TCP Fairness

By

Mohammed Yahya

A project report submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Internetworking
(Faculty of Science)

University of Alberta
February 2011

Abstract

In this project, we focus on the fairness performance of TCP within a simulated WAN environment. Our goal is to understand whether different TCP variants (SACK, CUBIC, HighSpeed, and Westwood) can share the bottleneck link fairly. Specifically, we would like to study the behavior of TCP when two different TCP variants compete with each other over the same bottleneck link. If each of them uses half of the bandwidth of the bottleneck link, then these TCP variants are fair to each other. If one of them consumes more than half of the bandwidth, then the TCP variant is not fair to the other one. Our experiment results show that the TCP variants under investigation are not fair to each other. When two of them compete for bandwidth, one of them always leads to a higher throughput.

Acknowledgements

I owe special thanks to Dr. Mike H.MacGregor who helped make this project possible and gave me useful comments and suggestions. I would also like to give special thanks to Dr. Qiang Ye who guided and lent his support at every step along the way. I am greatly indebted to both of their vision and persistence in achieving this goal.

Thanks also to my wife and a son who have also been very cooperative and patient throughout, especially during the long hours of LAB works.

Table of Contents

Chapter 1 Introduction

This chapter is about the nitty-gritty of Transmission Control Protocol and some of its variants. TCP is the most often used protocol, and resides at layer four in the protocol stack. TCP provides congestion control, reliability, and a stream on which to send data. To be competent, TCP tries to send as much data as possible before getting an ACK back.

1.1 Motivation

Theoretically, if X TCP sessions compete or share the same bottleneck link, each should get 1/X of link capacity. However, in real world scenarios, TCP flows with different RTTs that shared the same bottleneck link do not attain the equal amount of free bandwidth. Our objective is to study fairness and end-to-end performance in simulated WAN network in a LAB environment via the following methodology. First, we develop a formal reference topology that characterizes objectives such as initiating multiple concurrent tcp sessions which share the same bottle link of 100Mbps. Second, we perform an extensive set of experiments to quantify the impact of the key performance factors towards achieving the goal. For example, we study the variations of various parameters different TCP flavors, RTT, bandwidth, elapsed time as well as congestion window. Finally, we study the critical relationship between fairness and aggregate throughput in a WAN environment and determine whether the performance fairness exists, if it does then which TCP variant take the lead in aggressiveness.

1.2 TCP Variants

[7]TCP (Transmission Control Protocol) is the major transport protocol utilized in IP networks. The TCP protocol exists on the Transport Layer of the OSI Model. The TCP protocol is a connection-oriented protocol which provides end-to-end reliability. Connection-oriented means, that before two network nodes can communicate using TCP, they must first complete a handshaking protocol to create a connection. End-to-end reliability means, that TCP includes mechanisms for error detection and error correction between the source and the destination. The following figure 1-1 depicts a reliable TCP connection between the two nodes [7].

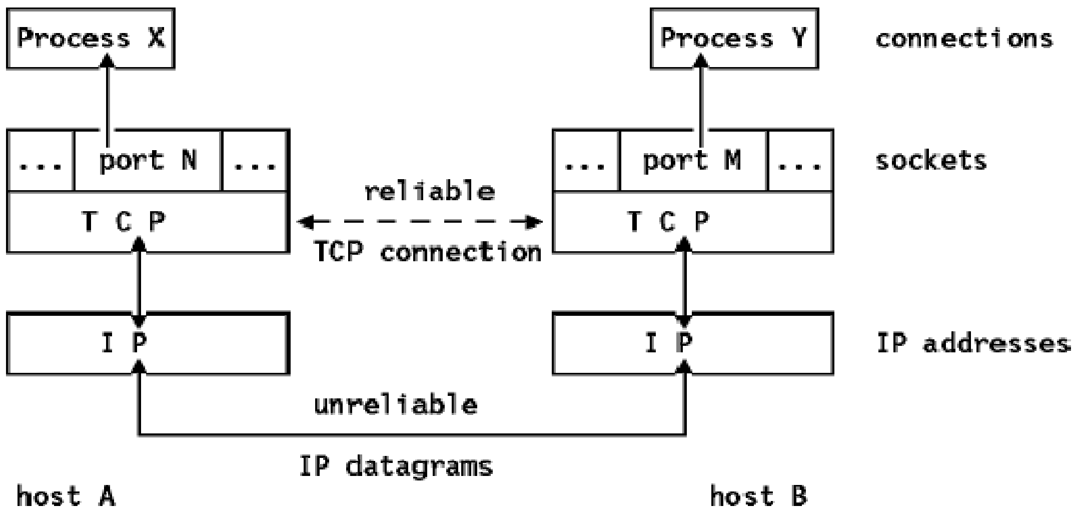

Figure 1-1: Reliable TCP Connection between two hosts

TCP data is encapsulated in an IP datagram. The below figure 1-2[11] depicts the format of the TCP header. Its normal size is 20 bytes unless options are present.

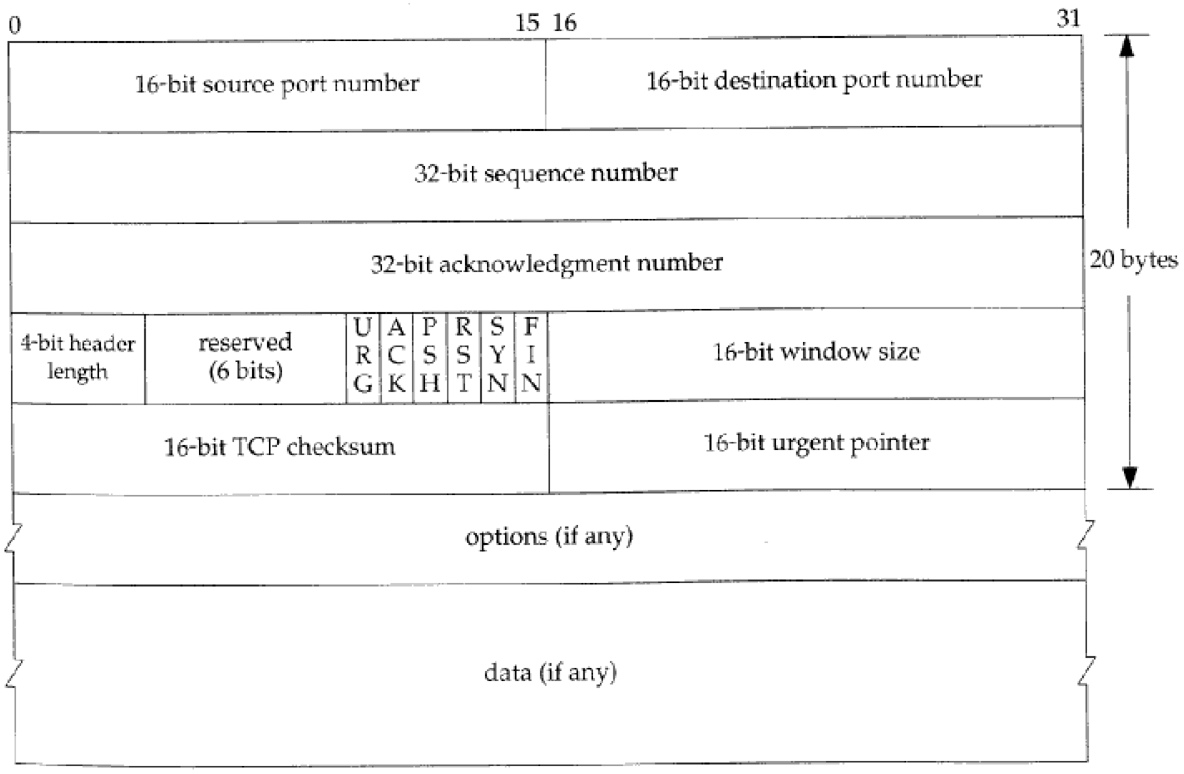

Figure 1-2: TCP Header

[4]The protocol keeps track of sent data, detects lost packets, and retransmits them if necessary. This is done by acknowledging every packet to the sender. Basically, this is how TCP works. There are some parameters of TCP connections that make things very

interesting and complicated. The first perturbation stems from the network itself, and materializes in the form of three parameters that pertain to every TCP connection.

- **Bandwidth**
  The bandwidth indicates how many bits per time frame the link can transport. It is usually denoted by Mbit/S or kbit/S and limited by the hardware.
- **Round-Trip Time (RTT)**
  Consider a network link between points X and Y. The time a packet needs to travel from X to Y and then back to X is called the Round-Trip Time. The RTT is highly variable, especially when a node on the way experiences congestion. Typically, we have an RTT from milliseconds to seconds (in the worst case).
- **Packet loss**
  Packets of a network transmission can get dropped. The ratio of lost packets to transported packets is called packet loss. There are many reasons for packet loss. A router might be under heavy load, a frame might get corrupted by interference (wireless networks like to drop packets this way), or an Ethernet switch may detect a wrong checksum.

TCP RENO
This is the classical model used for congestion control. It exhibits the typical slow start of transmissions. The throughput increases gradually until it stays stable. It is decreased as soon as the transfer encounters congestion, then the rate rises again slowly. The window is increased by adding fixed values. TCP Reno uses a multiplicative decrease algorithm for the reduction of window size. TCP Reno is the most widely deployed algorithm.

TCP SACK
TCP with SACK is an extension of TCP Reno and it works around the problems faced by TCP RENO, namely detection of multiple lost packets per RTT. SACK requires that segments not be acknowledged cumulatively but should be acknowledged selectively. With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost.

TCP CUBIC
CUBIC is a less aggressive variant of BIC (meaning, it doesn't steal as much throughput from competing TCP flows as does BIC). It does not rely on the receipt of ACKs to increase the window size. CUBIC's window size is dependent only on the last congestion event. With standard TCP, flows with very short RTTs will receive ACKs faster and therefore have their congestion windows grow faster than other flows with longer RTTs. CUBIC allows for more fairness between flows since the window growth is independent of RTT.

TCP HIGH SPEED
The main use is for connections with large bandwidth and large RTT (such as Gbit/s and 100 ms RTT). HSTCP's window grows faster than standard TCP and also recovers from losses more quickly. This behavior allows HSTCP to be friendly to standard TCP flows

in normal networks and also to quickly utilize available bandwidth in networks with large bandwidth delay products. HSTCP has the same slow start/timeout behavior as standard TCP.

TCP WESTWOOD [8]
Westwood addresses both large bandwidth/RTT values and random packet loss together with dynamically changing network loads. It analyses the state of the transfer by looking at the acknowledgement packets. Westwood is a modification of the TCP Reno algorithm.
TCPW relies on mining the ACK stream for information to help it better set the congestion control parameters: Slow Start Threshold (ssthresh), and Congestion Window (cwin). In TCPW, an "Eligible Rate" is estimated and used by the sender to update ssthresh and cwin upon loss indication, or during its "Agile Probing" phase.

1.3 TCP Window Size

[9]The TCP window size is by far the most important parameter to adjust for achieving maximum bandwidth across high-performance networks. Properly setting the TCP window size can often more than double the achieved bandwidth.
Technically, the TCP window size is the maximum amount of data that can be in the network at any time for a single connection. (It is the upper limit of the TCP congestion window.)
Think of a water hose. To achieve maximum water flow, the hose should be full. As the hose increases in diameter and length, the volume of water to keep it full increases. In networks, diameter equates to bandwidth, length is measured as round-trip time, and the TCP window size is analogous to the volume of water necessary to keep the hose full. On fast networks with large round-trip times, the TCP window size must be increased to achieve maximum TCP bandwidth.

2 Experimental Design

The focus in this chapter is on experiments design, topology and the test groups with possible scenarios.

2.1 Experimental Parameters

In this project, we used file transfer (FTP) as the application to study TCP fairness.
For the experiments, we considered different TCP variants, fixed bandwidths, different round-trip time, and a fixed file size of 100MB. Here are the details of these parameters:

1) TCP Variants: SACK enabled Reno, CUBIC, WESTWOOD, and HIGHSPEED are the initial variant candidates.

2) Bandwidths: We maintained 100Mbps of bandwidth on each link including the bottleneck. Thus, the bottleneck bandwidth was lower than the sum of Sender1 and Sender2's bandwidth.

3) Different Round-Trip Time: We introduced different delays by using TC to the experiments explicitly: 100ms fixed delay was introduced at Sender2 interface and variable delays of 20ms, 100ms, 200ms and 400ms were introduced at Sender1 interface.

4) File Size: 100MB file was transferred using FTP from both Senders to their respective receivers in Combo Scenarios.

2.2 TCP Window Adjustment

The Transmission Control Protocol (TCP) receive window size is the maximum amount of received data, in bytes, that can be buffered at one time on the receiving side of a connection. The sender can send only that amount of data before waiting for an acknowledgment and window update from the receiver.

2.2.1 Computing TCP Window Size

Theoretically the TCP window size should be set to the bandwidth delay product, which computes the volume of data that can be in the network between two machines. The bandwidth delay product (BDP) is:
Bottleneck Bandwidth * Round-Trip Time

To compute the bandwidth delay product for a pair of hosts, first estimate what the slowest link between them is. Often this is the 100 Mbit/sec ethernet the machine is connected to, or the 45 Mbit/sec DS3 link from the campus to the wide-area. Then use ping to find the round-trip time. For example, if the slowest link is a 45 Mbit/sec DS3 link, and the round-trip time is 30 milliseconds:

45 Mbit/sec * 30 ms
= 45e6 * 30e-3
= 1,350,000 bits / 8 / 1024
= 165 KBytes

2.2.2 Setting TCP Window Size

The TCP window size can be set on a per connection basis, as detailed below. Most OS and hosts have upper limits on the TCP window size. These may be as low as 64 KB, or as high as several MB. To enable TCP window sizes larger than 64 KB, TCP large window extensions (RFC 1323) must be enabled. Since TCP is a reliable transport, if any data is lost in transmission, TCP must be able to retransmit it. Thus TCP remembers all the sent data in a buffer until the other side acknowledges receiving it. The size of this buffer is the TCP window size.
The TCP window sizes are implemented by send and receive buffers on each end of the connection. To set these buffers, use the SO_SNDBUF and SO_RCVBUF socket options. Both ends of the connection must set these options.

2.2.3 Adjusting TCP window size

While the bandwidth delay product gives the theoretical value for the TCP window size that is not always the best value. Problems come because the OS's TCP implementation has bugs and/or the network has deficiencies. Usually we try values 10% above and below the calculated TCP window size. If one of those is better, we try values above and below that, repeating until the maximum bandwidth is reached. Remember there will be some variability in bandwidth due to other competing network traffic. Following are the calculated TCP window sizes and settings used in the experiments:

```
net.ipv4.tcp_moderate_rcvbuf = 0
net.ipv4.tcp_no_metrics_save=1

net.ipv4.tcp_rmem = 4096 87380 2076672
net.core.rmem_default = 110592
net.core.rmem_max = 191052
net.ipv4.tcp_wmem = 4096 16384 2076672
net.core.wmem_default = 110592
net.core.wmem_max = 110592
net.ipv4.tcp_mem = 48672 64896 97344
net.ipv4.tcp_rfc1337 = 0
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.tcp_sack = 1
net.ipv4.tcp_fack = 1
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_ecn = 0
net.ipv4.route.flush = 1
```

GROUP-1 Tests:

- **0.325ms and BW 100Mbps**

(Ping yields average RTT 0.350ms when no fixed delay is introduced)

RWIN/BDP = 4096*2=8192 (used default)

USED LINUX DEFAULT TCP WINDOW SETTINGS

- Baseline1-100MB-R1-SACK-BW100Mbps-NoDrop-NoFixedDelay-S1.log
- Baseline2-100MB-R1-CUBIC-BW100Mbps-NoDrop-NoFixedDelay-S1.log
- Baseline3-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-NoFixedDelay-S1.log
- Baseline4-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-NoFixedDelay-S1.log

- **100ms and BW 100Mbps**

RWIN/BDP = 1249280*2 = 2498560

```
net.ipv4.tcp_rmem=4096 2498560 2498560
net.core.rmem_default=2498560
net.core.rmem_max=2498560

net.ipv4.tcp_wmem=4096 16384 2498560
net.core.wmem_default=16384
net.core.wmem_max=2498560

net.ipv4.tcp_mem=2498560 2498560 2498560
```

- Baseline5-100MB-R1-SACK-BW100Mbps-NoDrop-FixedDelay100ms-S1.log
- Baseline6-100MB-R1-CUBIC-BW100Mbps-NoDrop-FixedDelay100ms-S1.log
- Baseline7-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-FixedDelay100ms-S1.log
- Baseline8-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-FixedDelay100ms-S1.log

GROUP-2 Tests:

1. **<u>20ms and BW 50Mbps</u>**

RWIN/BDP = 124928 * 2 = 249856

```
net.ipv4.tcp_rmem = 4096 249856 249856
net.core.rmem_default = 249856
net.core.rmem_max = 249856

net.ipv4.tcp_wmem = 4096 16384 249856
net.core.wmem_default = 16384
net.core.wmem_max = 249856

net.ipv4.tcp_mem = 249856 249856 249856
```

- Combo1-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay20ms-NoDrop-S1.log

Max Outstanding Data after laps of 5Sec to 12sec

- Combo2-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay20ms-NoDrop-S1.log

Max Outstanding Data after laps of 5Sec to 10sec

- Combo3-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay20ms-NoDrop-S1.log

Max Outstanding Data after laps of 4Sec to 20sec

2. **<u>100ms and BW 50Mbps</u>**

RWIN/BDP = 624640 * 2 = 1249280

```
net.ipv4.tcp_rmem = 4096 1249280 1249280
net.core.rmem_default = 1249280
net.core.rmem_max = 1249280

net.ipv4.tcp_wmem = 4096 16384 1249280
net.core.wmem_default = 16384
net.core.wmem_max = 1249280

net.ipv4.tcp_mem = 1249280 2498560 2498560
```

- Combo4-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay100ms-NoDrop-S1.log

Max Outstanding Data after laps of 4Sec to 20sec

- Combo5-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay100ms-NoDrop-S1.log
- Combo6-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay100ms-NoDrop-S1.log

3. <u>200ms and BW 50Mbps</u>

Max Outstanding Data after laps of 4Sec to 20sec

RWIN/BDP = 1249280 * 2 = 2498560

```
net.ipv4.tcp_rmem = 4096 2498560 2498560
net.core.rmem_default = 2498560
net.core.rmem_max = 2498560

net.ipv4.tcp_wmem = 4096 16384 2498560
net.core.wmem_default = 16384
net.core.wmem_max = 2498560
net.ipv4.tcp_mem = 2498560 2498560 2498560
```

- Combo7-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay200ms-NoDrop-S1.log
- Combo8-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay200ms-NoDrop-S1.log
- Combo9-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay200ms-NoDrop-S1.log

4. <u>400ms and BW 50Mbps</u>

Max Outstanding Data after laps of 4Sec to 20sec

RWIN/BDP = 2499584 * 2 = 4999168

```
net.ipv4.tcp_rmem = 4096 4999168  4999168
net.core.rmem_default = 4999168
net.core.rmem_max = 4999168
net.ipv4.tcp_wmem = 4096 16384 4999168
net.core.wmem_default = 16384
net.core.wmem_max = 4999168

net.ipv4.tcp_mem =  4999168  4999168  4999168
```

- Combo10-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay400ms-NoDrop-S1.log
- Combo11-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay400ms-NoDrop-S1.log
- Combo12-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay400ms-NoDrop-S1.log

GROUP-3 Tests:

a) <u>20ms and BW 50Mbps</u>

RWIN/BDP = 124928 * 2 = 249856

```
net.ipv4.tcp_rmem = 4096 249856 249856
net.core.rmem_default = 249856
net.core.rmem_max = 249856

net.ipv4.tcp_wmem = 4096 16384 249856
net.core.wmem_default = 16384
net.core.wmem_max = 249856

net.ipv4.tcp_mem = 249856 249856 249856
```

- Combo13-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay20ms-Drop100ms-S1.log

Max Outstanding Data after laps of 5Sec to 12sec

- Combo14-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay20ms-Drop100ms-S1.log

Max Outstanding Data after laps of 5Sec to 10sec

- Combo15-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay20ms-Drop100ms-S1.log

Max Outstanding Data after laps of 4Sec to 20sec

b) <u>100ms and BW 50Mbps</u>

Max Outstanding Data after laps of 4Sec to 20sec

RWIN/BDP = 624640 * 2 = 1249280

```
net.ipv4.tcp_rmem = 4096 1249280 1249280
net.core.rmem_default = 1249280
net.core.rmem_max = 1249280

net.ipv4.tcp_wmem = 4096 16384 1249280
net.core.wmem_default = 16384
net.core.wmem_max = 1249280

net.ipv4.tcp_mem = 1249280 2498560 2498560
```

- Combo16-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay100ms-Drop100ms-S1.log
- Combo17-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay100ms-Drop100ms-S1.log
- Combo18-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay100ms-Drop100ms-S1.log

c) <u>200ms and BW 50Mbps</u>

Max Outstanding Data after laps of 4Sec to 20sec

RWIN/BDP = 1249280 * 2 = 2498560

```
net.ipv4.tcp_rmem = 4096 2498560 2498560
net.core.rmem_default = 2498560
net.core.rmem_max = 2498560

net.ipv4.tcp_wmem = 4096 16384 2498560
net.core.wmem_default = 16384
net.core.wmem_max = 2498560
net.ipv4.tcp_mem = 2498560 2498560 2498560
```

- Combo19-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay200ms-Drop100ms-S1.log

- Combo20-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay200ms-Drop100ms-S1.log
- Combo21-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay200ms-Drop100ms-S1.log

d) <u>400ms and BW 50Mbps</u>

Max Outstanding Data after laps of 4Sec to 20sec

RWIN/BDP = 2499584 * 2 = 4999168

```
net.ipv4.tcp_rmem = 4096 4999168  4999168
net.core.rmem_default = 4999168
net.core.rmem_max = 4999168

net.ipv4.tcp_wmem = 4096 16384 4999168
net.core.wmem_default = 16384
net.core.wmem_max = 4999168

net.ipv4.tcp_mem =  4999168  4999168  4999168
```

- Combo22-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedDelay400ms-Drop100ms-S1.log
- Combo23-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedDelay400ms-Drop100ms-S1.log
- Combo24-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedDelay400ms-Drop100ms-S1.log
-

2.3 Experimental Topology

We used the following topology to study TCP fairness. In this project, we used five Linux machines and two Alcatel routers. A Linux machine was setup between the Alcatel routers that was used as a router and drop packets at its egress interface. The 100Mbps of bandwidth was configured among both the Alcatel and Linux routers.

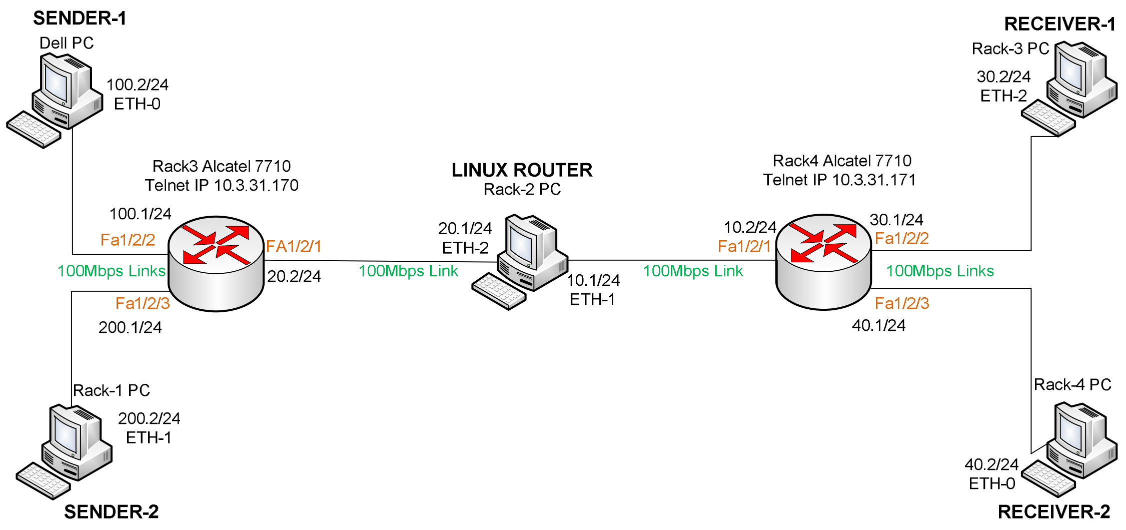

Figure 2-1: Experimental Topology Diagram

2.4 Testing Scenarios

The Linux and Alcatel routers result in the bottleneck, thus the most important principle that we followed was that the bottleneck link was always lower than the sum of the sender’s (Sender1 and Sender2) transmission rate of TCP. Experiments were conducted in the following three different test groups.

2.4.1 Group-1 Baseline Tests

These baseline tests were carried out with an individual TCP connection. A 100MB data file was transferred from Sender1 to Receiver1on 100Mbps bandwidth utilizing FTP by using SACK, CUBIC, HIGHSPEED and WESTWOOD TCP variants respectively with no forced packets drops and no fixed delays, however in Baseline5 through Baseline8 a delay 0f 100ms will be introduced.

1) Baseline1-100MB-R1-SACK-BW100Mbps-NoDrop-NoFixedDelay-S1
2) Baseline2-100MB-R1-CUBIC-BW100Mbps-NoDrop-NoFixedDelay-S1
3) Baseline3-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-NoFixedDelay-S1
4) Baseline4-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-NoFixedDelay-S1
5) Baseline5-100MB-R1-SACK-BW100Mbps-NoDrop-FixedDelay100ms-S1
6) Baseline6-100MB-R1-CUBIC-BW100Mbps-NoDrop-FixedDelay100ms-S1
7) Baseline7-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-FixedDelay100ms-S1
8) Baseline8-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-FixedDelay100ms-S1

2.4.2 Group-2 Combo Tests (No Packets Drops)

In these combo scenarios, couple of competing and concurrent TCP flows was set up. Flow1 was run between Sender1 and Receiver1 and the Flow2 was run between Sender2 and Receiver2 as depicted in the above Topology diagram. Sender2 was setup with constant 100ms delay along with invariable TCP SACK and Receiver2 was also setup with fixed TCP SACK in all the flow2 scenarios. On the other hand, Sender1 and Receiver1 was setup with different TCP variants as well as different Delays in each scenario, thus the RTT varies in Flow1 from 20ms, 100ms, 200ms to 400ms and TCP variants change to CUBIC, HIGHSPEED, and WESTWOOD. No packets drops were enforced in all of these tests.

1) Combo1-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay20ms-NoDrop-S1
2) Combo2-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay20ms-NoDrop-S1
3) Combo3-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixedDelay20ms-NoDrop-S1
4) Combo4-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay100ms-NoDrop-S1
5) Combo5-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay100ms-NoDrop-S1
6) Combo6-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixedDelay100ms-NoDrop-S1
7) Combo7-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay200ms-NoDrop-S1
8) Combo8-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay200ms-NoDrop-S1
9) Combo9-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixedDelay200ms-NoDrop-S1
10) Combo10-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay400ms-NoDrop-S1
11) Combo11-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay400ms-NoDrop-S1
12) Combo12-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixedDelay400ms-NoDrop-S1

2.4.3 Group-3 Combo Tests (Enforced Packets Drops)

The setup of Combo Group3 scenarios was exactly same as in Combo Group-2 except the fact that these tests had been designed to have Packets Drops on purpose for the period of 100ms. The dropping period was determined by the Outstanding (Unacknowledged) Data for each scenario. Outstanding Data Graphs were plotted for that reason using TCPTrace utility.

13) Combo13-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay20ms-Drop100ms-S1

14) Combo14-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay20ms-Drop100ms-S1
15) Combo15-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixedDelay20ms-Drop100ms-S1
16) Combo16-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-S1FixedDelay100ms-Drop100ms-S1
17) Combo17-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-HIGHSPEED-S1FixedDelay100ms-Drop100ms-S1
18) Combo18-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-WESTWOOD-S1FixD100ms-Drop100ms-S2
19) Combo19-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-S1R1CUBIC-S1FixedD200ms-Drop100ms-S1
20) Combo20-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedD200ms-Drop100ms-S1
21) Combo21-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-WESTWOOD-S1FixedD200ms-Drop100ms-S1
22) Combo22-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-S1FixedD400ms-Drop100ms-S1
23) Combo23-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1HIGHSPEED-S1FixedD400ms-Drop100ms-S1
24) Combo24-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1WESTWOOD-S1FixedD400ms-Drop100ms-S1

2.5 Experimental Procedure

2.5.1 Procedure Overview

We followed through the following steps and analyzed the results.

Step-0: Confirmed on all machines there's no TC policy implemented at ingress as well egress before each scenario.
Step-1: Ran NTP server at Linux Router (Also make sure it’s internet Connected)
Step-2: Ran NTP Client as well as NTP Daemon on Sender-1, Receiver-1, Sender-2, and Receiver-2 end and synched them all with the NTP Server
Step-3: Created a 100 MB data file on the dot at Sender-1 and Sender-2 ends
Step-4: Added a fixeddelay100ms at Sender-2 machine
Step-5: Set TCP Variant at Sender-1 and Sender-2 end as per the scenario
Step-6: Ran IPerf to test current bandwidth on the link
Step-7: Ran VSFTPD at Sender-1 as well as at Sender-2 end
Step-8: Created/Edited LFTP scripts at Receiver-1 as well as at Receiver-2 end
Step-9: Created/Edited Packet Drop script at Linux Router
Step-10: Ran TCPDump at all participating interfaces of all four machines
Step-11:Set Crontab at Receiver-1 and Receiver-2 ends (initiates lftp script, establishes connection with their respective Sender machines and fetch 100MB files at the same time)
Step-12: Setup Crontab for Packet Drop script only on Drop scenarios

Step-13: Waited until file transfer is done and then carried on with the post process

Post Process

Cleaned crontab at Receiver Machines as well as from Linux Router machine
Stopped tcpdump at all interfaces and killed tcpdump processes
Used Tcptrace and Wireshark for all the collected tcpdump traces for analysis

2.5.2 Procedure Details

STEP-0: Confirmed on all machines that there's no TC policy implemented at ingress as well egress before each scenario.

```
user2@ubuntu:~$ sudo tc qdisc show dev eth1
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
user2@ubuntu:~$ sudo tc qdisc show dev eth2
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1

user2@ubuntu:~$ sudo tc qdisc del dev eth2 root
RTNETLINK answers: No such file or directory

user2@ubuntu:~$ sudo tc qdisc del dev eth1 root
RTNETLINK answers: No such file or directory
user2@ubuntu:~$
```

STEP-1: Ran NTP server at Linux Router and synchronize it with time.nrc.ca and run Daemon

```
user2@ubuntu:~$ sudo /etc/init.d/ntp stop
    1.  Stopping NTP server ntpd

user2@ubuntu:~$ sudo ntpdate time.nrc.ca
 11 Dec12:27:46 ntpdate[6137]: adjust time server 132.246.168.164 offset 0.000003 sec

user2@ubuntu:~$ sudo /etc/init.d/ntp start
    5.  Starting NTP server ntpd                              [ OK ]
```

STEP-2: Ran NTP Client as well as NTP Daemon on Sender-1, Receiver-1 and Sender-2, Receiver-2 ends and synched them all with the server

```
mint4@mint4:~$ sudo /etc/init.d/ntp stop
    2.  Stopping NTP server ntpd
```

```
mint4@mint4-desktop:~$ sudo ntpdate 192.168.2.1
 11 Dec12:44:32 ntpdate[30160]: adjust time server 192.168.10.1 offset 0.000004 sec

mint4@mint4-desktop:~$ sudo /etc/init.d/ntp start
        Starting NTP server ntpd                                    [ OK ]
```

STEP-3: Created 100 MB data files on the dot at Sender-1 and Sender-2 ends

```
 mint4@mint4-desktop:~/ftp-files$sudo time dd if=/dev/zero of=100MB.dat bs=102400
count=1024

1024+0 records in

1024+0 records out

104857600 bytes (105 MB) copied, 1.37321 s, 76.4 MB/s

0.00user 0.19system 0:01.37elapsed 13%CPU (0avgtext+0avgdata 0maxresident)k

0inputs+204800outputs (0major+286minor)pagefaults 0swaps
```

STEP-4: Added a fixed delay of 100ms at Sender-2 machine

```
mint1@ubuntu:~$ sudo tc qdisc del dev eth1 root
mint1@ubuntu:~$ sudo tc -s qdisc ls dev eth1
qdisc pfifo_fast 0: root bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 rate 0bit 0pps backlog 0b 0p requeues 0

mint1@ubuntu:~$ sudo tc qdisc add dev eth1 root netem delay 100ms
```

STEP-5: Set TCP Variant at Sender-1,Receiver-1 and Sender-2,Receiver-2 ends as per the scenario by using

```
mint4@mint4-desktop:~$ sudo sysctl net.ipv4.tcp_congestion_control
net.ipv4.tcp_congestion_control = reno

mint4@mint4-desktop:~$ sudo sysctl -w net.ipv4.tcp_congestion_control=xxxxx
```

STEP-6: Ran IPerf to test current bandwidth on the link

```
user2@ubuntu:~$ sudo iperf -c 192.168.40.2
```

```
'------------------------------------------------------------
Client connecting to 192.168.40.2, TCP port 5001
TCP window size:   124 KByte (default)
------------------------------------------------------------
[  3] local 192.168.200.2 port 38348 connected with 192.168.40.2 port 5001
[  3]  0.0-10.0 sec    114 MBytes  95.1 Mbits/sec
user2@ubuntu:~$ '
```

Run IPERF client to measure bottlenect bandwidth Simultaneous bi-directional

```
user2@ubuntu:~$ sudo iperf -c 192.168.40.2 -r
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 192.168.40.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 192.168.200.2 port 53902 connected with 192.168.40.2 port 5001
[  5]  0.0-10.0 sec    113 MBytes  94.8 Mbits/sec
[  4] local 192.168.200.2 port 5001 connected with 192.168.40.2 port 51125
[  4]  0.0-10.1 sec    114 MBytes  94.1 Mbits/sec
user2@ubuntu:~$
```

STEP-7: Ran VSFTPD at Sender-1 as well as at Sender-2 end

```
mint1@ubuntu:~$ sudo /etc/init.d/vsftpd restart

 * Stopping FTP server: vsftpd                                              [ OK ]

 * Starting FTP server: vsftpd                                              [ OK ]

user2@ubuntu:~$
```

STEP-8: Created/Edited LFTP scripts at Receiver-1 as well as at Receiver-2 end

```
#! /bin/bash

export user=yahya1
export pass=honda126
export server=192.168.100.2
export send=100MB.dat
export serverdir=/home/yahya1/Dec2010
export clientdir=/home/mint/Dec2010/R1-Dec11
                                        /*Establish connection with the server
sudo lftp -u $user,$pass -p 9000 $server <<!EOF!
cd $serverdir                           /*change remote directory at FTP server
lcd $clientdir                          /*change local directory at FTP client
get $send                               /*Fetch a file
quit
!EOF!
```

STEP-9: Created/Edited Packet Drop script at egress of Linux Router

```
#! /bin/bash

#echo "Clearout Everything"
sudo tc qdisc del dev eth1 root
#sudo tc qdisc del dev eth2 ingress

#echo "implement No Packet Drop at Egress"
sudo tc qdisc add dev eth1 root netem loss 0%

#echo "Drop All Packets for specified period of time"
sleep 4
sudo tc qdisc change dev eth1 root netem loss 100%

#Drop the packet for the period of 100ms
sleep 0.1
```

```
sudo tc qdisc change dev eth1 root netem loss 0%

echo "End of TC\Netem Script"
```

STEP-10: Ran TCPDump at six interfaces of all five machines

```
user2@ubuntu:~$ sudo tcpdump -i eth0 tcp and port 60001 and host 192.168.40.2 -pU -w
/home/mint/Dec2010/S2-Dec11/Combo1-100MB-BW100Mbps-S2R2SACK-
S2FixD100ms-S1R1CUBIC-S1FixedDelay20ms-NoDrop-S1.log
```

STEP-11: Set Crontab at Receiver-1 and Receiver-2 ends (initiated lftp script, established connection with their respective Sender machines and fetched 100MB files at the same time)

```
mint4@ubuntu:~$sudo crontab -e
```

Step-12:Setup Crontab for Packet Drop script only on Drop scenarios

Step-13Waited until file transfer was done and then carried on with the post process

3 Experiment Results

In this chapter, we illustrated our tests results. We put together the raw data such as elapsed time, total packets, average window advertised, and throughput and average round trip time and summarized them according to the groups they belong.

The naming convention of the log file name is as under:
Baseline – No competing and concurrent TCP flow, only one stream with one TCP variant at a time
Combo – Two competing and concurrent TCP flows along with two TCP variants
Transferred file size – 100MB
Sender Nodes – S1 and S2
Receiver Nodes – R1 and R2
TCP Variants – SACK, CUBIC, HIGHSPEED, and WESTWOOD
Bandwidth – 100Mbps
Delay Introduced - FixedD , NoFixedD , FixD100ms , FixedDelay20ms, FixedDelay100ms , FixedDelay200ms , and FixedDelay400ms.

3.1 Group-1 Results

3.1.1 GROUP-1 Raw Results

filename:	Baseline1-100MB-R1-SACK-BW100Mbps-NoDrop-NoFixedD-S1.log
elapsed time:	0:00:09.288383
total packets:	106823
avg win adv:	5888 bytes
throughput:	11280069 Bps = 90.24Mbps
RTT avg:	5.3 ms

filename:	Baseline2-100MB-R1-CUBIC-BW100Mbps-NoDrop-NoFixedD-S1.log
elapsed time:	0:00:09.231627
total packets:	106758
avg win adv:	5888 bytes
throughput:	11337969 Bps = 90.7 Mbps
RTT avg:	5.3 ms

File name:	Baseline3-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-NoFixedD-S1.log
elapsed time:	0:00:09.204585
total packets:	106891
avg win adv:	5888 bytes
throughput:	11391888 Bps = 91.13 Mbps
RTT avg:	5.3 ms

```
filename:       Baseline4-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-NoFixedD-
S1.log
elapsed time:   0:00:09.198506
total packets:  106778
avg win adv:    5888 bytes
throughput:     11396046 Bps = 91.16 Mbps
RTT avg:        5.3 ms
```

3.1.2 Analysis of Baseline Scenario 1-4

The above baseline scenarios have been carried out with an individual TCP connection. A 100MB data file was transferred from Sender1 to Receiver1on 100Mbps bandwidth utilizing FTP by using SACK, CUBIC, HIGHSPEED and WESTWOOD TCP variants respectively with no forced packets drops and no fixed delays. Since there's no competing TCP session, tcp fairness question does not arise. However, it's clear that each TCP session achieved the most (above 90%) of the throughput and the bandwidth capacity appears to be utilized at optimal.

```
filename:       Baseline5-100MB-R1-SACK-BW100Mbps-NoDrop-FixedD100ms-
S1.log
elapsed time:   0:00:14.221256
total packets:  105892
avg win adv:    5888 bytes
throughput:     7373301 Bps = 58.98 Mbps
RTT avg:        1.8 ms
```

```
filename:       Baseline6-100MB-R1-CUBIC-BW100Mbps-NoDrop-FixedD100ms-
S1.log
elapsed time:   0:00:10.277454
total packets:  105860
avg win adv:    5888 bytes
throughput:     10202682 Bps = 81.6 Mbps
RTT avg:        6.1 ms
```

```
filename:       Baseline7-100MB-R1-HIGHSPEED-BW100Mbps-NoDrop-
FixedD100ms-S1.log
elapsed time:   0:00:11.947346
total packets:  106661
avg win adv:    5888 bytes
throughput:     8776643 Bps = 70.2Mbps
RTT avg:        2.4 ms
```

```
filename:       Baseline8-100MB-R1-WESTWOOD-BW100Mbps-NoDrop-
FixedD100ms-S1.log
```

```
elapsed time:   0:00:46.206613
total packets:  107616
avg win adv:    5888 bytes
throughput:     2269320 Bps = 18.15Mbps
RTT avg:        1.3 ms
```

3.1.3 Analysis of Baseline Scenario 5-8

The above baseline scenarios have been conducted with an individual TCP connection. A 100MB data file was transferred from Sender1 to Receiver1on 100Mbps bandwidth utilizing FTP by using SACK, CUBIC, HIGHSPEED and WESTWOOD TCP variants respectively with no forced packets drops but with fixed delay of 100ms at the Sender end.

Since there’s no competing TCP session, fairness can not be measure but, it appears that file transfer took only 10 to 14sec in all TCP sessions except the Baseline8.Throughput on these scenarios reduced with the induction of fixed delay as compare to the previous case. As Baseline8 suffered the most, I repeated this scenario for four times and ended up with the same result, and interesting enough, there were no retransmissions at all whatsoever. Then I ran the same scenario with CUBIC, HIGHSPEED and SACK the result turns to normal especially elapsed time come back to 10s-14s and higher throughputs from 58.98-81.6Mbps. It appears that WESTWOOD did not behave as it should for some reason.

3.2 GROUP-2 Results

3.2.1 GROUP-2 Raw Results

```
filename:       Combo1-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-
S1FixedDelay20ms-NoDrop-S1.log
elapsed time:   0:00:17.971444
total packets:  107330
avg win adv:    5840 bytes
throughput:     5834679 Bps = 46.67Mbps
RTT avg:        3.8 ms
```

```
filename:       Combo2-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay20ms-NoDrop-S1.log
elapsed time:   0:00:17.976913
total packets:  107442
avg win adv:    5840 bytes
throughput:     5832903 Bps = 46.66Mbps
RTT avg:        3.8 ms
```

```
filename:       Combo3-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay20ms-NoDrop-S1.log
elapsed time:   0:00:18.001697
total packets:  106824
avg win adv:    5840 bytes
throughput:     5824873 Bps = 46.59Mbps
RTT avg:        3.8 ms


filename:       Combo4-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixedDelay100ms-NoDrop-S1.log
elapsed time:   0:00:17.908259
total packets:  99834
avg win adv:    5856 bytes
throughput:     5855265 Bps = 46.84Mbps
RTT avg:        7.3 ms


filename:       Combo5-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay100ms-NoDrop-S1.log
elapsed time:  0:00:17.898431
total packets:  106719
avg win adv:    5856 bytes
throughput:     5858480 Bps = 46.86 Mbps
RTT avg:        6.9 ms


filename:       Combo6-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay100ms-NoDrop-S1.log
elapsed time:   0:00:17.877006
total packets:  106783
avg win adv:    5856 bytes
throughput:     5865501 Bps = 46.92 Mbps
RTT avg:        6.7 ms


filename:       Combo7-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixedDelay200ms-NoDrop-S1.log
elapsed time:   0:00:20.710211
total packets:  107454
avg win adv:    5888 bytes
throughput:     5063087 Bps = 40.5 Mbps
RTT avg:        32.8 ms
```

```
filename:       Combo8-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay200ms-NoDrop-S1.log
elapsed time:   0:00:23.284622
total packets:  107754
avg win adv:    5888 bytes
throughput:     4503298 Bps = 36.02Mbps
RTT avg:        6.5 ms


filename:       Combo9-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay200ms-NoDrop-S1.log
elapsed time:   0:01:29.158408
total packets:  108063
avg win adv:    5888 bytes
throughput:     1176082 Bps = 9.40 Mbps
RTT avg:        2.5 ms


filename:      Combo10-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-CUBIC-
S1FixedDelay400ms-NoDrop-S1.log
elapsed time:   0:00:36.513118
total packets:  108365
avg win adv:    5888 bytes
throughput:     2871779 Bps = 22.97Mbps
RTT avg:        3.8 ms


filename:       Combo11-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay400ms-NoDrop-S1.log
elapsed time:   0:00:47.164202
total packets:  108481
avg win adv:    5888 bytes     avg win adv:       3680970 bytes
throughput:     2223246 Bps = 17.78 Mbps
RTT avg:        2.5 ms


filename:       Combo12-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay400ms-NoDrop-S1.log
elapsed time:   0:03:09.842869
total packets:  108688
avg win adv:    5888 bytes
throughput:     552339 Bps = 4.42 Mbps
RTT avg:        2.6 ms
```

```
filename:       Combo13-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixedDelay20ms-Drop100ms-S1.log
elapsed time:   0:00:16.779011
total packets:  107189
avg win adv:    5888 bytes
throughput:     6249331 Bps = 49.99 Mbps
RTT avg:        1.7 ms
rexmt data pkts:        182


filename:       Combo14-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay20ms-Drop100ms-S1.log
elapsed time:   0:00:16.000658
total packets:  105397
avg win adv:    5888 bytes
throughput:     6553330 Bps = 52.42 Mbps
RTT avg:        1.8 ms
rexmt data pkts:        192


filename:       Combo15-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay20ms-Drop100ms-S1.log
elapsed time:   0:00:24.578703
total packets:  107134
avg win adv:    5888 bytes
throughput:     4266197 Bps = 34.13 Mbp
RTT avg:        1.7 ms
rexmt data pkts:        153


filename:       Combo16-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixedDelay100ms-Drop100ms-S1.log
elapsed time:   0:01:09.602114
total packets:  108062
avg win adv:    5856 bytes
throughput:     1506529 Bps = 12.05Mbps
RTT avg:        1.2 ms
exmt data pkts:        128


filename:       Combo17-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixedDelay100ms-Drop100ms-S1.log
elapsed time:   0:01:00.916180
total packets:  108181
avg win adv:    5856 bytes
throughput:     1721342 Bps = 13.77 Mbps
```

```
RTT avg:        1.2 ms
rexmt data pkts:        128

filename:       Combo18-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixedDelay100ms-Drop100ms-S1.log
elapsed time:   0:00:59.716264
total packets:  107820
avg win adv:    5856 bytes
throughput:     1755930 Bps = 14.04 Mbps
RTT avg:        1.2 ms
rexmt data pkts:        128


filename:       Combo19-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1CUBIC-S1FixedD200ms-Drop100ms-S1.log
elapsed time:   0:01:08.702101
total packets:  108135
avg win adv:    5888 bytes
throughput:     1526265 Bps = 12.21Mbps
RTT avg:        5.8 ms
rexmt data pkts:        207


filename:       Combo20-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
R1HIGHSPEED-S1FixedD200ms-Drop100ms-S1.log
elapsed time:   0:01:36.644445
total packets:  108349
avg win adv:    5888 bytes
throughput:     1084983 Bps = 8.68Mbps
RTT avg:        2.5 ms
rexmt data pkts:        320


filename:       Combo21-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
WESTWOOD-S1FixedD200ms-Drop100ms-S1.log
elapsed time:   0:01:53.715696
total packets:  108331
avg win adv:    5888 bytes
throughput:     922103 Bps = 7.37Mbps
RTT avg:        2.6 ms
rexmt data pkts:         52


filename:       Combo22-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1CUBIC-
S1FixedD400ms-Drop100ms-S1.log
elapsed time:   0:10:51.376915
```

```
total packets:  109372
avg win adv:    5840 bytes
throughput:     160978 Bps = 1.28Mbps
RTT avg:        2.5 ms
rexmt data pkts:        1


filename:       Combo23-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
R1HIGHSPEED-S1FixedD400ms-Drop100ms-S1.log
elapsed time:   0:10:49.166312
total packets:  109353
avg win adv:    5840 bytes
throughput:     161527 Bps = 1.3Mbps
RTT avg:        2.6 ms
rexmt data pkts:       0


filename:       Combo24-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
R1WESTWOOD-S1FixedD400ms-Drop100ms-S1.log
elapsed time:   0:10:48.770150
total packets:  109361
avg win adv:    5840 bytes
throughput:     161625 Bps = 1.29Mbps
RTT avg:        2.6 ms
rexmt data pkts:       0


filename:       Combo1-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD20ms-NoDrop-S2.log
elapsed time:   0:01:20.936827
total packets:  107758
avg win adv:   5840 bytes
throughput:     1295549 Bps = 10.36Mbps
RTT avg:        4.2 ms


filename:       Combo2-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD20ms-NoDrop-S2.log
elapsed time:   0:01:20.985613
total packets:  107829
avg win adv:    5840 bytes
throughput:     1294768 Bps = 10.35 Mbps
RTT avg:        4.3 ms
```

```
filename:      Combo3-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD20ms-NoDrop-S2.log
elapsed time:  0:01:20.744702
total packets: 107724
avg win adv:   5840 bytes
throughput:    1298631 Bps = 10.39Mbps
RTT avg:       4.3 ms


filename:      Combo4-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD100ms-NoDrop-S2.log
elapsed time:  0:01:20.845527
total packets: 107864
avg win adv:   5840 bytes
throughput:    1297012 Bps = 10.37Mbps
RTT avg:       4.4 ms


filename:      Combo5-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD100ms-NoDrop-S2.log
elapsed time:  0:01:20.804462
total packets: 107572
avg win adv:   5840 bytes
throughput:    1297671 Bps = 10.38Mbps
RTT avg:       4.4 ms


filename:      Combo6-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD100ms-NoDrop-S2.log
elapsed time:  0:01:20.937171
total packets: 107751
avg win adv:   5840 bytes
throughput:    1295543 Bps = 10.36Mbps
RTT avg:       4.2 ms


filename:      Combo7-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD200ms-NoDrop-S2.log
elapsed time:  0:00:16.564209
total packets: 106684
avg win adv:   5888 bytes
throughput:    6330371 Bps = 50.64Mbps
RTT avg:       29.7 ms
```

```
filename:       Combo8-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD200ms-NoDrop-S2.log
elapsed time:   0:00:15.172184
total packets:  106777
avg win adv:    5888 bytes
throughput:     6911174 Bps = 55.28Mbps
RTT avg:        9.5 ms


filename:       Combo9-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD200ms-NoDrop-S2.log
elapsed time:   0:00:13.911081
total packets:  106733
avg win adv:    5888 bytes
throughput:     7537703 Bps = 60.3 Mbps
RTT avg:        3.5 ms


filename:       Combo10-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD400ms-NoDrop-S2.log
elapsed time:   0:00:13.939919
total packets:  107218
avg win adv:    5888 bytes
throughput:     7522110 Bps = 60.17 Mbps
RTT avg:        6.7 ms


filename:       Combo11-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD400ms-NoDrop-S2.log
elapsed time:   0:00:14.765798
total packets:  107402
avg win adv:    5888 bytes
throughput:     7101384 Bps = 56.81 Mbps
RTT avg:        3.7 ms


filename:       Combo12-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD400ms-NoDrop-S2.log
elapsed time:   0:00:13.472012
total packets:  107411
avg win adv:    5888 bytes
throughput:     783366 Bps = 62.26 Mbps
RTT avg:        3.7 ms
```

3.2.2 Analysis of Group-2 Results

In the above combo scenarios, couple of competing and concurrent TCP flows/connections set up. Flow1 runs between Sender1 and Receiver1 and the Flow2 runs between Sender2 and Receiver2 as depicted in the above Topology diagram (Figure 2-1). Sender2 has been setup with constant 100ms delay along with invariable TCP SACK and Receiver2 is also setup with fixed TCP SACK in all the flow2 scenarios. On the other hand, Sender1 and Receiver1 have been setup with different TCP variants as well as different Delays in each scenario, thus the RTT varies in Flow1 from 20ms, 100ms, 200ms to 400ms and TCP variants change to CUBIC, HIGHSPEED, and WESTWOOD.

As can be seen from the above table-1, unlike Baseline scenarios, in Combo scenarios the performance of the TCP variants have been impacted by the active multiple TCP flows and the performance degraded by increasing the end-to-end delay. For moderate delay of 20ms and 100ms all the TCP variants in Flow1 yield a throughput of 46.79 Mbps on average whereas TCP variants in Flow2 yield around 10.36 Mbps on average. Thus CUBIC, HIGH SPEED, WESTWOOD take over greater bandwidth capacity as compare to the Flow2 TCP SACK.

For flows with larger delays (long distances), such as 200ms, the tremendous change can be seen from Flow2 with TCP SACK which captured greater part of the bandwidth, however, in Flow1, HIGHSPEED attained a lower throughput and WESTWOOD attained the lowest of only 9.4Mbps.

The performance of CUBIC, HIGHSPEED and WESTWOOD suffered from the implementation of greater delay of 400ms as flow2 with TCP SACK captured a greater part of the bandwidth about 59.74 on average, however, in Flow1, CUBIC yielded 22.97Mbps, HIGHSPEED attained lower throughput of 17.78Mbps and WESTWOOD attained the lowest of merely 4.42Mbps.

In the above scenarios, the inconsistent average round trip time (RTT) did not provide a basis for deciding a conclusion, however, fair treatment of different TCP flows should hold regardless of differences of round trip time.

Table 3-1 Group2 Raw Results

TCP Variants	**Throughput S1**	**S2**	**RTT S1**	**RTT S2**	**Elapsed Time S1**	**S2**
Delay 20ms						
1-S1 CBIC	46.67	10.36	3.8	4.2	0:00:17.971444	0:01:20.936827
2-S1 HSPD	46.66	10.35	3.8	4.3	0:00:17.976913	0:01:20.985613
3-S1 WWD	46.59	10.39	3.8	4.3	0:00:18.001697	0:01:20.744702
Delay 100ms						
4-S1 CBIC	46.84	10.37	7.3	4.4	0:00:17.908259	0:01:20.845527
5-S1 HSPD	46.86	10.38	6.9	4.4	0:00:17.898431	0:01:20.804462
6-S1 WWD	46.92	10.36	6.7	4.2	0:00:17.877006	0:01:20.937171
Delay 200ms						
7-S1 CBIC	40.5	50.64	32.8	29.7	0:00:20.710211	0:00:16.564209
8-S1 HSPD	36.02	55.28	6.5	9.5	0:00:23.284622	0:00:15.172184
9-S1 WWD	9.4	60.3	2.5	3.5	0:01:29.158408	0:00:13.911081
Delay 400ms						
10-S1 CBIC	22.97	60.17	3.8	6.7	0:00:36.513118	0:00:13.939919
11-S1 HSPD	17.78	56.81	2.5	3.7	0:00:47.164202	0:00:14.765798
12-S1 WWD	4.42	62.26	2.6	3.7	0:03:09.842869	0:00:13.472012

3.3 GROUP-3 Results

3.3.1 Group-3 Raw Results

```
filename:       Combo13-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD20ms-Drop100ms-S2.log
elapsed time:   0:01:21.666198
total packets:  107925
avg win adv:    5840 bytes
throughput:     1283978 Bps = 10.27 Mbps
RTT avg:        3.9 ms
rexmt data pkts:        86


 filename:      Combo14-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD20ms-Drop100ms-S2.log
elapsed time:   0:01:22.478850
total packets:  107825
avg win adv:    5840 bytes
throughput:     1271327 Bps = 10.17 Mbps
RTT avg:        3.8 ms
rexmt data pkts:        90


filename:       Combo15-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD20ms-Drop100ms-S2.log
elapsed time:   0:01:23.331028
total packets:  108106
avg win adv:    5840 bytes
throughput:     1258326 Bps = 10.06Mbps
RTT avg:        3.9 ms
rexmt data pkts:      89


filename:       Combo16-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
CUBIC-S1FixD100ms-Drop100ms-S2.log
elapsed time:   0:00:22.697283
total packets:  107850
avg win adv:    5856 bytes
throughput:     4619830 Bps = 36.95Mbps
RTT avg:        2.8 ms
rexmt data pkts:       491


filename:       Combo17-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
HIGHSPEED-S1FixD100ms-Drop100ms-S2.log
elapsed time:   0:00:22.268092
```

```
total packets:  107574
avg win adv:    5856 bytes
throughput:     4708872 Bps = 37.67
RTT avg:        2.7 ms
rexmt data pkts:        481


filename:       Combo18-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-R1S1-
WESTWOOD-S1FixD100ms-Drop100ms-S2.log
elapsed time:   0:00:22.338748
total packets:  107701
avg win adv:    5856 bytes
throughput:     4693978 Bps = 37.55Mbps
RTT avg:       2.9 ms
rexmt data pkts:        455


filename:       Combo19-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1CUBIC-S1FixedD200ms-Drop100ms-S2.log
elapsed time:   0:01:19.035652
total packets:  107975
avg win adv:    5888 bytes
throughput:     1326713 Bps = 10.61Mbps
RTT avg:        3.8 ms
rexmt data pkts:        554


filename:       Combo20-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1HIGHSPEED-S1FixedD200ms-Drop100ms-S2.log
elapsed time:   0:01:21.005914
total packets:  108007
avg win adv:    5888 bytes
throughput:     1294444 Bps = 10.35Mbps
RTT avg:        3.3 ms
rexmt data pkts:        579


filename:       Combo21-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1WESTWOOD-S1FixedD200ms-Drop100ms-S2.log
elapsed time:   0:01:20.524393
total packets:  108188
avg win adv:    5888 bytes
throughput:     1302184 Bps = 10.42Mbps
RTT avg:        3.4 ms
rexmt data pkts:        581
```

```
filename:       Combo22-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1CUBIC-S1FixedD400ms-Drop100ms-S2.log
elapsed time:   0:01:19.253966
total packets:  108617
avg win adv:    5888 bytes
throughput:     1323058 Bps = 10.58Mbps
RTT avg:        1.9 ms
rexmt data pkts:        566
```

```
file name:      Combo23-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1HIGHSPEED-S1FixedD400ms-Drop100ms-S2.log
elapsed time:   0:01:16.555112
total packets:  108394
avg win adv:    5888 bytes
throughput:     1369701 Bps = 10.95Mbps
RTT avg:        1.7 ms
rexmt data pkts:        519
```

```
filename:       Combo24-100MB-BW100Mbps-S2R2SACK-S2FixD100ms-
S1R1WESTWOOD-S1FixedD400ms-Drop100ms-S2.log
elapsed time:   0:00:50.551834
total packets:  108602
avg win adv:    5888 bytes
throughput:     2074259 Bps = 16.59Mbps
RTT avg:        1.6 ms
rexmt data pkts:        521
```

3.3.2 Analysis of Group-3 Results

The setup of Combo Group3 scenarios is exactly same as in Combo Group2 except the fact that these tests have been designed to have Packets Drops on purpose for the period of 100ms. The dropping period was determined by the Outstanding (Unacknowledged) Data for each scenario. Outstanding Data Graphs were generated for that reason using TCPTrace utility.

The diversity of the results is obvious from the above table. In the Delay 20ms Flow1 tests, HIGHSPEED has the highest throughput, second is CUBIC and third is WESTWOOD due to a lower RTT as compare to their counterpart TCP SACK in Flow2 scenarios. Thus Flow2 scenarios incurred the lowest throughput and the highest throughput time as a result.

In the Delay 100ms Flow1 tests, even though the capacity of the bandwidth was not utilized optimal but TCP SACK has the highest throughput among its other non-SACK TCP counterparts and lower elapsed time since both flows have the identical RTT (100ms in this case) thus the SACK TCP implementation started its recovery phase earlier than that of CUBIC, HIGHSPEED and WESTWOOD in this scenario.

In the last two “Delay 200ms” and “Delay 400ms” Flow1 tests, it appears that by increasing the end-to-end delay the performance of SACK TCP is lessen as it takes more time to recover than it did in the previous case. However, TCP SACK still received the larger proportion of the link bandwidth compared to its other counterpart non SACK TCP.

Last but not least, the average RTT in these scenarios show that the TCP flow (S2->R2) with SACK in 20ms, 100ms and 200ms took longer than the TCP flow (S1->R1) with CUBIC, HIGHSPEED and WESTWOOD, however, in 400ms scenario it is other way around. Even if the number packets dropped (within the 100ms dropping period) mostly in S2-SACK flows compared to S1-CUBIC, S1-HIGHSPEED, S1-WESTWOOD flows but TCP SACK recovered faster than its other counterpart.

Table 3-2 Group-3 Raw Results

Delay 20ms						
TCP Variants	**Throughput S1**	**S2**	**RTT S1**	**RTT S2**	**Elapsed Time S1**	**S2**
13-S1 CBIC	49.99	10.27	1.7	3.9	0:00:16.779011	0:01:21.666198
14-S1 HSPD	52.42	10.17	1.8	3.8	0:00:16.000658	0:01:22.478850
15-S1 WWD	34.13	10.06	1.7	3.9	0:00:24.578703	0:01:23.331028
Delay 100ms						
16-S1 CBIC	12.05	36.95	1.2	2.8	0:01:09.602114	0:00:22.697283
17-S1 HSPD	13.77	37.67	1.2	2.7	0:01:00.916180	0:00:22.268092
18-S1 WWD	14.04	37.55	1.2	2.9	0:00:59.716264	0:00:22.338748
Delay 200ms						
19-S1 CBIC	12.21	10.61	5.8	3.8	0:01:08.702101	0:01:19.035652
20-S1 HSPD	8.68	10.35	2.5	3.33	0:01:36.644445	0:01:21.005914
21-S1 WWD	7.37	10.42	2.6	3.4	0:01:53.715696	0:01:20.524393
Delay 400ms						
22-S1 CBIC	1.28	10.58	2.5	1.9	0:10:51.376915	0:01:19.253966
23-S1 HSPD	1.3	10.95	2.6	1.7	0:10:49.166312	0:01:16.555112
24-S1 WWD	1.29	16.59	2.6	1.6	0:10:48.770150	0:00:50.551834

4 Conclusions

In this project, we focused on the TCP fairness performance aspects within simulated WAN network in a LAB environment. Our analyses showed that almost all the experimental tests especially “forced-packet-drop tests” were lacking TCP fairness, thus it was difficult to determine the fairness performance of a particular TCP variant in the given scenarios. However, it’s clear from our results summary that the throughput TCP attained was inversely proportional to the round-trip-time thereby disciplining flows with longer RTTs.

For instance, TCP SACK was supposed to have higher performance than its other counterparts due to the fact that it acquires lesser needless re-transmissions in response to a congestion or retransmission events but it did not perform very well at the presence of larger delays. However, results do show that comparatively, it did perform far better than all other TCP variants used in our experiments, with larger RTT in the events of loss.

5 Future Work

This project can be extended further by using Gigabit link rather than Fast Ethernet and other TCP variants. Test bed topology should be designed in a way that we can utilize optimal capacity of the link may be by pumping more than two TCP flows with longer transmission flows by transferring bigger file sizes (keeping in mind the buffer sizes at the end hosts, so that no buffer overflow occurs).

This project was carried out using wired LAN; however, it could be conducted over a Wireless LAN whereby TCP fairness performance can be measured and compared.

Bibliography

[1] Enabling High Performance Data Transfers
System Specific Notes for System Administrators (and Privileged Users)
By Von Welch March 3rd, 2006; http://www.psc.edu/networking/projects/tcptune/#Linux

[2] TCP Tuning Guide - Linux TCP Tuning. ESnet Network Performance Knowledge Base. [Online] Lawrence Berkeley National Laboratory. [Cited: February 25, 2010] http://fasterdata.es.net/TCP-tuning/linux.html.

[3] Ostermann, S. tcptrace - Official Homepage. tcptrace - Official Homepage. [Online] Ohio University, 2009. [Cited: Feb 25, 2010.]
http://masaka.cs.ohiou.edu/software/tcptrace/.

[4] TCP and Linux' Pluggable Congestion Control Algorithms By René Pfeiffer
http://linuxgazette.net/135/pfeiffer.html

[5] Introduction to iproute2; Official Home Page LARTC.ORG [Cited Feb10, 2011]
http://lartc.org/howto/lartc.iproute2.html

[6] Controlling TCP Fairness in WLAN access networks Nicola Blefari-Melazzi, Andrea Detti, Alessandro Ordine, Stefano Salsano; Department of Electronic Engineering University of Rome "Tor Vergata" Via del Politecnico 1, 00133 Rome, Italy
http://twelve.unitn.it/docs/RM2-1.pdf

[7] The Transmission Control Protocol (TCP); Copyright © TopBits – [cited February 21, 2011]; http://www.tech-faq.com/tcp.html

[8] TCP WESTWOOD Home Page; TCP Westwood: Handling Dynamic Large Leaky Pipes [cited Feb15, 2011]; http://www.cs.ucla.edu/NRL/hpi/tcpw/

[9] A User's Guide to TCP Windows
http://web.archive.org/web/20080803082218/http://dast.nlanr.net/Guides/GettingStarted/TCP_window_size.html

[10] Speed Up Your Internet Connection in Ubuntu Linux – Part 2; August 3, 2007 | By: UbuntuLinuxHelp | 3 Comments; Posted in How To
http://ubuntulinuxhelp.com/speed-up-your-internet-connection-in-ubuntu-linux-part-2/

[11] TCP HEADER [cited April 10, 2011]

http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html

[12] Transmission Control Protocol; Categories: TCP/IP | Transport layer protocols; [cited Feb23, 2011]; http://en.wikipedia.org/wiki/Transmission_Control_Protocol

[13] Structure and Organization of The TCP/IP Guide [cited Dec 8, 2010] http://www.tcpipguide.com/free/t_StructureandOrganizationofTheTCPIPGuide.htm

[14] Wireless Networks Volume 11, Number 4, 383-399, DOI: 10.1007/s11276-005-1764-1 TCP Unfairness in Ad Hoc Wireless Networks and a Neighborhood RED Solution Kaixin Xu, Mario Gerla, Lantao Qi and Yantai Shu http://www.springerlink.com/content/g1780tw8pp40u883/