	1218632121	226350	10 0 0 6	5720	10.0.0.2	4094	178	179	2165394042	1678123680	28905	0		65160
393	1218632121	226361	10.0.0 6	5720	10.0 0.2	4094	179	180	2165394042	1678123680	28906	0		65160
394	1218632121	226372	10.0.0 6	5720	10.0 0.2	4094	180	181	2165394042	1678123680	28907	0		65160
395	1218632121	226383	10 0.0 6	5720	10.0.0.2	4094	181	182	2165394042	1678123680	28908	0		65160
396	1218632121	226397	10.0.0.6	5720	10.0.0.2	4094	182	183	2165394042	1678123680	28909	0		65160
397	1218632121	226409	10.0.0.6	5720	10.0.0.2	4094	183	184	2165394042	1678123680	28910	0		65160
398	1218632121	226420	10.0.0 6	5720	10 0.0.2	4094	184	185	2165394042	1678123680	28911	0		65160
399	1218632121	226431	10 0.0 6	5720	10.0.0.2	4094	185	186	2165394042	1678123680	28912	0		65160
400	1218632121	226442	10.0.0.6	5720	10.0 0.2	4094	186	187	2165394042	1678123680	28913	0		65160
401	1218632121	226453	10.0.0.6	5720	10.0.0.2	4094	187	188	2165394042	1678123680	28914	0		65160
402	1218632121	226465	10.0.0 6	5720	10.0.0.2	4094	188	189	2165394042	1678123680	28915	0		65160
403	1218632121	226476	10 0.0 6	5720	10.0.0.2	4094	189	190	2165394042	1678123680	28916	0		65160
404	1218632121	226487	10.0.0 6	5720	10.0.0.2	4094	190	191	2165394042	1678123680	28917	0		65160
405	1218632121	226498	10.0.0 6	5720	10.0.0.2	4094	191	192	2165394042	1678123680	28918	0		65160
406	1218632121	226509	10.0.0 6	5720	10.0.0.2	4094	192	193	2165394042	1678123680	28919	0		65160
407	1218632121	226520	10.0.0 6	5720	10.0.0.2	4094	193	194	2165394042	1678123680	28920	0		65160
408	1218632121	226531	10 0 0.6	5720	10.0.0.2	4094	194	195	2165394042	1678123680	28921	0		65160
409	1218632121	226542	10.0.0 6	5720	10.0 0.2	4094	195	196	2165394042	1678123680	28922	0		65160
410	1218632121	226553	10.0.0 6	5720	10.0.0.2	4094	196	197	2165394042	1678123680	28923	0		65160
411	1218632121	226565	10.0 0.6	5720	10.0.0.2	4094	197	198	2165394042	1678123680	28924	0		65160
412	1218632121	226577	10.0.0 6	5720	10.0.0.2	4094	198	199	2165394042	1678123680	28925	0		65160
413	1218632121	226609	10 0.0.6	5720	10.0.0.2	4094	199	200	2165394042	1678151192	28926	0		65160
414	1218632121	226625	10.0.0 6	5720	10.0.0.2	4094	200	201	2165394042	1678177256	28927	0		65160
415	1218632121	376702	10.0.0.2	4094	10.0.0.6	5720	212	212	1678177256	2165394042	22704	1448		5840
416	1218632121	377106	10.0 0.2	4094	10.0.0 6	5720	213	213	1678178704	2165394042	22705	1448		5840
417	1218632121	377108	10.0.0.2	4094	10.0.0 6	5720	214	214	1678180152	2165394042	22706	1448		5840
418	1218632121	377110	10.0.0.2	4094	10.0.0 6	5720	215	215	1678181600	2165394042	22707	1448		5840
419	1218632121	377112	10.0 0 2	4094	10.0.0 6	5720	216	216	1678183048	2165394042	22708	1448		5840
420	1218632121	377505	10 0.0.2	4094	10.0.0 6	5720	217	217	1678184496	2165394042	22709	1448		5840
421	1218632121	377508	10 0.0.2	4094	10 0.0 6	5720	218	218	1678185944	2165394042	22710	1448		5840
422	1218632121	377510	10 0.0.2	4094	10.0.0 6	5720	219	219	1678187392	2165394042	22711	1448		5840
423	1218632121	377905	10 0.0.2	4094	10.0.0 6	5720	220	220	1678123680	2165394042	22712	1448		5840
424	1218632121	377907	10 0.0.2	4094	10.0.0.6	5720	221	221	1678151192	2165394042	22713	1448		5840
425	1218632121	378452	10 0.0.2	4094	10.0.0 6	5720	222	222	1678188840	2165394042	22714	1448		5840
426	1218632121	378806	10.0.0.2	4094	10.0.0.6	5720	223	223	1678190288	2165394042	22715	1448		5840
427	1218632121	466157	10 0 0 6	5720	10 0 0 2	4094	201	202	2165394042	1678178704	28928	0		65160
428	1218632121	466187	10 0 0 6	5720	10 0.0.2	4094	202	203	2165394042	1678180152	28929	0		65160
429	1218632121	466210	10 0 0 6	5720	10 0 0.2	4094	203	204	2165394042	1678181600	28930	0		65160
430	1218632121	466232	10.0 0 6	5720	10.0 0 2	4094	204	205	2165394042	1678183048	28931	0		65160
431	1218632121	466254	10 0 0.6	5720	10.0.0 2	4094	205	206	2165394042	1678184496	28932	0		65160
432	1218632121	466279	10.0 0 6	5720	10 0 0.2	4094	206	207	2165394042	1678185944	28933	0		65160
433	1218632121	466290	10.0.0 6	5720	10.0.0.2	4094	207	208	2165394042	1678187392	28934	0		65160
434	1218632121	466303	10 0.0.6	5720	10.0 0.2	4094	208	209	2165394042	1678188840	28935	0		65160
435	1218632121	466316	10.0 0 6	5720	10 0 0.2	4094	209	210	2165394042	1678188840	28936	0		65160
436	1218632121	466329	10.0 0.6	5720	10.0 0.2	4094	210	211	2165394042	1678188840	28937	0		65160
437	1218632121	466342	10.0 0.6	5720	10.0.0 2	4094	211	212	2165394042	1678190288	28938	0		65160
438	1218632121	466353	10.0.0.6	5720	10.0 0.2	4094	212	213	2165394042	1678191736	28939	0		65160
439	1218632121	616651	10.0 0.2	4094	10.0 0.6	5720	224	224	1678191736	2165394042	22716	1448		5840
440	1218632121	616662	10.0.0.2	4094	10 0 0.6	5720	225	225	1678193184	2165394042	22717	1448		5840
441	1218632121	616999	10.0 0.2	4094	10 0.0.6	5720	226	226	1678194632	2165394042	22718	1448		5840
442	1218632121	617002	10 0.0 2	4094	10.0.0 6	5720	228	227	1678197528	2165394042	22720	1448		5840
443	1218632121	617399	10.0.0.2	4094	10.0.0 6	5720	229	228	1678198976	2165394042	22721	1448		5840
444	1218632121	617401	10.0.0.2	4094	10.0.0 6	5720	230	229	1678200424	2165394042	22722	1448		5840
445	1218632121	617403	10 0.0.2	4094	10.0.0.6	5720	231	230	1678201872	2165394042	22723	1448		5840
446	1218632121	617405	10.0.0.2	4094	10 0.0 6	5720	232	231	1678203320	2165394042	22724	1448		5840
447	1218632121	617799	10.0.0.2	4094	10.0 0 6	5720	234	232	1678206216	2165394042	22726	1448		5840
448	1218632121	617801	10.0 0.2	4094	10.0.0.6	5720	235	233	1678207664	2165394042	22727	1448		5840
449	1218632121	617803	10.0 0.2	4094	10.0 0.6	5720	236	234	1678209112	2165394042	22728	1448		5840
450	1218632121	617804	10.0.0.2	4094	10.0 0.6	5720	237	235	1678210560	2165394042	22729	1448		5840
451	1218632121	695089	10.0.0.2	4094	10.0.0.6	5720	227	236	1678196080	2165394042	22719	1448		5840
452	1218632121	695498	10.0.0.2	4094	10.0.0.6	5720	233	237	1678204768	2165394042	22725	1448		5840
453	1218632121	706151	10.0.0.6	5720	10.0.0.2	4094	213	214	2165394042	1678193184	28940	0		65160
454	1218632121	706180	10.0.0.6	5720	10.0.0.2	4094	214	215	2165394042	1678194632	28941	0		65160
455	1218632121	706204	10.0.0.6	5720	10.0.0.2	4094	215	216	2165394042	1678196080	28942	0		65160
456	1218632121	706221	10.0 0.6	5720	10.0.0.2	4094	216	217	2165394042	1678196080	28943	0		65160
457	1218632121	706232	10.0.0 6	5720	10.0.0.2	4094	217	218	2165394042	1678196080	28944	0		65160
458	1218632121	706243	10.0.0.6	5720	10.0.0.2	4094	218	219	2165394042	1678196080	28945	0		65160
459	1218632121	706254	10.0.0.6	5720	10.0.0.2	4094	219	220	2165394042	1678196080	28946	0		65160
460	1218632121	706265	10.0.0.6	5720	10.0.0.2	4094	220	221	2165394042	1678196080	28947	0		65160
461	1218632121	706277	10.0.0 6	5720	10.0.0.2	4094	221	222	2165394042	1678196080	28948	0		65160
462	1218632121	706288	10 0.0.6	5720	10.0.0.2	4094	222	223	2165394042	1678196080	28949	0		65160
463	1218632121	706299	10 0.0 6	5720	10.0.0.2	4094	223	224	2165394042	1678196080	28950	0		65160
464	1218632121	706309	10.0.0.6	5720	10.0.0.2	4094	224	225	2165394042	1678196080	28951	0		65160
465	1218632121	706338	10 0.0.6	5720	10.0.0.2	4094	225	226	2165394042	1678204768	28952	0		65160
466	1218632121	706351	10.0.0 6	5720	10.0 0.2	4094	226	227	2165394042	1678212008	28953	0		65160
467	1218632121	856602	10.0.0.2	4094	10.0.0 6	5720	238	238	1678212008	2165394042	22730	1448		5840
468	1218632121	856614	10.0 0.2	4094	10.0.0.6	5720	239	239	1678213456	2165394042	22731	1448		5840
469	1218632121	856994	10.0 0.2	4094	10.0.0.6	5720	240	240	1678214904	2165394042	22732	1448		5840
470	1218632121	856996	10.0.0.2	4094	10 0.0 6	5720	241	241	1678216352	2165394042	22733	1448		5840
471	1218632121	856997	10.0.0.2	4094	10.0 0 6	5720	242	242	1678217800	2165394042	22734	1448		5840
472	1218632121	857393	10.0.0.2	4094	10 0.0.6	5720	243	243	1678219248	2165394042	22735	1448		5840

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
473	1218632121	857395	10 0 0 2	4094	10 0 0 6	5720	244	244	1678220696	2165394042	22736	1448		5840
474	1218632121	857396	10 0 0 2	4094	10 0 0 6	5720	245	245	1678222144	2165394042	22737	1448		5840
475	1218632121	857398	10 0 0 2	4094	10 0 0 6	5720	246	246	1678223592	2165394042	22738	1448		5840
476	1218632121	857400	10 0 0 2	4094	10 0 0 6	5720	247	247	1678225040	2165394042	22739	1448		5840
477	1218632121	857794	10 0 0 2	4094	10 0 0 6	5720	248	248	1678226488	2165394042	22740	1448		5840
478	1218632121	857795	10 0 0 2	4094	10 0 0 6	5720	249	249	1678227936	2165394042	22741	1448		5840
479	1218632121	857797	10 0 0 2	4094	10 0 0 6	5720	250	250	1678229384	2165394042	22742	1448		5840
480	1218632121	857799	10 0 0 2	4094	10 0 0 6	5720	251	251	1678230832	2165394042	22743	1448		5840
481	1218632121	857801	10.0 0 2	4094	10 0 0 6	5720	252	252	1678232280	2165394042	22744	1448		5840
482	1218632121	857803	10.0 0 2	4094	10 0 0 6	5720	253	253	1678233728	2165394042	22745	1448		5840
483	1218632121	857805	10 0 0 2	4094	10 0 0 6	5720	254	254	1678235176	2165394042	22746	1448		5840
484	1218632121	858293	10 0 0.2	4094	10.0 0 6	5720	255	255	1678236624	2165394042	22747	1448		5840
485	1218632121	858295	10 0 0.2	4094	10 0 0 6	5720	256	256	1678238072	2165394042	22748	1448		5840
486	1218632121	946147	10 0 0 6	5720	10 0 0 2	4094	227	228	2165394042	1678213456	28954	0		65160
487	1218632121	946177	10 0 0 6	5720	10.0 0 2	4094	228	229	2165394042	1678214904	28955	0		65160
488	1218632121	946200	10 0 0 6	5720	10 0 0 2	4094	229	230	2165394042	1678216352	28956	0		65160
489	1218632121	946224	10 0 0 6	5720	10 0 0 2	4094	230	231	2165394042	1678217800	28957	0		65160
490	1218632121	946247	10 0.0 6	5720	10.0 0 2	4094	231	232	2165394042	1678219248	28958	0		65160
491	1218632121	946270	10 0 0 6	5720	10.0 0 2	4094	232	233	2165394042	1678220696	28959	0		65160
492	1218632121	946282	10 0 0 6	5720	10.0 0 2	4094	233	234	2165394042	1678222144	28960	0		65160
493	1218632121	946294	10 0 0 6	5720	10.0 0.2	4094	234	235	2165394042	1678223592	28961	0		65160
494	1218632121	946305	10.0.0 6	5720	10.0.0 2	4094	235	236	2165394042	1678225040	28962	0		65160
495	1218632121	946316	10.0 0 6	5720	10.0 0.2	4094	236	237	2165394042	1678226488	28963	0		65160
496	1218632121	946327	10 0 0 6	5720	10.0 0.2	4094	237	238	2165394042	1678227936	28964	0		65160
497	1218632121	946338	10 0 0.6	5720	10.0 0.2	4094	238	239	2165394042	1678229384	28965	0		65160
498	1218632121	946350	10.0 0.6	5720	10.0 0.2	4094	239	240	2165394042	1678230832	28966	0		65160
499	1218632121	946363	10.0.0 6	5720	10 0.0.2	4094	240	241	2165394042	1678233728	28967	0		65160
500	1218632121	946377	10.0.0.6	5720	10.0.0.2	4094	241	242	2165394042	1678236624	28968	0		65160
501	1218632121	946390	10.0 0 6	5720	10.0.0 2	4094	242	243	2165394042	1678239520	28969	0		65160
502	1218632122	96677	10.0.0.2	4094	10.0.0 6	5720	257	257	1678239520	2165394042	22749	1448		5840
503	1218632122	96692	10 0 0.2	4094	10 0 0 6	5720	258	258	1678240968	2165394042	22750	1448		5840
504	1218632122	97089	10 0 0.2	4094	10 0.0.6	5720	259	259	1678242416	2165394042	22751	1448		5840
505	1218632122	97091	10.0.0.2	4094	10 0.0 6	5720	260	260	1678243864	2165394042	22752	1448		5840
506	1218632122	97093	10.0.0.2	4094	10.0 0 6	5720	261	261	1678245312	2165394042	22753	1448		5840
507	1218632122	97095	10.0.0.2	4094	10.0.0.6	5720	262	262	1678246760	2165394042	22754	1448		5840
508	1218632122	97097	10.0.0.2	4094	10.0.0.6	5720	263	263	1678248208	2165394042	22755	1448		5840
509	1218632122	97099	10.0.0.2	4094	10 0.0 6	5720	264	264	1678249656	2165394042	22756	1448		5840
510	1218632122	97490	10.0 0.2	4094	10.0.0 6	5720	265	265	1678251104	2165394042	22757	1448		5840
511	1218632122	97492	10.0.0.2	4094	10.0.0.6	5720	266	266	1678252552	2165394042	22758	1448		5840
512	1218632122	97494	10.0 0 2	4094	10 0.0 6	5720	268	267	1678255448	2165394042	22760	1448		5840
513	1218632122	97496	10.0.0.2	4094	10 0.0 6	5720	269	268	1678256896	2165394042	22761	1448		5840
514	1218632122	97498	10.0 0.2	4094	10.0.0 6	5720	271	269	1678259792	2165394042	22763	1448		5840
515	1218632122	97500	10 0.0.2	4094	10.0.0.6	5720	272	270	1678261240	2165394042	22764	1448		5840
516	1218632122	97889	10 0 0.2	4094	10 0 0 6	5720	273	271	1678262688	2165394042	22765	1448		5840
517	1218632122	97891	10.0.0.2	4094	10.0.0.6	5720	274	272	1678264136	2165394042	22766	1448		5840
518	1218632122	97892	10 0.0.2	4094	10.0.0 6	5720	275	273	1678265584	2165394042	22767	1448		5840
519	1218632122	97894	10.0.0.2	4094	10.0.0 6	5720	276	274	1678267032	2165394042	22768	1448		5840
520	1218632122	97896	10.0.0.2	4094	10.0.0 6	5720	277	275	1678268480	2165394042	22769	1448		5840
521	1218632122	97898	10.0.0.2	4094	10 0.0 6	5720	278	276	1678269928	2165394042	22770	1448		5840
522	1218632122	97900	10.0.0.2	4094	10.0.0 6	5720	279	277	1678271376	2165394042	22771	1448		5840
523	1218632122	97901	10 0 0 2	4094	10.0 0 6	5720	280	278	1678272824	2165394042	22772	1448		5840
524	1218632122	98289	10 0.0.2	4094	10.0 0 6	5720	281	279	1678274272	2165394042	22773	1448		5840
525	1218632122	98291	10 0 0.2	4094	10 0 0 6	5720	282	280	1678275720	2165394042	22774	1448		5840
526	1218632122	98292	10 0 0 2	4094	10 0 0 6	5720	283	281	1678277168	2165394042	22775	1448		5840
527	1218632122	98294	10 0 0 2	4094	10 0 0 6	5720	284	282	1678278616	2165394042	22776	1448		5840
528	1218632122	98296	10.0.0.2	4094	10.0 0 6	5720	285	283	1678280064	2165394042	22777	1448		5840
529	1218632122	98298	10 0 0.2	4094	10 0 0 6	5720	286	284	1678281512	2165394042	22778	1448		5840
530	1218632122	98300	10 0 0.2	4094	10.0 0 6	5720	287	285	1678282960	2165394042	22779	1448		5840
531	1218632122	98302	10 0 0.2	4094	10 0 0 6	5720	288	286	1678284408	2165394042	22780	1448		5840
532	1218632122	98690	10 0 0.2	4094	10 0 0 6	5720	289	287	1678285856	2165394042	22781	1448		5840
533	1218632122	98693	10 0 0 2	4094	10 0.0 6	5720	290	288	1678287304	2165394042	22782	1448		5840
534	1218632122	98694	10 0 0 2	4094	10 0.0 6	5720	291	289	1678288752	2165394042	22783	1448		5840
535	1218632122	175383	10 0 0.2	4094	10 0 0 6	5720	267	290	1678254000	2165394042	22759	1448		5840
536	1218632122	175787	10.0.0.2	4094	10.0.0 6	5720	270	291	1678258344	2165394042	22762	1448		5840
537	1218632122	186219	10 0 0 6	5720	10 0.0.2	4094	243	244	2165394042	1678248208	28970	0		65160
538	1218632122	186239	10 0 0.6	5720	10 0 0 2	4094	244	245	2165394042	1678251104	28971	0		65160
539	1218632122	186254	10.0 0 6	5720	10 0.0 2	4094	245	246	2165394042	1678254000	28972	0		65160
540	1218632122	186270	10.0.0 6	5720	10.0 0 2	4094	246	247	2165394042	1678254000	28973	0		65160
541	1218632122	186282	10 0 0 6	5720	10.0.0 2	4094	247	248	2165394042	1678254000	28974	0		65160
542	1218632122	186296	10.0 0.6	5720	10.0.0.2	4094	248	249	2165394042	1678254000	28975	0		65160
543	1218632122	186308	10.0.0 6	5720	10 0 0.2	4094	249	250	2165394042	1678254000	28976	0		65160
544	1218632122	186319	10.0.0.6	5720	10.0.0.2	4094	250	251	2165394042	1678254000	28977	0		65160
545	1218632122	186330	10.0 0 6	5720	10.0 0.2	4094	251	252	2165394042	1678254000	28978	0		65160
546	1218632122	186341	10.0.0 6	5720	10 0.0.2	4094	252	253	2165394042	1678254000	28979	0		65160
547	1218632122	186352	10 0.0 6	5720	10 0.0 2	4094	253	254	2165394042	1678254000	28980	0		65160
548	1218632122	186363	10 0.0 6	5720	10.0.0 2	4094	254	255	2165394042	1678254000	28981	0		65160
549	1218632122	186373	10.0.0 6	5720	10.0.0 2	4094	255	256	2165394042	1678254000	28982	0		65160
550	1218632122	186385	10 0.0 6	5720	10.0.0.2	4094	256	257	2165394042	1678254000	28983	0		65160
551	1218632122	186396	10.0 0 6	5720	10.0.0.2	4094	257	258	2165394042	1678254000	28984	0		65160
552	1218632122	186406	10.0.0.6	5720	10.0.0 2	4094	258	259	2165394042	1678254000	28985	0		65160
553	1218632122	186417	10.0 0.6	5720	10.0.0 2	4094	259	260	2165394042	1678254000	28986	0		65160
554	1218632122	186429	10 0.0.6	5720	10 0.0.2	4094	260	261	2165394042	1678254000	28987	0		65160
555	1218632122	186440	10.0.0 6	5720	10 0 0.2	4094	261	262	2165394042	1678254000	28988	0		65160
556	1218632122	186450	10.0 0.6	5720	10.0.0.2	4094	262	263	2165394042	1678254000	28989	0		65160
557	1218632122	186462	10.0.0.6	5720	10 0.0.2	4094	263	264	2165394042	1678254000	28990	0		65160
558	1218632122	186472	10.0.0 6	5720	10 0.0.2	4094	264	265	2165394042	1678254000	28991	0		65160
559	1218632122	186483	10.0.0 6	5720	10.0 0.2	4094	265	266	2165394042	1678254000	28992	0		65160
560	1218632122	186494	10.0 0.6	5720	10.0.0.2	4094	266	267	2165394042	1678254000	28993	0		65160
561	1218632122	186505	10.0.0 6	5720	10 0.0.2	4094	267	268	2165394042	1678254000	28994	0		65160
562	1218632122	186516	10.0 0 6	5720	10.0.0 2	4094	268	269	2165394042	1678254000	28995	0		65160
563	1218632122	186531	10 0 0.6	5720	10.0.0.2	4094	269	270	2165394042	1678258344	28996	0		65160
564	1218632122	186571	10.0.0 6	5720	10.0.0 2	4094	270	271	2165394042	1678290200	28997	0		65160
565	1218632122	336764	10.0.0.2	4094	10.0.0 6	5720	292	292	1678290200	2165394042	22784	1448		5840
566	1218632122	337084	10.0.0.2	4094	10 0.0.6	5720	293	293	1678291648	2165394042	22785	1448		5840
567	1218632122	337086	10 0.0.2	4094	10.0.0.6	5720	294	294	1678293096	2165394042	22786	1448		5840
568	1218632122	337088	10.0.0 2	4094	10.0.0.6	5720	295	295	1678294544	2165394042	22787	1448		5840

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
569	1218632122	337090	10 0 0 2	4094	10 0 0.6	5720	296	296	1678295992	2165394042	22788	1448		5840
570	1218632122	337092	10.0.0.2	4094	10 0.0 6	5720	297	297	1678297440	2165394042	22789	1448		5840
571	1218632122	337484	10 0 0.2	4094	10 0 0 6	5720	298	298	1678298888	2165394042	22790	1448		5840
572	1218632122	337486	10 0.0 2	4094	10.0 0 6	5720	299	299	1678300336	2165394042	22791	1448		5840
573	1218632122	337487	10 0 0 2	4094	10 0 0 6	5720	300	300	1678301784	2165394042	22792	1448		5840
574	1218632122	337489	10 0 0.2	4094	10 0 0 6	5720	301	301	1678303232	2165394042	22793	1448		5840
575	1218632122	337491	10 0.0 2	4094	10.0.0 6	5720	302	302	1678304680	2165394042	22794	1448		5840
576	1218632122	337493	10.0 0.2	4094	10 0.0 6	5720	303	303	1678306128	2165394042	22795	1448		5840
577	1218632122	337495	10.0 0.2	4094	10.0.0 6	5720	304	304	1678307576	2165394042	22796	1448		5840
578	1218632122	337886	10.0.0.2	4094	10.0 0 6	5720	305	305	1678309024	2165394042	22797	1448		5840
579	1218632122	337888	10 0.0 2	4094	10 0.0 6	5720	306	306	1678310472	2165394042	22798	1448		5840
580	1218632122	337890	10 0 0 2	4094	10 0.0 6	5720	307	307	1678311920	2165394042	22799	1448		5840
581	1218632122	337891	10 0 0.2	4094	10.0 0 6	5720	308	308	1678313368	2165394042	22800	1448		5840
582	1218632122	338284	10 0.0 2	4094	10 0 0 6	5720	310	309	1678316264	2165394042	22802	1448		5840
583	1218632122	338286	10 0 0.2	4094	10 0.0 6	5720	311	310	1678317712	2165394042	22803	1448		5840
584	1218632122	338684	10 0 0.2	4094	10.0 0 6	5720	312	311	1678319160	2165394042	22804	1448		5840
585	1218632122	338686	10 0 0.2	4094	10.0.0 6	5720	313	312	1678320608	2165394042	22805	1448		5840
586	1218632122	338688	10 0 0.2	4094	10 0.0 6	5720	314	313	1678322056	2165394042	22806	1448		5840
587	1218632122	338689	10 0.0.2	4094	10.0 0 6	5720	315	314	1678323504	2165394042	22807	1448		5840
588	1218632122	338691	10.0.0.2	4094	10.0.0 6	5720	316	315	1678324952	2165394042	22808	1448		5840
589	1218632122	415829	10.0 0.2	4094	10 0.0 6	5720	309	316	1678314816	2165394042	22801	1448		5840
590	1218632122	426137	10.0 0 6	5720	10 0.0.2	4094	271	272	2165394042	1678291648	28998	0		65160
591	1218632122	426167	10 0.0.6	5720	10.0.0.2	4094	272	273	2165394042	1678293096	28999	0		65160
592	1218632122	426190	10 0.0 6	5720	10.0.0.2	4094	273	274	2165394042	1678294544	29000	0		65160
593	1218632122	426212	10.0 0 6	5720	10.0.0.2	4094	274	275	2165394042	1678295992	29001	0		65160
594	1218632122	426235	10.0 0.6	5720	10.0.0.2	4094	275	276	2165394042	1678297440	29002	0		65160
595	1218632122	426260	10 0 0 6	5720	10.0.0 2	4094	276	277	2165394042	1678298888	29003	0		65160
596	1218632122	426271	10 0.0 6	5720	10 0.0.2	4094	277	278	2165394042	1678300336	29004	0		65160
597	1218632122	426283	10.0.0 6	5720	10 0.0.2	4094	278	279	2165394042	1678301784	29005	0		65160
598	1218632122	426294	10.0.0.6	5720	10.0.0.2	4094	279	280	2165394042	1678303232	29006	0		65160
599	1218632122	426306	10.0 0 6	5720	10.0.0.2	4094	280	281	2165394042	1678304680	29007	0		65160
600	1218632122	426317	10.0 0.6	5720	10.0.0.2	4094	281	282	2165394042	1678306128	29008	0		65160
601	1218632122	426329	10.0 0.6	5720	10 0.0.2	4094	282	283	2165394042	1678307576	29009	0		65160
602	1218632122	426340	10 0.0 6	5720	10.0 0.2	4094	283	284	2165394042	1678309024	29010	0		65160
603	1218632122	426354	10 0.0 6	5720	10.0 0.2	4094	284	285	2165394042	1678311920	29011	0		65160
604	1218632122	426369	10 0.0 6	5720	10.0 0.2	4094	285	286	2165394042	1678314816	29012	0		65160
605	1218632122	426385	10 0.0.6	5720	10.0.0.2	4094	286	287	2165394042	1678314816	29013	0		65160
606	1218632122	426397	10 0 0.6	5720	10.0 0.2	4094	287	288	2165394042	1678314816	29014	0		65160
607	1218632122	426408	10.0 0.6	5720	10.0.0.2	4094	288	289	2165394042	1678314816	29015	0		65160
608	1218632122	426419	10 0 0 6	5720	10.0.0.2	4094	289	290	2165394042	1678314816	29016	0		65160
609	1218632122	426429	10 0 0 6	5720	10 0 0.2	4094	290	291	2165394042	1678314816	29017	0		65160
610	1218632122	426441	10 0.0.6	5720	10.0.0.2	4094	291	292	2165394042	1678314816	29018	0		65160
611	1218632122	426452	10 0 0.6	5720	10 0 0.2	4094	292	293	2165394042	1678314816	29019	0		65160
612	1218632122	426467	10.0 0 6	5720	10 0 0 2	4094	293	294	2165394042	1678326400	29020	0		65160
613	1218632122	576603	10.0 0.2	4094	10.0 0.6	5720	317	317	1678326400	2165394042	22809	1448		5840
614	1218632122	576979	10 0 0.2	4094	10.0 0 6	5720	318	318	1678327848	2165394042	22810	1448		5840
615	1218632122	576982	10 0 0.2	4094	10.0.0 6	5720	319	319	1678329296	2165394042	22811	1448		5840
616	1218632122	576984	10 0 0.2	4094	10.0 0 6	5720	320	320	1678330744	2165394042	22812	1448		5840
617	1218632122	576986	10 0 0.2	4094	10.0 0 6	5720	321	321	1678332192	2165394042	22813	1448		5840
618	1218632122	577379	10 0 0.2	4094	10.0 0 6	5720	322	322	1678333640	2165394042	22814	1448		5840
619	1218632122	577381	10 0 0.2	4094	10 0 0 6	5720	323	323	1678335088	2165394042	22815	1448		5840
620	1218632122	577383	10.0 0.2	4094	10.0 0.6	5720	324	324	1678336536	2165394042	22816	1448		5840
621	1218632122	577385	10 0 0 2	4094	10 0 0 6	5720	325	325	1678337984	2165394042	22817	1448		5840
622	1218632122	577386	10 0 0.2	4094	10.0 0 6	5720	326	326	1678339432	2165394042	22818	1448		5840
623	1218632122	577388	10 0 0.2	4094	10 0 0 6	5720	327	327	1678340880	2165394042	22819	1448		5840
624	1218632122	577390	10 0 0 2	4094	10 0 0 6	5720	328	328	1678342328	2165394042	22820	1448		5840
625	1218632122	577779	10.0 0.2	4094	10 0 0 6	5720	329	329	1678343776	2165394042	22821	1448		5840
626	1218632122	577781	10.0.0.2	4094	10.0 0.6	5720	330	330	1678345224	2165394042	22822	1448		5840
627	1218632122	577783	10 0 0.2	4094	10.0.0 6	5720	331	331	1678346672	2165394042	22823	1448		5840
628	1218632122	577785	10.0 0 2	4094	10.0 0 6	5720	332	332	1678348120	2165394042	22824	1448		5840
629	1218632122	577787	10 0 0 2	4094	10.0 0 6	5720	333	333	1678349568	2165394042	22825	1448		5840
630	1218632122	577789	10 0 0 2	4094	10 0 0 6	5720	334	334	1678351016	2165394042	22826	1448		5840
631	1218632122	577790	10 0 0 2	4094	10.0 0.6	5720	335	335	1678352464	2165394042	22827	1448		5840
632	1218632122	577793	10.0.0.2	4094	10 0.0 6	5720	336	336	1678353912	2165394042	22828	1448		5840
633	1218632122	578178	10.0.0 2	4094	10.0.0 6	5720	337	337	1678355360	2165394042	22829	1448		5840
634	1218632122	578180	10 0.0 2	4094	10 0.0 6	5720	338	338	1678356808	2165394042	22830	1448		5840
635	1218632122	578182	10.0 0.2	4094	10.0.0 6	5720	339	339	1678358256	2165394042	22831	1448		5840
636	1218632122	578184	10.0.0 2	4094	10.0.0 6	5720	340	340	1678359704	2165394042	22832	1448		5840
637	1218632122	578186	10.0 0.2	4094	10 0.0 6	5720	341	341	1678361152	2165394042	22833	1448		5840
638	1218632122	578188	10.0.0.2	4094	10.0.0.6	5720	342	342	1678362600	2165394042	22834	1448		5840
639	1218632122	578189	10.0.0.2	4094	10.0.0 6	5720	343	343	1678364048	2165394042	22835	1448		5840
640	1218632122	578191	10.0.0.2	4094	10.0.0 6	5720	344	344	1678365496	2165394042	22836	1448		5840
641	1218632122	578193	10 0.0.2	4094	10.0.0.6	5720	345	345	1678366944	2165394042	22837	1448		5840
642	1218632122	578578	10 0.0.2	4094	10.0.0 6	5720	346	346	1678368392	2165394042	22838	1448		5840
643	1218632122	578581	10 0.0.2	4094	10.0.0 6	5720	347	347	1678369840	2165394042	22839	1448		5840
644	1218632122	578583	10.0.0.2	4094	10.0.0.6	5720	348	348	1678371288	2165394042	22840	1448		5840
645	1218632122	578585	10 0.0.2	4094	10.0.0 6	5720	349	349	1678372736	2165394042	22841	1448		5840
646	1218632122	578586	10.0.0.2	4094	10 0.0 6	5720	350	350	1678374184	2165394042	22842	1448		5840
647	1218632122	578588	10.0.0.2	4094	10 0.0 6	5720	351	351	1678375632	2165394042	22843	1448		5840
648	1218632122	578978	10 0 0.2	4094	10 0.0.6	5720	352	352	1678377080	2165394042	22844	1448		5840
649	1218632122	578980	10.0.0.2	4094	10.0.0.6	5720	353	353	1678378528	2165394042	22845	1448		5840
650	1218632122	578982	10.0 0.2	4094	10.0.0 6	5720	354	354	1678379976	2165394042	22846	1448		5840
651	1218632122	578984	10 0.0.2	4094	10.0.0.6	5720	355	355	1678381424	2165394042	22847	1448		5840
652	1218632122	578986	10.0.0.2	4094	10.0.0.6	5720	356	356	1678382872	2165394042	22848	1448		5840
653	1218632122	666130	10 0 0.6	5720	10.0.0.2	4094	294	295	2165394042	1678327848	29021	0		65160
654	1218632122	666159	10 0.0 6	5720	10.0.0.2	4094	295	296	2165394042	1678329296	29022	0		65160
655	1218632122	666182	10.0.0.6	5720	10.0 0.2	4094	296	297	2165394042	1678330744	29023	0		65160
656	1218632122	666205	10 0.0.6	5720	10.0.0.2	4094	297	298	2165394042	1678332192	29024	0		65160
657	1218632122	666229	10.0.0.6	5720	10.0 0.2	4094	298	299	2165394042	1678333640	29025	0		65160
658	1218632122	666251	10.0.0.6	5720	10.0.0.2	4094	299	300	2165394042	1678335088	29026	0		65160
659	1218632122	666263	10 0.0 6	5720	10.0.0 2	4094	300	301	2165394042	1678336536	29027	0		65160
660	1218632122	666274	10.0.0.6	5720	10.0.0.2	4094	301	302	2165394042	1678337984	29028	0		65160
661	1218632122	666285	10.0.0.6	5720	10 0.0 2	4094	302	303	2165394042	1678339432	29029	0		65160
662	1218632122	666297	10.0.0.6	5720	10.0.0.2	4094	303	304	2165394042	1678340880	29030	0		65160
663	1218632122	666308	10 0.0 6	5720	10.0 0.2	4094	304	305	2165394042	1678342328	29031	0		65160
664	1218632122	666319	10 0.0.6	5720	10.0.0.2	4094	305	306	2165394042	1678343776	29032	0		65160

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
665	1218632122	666330	10.0 0 6	5720	10.0.0 2	4094	306	307	2165394042	1678345224	29033	0		65160
666	1218632122	666340	10.0 0 6	5720	10.0.0.2	4094	307	308	2165394042	1678346672	29034	0		65160
667	1218632122	666356	10 0 0 6	5720	10 0.0.2	4094	308	309	2165394042	1678349568	29035	0		65160
668	1218632122	666370	10 0.0 6	5720	10.0 0.2	4094	309	310	2165394042	1678352464	29036	0		65160
669	1218632122	666383	10 0 0 6	5720	10 0.0.2	4094	310	311	2165394042	1678355360	29037	0		65160
670	1218632122	666397	10 0 0 6	5720	10.0.0.2	4094	311	312	2165394042	1678358256	29038	0		65160
671	1218632122	666410	10 0.0 6	5720	10.0.0.2	4094	312	313	2165394042	1678361152	29039	0		65160
672	1218632122	666424	10.0 0.6	5720	10 0.0.2	4094	313	314	2165394042	1678364048	29040	0		65160
673	1218632122	666437	10 0 0.6	5720	10.0 0.2	4094	314	315	2165394042	1678366944	29041	0		65160
674	1218632122	666450	10.0.0 6	5720	10.0.0.2	4094	315	316	2165394042	1678369840	29042	0		65160
675	1218632122	666464	10.0 0.6	5720	10 0.0.2	4094	316	317	2165394042	1678372736	29043	0		65160
676	1218632122	666477	10 0 0.6	5720	10.0 0.2	4094	317	318	2165394042	1678375632	29044	0		65160
677	1218632122	666490	10 0 0 6	5720	10.0 0.2	4094	318	319	2165394042	1678378528	29045	0		65160
678	1218632122	666504	10 0.0 6	5720	10.0.0.2	4094	319	320	2165394042	1678381424	29046	0		65160
679	1218632122	666517	10.0.0.6	5720	10.0.0.2	4094	320	321	2165394042	1678384320	29047	0		65160
680	1218632122	816696	10.0 0.2	4094	10.0 0.6	5720	357	357	1678384320	2165394042	22849	1448		5840
681	1218632122	816716	10.0 0.2	4094	10.0.0 6	5720	358	358	1678385768	2165394042	22850	1448		5840
682	1218632122	817077	10.0 0.2	4094	10.0.0 6	5720	359	359	1678387216	2165394042	22851	1448		5840
683	1218632122	817079	10.0.0.2	4094	10.0.0 6	5720	360	360	1678388664	2165394042	22852	1448		5840
684	1218632122	817081	10.0.0.2	4094	10.0.0.6	5720	361	361	1678390112	2165394042	22853	1448		5840
685	1218632122	817083	10.0.0.2	4094	10.0.0 6	5720	362	362	1678391560	2165394042	22854	1448		5840
686	1218632122	817085	10.0 0.2	4094	10 0 0 6	5720	363	363	1678393008	2165394042	22855	1448		5840
687	1218632122	817086	10 0.0.2	4094	10.0.0.6	5720	364	364	1678394456	2165394042	22856	1448		5840
688	1218632122	817474	10.0.0.2	4094	10.0 0.6	5720	365	365	1678395904	2165394042	22857	1448		5840
689	1218632122	817476	10 0.0.2	4094	10 0.0 6	5720	366	366	1678397352	2165394042	22858	1448		5840
690	1218632122	817478	10.0.0.2	4094	10.0.0 6	5720	367	367	1678398800	2165394042	22859	1448		5840
691	1218632122	817480	10 0 0.2	4094	10.0 0.6	5720	368	368	1678400248	2165394042	22860	1448		5840
692	1218632122	817482	10.0.0.2	4094	10.0.0.6	5720	369	369	1678401696	2165394042	22861	1448		5840
693	1218632122	817875	10 0.0.2	4094	10.0.0 6	5720	370	370	1678403144	2165394042	22862	1448		5840
694	1218632122	817877	10.0.0.2	4094	10 0.0 6	5720	371	371	1678404592	2165394042	22863	1448		5840
695	1218632122	817879	10.0.0.2	4094	10.0.0.6	5720	372	372	1678406040	2165394042	22864	1448		5840
696	1218632122	817881	10.0.0.2	4094	10.0.0.6	5720	373	373	1678407488	2165394042	22865	1448		5840
697	1218632122	817883	10.0 0.2	4094	10.0.0 6	5720	374	374	1678408936	2165394042	22866	1448		5840
698	1218632122	817884	10.0 0 2	4094	10.0.0.6	5720	375	375	1678410384	2165394042	22867	1448		5840
699	1218632122	817886	10.0.0.2	4094	10 0.0 6	5720	376	376	1678411832	2165394042	22868	1448		5840
700	1218632122	817888	10 0.0 2	4094	10 0 0.6	5720	377	377	1678413280	2165394042	22869	1448		5840
701	1218632122	818276	10.0.0 2	4094	10.0.0 6	5720	378	378	1678414728	2165394042	22870	1448		5840
702	1218632122	818278	10.0.0 2	4094	10.0.0.6	5720	379	379	1678416176	2165394042	22871	1448		5840
703	1218632122	818280	10.0.0 2	4094	10.0 0 6	5720	380	380	1678417624	2165394042	22872	1448		5840
704	1218632122	818282	10 0.0.2	4094	10.0.0.6	5720	381	381	1678419072	2165394042	22873	1448		5840
705	1218632122	818283	10.0.0.2	4094	10.0 0.6	5720	382	382	1678420520	2165394042	22874	1448		5840
706	1218632122	818285	10 0.0.2	4094	10.0 0 6	5720	383	383	1678421968	2165394042	22875	1448		5840
707	1218632122	818674	10 0 0.2	4094	10.0.0.6	5720	384	384	1678423416	2165394042	22876	1448		5840
708	1218632122	818675	10 0 0.2	4094	10 0.0 6	5720	385	385	1678424864	2165394042	22877	1448		5840
709	1218632122	818677	10 0 0.2	4094	10.0.0 6	5720	386	386	1678426312	2165394042	22878	1448		5840
710	1218632122	818679	10 0.0.2	4094	10 0 0.6	5720	387	387	1678427760	2165394042	22879	1448		5840
711	1218632122	818681	10 0.0 2	4094	10 0.0 6	5720	388	388	1678429208	2165394042	22880	1448		5840
712	1218632122	818683	10.0.0 2	4094	10 0.0.6	5720	389	389	1678430656	2165394042	22881	1448		5840
713	1218632122	818685	10.0.0 2	4094	10 0.0.6	5720	390	390	1678432104	2165394042	22882	1448		5840
714	1218632122	819073	10 0.0.2	4094	10 0.0.6	5720	392	391	1678435000	2165394042	22884	1448		5840
715	1218632122	819075	10 0.0.2	4094	10 0 0 6	5720	393	392	1678436448	2165394042	22885	1448		5840
716	1218632122	819077	10 0.0.2	4094	10.0.0 6	5720	394	393	1678437896	2165394042	22886	1448		5840
717	1218632122	819079	10.0 0 2	4094	10 0.0 6	5720	396	394	1678440792	2165394042	22888	1448		5840
718	1218632122	819081	10 0 0.2	4094	10.0.0 6	5720	397	395	1678442240	2165394042	22889	1448		5840
719	1218632122	819082	10 0 0.2	4094	10 0 0 6	5720	398	396	1678443688	2165394042	22890	1448		5840
720	1218632122	819084	10.0.0 2	4094	10.0 0 6	5720	399	397	1678445136	2165394042	22891	1448		5840
721	1218632122	819473	10 0.0 2	4094	10 0.0 6	5720	400	398	1678446584	2165394042	22892	1448		5840
722	1218632122	819475	10 0.0 2	4094	10.0 0.6	5720	401	399	1678448032	2165394042	22893	1448		5840
723	1218632122	896699	10 0.0.2	4094	10 0 0.6	5720	391	400	1678433552	2165394042	22883	1448		5840
724	1218632122	897072	10.0.0 2	4094	10.0.0 6	5720	395	401	1678439344	2165394042	22887	1448		5840
725	1218632122	906200	10 0.0 6	5720	10 0.0.2	4094	321	322	2165394042	1678393008	29048	0		65160
726	1218632122	906220	10 0.0 6	5720	10 0 0.2	4094	322	323	2165394042	1678395904	29049	0		65160
727	1218632122	906234	10 0.0 6	5720	10.0.0.2	4094	323	324	2165394042	1678398800	29050	0		65160
728	1218632122	906248	10.0.0 6	5720	10.0 0 2	4094	324	325	2165394042	1678401696	29051	0		65160
729	1218632122	906261	10 0.0 6	5720	10.0 0.2	4094	325	326	2165394042	1678404592	29052	0		65160
730	1218632122	906275	10.0 0 6	5720	10.0.0.2	4094	326	327	2165394042	1678407488	29053	0		65160
731	1218632122	906289	10.0 0 6	5720	10.0 0 2	4094	327	328	2165394042	1678410384	29054	0		65160
732	1218632122	906302	10.0.0.6	5720	10 0.0.2	4094	328	329	2165394042	1678413280	29055	0		65160
733	1218632122	906317	10.0.0 6	5720	10.0.0 2	4094	329	330	2165394042	1678416176	29056	0		65160
734	1218632122	906330	10.0.0.6	5720	10 0.0.2	4094	330	331	2165394042	1678419072	29057	0		65160
735	1218632122	906344	10.0.0.6	5720	10.0.0 2	4094	331	332	2165394042	1678421968	29058	0		65160
736	1218632122	906357	10.0.0.6	5720	10.0.0.2	4094	332	333	2165394042	1678424864	29059	0		65160
737	1218632122	906371	10 0.0.6	5720	10.0.0.2	4094	333	334	2165394042	1678427760	29060	0		65160
738	1218632122	906384	10 0.0 6	5720	10.0.0.2	4094	334	335	2165394042	1678430656	29061	0		65160
739	1218632122	906399	10.0 0 6	5720	10 0.0.2	4094	335	336	2165394042	1678433552	29062	0		65160
740	1218632122	906414	10 0.0.6	5720	10.0.0.2	4094	336	337	2165394042	1678433552	29063	0		65160
741	1218632122	906426	10 0.0 6	5720	10.0 0.2	4094	337	338	2165394042	1678433552	29064	0		65160
742	1218632122	906438	10.0.0 6	5720	10.0 0.2	4094	338	339	2165394042	1678433552	29065	0		65160
743	1218632122	906451	10.0.0.6	5720	10.0.0.2	4094	339	340	2165394042	1678433552	29066	0		65160
744	1218632122	906463	10 0.0 6	5720	10 0.0 2	4094	340	341	2165394042	1678433552	29067	0		65160
745	1218632122	906474	10 0.0.6	5720	10.0 0.2	4094	341	342	2165394042	1678433552	29068	0		65160
746	1218632122	906485	10 0 0 6	5720	10 0.0.2	4094	342	343	2165394042	1678433552	29069	0		65160
747	1218632122	906496	10 0.0.6	5720	10.0.0 2	4094	343	344	2165394042	1678433552	29070	0		65160
748	1218632122	906508	10 0 0 6	5720	10 0.0.2	4094	344	345	2165394042	1678433552	29071	0		65160
749	1218632122	906523	10.0 0 6	5720	10.0.0.2	4094	345	346	2165394042	1678439344	29072	0		65160
750	1218632122	906536	10 0.0.6	5720	10 0.0.2	4094	346	347	2165394042	1678449480	29073	0		65160