

Analyzing Performance of TCP during Handoff in Real Mobile Networks

Project Report

by

Lovejeet Singh

Entry No. 2010CS50285

Shantanu Chaudhary

Entry No. 2010CS50295

Siddharth Batra

Entry No. 2010CS50297

Under the guidance of

Prof. Vinay Ribeiro



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Contents

1	Introduction	2
2	Work Done	2
2.1	Parameters Studied	2
2.2	Experiment Setup	3
2.3	Software Tools Stack	4
3	Handoffs in 3G Networks	5
3.1	Theory	5
3.2	Reasons for Handoffs	6
4	Transmission Control Protocol	6
4.1	TCP Cubic	6
4.2	TCP Veno	7
4.3	TCP Westwood	8
5	Results and Observations	9
5.1	Phase 1	10
5.2	Phase 2	16
6	Conclusion	19
7	Acknowledgement	26
8	Appendix	26
9	References	27

1 Introduction

TCP is a core protocol of the internet protocol suite. TCP promises sure shot delivery as part of the transport layer. Over the passage of time, various variants of TCP have been designed which differ in terms of how aggressively they manipulate transmission windows, how they deal with packet loss, packet time outs etc. In our project, we study the performance of some variants of TCP during hand off in real mobile networks. This report presents the final phase of our study. We begin by presenting the task we undertook during the course of the project. Then we briefly discuss handoffs in mobile networks and follow up by throwing some light on the TCP variants we used for our study.

In the subsequent sections following theory about TCP, we present the data gathered and our observations. We explain as to how Westwood and Veno perform better over cubic and achieve a much higher throughput over 3G wireless networks.

2 Work Done

The study of TCP over wired/wireless links requires analysis of certain parameters of the network like latency, RTT, RSSI, throughput achieved by the connection. In this section, we explain our choice of parameters, why we chose them and how they were measured and analysed.

2.1 Parameters Studied

Congestion window variation with time

Congestion window is an index of senders transmission rate. Each of the three TCP variants follow different algorithms to manipulate the congestion window with time based on packet loss during congestion or dealing with duplicate acknowledgments. A plot of congestion window would help us to determine when the network is under congestion.

Throughput

Based on manipulation of congestion window, the throughput achieved over the connection by each of the TCP variant will vary. We aim at achieving maximum throughput over the connection and analyse how it varies with

hand offs.

RSSI

Received Signal Strength Indicator is a measurement of the power present in the received signal at the receiver radio antenna. It is a parameter that takes into account both E_c/I_0 (Ratio of received energy per chip and the interference level, in dB) and RSCP (Received Signal Code Power, which is the collected RF energy after descrambling process, represented in dBm). RSSI can give an insight into the hand off process as a hand off from one Base Transceiver Station to another Base Transceiver Station can take place if the signal strength is better at the other tower.

$$RSSI = RSCP - E_c/I_0$$

(on dB scale)

Sequence Number variation with time

A plot of sequence number variation over time tells about potential timeouts and corresponding packet loss that occurred over the connection by showing any re-transmissions.

2.2 Experiment Setup

RF Drive Testing

Drive testing refers to assessing and monitoring the Quality of Service (QoS), capacity and coverage of the radio access network (RAN) by collecting data from mobile radio network air interface measurement equipment like mobile phone or a laptop with a 3G dongle, while driving a motor vehicle. It is used by wireless carriers to assess the experience of a wireless network subscriber in a specific area and then to make directed changes in their networks so as to provide better coverage and service to their customers.

Our goal is to monitor performance of TCP during handoff in cellular networks. So by using RF drive testing we will be able to know the current Base Transceiver Station (BTS) and also monitor any handoffs by noticing change in the CELLID of the BTS which uniquely identifies it in a particular area. Also, we will be able to monitor Received signal strength indication (RSSI) to better assess the latencies caused due to handoffs.

We used an android application named RF Signal Tracker which monitors and logs the current CELLID, RSSI, mobile equipment position (Latitude

and Longitude), type of connection(EDGE or HSPA), cell-tower position (in case of EDGE). The application logs these parameters at certain irregular intervals.

2.3 Software Tools Stack

Iperf

We used iperf, a linux tool which can be used to create TCP as well as UDP data streams and measure the throughput of the connection over which these streams are created. The tool consists of a server as well as a client. The client can be used to simulate a TCP connection by sending data to the server which just dumps the received data after acknowledging it. We spawned an iperf server as a daemon on a remote machine with a public IP (agni.iitd.ac.in). Next, we executed an iperf client on our mobile linux machine and connected it to the remote iperf server for capturing data on above mentioned parameters while carrying out the RF drive test.

TCPProbe

TCPPROBE is a linux module which can be used to analyse packets. The module can be loaded as a daemon and associated with any particular port to listen on it. It prints information to a specified output file for every packet it sees on that associated port. It covers details such as sender, receiver, ports of communication, congestion window size, slow start threshold, packet size, packet sequence numbers, size in packet.

GNUPlot

It is a linux utility for plotting graphs.

AWK

AWK script is a fast scripting language that can be used to extract information from files that have data in a delimited form.

To sum up, we have tried to accomplish the following tasks:

- Studying TCP implementations in linux kernel (TCP-Cubic, TCP-Veno, TCP-Westwood)

- Configuring TCP Modules in linux kernel (Linux-Generic-3.16 on Ubuntu 14.04) (see appendix)
- Determining Tool setup
- Data gathering via RF drive testing to gather data on RSSI, throughput, congestion window variation etc.
- Qualitative as well as quantitative analysis of gathered data

3 Handoffs in 3G Networks

3.1 Theory

Cellular data networks have recently seen an explosion in their usage due to the widespread deployment of 3G technologies and the rapid proliferation of smartphones. People are increasingly using their smartphones on the go and expect always on, high quality connectivity at all times. A key network primitive that enables continuous connectivity in cellular networks is handoff, or the transfer of a devices connection from one cell sector to another.

In the Context of telecommunications, the main purpose of doing a hand off is to transfer the radio connection between radio channels, whilst maintaining an ongoing connection. Accordingly, ETSI and 3GPP define hand off as the transfer of a user's connection from one radio channel to another (can be the same or different cell).

A hand off operation that can minimize or even eliminate the delay for establishing the new connection to the new access point (AP) is called a fast hand off. If the hand off operation minimizes the data loss during the establishment of the new connection, then it is called a smooth hand off. A hand off that is both fast and smooth is called a seamless hand off.

In the recent years, Intersystem handoffs have been one of the most active areas of mobility since the overlay of heterogeneous networks has become more prominent especially in areas of dense data traffic. Continuous investigations are carried out regularly with new algorithms being proposed at a high rate trying to find the best solution for various environments, mainly to minimize the latency encountered in carrying out the procedure.

3.2 Reasons for Handoffs

The most important and very obvious criterion for hand off to occur is the degrading signal quality. Also the Cellular structure, used to enhance the spectral efficiency and provide frequency reuse ability, is a chief entry factor for hand offs to be predominant from the start. To allow for hand off, adjacent cells are designed so as to overlap and constitute areas for reception of signal from multiple base stations (BS). Some of the chief reasons for hand off are listed down:

1. Better RSSI from another CELL
2. Traffic based, load based
3. Reception Quality, BER
4. Service based
5. Speed based

4 Transmission Control Protocol

For our analysis of the TCP, we have used the following TCP variants. These variants are part of the linux kernel and can easily be plugged by the root as a privileged user. (see appendix)

4.1 TCP Cubic

TCP Cubic is the default linux congestion-control mechanism currently in use, as of 2013. Cubic is a less aggressive and more systematic derivative of BIC, in which the window is a cubic function of time since the last congestion event, with the inflection point set to the window prior to the event. Cubic is used by default in Linux kernels from version 2.6.19 to 3.1.

The cubic relation between the window and time helps TCP Cubic to be aggressive and fair to other congestion control mechanisms at the same time. The congestion window increases rapidly (function is convex) when a network ceiling is surpassed without any losses. However, Cubic also makes attempts to slow down the growth of the congestion window sharply as it

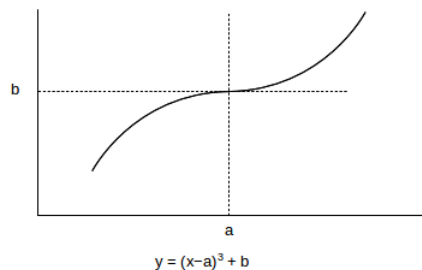


Figure 1: TCP Cubic

approaches the current network ceiling (function is concave), thus being fair to other congestion control mechanisms.

Figure 1 depicts the relation between the contention window size (Y-axis) and time (X-axis). (a,b) is the inflection point. We can see how the growth of the contention slows down as the inflection point is reached and how it catches pace after crossing this point.

4.2 TCP Veno

Veno is built over TCP Vegas and TCP Reno and is effective in dealing with random packet loss over wireless networks. TCP Reno treats the occurrence of packet loss as a manifestation of network congestion but in wireless environment apart from the network congestion, packet loss is often induced by noise, link error or some other factors not related to congestion. A key ingredient of Veno is that it monitors the network congestion level and uses that information to decide whether packet losses are likely to be due to congestion or random bit errors (random loss).

Specifically, it refines the multiplicative decrease algorithm of TCP Reno by adjusting the slow-start threshold according to the perceived network congestion level rather than a fixed drop factor. Generally speaking, TCP Veno borrows the idea of congestion detection scheme from TCP Vegas and intelligently integrates it into TCP Reno's congestion window adjustment phase. Veno involves only modification of Reno on the sender side without requiring any changes of the receiver protocol stack or intervention of the intermediate network nodes.

4.3 TCP Westwood

Westwood is a sender side modification of TCP congestion window algorithm TCP New Reno. It tries to improve performance of the congestion algorithm on both wired as well as wireless links. TCP cubic is equally sensitive to random loss and congestion loss but TCP westwood employs certain heuristics to distinguish random loss and congestion loss. Since cubic cannot distinguish between random loss over wireless link and loss due to congestion, it reacts by drastically reducing the congestion window, hence reducing the senders transmission rate, whereas, a better move is not to decrease the transmission rate.

The key idea in westwood is to use a bandwidth estimate to directly control the congestion window and the slow start threshold. This estimation of bandwidth is done by monitoring the TCP ACKs, that is, the source performs an end-to-end estimate of the bandwidth available along the TCP connection by measuring and averaging the rate of returning ACKs. To elaborate, westwood uses a discrete time varying low pass filter for the bandwidth estimation.

$$\hat{b}_k = \alpha_k \hat{b}_{k-1} + (1 - \alpha_k) \left(\frac{b_k + b_{k-1}}{2} \right)$$

where b_k is the filtered estimate of the available bandwidth at time $t=t_k$. The above filter takes care of how bandwidth estimate should be changed to accommodate duplicate acks and packet loss. The filter basically takes into account that a recent bandwidth estimate should have more significance over past bandwidth estimates. Based on whether it receives duplicate acks or packet faces a coarse timeout, the algorithms uses the following strategies:

```
if (n dupacks are received)
    ssthresh = (BWE*RTTmin) / segsize;
    if (cwin > ssthresh) /* Congestion avoidance */
        cwin = ssthresh;
    endif
endif
```

For a coarse timeout over the connection:

```
if (coarse timeout expires)
    ssthresh = (BWE*RTTmin)/segsize;
    if (ssthresh < 2)
        ssthresh = 2;
    endif;
    cwin = 1;
endif
```

ssthresh : Congestion window threshold

cwin : congestion window size

segsize : segment size

BWE : Bandwidth estimate

RTTmin : Minimum round trip time

The pseudo code for a coarse timeout sets the congestion window to 1 and threshold to the recent bandwidth estimate. This mimics the TCP Reno behavior but offers a speedy recovery at the event of a timeout.

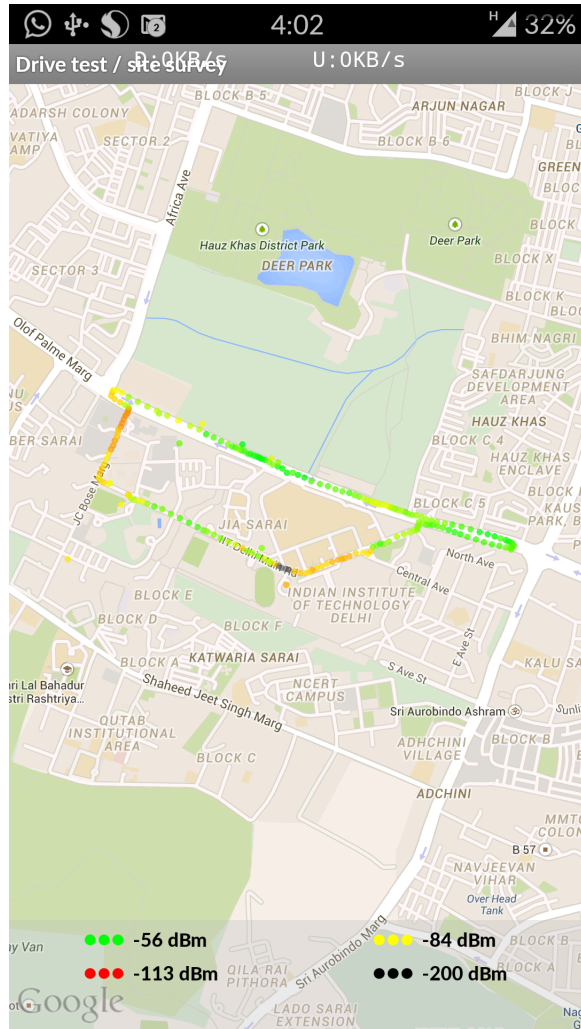
5 Results and Observations

We setup our apparatus (laptop + cell phone) in a car and performed the RF drive test. For getting a better understanding of the experiment and to avoid any mistake in the experiment, we performed the experiment in two phases. Since, we had to analyse performance of cubic, veno and westwood, we had to repeat the RF drive test thrice in each data gathering phase. We tried to maintain similar conditions for each phase. For the analysis, we have tried to relate the variation in congestion window/slow start threshold with RSSI (potential handoffs). The robustness of each of the TCP variant can be inferred by observing how much throughput each variant achieves under similar conditions of handoffs. Other parameters such as timeouts, RSSI can also be inferred from the gathered data.

5.1 Phase 1

The first phase was performed during daytime in afternoon between 3:00-5:00p.m. image below shows the route for our drive test. It took us roughly 600-700 seconds to complete each drive in this phase. The results from this phase do not help in concretely deciding which variant performs better while enduring handovers. In each of the case, there was a lot of fading observed. The only parameter that tells something about the performance is throughput.

Below are the observations and results from the first phase of the test. The map below shows the path taken by us to collect data for our first trip.



Cubic

A throughput of 49KB/sec was achieved in case of TCP Cubic. Number of handoffs that took place were 6.

Cubic is a more recent congestion-control mechanism. Cubic isn't optimised to perform on wireless networks. This is evident from the fact that it achieves the lowest throughput out of the 3 congestion-control mechanisms analyzed by us. In this phase of data collection, not many handoffs occurred and hence, we could not formulate a concrete inference on how and why Cubic is performing. However, we can see from the graph that for the first 2 or 3 handoffs, TCP Cubic handles it very well, and there isn't any congestion window falls. However, the window frequently lowers down when the signal strength lowers down. This lead to an overall large no of congestion window lowerings and hence, the total throughput is lower when compared to Veno and Westwood. Figures 2 and 3 depict the results for TCP Cubic for phase 1.

Veno

Veno achieved a throughput of 59.6KB/sec with a total of 6 handoffs. Veno is optimised to perform better for wireless networks, this is evident from the fact that there is a little variation in the contention window size and slow start threshold. The packet loss due to bit error is handled in a better way in case of TCP Veno and hence slow start threshold is not reduced drastically for these false congestion alarms. Moreover it can be seen from the graphs that the handoffs are occurring precisely at positions resulting in better RSSI values.

This phase could not actually capture the behaviour of TCP Veno completely and further observations are written with the second phase results.

Figures 4 and 5 depict the results for TCP Veno for phase 1.

Westwood

Westwood achieved a throughput of 55KB/sec. No of times handoff takes place 17. The first thing that we observe in case of westwood is that the segment size achieved by the congestion window is less than that of cubic and veno. Between 25-225 seconds of the test, in spite of the large scale fading experienced by the connection, the congestion window is maintained at a steady value. This is not the case for cubic, where it blindly reduces the window interpreting random loss over medium as packet loss. It is at time

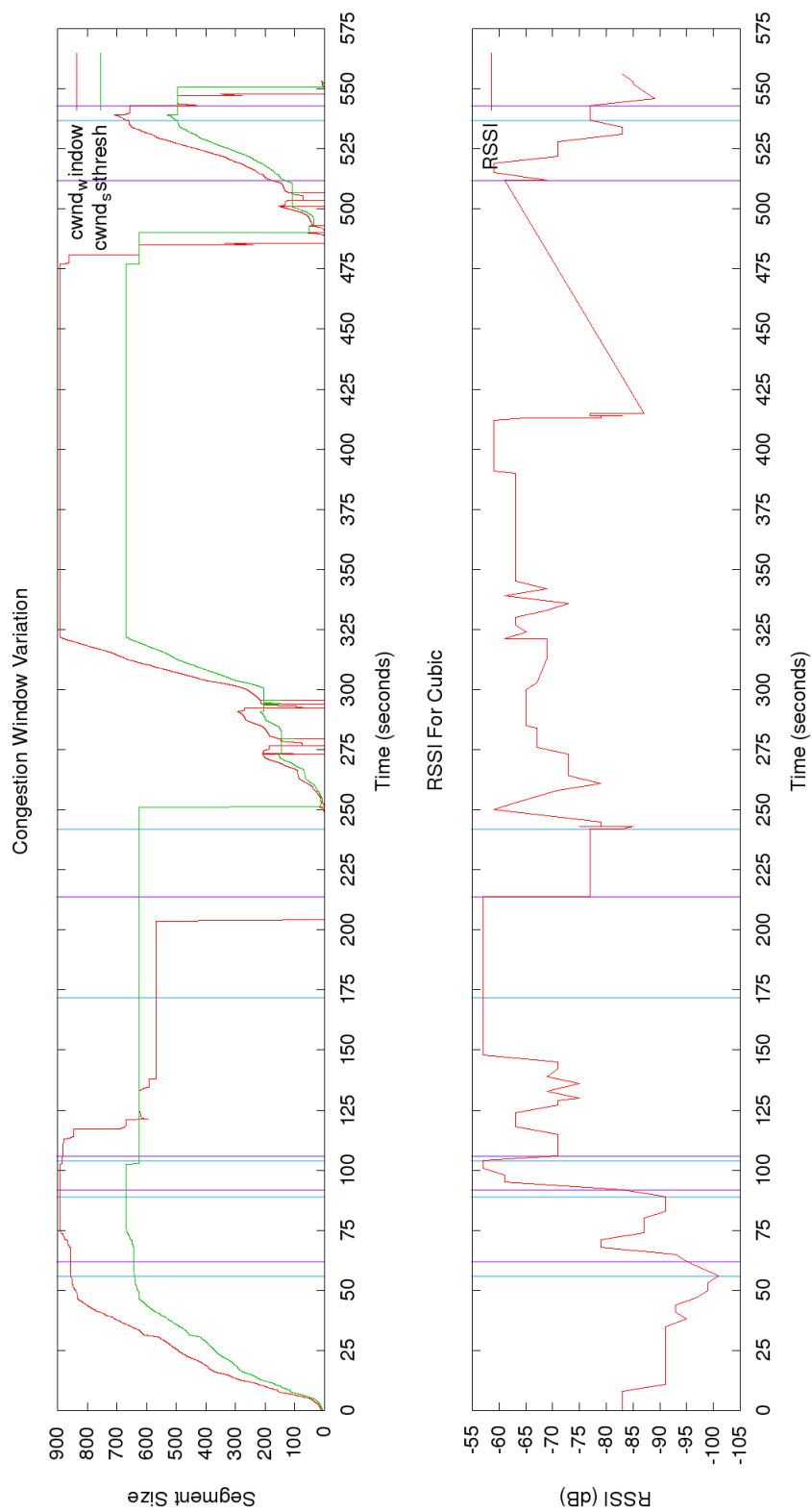


Figure 2: Cubic 1, Phase 1

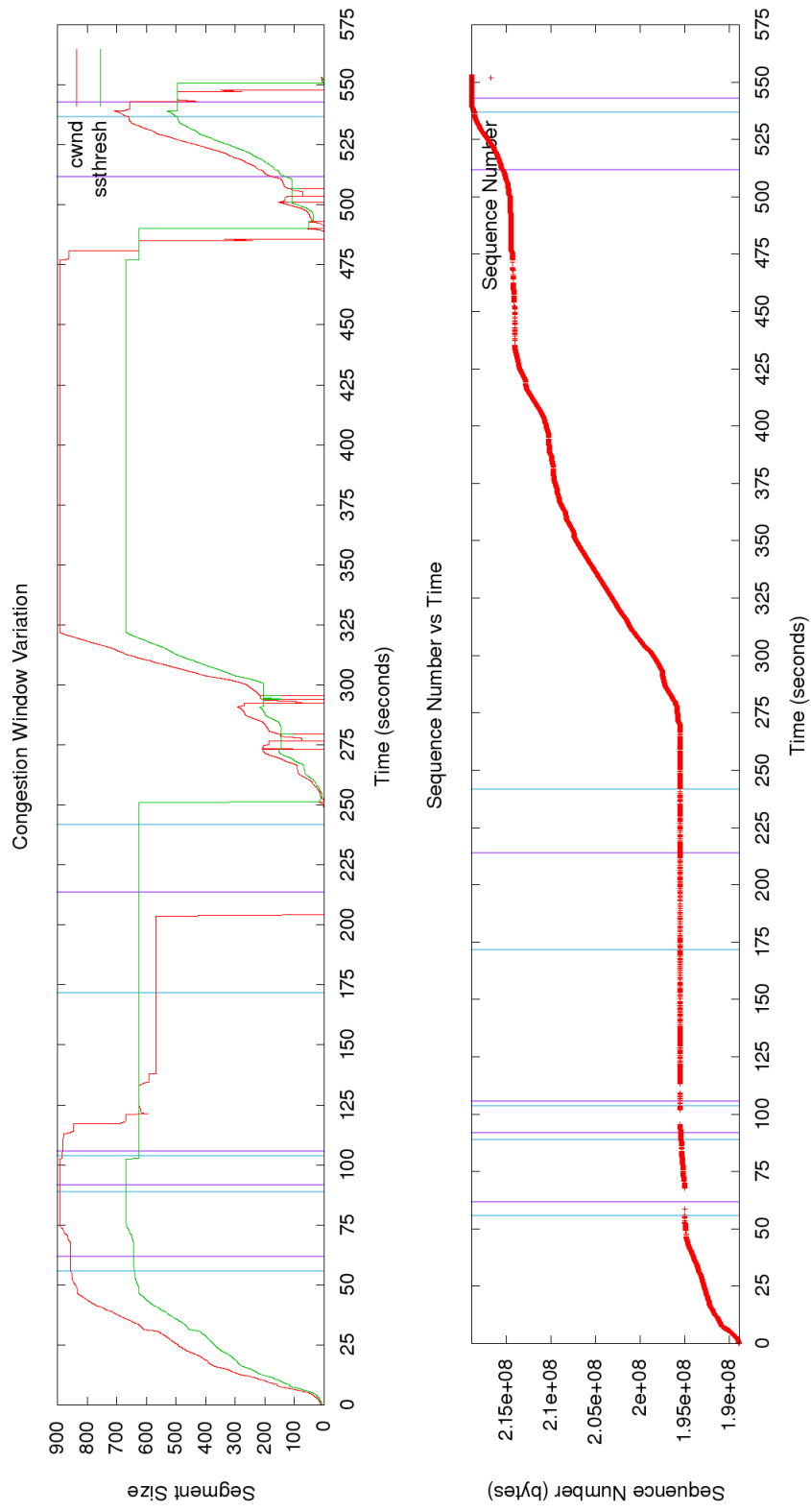


Figure 3: Cubic 2, Phase 1

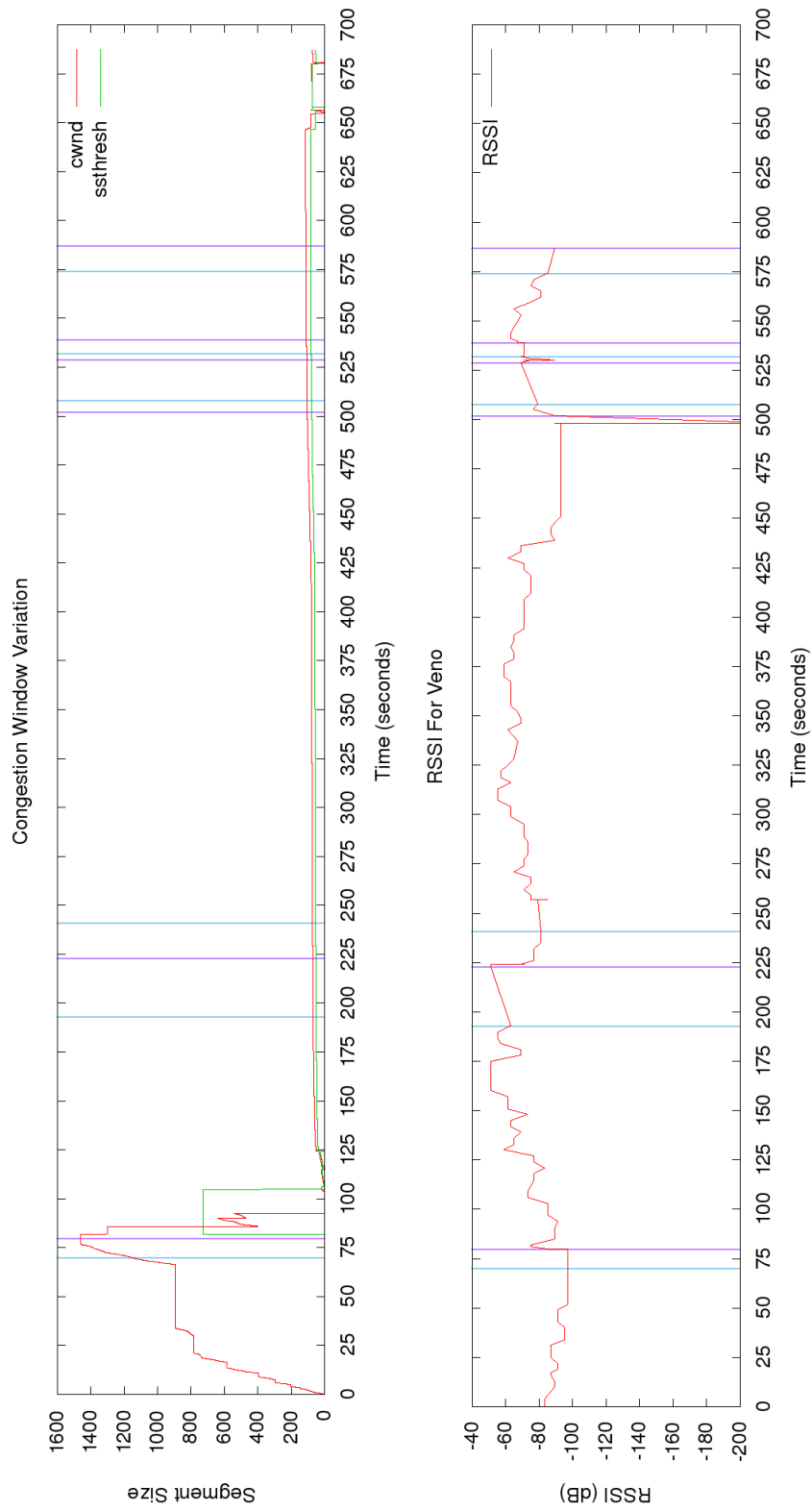


Figure 4: Veno 1, Phase 1

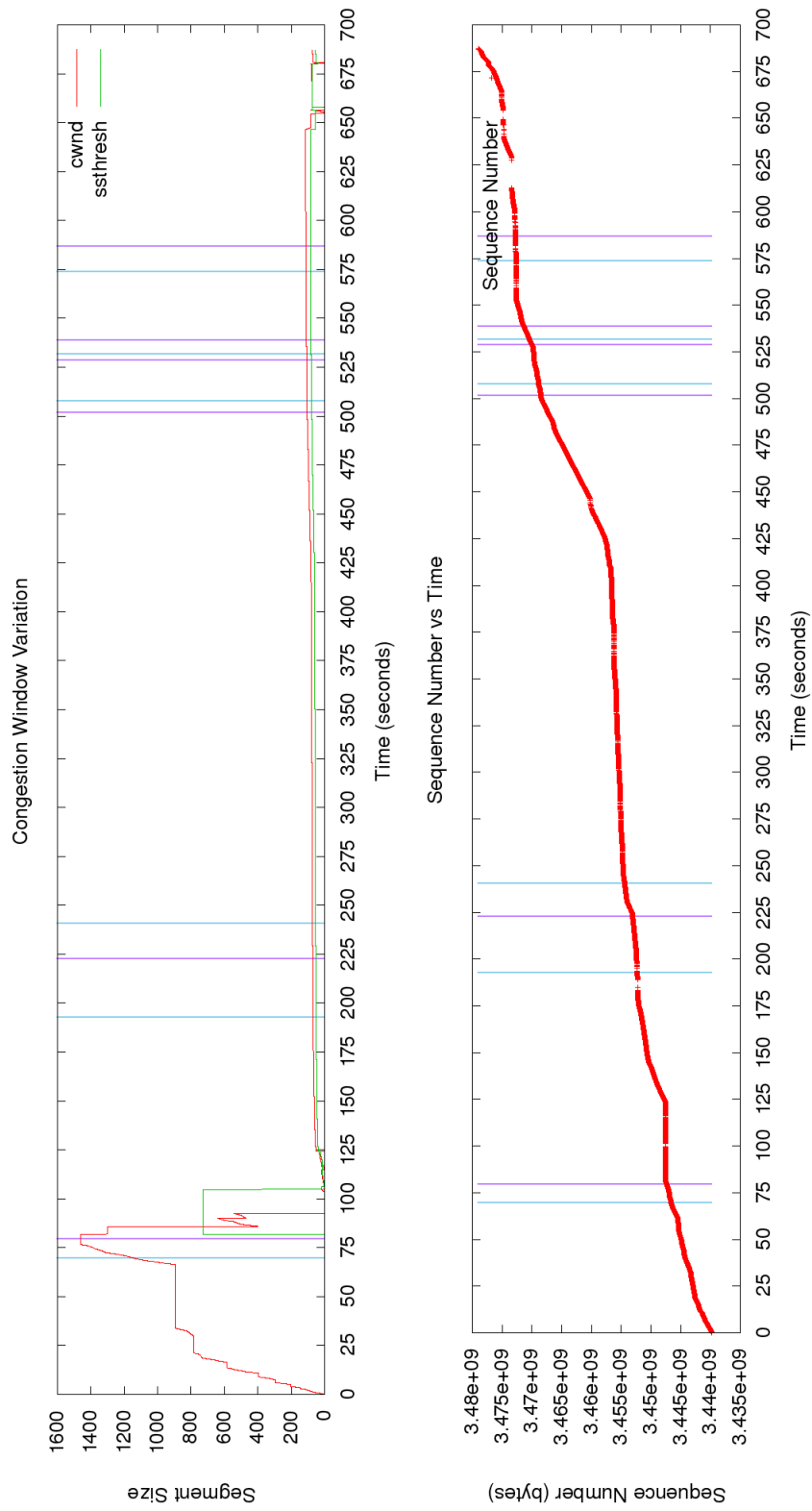


Figure 5: Veno 2, Phase 1

250 seconds that the congestion window drops to 1 when there is a handoff and RSSI decreases. The variation in window continues till $t=375$ seconds. Between $t=375-400$ sec, we speculate that the window is set according to the bandwidth estimate. The window is then additively increased steadily up to $t=625$ seconds where the connection experiences a handoff. The congestion window is then set to the recent estimate of available bandwidth of the pipe. Between $t=650-670$, there are a lot of handoffs. The congestion window experiences a lot of spikes here but is then steadily maintained. The sequence number vs time variation plot complements the observations. It is also clear that whenever RSSI increases, the slope for sequence number plot also increases.

Figures 6 and 7 depict the results for TCP Westwood for phase 1.

5.2 Phase 2

Phase 2 yielded more concrete results and observations than phase 1. The functioning of each variant became clear in this phase. The drive test was carried out between 12:30-1:30 a.m. in a manner similar to the previous one. But this time we took a different route for the drive test so as to gather more data points of cellular network. The map below shows the path taken by us to collect data for our first trip.



Cubic

A throughput of 80KB/sec was achieved in case of TCP Cubic. Number of handoffs that took place were 15.

In this phase of data collection, we could manage a significant number of handoffs, 15 to be precise compared to 6 in phase 1. Here, TCP Cubic lowers down the congestion window on almost every handoff and also on signal strength drops. TCP Cubic resets its congestion window every time a handoff occurs. Finally, we speculated that TCP Cubic is not capable of differentiating between packet loss due to congestion and handoffs, and

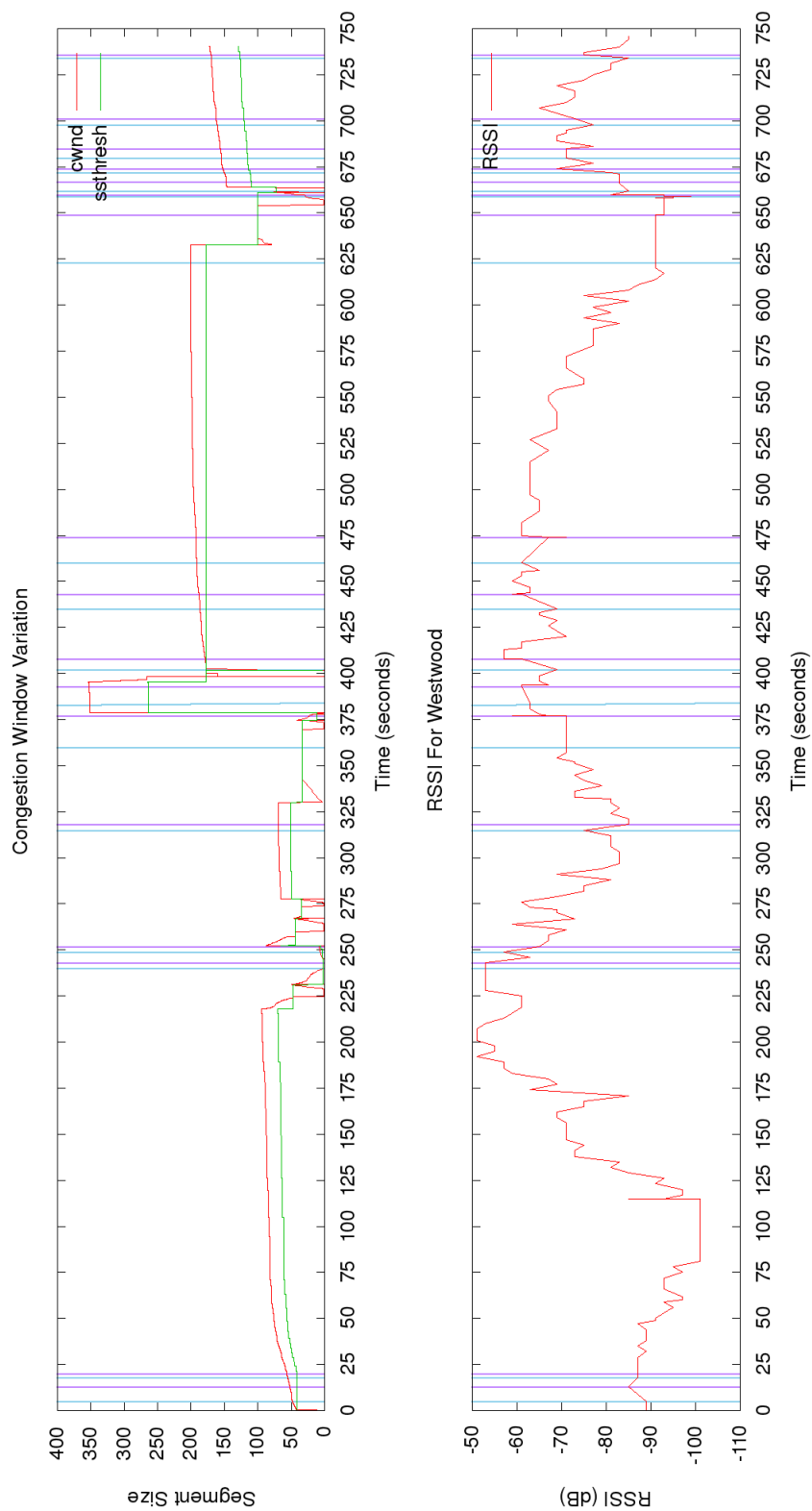


Figure 6: Westwood 1, Phase 1

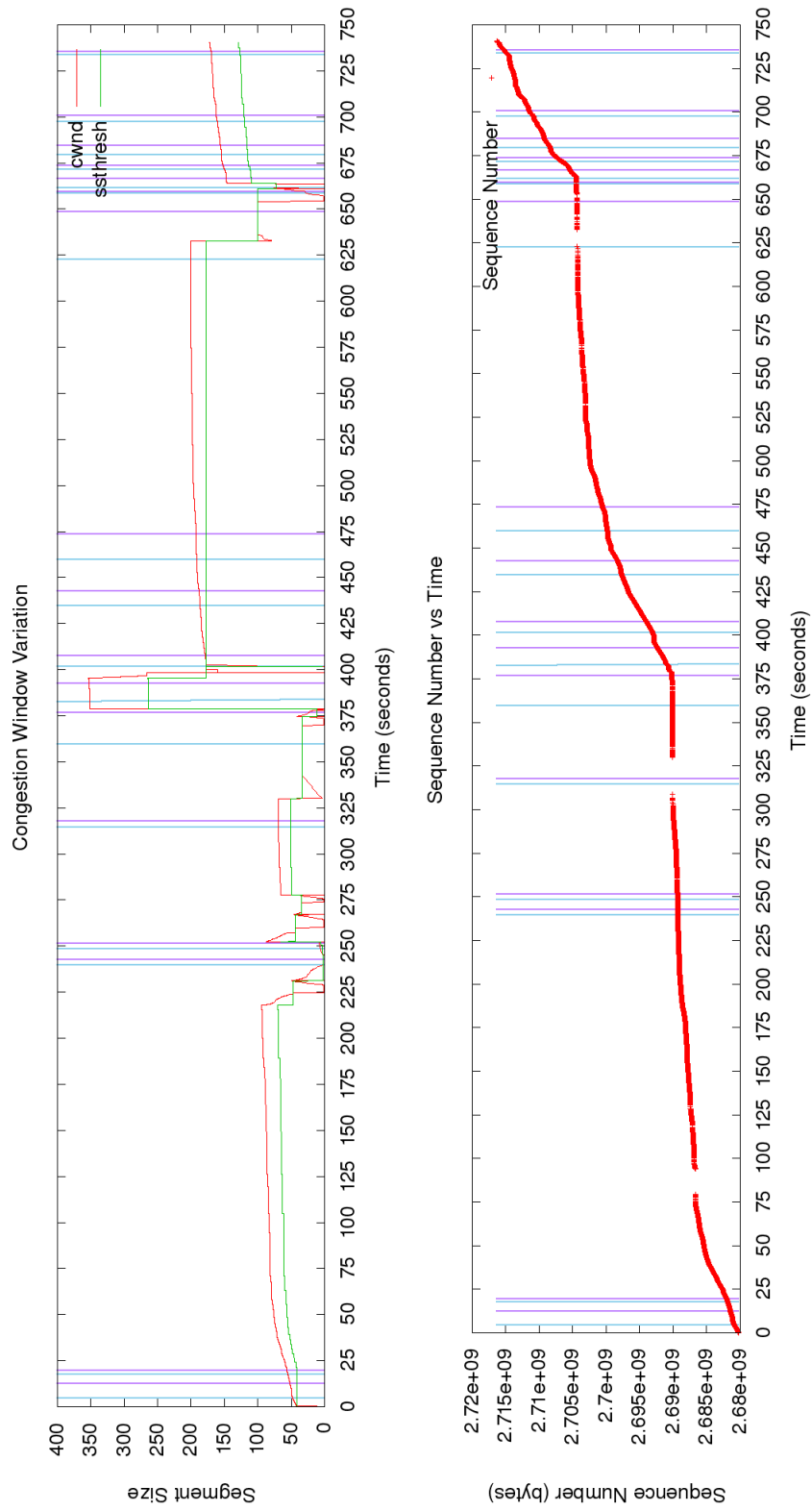


Figure 7: Westwood 2, Phase 1

thus treats every handoff a sign of congestion on the network. This leads to resetting of the congestion window very frequently and thus, leading to an overall lowest throughput of 80 KBps among the three. Figures 8 and 9 depict the results for TCP Cubic for phase 2.

Veno

Veno achieved a throughput of 196KB/s with a total of 14 handoffs. The graph for congestion window and slow start threshold for this phase sheds more light on the clever Veno congestion avoidance algorithm which uses the network congestion level to decide whether the loss is random or due to a congestion. We can clearly see that both Cubic and Westwood have experienced losses (possibly due to bit error as well) but in case of Veno we see a significant decrease in cases of inappropriate ssthreshold decrease.

Fading in the signal strength causes a lot of handoffs but Veno seems to be completely resilient to handoff and keeps performing very well over the complete phase. The plot of sequence numbers also shows a smooth plot with minimum retransmissions.

Figures 10 and 11 depict the results for TCP Veno for phase 2.

Westwood

Throughput achieved 195KB/s with Number of handovers 13 From the above graphs it is very clear that westwood succeeds in retaining a steady congestion window which is why it achieves twice the throughput than cubic. It can also be observed that between $t=0-150$ secs. There are mild congestion events at approximately $t=160$ and $t=450$. As a result, the congestion window is adjusted according to the estimated bandwidth available in the pipe. The observations in this plot are also supported by the plot of sequence number variation with time. The plot of sequence number has a steady slope which signifies minimum retransmissions in spite of significant number of handovers. Figures 12 and 13 depict the results for TCP Westwood for phase 2.

6 Conclusion

It is clear from the experiment that TCP Veno and TCP Westwood perform significantly better than the default congestion avoidance algorithm of linux. Veno and Westwood employ different heuristics for differentiating random

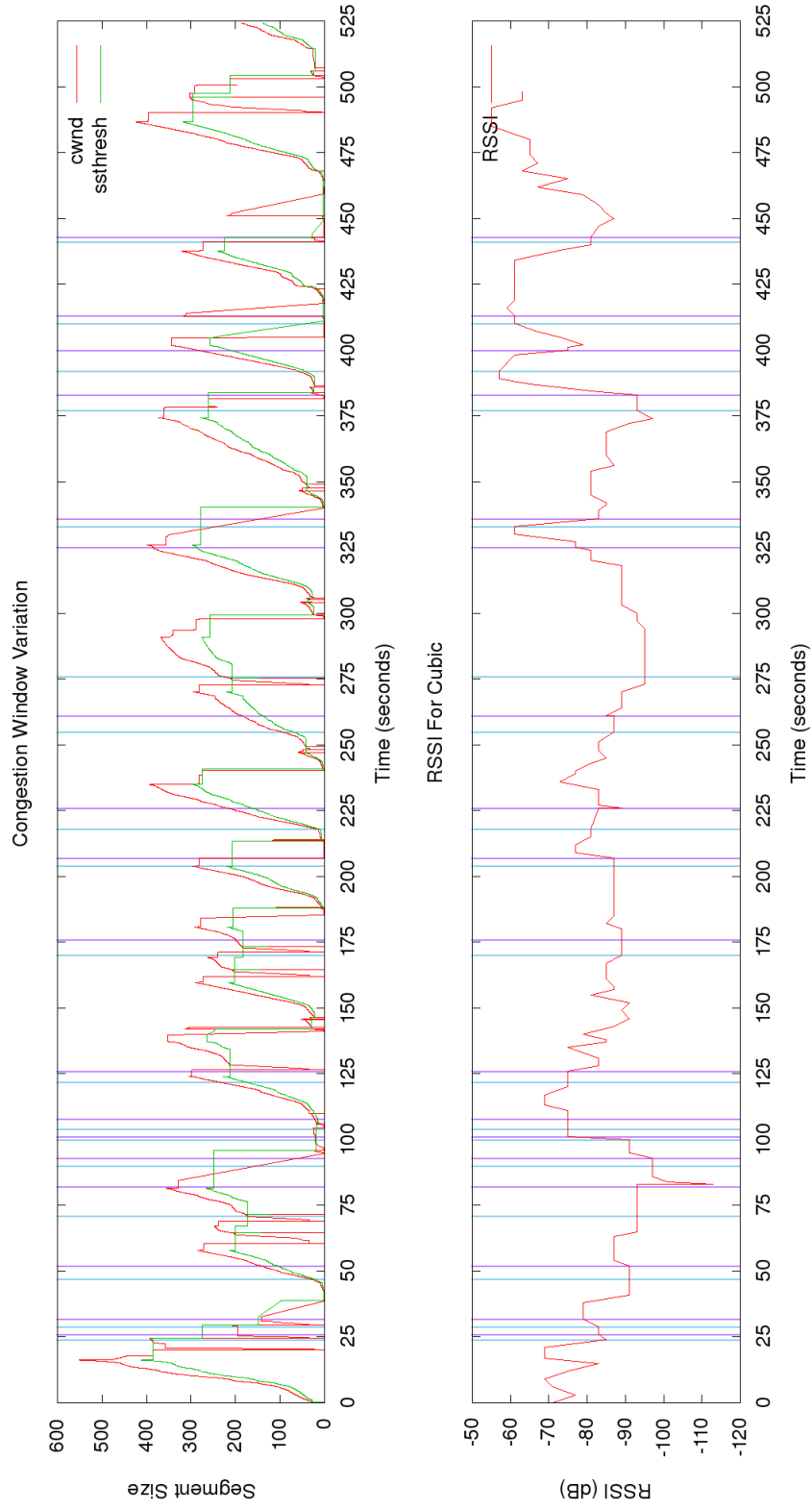


Figure 8: Cubic 1, Phase 2

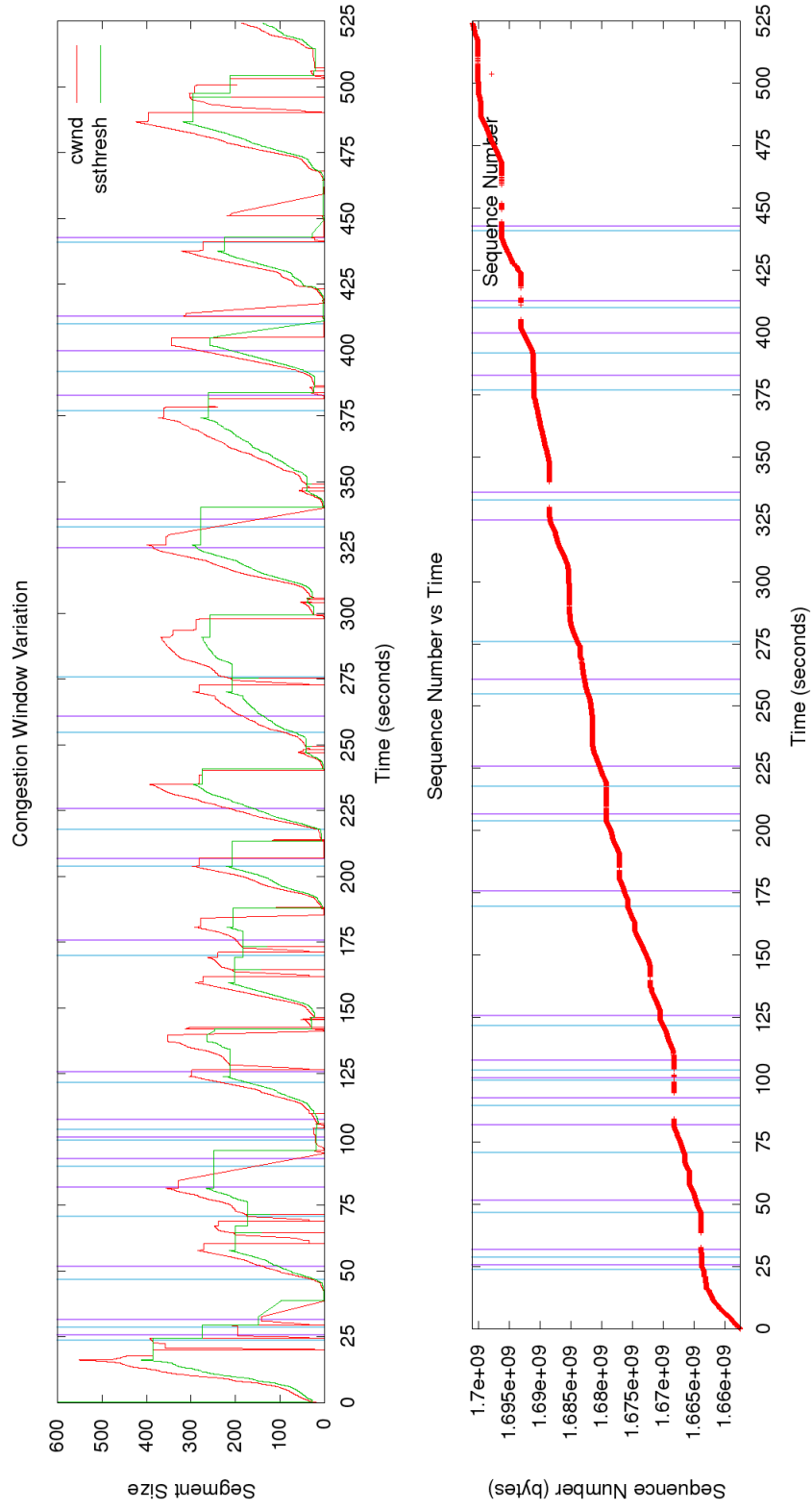


Figure 9: Cubic 2, Phase 2

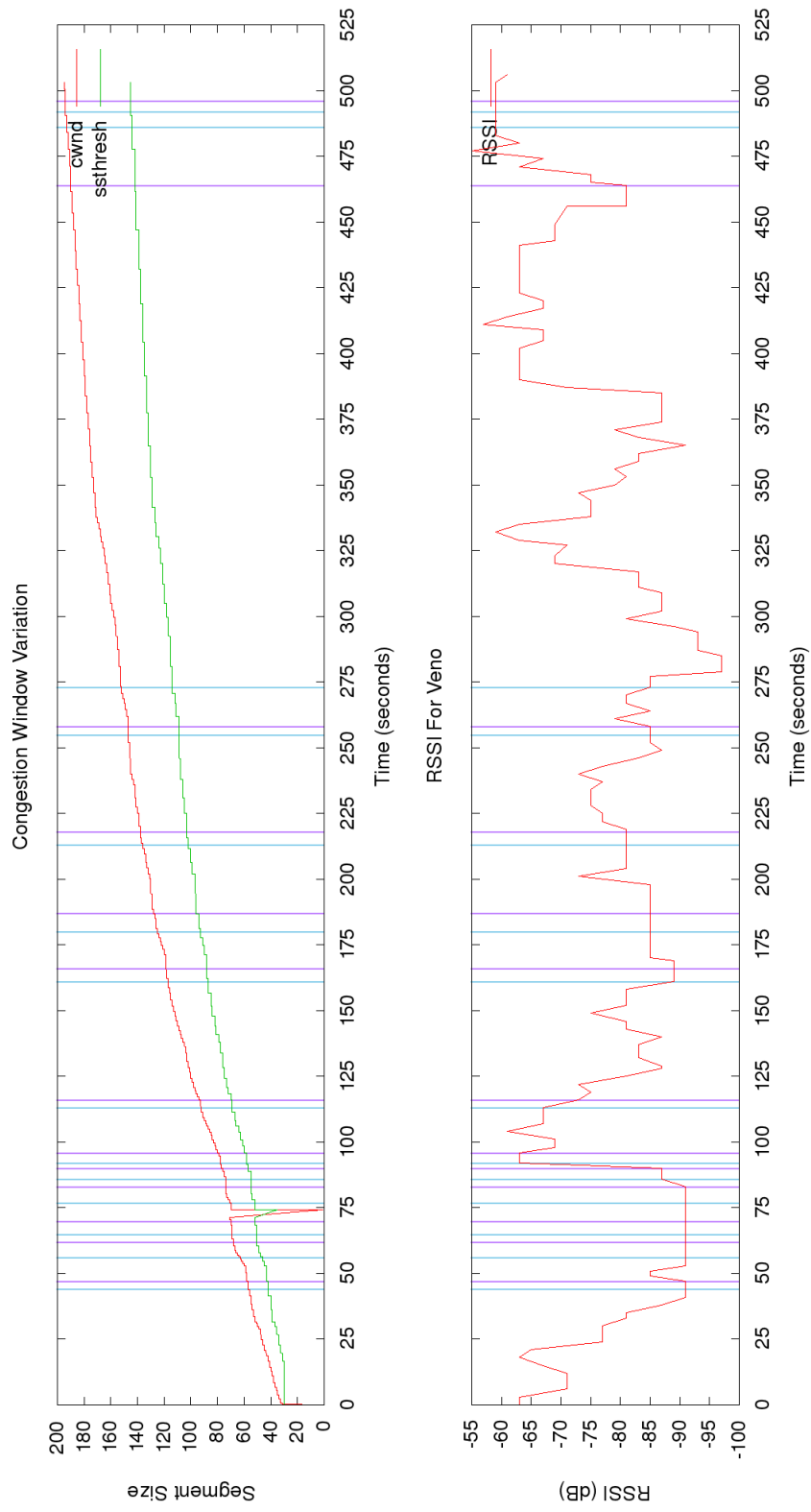


Figure 10: Veno 1, Phase 2

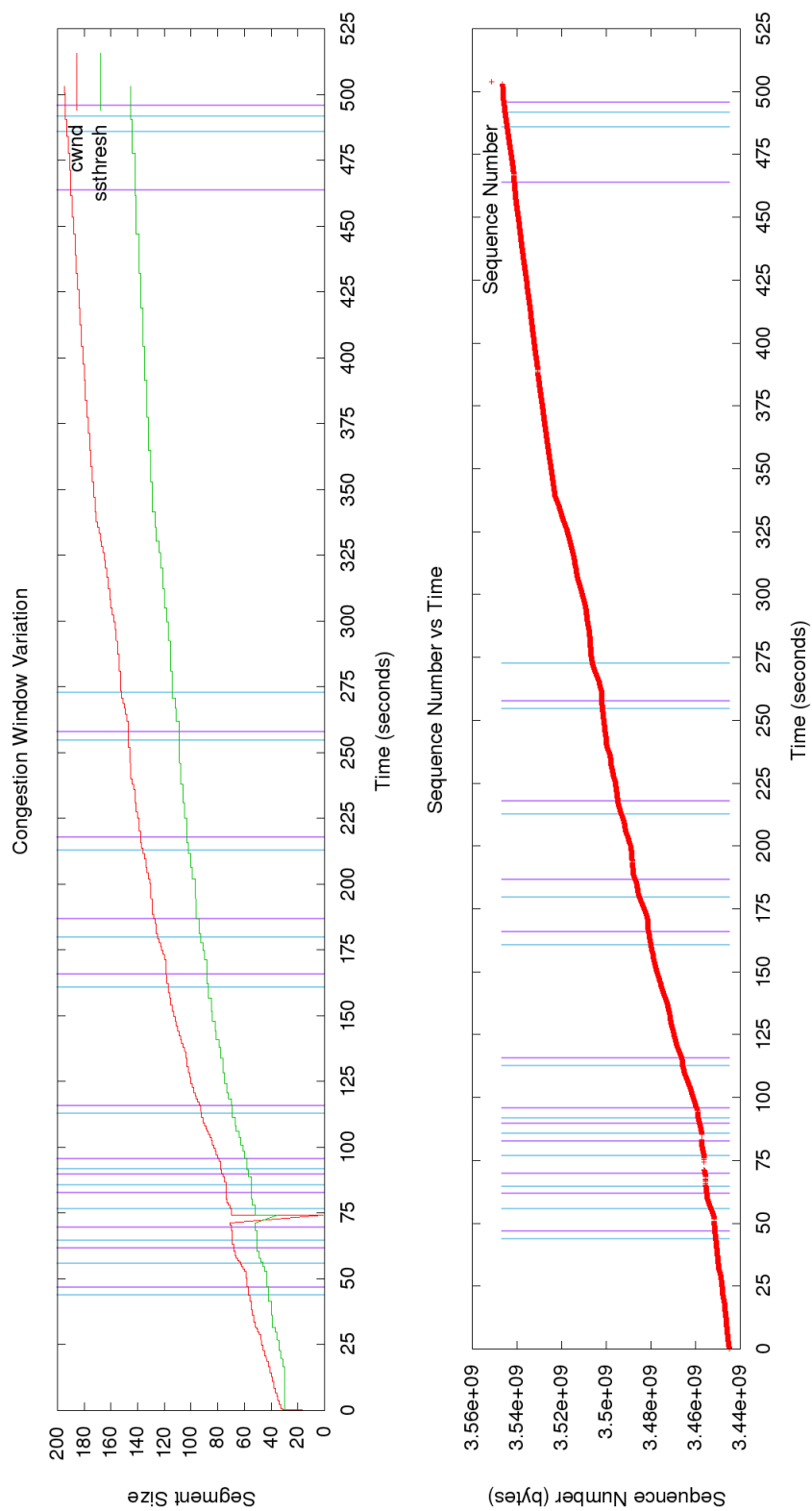


Figure 11: Veno 2, Phase 2

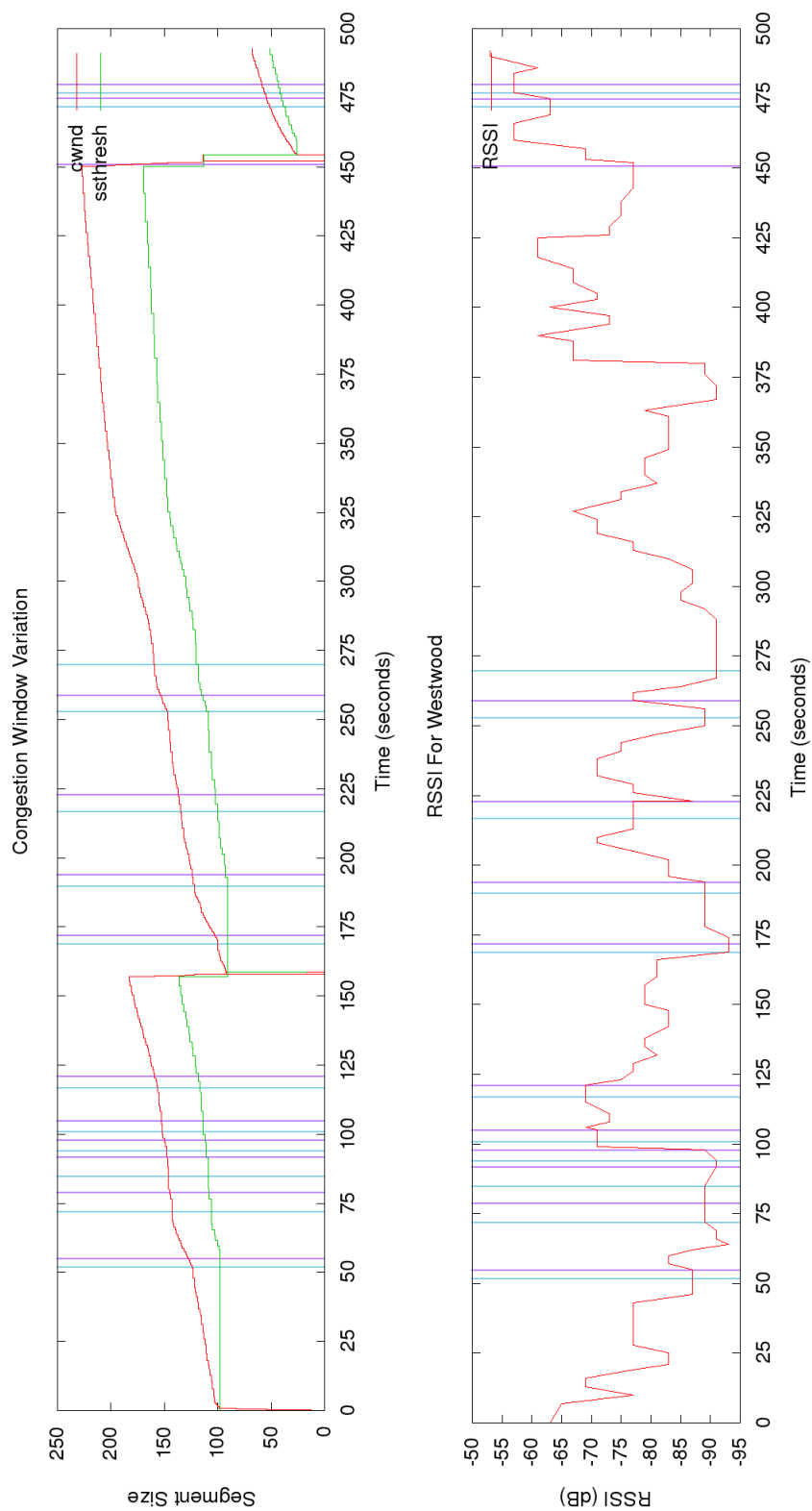


Figure 12: Westwood 1, Phase 2

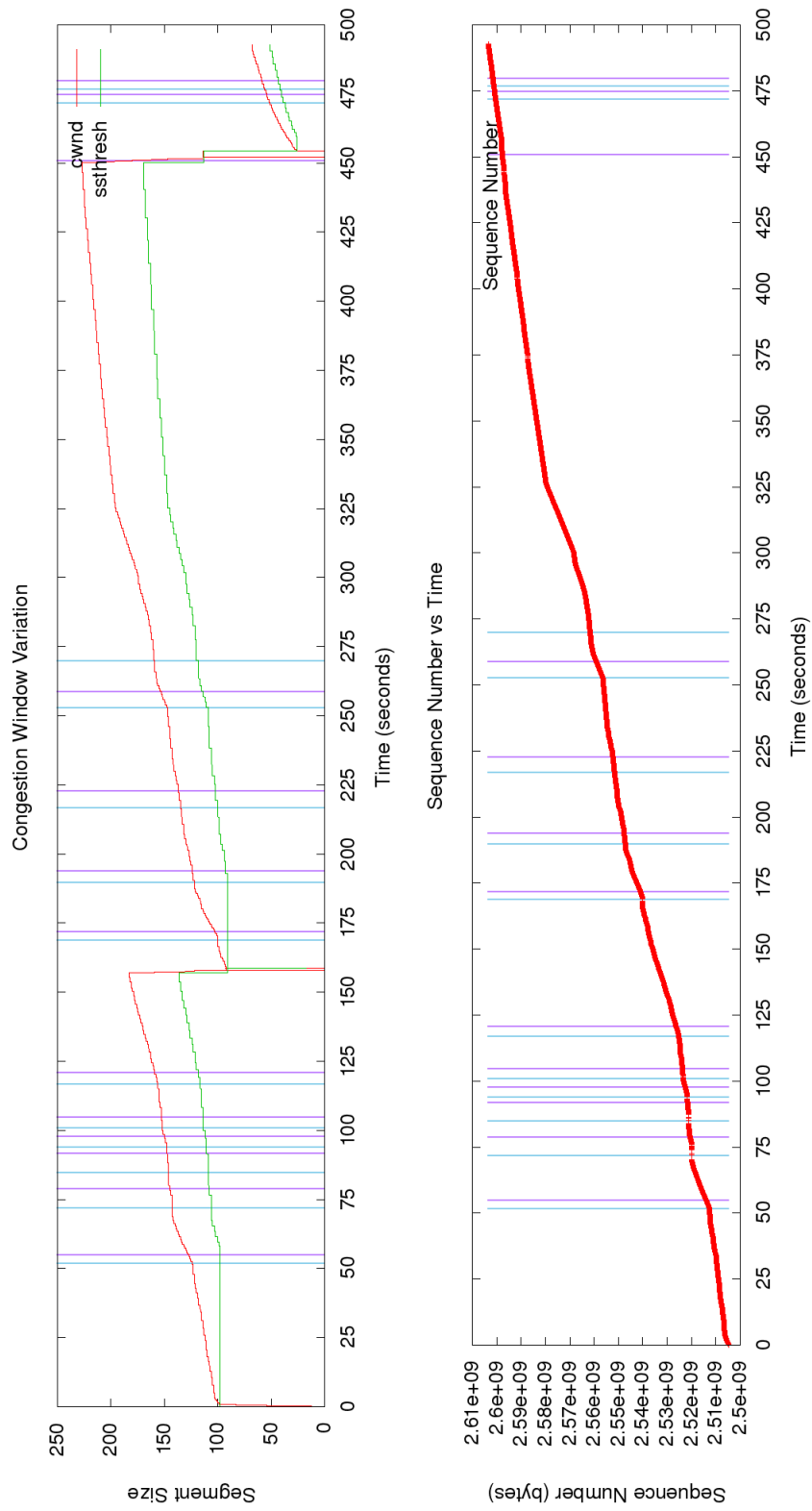


Figure 13: Westwood 2, Phase 2

packet loss and packet loss due to congestion and appropriately manipulate the congestion window and slow start threshold to achieve near maximum throughput of the connection.

7 Acknowledgement

We have learnt some very interesting aspects about TCP from this experiment. We would like to thank our course supervisor Dr. Vinay Rebeiro for suggesting such an interesting experiment. We would also like to thank our colleagues Harsh Singhal, Aakash Gupta and Shanu Agarwal for their suggestions and knowledge about the subject of wireless networks. Finally, we would like to thank our supervisor, Dr Rebeiro and Dr. Aaditeshwar Seth for providing us with a linux machine for our study.

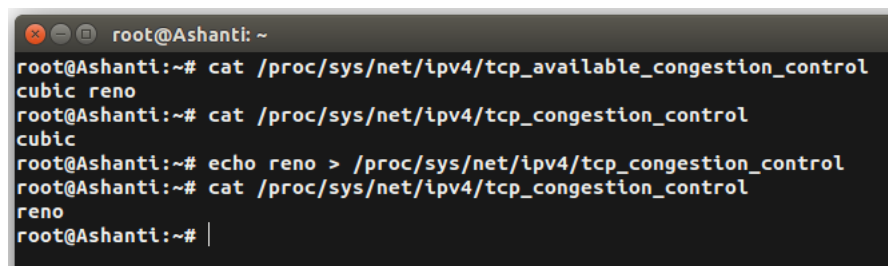
8 Appendix

Appendix A: Configuring TCP in linux We can switch between different TCP variants (congestion control algorithms) using the following command:

```
$ echo X > /proc/sys/net/ipv4/tcpcongestioncontrol
```

where X is the congestion control algorithm listed for the linux kernel.

NOTE: X must be in the list of available congestion control algorithms. The above command must be executed with root privileges.



```
root@Ashanti: ~  
root@Ashanti:~# cat /proc/sys/net/ipv4/tcp_available_congestion_control  
cubic reno  
root@Ashanti:~# cat /proc/sys/net/ipv4/tcp_congestion_control  
cubic  
root@Ashanti:~# echo reno > /proc/sys/net/ipv4/tcp_congestion_control  
root@Ashanti:~# cat /proc/sys/net/ipv4/tcp_congestion_control  
reno  
root@Ashanti:~# |
```

Above screenshot shows:

- Available congestion control algorithms in Ubuntu 14.04
- Default congestion control algorithm in Ubuntu 14.04
- New congestion control algorithm after the command is executed

9 References

1. TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks Cheng Peng Fu, Associate Member, IEEE, and Soung C. Liew, Senior Member, IEEE
<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1177186>
2. TCP Vegas: New Techniques for Congestion Detection and Avoidance Lawrence S. Brakmo Sean W. OMalley Larry L. Peterson
<http://pages.cs.wisc.edu/~akella/CS740/S08/740-Papers/BOP94.pdf>
3. TCP Cubic, <http://intronetworks.cs.luc.edu/current/html/newtcps.html>
4. TCP Westwood, Casetti, Gerla, Mascolo, TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks