Impact of High TCP's Initial Window in Congested Links

by

Balekaki Gerald Nathan
Reg No: 2009/HD18/16186U
B.Stat(Mak)
Email: balekaki.gerald@cit.mak.ac.ug, Phone Number; +256 782 938361

A Dissertation Report Submitted to School of Graduate Studies in Partial Fulfillment for the Award of Master of Science in Data Communications and Software Engineering Degree of Makerere University

Option: Communication Networks

April, 2012

Declaration

I, Balekaki Gerald Nathan do hereby declare that this Dissertation Report is original and has not been published and/or submitted for any other degree award to any other University before.

Signature ..... Date: .....

Balekaki G. Nathan
B.Stat (Mak)
Department of Networks
School of Computing and Informatics Technology
Makerere University.

Approval

This Dissertation Report has been submitted with the approval of the following supervisor.

Signature ..... Date: .....

Prof. Idris A. Rai
Department of Networks
School of Computing and Informatics Technology
Makerere University.

Dedication

To my beloved family and friends, and to my brother Kayondo Andrew (RIP) for ever you will be remembered.

Acknowledgment

I would like to thank my supervisor Prof. Idris A. Rai for the in-depth understanding of this work. The research skills that he imparted in me. I express my total appreciation to your brilliant support through constant discussions and mentorship.

I am grateful to the School of Computing and Information Technology, together with the entire staff of DICTS especially Wambua J. Kimaili for your support in building the testbed.

To Makerere Communication Network (MCN) Research group members, Eng. Onek Paul, Mr. Sseguya Ronald, Miss. Aye Racheal and Miss. Nalwanga Doreen. Thank you for your full devotion towards discussions.

Lastly, to the most high Almighty God for having answered my prayers. All Glory and honour be to you Oh Lord My God.

Contents

List of Figures

List of Tables

List of Acronyms

ACK	Acknowledgement
CDF	Cumulative Distribution Function
CWND	Congestion Window
HTTP	Hyper Text Transfer Protocol
IE8	Internet Explorer Version Eight
IP	Internet Protocol
IW	Initial Window
KB	Kilo Byte
LAN	Local Area Network
LAS	Least Attained Service
MB	Mega Byte
Kbps	Kilo bits per second
ms	Millisecond
MSS	Maximum Segment Size
NetEM	Network Emulator
QoS	Quality of Service
RAM	Random Access Memory
RTO	Retransmission Timeout
RTT	Round Trip Time
RTTVAR	Round Trip Time Variation
SPDY	SPeeDY Protocol
SRTT	Smoothed Round Trip Time
SS	Slow Start
TCP	Transmission Control Protocol

Abstract

Recent studies proposed that the permitted TCP initial window be increased from between 2 and 4 segments to 10 segments estimated to 15KB of data. The increase has been mainly motivated by accelerated global Internet growth coupled with high speeds and penetration level today. Over 95% of Internet traffic are short flows and the majority of the traffic is Transmission Control Protocol(TCP). Previous work have studied the performance of TCP with IW10, but are not conclusive on the benefits of IW10 for short flows under highly congested links. Congested links are links known to be always fully utilized, which is the case for many links in developing countries. In this study, we investigate the impact of initial window of 3 packets (IW3) and initial window of 10 packets (IW10) on short flows of different sizes by using emulation methods. We also measure the performance of TCP for short flows sharing IW3 and IW10 concurrently (also known as hybrid). Our empirical results revealed that IW10 performs poorly for short flows of less than 10 packets under highly congested links but also much better for short flows carrying bursty traffic.

Chapter 1

Introduction

In this Chapter, Section 1.1 presents the background and motivation of the research. We state the problem that led to this research in Section 1.2 while in Section 1.3, we outline the main and specific objectives of the research. Section 1.4 and 1.5 provides the scope and justification of the research respectively. Finally, we outline the structure of the thesis in Section 1.6.

1.1 Background

TCP congestion window was introduced in 1988 by Van Jacobson as part of TCP's congestion control algorithm [4], the congestion window limits the amount of unacknowledged data the TCP sender injects into a network to prevent overwhelming of the network with larger bursts of data traffic. The initial value of congestion window which is also known as the *initial congestion window*, is mainly used by the TCP during congestion control, both at the beginning of a new connection or after a timeout.

The TCP's initial window value has been incremented overtime. Originally, about a decade ago, the initial window was increased from one segment to roughly four segments [5], since then the Internet has continued to experience accelerated growth of penetration to over 487 million unique IP addresses, higher broadband adoption levels [6], and huge deployment of heavy bandwidth applications such as Google Earth, Youtube, Web browsers like Internet

Explorer (IE8), Mozilla Firefox, Google Chrome and many more browsers that had devised means of opening up multiple connections in order to increase speed of Web downloads.

In order to cope up with the increasing demands of Internet speeds, there has been a proposal to increase the TCP's initial window from between 2 and 4 segments, as specified in [7] to 10 segments estimated at 15KB of data [8] such that much more data can be transmitted at connection startup and eventually will increase exponentially as required by standard slow start algorithm (SS) [2], [9]. There is also an argument in [3] supporting the use of a high TCP initial window size of 10 segments. Despite all the benefits described in [7], [3], [10] IW10 is associated with, most of them have been observed in high-speed links but not congested links.

Recent measurement studies presented in [3], [11], [12] have shown high initial window improves the response time of short flows by up to 10% on average thus improving the overall performance of many web services without risking congestion collapse. Additionally, for links such as high-speed links, high initial window has an additional benefit of improving link utilization. While this is intuitive for non-congested links, similar benefits can not be obviously acclaimed for highly congested links. For instance, the study in [3] also shows that the average response time may worsen if the high initial congestion window is used for browsers with multiple concurrent connections connected to low-speed links. Most of these studies support the use of high initial window but do not clearly show its impact on short flows in congested networks, which is the case in developing countries especially those largely depicted in Africa and South America are usually congested with a high number of users accessing a limited access link at the same time.

1.2 Statement of the Problem

Internet standard bodies such as IETF and Google have proposed to increase and standardize the initial congestion window from 3 to 10 segments [3],[8],[13], without clearly showing the benefits of IW10 on short flows in highly congested and slow access links which are largely depicted in developing countries such as in Africa and South America [6],[14]. Using IW10 other than IW3 means that much more data would be injected into a link at connection

startup and expected to increase exponentially, therefore low access links with congested bottlenecks congest too soon which might lead to a congestion collapse [8],[15]. We study this problem through experimentation to evaluate the performance of TCP with IW3 against IW10 on short flows under different congestion patterns.

1.3 Main Objectives

To investigate the performance of TCP with IW10 on short-lived traffic under very highly congested links.

1.3.1 Specific Objectives

1. To perform experiments to study the performance of TCP with IW3 and IW10 under very high congestion.
2. To observe the effect of TCP with high initial window in highly congested access links.
3. To evaluate the performance of TCP with IW3 and IW10 and also suggest an end-user solution for IW10 problem in congested networks.

1.4 Scope

The research is confined to advanced inter-networking protocols typically Transmission Control Protocol (TCP) of the type standard TCP of variant TCP NewReno with SACK extension. This is the default TCP that is also widely deployed in most networking devices. The study only focuses at a single parameter known as *TCPs initial congestion window* for congestion control. This research is also limited to slow and congested links mainly those observed widely in Africa for example in Uganda at Makerere University.

1.5 Significance of the Study

High initial window i.e IW10 penalizes short flows of at least 10 packets by prolonging the flow completion time resulting into lower throughput due to high congestion at the access link. However, IW10 significantly improves flow completion time of short flows carrying bursty traffic i.e flows with sizes of 21 and 22 packets under very high congestion conditions.

The work done in this research will help Google and network managers such as those in developing countries to have a clear understanding of the impact of increasing TCP initial windows on their networks.

1.6 Thesis Outline

The thesis is organized into five chapters. In Chapter 1, we provide the motivation and background of the study, define the problem and mention the thesis contributions. The remainder of the thesis is structured in four chapters. In the next chapter, we present the Literature Review. We give an overview of standard TCP and also present related work on high initial windows in Chapter 3, we design and describe the testbed. Chapter 4 present and discuss the performance results of IW10, while Chapter 5 draws conclusions and also present future work.

Chapter 2

Literature Review

This chapter is dedicated to reviewing previous works on new and existing initial TCP congestion windows in various network scenarios. Sections 2.1 and 2.1.1 reviews TCP/IP model and briefly explains standard TCP and congestion control mechanisms respectively, while in Section 2.3 we focused on related work on higher initial window sizes and also highlight the need to speed the Internet.

2.1 TCP/IP Architectural Model

The Transmission Control Protocol and Internet Protocol or TCP/IP protocol suite [16] is named from its most important protocols of the Internet. The main design goal of TCP/IP was to build an interconnection of networks, referred to as an internetwork, or internet, that provided universal communication services over heterogeneous physical networks. The clear benefit of such an internetwork is the enabling of communication between hosts on different networks as shown in Figure 2.1, perhaps separated by a large geographical area.

TCP/IP is the protocol that runs on the Internet and TCP alone accounts for the majority of the Internet traffic [17],[18] including that for the most common application protocols such as Simple Mail Transfer Protocol(SMTP), Domain Name Services(DNS), Hyper text Transfer Protocol (HTTP), File Transfer Protocol(FTP) and remote access (Telnet, SSH).

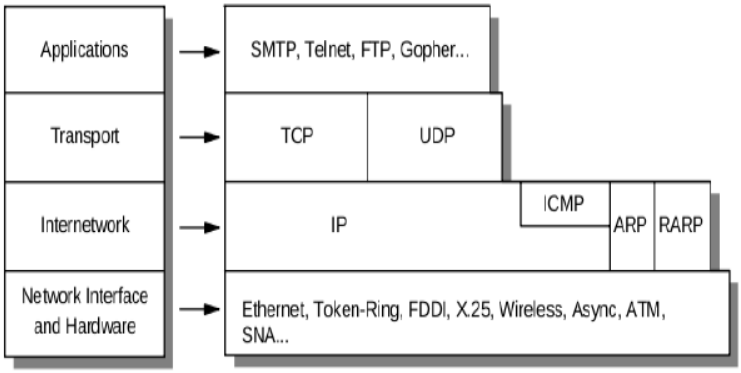


Figure 2.1: Detailed architectural model [adopted from [1]]

2.1.1 TCP Overview

TCP is the connection-oriented transport layer protocol for the TCP/IP protocol stack. TCP is connection-oriented because before one application process can begin to send data to another, the two processes must first perform a handshake with each other that is, they must send some preliminary segments to each other to establish the parameters of the ensuing reliable data transfer. Thereafter, during the data transfer phase, the receiver must acknowledge all successfully transferred data.

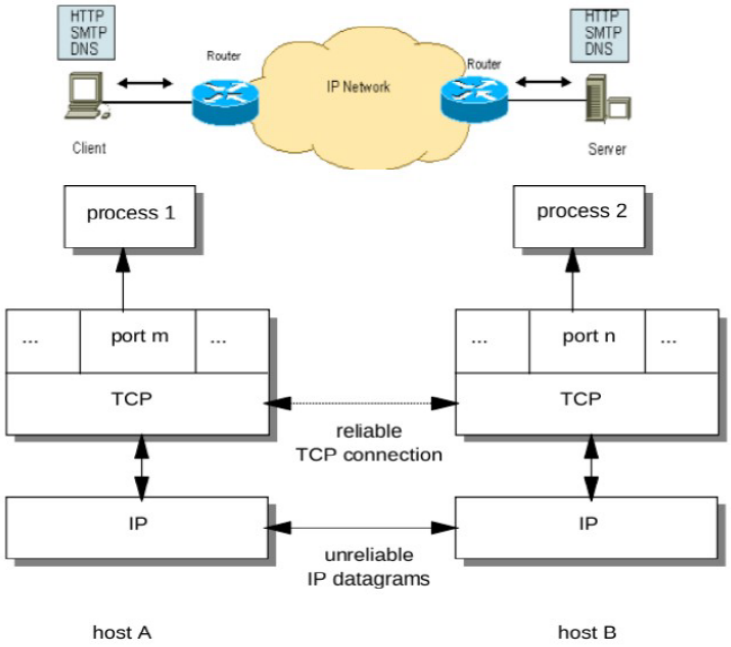


Figure 2.2: TCP/IP host-to-host communication

TCP is designed to provide for accurate delivery of data, multiplexing, demultiplexing, and error detection. TCP and UDP differ in many ways. The most fundamental difference is that UDP is connectionless, while TCP is connection-oriented. UDP is connectionless because it sends data without ever establishing a connection.

Once a TCP connection is established, the two application processes can send data to each other, this is because TCP is full-duplex they can send data at the same time. For instance, consider the sending of data from the client process to the server process. The client process passes a stream of data through the socket (the door of the process). Once the data passes through the door, the data is now in the hands of TCP running in the client as shown in Figure 2.3

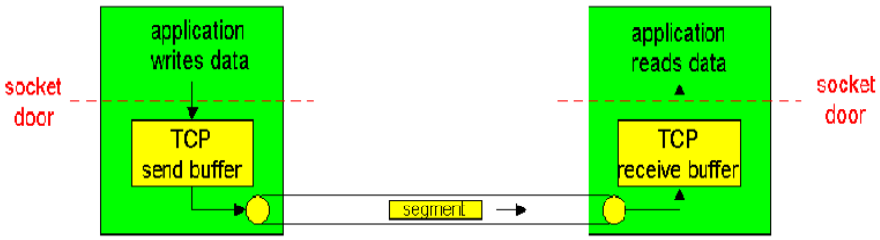


Figure 2.3: TCP send and receive buffers [adopted from [2]]

TCP sender will direct the data to the connection's sender buffer, which is the buffer size agreed upon during the initial three-way handshake. TCP will always capture chunks of data from the send buffer. The maximum amount of data that can be grabbed and placed in a segment is limited by the *Maximum Segment Size (MSS)*. The MSS used in most TCP implementation is commonly configured with value of 1,500 bytes of data[7],[2]. TCP encapsulates each chunk of client data with TCP header, thereby forming TCP segments. The segments are then forwarded to the network layer, where they are separately encapsulated within network layer thereby forming IP datagrams which are sent into the network. At the receiver, segments are received and placed in the TCP connection's receive buffer. The

application reads the stream of data from this buffer. Each side of the connection has its own send buffer and its own receive buffer as in Figure 2.3.

2.1.2 TCP Congestion Control

The fundamental TCP congestion control strategy is to send packets into the network without a reservation and then the host reacts to observable events. TCP's primary function is to properly match the transmission rate of the sender to that of the receiver on the network. It is important for the transmission to be at a high enough rate to ensure good performance, but also to protect against overwhelming the network or receiving host. Originally TCP assumed FIFO queuing and each source determines how much capacity is available to a given flow in the network.

2.1.3 TCP Congestion Window

Congestion Window (cwnd) is a variable held by the TCP source for each connection. The main aspect of TCP is congestion control. TCP uses a number of mechanisms to achieve high performance and avoid congestion collapse, where network performance can fall by several orders of magnitude. These mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger collapse. They also yield an approximately max-min fair allocation between flows.

$$
\begin{aligned}
Wnd_{max} &= min(cwnd, advertised_{wnd}) \\
Effective_{Wnd} &= Wnd_{max}(LastByteSent - LastByteAcked)
\end{aligned}
\tag{2.1}
$$

The cwnd is set based on the perceived level of congestion. The host receives implicit (packet drop) or explicit (packet mark) indications of internal congestion.

2.1.4 Additive Increase / Multiplicative Decrease (AIMD)

Additive Increase is a reaction to perceived available capacity. For every congestion window's worth of packets sent, the cwnd increased fractionally by one packet for each arriving acknowledgment.

$$increment = MSS * (MSS/cwnd)$$
$$cwnd = cwnd + increment \tag{2.2}$$

When loss is detected, the policy is changed to multiplicative decrease, which may for instance, cut the congestion window in half after loss. The key assumption is that a dropped packet and the resultant timeout are due to congestion at a router.

2.2 Standard TCP Congestion Control Algorithms

Modern implementations of TCP contain four intertwined algorithms: Slow-start, congestion avoidance, fast retransmit, and fast recovery, these are briefly explained in Section 2.2.

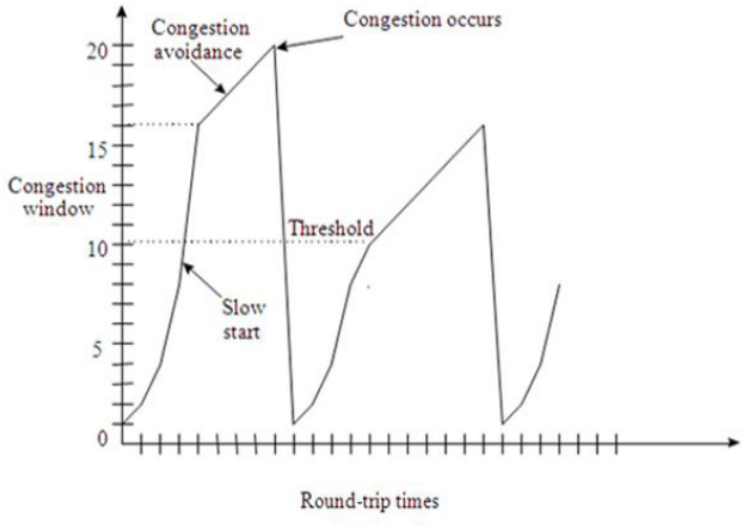


Figure 2.4: Intertwined TCP congestion control algorithms

2.2.1 Basic Slow Start

Slow Start (SS) algorithm requires TCP to initiate gradually by increasing its congestion window as it gains confidence about the networks throughput. During slow start phase, the algorithm sets the initial congestion window to one segment and increases the window by one segment size for each acknowledgment (ACK) received. The slow start increases exponentially until the window reaches the value of the receivers advertised window or detects network congestion.

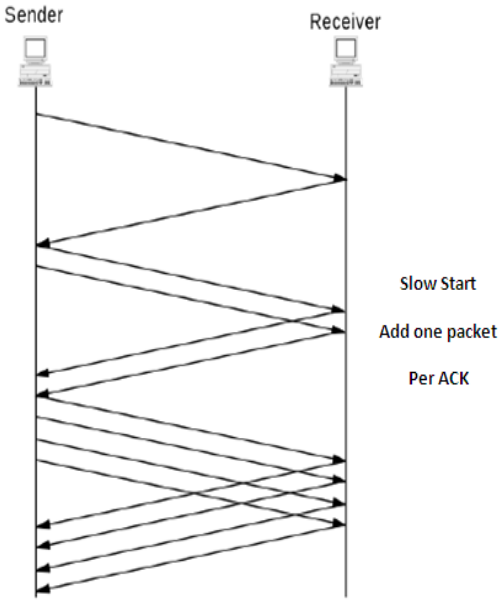


Figure 2.5: Slow Start Phase

If the receiver acknowledges every incoming packet and no packet or acknowledgment losses are observed during slow start, the amount of time required for the congestion window to reach the receivers advertised window is given in Equation (2.3).

$$Slowstarttime = RTT * log_2(W) \tag{2.3}$$

Where W is given as receiver's window size and RTT is round trip time of a specific network link. For a short flow, the Round trip time makes up of significantly larger fraction

of the total transfer time of a flow. Moreover, packet losses at slow-start significantly affect the response times of short flows because small window sizes at slow-start phase cannot trigger fast retransmission. Instead, of the lost packets relies on retransmission timeout (RTO) with minimum RTO value of 3 seconds [4], [1], [16] which is significantly longer than short flows transfer time.

However, if the network is congested, then acknowledgments will take longer to arrive back to the source and slow-start will be actually slow. Delayed acknowledgments are said to be popular in many TCP implementation today. Using delayed acknowledgments the sender transmits single segments and waits for the corresponding ACK. The amount of time required by the congestion window to reach the advertised window for delayed acknowledgments is increased as given in Equation 2.4.

$$Slowstarttime = 2RTT * log_2(W) \tag{2.4}$$

For every incoming packet arriving at the receivers does not generate immediate acknowledgments rather wait for the second segment or delayed acknowledgment timer before transmitting an acknowledgment. Acknowledgments for data sent, or lack of acknowledgments, are used by senders to infer network conditions between the TCP sender and receiver. Coupled with timers, TCP senders and receivers can alter the behavior of the flow of data. This is more generally referred to as congestion control and/or network congestion avoidance.

2.2.2 Congestion Avoidance

In the Congestion Avoidance phase, when a retransmission timer expires or reception of duplicate ACKs occurs, this implicitly signals the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into slow start mode. If congestion was indicated by duplicate ACKs, the fast retransmit and fast recovery algorithms are invoked for more information see [11], [17], [18]. As data is received during Congestion Avoidance, the congestion window is increased. However, slow start is only used up to the

halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window.

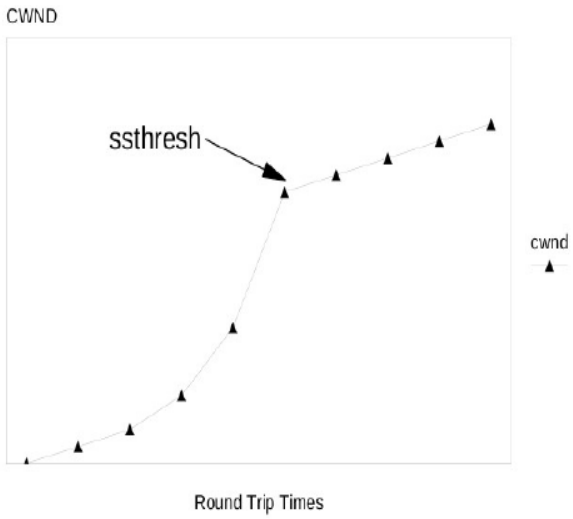


Figure 2.6: TCP slow start and congestion avoidance behavior in action

2.2.3 Fast Retransmission

When a duplicate ACK is received, the sender does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgment. Typically no more than one or two duplicate ACKs should be received when simply out-of-order conditions exist. If however, more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost [17]. The TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fact that the receiver had enough time to send three duplicate ACKs.

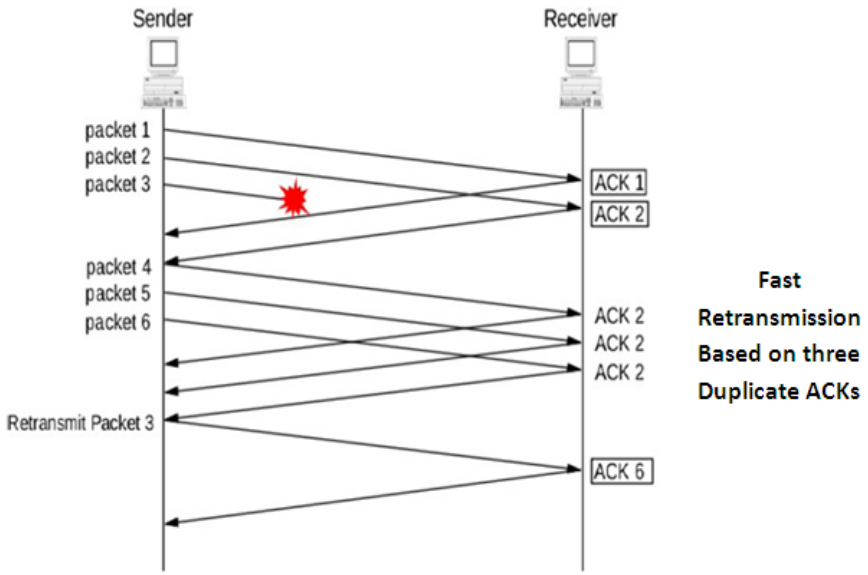


Figure 2.7: Fast Retransmission triggered by 3 DUP ACKs

Upon receipt of three duplicate ACKs as shown in Figure 2.7, the sender does not even wait for a retransmission timer to expire before retransmitting the lost packet. Fast retransmission eliminates about half the coarse-grain timeouts which yields approximately 20% [19] in throughput improvement. However, retransmission does not eliminate all the timeouts due to small window sizes at the source.

2.3 Related Work

In this Section, we reviewed related work on mainly Internet traffic and clearly review recent work on using high initial congestion window.

2.3.1 Internet Traffic Characteristics

Measurement studies have shown that Internet traffic exhibits the mass disparity phenomenon [20], [21]]. Analysis of TCP/IP packets source and destination addresses typically shows that the distribution of packet traffic among hosts is highly nonuniform. A common

observation is that 10% of hosts account for 90 % of traffic. The majority of the flow traffic is flows are short of less than 15KB, which is approximately 10 packets The short flows in the Internet traffic are mainly web traffic originating from the users responses while long flows are known to originate from peer-to-peer applications

2.3.2 Packet Size Distribution

The packet sizes in bytes traversing the Internet have a bimodal distribution. About 50% of the packets carry the maximum number of data bytes permitted by the *maximum transmission unit (MTU)* parameter defined for a network interface. About 40% of packets are small of sizes 40 bytes this is because of the prevalence of (header-only) TCP acknowledgment packets for data received [22]. The remaining 10% of packets are somewhat randomly scattered between the two extremes, based on how much user data remains in the last packet of a multipacket transfer. Packet arrivals occur independently at random times, with a well defined average rate. More formally, the inter-arrival times between events in a *Poisson process* are exponentially distributed and independent, and no two events happen at exactly the same time.

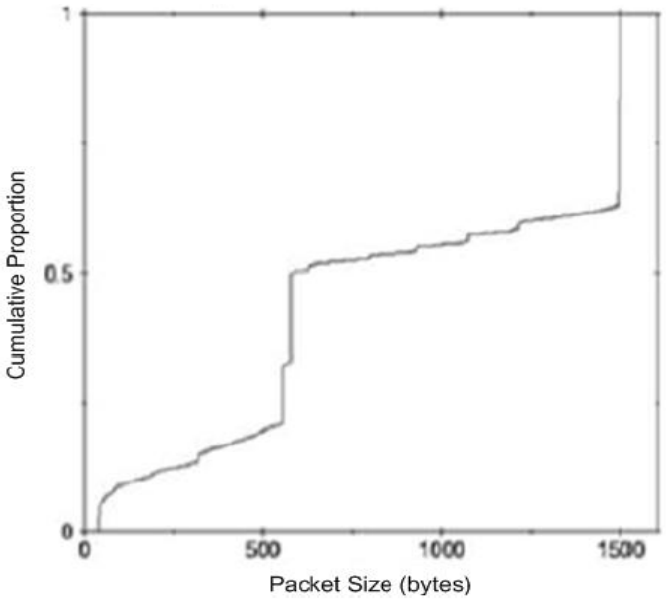


Figure 2.8: Packet Size Distribution

Figure 2.8 shows the percentage of packets present for each packet size. From this it can be seen that over 50% of packets are very small, under 100 bytes

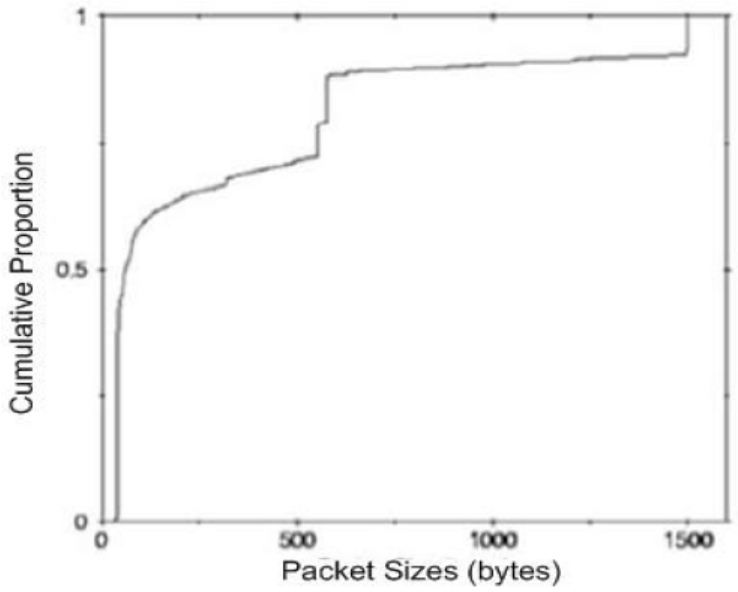


Figure 2.9: Data Distribution

Figure 2.9 shows the data that each packet size is responsible for. From this it can be seen that while maximum sized, 1500 byte packets only make up approximately 10% of packets, they carry approximately 40% of all data.

2.3.3 Bursty Property of Internet Traffic

Detailed studies of Internet traffic show that the packet arrival process is bursty, rather than Poisson. That is, rather than having independent and exponentially distributed inter-arrival times, Internet packets arrive in mass [21] This bursty structure is due in part to the data transmission protocols. The result is that queuing behavior can be much more variable than that predicted by a Poisson model. Given this finding, the value of the simple (Poisson) network traffic models used in network performance studies is doubtful. This realization has motivated recent research on network traffic modeling.

Although the packet arrival process is not Poisson, there is strong evidence that the session arrival process is Poisson. That is, Internet users seem to operate independently and at random when initiating access to certain Internet resources. This observation has been noted for several network applications. For example, work [21] found out that a Poisson arrival process effectively models the session arrival process when they use a time-varying rate. Similarly, the process is effective for modeling user's requests for individual web pages on a web server

2.4 The Need to Speed up Internet

The Internet today has become so popular and so does its networked heavy multimedia applications which have placed heavy demands on the network in terms of throughput and responsiveness. The world currently experiences accelerated growth of Internet connection speeds now at 1.7mbits/sec on average with very high broadband (estimated at 2 mbits/sec) adoption levels of over 50%. Over 95% of Internet traffic is carried by TCP of which the majority of traffic are short flows of less than 10KB [3], [12]. Despite all these advancement in terms of speeds, new technologies and heavy applications such as Maps, Youtube and browsers that had devised means of opening up to six TCP connections in one single domain mostly to boost start-up performance during web page downloads [7], [3]. Speeding up the Internet has remained so crucial in internetworking protocols.

Related work in [18], [23] [24] have shown great effort in speeding up the Internet rather than increasing the initial window. Some provide a solutions in such a way that Internet flows' response time can be improved and they have been classified into two categories namely; 1) based on modifying the existing congestion control protocols such as in [18] SPDY known as "SpeeDY an application-layer protocol for transporting content over the web and basically designed for minimal latency. 2) On scheduling and active queue management schemes in routers. *Least Attained Service (LAS)* scheduling and its inherent queue management scheme has been shown to improve the response time of short flows at a congested access link by speeding up the slow start phase and avoiding packet losses [20]. Similar work done in [23] relies on the feedback information from additional routers to

improve performance in terms of reduced packet loss and fairness.

2.4.1 High Initial TCP Congestion Window

The earliest proposal on increasing TCP's initial congestion window dates back more than a decade ago [11], [12] mainly to speed up the Internet. TCP initial window has remained unchanged since 2002 [7] to only 4KB yet the use of a high initial window of 10 segments will benefit huge bandwidth applications. Recent work on larger initial window [7], [8], [3] mainly focuses on increasing the TCP's initial window from four(4) segments to at least ten (10) segments. It is observed that high initial window is good for high-speed connections in reducing latency by over 10% on the network [3], [12]. Simulation studies on high TCP initial window have also shown an improvement in the response time of short flows which have led to the adoption of initial window size of 4KB that is less than 3 segments. However, simulations studies did not consider the realistic distribution of Internet traffic [11], and only considered the highest window size of 4KB. The upper bound for the TCP's initial congestion window can be given in Equation 2.5.

$$initcwnd = min(4 * MSS; max(2 * MSS; 4380)) \tag{2.5}$$

The web has become much more popular now, leading to significant changes to Internet traffic distributions. The Internet traffic measurement studies have shown that Internet flow size distributions are highly skewed, this means it constitutes of many short flows (about 99% of the flows) that contribute to less than 50% of the bytes, and a tiny fraction of the largest flows (i.e less than 1 %) contributes to nearly half of the bytes [3], [21], [22]. The observation of many short flows in the Internet is likely to persist due to the popularity of web and a plethora of existing and emerging web based applications or services. For instance, search service presents results using short flows, more than 90% of which are less than (10KB), and about 90% of HTTP objects from top 100 to 500 web sites are less than 16KB [3] as shown in Figure 2.10.

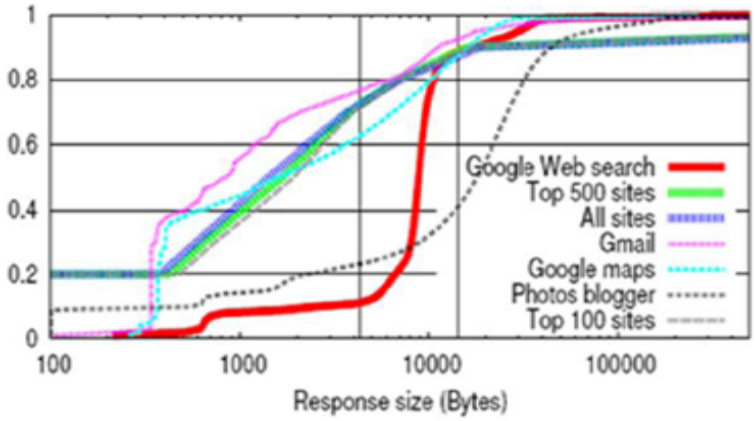


Figure 2.10: CDF of HTTP of Response Sizes [adopted from [3]]

Web users naturally require low response time due to interactive nature of web browsing. However, the existing Internet's congestion control protocols do not consider the required urgency of short flows transfers, neither do they take into account the heavy-tailed flow size distribution of Internet traffic. Short flows are transferred during the slow start phase of the TCP protocol which is known to be dominated by connection's round-trip time. A newly established connection has no idea of congestion state on the path and the slow start phase is equivalent to probing the unknown path.

2.4.2 TCP Timeouts and Retransmissions

It has been shown [19] that *AIMD* is a necessary condition for TCP congestion control to be stable. Because the simple congestion control mechanism involves timeouts that cause retransmissions therefore it is important that hosts have an accurate timeout mechanism. However, standard TCP sender maintains two state variables, *smoothed round-trip time (SRTT)* and *round-trip time variation (RTTVAR)* [4], [25], [26]. Until a RTT measurement has been made for a packet sent between the sender and receiver, the sender sets RTO to three seconds.

When the first RTT measurement is made, the host sets $SRTT = R$, $RTTVAR = R/2$,

then retransmission timeout is set as a function of average RTT and standard deviation of RTT. However, TCP hosts only sample round-trip time once per RTT using coarse-grained clock as in Equation 2.6.

$$RTO = SRTT + max(G; 4RTTVAR) \tag{2.6}$$

Where G denotes the clock granularity estimated at less than 100 ms. When a subsequent RTT measurement are made, the host computes RTTVAR and SRTT given in Equation 2.7.

$$\begin{aligned} RTTVAR &= (1-\beta)RTTVAR + \beta(SRRT - R), \\ SRTT &= (1-\alpha)SRRT + \alpha R \end{aligned} \tag{2.7}$$

Where α = *1/8* and β= *1/4*, as recommended in [4]. Therefore combining the two variables, a TCP sender sets its value of RTO according to Equation 2.8 as

$$RTO = max(minRTO; SRTT + max(G; 4RTTVAR)) \tag{2.8}$$

2.4.3 TCP Minimum RTO

While the basic RTO algorithm computation is still based on a minimum RTO of 3 seconds, IETF has a retransmission timeout policy for standard TCP that limit the minimum RTO to at least 1 second [7], [26]. The major motivations for this lower bound are to protect TCP from spurious retransmission timeouts and extensive idle periods. However, there has been two main limitations of reducing the RTO, namely: 1) *the Clock Granularity* at 500 ms for most operating systems at that time (1988), though modern studies estimates the value of granularity to be much lower than 500 ms since research has shown that finer granularity less than 100 ms perform much better than coarse granularity [25].

For instance, if the RTT equals the clock granularity, then the timeout may falsely expire before the acknowledgments arrival at the server, 2) *the Delayed Acknowledgments* usually

set to 200 ms in common setups, in this case assume an ACK is delayed for more than the current TCP-RTO the timer would expire spuriously.

Google has recommended reducing initial RTO to 1 second [27]. There is indeed, need to revisit congestion protocols to see how they can be re-engineered to improve the average response time of Internet flows. Ideally, it is desirable that short flows do not experience packet losses, and that their transfer times are not influenced by unnecessary round-trip times.

If a TCP data segment is lost in the network, a receiver will never even know it was once sent. However, the sender is waiting for an acknowledgment for that segment to return. If an acknowledgment does not return, the senders retransmission timer expires which causes a retransmission of the segment. If however the sender had sent at least one additional segment after the one that was lost and that latter segment is received correctly, the receiver does not send an acknowledgment for the latter, or out-of-order segment. The receiver can not acknowledge out-of-order data instead it must acknowledge the last contiguous byte it has received in the byte stream prior to the lost segment.

In this case, the receiver will send an acknowledgment indicating the last contiguous byte it has received. If that last contiguous byte was already acknowledged, we call this a duplicate ACK [28], [29]. The reception of duplicate acknowledgments can implicitly tell the sender that a segment may have been lost or delayed. The sender knows this because the receiver only generates a duplicate acknowledgment when it receives other out-of-order segments.

2.5 Hierarchical Link Sharing

In this section, we explain the use of link-sharing mechanisms in packet networks. Hierarchical link-sharing allows multiple agencies, protocol families, or traffic types to share the bandwidth on a link in a controlled fashion [30]. The main goal of link-sharing is that each class with adequate demand should be able to acquire approximately its allocated bandwidth in a time interval during congestion. In this research we use link sharing to allow traffic types

to share the bandwidth of the single bottleneck link as illustrated in Figure 2.11.

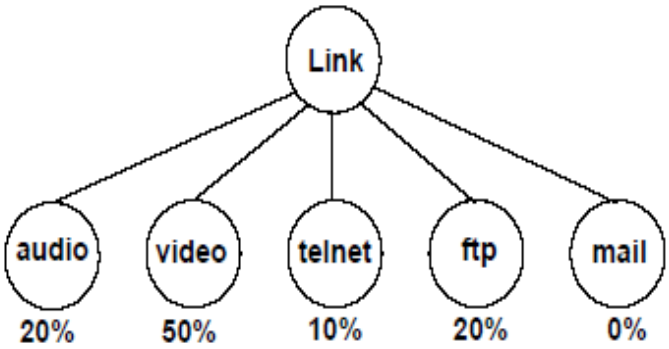


Figure 2.11: Illustration of Bandwidth Link sharing

Figure 2.11 illustrates link-sharing bandwidth allocated between service classes on a slow (such as 64 kbps link). Each class is expressed as a percentage of the overall link bandwidth). These link-sharing allocations could be either static or permanently assigned by the network administrator) or dynamic (varying in response to current conditions on the network, according to some predetermined algorithm). For a class with a link-sharing allocation of zero, such as the mail class in Figure 2.11, the bandwidth allocated to this class is determined by the other scheduling mechanisms such as active queue mechanisms configured at the gateway [31]. We note that the link-sharing mechanisms do not fully provide any bandwidth to this class during congestion [32].

2.5.1 HTB Scheduler Algorithm

Hierarchical Token Bucket (HTB) is a class based packet scheduler similar to *Class Based Queuing (CBQ)* [33]. HTB algorithm supports hierarchical link sharing, traffic prioritization and bandwidth borrowing between classes. HTB is one of the packet scheduling algorithms embedded in the Linux operating system kernel for network traffic control.

The algorithm HTB is designed basing on the token bucket theory [34], [31]. It's design can be viewed as comprising of Token bucket filters (TBFs) placed in a tree-like structure to

form the bandwidth sharing hierarchy. This hierarchy consists of a root class, several parent classes and associated leaf classes as shown in Figure 2.12.

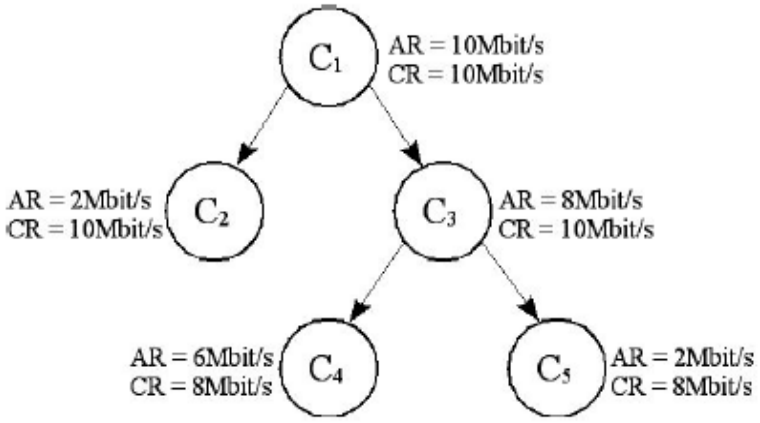


Figure 2.12: Illustration of HTB hierarchy

The top-most TBF is the root class (does not have a parent class associated with it). TBFs placed lower in the hierarchy form parent classes, while those at the end of the hierarchy are known as leaf classes. Parent classes are used to give a guideline on how excess bandwidth is to be shared among child classes attached to them. While for the leaf TBF controls a single class of traffic that may contain one or more traffic flows.

Note that each class in HTB is associated with an assured rate (AR), a ceiling rate (CR), and a priority (P). Due to unstable network conditions, the actual rate of each class can be derive as actual rate (Rc) of a class (c) as given in [31].

$$Rc = min(CRc, ARc + Bc) \tag{2.9}$$

Based on equation 2.9, a class can have an actual rate at least equal or greater than its assured rate (ARc), varying up to its ceiling rate (CRc), depending on the amount of

bandwidth borrowed from other classes (Bc). The borrowed bandwidth depends on the traffic flow in other classes as well as their associated priority.

2.6 Conclusion

In this Chapter, we basically reviewed literature on TCP congestion and control. We also briefly justified the need to have a high TCP initial window. In the next Chapter, we explain emulation method used in networking for setting up a testbed.

Chapter 3

Emulation of Congested Network

In this Chapter, we present the architecture of our testbed and give a brief discussion of the network emulator used in our experiments.

3.1 Introduction

To study the performance of the IW10, we needed to emulate a congested network emphasizing a bottleneck link. To find out how the IW10 perform over different traffic patterns, an experimental testbed is built. We also present Network Emulator as described as *NetEm* in Section 3.3, used to model a congested link. We then generate long-lived TCP, UDP and a mixture of UDP and TCP referred as background traffic. IW10 Measurements of short flows are taken under the above background traffic patterns, we also present results of the IW10 performance in the next Chapter.

3.2 Experimental Testbed

In this Section, we define our testbed on the simple dumbbell topology shown in Figure 3.1. We also recognize that this topology is a limited one, but the behavior of standard TCP on this topology is well studied and so it provides a natural starting point. Our setup consists of five (5) Linux boxes and two (2) Ethernet switches. The one (1) server in the middle uses Netem to emulate bandwidth and delay on the link.

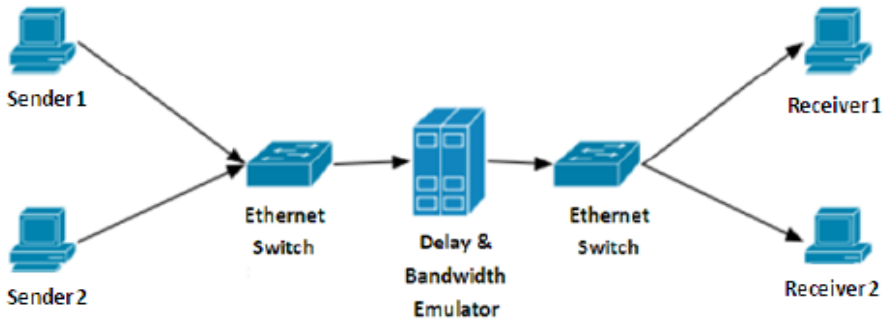


Figure 3.1: Experimental Setup

The other 4 servers are grouped into two (2) server/client pairs. Each pair may generate either a unidirectional long-lived flow using *Iperf* or a bidirectional short lived flow using *Netperf* software tools. Long-lived flows are used to emulate bulk data transfer like FTP while short-lived flows emulate short request/response traffic like Web.

Table 3.1: Parameters used in Experimentation

Parameters	Lower Link	Fast Link
Bandwidth	64 kbps	1 Mbps
Round-trip time	200 ms	200 ms
Queue length	40 packets	500 packets
Test duration	3600 sec	600 sec

All tests were conducted on an experimental testbed. High-end computers were connected to switches to form the branches of a dumbbell topology. All sender and receiver machines used in the tests have identical hardware and software configurations as shown in Table 3.2. The emulator runs on Linux software, it is also configured as our router between two LANs. Apart from the network emulator, all other computers run an upgraded version of the Linux 2.6.6 kernel. Flows are injected using *iperf* and *netperf* into the testbed. In order to obtain a good representation of the run-to-run variability in performance metrics, each individual tests were repeated at least three (3) times and the mean value was taken.

In this Chapter, we conduct two sets of experiments; 1) those that measure TCP performance for short flows under a single initial window value i.e IW3 or IW10 where each case is run separately, 2) those measuring performance under hybrid windows i.e IW3 and IW10 running concurrently.

3.3 Network Emulator (NetEm)

In this Section, we briefly describe our Network Emulator(NetEm). NetEm is a software tool that is used to model a bottleneck link between two real hosts. In our research, we use Netem as one of the most efficient tool to emulate delay and bandwidth over a single link. Previous studies such as [35],[36] have highly recommended the use of emulation in networking as a most effective method of evaluating real TCP implementations over very slow networks or even satellite channels.

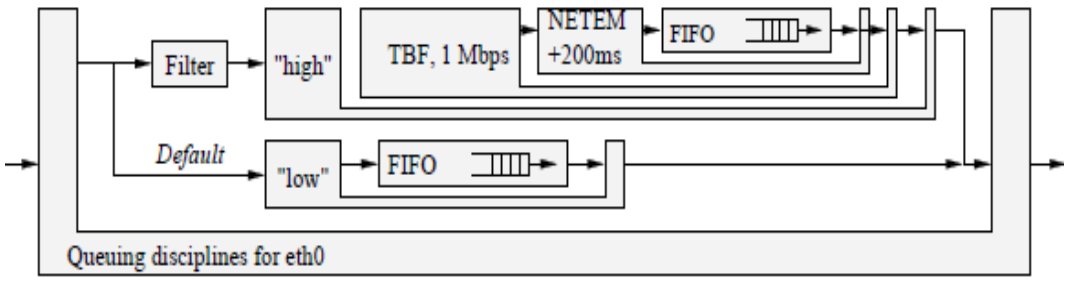


Figure 3.2: Netem implemented in Linux box

Figure 3.2, represents an implementation of NetEm in a typical Linux box. Netem is configured by the help of a Traffic Control program given in [37] and uses a FIFO queuing discipline by default for the outbound queue but other queues can be used [38]. In our experiments, we involve a number of tools used in evaluating TCP performance as described in [39]. However, we choose Wireshark measurement tool discussed in the next Section and also outline some of the tools used in our experimentation given in Table 3.2 including the purpose of each tool and the systems on which it runs.

3.4 Wireshark Measurement Tool

In this Section 3.4, we present a brief overview of wireshark. The tool is loaded on the emulator to collect data by capturing packets on listening onto a receiving interface and interactively browse the traffic running on a computer network [40].

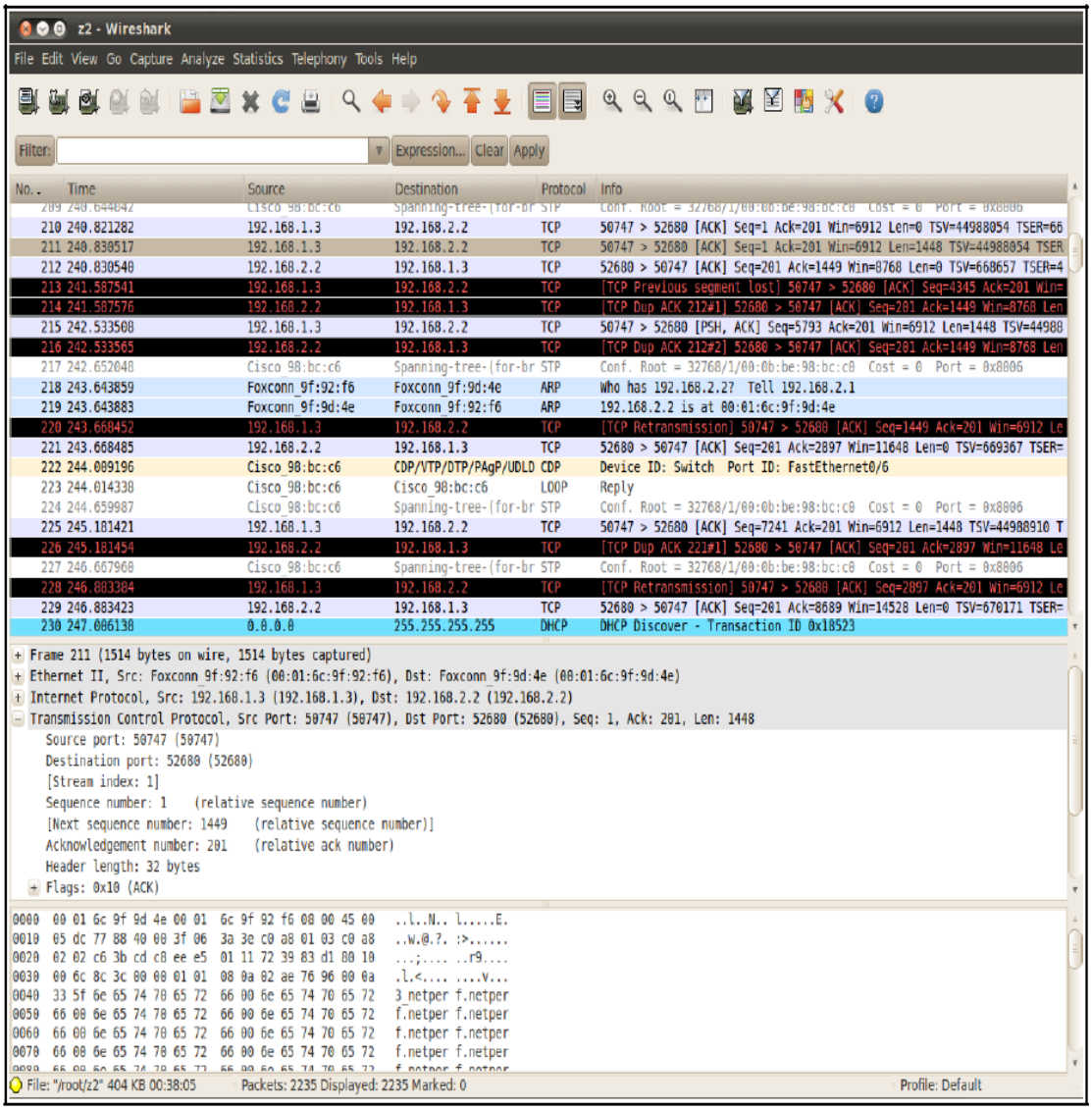


Figure 3.3: Wireshark captures from a Live Network

Figure 3.3 shows a snap shot of a wireshark trace file. The data collected from the trace files can be stored and then loaded for analysis.

Table 3.2: Software Tools used in Experimentation

Tool	Purpose	Running on
iperf	Long-lived TCP/UDP flow generation	both sender and receiver
netperf	Short-lived TCP flow generation	both sender and receiver
iproute2	TCP initcwnd and initrwnd configuration	both clients and server
wireshark	Network packet measurement tool	on the router
netem	Delay configurations	only at the emulator or router
htb	Bandwidth allocation	only sender
ifcong	Buffer setting	only linux router
tc	Traffic control	all linux boxes

3.5 Performance Metrics

To gauge the performance of high TCP initial congestion window size, we focused on the following TCP performance metrics defined in next Section.

3.5.1 Throughput

This is the only straightforward way to gauge the performance of TCP and can be defined literally as the time required to transfer a given number of bits of data.

$$Throughput = \frac{B_T}{FCT} \tag{3.1}$$

We take the average throughput for individual flows over a given period of time and can be computed as in Equation 3.1

$$Avg.Throughput = \frac{B_T}{FCT * N_F} \tag{3.2}$$

3.5.2 Flow Completion Time

This metric FCT stands for flow completion time in seconds, B_T is total number of bytes where as total number of flows is given as N_F. FCT can be defined as the amount of time

required to transfer a given number of bytes in a single flow. For instance; the amount of time required for a flow of 10 packets to elapse.

$$Avg.FCT = \frac{T_{FCT}}{N_F} \tag{3.3}$$

We measure T_{FCT} as total flow completion time for all flows in a specific duration of time, and then compute the average flow completion time by dividing by total number of flows as in Equation 3.3.

3.5.3 Packet Loss

This simply implies the number of packets dropped by the network, it is used in our study as an indicator of the aggressive of initial windows. Packet loss can be approximated by using the number of retransmissions which must be always greater than or equal to the number of packet drops.

3.6 Congestion Patterns

In this section, we define three congestion patterns i.e. 1) *Constant congestion*; where congestion rate is flat on the link and only induced by UDP background traffic. We also clearly state in our study that constant congestion is used to illustrate a constantly congested link. 2) *Varying congestion* is induced by TCP traffic only. 3) *Mixed congestion* is induced by mixed traffic such as TCP and UDP traffic. We also consider congestion level as a percentage of congestion on a link induced by each of the congestion patterns.

3.7 Conclusion

We have discussed the architecture of our testbed, we also defined the different parameters and tools used in our experimentation. In the next Chapter, we present the detailed empirical results .

Chapter 4

The Performance Evaluation

In this Chapter, we discuss the empirical results that show the performance of IW3, IW10 and the hybrid initial windows in different congestion patterns.

4.1 Introduction

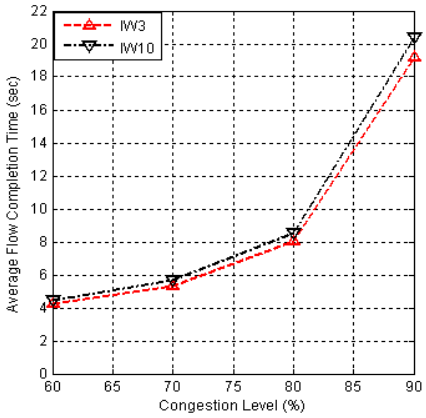
Network performance evaluation is a very important element in measuring the behavior of a links. We discuss the performance of single initial windows and hybrid window size in Section 4.2 and 4.3 respectively under different congestion patterns. In these Sections, we also measure throughput ratios for single and hybrid initial windows under the similar conditions. Finally, we demonstrate the effect of initial windows on packet loss and retransmissions.

4.2 Performance of Single Initial Windows

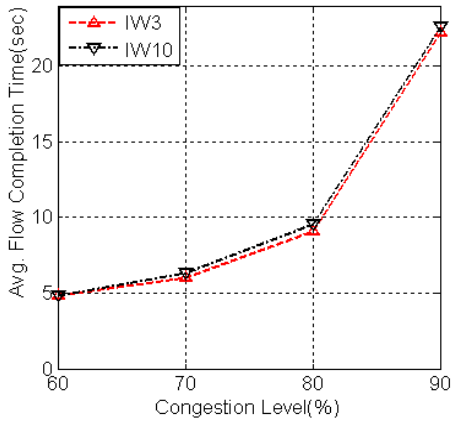
In this Section, we analyze the effect of using a single value of TCP initial window subjected to higher levels of congestion of about 60% and above while observing the behavior of short flows with different sizes i.e 9, 10, 21 and 22 packets.

4.2.1 IW3 Vs IW10 Under Constant Congestion

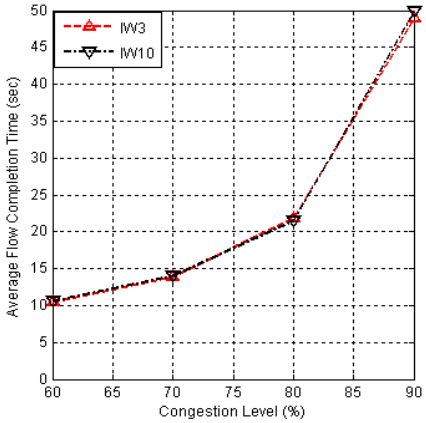
We evaluate the initial windows performance on short flows with 9 packets against 10 packets, then 21 packets against 22 packets under high constant congestion (approximated at 60% congestion level) on the link.



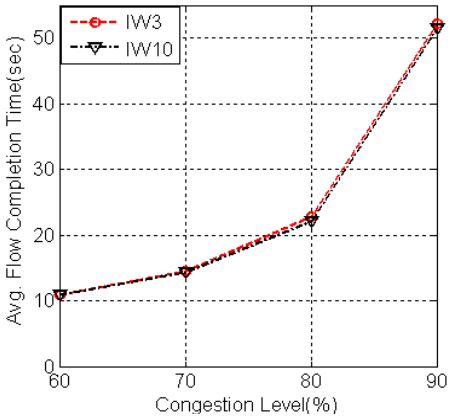
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets



(d) Short flows of 22 packets

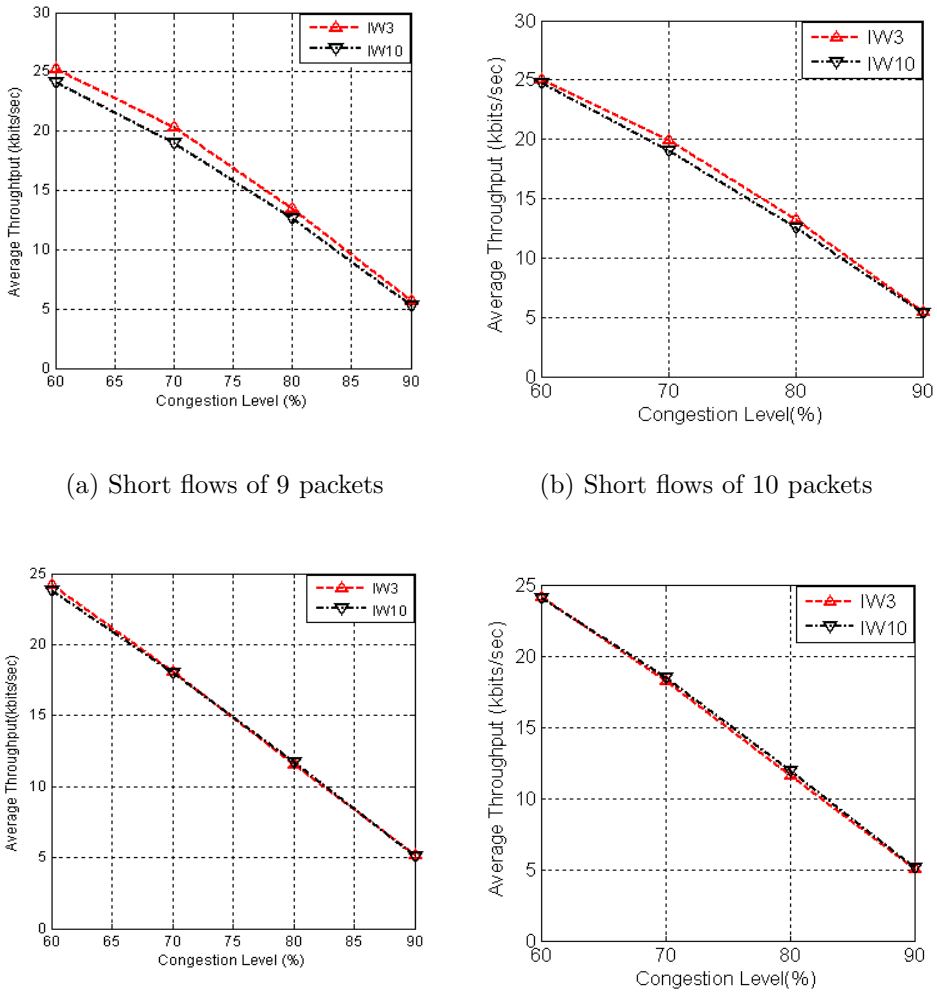
Figure 4.1: Single IW: Average completion time under constant congestion

Figure 4.1 shows average flow completion time of short flows with 9, 10, 21 and 22 packets over a network link that has been subjected to a high and constant level of congestion. We observe that as congestion increases average flow congestion time of short flows increases regardless the flow sizes. This is due to the fact that during congestion, TCP tends to reduce on the transmission rate whenever congestion or packet drops are detected thus delaying flows.

Note that there are two phases for all figures. Phase 1 ranges from congestion levels 60% to 80%, here we observe that average flow completion time gradually increases with increase in congestion level, whereas phase 2 is above 80% congestion while average flow completion

time drastically increases with increase in congestion level. We explain this pattern in such a way that the bandwidth is still sufficient to accommodate short flows in Phase 1. In the contrary, in phase 2 the bandwidth is insufficient to accommodate the short flows. Also note in Figures 4.1(a) and 4.1(b), the average completion time of flows under TCP with IW10 is slightly longer than completion time for TCP with IW3 whilst Figures 4.1(c) and 4.1(d) show average completion time for TCP with IW10 is slightly shorter than completion time for TCP with IW3 at high congestion. We further observe that the differences in IW3 and IW10 flow completion times reduces as flow sizes increase, on the other hand for short flows with 21 and 22 packets, the IW3 and IW10 flow completion time differences are so negligible. This is due to the fact that short flows carrying 21 and 22 packets adds more congestion compared to shorter flows of less than 10 packets into the pipe. Short flows with 21 and 22 packets also exhibit bursty property which is much more tolerable by larger IW10 as opposed to smaller windows. In Figure 4.1, we observe that average completion time is reduced when a flow size is reduced by one packet. We note that doubling the flow size, also doubles the average completion time of a short flow.

TCP with IW10 exhibits latency longer than TCP with IW3 for constant congestion pattern for short flows less than 10 packets. The average delays are longer for flows of sizes 10 and 22 packets compared to sizes 9 and 21, this is to be expected since the former flow sizes transfer one more window than that latter flows. For instance at 90% congestion figure 4.1(a) and 4.1(b) shows that average latency is less than 20 seconds for flow with 9 packets and above 20 seconds for short flows with 10 packets. This same phenomenon can be observed for flow sizes with 21 and 22 packets. In fact, the performance difference in terms of average delays is higher for flows with 9 packets than with 10 packets. The performance is observed to fade away as flow sizes increase.



(a) Short flows of 9 packets

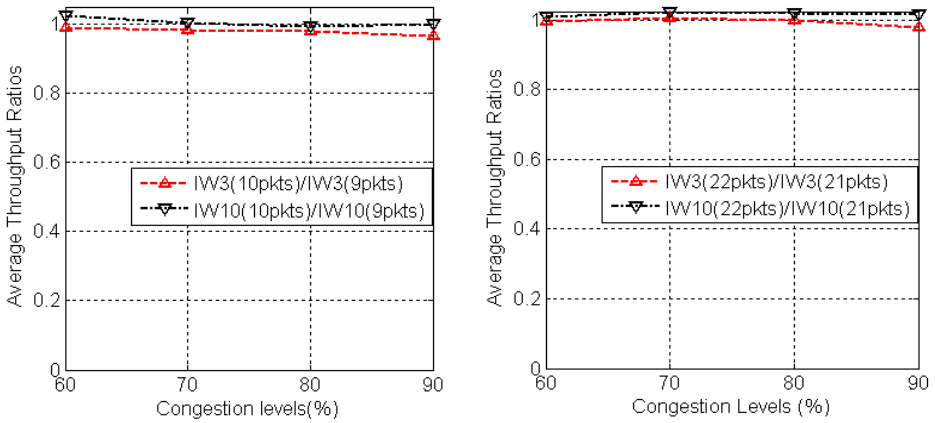
(b) Short flows of 10 packets

(c) Short flows of 21 packets

(d) Short flows of 22 packets

Figure 4.2: Single IW: Average throughput under constant congestion

Figure 4.2 illustrate the average throughput of short flows of sizes 9, 10, 21 and 22 packets respectively. We observe as congestion levels on the link increases, throughput reduces steadily. Recall that flow completion time is inversely related to throughput. In Figures 4.2(a) and 4.2(b) of short flows with 9 and 10 packets respectively, we observe IW3 throughput of short flows with 9 and 10 packets slightly higher than IW10 throughput. This is due to the fact that IW3 experiences shorter completion time as reflected in Figures 4.1(a) and 4.1(b). In contrast, short flows of 21 and 22 packets, indicate IW10 with a slightly higher amount of throughput than IW3 throughout all congestion percentages. However, the difference in IW3 and IW10 throughput is too negligible. Conclusively, IW3 is seen to favour flows of less than 10 packets where as IW10 favours larger flows of 21 and 22 packets.



(a) Throughput ratio of 10 to 9 packets

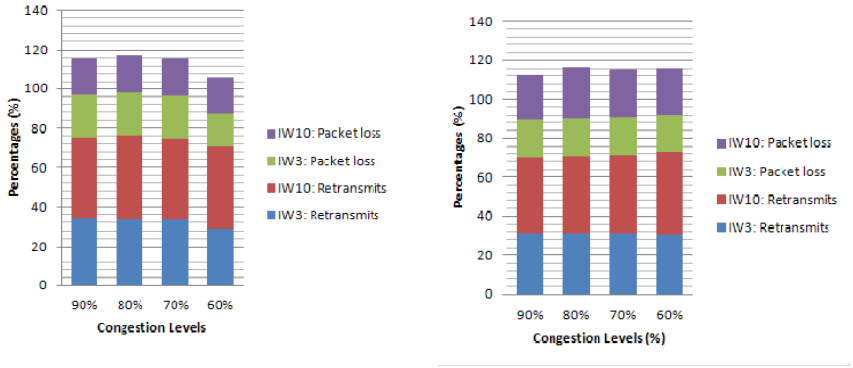
(b) Throughput ratio of 22 to 21 packets

Figure 4.3: Single IW: Throughput ratios under constant congestion

In Figure 4.3, we considered throughput ratios of short flows with 10 packets to 9 packets and short flows of 22 packets to 21 packets respectively. Figure 4.3(a) compares IW3 and IW10 throughput ratios of short flows with 10 and 9 packets. The ratios of IW3 (10 pkts) to IW3 (9 pkts) are less than 1 while those of IW10(10 pkts) to IW10(9 pkts) are greater than 1. This phenomenon in Figure 4.3(a) is observed to be the same as for ratios of IW3 (22 pkts) to IW3(21 pkts) and IW10(22 pkts) to IW10(21 pkts) respectively in Figure 4.3(b). When using IW3 to send short flows of 10 packets as opposed to 9 packets under constant congestion on the link, we observe that average throughput computed by sending 9 packets is much greater than average throughput computed from sending only 10 packets. Ideally, this is not expected from a non-congested link as stated by previous work where by the more packets sent the better the throughput. Previous work only concentrated on short flows with 10 packets. Therefore, in our study we recommend flows of 9 packets as opposed to 10 packets when using IW3 under constant congestion on the link. We also tested IW3 by sending short flows of 21 and 22 packets and we observed that 21 packets has a slightly higher throughput than 22 packets under similar link conditions.

On the other hand, when using IW10 we observe that IW10 perform much better with flows of 10 packets as opposed to 9 packets under constant congestion which occupies between 60% to 90% of the link. This implies that the pipe still has sufficient bandwidth of about 40% to 10% bandwidth available to accommodate 10 packets much more than 9 packets as

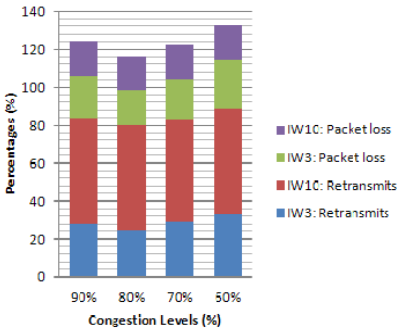
it expected in a non-congested link. IW10 was also tested with flows of 21 and 22 packets and realized 22 packets have slightly higher throughput than 21 packets when using IW10 under constant level of link congestion.



(a) Short flows of 9 packets

(b) Short flows of 10 packets

(c) Short flows of 21 packets



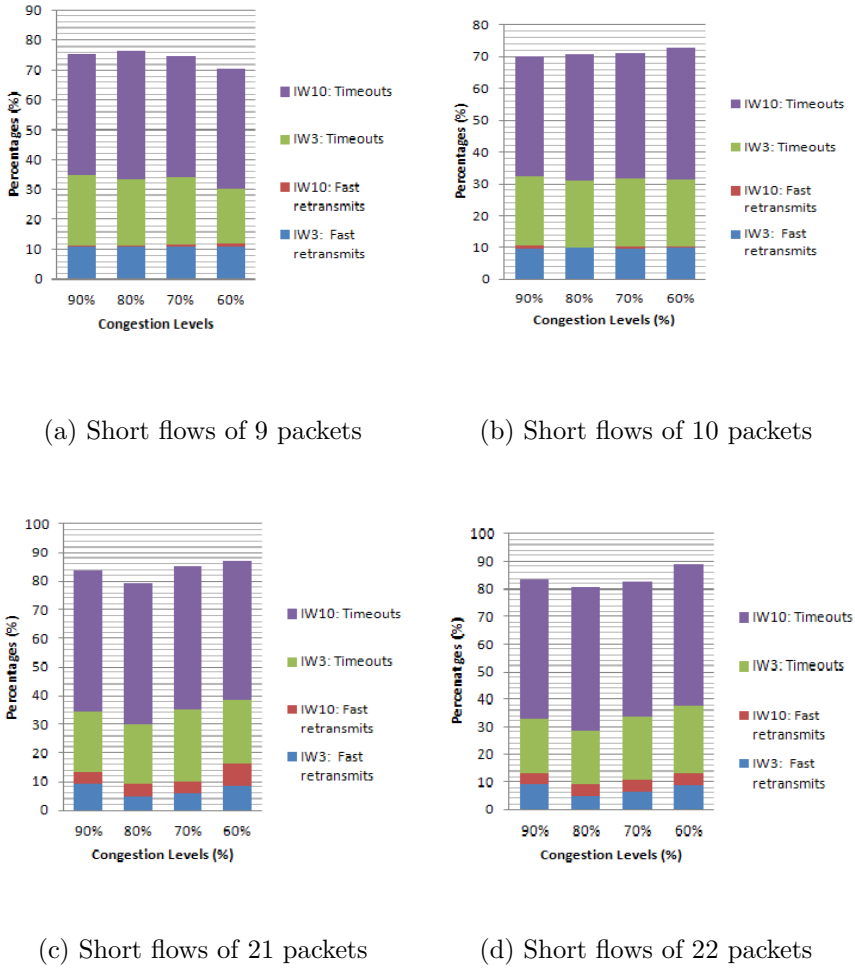
(d) Short flows of 22 packets

Figure 4.4: Single IW: Packet losses and Retransmissions under constant congestion

Figure 4.4(a) shows less packets dropped by IW10 but with a higher average latency as opposed by IW3 under short flows with 9 packets. While under short flows with 10 packets, IW10 packet losses are more than IW3 and also with a higher average latency. This implies that losses for IW10 experience timeouts and less fast retransmits as in Figure 4.5. On the other hand, as an extra packet is added on the flows under constantly congested links, IW10 with 10 packets has more packet loss than short flows of 9 packets. While IW3 packet losses reduce with short flows of 10 packets as opposed to 9 packets.

While for short flows with 22 and 21 packets, we observe that these bursty flows are

penalize by IW3 in terms severe packet losses as opposed to IW10 dropping less packets in bursty traffic. We also note that retransmissions are twice as much as packet losses in both IW3 and IW10 regardless the flow sizes. We also must note that IW10 retransmissions are too high for all cases, this implies that there are multiple retransmits that are also lost many times until timeout as in Figure 4.5. We conclude by stating the fact that results for short flows depend on the exact number of the packets considered. Ideally, flows with 10 packets are transfered by more than one window but only one window by IW10. This is not the case for constant congestion pattern, under this pattern TCP with IW10 penalizes itself due to higher bursts sent at the same time. We also note that the number of short flows injected to the network fills in the pipe causing short flows to experience some loss of packets. This loss is severe at IW10 than IW3.



(a) Short flows of 9 packets

(b) Short flows of 10 packets

(c) Short flows of 21 packets

(d) Short flows of 22 packets

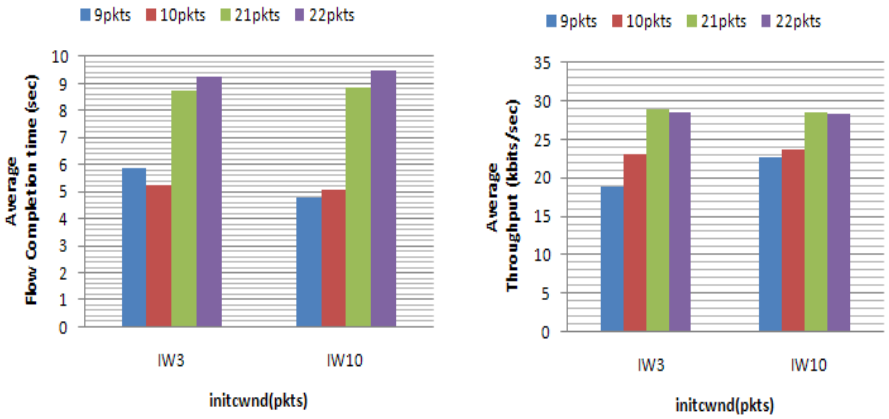
Figure 4.5: Single IW: Timeouts and Fast retransmits under constant congestion

Figure 4.5 show initial window effect of short flows on timeouts and fast retransmits under very high constant congestion. Under short flows of 9 packets and 10 packets as illustrated in 4.5(a) and 4.5(b) respectively, clearly indicate IW10 Timeouts are twice as much as IW3 timeouts irrespective of the flow sizes. However, IW10 timeouts are higher with flows of 10 packets than flows with 9 packets. This is also attributed from the severe retransmissions IW10 experiences thus leading to multiple timeouts. While fast retransmissions by IW10 are so negligible (approx. at 0.005%) compared those by IW3 which are significantly more (approx. at 10%). For 21 and 22 packets, we also note quite a similar pattern. However, in this case, IW10 timeouts are more than timeouts recorded by flows of less than 10 packets. We also observe that TCP with IW3 still records a greater percentage of fast retransmissions as opposed to TCP with larger window of IW10 which has significantly smaller number of fast retransmissions. This is not expected in a non-congested link with a larger initial window sizes.

Under constant congested links, TCP with IW3 performs better for short flows less than 10 packets as opposed to TCP with IW10. Previous work show different results because they didn't consider high constant congestion pattern in their experiment. For instance, bursts of IW10 doesn't lead to fast retransmits compared to IW3 case as it would be expected. But the larger the window the higher the likelihood of going into fast retransmits. Our results do not show this likelihood after considering a constant congestion pattern.

4.2.2 IW3 Vs IW10 Under Varying Congestion

Considering varying congestion which is also regarded as less congestion since TCP's rate is unstable unlike constant congestion induced by UDP flow, we note that Long TCP flows induce about 68% congestion on a slow link with flows less than 10 packets and about 58% congestion with short flows of 21 and 22 packets. This implies that as the flow size increases, more congestion grows on the link. In this study, we also note that the higher the link utilization by competing Long flows, the lower the throughput of short flows.



(a) Flow Completion Time (sec) (b) Throughput(kbits/sec)

Figure 4.6: Single IW: Flow completion time and throughput under varying congestion

Figure 4.6, illustrate flow completion time and average throughput of short flows of sizes 9, 10, 21 and 22 packet under unstable levels of congestion on the link. We clearly observe IW3 average short flow completion time of 10 packets is much shorter than short flows with 9 packets, this is why Figure 4.6(b) exhibits a higher throughput by IW3 for flows with 10 packets as opposed to 9 packets. While under IW10, flows with 10 packets have slightly more flow completion time than 9 packets thus leading to a flow lower throughput. We also evaluate performance of short flows of 21 packets against 22 packets using IW3 and IW10 respectively. Flows with 21 packets record a lower flow completion time than 22 packets irrespective of the initial window size under consideration. This implies that as a single packet is added to the bursty flow size, the throughput reduces as congestion increases.

We also further note that IW10 is a better performer than IW3 for short flows of 9 and 10 packets under this non-constant congestion pattern in terms of shorter completion time and higher throughput in Figures 4.6(a) and 4.6(b) respectively. However, this is not the case for short flows of 21 and 22 packets, where we clearly indicate that IW3 with shorter completion time and higher throughput. This is due to less congestion and IW10 being very aggressive to the sufficient available bandwidth as opposed to IW3 which carries fewer packets hence lower throughput. We also remember that short flows of less than 10 packets leave about 40% available bandwidth for IW10 to transmits all the packets in one window, where as

flows of 21 and 22 packets leave about 30% available bandwidth, which is still sufficient for IW10 to outperform IW3 under varying congestion pattern

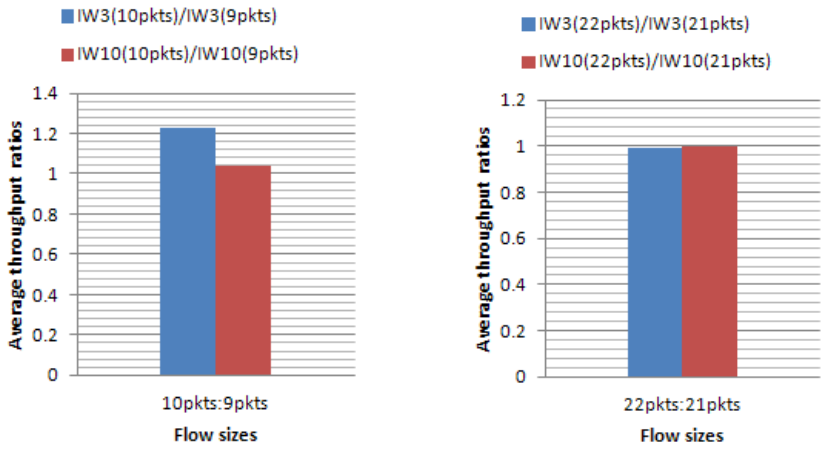
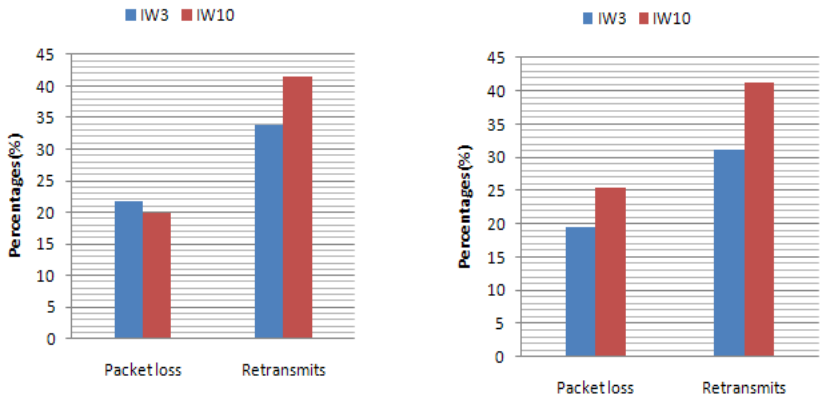


Figure 4.7: Single IW: Throughput ratios under varying congestion

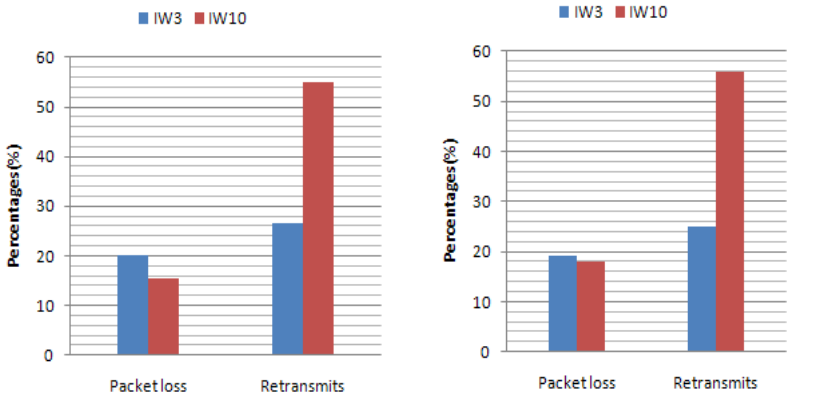
Figure 4.7, illustrate throughput ratios of IW3(10pkts) to IW3(9pkts) compared to IW10(10pkts) to IW10(9pkts) under a variant congestion pattern. We note that both ratios are greater than 1. This implies that as the flow size is increased from 9 to 10 packets, we observe that TCP with IW3 has much higher throughput by sending short flows with 10 packets as opposed to 9 packet. Where as TCP with 10 packets indicates slightly more throughput by sending 10 packets as opposed to 9 packets. On the other hand, IW3(22pkts) to IW3(21pkts) ratio is less than 1. This implies that TCP with IW3 favors short flow of 21 packets as opposed to 22 packets. This pattern is similar to IW10 under the same flow sizes but the performance difference is very negligible. That's why ratio values are very close to one as shown in Figure 4.2.2.

Finally, we note TCP with IW10 throughput values always higher than IW3 values in a less congested link for short flows of at least 10 packets as expected and mostly favors 10 packets as opposed to 9, but this is contrary for short flows with 21 and 22 packets where by TCP with IW3 is a better window than IW10 and favors 21 packets as opposed to 22 packets.



(a) Short flows of 9 packets

(b) Short flows of 10 packets



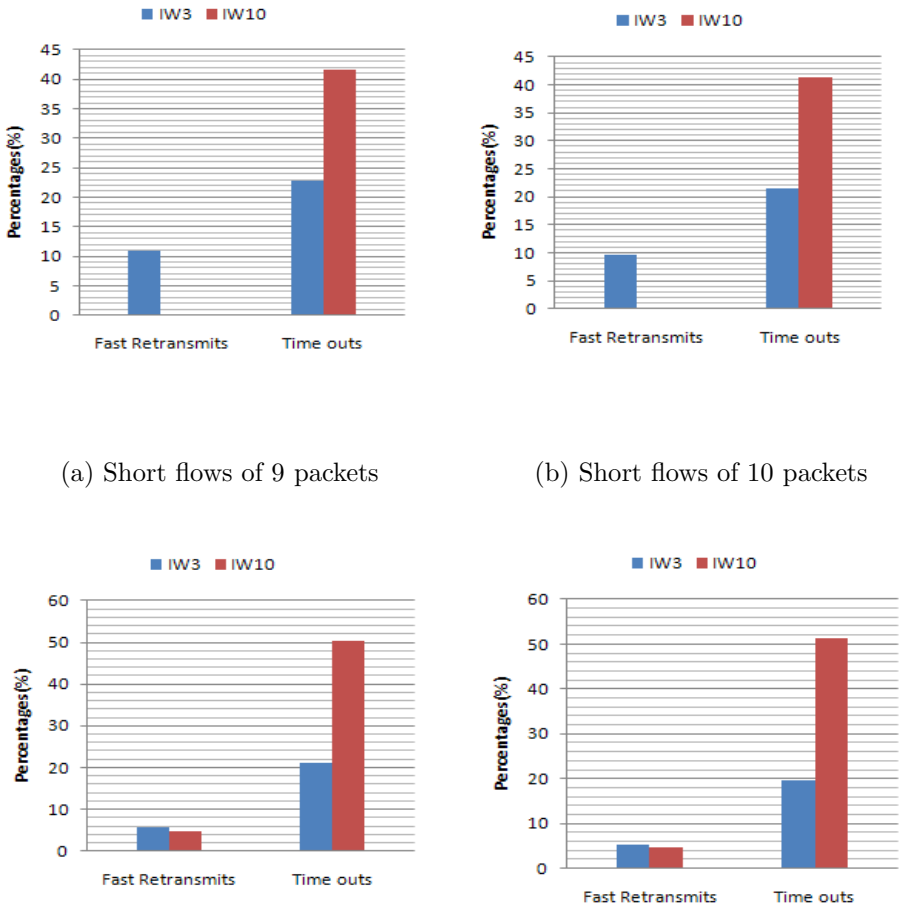
(c) Short flows of 21 packets

(d) Short flows of 22 packets

Figure 4.8: Single IW: Packet loss and retransmits under varying congestion

Figure 4.8, illustrates packet loss and retransmission in varying congestion on the link. We observe short flows with 9 packets under TCP with IW3 drops lesser packets than TCP with IW10, this is because shorter flows with 9 packets are transmitted on the less congested link more comfortably with a larger initial window of IW10 as opposed to smaller windows such as IW3, since smaller windows require multiple transmissions. On the other hand, short flows of 21 packets drop more packets under TCP with IW3 than TCP with IW10. Much as IW10 drops a smaller number of packets, IW10 retransmissions are more than IW3 retransmissions. We also observe that IW3 drops more packets under short flows of 21 packets than 22 packets. While IW10 drops more with flows of 22 packets than 21 packets.

We finally note, regardless the flow sizes, IW10 retransmissions are much more than IW3 retransmits and even more when short flows carry bursty traffic.



(a) Short flows of 9 packets

(b) Short flows of 10 packets

(c) Short flows of 21 packets

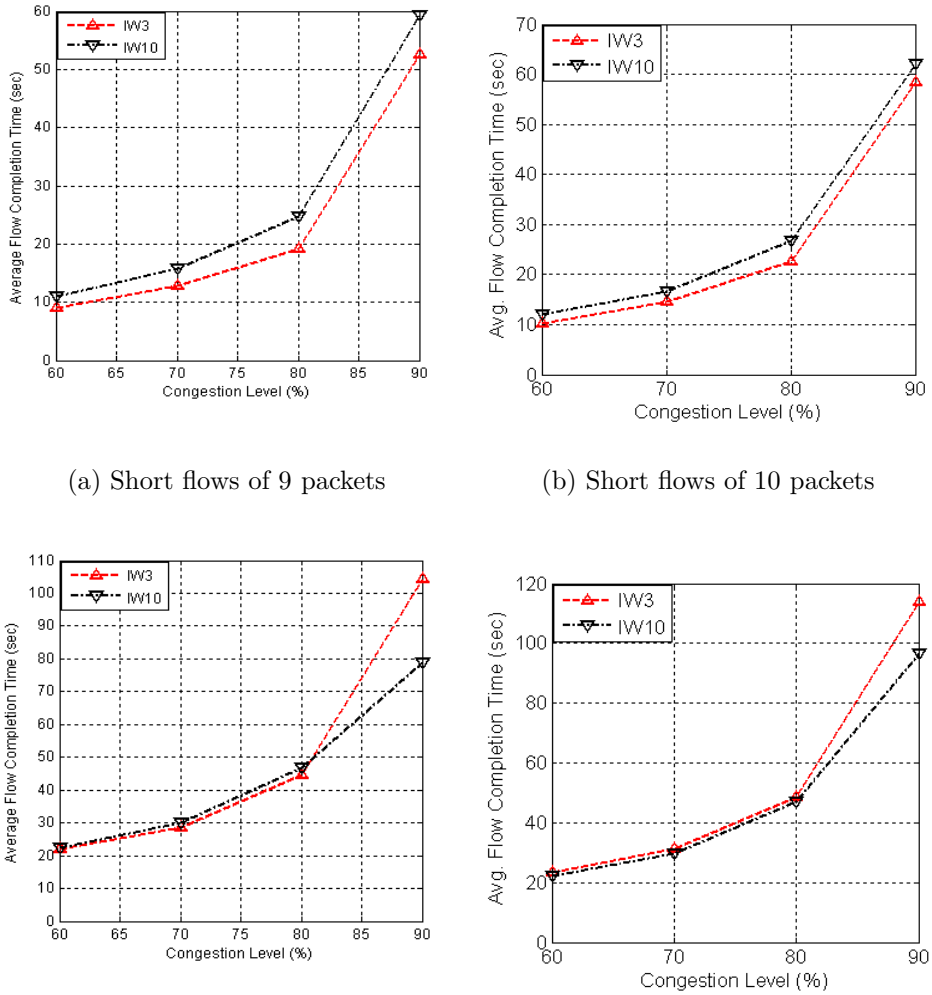
(d) Short flows of 22 packets

Figure 4.9: Single IW: Timeouts and Fast retransmits under varying congestion

Figure 4.9 illustrate the effect of short flows under IW3 and IW10 on timeouts and fast retransmits. We observe short flows under 9 and 10 packets, the number of timeouts by IW10 is almost twice the number of timeouts in IW3. However, IW10 fast retransmissions is very minimal compared to IW3 fast retransmits. While for short flows with 21 and 22 packets respectively, IW10 timeouts in both flow sizes are twice as much as IW3 timeouts. We also observe that IW10 fast retransmissions are less than IW3 retranmits under flows less than 22 packets but not less than 10 packets.

4.2.3 IW3 Vs IW10 Under Mixed Congestion

We evaluate IW10 against IW3 under a mixed congestion pattern on the access link i.e constant plus varying congestion pattern. Figure 4.10 clearly illustrates the performance of short flows of sizes 9, 10, 21 and 22 packets in regards to average flow completion time. Under mixed congestion, we compose a mixture of long TCP and UDP flows as background traffic competing with numerous short flows. The long persistent UDP flows roughly occupies about 90% of the link bandwidth where as long TCP account for only 6% in our mixed congestion. Figure 4.10 shows average flow completion time of different short flow sizes under mixed



(a) Short flows of 9 packets

(b) Short flows of 10 packets

(c) Short flows of 21 packets

(d) Short flows of 22 packets

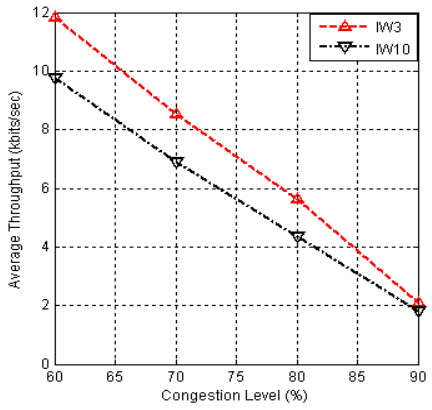
Figure 4.10: Single IW: Average flow completion time under mixed congestion

congestion pattern and we observe as congestion increases average flow completion time

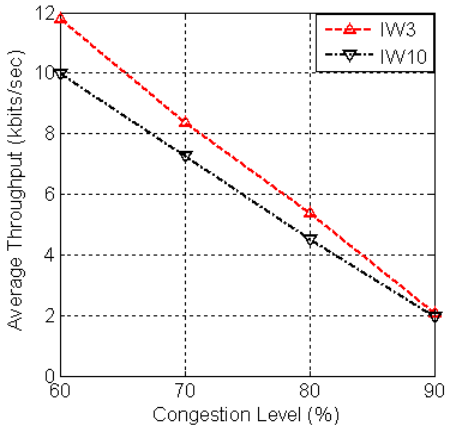
reduces steadily. We also clearly note in Figures 4.10(a) and 4.10(b), IW3 is better in terms of shorter average flow completion time as opposed to IW10 which have longer average flow completion time with short flows of sizes 9 and 10 packets respectively. This implies the general performance in terms of average latency is poorer with 10 than in 9 packets e.g average latency at 90% for IW3 is above 50 seconds for 9 packets and close to 60 seconds for short flows with 10 packets as shown in Figure 4.10(a). However, we observe the performance gap between IW10 and IW3 average completion time narrows as a single packet is added to a flow size. For instance, Figure 4.10(a) shows bigger performance difference between IW3 and IW10 as compared to Figure 4.10(b) which further raises a concern of previous results that only considered short flows with 10 packets.

Where as on the other hand, larger short-flows with 21 and 22 packets shown in Figures 4.10(c) and 4.10(d) respectively, indicate IW10 performance much better in terms of shorter average latency compared to TCP with IW3 average latency when the link is subjected to heavily mixed congestion. Under this congestion pattern, we observe the performance difference of IW3 and IW10 in term of average flow completion time under 21 and 22 packets is minimal especially between 60% and 80% congestion. This is expected since an extra packets sent in a window of 22 packets become a small fraction of the total flow latency. Beyond 80%, the performance gap widens as congestion grows higher while indicating IW10 a better window since it is a larger window which is resistant to bursty traffic flows compared to smaller windows whose performance is detrimental according to the flow size. We generally conclude by noting that IW3 is favoured by short flows of at least 10 packets while IW10 is favoured by short flows with 22 packets under a mixed traffic pattern.

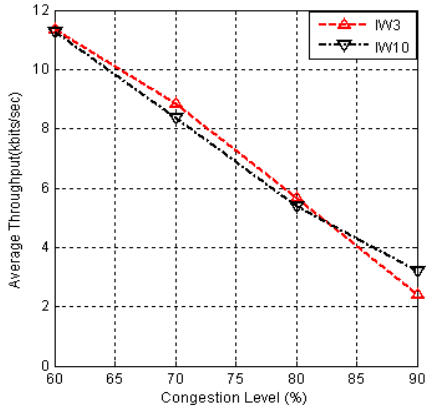
In Figure 4.11 shows the effect of using TCP with IW3 and TCP with IW10 in regarding average flow throughput on the link that has been subjected to mixed congestion. We observe in all the figures, as congestion grows from 60% towards 100%, throughput steadily reduces towards zero. Short flows with sizes of 9 and 10 packets shown in Figures 4.11(a) and 4.11(b) respectively, indicate TCP with IW3 has a higher average throughput compared to IW10 average throughput under mixed congestion. However, the performance difference between IW3 and IW10 throughput values is so wide especially at 60% congestion level but narrows as congestion increases. In fact at 90% congestion level, we observe TCP with IW3 and



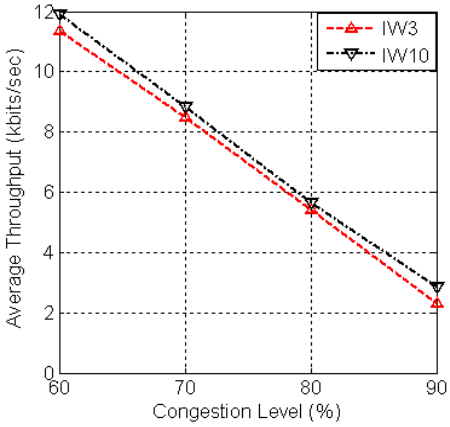
(a) Short flows of 9 packets



(b) Short flows of 10 packets



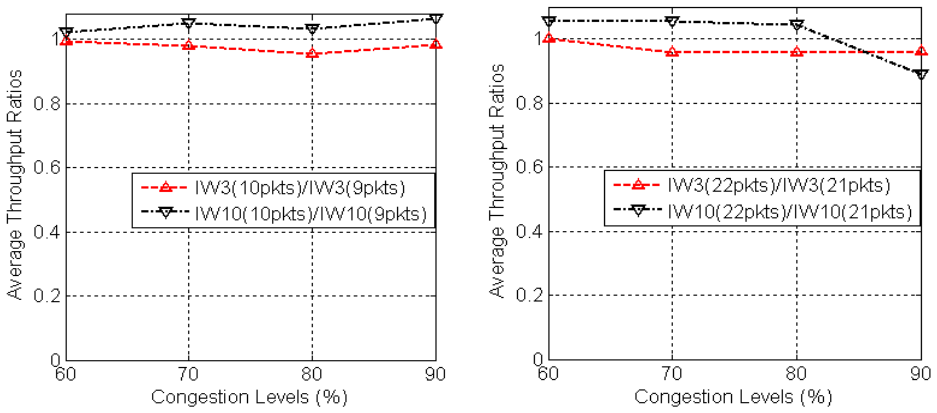
(c) Short flows of 21 packets



(d) Short flows of 22 packets

Figure 4.11: Single IW: Average throughput under mixed congestion

IW10 average throughput values quite similar. While on the other hand, Larger short-flows of 21 and 22 packets in Figures 4.11(c) and 4.11(d) respectively, illustrate a quite different pattern in that average throughput under these flow sizes is much greater in TCP with IW3 as opposed in TCP with IW10 especially between 60% and 80% congestion levels on the link. But beyond 80% congestion the performance pattern changes immediately, and IW3 continues to deteriorate while carrying larger packets of packets.



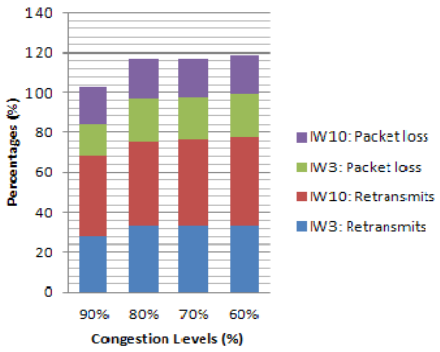
(a) Throughput ratios of 10 to 9 packets

(b) Throughput ratios of 22 to 21 packets

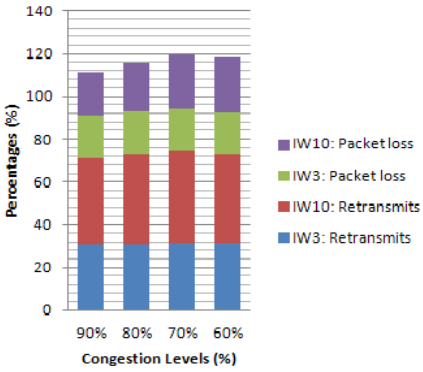
Figure 4.12: Single IW: Throughput ratios under mixed congestion

Figure 4.12 illustrates average throughput ratios of short flows sizes 9 to 10 packets against 21 to 22 packets. Under this figure, we observe that IW3 favours throughput of flows with 9 packets much more than short flows with 10 packets. This is indicated in the throughput ratio of 10 packets to 9 packets, which is less than 1 throughout all congestion levels. While this is contrary for IW10, which favors flows with 10 packets as opposed to 9 packets. This patterns applies for larger flows with 21 and 22 packets as well for IW3 but not IW10. For instance, under very high congestion of about 80% and 90% IW10 alternates the ratios as shown in Figure 4.12(b). We conclude by noting that as an extra packet is added onto a flow with 9 or 21 packets, IW3 is seen to be overwhelmed since the window deteriorates throughput performance as congestion grows heavily on a link.

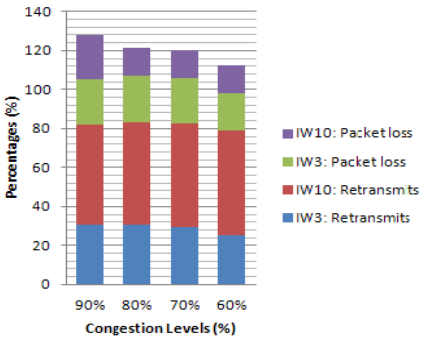
In Figure 4.13, we observe packet losses are much more in short flows carrying 10 packets than 9 packets under mixed congestion. This is due to extra congestion induced by that extra packet sent. We also note TCP with IW10 drops more packets than TCP with IW3 under short flows of 10 packets as opposed to TCP with IW10 which drops less packets than TCP with IW3 under short flows of 9 packets as congestion increases. This is because flows with 9 packets transmitted under IW3 suffer much more resistance as compared to IW10 which is less resistant since most of the packets are carried through the pipe within one window. We also observe that IW10 retransmissions are higher than IW3 retransmits for flows less than 10 packets. On the other hand, short flows with 21 and 22 packets exhibit



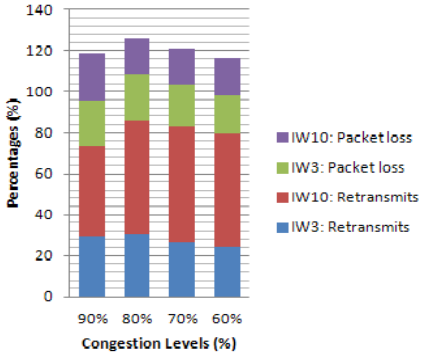
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets

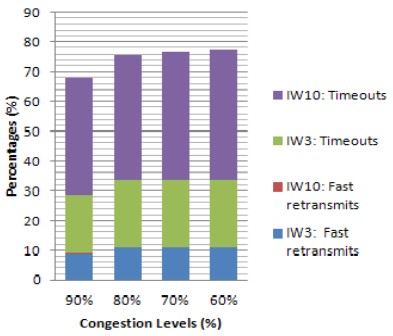


(d) Short flows of 22 packets

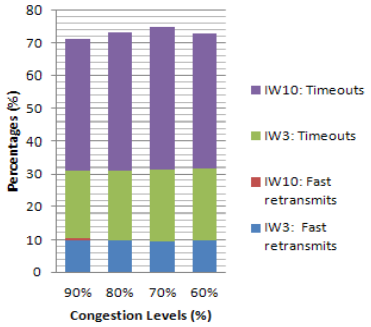
Figure 4.13: Single IW: Packet loss and retransmits under mixed congestion

TCP with IW10 quite more packet losses than IW3 as congestion increases. Finally, we note that IW10 retransmissions are massively higher than IW3 retransmissions for short flows of 21 and 22 packets. This implies that we experience many time packets being retransmitted more than twice especially when TCP detects more packet losses due to congestion.

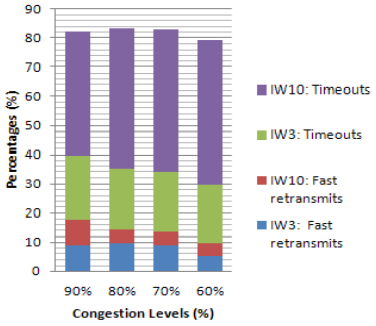
In Figure 4.14, under short flows less than 10 packets, we observe that IW10 timeouts are twice as much as IW3 timeouts. This is mainly depicted by multiple retransmissions (retransmit a packets more than twice) from severe packet losses and delayed acknowledgments which is very common under very highly congested network. Under short flows of 21 and 22 packets, we still maintain a similar pattern for timeouts as that for flows less than 10 packets. However, the difference comes in fast retransmissions where by flows with less than 10



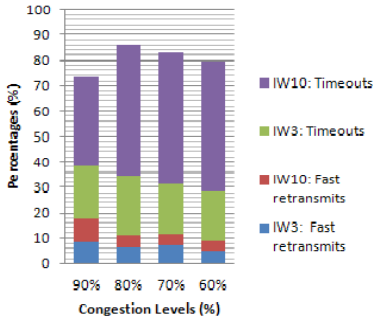
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets



(d) Short flows of 22 packets

Figure 4.14: Single IW: Timeouts and fast retransmits under mixed congestion

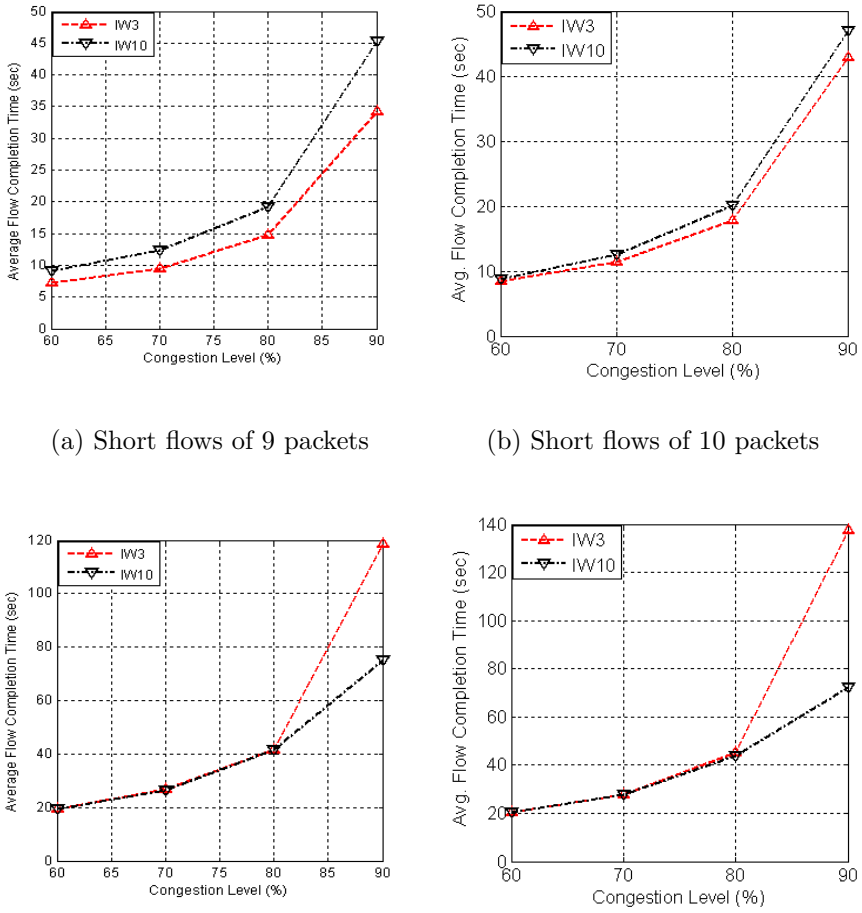
packets record very minimal percentage compared to larger short flows of 21 and 22 packets which records IW10 with smaller percentage (about 5%) as opposed to being negligible as for 10 packets but IW3 records roughly about 10% under bursty flows.

4.3 Performance of Hybrid Initial Window

In this Section, we evaluate TCP performance of a hybrid initial window size of 3 and 10 packets i.e (IW3+IW10) on short flows. We configured half of the traffic under IW3 while the other set to IW10, then capture performance of each individual initial window under heavy congestion.

4.3.1 Hybrid(IW3+IW10) Under Constant Congestion

Figure 4.15 shows average flow completion time against high congestion percentages for short flows of different sizes under constant rate of congestion flow.



(a) Short flows of 9 packets

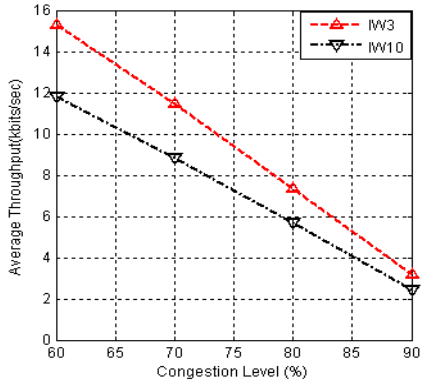
(b) Short flows of 10 packets

(c) Short flows of 21 packets

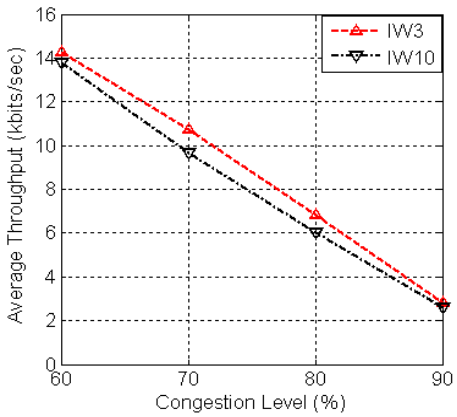
(d) Short flows of 22 packets

Figure 4.15: Hybrid IW: Average completion time under constant congestion

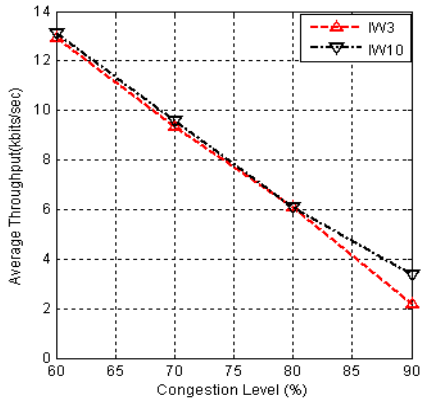
In Figure 4.15, we observe short flows of less than 10 packets, exhibit IW3 with a much shorter time to complete flows than IW10 under constant rate of congestion as illustrated in Figure 4.15(a) and 4.15(b) respectively. While this is contrary to short flows of 22 packets which sends bursty amount of data on the link. under this, IW10 tends to favour larger bursts of packets such as 21 and 22 packets, that is why IW10 indicates insignificant performance difference between 60% and 80% of relatively high congestion. But beyond 80% congestion (extremely high congestion), we note a very big performance difference between IW10 and IW3 average flow completion time. In fact the gap continues to grow as more congestion increases, this is generally because the larger flows with IW3 is severely penalized much more than larger flows with IW10 as shown in Figures 4.15(c) and 4.15(d) respectively.



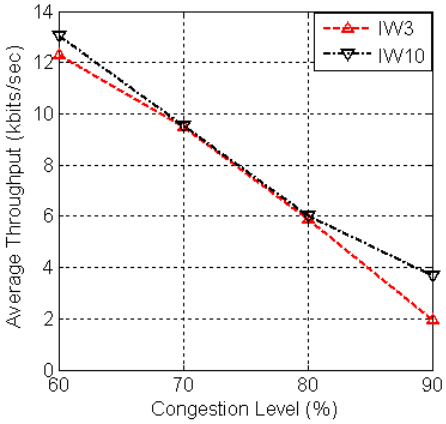
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets

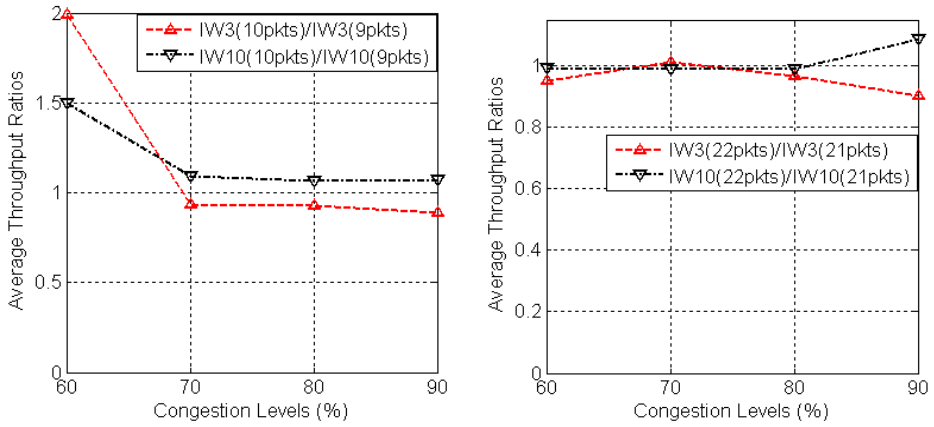


(d) Short flows of 22 packets

Figure 4.16: Hybrid IW: Average throughput under constant congestion

Figures 4.16 shows the performance of the TCP with hybrid initial window sizes of IW3 and IW10, in terms of average throughput of short flows of 9, 10, 21 and 22 packets under constant congestion pattern. For short flows with less than 10 packets, average throughput reduces steadily to zero as congestion increases towards 100% on the link. However, IW10 reduces much more compared to IW3, this is clearly illustrated at each individual congestion level where by IW10 throughput is always lower than IW3 average throughput, simply because of IW3 shortest average completion time clearly illustrated in **??**sub105).

On the other hand, for short flows of 21 and 22 packets exhibits a different pattern with flows less than 10 packets. We observe TCP with IW10 in bursty flows records slightly higher throughput than IW3 average throughput. This can be clearly seen between 60% and 80% congestion levels. Beyond 80% congestion, as congestion grows towards 100%, throughput reduces towards zero. At this point of very high congestion, TCP with IW3 deteriorates much faster to zero as opposed to TCP with IW10 as clearly illustrated in Figures 4.16(c) and 4.16(d) respectively.



(a) Throughput ratio of 10 to 9 packets

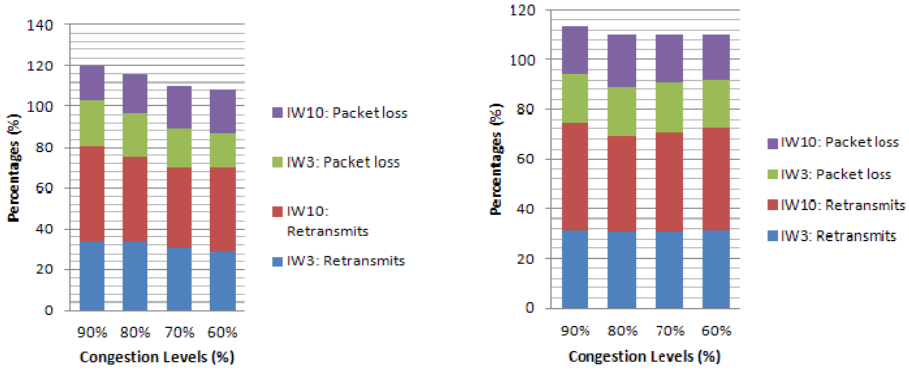
(b) Throughput ratio of 22 to 21 packets

Figure 4.17: Hybrid IW: Throughput ratios under constant congestion

Figure 4.17(a) shows throughput ratios of flow with 10 to 9 packets. Firstly, we observe IW3 and IW10 ratios cross cutting in both figures. Nevertheless, we observe on average that throughput ratios of IW3 appears way below IW10 ratios as congestion continues to grow on the link. For example, below 70% congestion IW3 ratios more than 1 while beyond this point, all IW3 throughput ratios are less than 1, implying that IW3 favours flows with 9

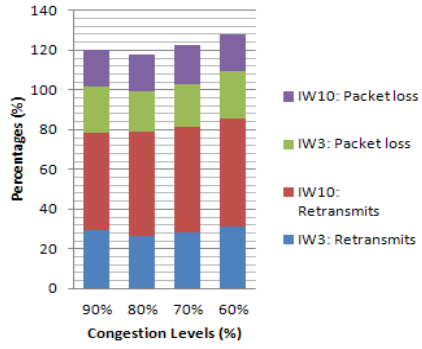
packets in terms of higher throughput as opposed to 10 packets under a constantly congested link. We also note the all IW10 ratios are above 1, this implies that despite the congestion levels flows with 10 packets records better throughput as opposed to flows with 9 packets.

Where as under throughput ratios of flows with 22 to 21 packets illustrated in Figure 4.17(b). Almost all IW3 throughput ratios at all congestion levels are less than 1 and just below IW10 throughput ratios except at 70% where IW3 ratios is slightly higher than IW10 ratios. Finally, we note the IW10 ratios are along ratios of 1 and above, implying that insignificant performance difference in throughput between for flows with 22 and 21 packets but better to send 22 packets than 21 packets . While IW3 ratios are less than 1, meaning it is better to send 21 packets as opposed to 22 packets.

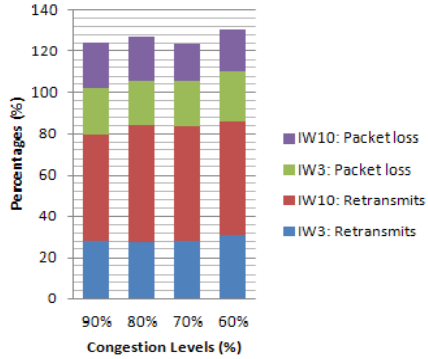


(a) Short flows of 9 packets

(b) Short flows of 10 packets



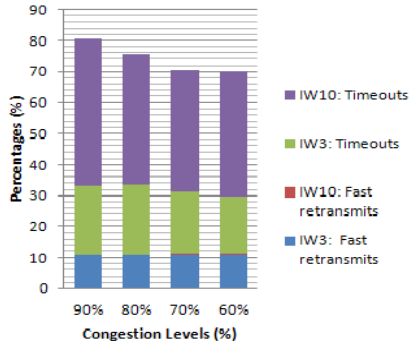
(c) Short flows of 21 packets



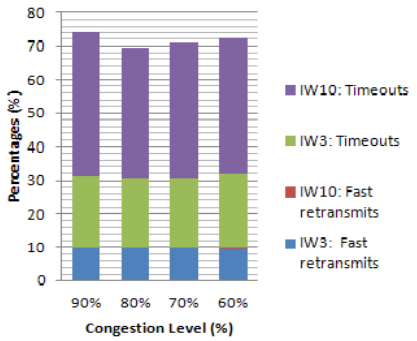
(d) Short flows of 22 packets

Figure 4.18: Hybrid IW: Packet loss and retransmits under constant congestion

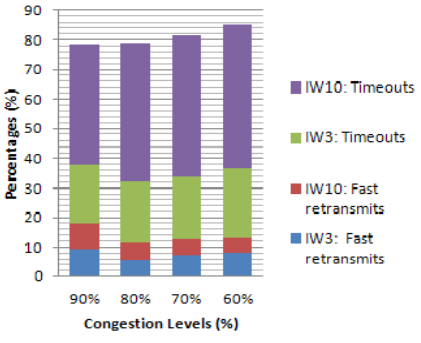
Figure 4.18, illustrate short flows of less than 10 packets under IW3 with a higher percentage of packet loss on a link with congestion of about 70% and above. While below this percentage, we observe IW10 packet loss percentages to be higher than IW10 as shown in Figures 4.18(a) and 4.18(b) respectively. However, for short flows with 21 and 22 packets that carry bursty traffic, indicate we observe IW3 packet loss to be greater than IW10 throughout the congestion levels on the link. Retransmissions in all figures are show that IW10 results are always outstanding much more than IW3 retransmits. In fact the more the packets in a flow, the more the retransmits irrespective of the window size as shown in Figure 4.18.



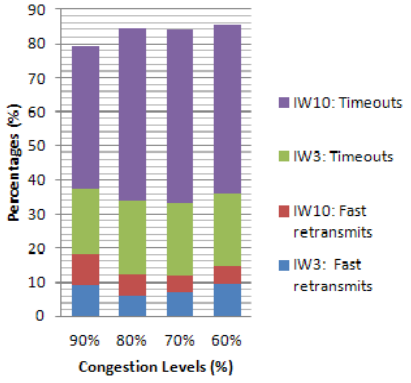
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets



(d) Short flows of 22 packets

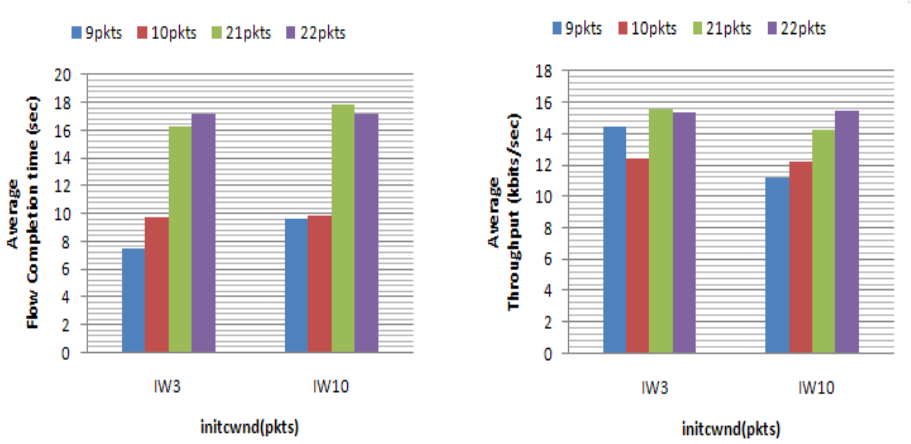
Figure 4.19: Hybrid IW: Timeouts and fast retransmits under constant congestion

Figure 4.19 illustrate the effect of the hybrid window sizes on short flows of different sizes regarding performance in terms of timeouts and fast retransmits. We clearly observe that

short flows less than 10 packets in Figures 4.19(a) and 4.19(b) respectively indicate IW10 timeouts also too outstanding compared to IW3 timeouts. This is because many times packets retransmitted are as a result of timeouts thats why we have as many timeouts as retransmissions as shown in Figures 4.19 and 4.18 respectively. We also note that under very high congestion percentages e.g 80% and above, fast retransmission effect diminish as congestion increases and also it is seen that IW10 fast retransmits are too minimal as opposed to IW3 irrespective of any congestion percentage. These effects are the same even with flows of 21 and 22 packets. However, we also note that under IW10, fast retransmission effect is significant for larger flows as opposed for flows less than 10 packets

4.3.2 Hybrid (IW3+IW10) Under Varying Congestion

Figures 4.20(a) and 4.20(b) represents average flow completion time and throughput for short flows with different sizes under unstable congestion.



(a) Flow completion time (sec) (b) Throughput(kbits/sec)

Figure 4.20: Hybrid IW: Flow completion time and throughput under varying congestion

Figure 4.20(a) illustrate longer completion time by TCP with IW10 compared to IW3 average completion time for short flow sizes less than 10 packets, implying IW3 a better window since it takes shorter time to finish flows which also contributes to higher amounts in throughput as illustrated in Figure 4.20(b). On average, the performance when IW3 is combined with

IW10, indicates IW3 is far much better performer than IW10 when subjected to unstable congestion. While on the other hand, for flows with 21 and 22 packets, indicate that IW3 favors 21 packets instead of 22 packets while this is the inverse with IW10. Generally IW10 is a better window when it comes to bursty flows unlike IW3 which is good for smaller flow sizes

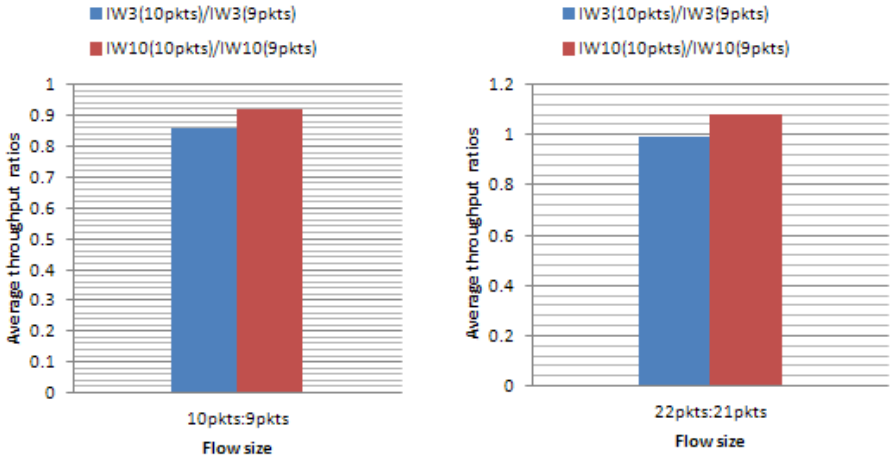
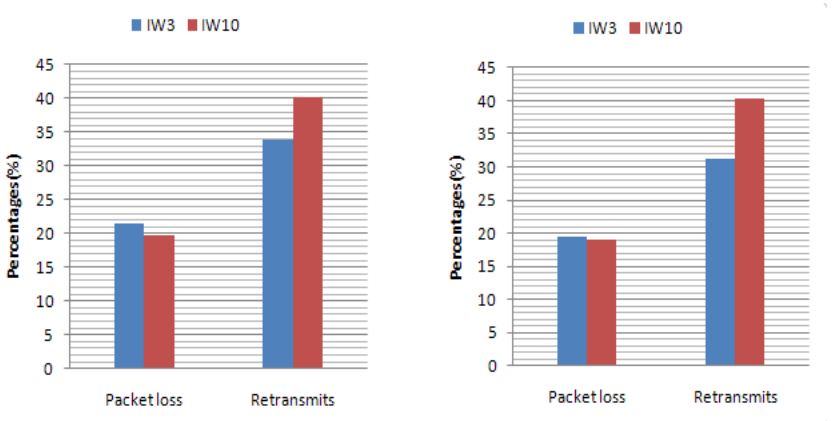


Figure 4.21: Hybrid IW: Throughput ratios under varying congestion

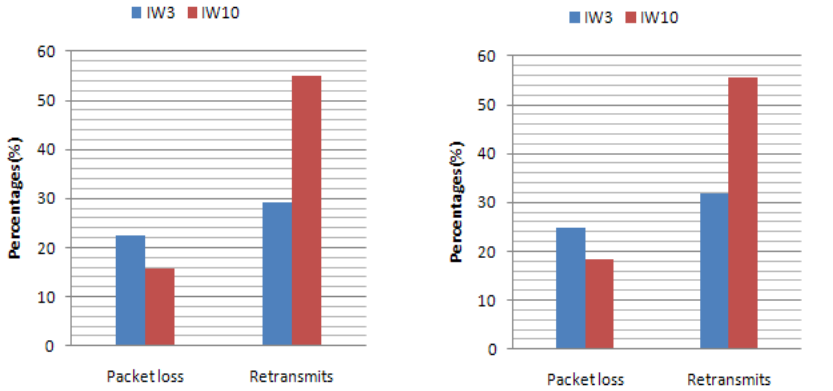
Figure 4.21 illustrates average throughput ratios of flows with 10 to 9 packets and 22 to 21 packets. Under less congestion or unstable congestion, we observe that short flows with 9 packages are favoured by both window sizes as opposed to 10 packets. However, we note that IW10 throughput ratios are greater than IW3 ratios as shown in Figure 4.3.2 and 4.3.2 respectively. However, it is important also note that under larger flows of 21 and 22 packets, all throughput ratios are much more closer or above 1 as opposed under flows less than 10 packets. We conclude that IW10 ratios are higher than IW3 ratios in both cases which implies that, as more and more packets are added to the flow sizes the more the throughput irrespective of the window size as long as the link is less congested.

Figure 4.22 illustrate packet losses with varying flow sizes transmitting packets across a link subjected to unstable or varying congestion. We clearly observe that IW3 drops slightly more packets compared to IW10 for flows less than 10 packets and much more are dropped in bursty flows i.e and 22 packets by IW3 than IW10 as in Figures 4.22(c) and 4.22(d) respectively. We also note that flows less than 10 packets, have outstanding percentages of retransmissions under TCP with IW10 and much more with bursty flows compared to TCP



(a) Short flows of 9 packets

(b) Short flows of 10 packets



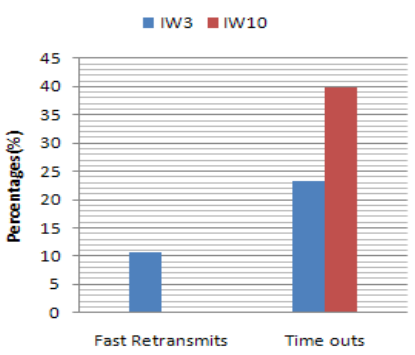
(c) Short flows of 21 packets

(d) Short flows of 22 packets

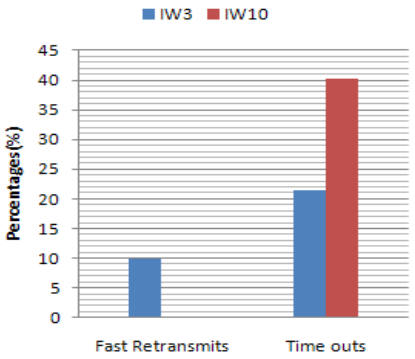
Figure 4.22: Hybrid IW: Packet loss and retransmits under varying congestion

with IW3. This effect is as a result of extra congestion being induced as packets are added to the flow size which therefore accelerates the more packet losses and retransmissions.

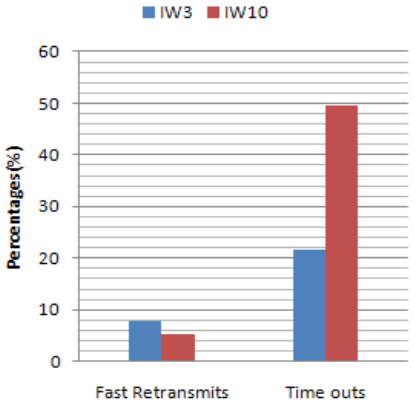
Figure 4.23 illustrates the effect of hybrid initial window size on fast retransmissions and timeouts of packets with unstable congestion on the link. We observe under flow sizes of 9 and 10 packets, TCP with IW10 records very minimal percentages of fast retransmits (approx. 0.05%) unlike TCP with IW3 which indicate significant percentages of approximately 10% as in Figures 4.23(a) and 4.23(b). However, the percentages of packet timeouts under TCP with IW10 are almost twice IW3 timeouts just as it is the case in Figure 4.22 on retransmissions. While for short flows of 21 and 22 packets exhibit quite similar effect. However, we note the



(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets

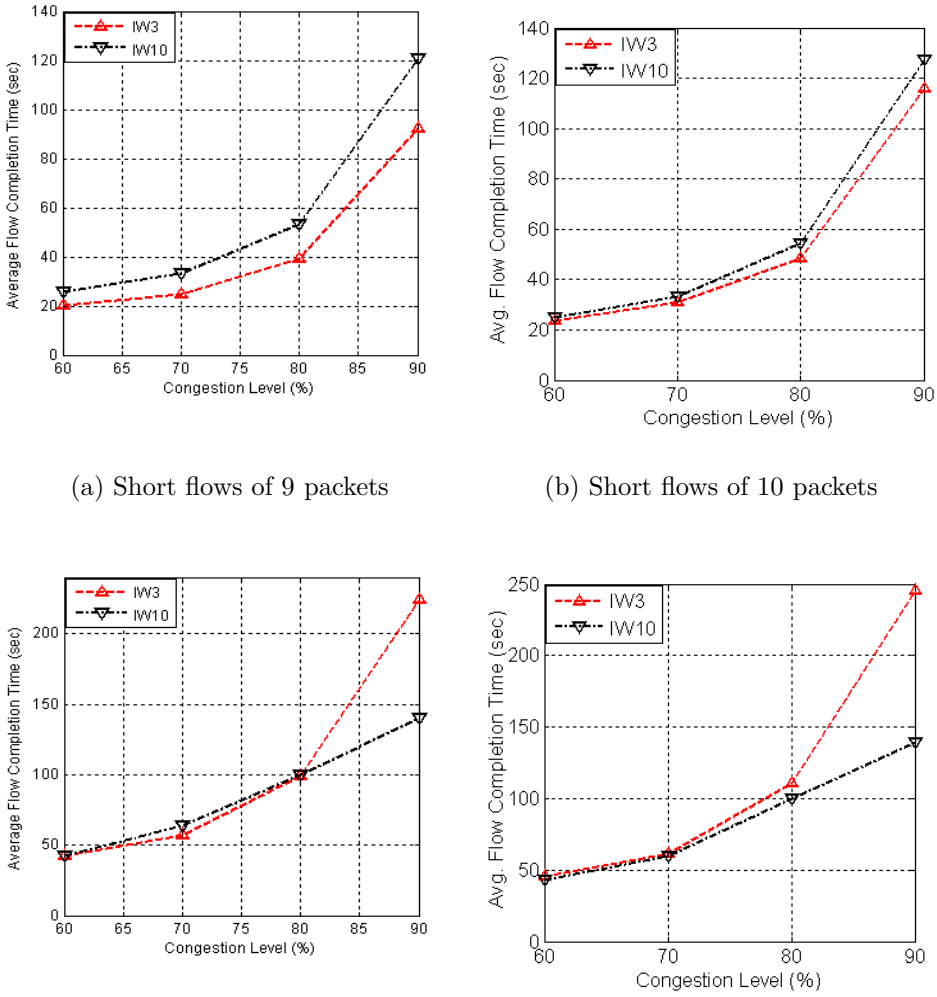
(d) Short flows of 22 packets

Figure 4.23: Hybrid IW: Timeouts and fast retransmits under varying congestion

timeouts under TCP with IW10 are more than twice timeouts by TCP with IW3. Finally, we observe that as the number of flows increases, the percentage of timeouts thus increases irrespective of the window size. We also note that TCP with IW10 has fast retransmits percentages as much more significant unlike for flows less than 10 packets. Nevertheless, TCP with IW3 in bursty traffic indicate a higher percentage of fast retransmissions as opposed to TCP with IW10 with smaller percentages as in Figures 4.23(b) and 4.23(d) respectively.

4.3.3 Hybrid(IW3+IW10)Under Mixed Congestion

We evaluate the performance of hybrid initial windows under mixed congestion. We also compare the individual IW3 and IW10 results of completion time, average throughput and packet losses as shown in this Section.



(a) Short flows of 9 packets

(b) Short flows of 10 packets

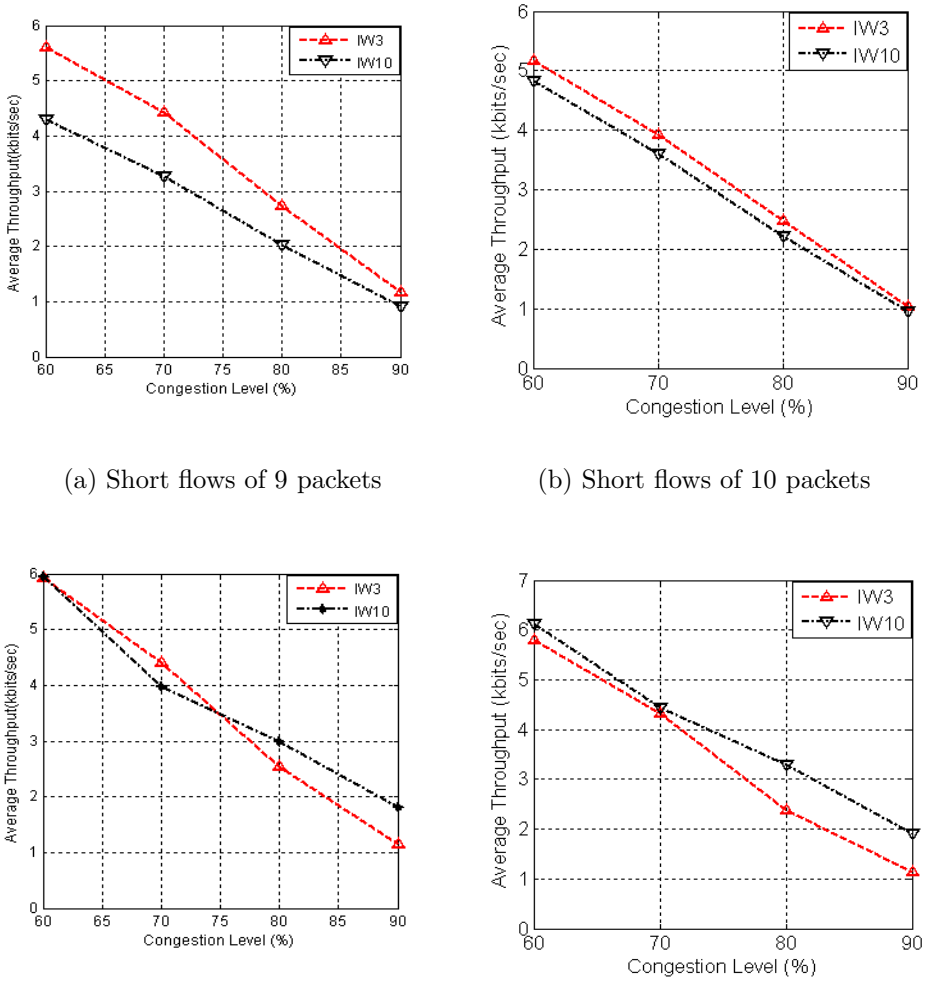
(c) Short flows of 21 packets

(d) Short flows of 22 packets

Figure 4.24: Hybrid IW: Average completion time under mixed congestion

In Figure 4.24 we observe that IW10 flow completion time at all congestion points are much more than IW3 completion time. We also note that the performance difference between IW3 and IW10 deteriorates as the flow size increases from 9 to 10 packets. For example at highest congestion point, TCP with IW10 shows flow completion time for short flows with 9 packets is 120 seconds as opposed to 130 seconds for short flows with 10 packets. Where

as TCP with IW3 indicate average flow completion time for short flows with 9 packets just about 90 seconds as opposed almost 120 seconds for flows with 10 packets. We then observe a greater deterioration in performance by TCP with IW3 than IW10 in terms of longer flow completion time. However, on the side of 21 and 22 packets TCP with IW3 deteriorates performance much more as opposed to IW10 when congestion increases. We note between 80% and 90% congestion levels, as more congestion continues to pile in the pipe TCP with IW3 throughput grows drastically as opposed to TCP with IW10 throughput which increases steadily as congestion grows further. Lastly, this implies average flow completion is much more affected by TCP with IW3 than IW10 flow completion time under bursty traffic.



(a) Short flows of 9 packets

(b) Short flows of 10 packets

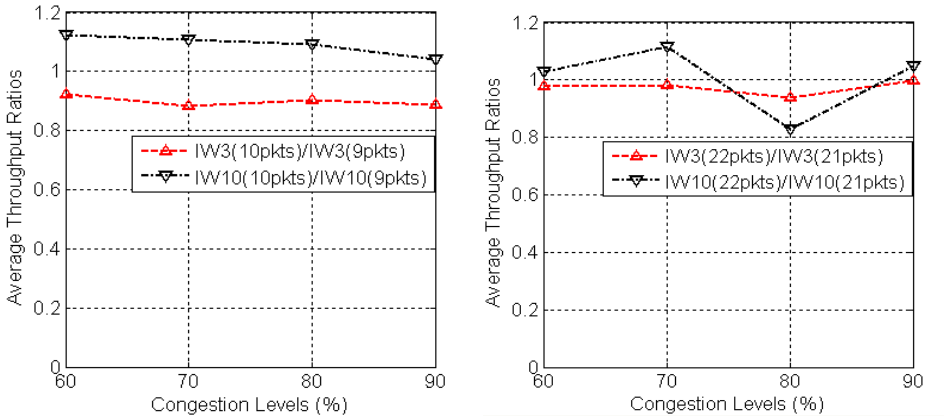
(c) Short flows of 21 packets

(d) Short flows of 22 packets

Figure 4.25: Hybrid IW: Average throughput under mixed congestion

Figure 4.25 illustrate average throughput of short flows under mixed congestion. We observe

that short flows less than 10 packets indicate IW10 with much lower average throughput compared to IW3 as congestion increases on the pipe. In fact at very high congestion points such as 90% level, we note the average throughput for IW10 is equally the same but as congestion levels drop, IW10 throughput increases even much more for flows with 10 packets than 9 packets. While on the side of bursty short flows with 21 and 22 packets. In particular under flows with 21 packets, we observe that TCP with IW3 at congestion point less or equal than 75% indicate average throughput more than IW10 while beyond this point the patterns alternates in that TCP with IW10 has higher throughput than IW3 as congestion grows towards 100% on the link. While under 22 packets, it is clearly shown TCP with IW10 is much better than IW3 in terms of average throughput as congestion grows. Finally, we note that as a flow size is increased from 21 to 22 packets, we observe that TCP with IW3 average throughput reduces at all congestion points while for TCP with IW10 increases average throughput.



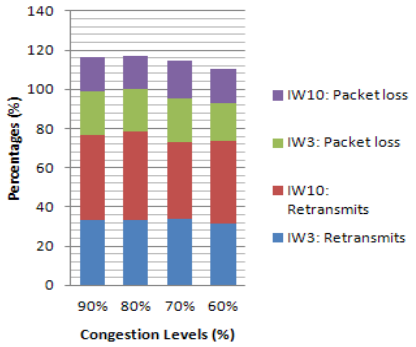
(a) Throughput ratio of 10 to 9 packets

(b) Throughput ratio of 22 to 21 packets

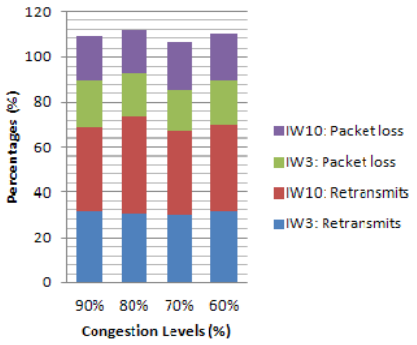
Figure 4.26: Hybrid IW: Throughput ratios under mixed congestion

In Figure 4.26, we observe that throughput ratios of short flows with 10 packets to 9 packets under IW3 and IW10 in mixed congestion. As a packet is added, we observe that IW10 throughput ratios are greater and above 1 as opposed to IW3 ratios which are less than 1. This implies that TCP with IW10 favors flows with 10 packets as opposed to 9 packets while with TCP with IW3 indicate flows of 9 packets being favored as opposed to 10 packets at all congestion points. As an extra packet is increased from 21 to 22 packets, the throughput

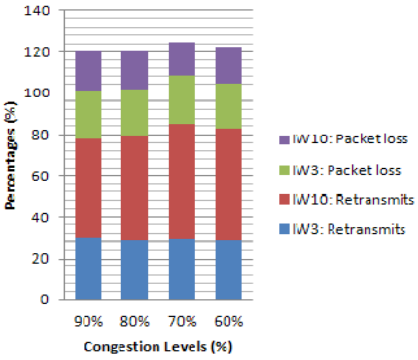
ratios of these flows under TCP with IW3 are slightly flat or constant and less than 1, implying flows with 21 packets are favoured in terms of greater throughput as opposed to 22 packets. While under TCP with IW10, the throughput ratios tend to varies between 0.8 and 1.2 as congestion grows more on the link.



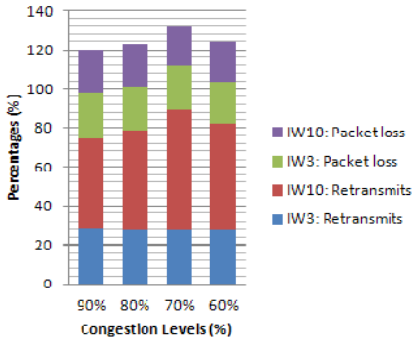
(a) Short flows of 9 packets



(b) Short flows of 10 packets



(c) Short flows of 21 packets

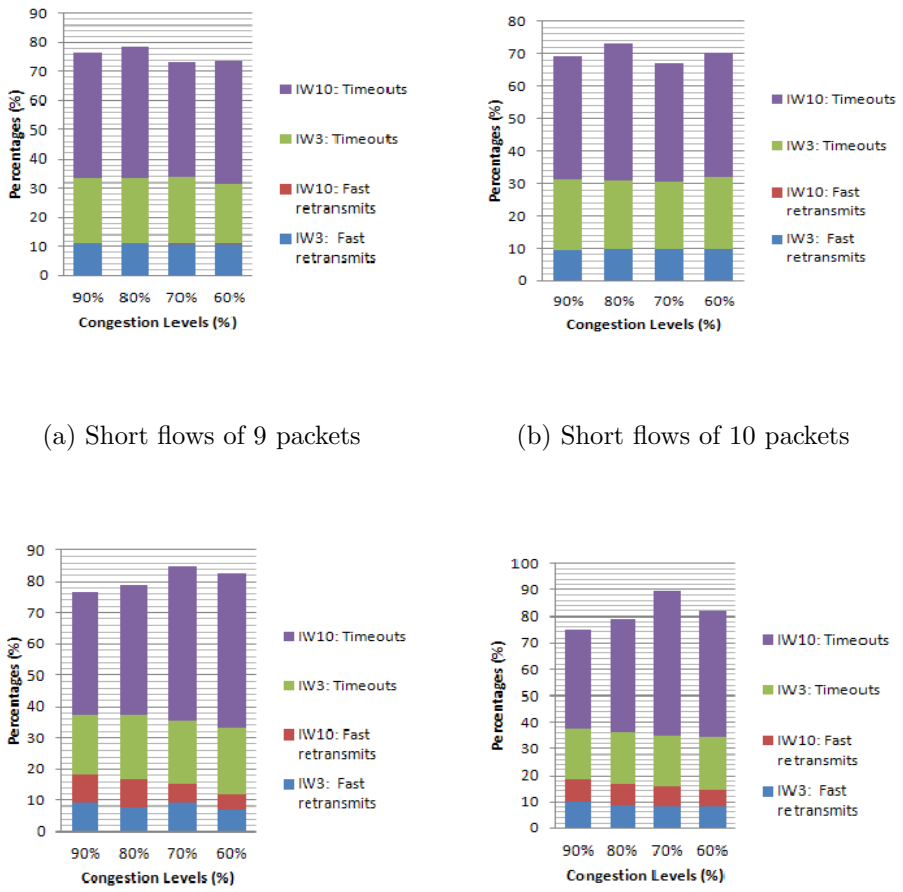


(d) Short flows of 22 packets

Figure 4.27: Hybrid IW: Packet loss and retransmits under mixed congestion

Figures 4.27(a) and 4.27(b) are quite similar. However, we also note that packet losses and retransmissions are much more under short flows of 10 packets as opposed to flows with 9 packets regardless the initial window size. For flows with 9 packets, we observe TCP with IW3 has more packet losses than IW10. But for flows with 10 packets IW10 packet losses are higher than IW3 losses where as TCP with IW10 retransmissions irrespective of the flow sizes are always higher than IW3 retransmissions. While under 21 and 22 packets, IW3

packet losses are still more than IW10. However, we must note the TCP with IW10 is almost twice the number of retransmission in IW3 as shown in Figure 4.27(d).



(a) Short flows of 9 packets

(b) Short flows of 10 packets

(c) Short flows of 21 packets

(d) Short flows of 22 packets

Figure 4.28: Hybrid IW: Timeouts and fast retransmits under mixed congestion

Figure 4.28 illustrates timeouts and fast retransmits effect of a hybrid window under mixed congestion on a limited link. When the window side is combined, we notice that TCP with IW10 records a higher amount of timeouts, almost twice the number of timeouts in IW3. Where as IW10 fast retransmits are negligible but not for TCP with IW3 which has a reasonably number of fast retransmits (approx. at 10%) under both short flows of 9 and 10 packets. However, short flows of 21 and 22 packets indicate that IW10 timeouts have higher percentages i.e twice as much as timeouts by TCP with IW3. Fast retransmits under bursty flows show that they are more in IW3 than IW10. Finally, we observe that under larger flow

size such as 21 and 22 packets, fast retransmits are very significant as opposed to short flows less than 10 packets which show fast retransmits as very minimal.

4.4 Conclusion

In this Chapter we have presented our findings of using TCP configured under IW3, IW10 and combined or hybrid initial windows in highly congested environments. We also based our results on the performance of TCP short flow sizes of 9, 10, 21 and 22 packets only. We mainly focused on three metrics; flow completion time, throughput and packet losses of each flow.

Chapter 5

Conclusions and Future Outlook

This Chapter concludes the experience in conducting TCP research, we identified three congestion patterns induced by TCP and UDP flow of traffic through experimentation. We then measured the performance of TCP with initial window size of a single value(i.e IW3 or IW10) and also hybrid or combined window size i.e (IW3+IW10) under different flow sizes. Finally we present a brief summary of our results and future work in Section 5.1 and 5.2 respectively.

5.1 Summary of the Results

In this Section, we summarize the findings of the research. Firstly, our results clearly show that TCP short flows with less than 10 packets configured under IW3 provides much shorter flow completion time which results into higher flow throughput as opposed to TCP with IW10 which is observed as aggressive thus adding much more congestion than IW3 as far TCP is concerned. However, larger or bursty flows of TCP with IW10 significantly outperforms bursty flows configured with IW3 especially under constant and mixed congestion but not under varying congestion whereby the performance of shorter flows with less than 10 packets configured with IW10 is enhanced as expected much more than IW3.

It is also important to note that as a flow size increases by a single packet, the performance of IW3 deteriorates much more than IW10 in terms of severe packet drops under bursty flows and very high congestion. However, all short flow sizes are seen to have very high number

of retransmission under TCP with IW10 in contrast to IW3 as observed in each congestion pattern. This is so because IW10 is a more aggressive window as opposed to IW3. Even under hybrid, IW3 out competes IW10 especially under constant and mixed congestion patterns. Although the performance results of the hybrid window are better than single configured initial window under a congested setup.

Lastly, results on packets retransmissions and timeouts under IW10 are shown to be incredibly higher than IW3 all the time regardless the flow size and the congestion levels. This implies that a flow experiences multiple retransmissions to a single packet lost. Where as packet losses are much more for shorter flows of less than 10 packets under IW10 than IW3 but not for bursty flows of less than 22 packets under highly congested link (estimated to 80% congestion percentage on the link). We also noted that fast retransmissions under IW10 are very minimal for only flows less than 10 packets but not 22 packets in all congested setups.

5.1.1 General Conclusion

In this Section, we finally present the general conclusion of the results. Firstly, we conclude by stating that IW10 is a poor performance initial window for flows less than 10 packets on a link that is experiencing congestion of about 80% and beyond. However, under a non-congested or slightly congested to about 60% congestion and below, IW10 initial value is very good as shown by previous work.

We also compared the experimental setting of the previous studies with IW10 and IW3 for short flows of 10 packets against 9 packets. For the former, TCP with IW3 sends an extra window of one additional window of one packet compared to sending only 9 packets. We have shown that the performance of TCP with 10 packets is poorer than 9 packets for IW3 which favours the case of IW10 as shown in previous work. We also extended this for both short flows of 22 against 21 packets whereby short flows of larger packets overwhelm TCP with IW3 much more than IW10, thus resulting to deterioration in the performance. It is at this point that we recommend an automated or designated initial window value as network congestion changes.

5.2 Future Outlook

During the evaluation of IW10 described in Chapter 4, we evaluated the performance of initial values of standard TCP referred as TCP New-Reno (the default in Linux kernels since version 2.6.35). It would be interesting to evaluate the initial windows performance with other slower TCP variants other the New reno.

It may also be interesting to explore other initial window values other than IW3 and IW10 only, or even beyond 10 segments using different approaches.

Bibliography

[1] L. Parziale, D. T. Britt, C. Davis, J. Forrester, W. Liu, C. Mathews, and N. Rosselot, *TCP/IP Tutorial and Technical Overview*. Redbooks, dec 2006.

[2] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Featuring the Internet*. Addison- Wesley, jul 2004.

[3] N. Dukkipati, T. Refice, Y. Cheng, J. C. T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCPs Initial Congestion Window," in *Proc. ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, jul 2010.

[4] V. Jacobson and J. K. Michael, "Congestion Avoidance and Control," in *Proc.SIGCOMM 88, ACM*, Stanford, CA, aug 1988.

[5] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," sep 1998, iETF RFC 2414.

[6] A. T. Inc, "State of the Internet," Cambridge, MA, First Quarter 2010, report.

[7] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's Initial Window," oct 2002, iETF RFC 3390.

[8] H. K. J. Chu and D. Nandita, "Increasing TCP's Initial Window," mar 2010, iETF Internet Draft: draft-hkchu-tcpm-initcwnd-00.txt.

[9] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," apr 1999, iETF RFC 2581.

[10] H. K. J. Chu, D. Nandita, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," apr 2011, iETF Internet Draft: draft-ietf-tcpm-initcwnd-00.txt.

[11] K. Poduri and K. Nichols, "Simulation Studies of Increased Initial TCP Window Size," sep 1998, iETF RFC 2415.

[12] M. Allman, C. Hayes, and S. Ostermann, "An Evaluation of TCP with Larger Initial Windows," in *Proc. ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, jul 1998.

[13] D. Nandita, Y. Cheng, H. K. J. Chu, and M.Mathis, "Increasing TCP's Initial Window," jul 2010, iETF Internet Draft: draft-hkchu-tcpm-initcwnd-01.txt.

[14] A. T. Inc, "State of the Internet," Cambridge, MA, Third Quarter 2009, report.

[15] M. Veeraraghavan, "TCP Enhancements," apr 2009, writeup.

[16] J. Padhye, V. Firoiu, and D. Towsley, "Modelling TCP Reno performance: A simple Model and its Empirical Validation," in *Proc. IEEE transactions on Networking*, vol. 8, no. 2, apr 2000.

[17] A. Kuzmanovic and E.W.Knightly, "Low-Rate TCP- Targeted Denial of Service Attacks: (The Shrew vs. the Mice and Elephants)," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, jul 2003.

[18] G. Inc, "SPDY: An Experimental Protocol for A Faster Web," 2009, http://dev.chromium.org/spdy.

[19] L. Peterson and B. S. David, *Computer Networks: A system Approach.* Morgan. Kaufmann, may 2003.

[20] A.I.Rai, E.W.Biersack, and G.Urvoy-Keller, "Analyzing the performance of TCP flows in Packet Networks with LAS schedulers," Ph.D. dissertation, Institut Eurecom, Sophia-Antipolis,06904 FRANCE, sep 2003.

[21] V.Paxson and S.Floyd, "Wide-Area Traffic: The Failure of Poison Modeling," in *Proc. IEEE/ACM Transactions on Networking*, vol. 3, no. 3, jun 1995.

[22] C. Williamson, "Internet Traffic Measurement," university of Calgary.

[23] M. Scharf, "Performance Evaluation of Fast Startup Congestion Control Schemes," 2010, work still in Progress.

[24] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," in *Proc. IEEE InfoComm*, mar 2000.

[25] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," nov 2010, iETF RFC 2988.

[26] P. Ioannis and T. Vassilis, "On the Properties of an adaptive TCP minimum RTO," *in Computer Network Journal (COMNET)*, sep 2007.

[27] H. K. J. Chu, V. Paxson, and M. Allman, "Reducing initial RTO," mar 2010, internet Draft: draft-paxson-tcpm-rfc2988bis-00.txt.

[28] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," jan 2001, iETF RFC 2001.

[29] V. Jacobson, "Modified TCP Congestion Control and Avoidance Algorithms," apr 1990.

[30] W. Kimaili, "Implementing Bandwidth Management in a low-bandwidth Environment," Master's thesis, Makerere University, Kampala, Ug, jul 2007.

[31] D. Ivancic, N. Hadjina, and D. Basch, "Analysis of precision of the htb packet scheduler," in *Proc. ICECom 2005, Applied Electromagnetics and Communications conference*, oct 2005, pp. 1–4.

[32] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks," in *Proc. IEEE/ACM Transactions on Networking*, vol. 3, no. 4, aug 1995.

[33] I. Wakeman, A. Ghosh, J. Crowcroft, V. Jacobson, and S. Floyd, "Implementing Real Time Packet Forwarding Policies using Streams," in *Proc. Usenix 1995 Technical Conference*, jan 1995, pp. 71–82.

[34] J. Valenzuela, A. Monleon, I. S. Esteban, M. Portoles, and O. Sallent, "A hierarchical token bucket algorithm to enhance qos in ieee 802.11: proposal, implementation and evaluation," in *Proc. IEEE 60th, Vehicular Technology Conference*, sep 2004, p. 26592662.

[35] S. Jong, P. Danzig, L. Zhen, and Y.Limin, "Evaluation of TCP Vegas: Emulation and Experiment," in *Proc. ACM SIGCOMM*, aug 1995.

[36] T. Henderson and R. Katz, "Transport Protocols for Internet-Compatible Satellite Networks," *IEEE Journal on Selected Areas of Communications*, sep 1999.

[37] B. Hubert, T. Graf, G. Maxwell, R. V. Mook, M. V. Oosterhout, P. B. Schroeder, J. Spaans, and P. Larroy, "Linux Advanced Routing & Traffic Control HOWTO," oct 2005, http://lartc.org.

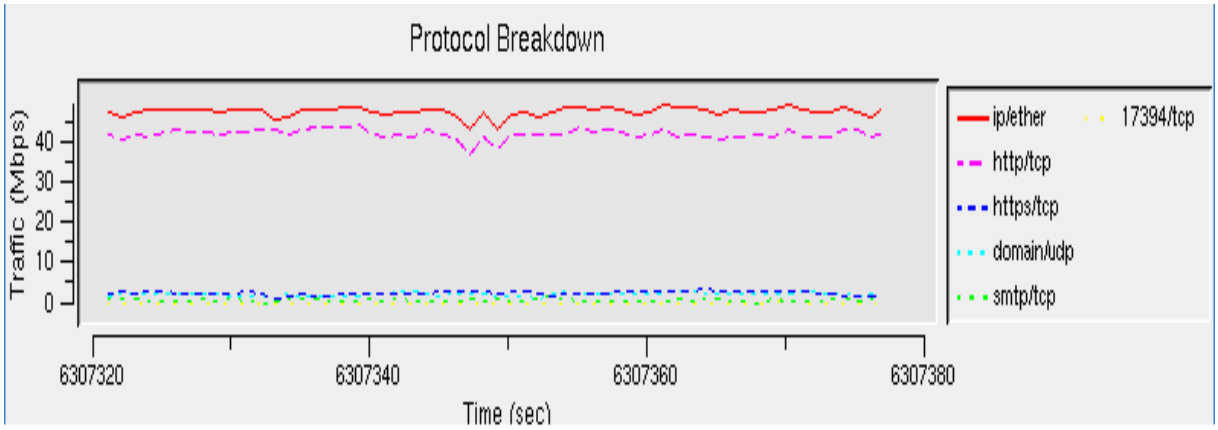
[38] S. Hemminger, "Network Emulation with NetEm," apr 2005, open Source Development Lab.

[39] S. Parker and C. Schmechel, "Some Testing Tools for TCP Implementers," aug 1998, iETF RFC 2398.

[40] R. Sharpe, U. Lamping, and E. Warnicke, "Wireshark Users Guide 35146 for Wireshark 1.4," apr 2010, wireshark User Manaul.

Appendices

Appendix A: Protocol breakdown on a congested link



Appendix B: Bandwidth and Delay Configurations

Algorithm 1 Bottleneck Bandwidth-Delay and Buffer Configurations

$\#tc\ qdisc\ add\ dev\ eth1\ root\ handle\ 1:0\ htb$
$\#tc\ qdisc\ add\ dev\ eth2\ root\ handle\ 1:0\ htb$**{adds queuing discipline (FIFO)}**
$\#tc\ class\ add\ dev\ eth1\ parent\ 1:0\ classid\ 1:1\ htb\ rate\ 64kbit\ burst\ 1632b$
$cburst\ 1632b$
$\#tc\ class\ add\ dev\ eth2\ parent\ 1:0\ classid\ 1:1\ htb\ rate\ 64kbit\ burst\ 1632b$
$cburst\ 1632b$ **{classify the traffic type}**
$\#tc\ filter\ add\ dev\ eth1\ parent\ 1:0\ protocol\ ip\ prio\ 1\ u32\ match\ ip\ src$
$192.168.1.0/24\ flowid\ 1:1$
$\#tc\ filter\ add\ dev\ eth2\ parent\ 1:0\ protocol\ ip\ prio\ 1\ u32\ match\ ip\ dst$
$192.168.2.0/24\ flowid\ 1:1$ **{adds a filter on incoming and outgoing traffic}**
{Delay Configurations using Netem}
$\#tc\ qdisc\ add\ dev\ eth1\ parent\ 1:1\ handle\ 10\ netem\ delay\ 100ms$
$\#tc\ qdisc\ add\ dev\ eth2\ parent\ 1:1\ handle\ 10\ netem\ delay\ 100ms$
$\#ifconfigeth1txqueuelen40$**{Buffer configurations}**

Appendix C: Short flows traffic generation script

Algorithm 2 **Short flow generation based on Poisson distribution**

$from\ \ time\ \ import\ \ time$
$from\ \ time\ \ import\ \ sleep$
$import\ \ random$
$import\ \ os$
$import\ \ commands$
$f = 0$
$s = time()$
$t = time() - s$
$k = int(raw_input\ \ ("Enter\ \ flow\ \ test\ \ duration:"))$
while $(time() - s) <= k :$ **do**
  $x = random.expovariate(50)$
  $f = f + 1$
  $print\ \ ("-----------------------------------")$
  $print\ \ 'Generate\ \ short - lived\ \ flow\ \ no:', f, x, time() - s$
  $print\ \ ("-----------------------------------")$
  $cmd\ \ =\ \ "netperf\ \ -\ l\ \ -\ 1\ \ -\ H\ \ 192.168.1.2, 4\ -\ t\ TCP_RR\ \ -\ -\ -$
  $r\ \ 200, 31500"\ \ status,\ \ output = commands.getstatusoutput(cmd)$
  $print\ \ "Output:",\ \ output$
  $sleep(x)$
end while
if t == k **then**
  $print\ \ ("End\ \ of\ \ test\ \ time")$
end if

Appendix D: TCP Initial Window Configurations

Algorithm 3 Initial Window Configuration on Linux (kernel version ¿= 2.6.35)

$\#ip\ route\ show$ {**on the server side**}
$\#ip\ route\ change\ default\ via$ 192.168.1.1 $dev\ eth1\ initcwnd\ < iw = 3\ or\ 10 >$
{on the sender side}
$\#ip\ route\ change\ default\ via$ 192.168.2.1 $dev\ eth2\ initrwnd\ < initrwnd = 16,$
$when\ iw = 10\&default\ wheniw = 3 >$ **{on the receiver side}**
$\#SNDBUF\ value >= IW * MSS${**If the server process explicitly set SNDBUF**}
$\#IW * MSS > /proc/sys/net/ipv4/tcp_wmem$ **{while increasing the initial socket buffer}**
$\#cat/proc/sys/net/ipv4/tcp_wmem$ **{Restart server process on editing}**