University of Southern Queensland

Faculty of Engineering & Surveying

Investigation of TCP Performance Over Wireless Internet.

A dissertation submitted by

Peh Wee Liang

in fulfilment of the requirements of

ENG 4112 Research Project

towards the degree of

Bachelor of Electrical and Electronic Engineering

Submitted: October, 2004

Abstract

The demand for providing Internet services over wireless links has grown rapidly in recent years. Although TCP (Transmission Control Protocol) has been performing well over the traditional wired networks where packet losses occur mostly because of congestion, it cannot react efficiently in wireless networks, which suffer from significant non-congestion-related losses due to reasons such as bit errors and handoffs.

The main reason for this poor performance of TCP is the fact that it cannot distinguish between packet losses due to wireless errors from those due to congestion. It responds to all losses by invoking congestion control and avoidance algorithms. Moreover, TCP sender cannot keep the size of its congestion window at optimum level and always has to retransmit packets after waiting for timeout, which significantly degrades end-to-end delay performance of TCP.

This issue has attracted significant research interests and many schemes have been proposed to address the issue. This project will investigate the performance of a few representative mechanisms, which will improve both throughput and delay performance of TCP in wireless environment significantly. That is, Snoop protocol, Explicit Loss Notification (ELN), Explicit Congestion Notification (ECN).

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analysis and conclusion set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Peh Wee Liang

D11338609

____

Signature

____

Date

Acknowledgments

I would like to express my gratitude toward Dr. Hong Zhou and Dr. John Leis for their constant information, suggestions and guidance. I am grateful for the opportunity of working under them on such an interesting Undergraduate Research project.

My thanks is also extended to friends and classmates, who had advise and seen me through the difficulties I had encountered throughout the course of the project.

Most importantly I would like to thank my family whose constant support, kind patience and understanding I could not do without.

Peh Wee Liang

University of Southern Queensland

October 2004

Contents

List of Figures

Chapter 1

Introduction

1.1 Project Background

The term wireless refers to telecommunication technology in which radio waves such as infrared waves and microwaves are used to carry a signal to connect communication devices, instead of cables or wires. These devices include pagers, cell phones, portable PCs, computer networks, location devices, satellite systems and handheld digital assistants. Wireless technology is rapidly evolving, and is fast becoming an important role in the lives of people around the world. Wireless technology enables users to physically move while using an appliance, such as a handheld PC, paging device, or phone. Without the physical connection of cables or wires, this technology allows users to check stocks and email from their internet-enabled devices.

Wireless networking also arises with the ever-increasing need for businesses to lower costs and support mobility of workers. Compared with wired networking, wireless capability offers more timeliness, affordability, and efficiency. When performing installations, there are many tangible cost savings with using less wire between the user's appliance and a server. If mountains, highways or other buildings obstruct the way of a connection, a wireless solution may be more economical than installing physical cable. Wires and connectors can easily break through misuse and normal wear and tear. Therefore using less cable reduces the downtime of the network and the costs associated with replacing cables, and makes the network available for use much sooner. http://wireless.ittoolbox.com/pub/wireless_overview.htm (2002)

Pilosof *et al*. (2002) noted that the result of the growth in usage of wireless networking, has caused the focus to turn to deploying wireless Internet over hot spots such as airports, hotels, cafes, and other areas from which people can have uninterrupted public access to the Internet. As these networks see increasing public deployment, it is important for the service providers to be able to ensure that access to the network by different users and applications remains impartial. Since the majority of today's applications in Internet use Transmission Control Protocol (TCP), this project will focus on the performance of TCP in wireless Internet.

1.2 Project Aims

The demand for providing Internet services over wireless links has grown rapidly in recent years. Although TCP has been performed well over wired networks, it cannot respond efficiently in wireless networks. As a result, this issue has attracted significant research interests and, many modifications and new solutions have been proposed to improve TCP's performance.

1.3 Specific Objectives

This research will investigate the performance of a few representative mechanisms, that is, Snoop protocol, Explicit Loss Notification (ELN), Explicit Congestion Notification (ECN). This project also aims to:

- Study protocols and representative mechanisms: TCP/IP, Snoop protocol, Explicit Loss Notification (ELN) and Explicit Congestion Notification (ECN).
- Compare the performance with regular TCP with the representative mechanisms.
- Identify the advantages and disadvantages of the representative mechanisms.
- Study simulation tool, Network Simulator (NS2) and Linux.

Chapter 2

Transmission Control Protocol (TCP)

2.1 Introduction

Transmission Control Protocol (TCP) is the most widely used transport layer protocol in the Internet. Most popular Internet applications, such as the Web and file transfer, use the reliable services provided by TCP. Hassan and Jain (2001) noted that the performance perceived by users of these Internet applications depends largely on the performance of TCP.

In the Internet protocol suite, Internet Protocol (IP) is a best-effort service and TCP is a reliable service. IP provides the basic packet forwarding while TCP implements the flow controls, acknowledgements and retransmissions of lost or corrupted packets. This split in services "decentralizes" the network and moves the responsibility for reliable delivery to end systems. TCP is an end-to-end transport protocol, meaning that it runs in end systems, not the network. IP is a network protocol.
http://www.linktionary.com/f/flow_control.html (2001)

TCP is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment is transmitted with that segment and is called the segment sequence number. Segments also carry an acknowledgment number, which is the sequence number of the next expected data octet of transmissions in the reverse direction. When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted. An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken the responsibility to do so. To govern the flow of data between TCP, a flow control mechanism is employed. http://www.ietf.org/rfc/rfc0793.txt (1981)

As noted by Biswas (2003), TCP is almost globally accepted as the standard for end-to-end reliable communication protocol. Therefore it is not feasible at any time to make changes to the core of TCP protocol, and expect the globe community to move over to the new version. Hence, efforts are being made to work around the shortcoming of TCP by hiding the underlying inconsistencies of the wireless link from the protocol. Some of the solutions that have been proposed are the Snoop Protocol, ELN and ECN, which increase TCP's performance by hiding the packet losses over the wireless link.

Therefore in this project, the flow control, and congestion control mechanisms in TCP will be looked at in detail. The reasons for TCP's poor performance in wireless Internet will also be discussed.

2.2 Flow Control

Flow-control mechanisms control packet flow so that a sender does not transmit more packets than a receiver can process. Flow controls are necessary because senders and receivers are often unmatched in capacity and processing power. A receiver might not be able to process packets at the same speed as the sender. If buffers fill, packets are dropped. The goal of flow-control mechanisms is to prevent the dropping of packets that must be retransmitted.

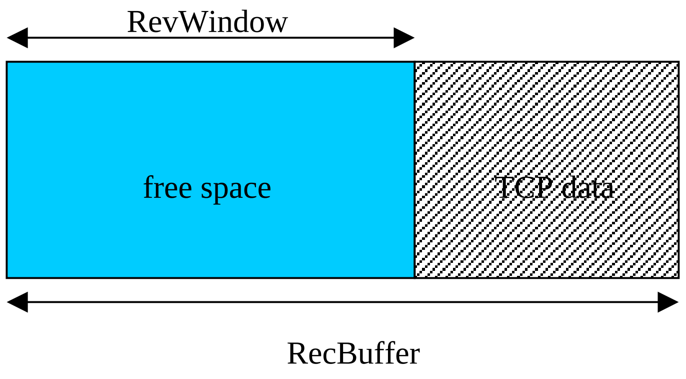


Figure 2.1: Receiver Buffer

Once the TCP connection is established, each host will advertise its receiver window size called the ’RevWindow’ as shown in Figure 2.1. It is the amount of receiving buffer available on each host. It is also the maximum amount of data the sender can send to the receiver at a time. The receiver advertises the size of its receiver window with each acknowledgement it sends to the sender. This causes the sender to not send excess data, and hence the receiver buffer never overflows. When the receiver’s buffer is full, a window size of zero is advertised. The sender will stop sending data to the receiver without invoking any congestion control mechanism.

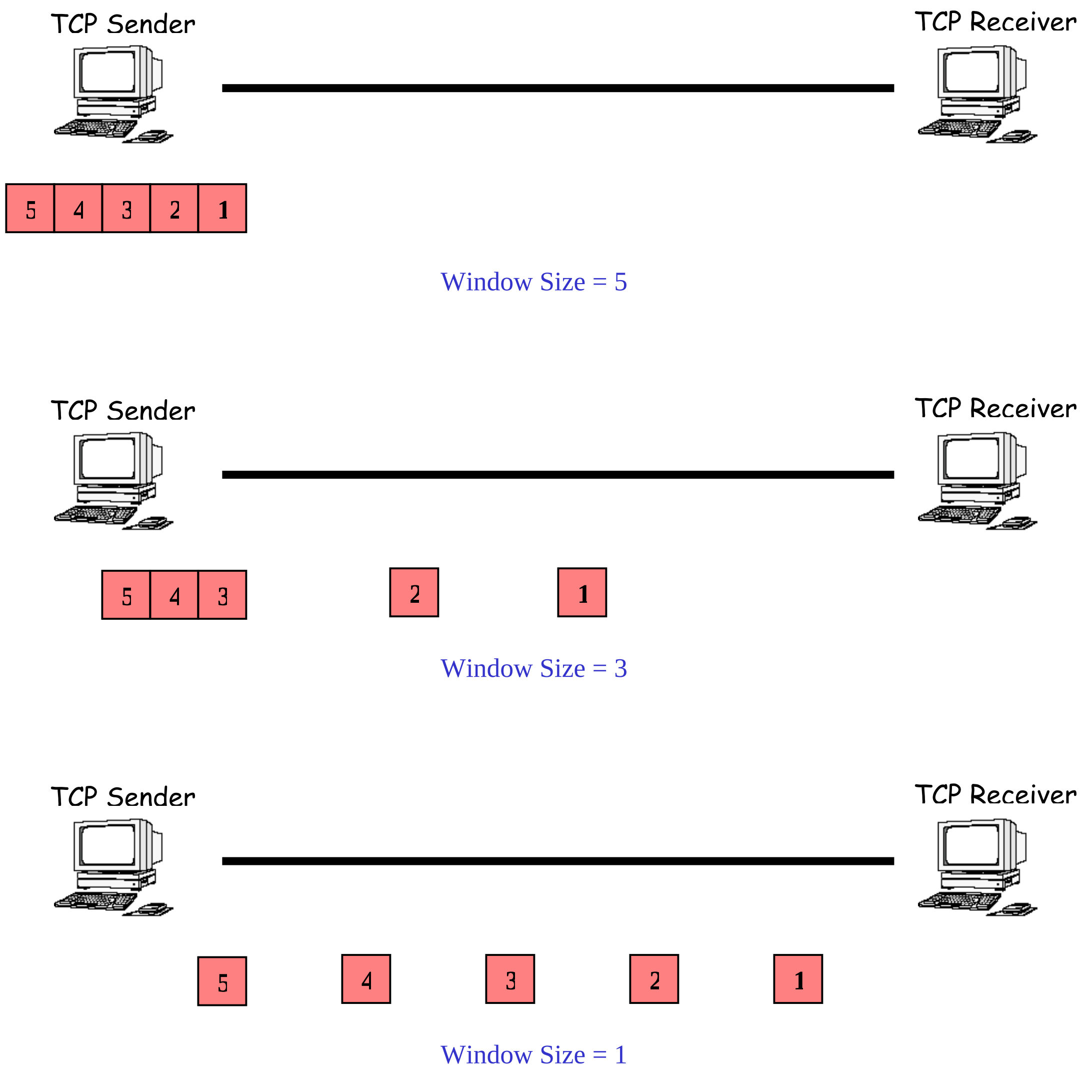


Figure 2.2: Flow Control part 1

With reference to Figure 2.2, the initial receiver’s advertise window size is five. Thus the sender knows that it can only send up to a maximum of five packets at a time. As the packets are transmitted over the link, window size is reduced accordingly.

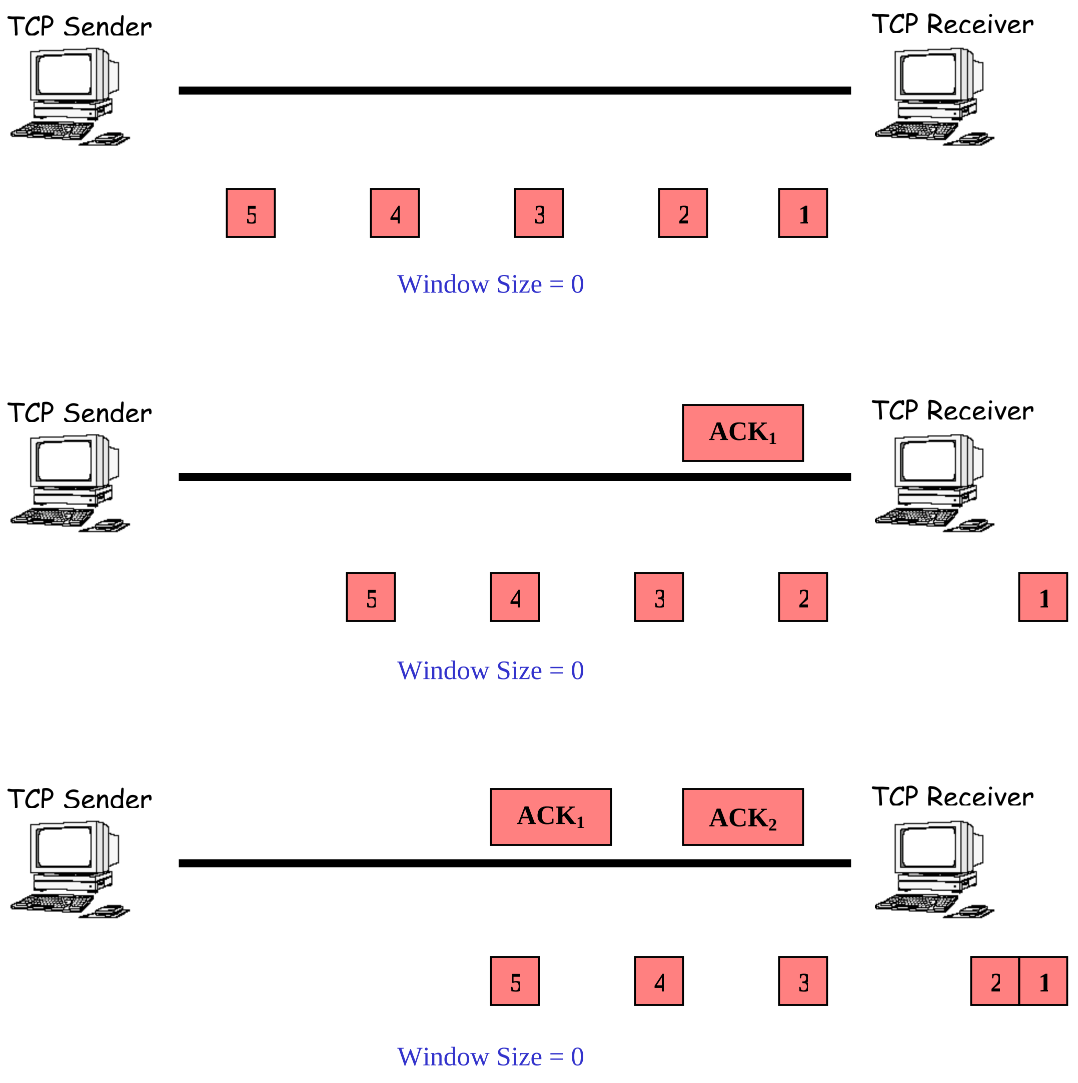


Figure 2.3: Flow Control part 2

As shown in Figure 2.3, after five packets are being transmitted by the sender, the advertise window size will be zero, thus telling the sender to stop sending any data. The receiver will indicate its current window size to the sender by sending an acknowledgement (ACK) for every packet it received. In ACK_1, the receiver has advertised a window size of one. The reason why the sender is still not transmitting any data at the moment is because ACK_1 has not reach the sender. Therefore the sender can only see a window size of zero and it will not transmit any data.

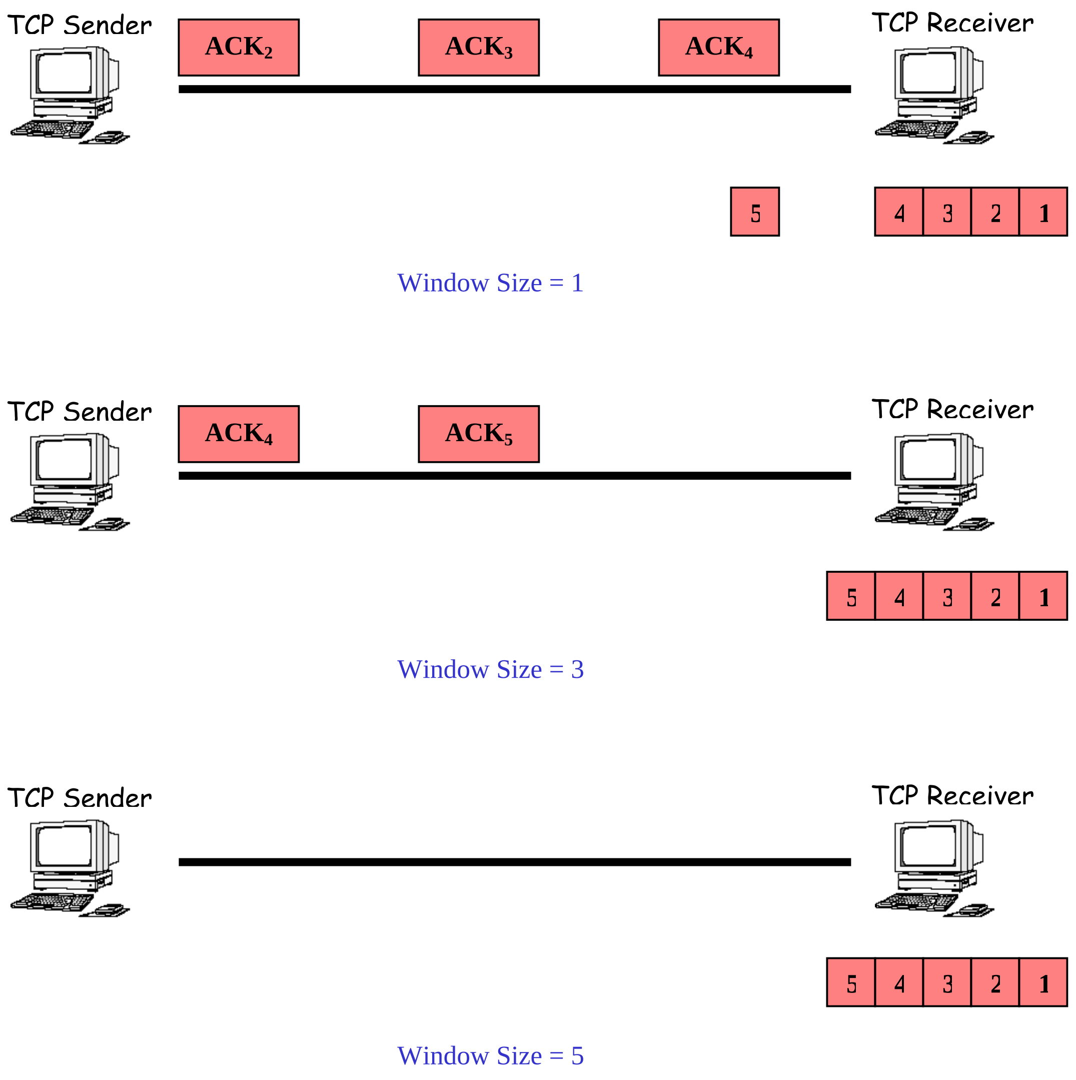


Figure 2.4: Flow Control part 3

In Figure 2.4, when ACK_1 reaches the sender, the window size is increased to one. This will notified the sender that it could now transmit one packet of data across the link. As more ACKs are transmitted over the link, window size will increase accordingly. This will go on until all the ACK have reached the sender, thus telling the sender that it can transmit up to a maximum of five packets again.

2.3 Congestion Control

Congestion control mechanisms allow network systems to detect network congestion (a condition in which there is more traffic on the network than can be handled by the network or network devices) and throttle back their transmission to alleviate the congestion. Congestion occurs on busy networks. When it occurs, end systems and the network must work together to minimize the congestion. In contrast, flow controls are used between end systems. A receiver uses flow controls to signal to the sender that it is overloaded. The sender then throttles back or stops its transmission.
http://www.linktionary.com/f/flow_control.html (2001)

Without congestion control, the receiver may indicate a large window, which encourages transmissions. However if more data packets arrive than can be accepted, it will be discarded. This will result in excessive retransmissions, adding unnecessarily to the load on the network and the TCP. Indicating a small window may restrict the transmission of data to the point of under-utilizing the available bandwidth on the link.
http://www.ietf.org/rfc/rfc0793.txt (1981)

2.3.1 Slow Start

Slow Start mechanism is a feature in TCP, used by the sender to control the transmission rate. This is accomplished through the return rate of acknowledgements from the receiver. The rate at which the sender can transmit data is determined by the rate of ACK transmitted by the receiver.

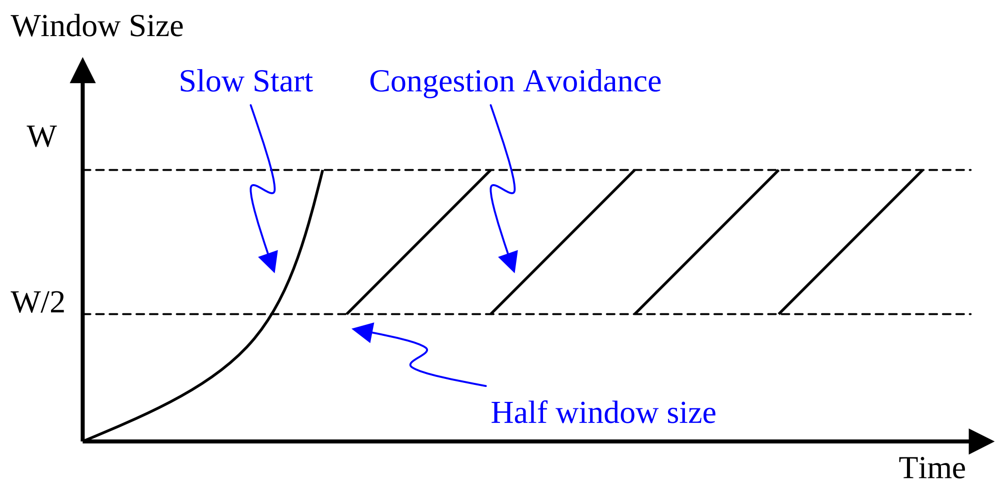


Figure 2.5: TCP transmission window

When a TCP connection is established, the Slow Start algorithm initializes a congestion window to one segment. Kristoff (2003) found out that when the receiver returns ACKs, it would cause the congestion window to increase by one segment for each ACK returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the transmission window.

When the network is not congested and network response time is good, Slow Start algorithm will increase the window exponentially to determine the available bandwidth on the link as shown in figure 2.5. For the first successful transmission and acknowledgement of a TCP segment, the algorithm will increased the window to two segments. After successful transmission of these two segments and acknowledgements completes, the window is increased to four segments. Then eight segments, then sixteen segments and so on, up to the maximum window size advertised by the receiver or until congestion finally does occur.

2.3.2 Congestion Avoidance

Congestion Avoidance is used to slow the transmission rate during Slow Start if the network is forced to drop one or more packets due to overload or congestion. Congestion Avoidance is used with Slow Start to keep the data transfer un-interrupted, so it doesn't slow down and stay slow.

A retransmission timer expiring or the reception of duplicate ACKs in the Congestion Avoidance algorithm can implicitly signal the sender that there is a network congestion situation. This would cause the sender to set its transmission window to one half of the current window size, but to at least two segments as shown in Figure 2.5. However 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 situation was indicated by duplicate ACKs, the Fast Retransmit algorithm to be discussed in the next section will be invoked.

Kristoff (2003) found out that 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. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

2.3.3 Fast Retransmit

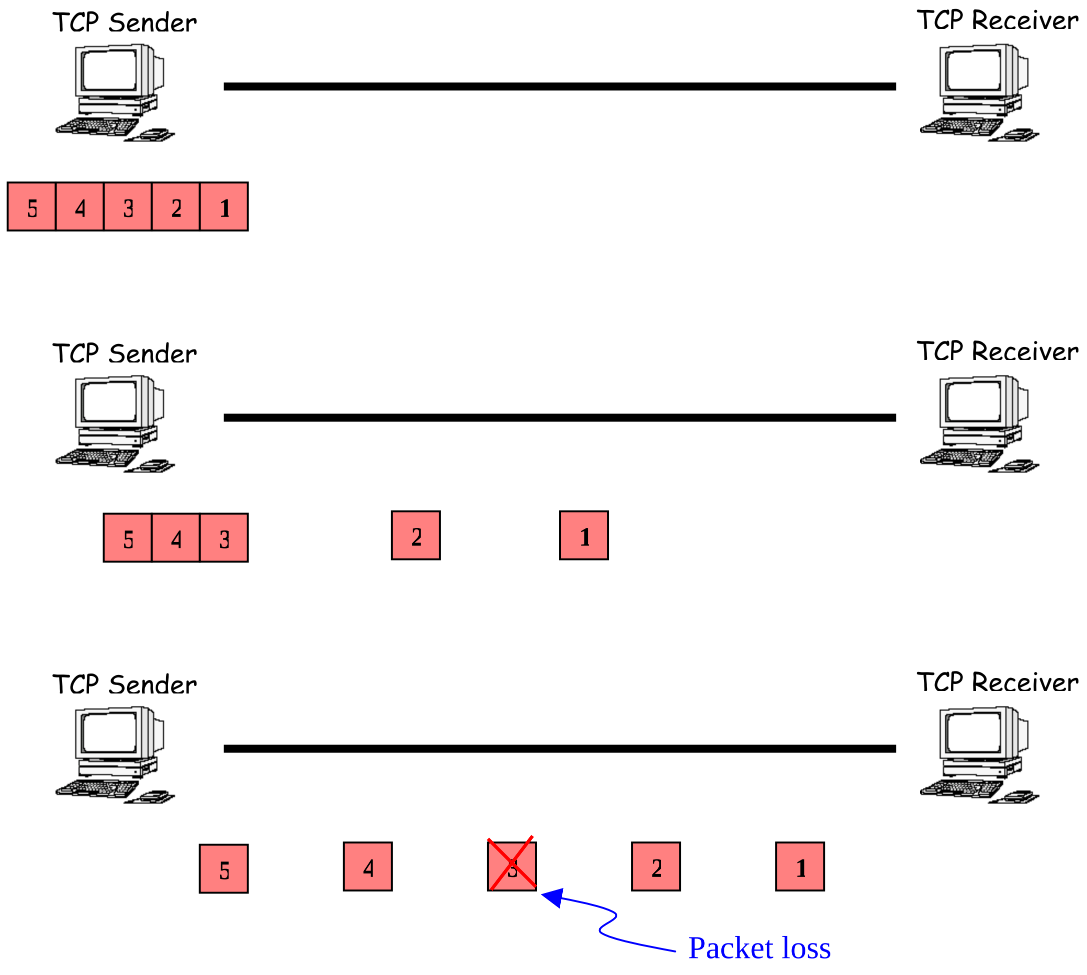


Figure 2.6: Fast Retransmit part 1

As shown in Figure 2.6, as the data packets are transmitted over the link, packet three is lost in the network. TCP sender will use the cumulative acknowledgements it receives to determine which packet have reached the receiver, and provides reliability by re-transmitting lost packets, which will be discuss further below.

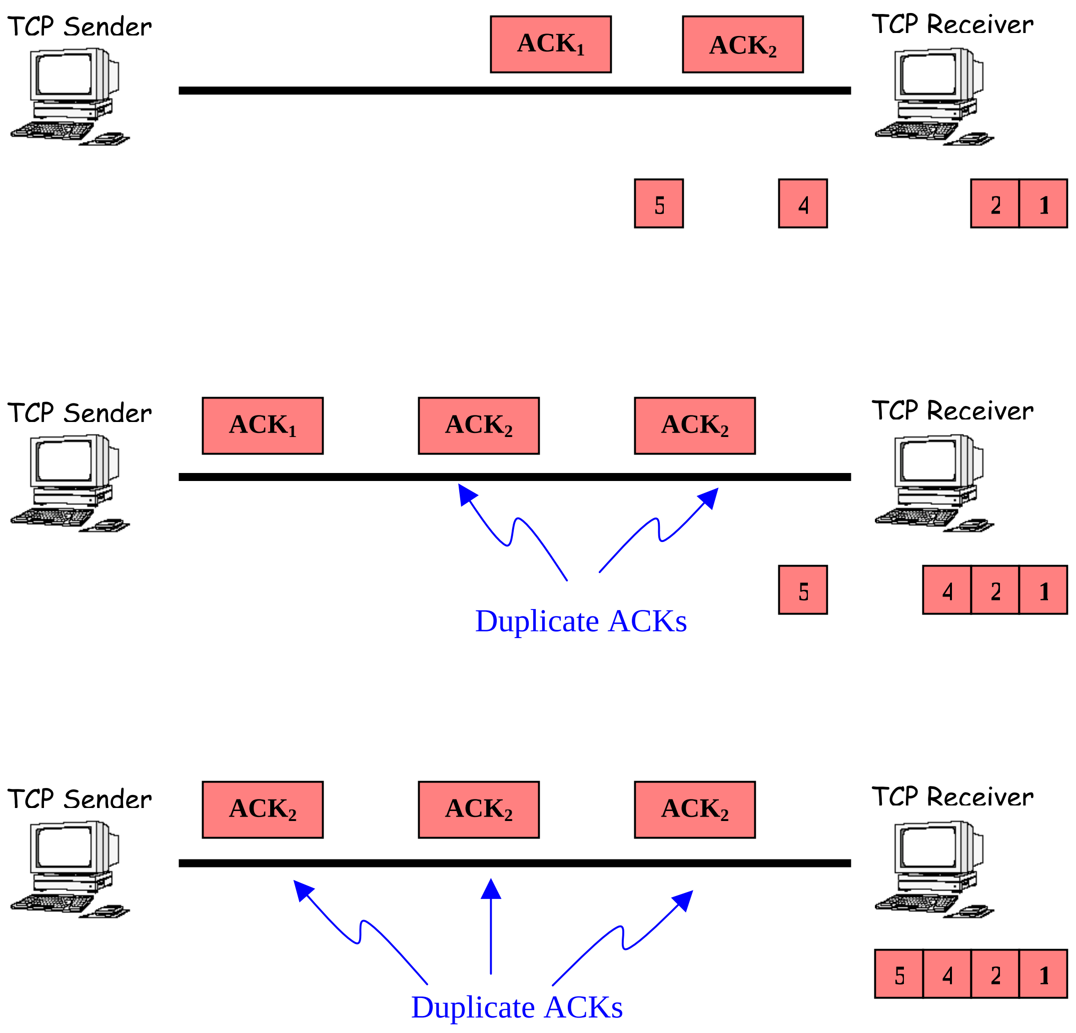


Figure 2.7: Fast Retransmit part 2

With reference to Figure 2.7, Biswas (2003) noted that the indication of packet loss is a Duplicate ACK. Whenever a packet arrives out of sequence; only the ACK for last packet received in sequence is sent back to the sender. Hence, when a Duplicate ACK arrives, TCP sender identifies the cause to be either a packet loss or a delayed packet receipt. If a third DUPACK is received, TCP confirms packet loss and performs a fast retransmit.

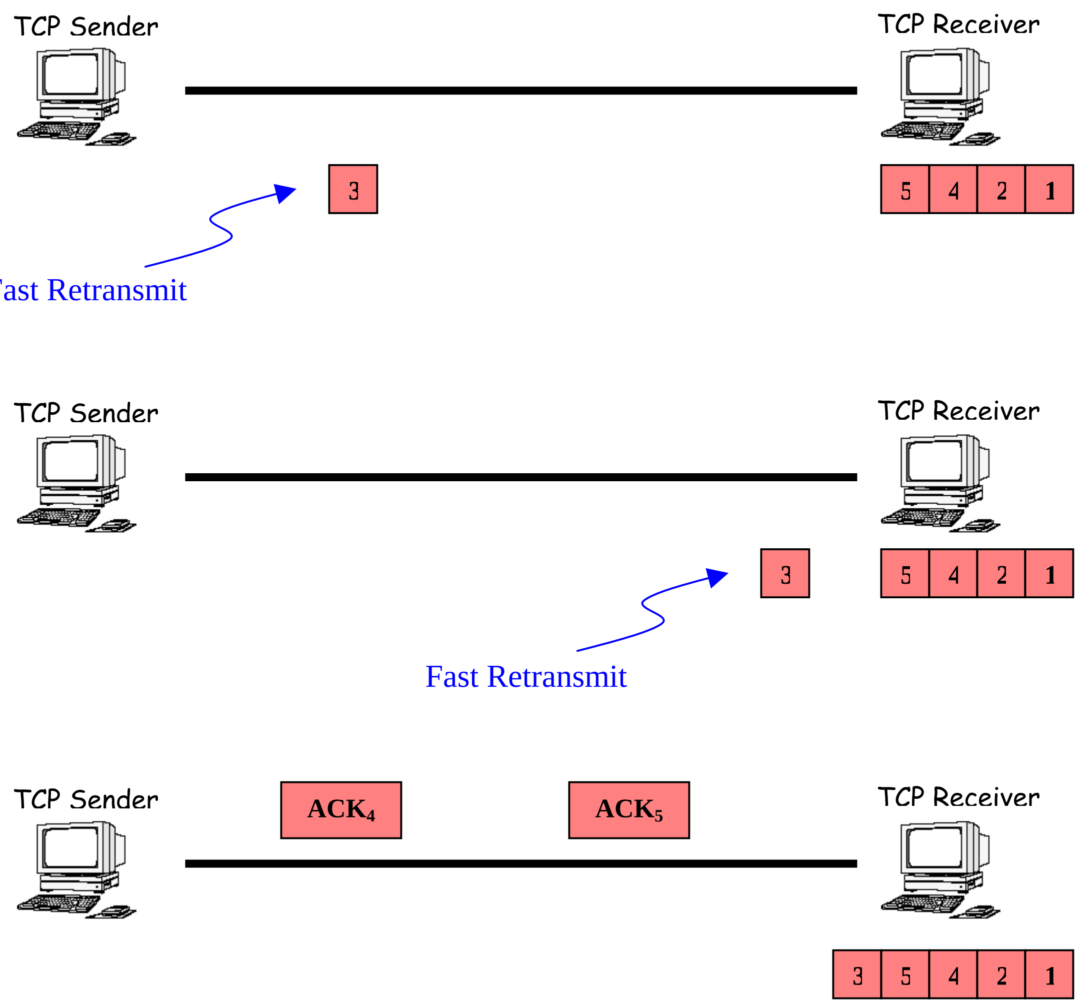


Figure 2.8: Fast Retransmit part 3

Immediately following Fast Retransmit, the sender will be in Congestion Avoidance mode with reference to Figure 2.8. Thus causing the sending window size to be reduced by half. The sender will then increases its window size by one unit every average round trip time.

2.4 TCP in Wireless Link

Traditionally, TCP has been tuned for networks comprising wired links and stationary hosts. It assumes congestion in the network to be the primary cause for packet losses and unusual delays, and adapts to it. Balakrishnan and Katz (1998) noted that TCP reacts to packet losses by re-transmitting missing data, and simultaneously invoking congestion control by reducing its transmission (congestion) window size and reducing its retransmission timer. These measures will lower the level of congestion on the intermediate links.

However Sharma and Hu (2002) argued that although TCP has been greatly enhanced in its capability to adapt to high-speed links, many versions of TCP over wireless links still couldn’t keep the comparative throughput as TCP in wired networks. The main disadvantage is that traditional TCP assumes that all packet losses are due to network congestion. It is important that this assumption needs significant modification in wireless Internet applications because most packet losses are due to wireless link errors.

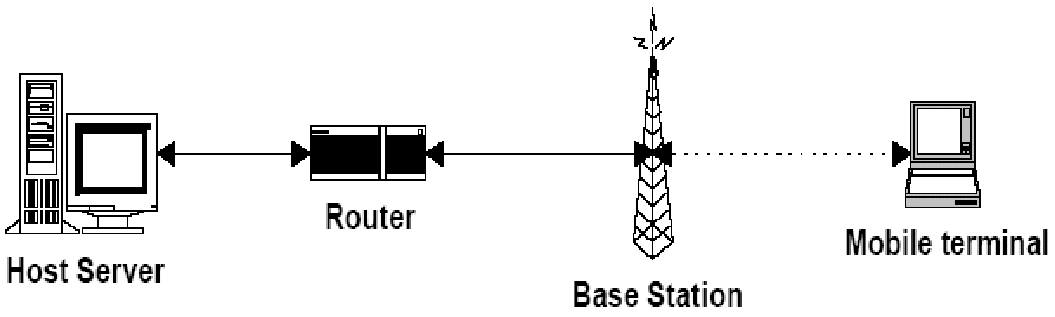


Figure 2.9: Wireless link using TCP.

Balakrishnan and Katz (1998) also found out that if packets are lost for reasons other than congestion, these measures will result in an unnecessary reduction in end-to-end throughput and hence, in sub-optimal performance. As shown in Figure 3, communication over wireless links is often characterized by high bit-error rates due to channel fading, noise or interference, and intermittent connectivity due to handoffs. TCP performance in such networks suffers from significant throughput degradation and very high interactive delays because the sender misinterprets corruption in the wireless links for congestion.

Chapter 3

Snoop Protocol

3.1 Introduction

The Snoop Protocol was designed to solve the burst/intermittent packet loss due to high bit error rates and short temporary disconnections experience by TCP in wireless link.

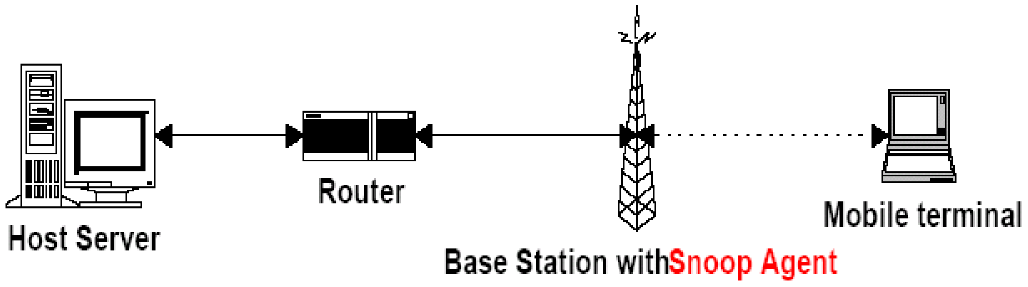


Figure 3.1: Adding the Snoop agent.

Snoop performs local retransmission and recovery. Biswas (2003) found out that it shields the sender from the inconsistency of the wireless link, without sacrificing the end-to-end semantics, or requiring any changes to the existing implementations of TCP.

It does not change or interfere with the content of the TCP packets that flow between the Fixed Hosts (FH) and Mobile Hosts (MH). In wireless link, we expect to have administrative control over the last hop router, or base station (BS). The Snoop agent is designed to reside on the router between the wired and wireless link, referred to as the gateway, or base station (BS) as shown in Figure 3.1. Snoop is TCP aware, and using its knowledge of the congestion control mechanism in TCP along with its capability of identifying packet losses.

Balakrishnan *et al* (1998) noted that the role of the snoop agent is to monitor the TCP packets transmitted from a fixed host to a mobile host and vice versa. The agent caches all those packets locally and in the case of receiving duplicate acknowledgments (ACKs), retransmits the packets promptly and suppresses duplicate ACKs. The Snoop protocol performs retransmission of lost packets locally (at the base station) and hence avoids lengthy fast retransmission and congestion control at the sender side. By this method, end-to-end semantics of TCP is maintained and performance of TCP is improved.

The snoop module maintains a cache of TCP packets sent from the FH that haven't yet been acknowledged by the MH. When a new packet arrives from the FH, the snoop module adds it to its cache and passes the packet on to the routing code, which performs the normal routing functions. The snoop module also keeps track of all the acknowledgments sent from the mobile host. When a packet loss is detected (either by the arrival of a duplicate acknowledgment or by a local timeout), it retransmits the lost packet to the MH if it has the packet cached. Thus, the base station (snoop) hides the packet loss from the FH by not propagating duplicate acknowledgments, thereby preventing unnecessary congestion control mechanism invocations.

3.2 Algorithms

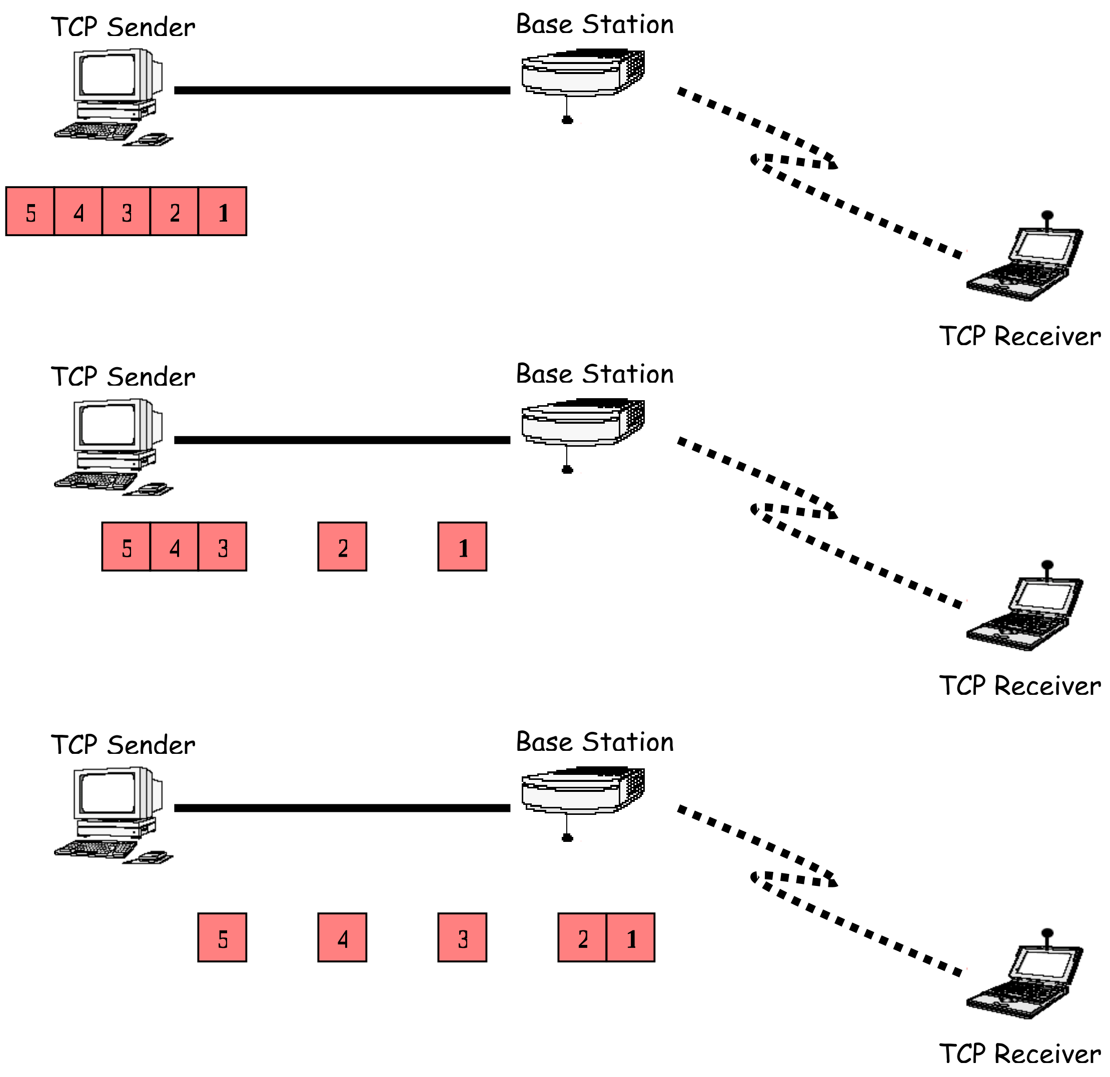


Figure 3.2: Snoop Protocol part 1

The Snoop Protocol introduces a module, called the snoop agent, at the Base Station. The agent monitors every packet that passes through the TCP connection in both directions. The agent maintains a cache of TCP segments sent across the link that have not yet been acknowledged by the receiver as shown in Figure 3.2.

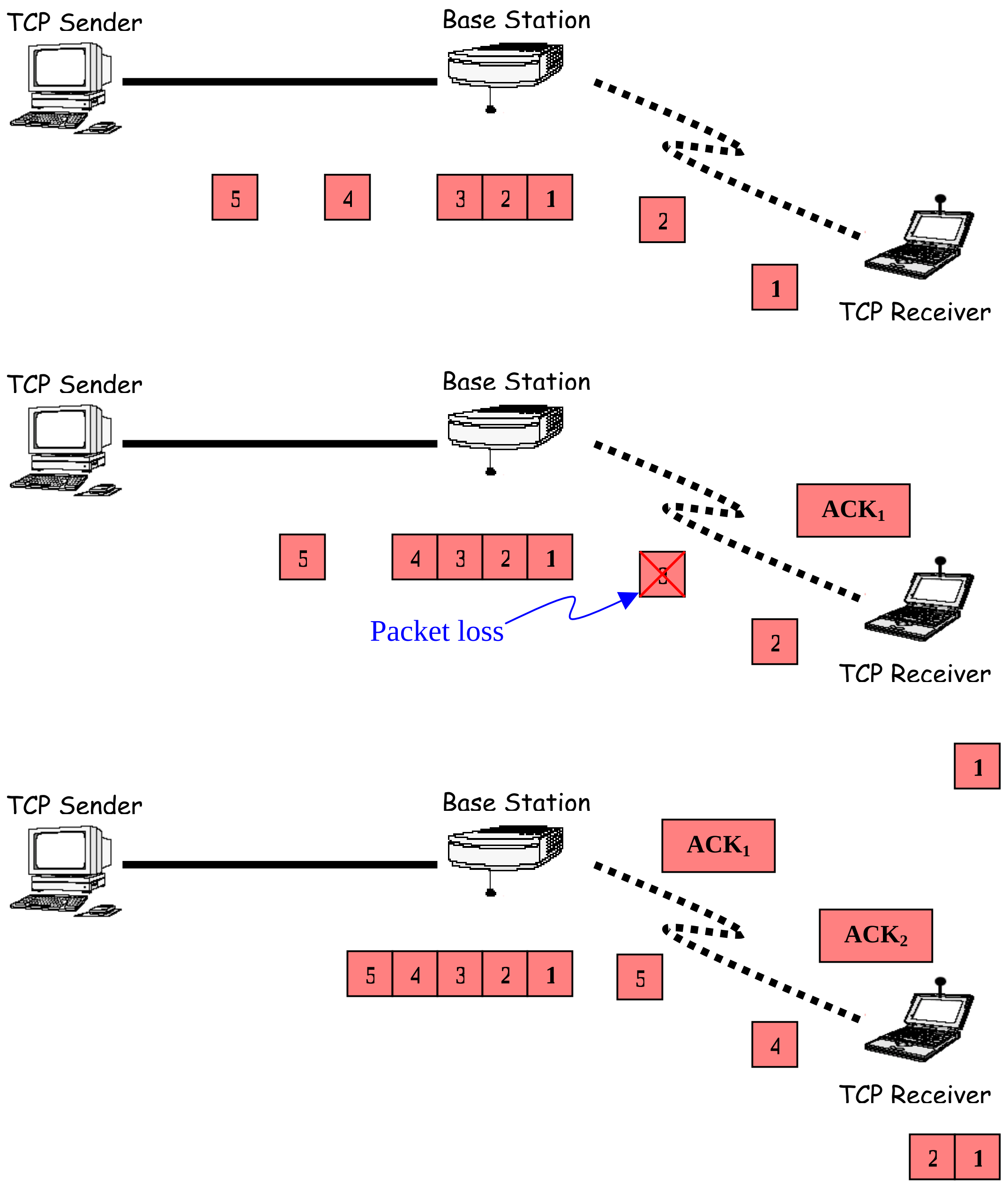


Figure 3.3: Snoop Protocol part 2

As the data packets are transmitted over the network, packet three is lost in the wireless link. TCP Receiver has already received packet one and two, therefore ACK_1 and ACK_2 are return to the sender as shown in Figure 3.3.

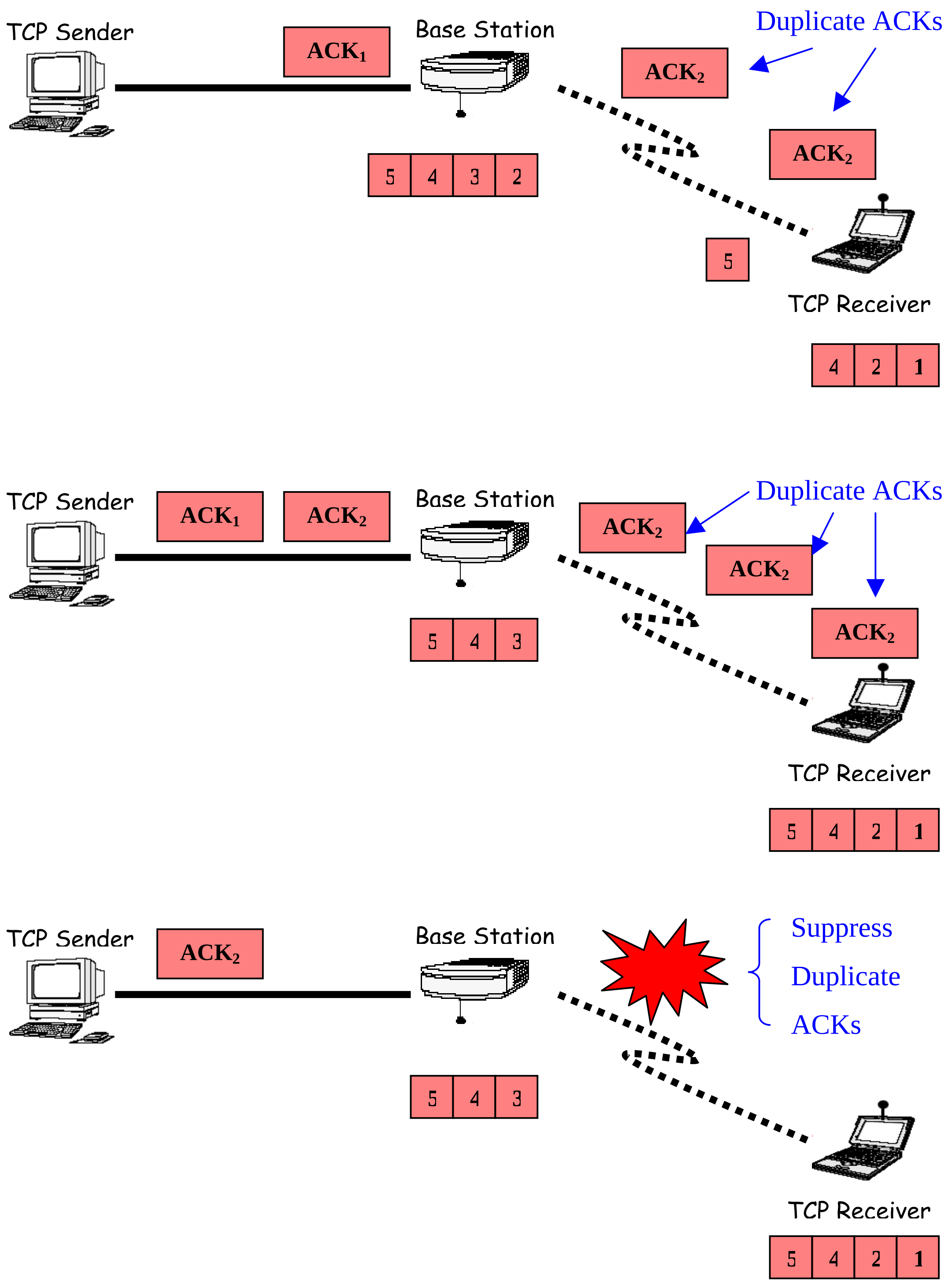


Figure 3.4: Snoop Protocol part 3

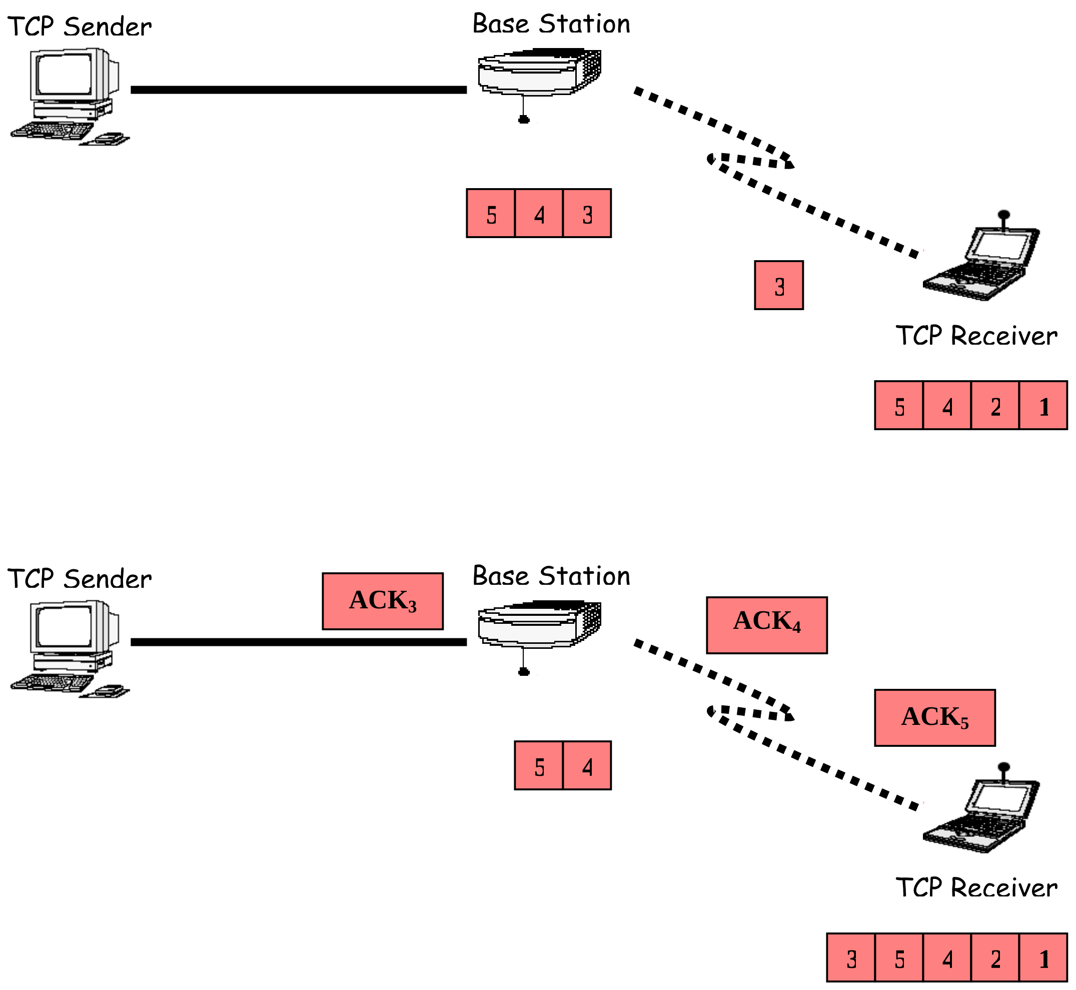


Figure 3.5: Snoop Protocol part 4

Figure 3.4 showed that a packet loss is detected by the arrival of a small number of duplicate acknowledgments (ACKs) from the receiver. The Snoop agent at the Base Station will then suppresses the duplicate acknowledgments to prevent the duplicate ACKs from reaching the sender.

In Figure 3.5, Snoop agent retransmits the lost packet from its cache and clear the packet that has been acknowledged from its cache. Since the Base Station retransmits the loss packet, the sender does not need to trigger its congestion control algorithms.

3.3 Performance Analysis

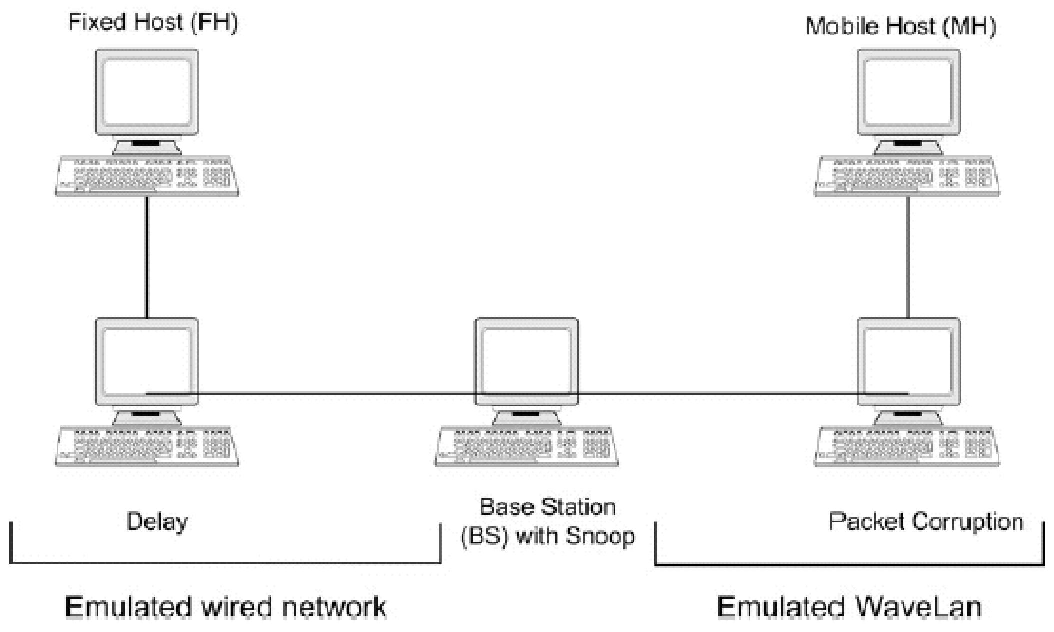


Figure 3.6: Snoop Protocol test bed setup 1

In figure 3.6, Biswas (2003) conducted an experiment to analysis the performance of Snoop module. Bandwidth between the Fixed Host (FH) and the Base Station (BS) was set at 10 Mbps, and the bandwidth available between MH and BS was set at 2 Mbps. There was no packet loss over the wired link, and a 2% packet loss on the wireless link. A constant delay of 200ms was applicable on the link, whose delay was not varied. Delay agent was used to add delay to the fixed network and the packets were corrupted using the packet corruption unit. Three tests were performed on the Snoop module to see its effectiveness in different scenarios. Two of them were with variable delays between the base station and the fixed/mobile hosts, and one of them was with variable packet loss over the wireless link.

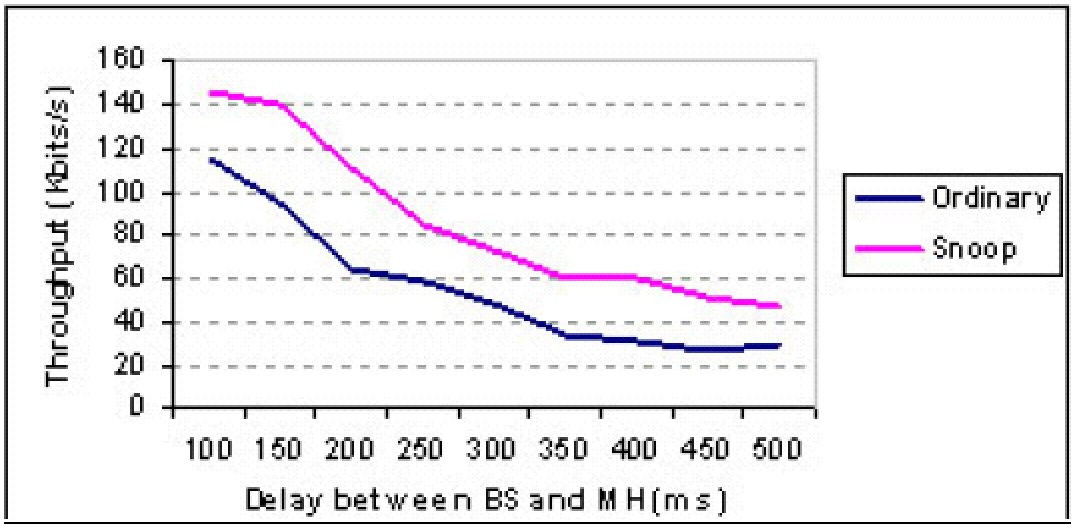


Figure 3.7: Variable delay between BS and MH

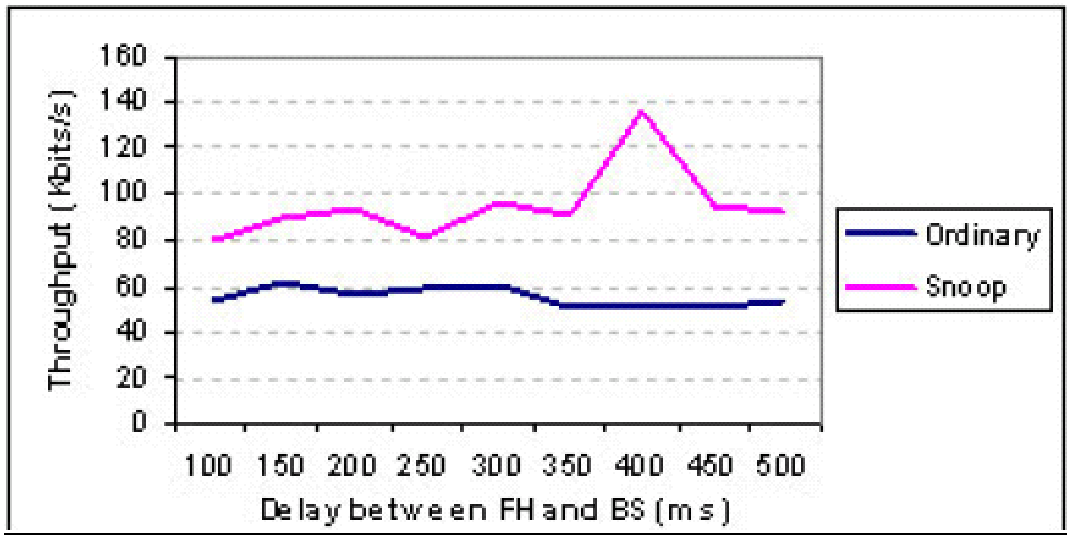


Figure 3.8: Variable delay between BS and FH

With reference to Figure 3.7, Biswas (2003) found out that although TCP performance deteriorates with increased delay over the wireless hop, Snoop still managed to obtain a better throughput than the normal TCP. The performance improvement is close to two times that of normal TCP.

Biswas (2003) also noted that with low delay on the wired link, the performance improvement with Snoop is not as high as shown in Figure 3.8. Significance performance improvement was noted when the delay was about 400ms. This was due to the increased delay on the wired link, thus the Snoop agent would had more time for itself to time out and perform local recovery when the wireless losses occurred. In this test scenario, the peak performance of Snoop was about three times that of normal TCP.

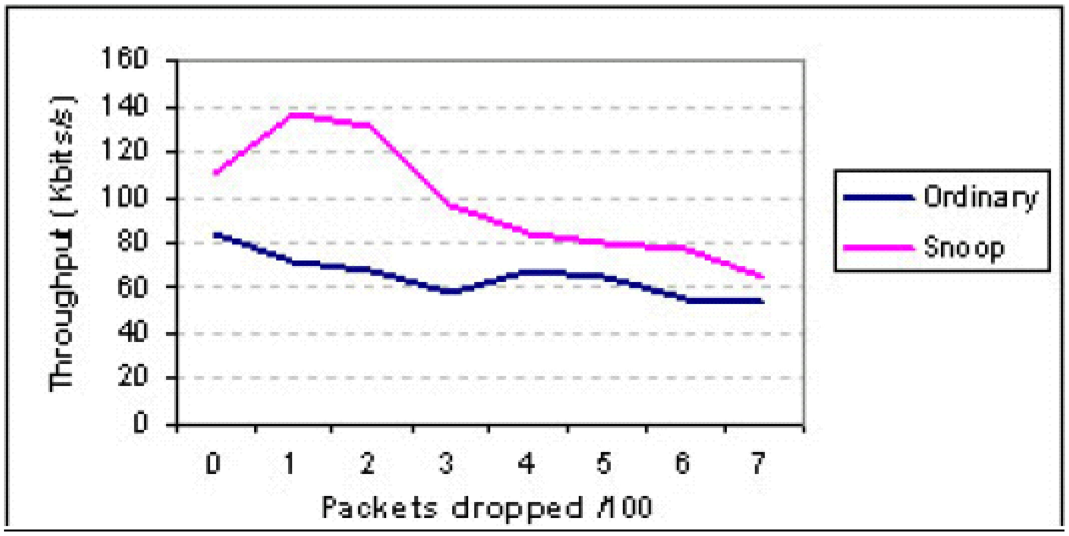


Figure 3.9: Snoop performance with different corruption rates

With reference to Figure 3.9, Biswas (2003) also found out that when packets were corrupted over the wireless link, Snoop still managed to give a consistent performance higher than normal TCP. Snoop produced a throughput of twice the normal TCP when there was 2% corruption on the link. However the quantity of performance improvement with snoop dropped with higher bit corruption rates, due to the delay on the links did not allow snoop with enough time to attempt multiple local recoveries.

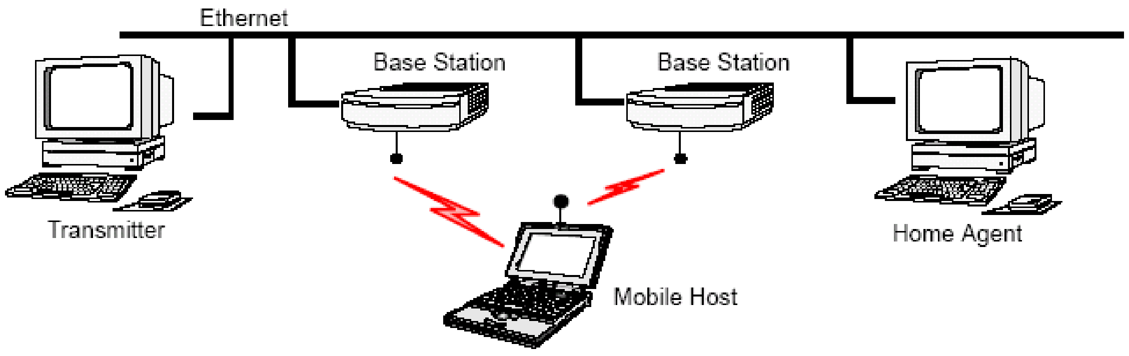


Figure 3.10: Snoop Protocol test bed setup 2

Showed in Figure 3.10 is a separate experiment done by Balakrishnan *et al* (1998). The simulation parameters are set as when there was no packet loss, the maximum throughput achieved by a TCP connection over the wireless link was about 1.6 Mbps. The rated maximum raw bandwidth of the wireless link was 2 Mbps. Data transfer from FH to MH was monitored. The maximum possible window size for the connection was 64 KBytes and the maximum TCP segment size was 1460 bytes.

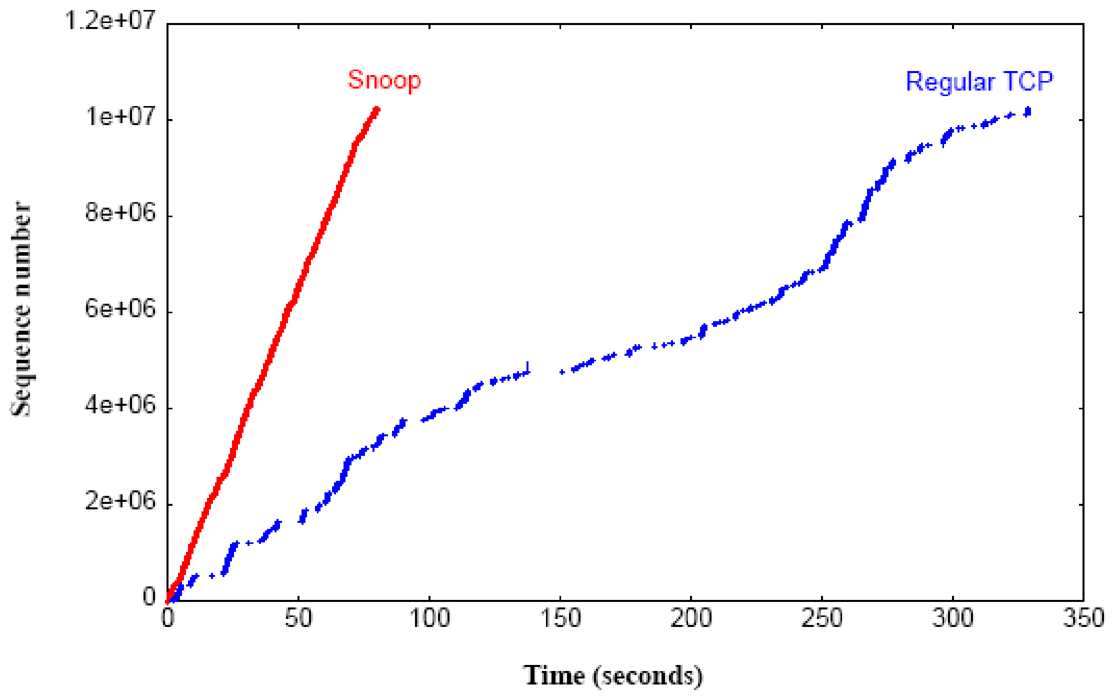


Figure 3.11: Throughput received by the mobile host at different bit-error rates

Figure 3.11 plots the sequence numbers of the received TCP packets versus time. It shows the comparison of sequence number progression in a connection using the Snoop protocol and a connection using normal TCP for a Poisson-distributed bit error rate of $3.9x10^{-6}$ (a bit error every 256 Kbits on average). Balakrishnan *et al* (1998) found out that the Snoop protocol managed to maintain a high and consistent throughput performance. On the other hand, normal TCP invoked congestion control procedures unnecessarily for several times during the duration of the connection. This event appeared as the flat and empty regions of the curve and deteriorates the throughput considerably.

3.4 Advantages and Disadvantages

The advantage as noted by Balakrishnan *et al* (1998) was that Snoop mechanisms improved the performance of the connection in both directions, without sacrificing any of the end-to-end semantics of TCP, modifying host TCP code in the fixed network or re-linking existing applications. The combination of improved performance preserved protocol semantics and full compatibility with existing applications. Snoop mechanism also had the advantage that the connection would not be idle for much time after a handoff since the new base station would forward cached packets as soon as the mobile host is ready to receive them. Another advantage was that it resulted in low-latency handoffs for non-TCP streams as well, especially continuous media streams. Snoop protocol was similar to link-level retransmissions over the wireless links in that both schemes perform retransmissions locally to the mobile host. It was closely coupled to TCP, and so did not perform many redundant re-transmissions. Packets retransmitted by the sender that arrived at a base station were already cached there. This happened most often because the sender often transmits half a window’s worth of data and several of these packets were already in the cache.

Balakrishnan and Katz (1998) also conducted experiments using TCP Reno, TCP SACK and the Snoop protocol using the Web workload, varying the number of concurrent TCP connections from 1 to 4, as well as using persistent-HTTP. Figure 3.12 shows the number of separate downloads using 1 to 4 concurrent connections per client as well as the performance of persistent-HTTP for the three protocol. It depicts the number of successfully completed individual downloads (connections) in 1000 seconds.

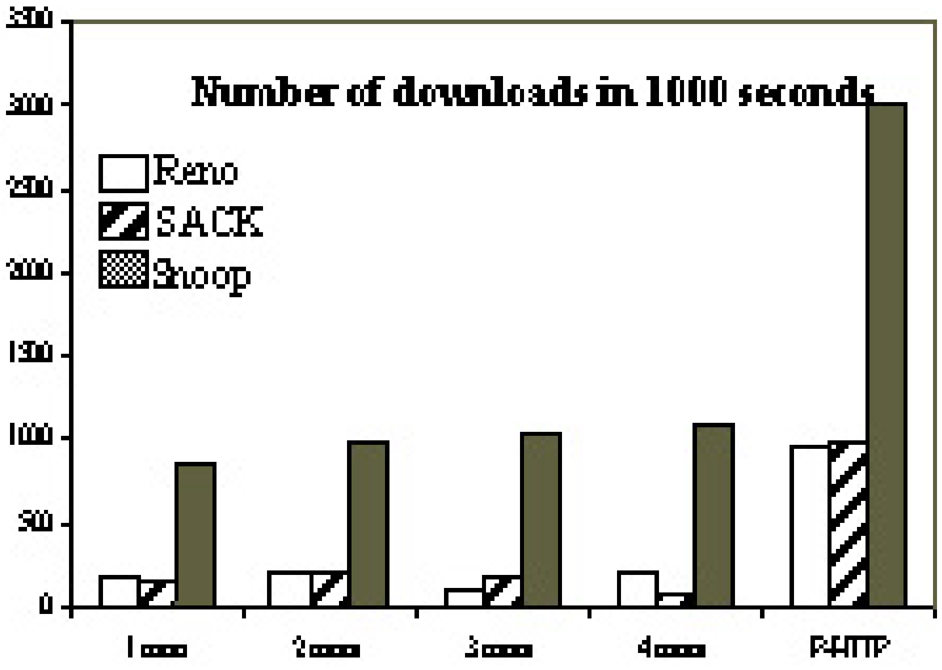


Figure 3.12: Performance on a Web workload in different protocol configurations

As shown in Figure 3.12, the advantage of Snoop protocol was the increased in performance of between three and six times than the other protocols. Balakrishnan and Katz (1998) found out that not only did the Snoop protocol performed well for large bulk transfers, but it also resulted in significant performance improvements for shorter transfers (combined with occasional long ones) that characterized Web workloads today. The performance improvement was between a factor of three and six for this realistic workload under experimentally measured and realistic wireless error conditions.

The disadvantage with Snoop protocol was that if the connection between the BS and the MH was unreliable, the FH might get timed out when waiting for the acknowledgement from the MH.
http://www.comp.leeds.ac.uk/sy22/web_pages/113/WirelessTCP.html#top

Biswas (2003) also noted that Snoop protocol is designed for handling connections where the bulk of the data is transferred from the FH to the MH. Therefore Snoop is only effective when the FH is the sender and the MH is the receiver.

West and Vaidya (1997) found out that one of the greatest disadvantages of Snoop protocol was that it requires the ACKs to follow the same path as the data in order to shield the sender from losses. This was not a problem for network topologies containing a single wireless path, which every packet must traverse. However it did became a problem when multiple wireless paths were possible, or with asymmetric links where the sender used a high bandwidth, high delay path (such as a satellite link) to send the data and the receiver used a low bandwidth terrestrial path to return the ACKs. Snoop had no method of informing the sender when the base station experiences a period of high errors and this could lead to unnecessary time out, which invoke congestion avoidance procedures.

Chapter 4

Explicit Loss Notification (ELN)

4.1 Introduction

Ding and Jamalipour (2001) found out that the poor performance of TCP in error-prone wireless networks is mainly due to lack of explicit information at the transport layer on the reason of a packet loss. For the wireless networks, if we can explicitly inform TCP the reason of a packet loss, then TCP will be able to maintain its throughput (i.e. not to reduce the congestion window size) if the packet has been lost not because of network congestion.

Ding and Jamalipour (2001) mentioned that the methods discussed in the previous section do not actually let the TCP sender determine clearly whether the packet is lost due to wireless error or network congestion. This makes the TCP sender retransmits the packet as usual (or quicker than usual) and then cannot keep the throughput high in the error prone environment.

Although Snoop protocol is a good method to improve the performance of TCP in wireless network on fixed host to mobile host direction, it retransmits the lost packet like other link layer solutions, now locally but through its snoop agent. Therefore Snoop protocol also suffers from not being able to completely shield the sender from the wireless losses.

Based on Snoop protocol, a new protocol called Explicit Loss Notification (ELN) with Acknowledgment (ELN-ACK), which can overcome the limitations of the Snoop protocol. As noted by Ding and Jamalipour (2001), in ELN-ACK protocol implementation, modifications are made to the structure of acknowledgment packet, and the software part at base station, mobile host and fixed host. Those modifications, however, can be maintained at minimal compared with other schemes. The method still looks at the throughput and delay performance improvement of TCP on the fixed host to mobile host direction.

Ding and Jamalipour (2001) also noted that, in ELN-ACK a new form of acknowledgment packet called ACK_{ELN} is used. The sequence numbers of the four most recently lost packets judged by the MH and one bit (called ELN bit) to indicate the reason of the lost packet, are included in the ACK_{ELN} acknowledgment packet. The ELN bit is judged at the BS. ELN agent at the BS checks the information stored in the ELN bit to see if the packet has been lost before it is arrived at the BS. After the ELN agent at BS processes the ACK_{ELN}, it continues to transmit back to FH. When the FH (the original sender) receives the ACK_{ELN}, the TCP sender will know the reason of packet loss from the ELN bit, explicitly.

Similar to the snoop agent used in the Snoop protocol, an ELN-ACK agent is introduced at the BS as noted by Ding and Jamalipour (2001). It has two main functions, the first one is to judge and store the packet loss information transmitted from the FH. The second function is to judge the value of ELN bit. When the BS receives an ACK_{ELN}, it will judge the reason for lost packet based on the stored information in the acknowledgment packet. Data processing procedure at the ELN-ACK agent is very similar to the one used in the Snoop protocol.

Ding and Jamalipour (2001) also found out that when the FH receives the ACK_{ELN}, it acts with the information stored in the ELN bit. If the ELN bit is ‘1,’ it means that the corresponding packet is lost due to wired segment congestion and thus it will proceed with the same procedure as in the window algorithm. If the ELN bit is ‘0,’ it means that the corresponding packet is lost due to wireless error and thus it retransmits the packet immediately without any window reduction.

4.2 Algorithms

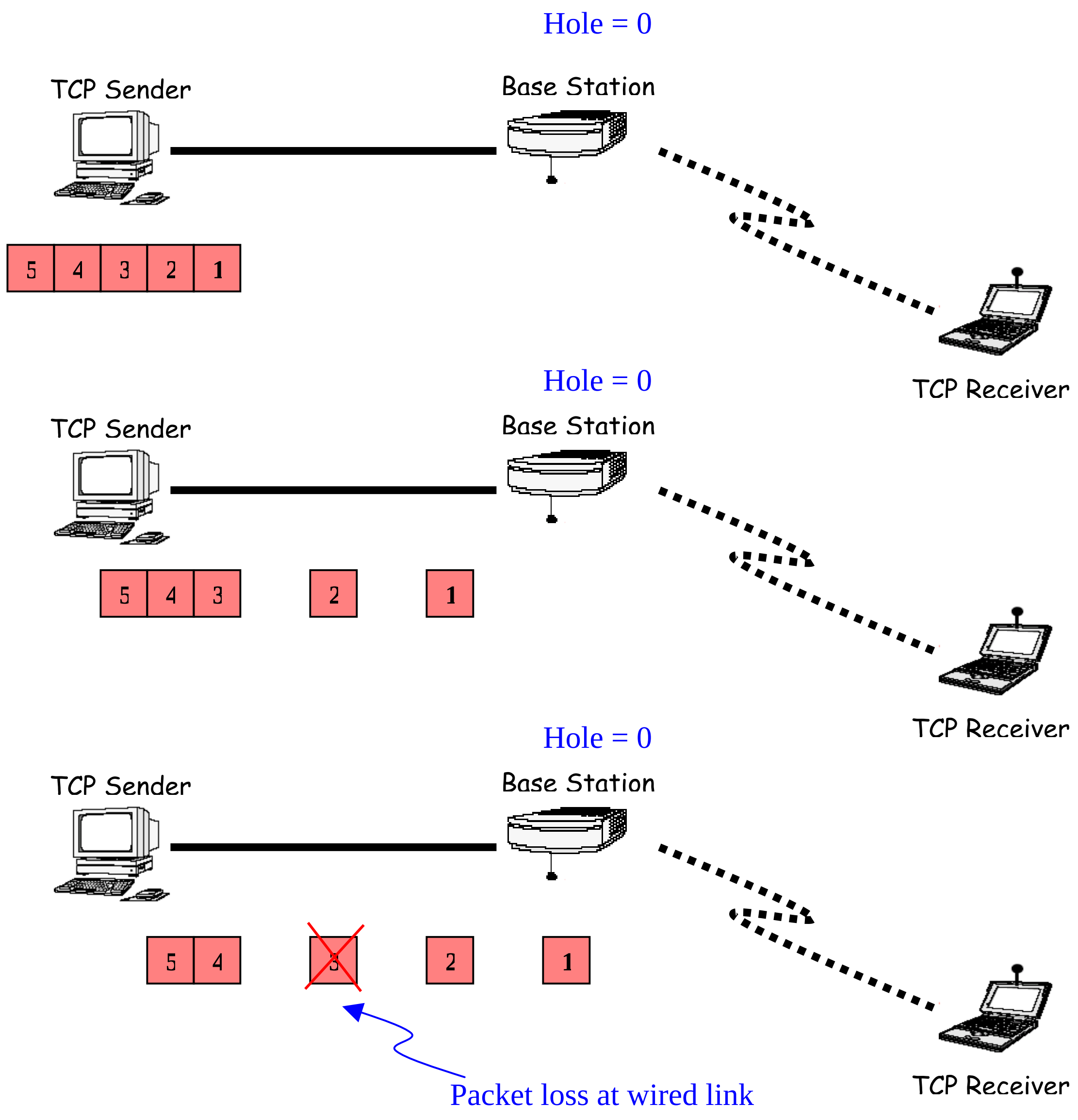


Figure 4.1: ELN part 1

In Figure 4.1, the ELN agent is introduced at the base station. It monitors all TCP segments that arrive over the network. It keeps track of holes in the sequence space as it receives data segments, where a hole is a missing interval in the sequence space. As the data packets are transmitted over the network, packet three is lost in the wired link.

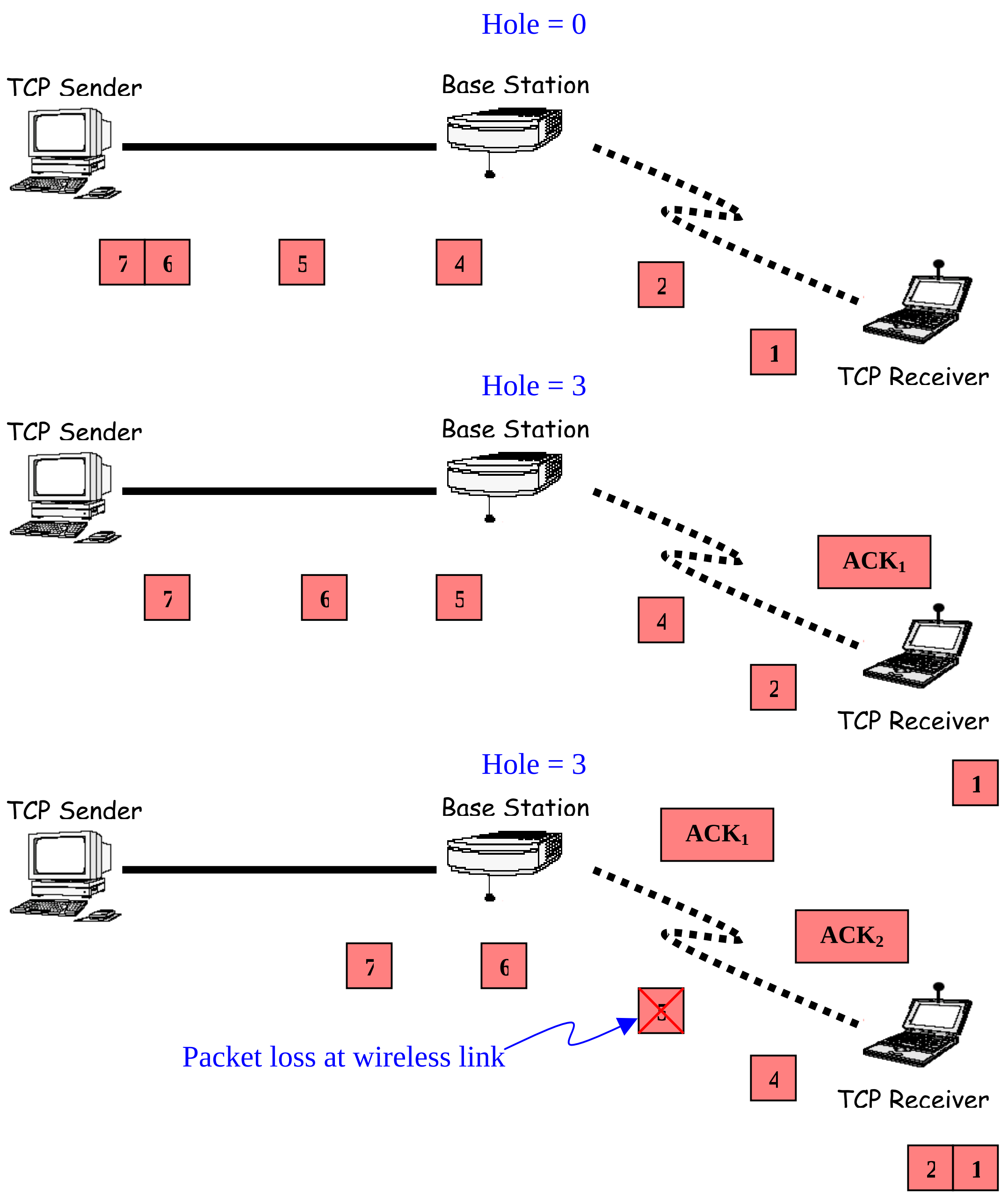


Figure 4.2: ELN part 2

A new data segment that is out of the normal increasing sequence creates a hole, because segments in between have been lost. ELN agent will store the packet loss information as hole at the base station as shown in Figure 4.2. As the data packets are transmitted over the network, packet five is lost in the wireless link.

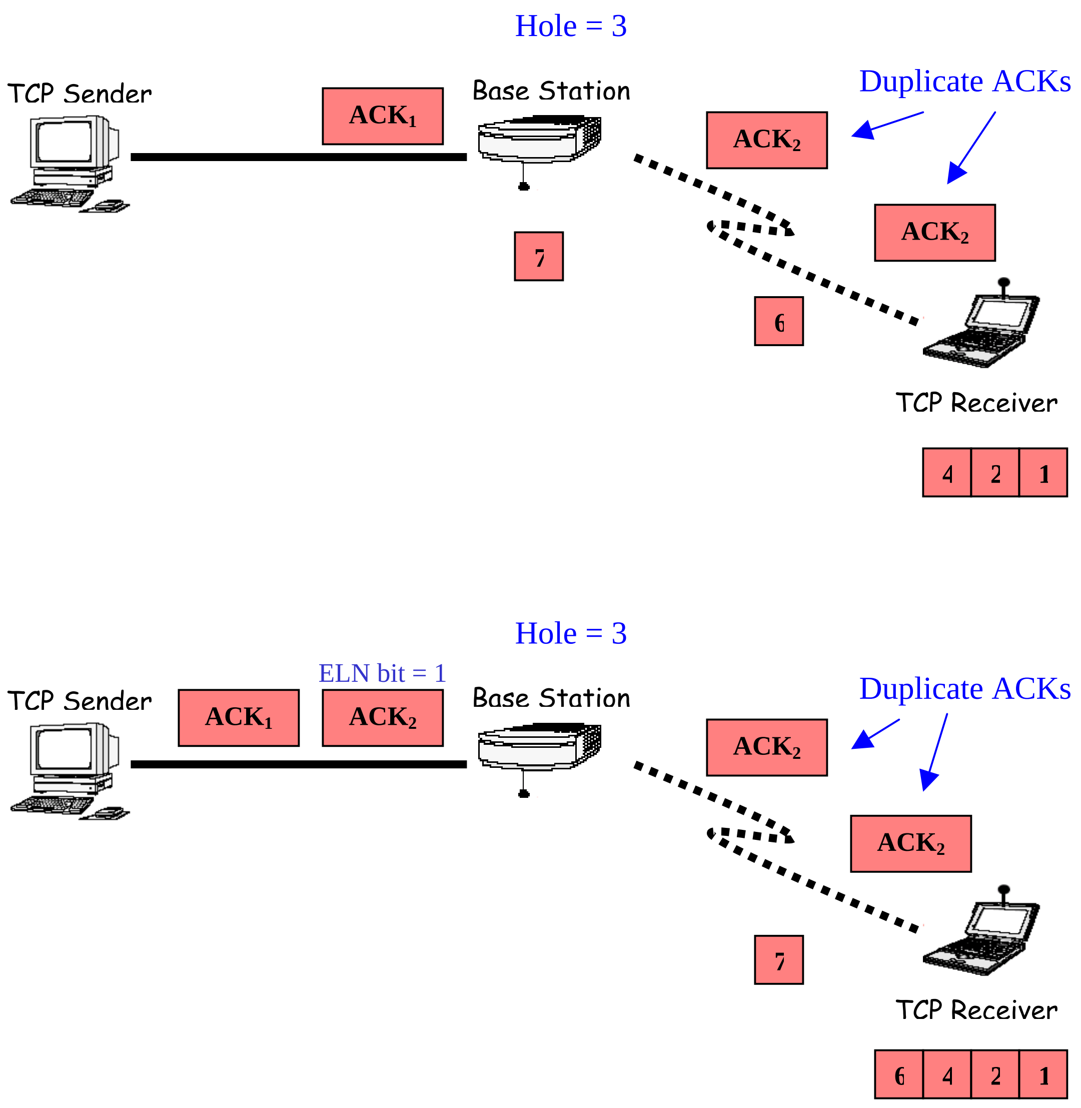


Figure 4.3: ELN part 3

Figure 4.3 showed that when duplicate ACKs, arrive from the receiver, the agent at the base station consults its list of holes. It sets the ELN bit on the ACK_{ELN} as ‘1’ before forwarding to the sender.

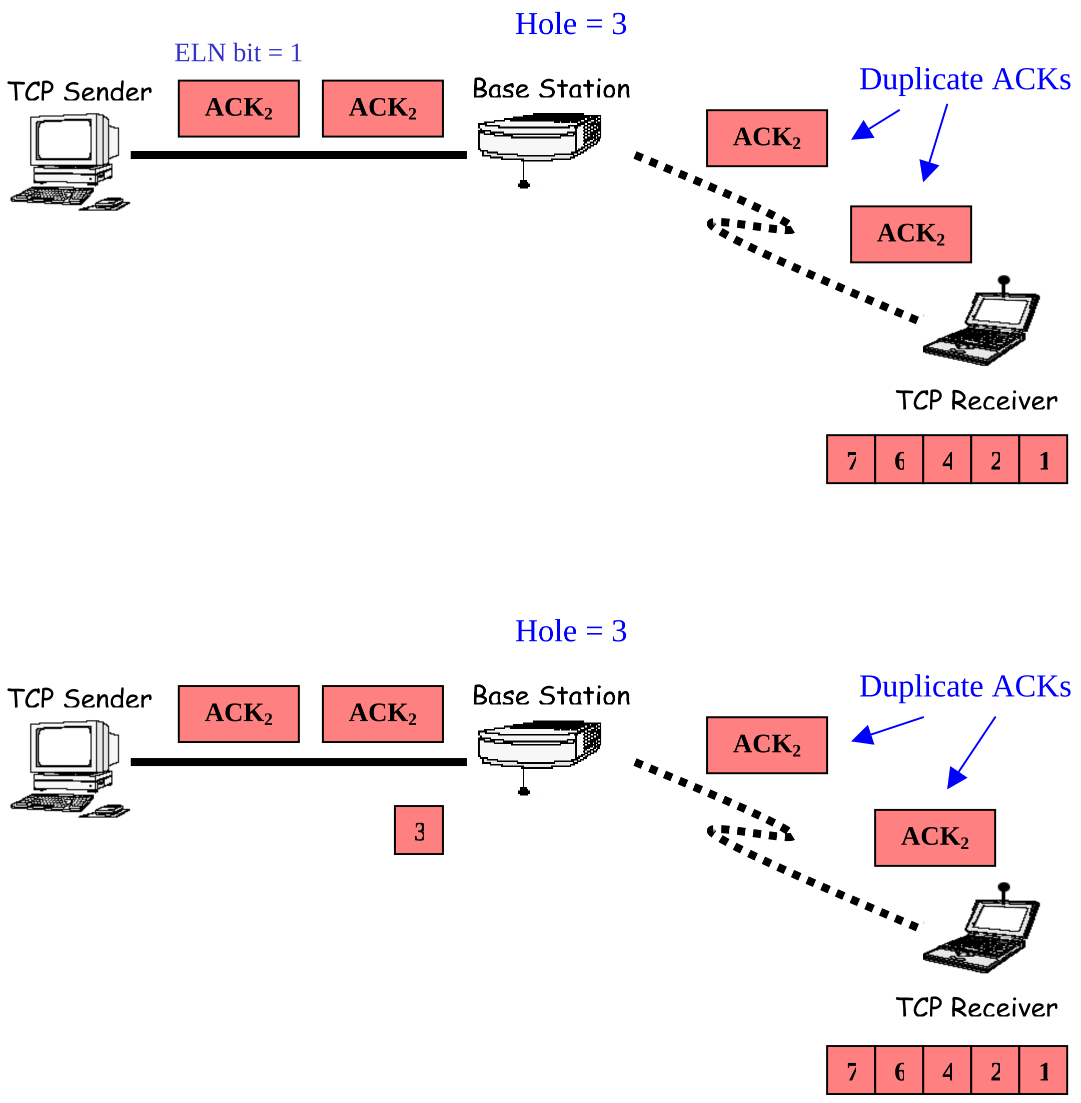


Figure 4.4: ELN part 4

Figure 4.3 showed that upon getting the ACK_{ELN} with ELN bit set as '1'. The source knows that packet is lost at wired link. Therefore it will retransmit the loss packet and take congestion control actions.

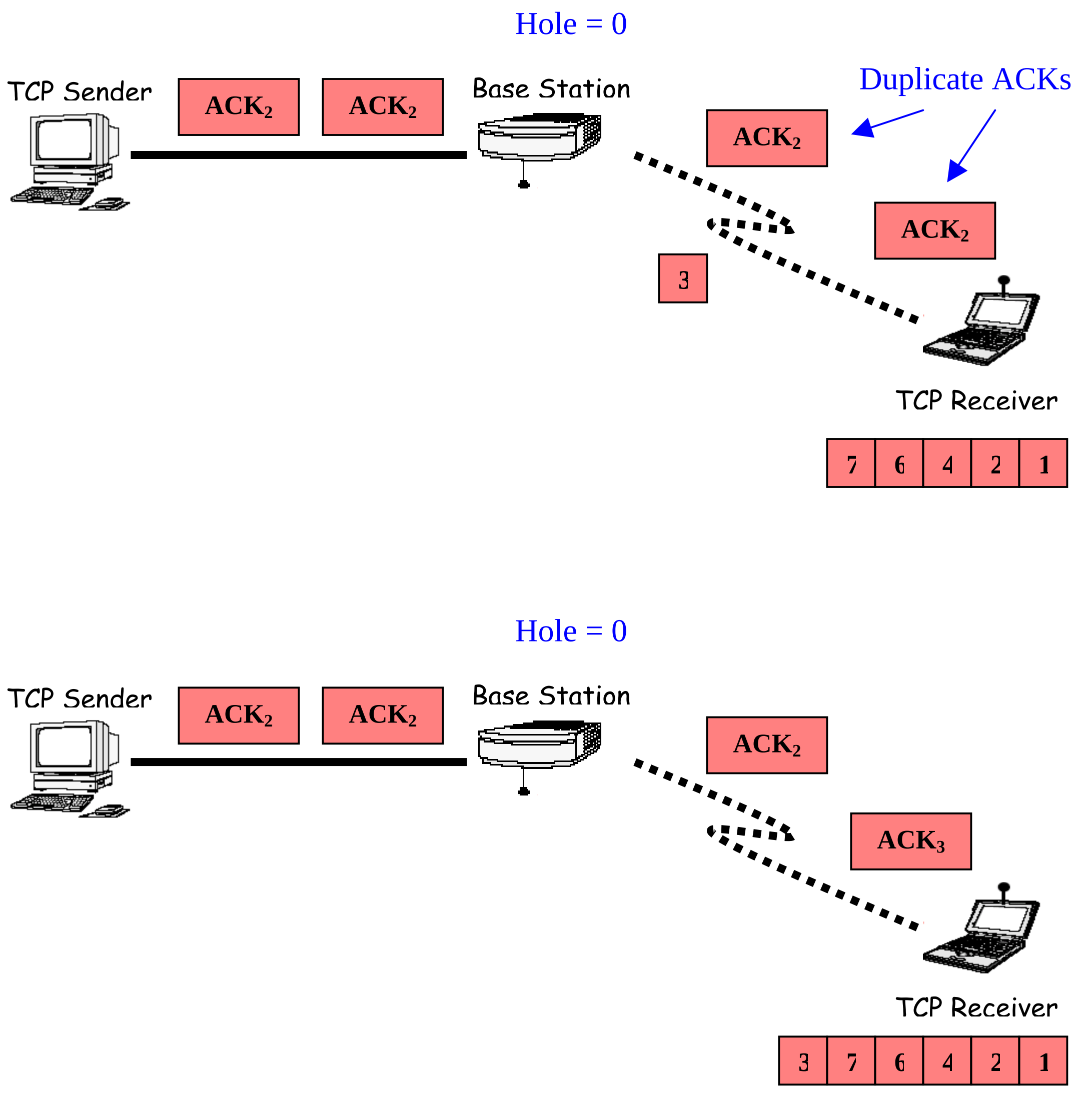


Figure 4.5: ELN part 5

Figure 4.5 showed that once packet three, which is lost at the wired link had been retransmitted, ELN agent would clear the packet loss information at the base station. When the receiver successfully receives packet three, the normal transmitting of ACKs will continue.

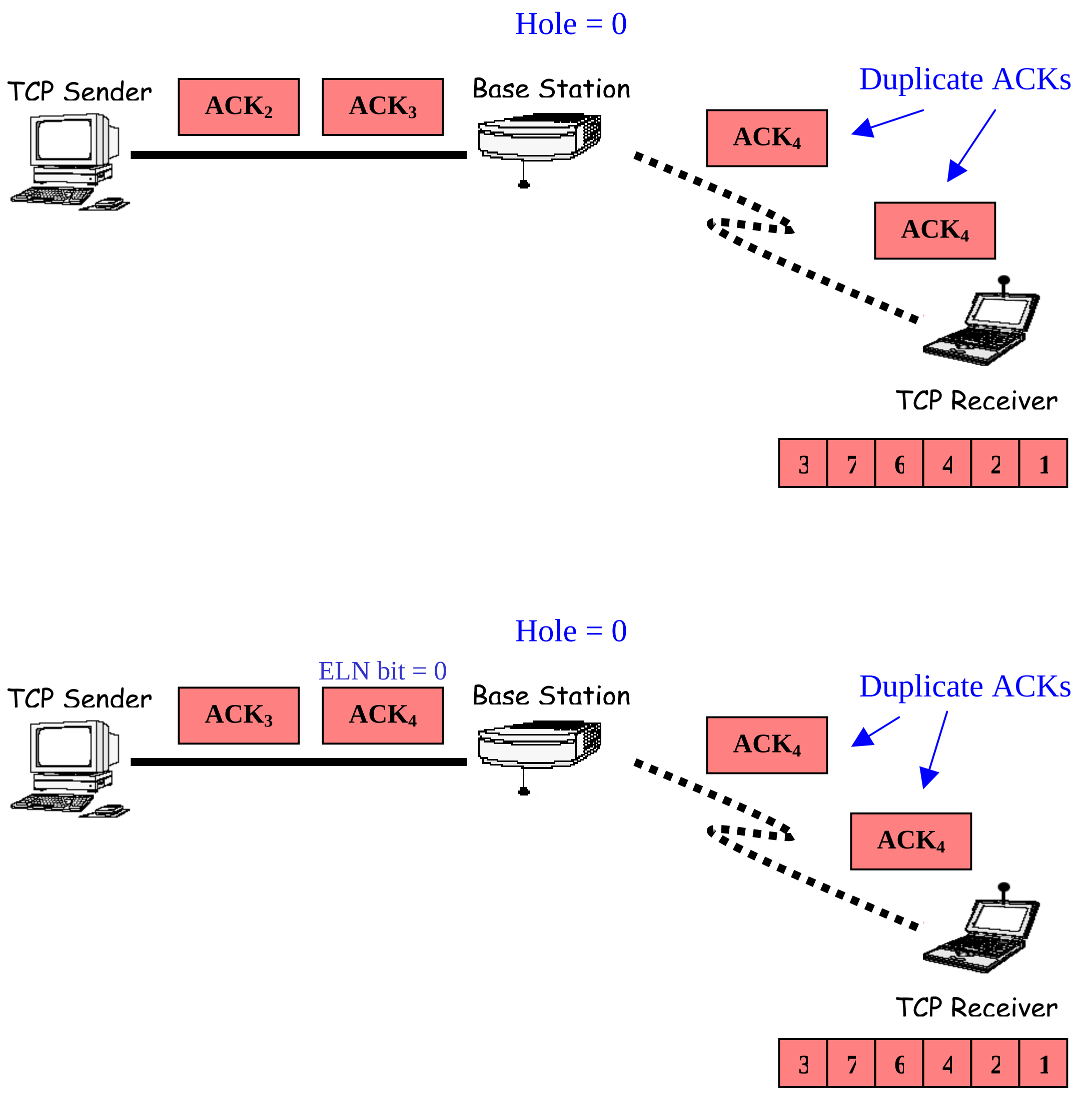


Figure 4.6: ELN part 6

Figure 4.6 showed that when duplicate ACKs, arrive from the receiver, the agent at the base station consults its list of holes. It sets the ELN bit on the ACK_{ELN} as ‘0’ before forwarding to the sender.

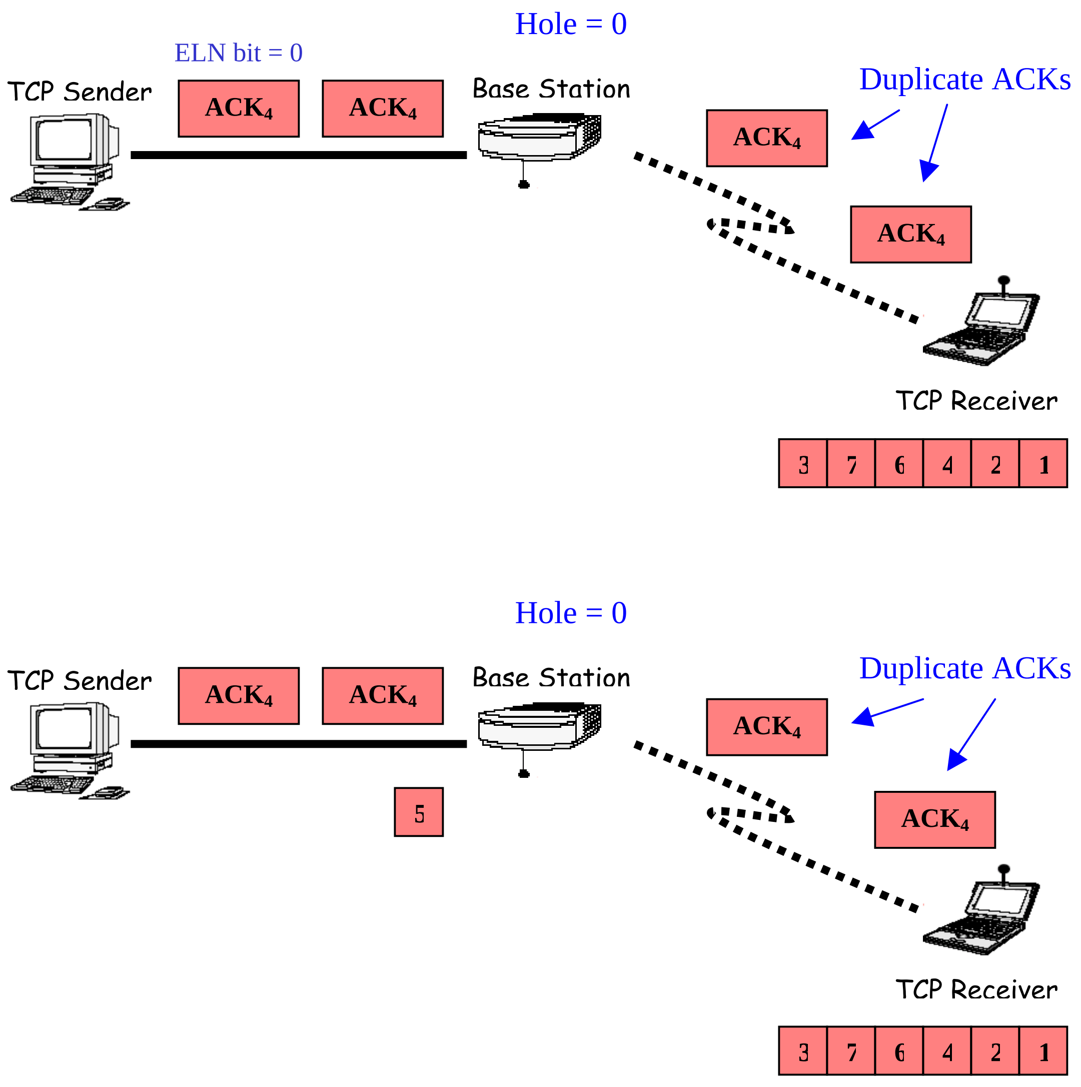


Figure 4.7: ELN part 7

Figure 4.3 showed that upon getting the ACK_{ELN} with ELN bit set as ‘0’. The source knows that packet is lost at wireless link. Therefore it will retransmit the loss packet and not take any congestion control actions.

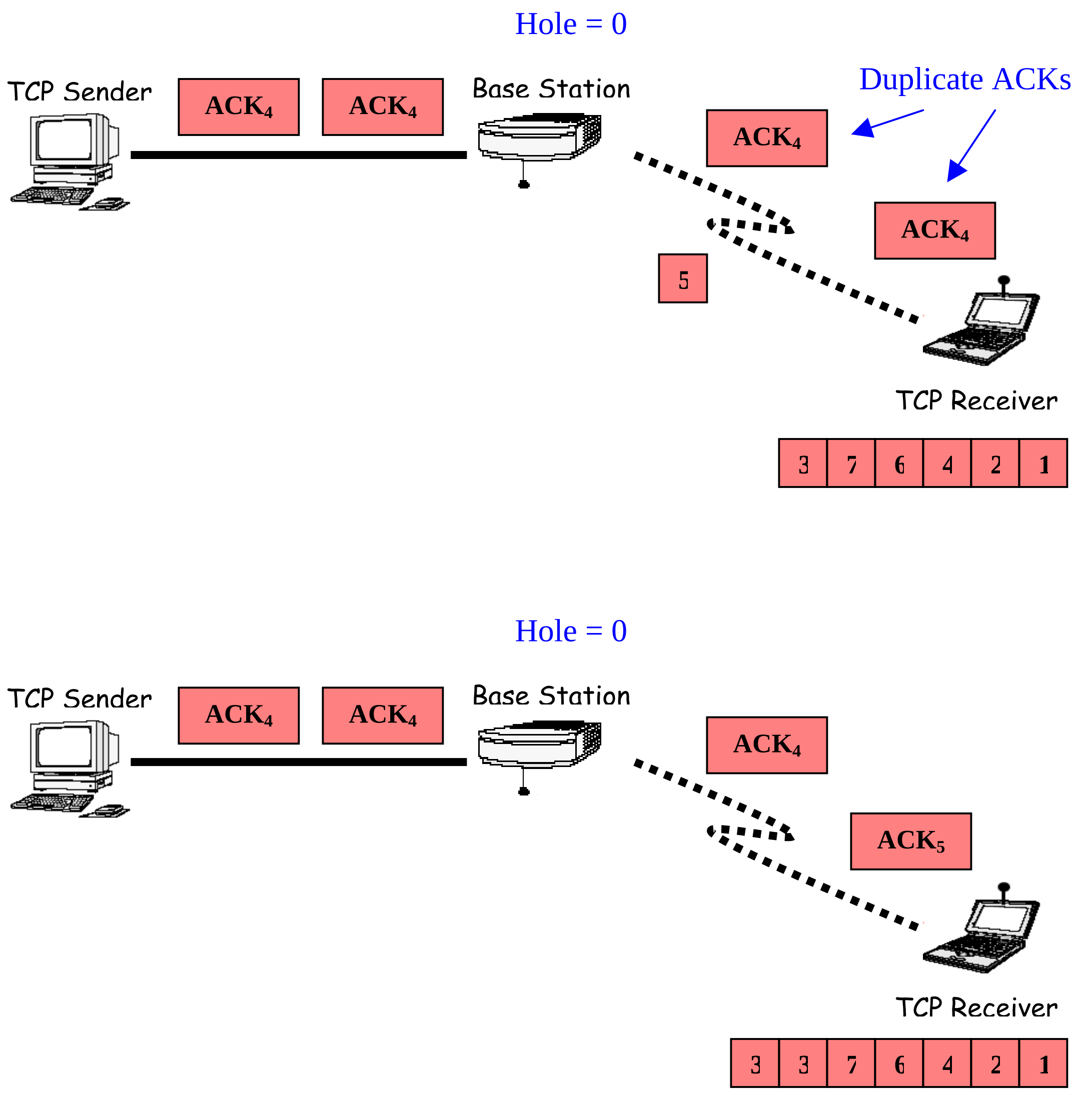


Figure 4.8: ELN part 8

Figure 4.8 showed that packet five, which is lost at the wireless link had been retransmitted. When the receiver successfully receives packet three, the normal transmitting of ACKs will continue.

4.3 Performance Analysis

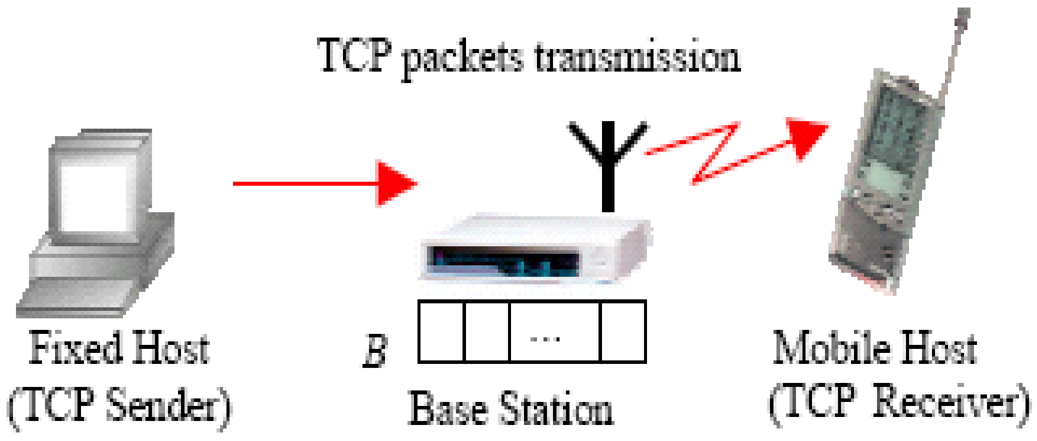


Figure 4.9: ELN test bed setup

Ding and Jamalipour (2001) conducted a simple network simulation send TCP packets from a fixed host to a mobile host. The base station includes a finite-buffer drop-tail gateway, and the network consists wired and wireless links. ELN-ACK protocol has been implemented using C++ programming and Network Simulator was used to simulate the TCP packet transmission in wired cum wireless segments of the network.

In the simulation, the buffer size in the base station was set at 5 packets. Bandwidth of bottleneck link from base station to mobile host was set at 100 packets/msec. Propagation delay was set at 0.2 msec. This includes the time between the release of a packet from the source and its arrival into the link buffer, the time between the transmission of the packet on the bottleneck link and its arrival at its destination and the time between the arrival of the packet at the destination and the arrival of the corresponding acknowledgment at the source.

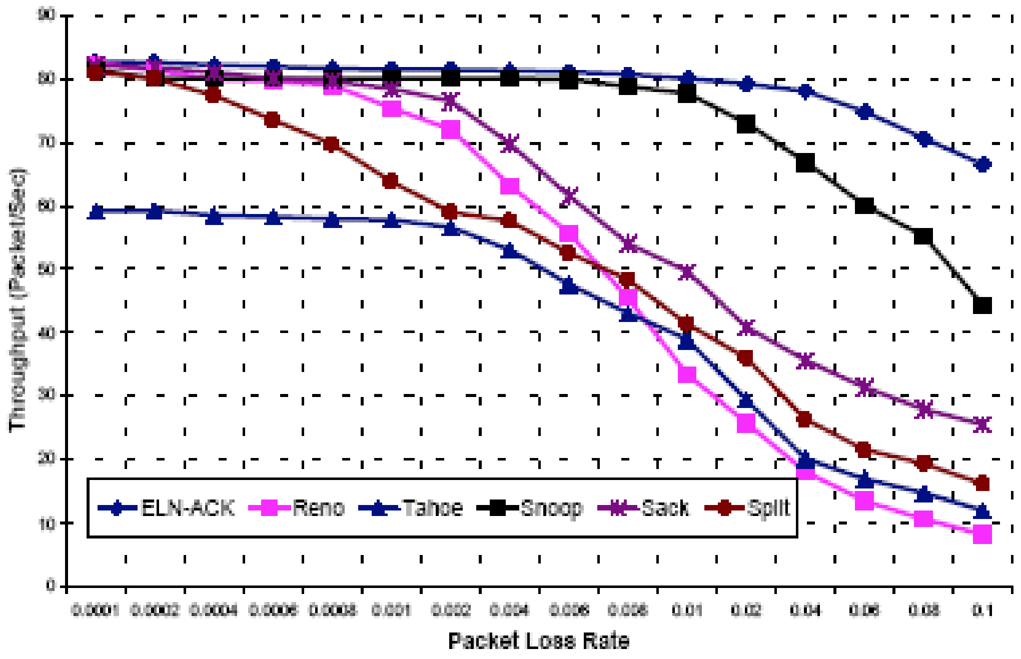


Figure 4.10: Throughput performance comparison

In Figure 4.10, ELN-ACK protocol was compared with different performance enhancing mechanisms such as Snoop protocol (snoop), Selective acknowledgement (Sack) and Split connection (Split). Two versions of TCP were also taken into consideration; which were TCP-Reno and TCP Tahoe.

Based on the results shown in this figure Ding and Jamalipour (2001) found out that Snoop and ELN-ACK protocols provide significant performance improvement when the packet loss rate becomes larger than around 0.3%. The throughput performance of the Snoop and ELN-ACK protocols remains very close until the packet loss rate of 1%. Beyond that, the ELN-ACK outperforms the Snoop protocol. The increase in the performance of Snoop and ELN-ACK compared with other TCP methods is clear since these two protocols provide better differentiation of the packet loss types over wireless link and the that of wired link. On top of that, the ELN-ACK protocol was able to improve the throughput performance even more by sending information on the reason of packet loss to the TCP sender whereas the Snoop protocol tries to handle all wireless-related losses at its snoop agent located in the base station. This goes to show that, ELN-ACK protocol was able to add extra features to the Snoop protocol and immunes all packet loss even when the packet loss rate is high and the snoop agent cannot handle them.

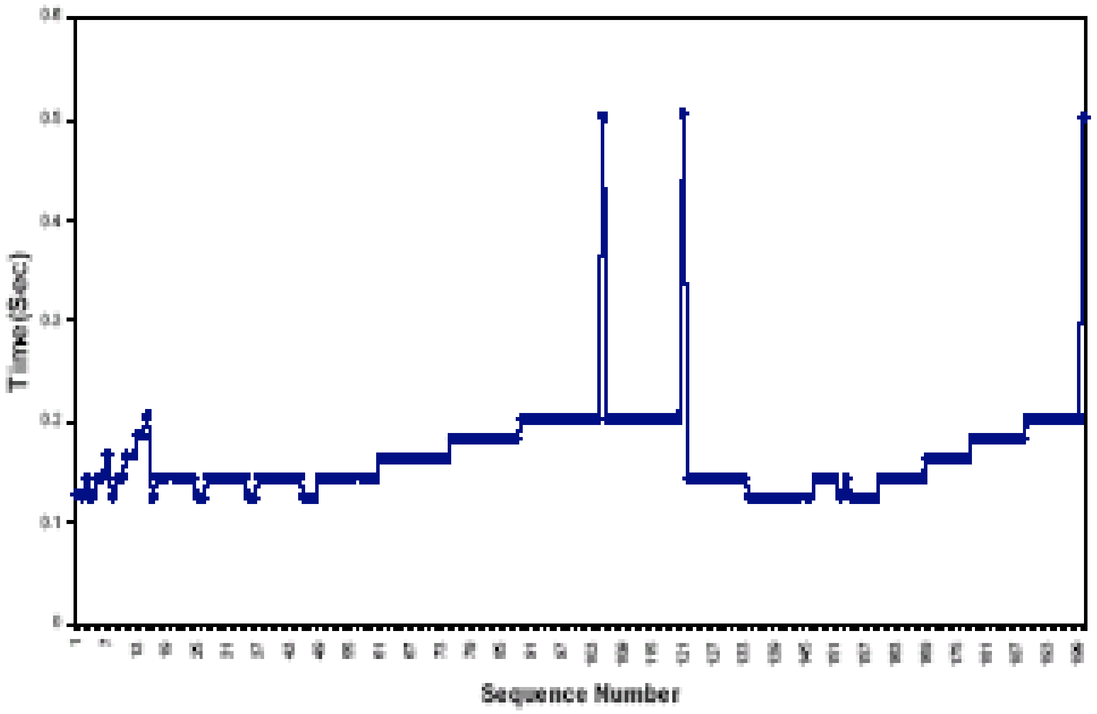


Figure 4.11: End-to-end delay for TCP-Reno (without wireless error)

The delay of 200-packet transmission using TCP-Reno and the proposed ELN-ACK protocols are shown in Figure 4.11 to 4.13. TCP-Reno produced quite good end-to-end delay performance in absence of wireless error, as shown in Figure 4.11. Ding and Jamalipour (2001) noted that the mean end-to-end delay is about 0.15 to 0.2 second. There were two packets with delay around 0.5 second. This was due to network congestion, causing these two packets to be retransmitted by the loss recovery mechanism.

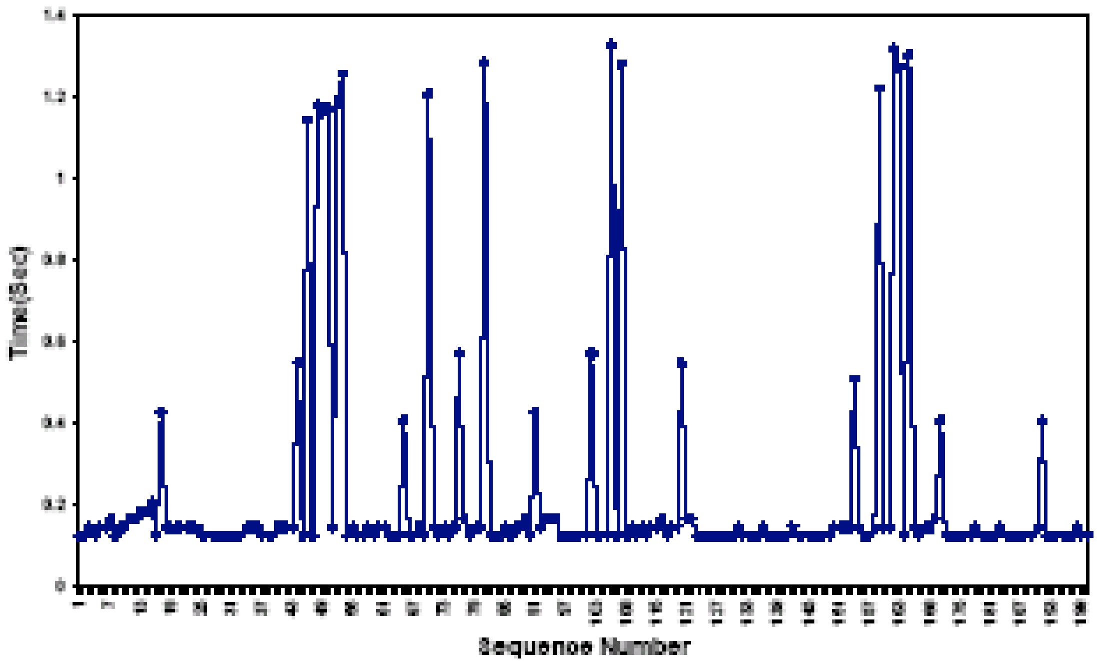


Figure 4.12: End-to-end delay for TCP-Reno (wireless packet loss rate = 0.1)

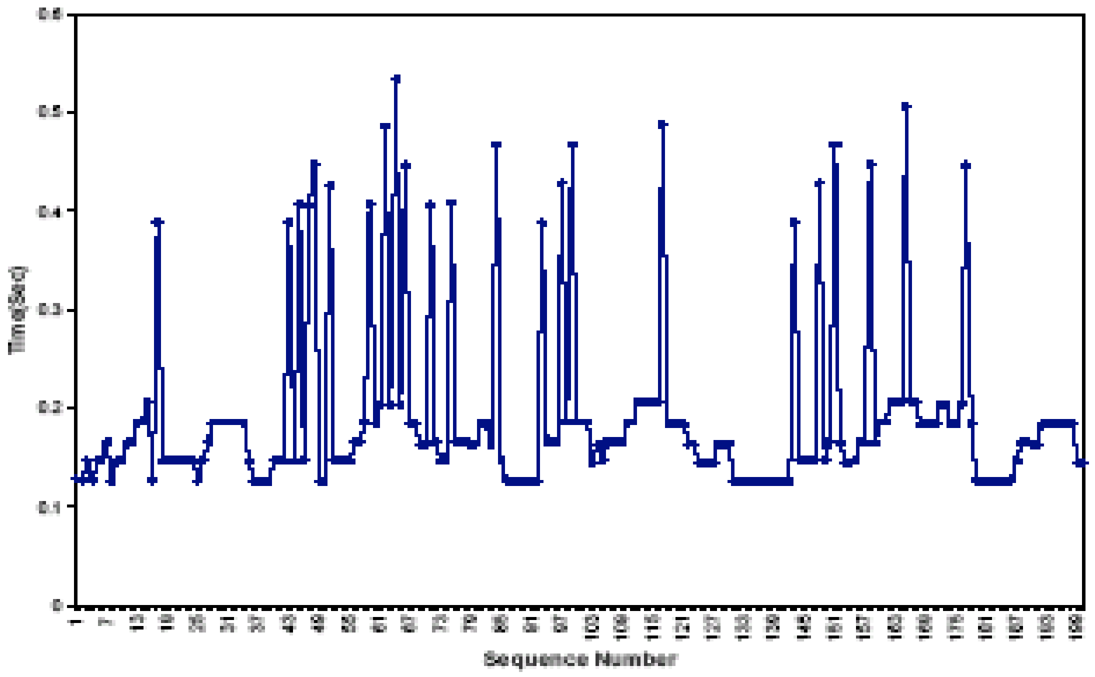


Figure 4.13: End-to-end delay for ELN-ACK (wireless packet loss rate = 0.1)

In Figure 4.12 and 4.13, the packet loss rate on the wireless link is 0.1, which means that in every 200-packet transmission there are about 20 packets lost in the wireless channel. Based on the simulation results in Figure 4.12, TCP-Reno had a mean transmission delay of about 0.15 to 0.25 second. In the 200 packets transmission, there were 24 packets whose transmission delay was significantly above 0.2 second. These packets were lost either because of network congestion or wireless error in the network. Of the 24 retransmitted packets, 11 packets have the delay around 0.4 to 0.6 second. This means that these packets are retransmitted by loss recovery mechanism without invoking time out, 13 other packets have a delay around 1 to 1.4 second, which means that these packets are timed out. Due to the wireless packet loss, the TCP sender always has to wait for time out before re-transmitting a lost packet.

On the other hand, Ding and Jamalipour (2001) found out that ELN-ACK algorithm can efficiently avoid the timeout by re-transmitting a lost packet immediately. As shown in Figure 4.13, there was no packet delay above 1 second in the ELN-ACK algorithm. All lost packets were retransmitted by loss recovery mechanism or by fast retransmission. From Figure 4.13, Ding and Jamalipour (2001) also found out that the mean end-to-end delay for packet transmission was about 0.1 to 0.2 second. There were 23 packets with end-to-end delay between 0.4 to 0.6 seconds. Most of these packets were lost in wireless channel when it was first transmitted by the TCP-sender. Unlike the TCP-Reno, the TCP sender knows that these packets were lost due to wireless error and not due to network congestion. Thus the lost packet can be retransmitted efficiently without incurring any window deduction, which avoids long idle time to wait for timeout.

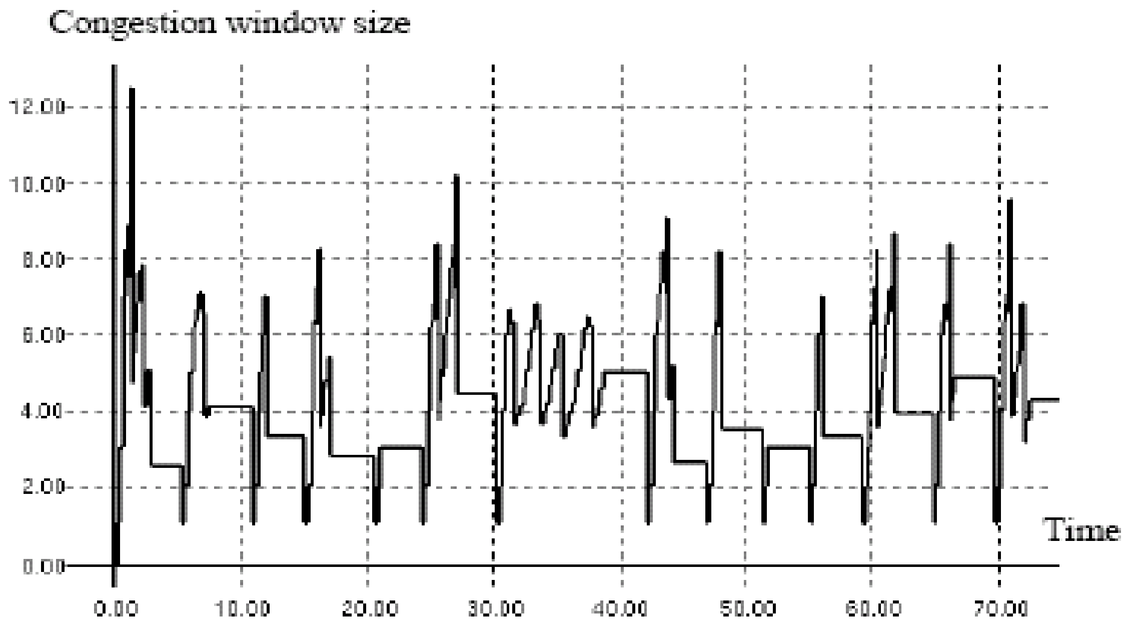


Figure 4.14: Window evolution for TCP-Reno (wireless packet loss rate = 0.1)

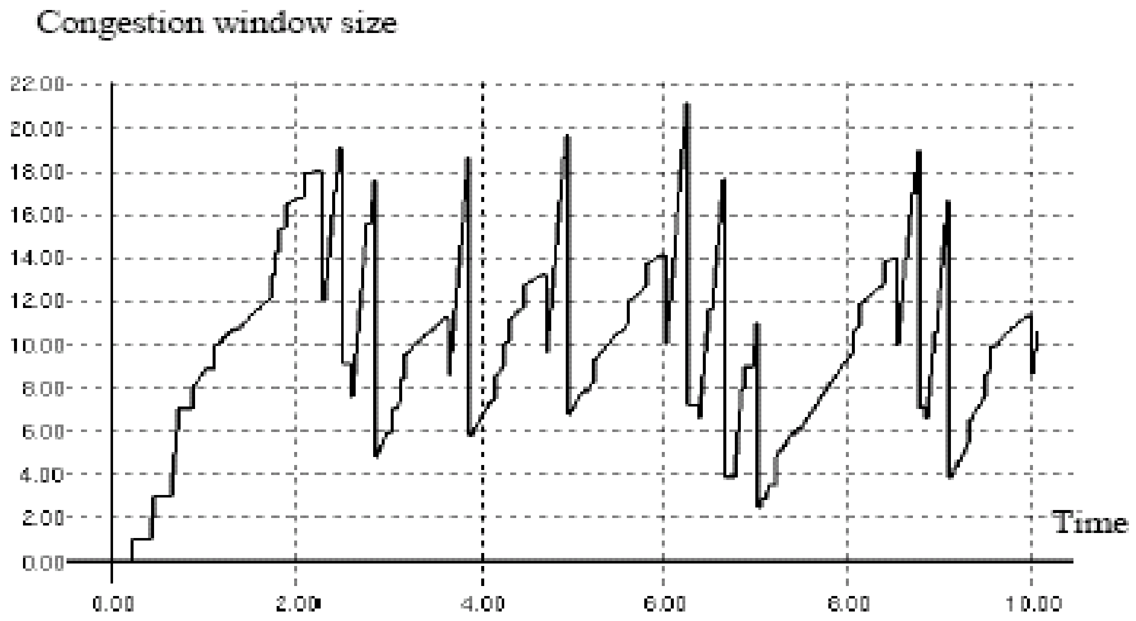


Figure 4.15: Window evolution for ELN-ACK (wireless packet loss rate = 0.1)

Figures 4.14 and 4.15 show the congestion window size in the procession of transmitting 200 packets for TCP-Reno and TCP with ELN-ACK, respectively. With reference to Figure 4.14, Ding and Jamalipour (2001) noted that the main reason for the occurrence of these timeouts in the TCP-Reno algorithm was the small congestion window, which did not transmit enough duplicate acknowledgment to the TCP sender. The number of duplicate ACKs arrived were also not sufficient to trigger a fast retransmission. This caused a timeout-driven retransmission that keeps the link idle for long periods of time. As shown in Figure 4.14, the congestion window size was not able to increase big enough and was always reduced to one due to timeout. Therefore in a high packet loss environment, the TCP-Reno cannot efficiently transmit packet.

When there is no wireless packet loss, TCP-Reno sender was able to retransmit packet by using the loss recovery algorithm as the only packet loss in transmission was due to network congestion. The window size is halved when loss recovery happens, but no time out happens, because the congestion window was kept big enough and there were sufficient duplicate ACKs transmitted back to trigger the loss recovery. From Figure 4.15, it can be seen that the ELN-ACK is an effective way that can retransmit the lost packet quickly due to wireless error, keep the congestion window wide, and thereby eliminating timeout and long idle time periods. Compared with TCP-Reno, ELN-ACK algorithm significantly improves the end-to-end delay performance.

4.4 Advantages and Disadvantages

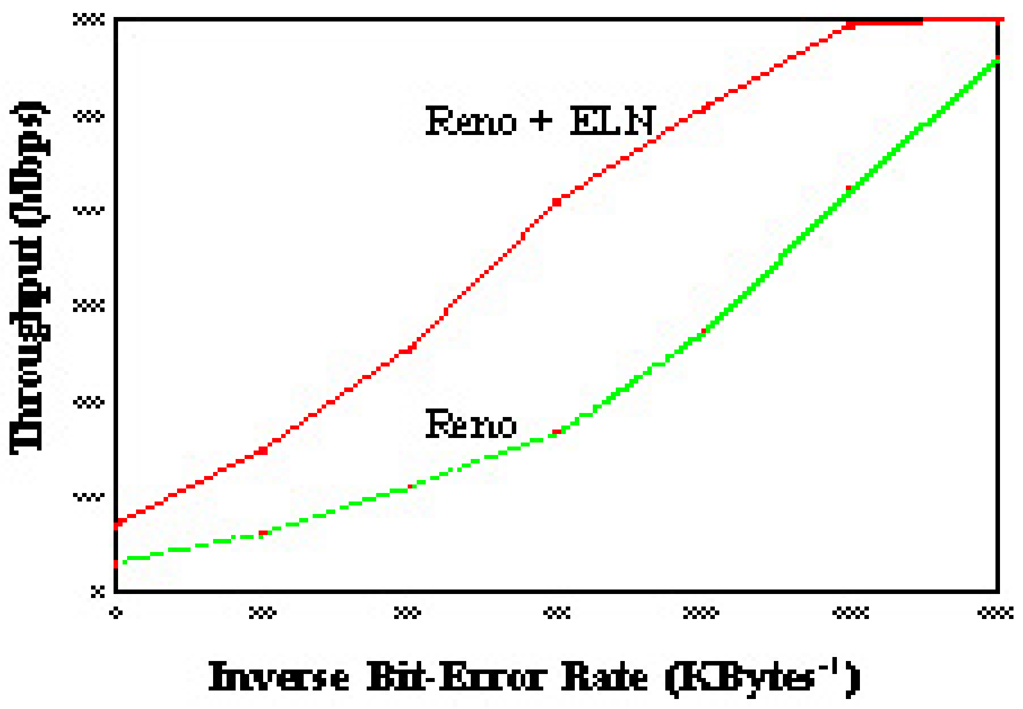


Figure 4.16: Throughput of TCP Reno and Reno enhanced with ELN

Balakrishnan and Katz (1998) found out from their experiments, which measured the performance of data transfer from the MH to the FH. The experiment was conducted over a range of exponentially-distributed bit-error rates. As shown in Figure 4.14, there were significant performance benefits of using the snoop protocol coupled with the ELN mechanism. These measurements were made for wide-area transfers between UC Berkeley and IBM Watson, across one wireless WaveLAN hop and 16 Internet hops. At medium to high error rates, the performance improvement due to ELN is roughly a factor of 2. At lower error rates, TCP Reno performs quite well as expected, and the benefits of ELN are not as pronounced. The main advantage of ELN was that it helped to maintain a large TCP congestion window even when wireless error rates were high, reacting only to congestion.

However Ewerlid (2001) noted that the main disadvantage of ELN mechanism was that it required TCP-stack modifications at all endpoints. Therefore ELN mechanism required standardization of modifications in TCP followed by widespread acceptance of these changes.

Chapter 5

Explicit Congestion Notification (ECN)

5.1 Introduction

Ramakrishnan *et al* (2001) noted that loss, as an indication of congestion in the network is appropriate for pure best-effort data carried by TCP, with little or no sensitivity to delay or loss of individual packets. In addition, TCP's congestion management algorithms have techniques built-in to minimize the impact of losses, from a throughput perspective. However, these mechanisms are not intended to help applications that are in fact sensitive to the delay or loss of one or more individual packets. Interactive traffic such as telnet, web-browsing, and transfer of audio and video data can be sensitive to packet losses or to the increased latency of the packet caused by the need to retransmit the packet after a loss.

Durresi *et al* (2002) found out that, congestion remains the main obstacle to Quality of Service (QoS) on the Internet. Congestion is a critical problem especially in wireless networks, where TCP congestion control performance is affected by intrinsic wireless link characteristics such latency, bandwidth, packet loss due to congestion, and losses due to transmission errors links. One of most promising schemes to improve TCP congestion control is Explicit Congestion Notification (ECN). ECN is the only mechanism that delivers explicit congestion signals to the source. So improving the ECN feedback is essential for the future data, wireless networks and their QoS guarantees.

As noted by Deshpande (1999), currently TCP assumes that all the losses are due to congestion and does not distinguish between losses due to wireless link and those due to congestion. As the wireless networks have higher bit-error rates than fixed networks, determining whether a segment was lost due to congestion or wireless link may allow TCP to achieve better performance in high Bit Error Rate (BER) environments than currently possible. Adding ECN mechanism to TCP may help to improve TCP performance in wireless link.

Active Queue Management (AQM) mechanism is used in ECN to detect congestion before the queue overflows, and provide an indication of this congestion to the end nodes. Thus, active queue management can reduce unnecessary queuing delay for all traffic sharing that queue. Active queue management avoids some of the bad properties of dropping on queue overflow, including the undesirable synchronization of loss across multiple flows as noted by Ramakrishnan *et al* (2001). More importantly, active queue management means that transport protocols with mechanisms for congestion control do not have to rely on buffer overflow as the only indication of congestion.

AQM can set a Congestion Experienced (CE) bit in the packet header instead of dropping the packet, when such a field is provided in the IP header and understood by the transport protocol. The use of the CE bit with ECN allows the receiver to receive the packet, avoiding the potential for excessive delays due to retransmissions after packet losses.

5.2 Algorithms

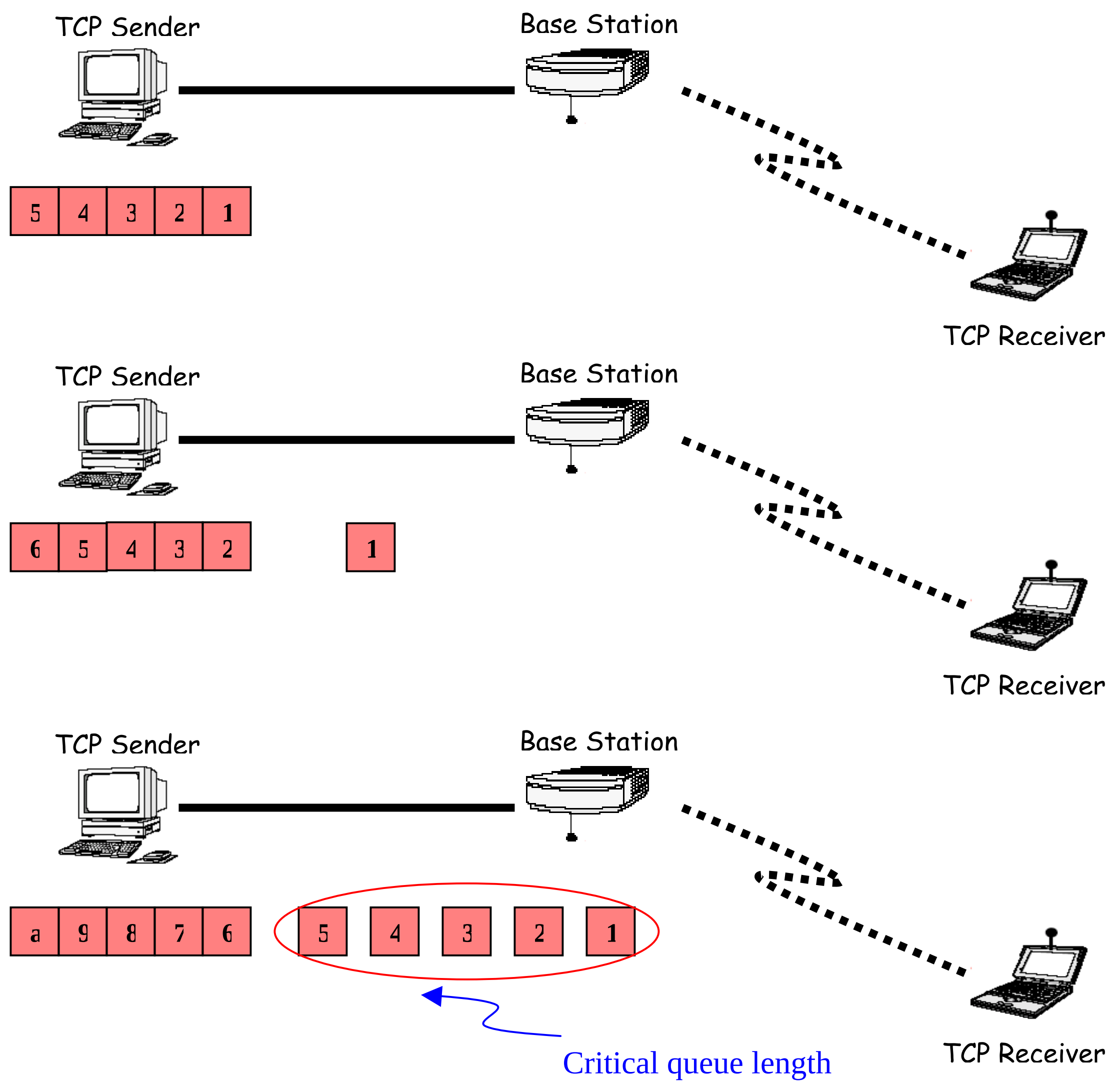


Figure 5.1: ECN part 1

In Figure 5.1, The ECN agent with AQM is introduced at the Base Station in the ECN capable network. AQM keeps track of the average queue length at the input of the Base Station. Critical queue length indicates congestion at the network. Base Station's buffer is not yet full, but it is preparing to drop packet.

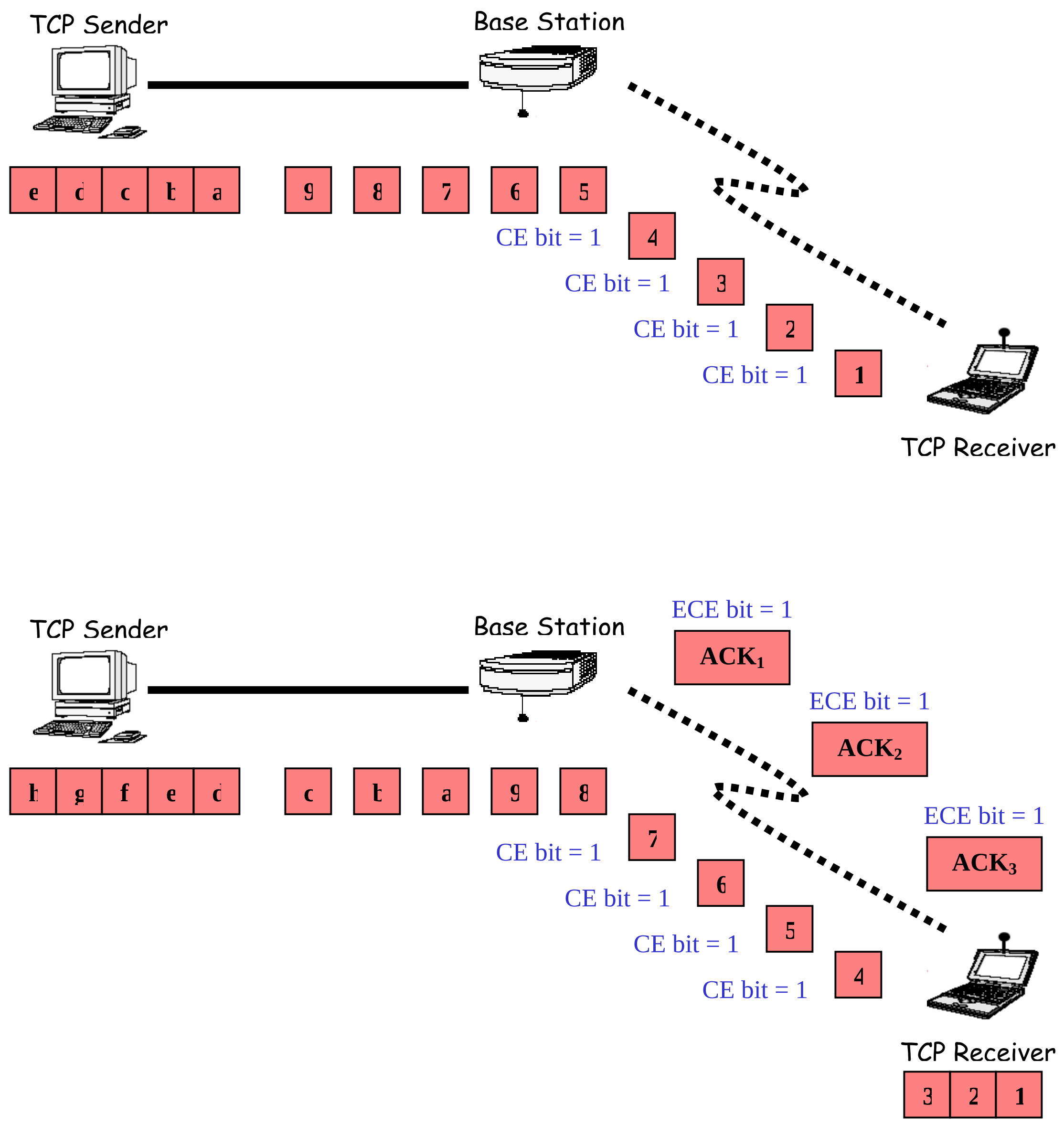


Figure 5.2: ECN part 2

Instead of dropping the packets, AQM sets the CE bit in the packet header to ‘1’, indicating congestion at the network as shown in Figure 5.2. Upon receipt of a packet with the CE bit set, TCP Receiver sends back an acknowledgment (ACK) with the ECN-Echo (ECE) bit set to ‘1’ in its header.

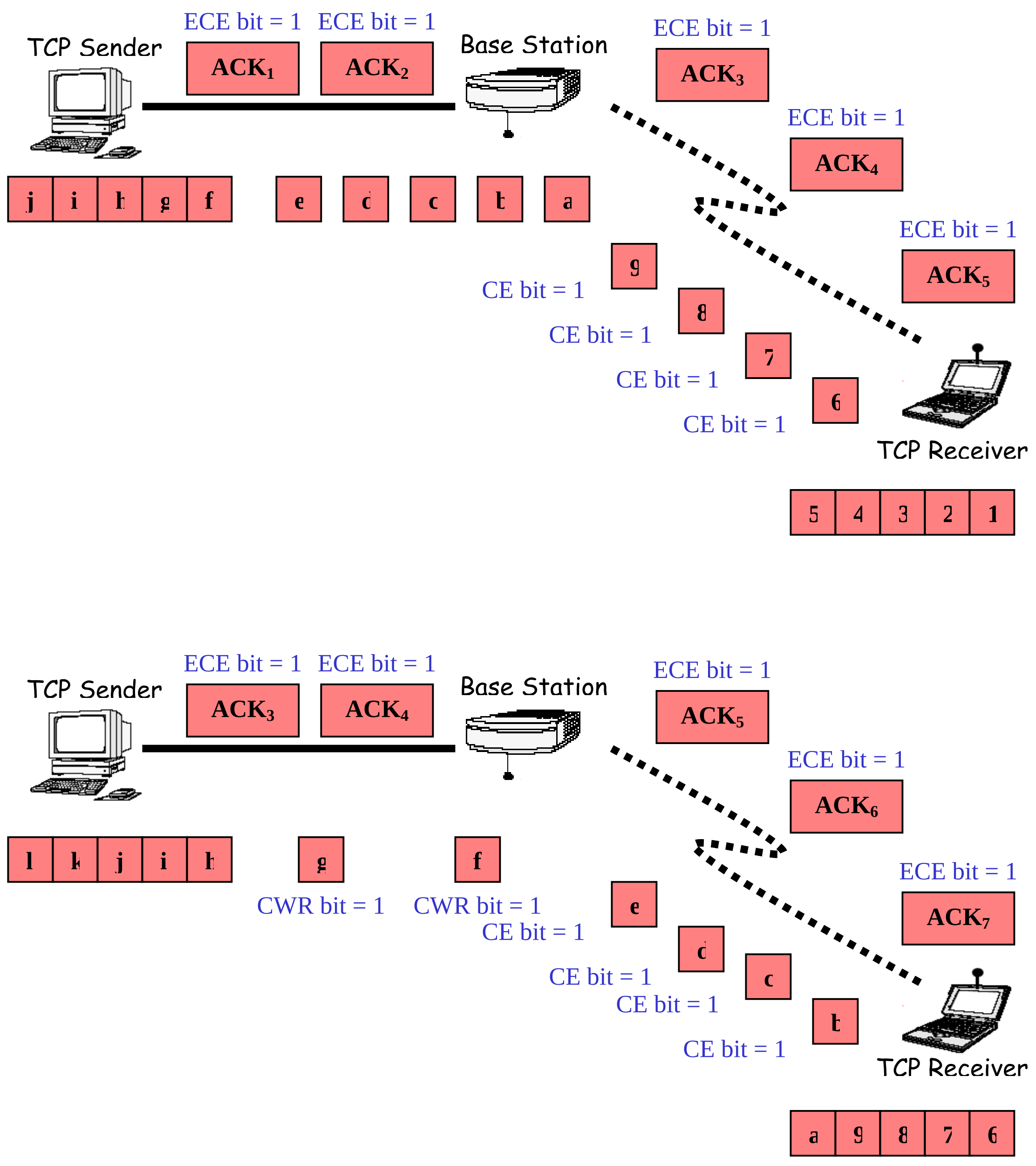
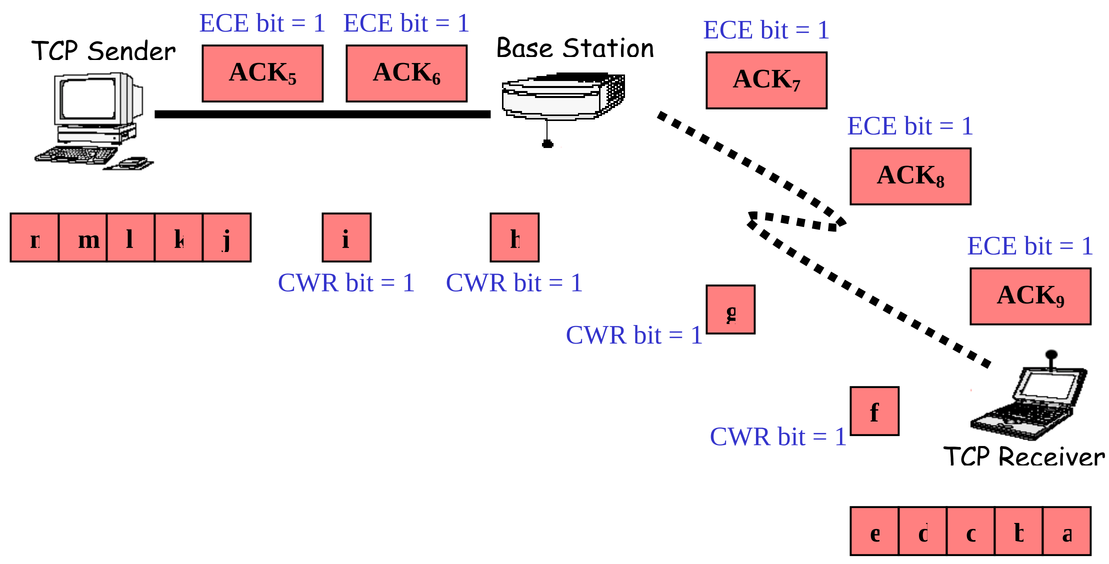


Figure 5.3: ECN part 3

As shown in Figure 5.3, these effectively notify the TCP Sender of the congestion problem in the network. Upon receipt of the first ACK carrying the ECE bit. The TCP Sender must trigger congestion control mechanisms. Congestion window will be halve and the sender sets the Congestion Window Reduced (CWR) bit in the header of the next data segment it transmits to ‘1’.



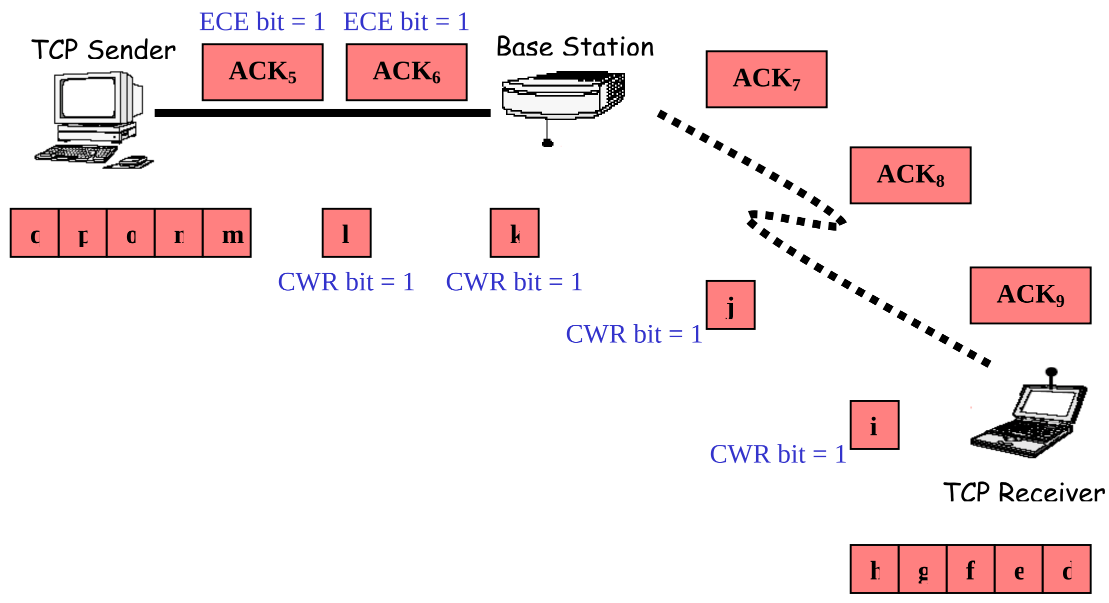
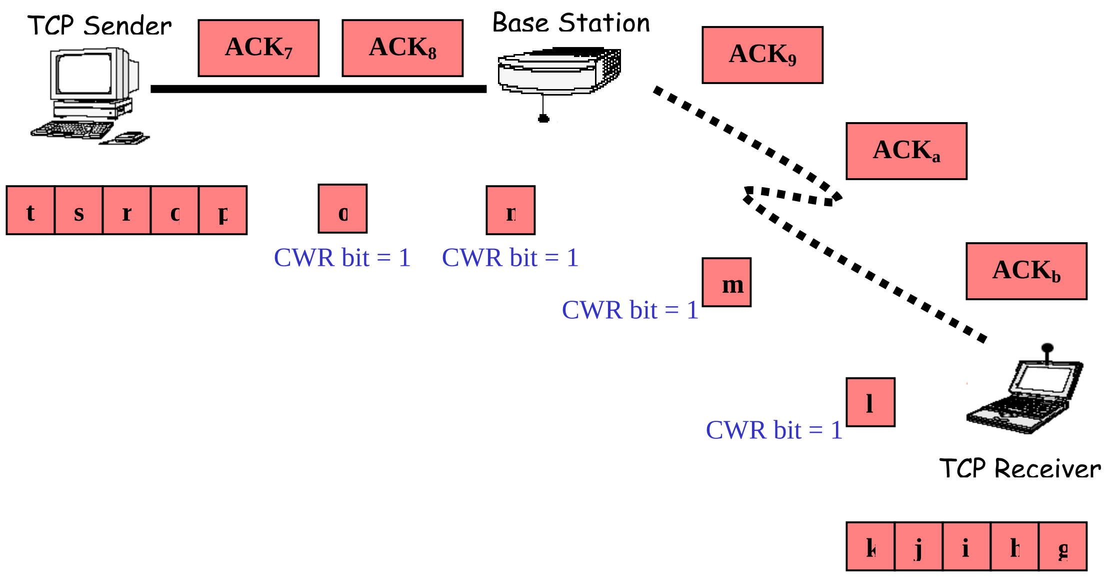


Figure 5.4: ECN part 4

As Shown in Figure 5.4, when Base Station detects queue length is no longer critical, it will stop setting the CE bit. The receiver continues to set the ECE bit in ACK to ‘1’, until it receives notification from the sender, via the CWR bit, that the congestion window has been reduced. Upon receipt of the first packet carrying the CWR bit. The receiver stop setting the ECE bit.



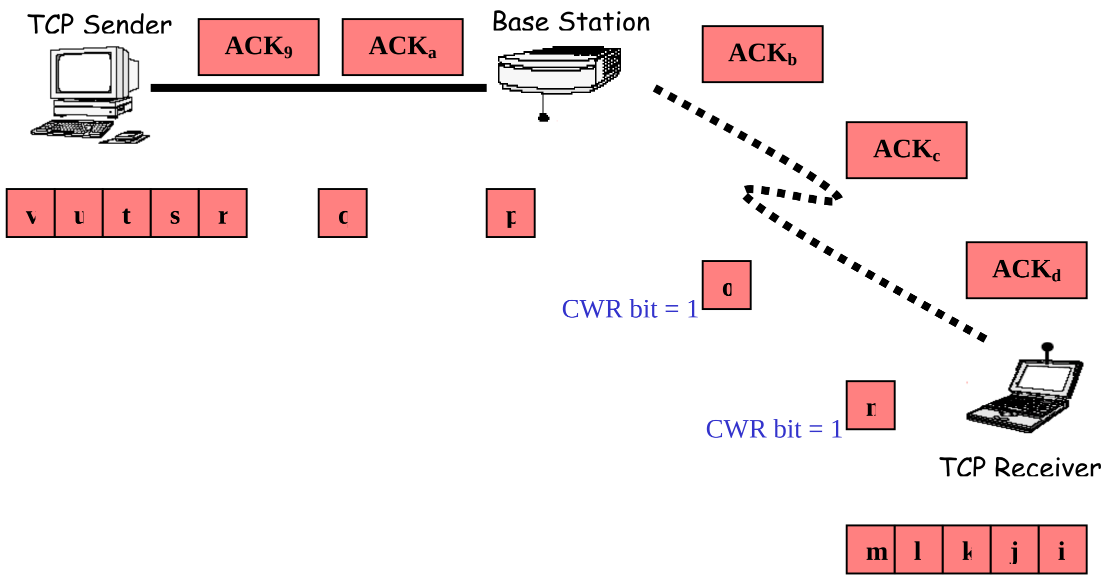


Figure 5.5: ECN part 5

As shown in Figure 5.5, upon receipt of the first ACK without the ECE bit. The sender stop setting the CWR bit.

5.3 Performance Analysis

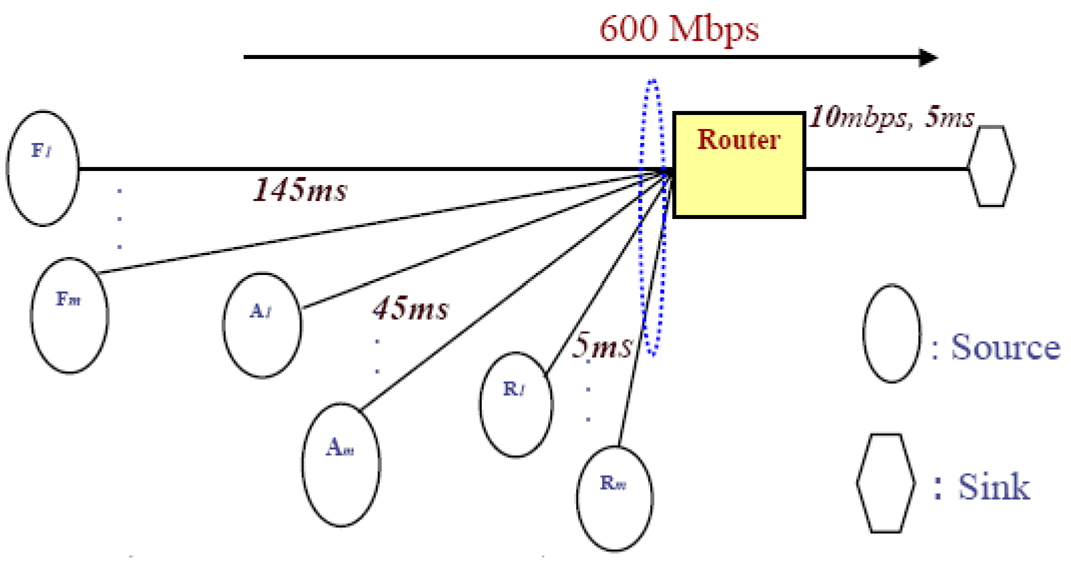


Figure 5.6: ECN test bed setup 1

Kinicki and Zheng (2001) conducted an experiment with the above test bed setup in Figure 5.6 to evaluate the performance of Random Early Detection (RED) mechanism and ECN. RED detests congestion “early by maintaining an exponential-weighted average queue size. RED probabilistically drops packets before the queue overflows to signal congestion to TCP sources, whereas ECN is a RED extension that marks packets to signal congestion. The experiment was conducted with Network Simulator 2 (NS2) with average queue length threshold for triggering probabilistic drops/marks set at 5, buffer size set at 50 packets and 100 seconds of simulation time. The goodput (Mbps) of RED and ECN were evaluated. Goodput is the rate at which packets arrive at the receiver. It differs from throughput as retransmissions are excluded from goodput.

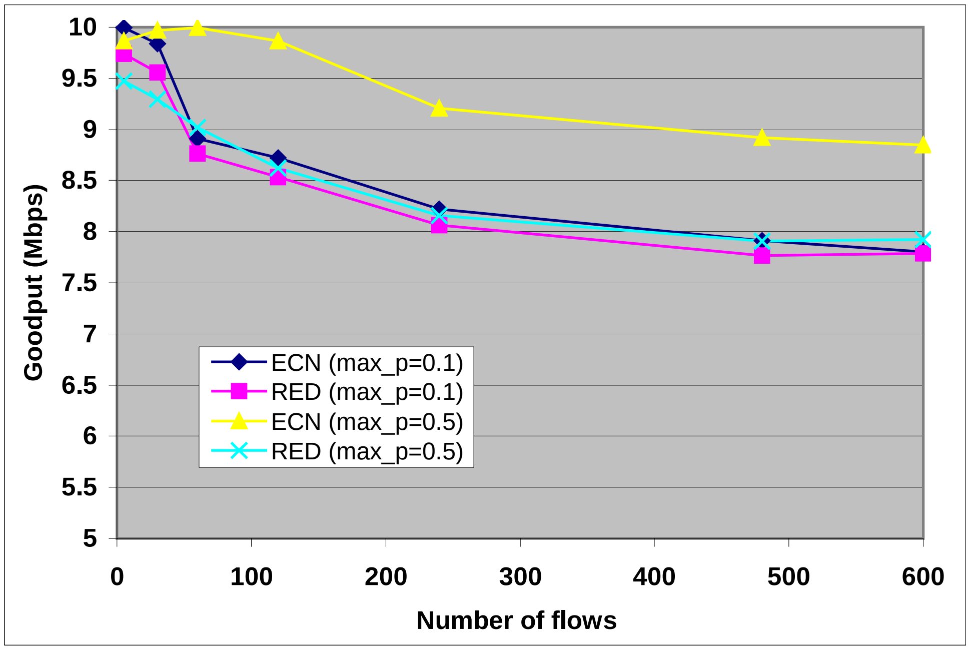


Figure 5.7: RED and ECN Goodput

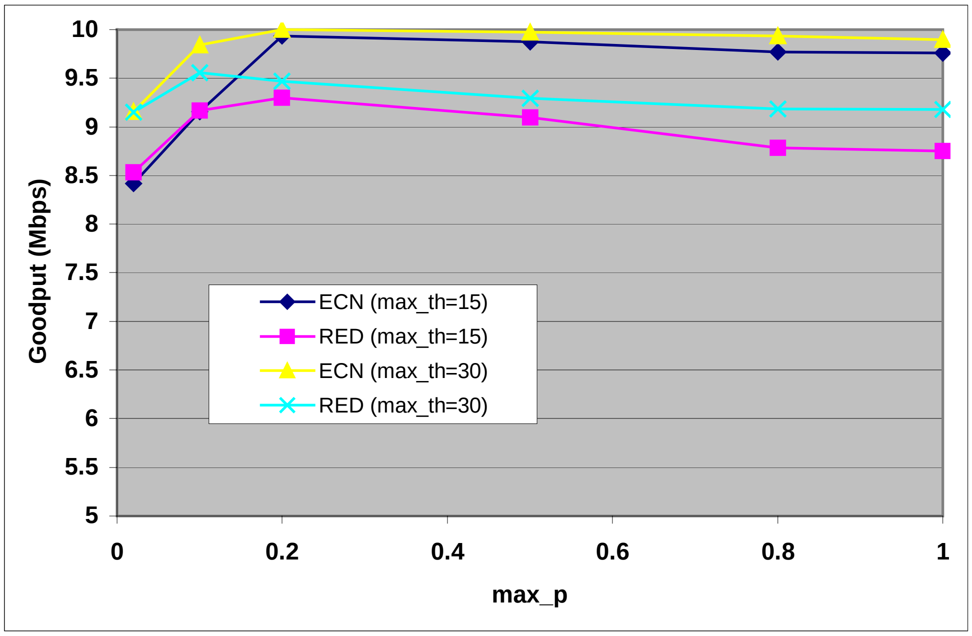


Figure 5.8: Goodput with 30 flows

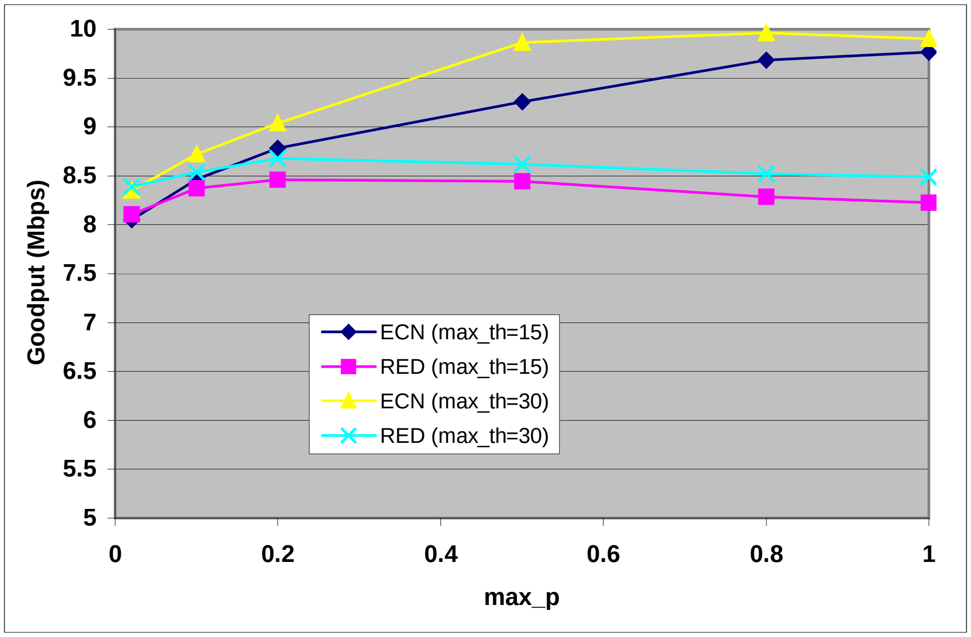


Figure 5.9: Goodput with 120 flows

In Figure 5.7, the average queue length threshold for triggering probabilistic drops/marks was set at 5, average queue length threshold for triggering forced drops was set at 30. Kinicki and Zheng (2001) noted that ECN provided higher goodput than RED. When the number of flows generating the demand is high, ECN performed better with a more aggressive maximum dropping/marking probability (max_p) setting.

In Figure 5.8 and 5.9, Kinicki and Zheng (2001) found out that for a fixed demand, as the number of flows increased, the performance of both RED and ECN decreased. This maybe caused by buffer contention at the router and flow lockout. However when there were many flows, increasing average queue length threshold for triggering forced drops (max_th) would improve ECN goodput. This may also be caused by the increased in max_p; therefore the number of packets marked increases. Hence TCP source will react faster to congestion.

5.4 Advantages and Disadvantages

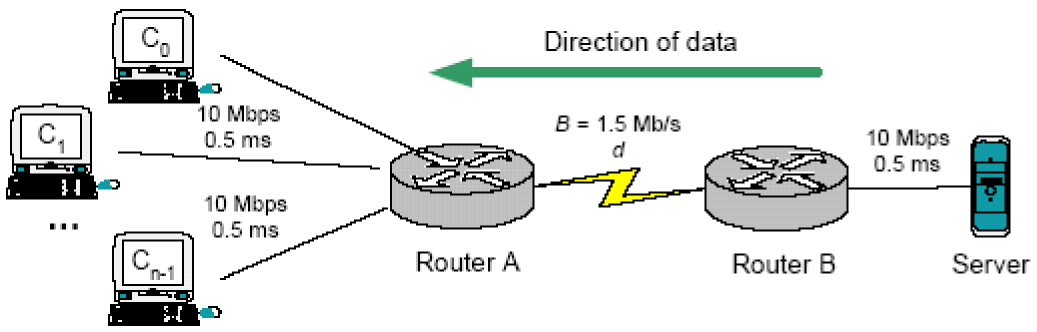


Figure 5.10: ECN test bed setup 2

One of the advantages of ECN can be found from the experiments conducted by Pentikousis and Badr (2003). The experiment was carried out using Network Simulator (NS2) for the topology as shown in Figure 5.9, all clients have the same TCP configuration and simultaneously initiate 20-MB downloads from the server. These large transfers allow TCP/ECN to display its best potential. The queue management mechanism used at router B was either Drop Tail (DT) or RED, as indicated in the results. If DT is used at router B, the only parameter set was the maximum buffer size. The tests were conducted with Queue Length (QL) ranging from 4 to 256 packets. The bandwidth B is the capacity of the bottleneck link along the connection path, which was set at 1.5 Mb/s. The minimum (minth) thresholds of the “RED region” were from 1 to 80. The terms “conservative” and “aggressive” (marked as “C” and “A” in the figures) were used to denote a maximum dropping probability of 2% and 10%, respectively.

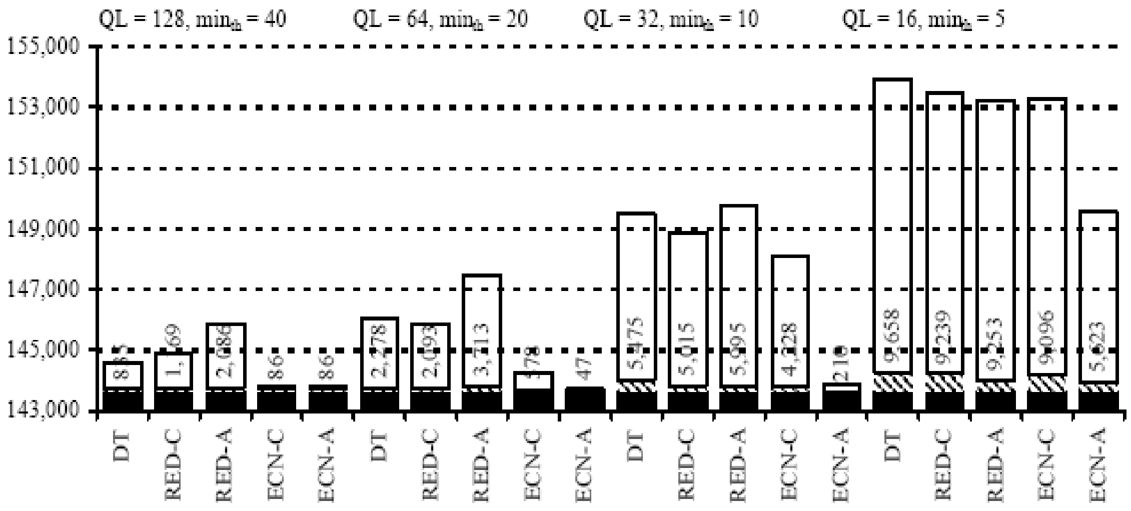


Figure 5.11. Total number of packets sent by the FTP server to all ten clients.

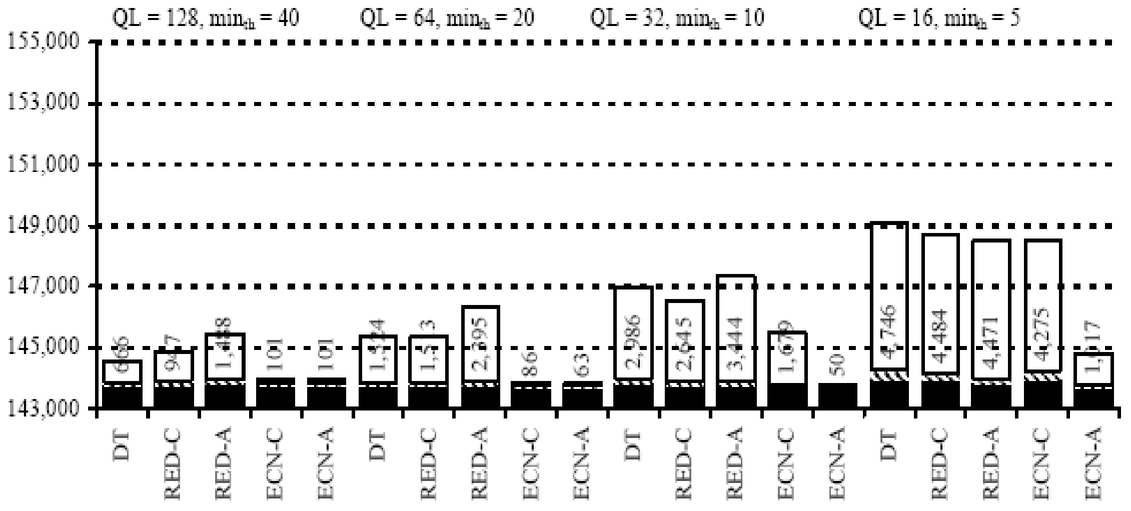


Figure 5.12: Total number of packets sent by the FTP server to all ten clients (with longer delays)

With reference to Figure 5.10 and 5.11, the number of packets dropped at the gateway was shown in white. The shaded area showed duplicate and retransmitted packets. The total application payload was shown in black. Pentikousis and Badr (2003) found out that DT, conservative and aggressive RED, and conservative ECN cause more packet drops as QL decreases. On the other hand, aggressive ECN drops fewer packets with QL = 64 than QL = 128. It was also remarkable that with QL = 32, aggressive ECN drops an extremely small fraction of packets when compared to the others. ECN in itself cannot prevent packet drops entirely, but it can reduce them dramatically. In general, TCP performs better when aggressive ECN is used: the sender sends fewer total segments, with, furthermore, a higher proportion of delivered-to-dropped packets.

Another advantage was that an all-ECN network could allow TCP to achieve the same goodput efficiency and level of packet losses as a DT-based network in which routers used buffers at least twice as large. This can be an important incentive for network operators, especially if they can enable ECN by simply upgrading the software of existing routers. With DT they would have to double the buffer space provided to the outgoing link in order to realize the same level of packet drops. An additional benefit from using ECN with half the buffer size was that the maximum possible queuing delay was halved as well

Introducing longer delays at the bottleneck link increases the delays in the TCP congestion control feedback loop, forcing TCP senders to become less aggressive. Meanwhile, the number of packets buffered in the network increases as well. Thus, for a given QL the increase in propagation delay caused fewer packets drops. This was illustrated in Figure 5.11, which also showed that when QL was 32 or 16, the relative advantage for a TCP/ECN sender was actually improving in terms of packet losses. Aggressive ECN was still the best choice in terms of packets dropped across all configurations.

Pentikousis and Badr (2003) also noted that many network operators charge their customers by the amount of traffic they carry through their routers. The revenues foregone by dropping a packet under RED could be significant in many cases: a packet dropped while the router is in the RED region is a packet that will not be charged to the customer under such pricing models. On the other hand, network costs could be reduced with aggressive ECN, which yields high goodput efficiency and fewer packet drops, and, hence, higher operating margins.

Secondly, the simulation demonstrated that an all-ECN network allows for a fairer allocation of resources by effectively mitigating lockouts. ECN can convey congestion information in a timely manner and can diminish packet drops, thus increasing the delivered-to-dropped packet ratio. It showed that aggressive ECN was more successful than conservative ECN in reducing packet drops, promoting a fairer environment, increasing network efficiency, and delivering higher and more even performance to individual connections. A rough rule of thumb, the aggressive ECN can deliver the same level of goodput efficiency and number of packet drops with only half the buffer space of DT at most. The TCP/ECN sender had a competitive advantage over ECN-unaware senders because it reacted faster to incipient congestion and can thus avoid unsuccessful segment transmissions.

However Floyd (1994) noted that there were two disadvantages or potential problems with ECN concerning non-compliant ECN connections and the potential loss of ECN messages in the network. A non-compliant TCP connection could set the ECN field to indicate that it was ECN-capable, and then ignore ECN notifications. Non-compliant connections could also ignore Source Quench messages. However, for a network that uses only packet drops for congestion notification, a non-compliant connection could also refrain from making appropriate window decreases in response to packet drops. A non-compliant connection interested in reliable delivery cannot ignore packet drops completely, but in the absence of monitoring and controls, a non-compliant connection could cause congestion problems in either an ECN or a non-ECN environment. A problem with ECN messages that had no counterpart with packet drops was that an ECN message (e.g., a Source Quench message, or a TCP ACK packet with the ECN field set) could be dropped by the network, and the congestion notification could fail to reach the end node. Therefore neither Source Quench messages nor the use of ECN fields in packet headers could guarantee that the TCP source would receive each notification of congestion. However, with RED gateways, the gateway does not rely on the source to respond to each congestion notification. The gateway will continue to set the ECN field in randomly chosen packets as long as congestion persists at the gateway. In addition, a gateway implementing RED algorithms is particularly unlikely to drop a high fraction of packets.

Chapter 6

Simulation Methodology

6.1 Selection of Simulation Tools

For the design and implementation of communication protocols and algorithms, the use of simulation tools means a substantial productivity increase. Using simulations, protocols do not need to be implemented in explicit detail. In most cases, simulation of one or more protocol layer provides significant and sufficient results.

The deployment and the debugging of wireless applications on a real network can be rather difficult if large networks are considered. Therefore simulation is an important tool that can often help to improve or validate protocols. All simulators provide a complete toolkit to the developers that enable metrics collection and various wireless network diagnostics. The main characteristics that divide them are mainly; accuracy, speed, ease of use, and monetary expense.

Ubik and Klaban (2003) noted that simulation should create a model of a system, which is used to explain system behavior and to see how the system performs under varying conditions in order to design the system with desirable performance characteristics.

Simulation should also represent the system on a smaller scale for easier study when compared to the full-scale physical system. By using simulation, the researcher should be allow to study a system in well-defined and well-known conditions, repeatability if necessary in order to understand events.

Cavin *et al* (2002) noted that there were several popular simulators, such as OPNET Modeler, Network Simulator 2 (NS2) or GloMoSim available for network simulations. Each of them provided advanced simulation environments to test and debug any kind of networking protocols, including wireless applications. However for the simulations to be helpful, it was necessary that the simulated behaviors match as closely as possible the physical situation. This latter requirement implied to address several issues. Firstly, the application was likely to rely on existing components, such as collision detection module, radio propagation or MAC protocols. The correct modeling of these components in the simulator was crucial. Each algorithm that was being evaluated was modeled in detail, but the interaction with the other layers was often not taken into account. Secondly, the simulation parameters and its environment (mobility schemes, power ranges, connectivity) must be realistic. Incorrect initial conditions, for example may lead to unexpected results not exploitable in a real network.

OPNET Modeler is a network simulator developed by OPNET. It can simulate all kinds of wired networks, and implement 802.11 compliant MAC layer. Although OPNET is designed for companies to analysis or restructure their network, it is still possible to implement specific algorithm by reusing a lot of existing components. Most part of the deployment is made through a hierarchical graphic user interface.

NS2 is an open-source simulation tool from Lawrence Berkeley Laboratory that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing. LAN routing and broadcasts are part of routing algorithms. Queuing algorithms include fair queuing, deficit round-robin and FIFO.

GloMoSim is a scalable simulation environment for wireless and wired networks systems developed initially at UCLA Computing Laboratory. It is designed using the parallel discrete-event simulation capability provided by a C-based parallel simulation language. GloMoSim currently supports protocols for purely wireless networks. It is build using a layered approach. Standard Application Programming Interface (API) is used between the different layers. This allows the rapid integration of models developed at different layers by users.

NS2 was chosen for this project, as it is an event-driven network simulator, which is popular with the networking research community. It includes numerous models of common Internet protocols including several newer protocols, such as reliable multicast and TCP selective acknowledgement. Network animator, Nam, also provides packet-level animation and protocol specific graph for design and debugging of network protocols. Additionally, different levels of configuration are present in NS2 due to its open source nature, including the capability of creating custom applications and protocols as well as modifying several parameters at different layers.

The freeware nature of NS2 is also attractive compared to the need to enter into an OPNET Modeler license agreement and associated direct costs. On top of that, NS2's code source is split between C++ for its core engine and OTcl, an object oriented version of TCL for configuration and simulation scripts. The combination of the two languages offers an interesting compromise between performance and ease of use. A highly dynamic newsgroups and source codes are also available on the web to provide assistance to most of the problems encounter while using NS2.

6.2 Introduction to Network Simulator 2 (NS2)

Ns2 is an event driven, object oriented network simulator which support networking research (traffic studies, protocol design and comparison) and education. The wide range of platform support provided in NS includes Unix (FreeBSD, Linux, SunOS and Solaris) and Windows (Cygwin for win9x/2000/XP). NS2 provides a collaborative environment, as its software is freely distributed and open source. Ns2 has users span across 50 countries with about 300 posts to its mailing list every month. NS2 also has periodical release with over 100 test suites and examples. Users are able to share code, protocols and models. This allows easy comparison of similar protocols, which in turn increase the reliability of the results. A stability validation is also available at its website (http://www.isi.edu/nsnam/ns/ns-tests.html).

NS2 is written in C++ and Otcl to separate the control and data path implementations. The simulator supports a class hierarchy in C++ and a corresponding hierarchy within the Otcl interpreter. NS2 uses two languages due to different tasks having different requirements and simulation of protocols requires efficient manipulation of bytes and packet headers making the run-time speed very important. On top of that, there is a need to vary some parameters in network studies and to quickly examine a number of scenarios. Therefore the time taken to change the model and run it again is of great important.

C++ is used in NS2 for detailed protocol implementation and in general for cases where every packet of a flow needs to be processed. For example, if you want to implement a new queuing discipline, then C++ is the language of choice. On the other hand, Otcl is suitable for configuration and setup. Otcl runs quite slowly, but it can be changed very quickly making the construction of simulations easier. In NS2, the compiled C++ objects can be made available to the Otcl interpreter. In this way, the ready-made C++ objects can be controlled from the OTcl level.

6.2.1 Features

To calculate the results from the simulations, data can be collected using tracing objects. Tracing objects are designed to record packet arrival time at which they are located. The traces also enable recording of packets whenever an event such as packet drop or arrival occurs in a queue or a link.

```
set trace_file [open out.tr w]
$ns trace-all $trace_file
$ns flush-trace
close $trace_file
```

All events from the simulation can be recorded to a file with the above commands. It would generate a trace file called "out.tr" that can be used for simulation analysis. Figure 6.1 shows the trace format and example trace data from "out.tr".

Event	Time	From Node	To node	Pkt Type	Pkt Size	Flags	Fid	Src Addr	Dst Addr	Seq Num	Pkt Id

```
+ 1.64375 0 2 cbr 310 ------- 0 0.0 3.1 225 201
- 1.64375 0 2 cbr 310 ------- 0 0.0 3.1 225 201
r 1.64471 2 1 cbr 310 ------- 1 3.0 1.0 195 201
r 1.64566 2 0 ack 40 ------- 2 3.2 0.1 82 602
+ 1.64566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
- 1.64566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
```

+ : enqueue	(at queue)	src_addr : node.port
- : dequeue	(at queue)	dst_addr : node.port
d : drop	(at queue)	
r : receive	(at to_node)	

Figure 6.1: Trace Format Example

Each trace line starts with an event (+, -, d, r) descriptor followed by the simulation time (in seconds) of that event, and from and to node, which identify the link on which the event occurred. The next information in the line before flags (appeared as "------" since no flag is set) is packet type and size (in Bytes). Currently, NS2 implements only the Explicit Congestion Notification (ECN) bit, and the remaining bits are not used. The next field is flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script. Even though fid field may not use in a simulation, users can use this field for analysis purposes. The fid field is also used when specifying stream color for the NAM display. The next two fields are source and destination address in forms of "node**.**port". The next field shows the network layer protocol's packet sequence number. Even though User Datagram Protocol (UDP) implementations do not use sequence number, NS2 keeps track of UDP packet sequence number for analysis purposes. The last field shows the unique id of the packet.

When the simulation topology is relatively simple and the number of sources is limited, an effective method would be to trace all events from the simulation to a specific file and then calculating the desired quantities from this file by using perl or awk and Matlab. However, with complex topologies and many sources, this method of collecting data can become slow. The trace files may also consume a large amount of disk space.

A plotting program called the ‘xgraph’ is available in the NS2 package, which can be used to create graphic representations of simulation results. As shown in the commands below, the ‘xgraph’ can be call from within the finish procedure.

```
proc finish {} {
        global trace_file
        #Close the output files
        close $trace_file
        #Call xgraph to display the results
        exec xgraph out.tr -geometry 800x400 &
        exit 0
}
```

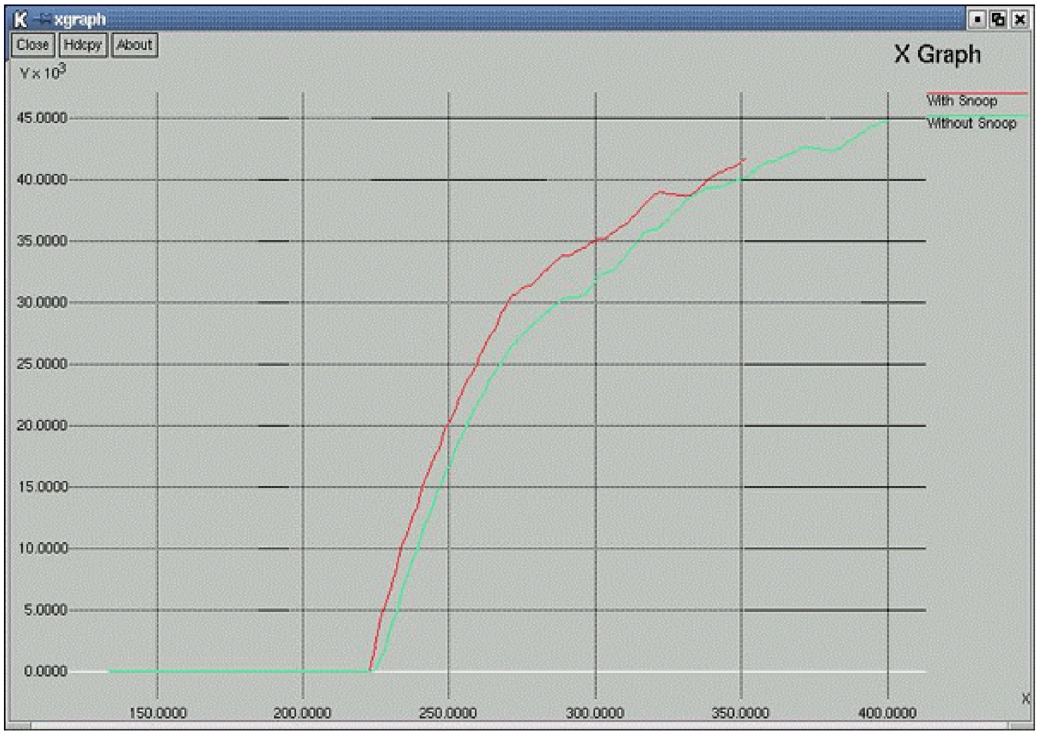


Figure 6.2: X Graph

6.2.2 Network Animator (NAM)

NAM is a network animator tool that works with NS2. It takes in a NAM trace file generated by NS2 during a simulation run of the network and animates the process.

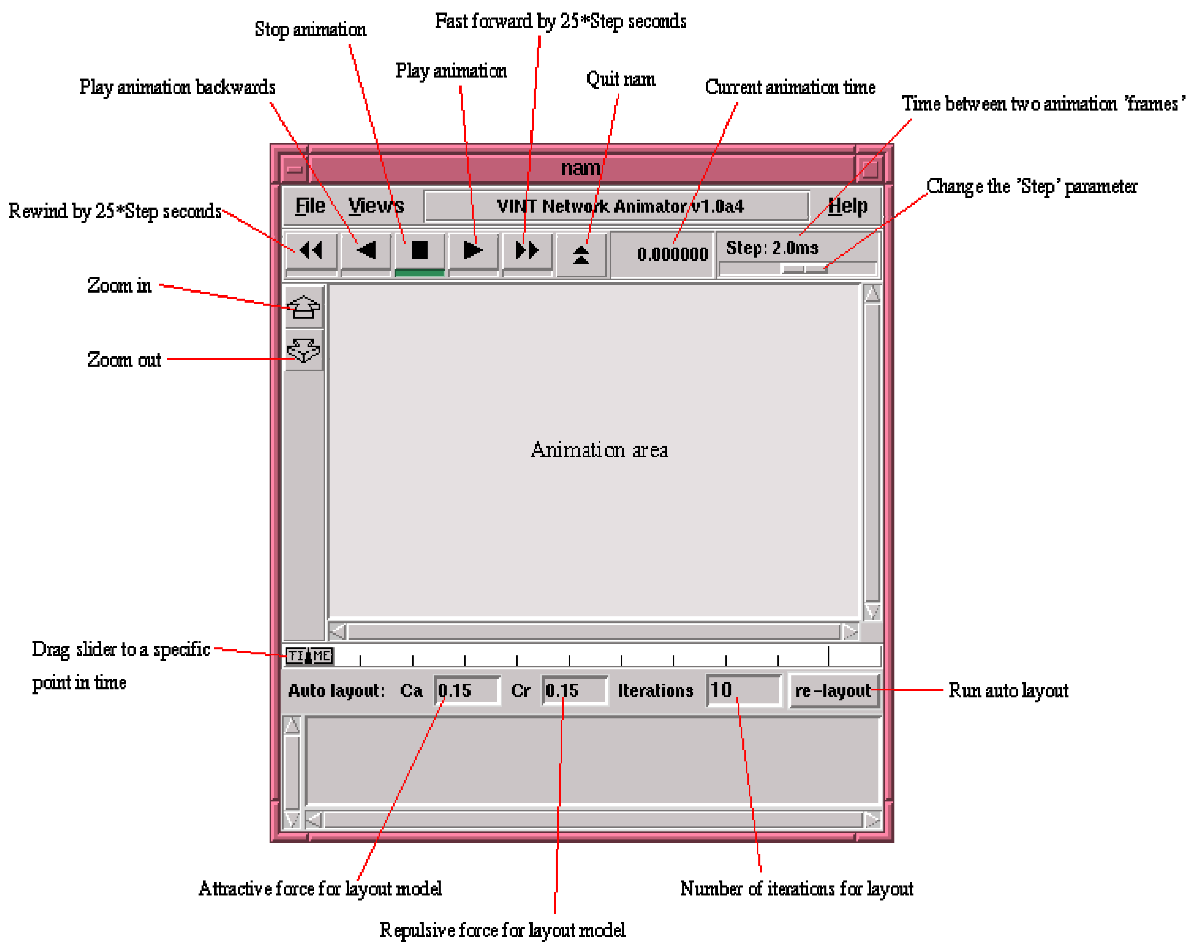

Stop animation
Fast forward by 25*Step seconds
Play animation backwards
Play animation
Quit nam
Current animation time
Time between two animation 'frames'
nam
Rewind by 25*Step seconds
File
Views
VINT Network Animator v1.0a4
Help
Change the 'Step' parameter
0.000000
Step: 2.0ms
Zoom in
Zoom out
Animation area
Drag slider to a specific point in time
TIME
Auto layout:
Ca
0.15
Cr
0.15
Iterations
10
re-layout
Run auto layout
Attractive force for layout model
Number of iterations for layout
Repulsive force for layout model

Figure 6.3: NAM display

NAM provides a visual view of the simulation, which includes the overall topology (nodes and links). It can also monitor individual nodes, links and the status (queued and dropped) of packets in transmission. It does allow simple editing of the topology layout, and it also has layout algorithm that will automatically try and layout the nodes to maximize the distances between them. Adding colour to the packets for a particular flow of traffic is possible, but this has to be done through code in the NS2 simulation.

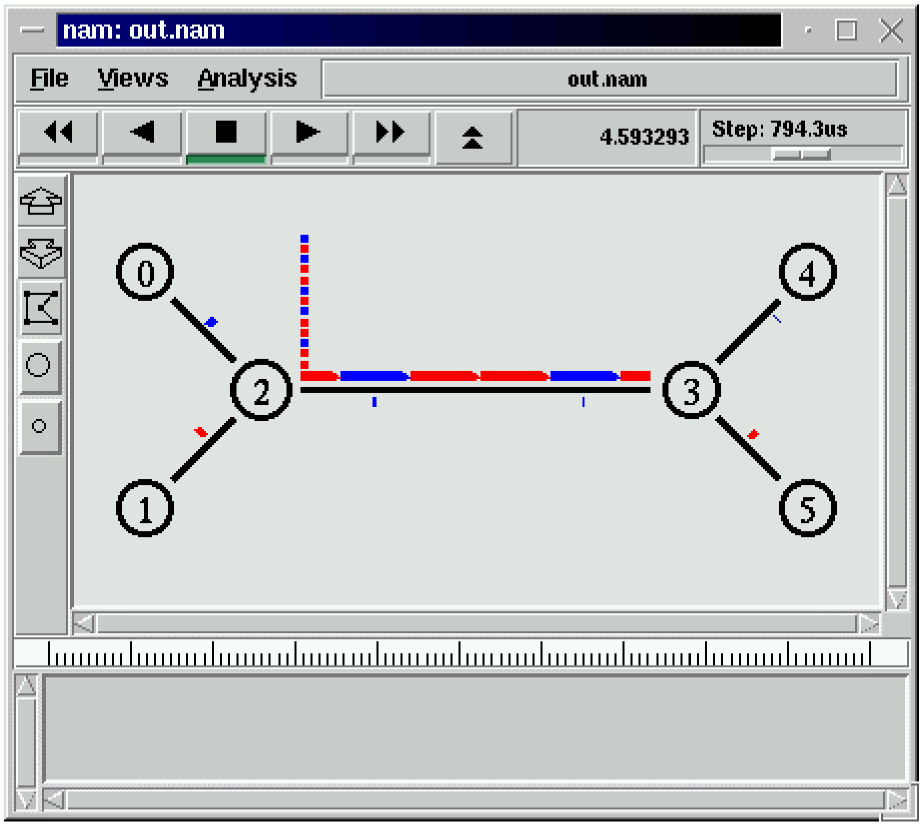


Figure 6.4: NAM display for a simple communication scenario

6.2.3 Basic Command

The following steps are used to generate simulation script in NS2:

- Create an object of the ns2 simulator.
- Create objects for network nodes, links and queues attached to links and specify their parameters, thus creating the network topology.
- Create objects for TCP sender and TCP receiver and specify maximum window size.
- Create objects for sending application and receiving application and attach them to objects for TCP sender and TCP receiver, respectively.
- Schedule events, such as start and end times of data streams and when the simulation should stop.
- Start simulation.

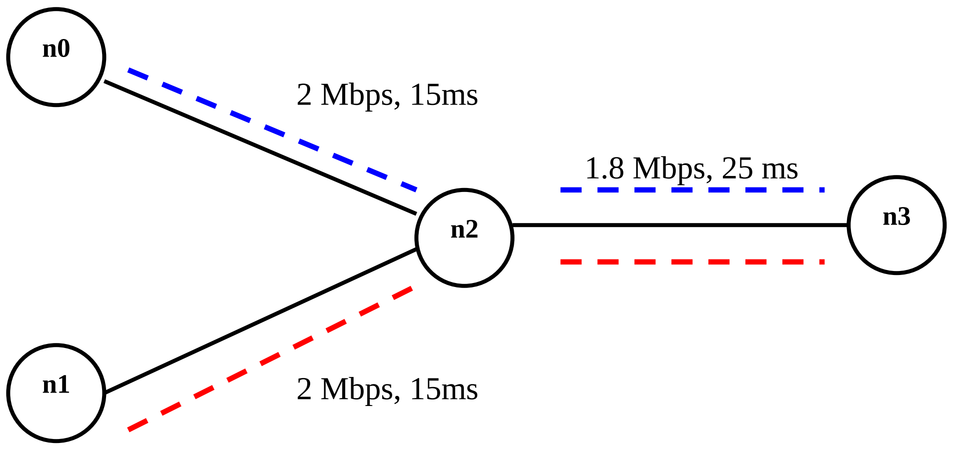


Figure 6.5: A Simple Network Topology and Simulation Scenario

The network consists of 4 nodes (n0, n1, n2, n3) as shown in above Figure 6.5. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 15 ms of delay. The duplex link between n2 and n3 has 1.8 Mbps of bandwidth and 25 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "udp" agent that is attached to n0 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "tcp" agent is attached to n1, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate 1 KByte packets at the rate of 1 Mbps. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.5 sec and stop at 3.0 sec. The Otcl script for the simulation topology in Figure 6.5 is as follow:

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set tracefd  [open simple.tr w]
$ns trace-all $tracefd
set namtracefd [open simple.nam w]
$ns namtrace-all $namtracefd

#Define a 'finish' procedure
proc finish {} {
        global ns tracefd namtracefd
        $ns flush-trace
        #Close the NAM trace file
        close $tracefd
        close $namtracefd
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}
```

```
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 15ms DropTail
$ns duplex-link $n1 $n2 2Mb 15ms DropTail
$ns duplex-link $n2 $n3 1.8Mb 25ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

```
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.5 "$ftp start"
$ns at 3.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

In order to start run the above Otcl script, save it as **myexample.tcl** in a prefer directory. Make sure that the current path is pointing to the prefer directory where **myexample.tcl** is saved. Start NS2 with the command **ns myexample.tcl** at the command prompt. The following is the explanation of the above Otcl script.

set *ns* [new Simulator]:

This function will generates an NS simulator object instance, and assigns it to variable ns.

***$ns* color *fid color* :**

This function will set color of the packets for a flow specified by the flow id (fid). The member function of "Simulator" object is for the NAM display, and has no effect on the actual simulation.

***$ns* namtrace-all *file-descriptor* :**

This member function tells the simulator to record simulation traces in NAM input format. It also gives the file name that the trace will be written to later by the command ***$ns*** flush-trace. Similarly, the member function trace-all is for recording the simulation trace in a general format.

proc *finish* {}:

This function is called after this simulation is over by the command ***$ns*** at 5.0 **"*finish*"**. In this function, post-simulation processes are specified.

set *n0* [*$ns* node]:

The member function node creates a node. A node in NS is compound object made of address and port classifiers. Users can create a node by separately creating an address and a port classifier objects and connecting them together.

***$ns* duplex-link *node1 node2 bandwidth delay queue-type* :**

This function creates two simplex links of specified bandwidth and delay, and connects the two specified nodes. In NS, the output queue of a node is implemented as a part of a link; therefore users should specify the queue-type when creating links. In the above simulation script, DropTail queue is used. If the reader wants to use a RED queue, simply replace the word DropTail with RED. Like a node, a link is a compound object, and users can create its sub-objects and connect them and the nodes.

***$ns* queue-limit *node1 node2 number* :**

This line sets the queue limit of the two simplex links that connect node1 and node2 to the number specified.

***$ns* duplex-link-op *node1 node2* :**

The next couple of lines are used for the NAM display.

set *tcp* [new *Agent/TCP*]:

This line shows how to create a TCP agent. But in general, users can create any agent or traffic sources in this way.

***$ns* attach-agent *node agent* :**

This member function attaches an agent object created to a node object. This function will call the attach member function of specified node, which attaches the given agent to itself.

***$ns* connect *agent1 agent2* :**

When two agents that will communicate with each other are created, a logical network connection is established between them. This line establishes a network connection by setting the destination address to one another's network and port address pair.

***$ns* at *time "string"* :**

This member function of a simulator object will schedule the execution of the specified string at given simulation time. The scheduler will call a ‘start’ member function of the traffic source object, which will start to transmit data. In NS2, traffic source does not transmit actual data, but it signal the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.

***$ns* run :**

After all network configurations, scheduling and post-simulation procedure specifications are done; this line will execute the simulation.

Chapter 7

Installation of Linux and Network Simulator 2 (NS2)

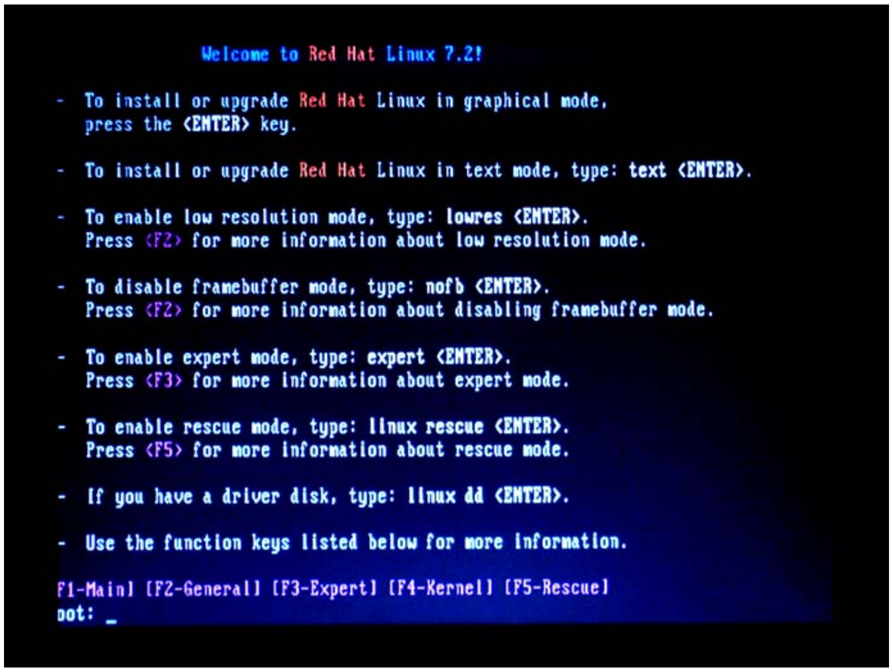
7.1 Red Hat Linux Installation Process

Before the start of the installation process, the computer must have enough un-partitioned disk space for the installation of Red Hat Linux. Or there are one or more partitions that can be deleted, thereby freeing up enough disk space to install Red Hat Linux 7.2

A workstation installation, choosing to install GNOME or KDE, requires at least 1.5 GB of free space. Choosing both GNOME and KDE requires at least 1.8 GB of free disk space. A server installation requires 1 GB for a minimal installation without X (the graphical environment), at least 1.3 GB of free space if all components (package groups) other than X are installed, and at least 1.8 GB to install all packages including GNOME and KDE.

A laptop installation, when you choose to install GNOME or KDE, requires at least 1.5 GB of free space. If you choose both GNOME and KDE, you will need at least 1.8 GB of free disk space. A custom installation requires 350 MB for a minimal installation and at least 3.5 GB of free space if every package is selected. For more information about system requirements, users may want to visit Red Hat Linux 7.2: The Official Red Hat Linux x86 Installation Guide at http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/install-guide/index.html

To install Linux, turn on computer and insert disk 1 of Red Hat Linux 7.2. At the boot prompt, press the 'Enter' key. After the basic Linux Kernel is loaded, anaconda (Red Hat installation program) should load. An easy-to- use graphical user interface will guide the user through the install process.



```
              Welcome to Red Hat Linux 7.2!

 -  To install or upgrade Red Hat Linux in graphical mode,
    press the <ENTER> key.

 -  To install or upgrade Red Hat Linux in text mode, type: text <ENTER>.

 -  To enable low resolution mode, type: lowres <ENTER>.
    Press <F2> for more information about low resolution mode.

 -  To disable framebuffer mode, type: nofb <ENTER>.
    Press <F2> for more information about disabling framebuffer mode.

 -  To enable expert mode, type: expert <ENTER>.
    Press <F3> for more information about expert mode.

 -  To enable rescue mode, type: linux rescue <ENTER>.
    Press <F5> for more information about rescue mode.

 -  If you have a driver disk, type: linux dd <ENTER>.

 -  Use the function keys listed below for more information.

F1-Main] [F2-General] [F3-Expert] [F4-Kernel] [F5-Rescue]
oot: _
```

Figure 7.1: Red Hat installation process (Boot Prompt)

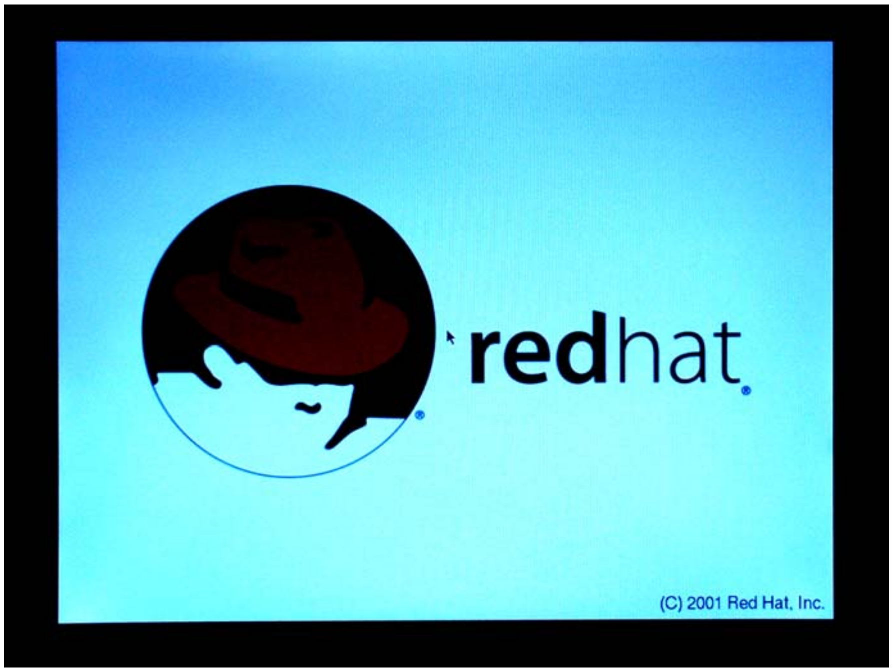
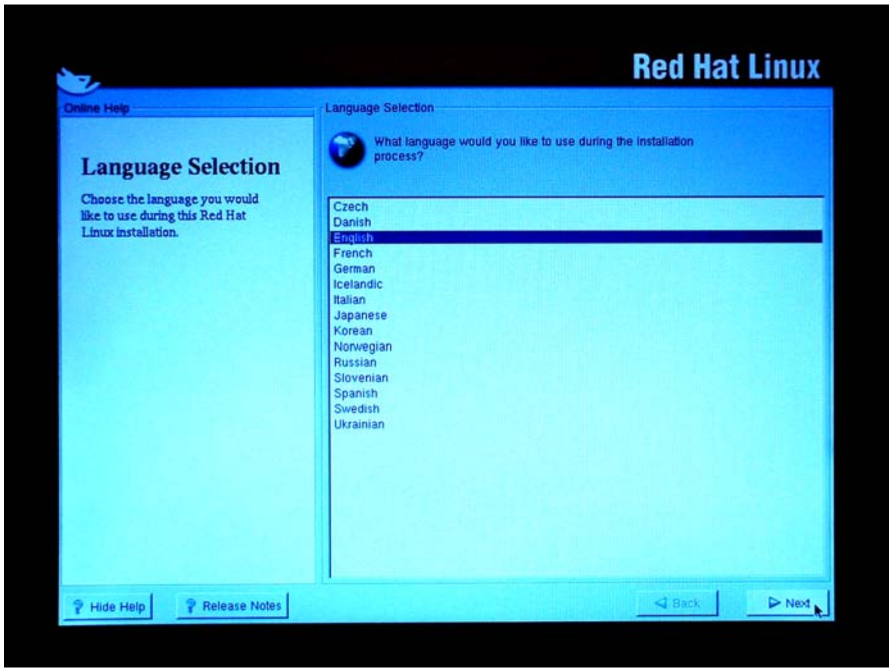


Figure 7.2: Red Hat installation process (Anaconda)



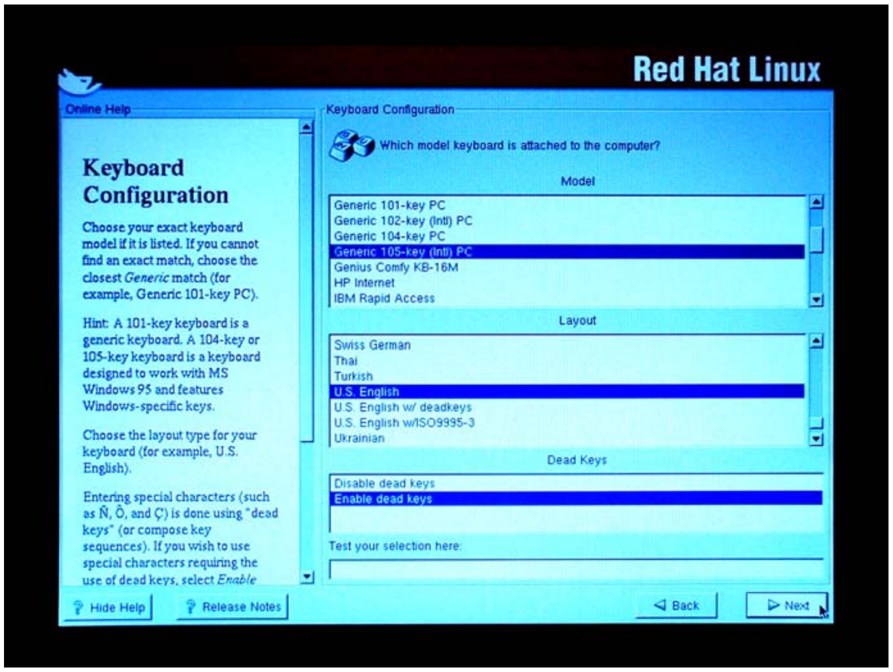
Language Selection

Choose the language you would like to use during this Red Hat Linux installation.

What language would you like to use during the installation process?

Czech
Danish
English
French
German
Icelandic
Italian
Japanese
Korean
Norwegian
Russian
Slovenian
Spanish
Swedish
Ukrainian

Figure 7.3: Red Hat installation process (Language Selection)



Keyboard Configuration

Choose your exact keyboard model if it is listed. If you cannot find an exact match, choose the closest *Generic* match (for example, Generic 101-key PC).

Hint: A 101-key keyboard is a generic keyboard. A 104-key or 105-key keyboard is a keyboard designed to work with MS Windows 95 and features Windows-specific keys.

Choose the layout type for your keyboard (for example, U.S. English).

Entering special characters (such as Ñ, Ô, and Ç) is done using "dead keys" (or compose key sequences). If you wish to use special characters requiring the use of dead keys, select *Enable*

Which model keyboard is attached to the computer?

Model

- Generic 101-key PC
- Generic 102-key (Intl) PC
- Generic 104-key PC
- Generic 105-key (Intl) PC [✓]
- Genius Comfy KB-16M
- HP Internet
- IBM Rapid Access

Layout

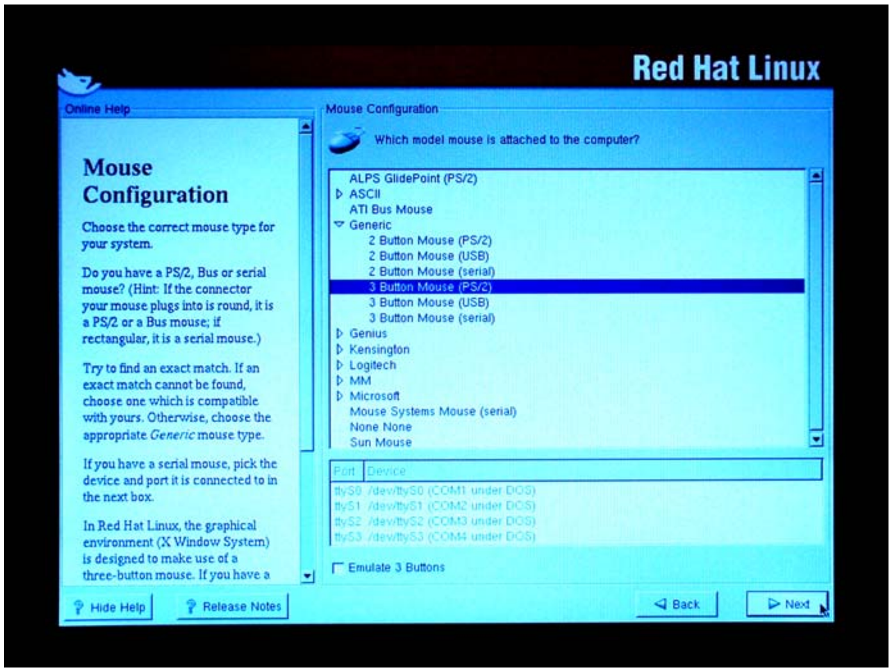
- Swiss German
- Thai
- Turkish
- U.S. English [✓]
- U.S. English w/ deadkeys
- U.S. English w/ISO9995-3
- Ukrainian

Dead Keys

- Disable dead keys
- Enable dead keys [✓]

Test your selection here:

Figure 7.4: Red Hat installation process (Keyboard Configuration)



Mouse Configuration

Choose the correct mouse type for your system.

Do you have a PS/2, Bus or serial mouse? (Hint: If the connector your mouse plugs into is round, it is a PS/2 or a Bus mouse; if rectangular, it is a serial mouse.)

Try to find an exact match. If an exact match cannot be found, choose one which is compatible with yours. Otherwise, choose the appropriate *Generic* mouse type.

If you have a serial mouse, pick the device and port it is connected to in the next box.

In Red Hat Linux, the graphical environment (X Window System) is designed to make use of a three-button mouse. If you have a

Which model mouse is attached to the computer?

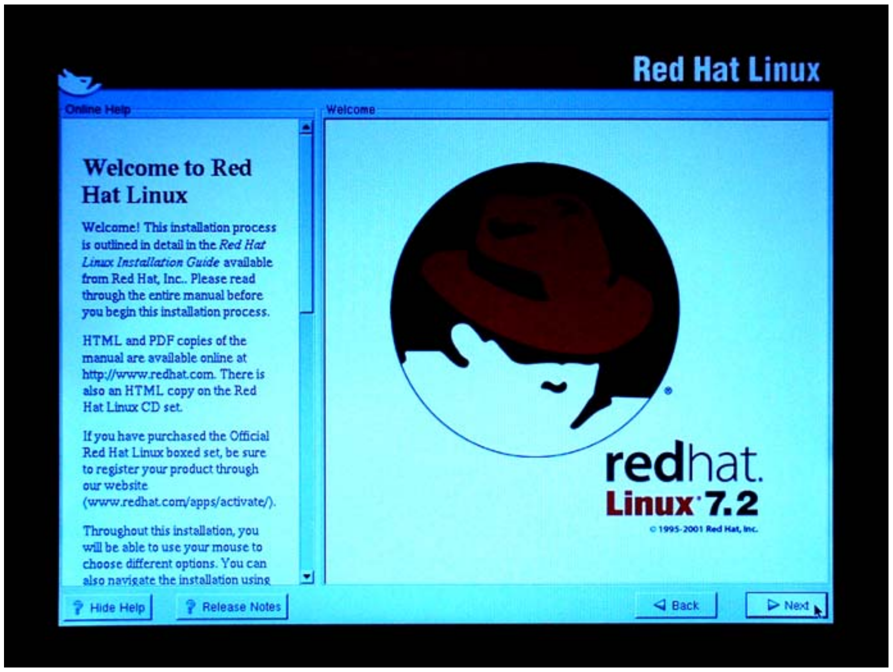
```
   ALPS GlidePoint (PS/2)
▷ ASCII
   ATI Bus Mouse
▽ Generic
      2 Button Mouse (PS/2)
      2 Button Mouse (USB)
      2 Button Mouse (serial)
      3 Button Mouse (PS/2)
      3 Button Mouse (USB)
      3 Button Mouse (serial)
▷ Genius
▷ Kensington
▷ Logitech
▷ MM
▷ Microsoft
   Mouse Systems Mouse (serial)
   None None
   Sun Mouse
```

Selected: 3 Button Mouse (PS/2) [✓]

Port	Device
ttyS0	/dev/ttyS0 (COM1 under DOS)
ttyS1	/dev/ttyS1 (COM2 under DOS)
ttyS2	/dev/ttyS2 (COM3 under DOS)
ttyS3	/dev/ttyS3 (COM4 under DOS)

☐ Emulate 3 Buttons

Figure 7.5: Red Hat installation process (Mouse Configuration)



Welcome to Red Hat Linux

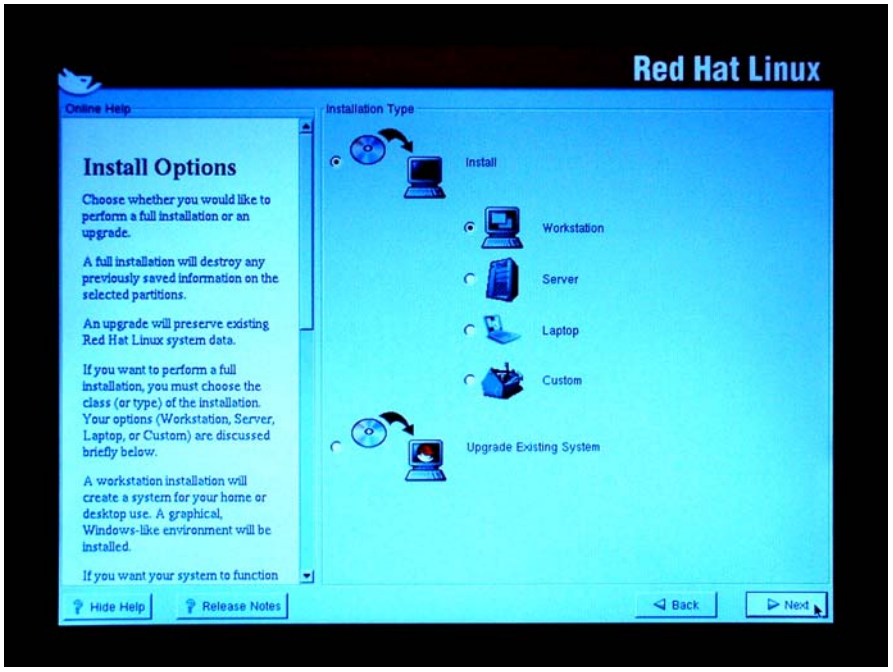
Welcome! This installation process is outlined in detail in the *Red Hat Linux Installation Guide* available from Red Hat, Inc.. Please read through the entire manual before you begin this installation process.

HTML and PDF copies of the manual are available online at http://www.redhat.com. There is also an HTML copy on the Red Hat Linux CD set.

If you have purchased the Official Red Hat Linux boxed set, be sure to register your product through our website (www.redhat.com/apps/activate/).

Throughout this installation, you will be able to use your mouse to choose different options. You can also navigate the installation using

Figure 7.6: Red Hat installation process (Welcome to Red Hat Linux)



Install Options

Choose whether you would like to perform a full installation or an upgrade.

A full installation will destroy any previously saved information on the selected partitions.

An upgrade will preserve existing Red Hat Linux system data.

If you want to perform a full installation, you must choose the class (or type) of the installation. Your options (Workstation, Server, Laptop, or Custom) are discussed briefly below.

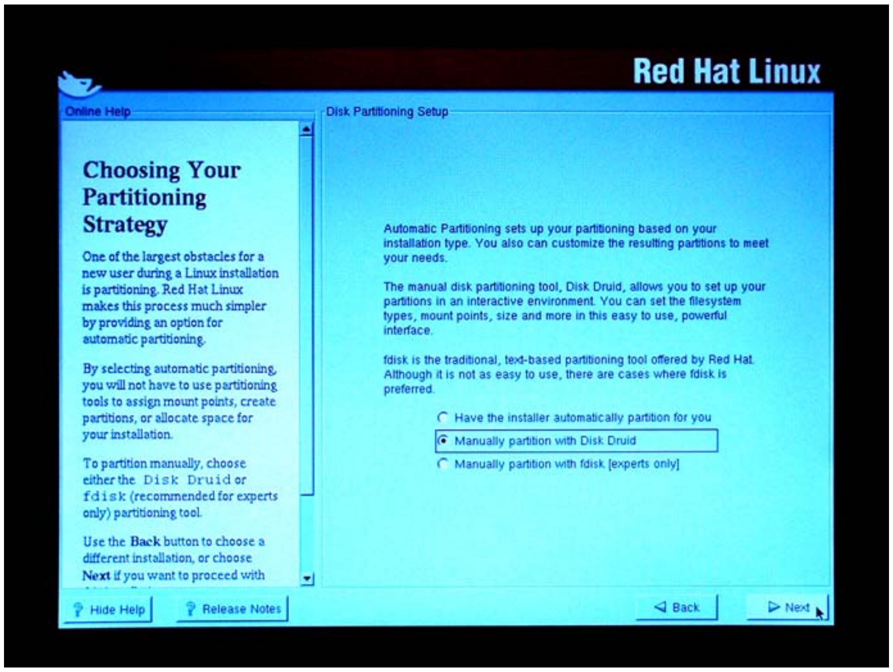
A workstation installation will create a system for your home or desktop use. A graphical, Windows-like environment will be installed.

If you want your system to function

[✓ Install]
- [✓ Workstation]
- Server
- Laptop
- Custom

Upgrade Existing System

Figure 7.7: Red Hat installation process (Install Options)



Choosing Your Partitioning Strategy

One of the largest obstacles for a new user during a Linux installation is partitioning. Red Hat Linux makes this process much simpler by providing an option for automatic partitioning.

By selecting automatic partitioning, you will not have to use partitioning tools to assign mount points, create partitions, or allocate space for your installation.

To partition manually, choose either the `Disk Druid` or `fdisk` (recommended for experts only) partitioning tool.

Use the **Back** button to choose a different installation, or choose **Next** if you want to proceed with

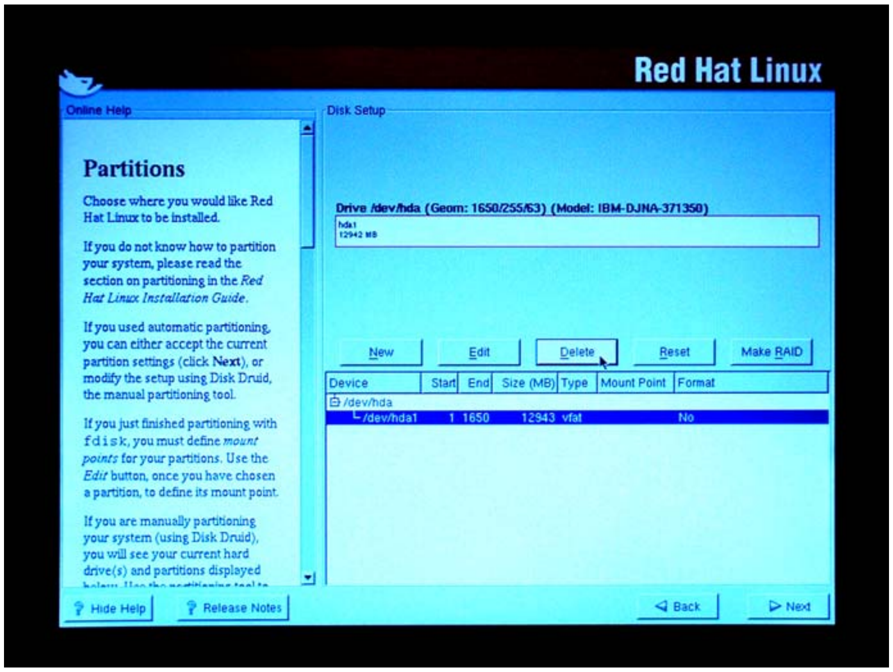
Automatic Partitioning sets up your partitioning based on your installation type. You also can customize the resulting partitions to meet your needs.

The manual disk partitioning tool, Disk Druid, allows you to set up your partitions in an interactive environment. You can set the filesystem types, mount points, size and more in this easy to use, powerful interface.

fdisk is the traditional, text-based partitioning tool offered by Red Hat. Although it is not as easy to use, there are cases where fdisk is preferred.

- Have the installer automatically partition for you
- [✓ Manually partition with Disk Druid]
- Manually partition with fdisk [experts only]

Figure 7.8: Red Hat installation process (Partition part 1)



Partitions

Choose where you would like Red Hat Linux to be installed.

If you do not know how to partition your system, please read the section on partitioning in the *Red Hat Linux Installation Guide*.

If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using Disk Druid, the manual partitioning tool.

If you just finished partitioning with `fdisk`, you must define *mount points* for your partitions. Use the *Edit* button, once you have chosen a partition, to define its mount point.

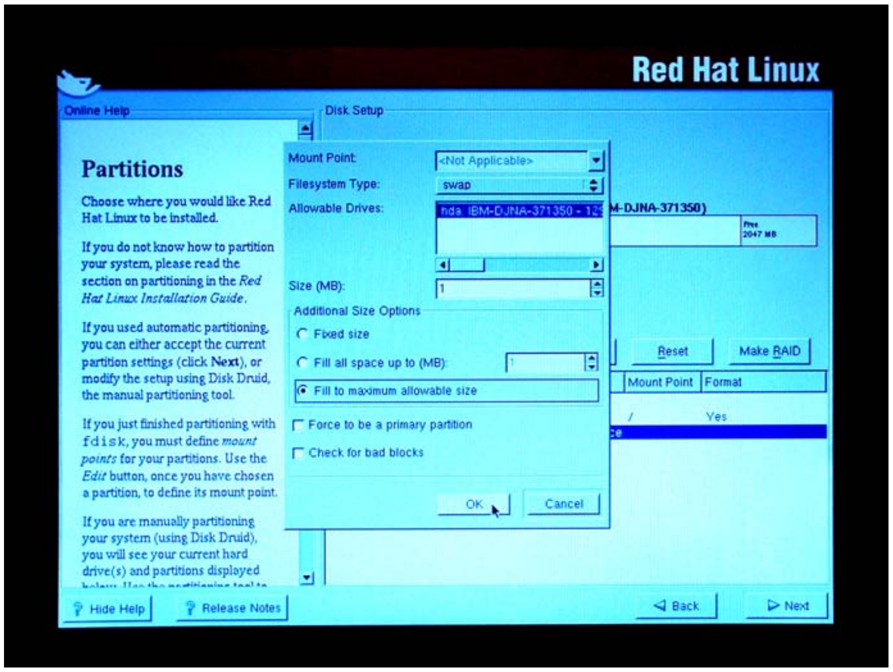
If you are manually partitioning your system (using Disk Druid), you will see your current hard drive(s) and partitions displayed

Drive /dev/hda (Geom: 1650/255/63) (Model: IBM-DJNA-371350)

hda1
12942 MB

Device	Start	End	Size (MB)	Type	Mount Point	Format
/dev/hda						
/dev/hda1	1	1650	12943	vfat		No

Figure 7.9: Red Hat installation process (Partition part 2)



Partitions

Choose where you would like Red Hat Linux to be installed.

If you do not know how to partition your system, please read the section on partitioning in the *Red Hat Linux Installation Guide*.

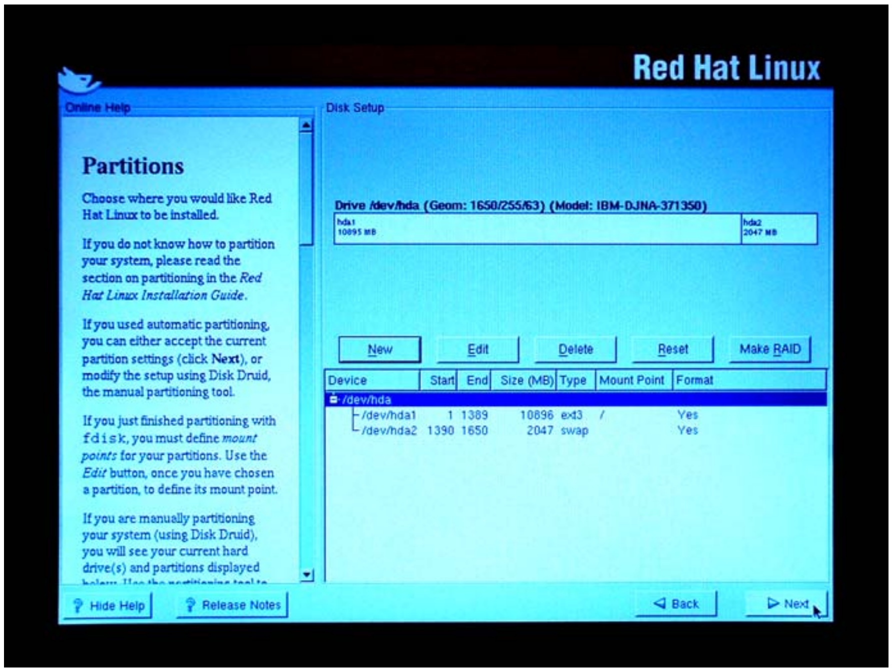
If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using Disk Druid, the manual partitioning tool.

If you just finished partitioning with `fdisk`, you must define *mount points* for your partitions. Use the *Edit* button, once you have chosen a partition, to define its mount point.

If you are manually partitioning your system (using Disk Druid), you will see your current hard drive(s) and partitions displayed

Mount Point: <Not Applicable>
Filesystem Type: swap
Allowable Drives: hda IBM-DJNA-371350 - 12
Size (MB): 1
Additional Size Options
( ) Fixed size
( ) Fill all space up to (MB): 1
(•) Fill to maximum allowable size
[ ] Force to be a primary partition
[ ] Check for bad blocks

Figure 7.10: Red Hat installation process (Partition part 3)



Partitions

Choose where you would like Red Hat Linux to be installed.

If you do not know how to partition your system, please read the section on partitioning in the *Red Hat Linux Installation Guide*.

If you used automatic partitioning, you can either accept the current partition settings (click **Next**), or modify the setup using Disk Druid, the manual partitioning tool.

If you just finished partitioning with `fdisk`, you must define *mount points* for your partitions. Use the *Edit* button, once you have chosen a partition, to define its mount point.

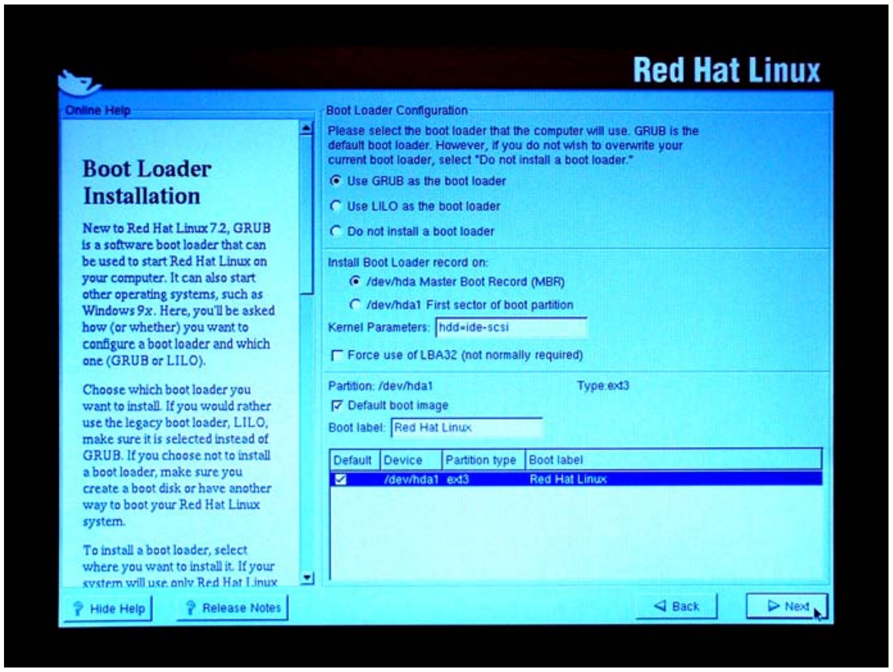
If you are manually partitioning your system (using Disk Druid), you will see your current hard drive(s) and partitions displayed

Drive /dev/hda (Geom: 1650/255/63) (Model: IBM-DJNA-371350)

hda1 10895 MB | hda2 2047 MB

Device	Start	End	Size (MB)	Type	Mount Point	Format
/dev/hda						
/dev/hda1	1	1389	10896	ext3	/	Yes
/dev/hda2	1390	1650	2047	swap		Yes

Figure 7.11: Red Hat installation process (Partition part 4)



Boot Loader Installation

New to Red Hat Linux 7.2, GRUB is a software boot loader that can be used to start Red Hat Linux on your computer. It can also start other operating systems, such as Windows 9x. Here, you'll be asked how (or whether) you want to configure a boot loader and which one (GRUB or LILO).

Choose which boot loader you want to install. If you would rather use the legacy boot loader, LILO, make sure it is selected instead of GRUB. If you choose not to install a boot loader, make sure you create a boot disk or have another way to boot your Red Hat Linux system.

To install a boot loader, select where you want to install it. If your system will use only Red Hat Linux

Boot Loader Configuration

Please select the boot loader that the computer will use. GRUB is the default boot loader. However, if you do not wish to overwrite your current boot loader, select "Do not install a boot loader."

● Use GRUB as the boot loader
○ Use LILO as the boot loader
○ Do not install a boot loader

Install Boot Loader record on:

● /dev/hda Master Boot Record (MBR)
○ /dev/hda1 First sector of boot partition

Kernel Parameters: hdd=ide-scsi

☐ Force use of LBA32 (not normally required)

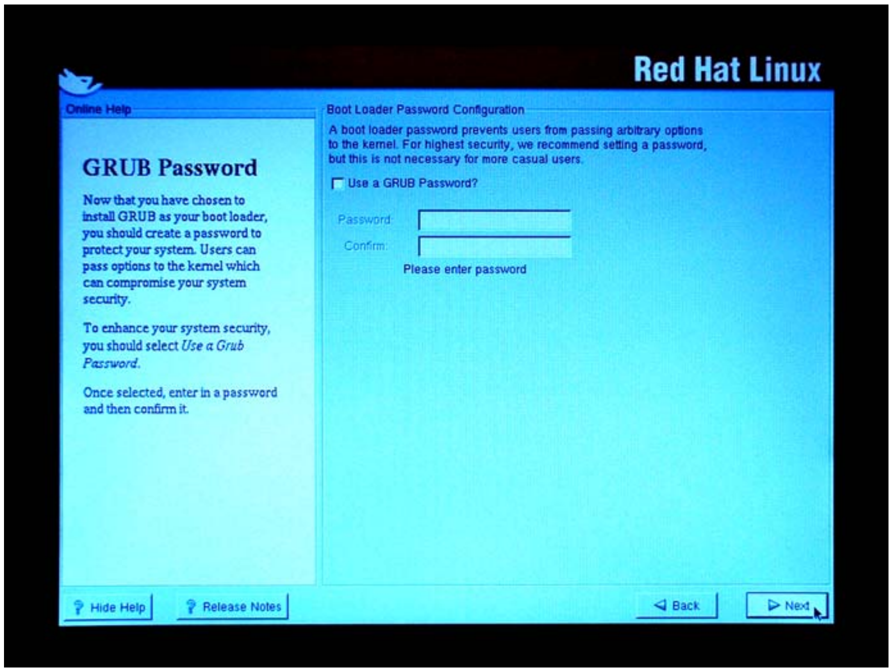
Partition: /dev/hda1 Type:ext3

☑ Default boot image

Boot label: Red Hat Linux

Default	Device	Partition type	Boot label
✓	/dev/hda1	ext3	Red Hat Linux

Figure 7.12: Red Hat installation process (Boot Loader Installation)



GRUB Password

Now that you have chosen to install GRUB as your boot loader, you should create a password to protect your system. Users can pass options to the kernel which can compromise your system security.

To enhance your system security, you should select *Use a Grub Password*.

Once selected, enter in a password and then confirm it.

Boot Loader Password Configuration

A boot loader password prevents users from passing arbitrary options to the kernel. For highest security, we recommend setting a password, but this is not necessary for more casual users.

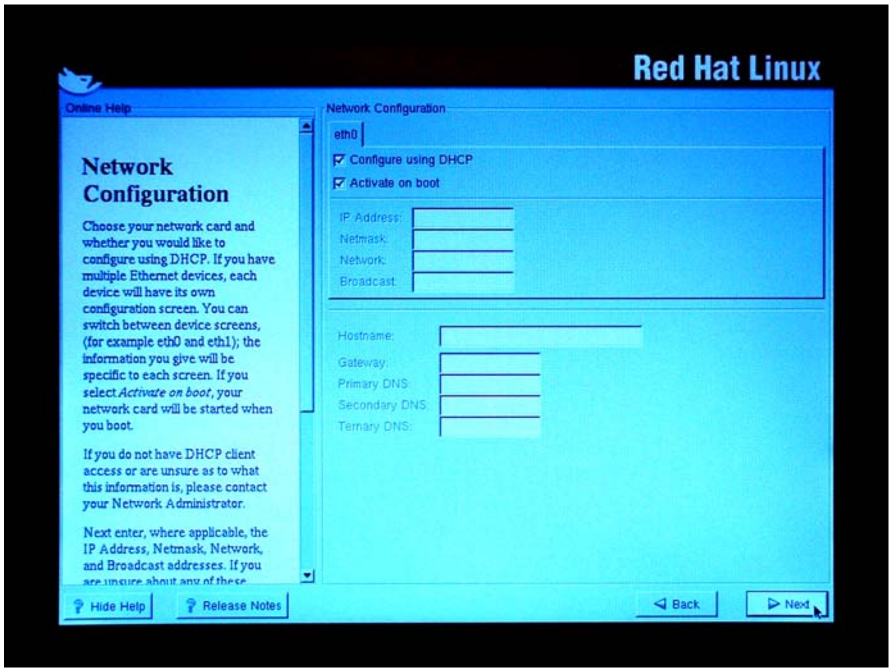
☐ Use a GRUB Password?

Password:

Confirm:

Please enter password

Figure 7.13: Red Hat installation process (GRUB Password)



Network Configuration

Choose your network card and whether you would like to configure using DHCP. If you have multiple Ethernet devices, each device will have its own configuration screen. You can switch between device screens, (for example eth0 and eth1); the information you give will be specific to each screen. If you select *Activate on boot*, your network card will be started when you boot.

If you do not have DHCP client access or are unsure as to what this information is, please contact your Network Administrator.

Next enter, where applicable, the IP Address, Netmask, Network, and Broadcast addresses. If you are unsure about any of these

Network Configuration

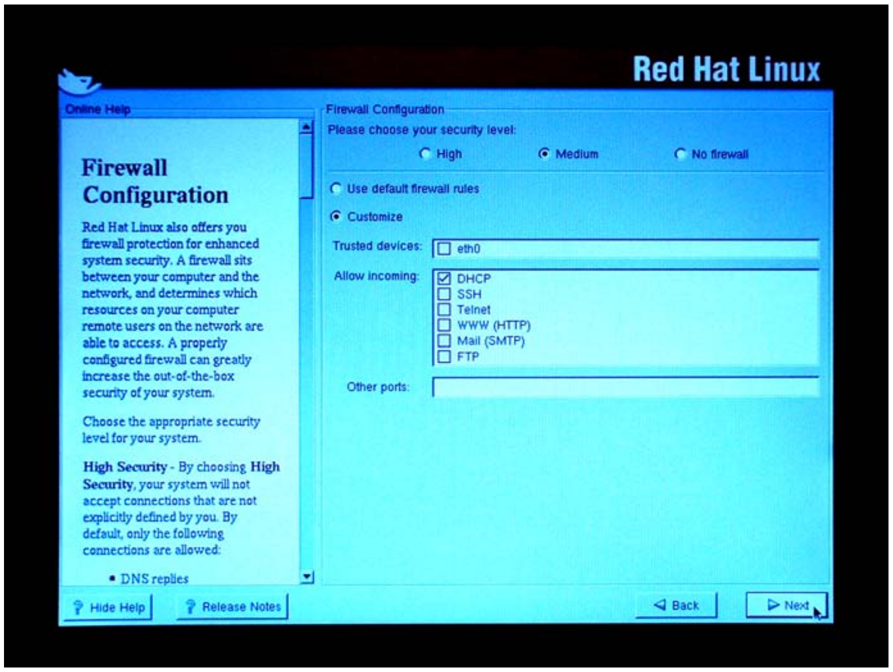
eth0

☑ Configure using DHCP
☑ Activate on boot

IP Address:
Netmask:
Network:
Broadcast:

Hostname:
Gateway:
Primary DNS:
Secondary DNS:
Ternary DNS:

Figure 7.14: Red Hat installation process (Network Configuration)



Firewall Configuration

Red Hat Linux also offers you firewall protection for enhanced system security. A firewall sits between your computer and the network, and determines which resources on your computer remote users on the network are able to access. A properly configured firewall can greatly increase the out-of-the-box security of your system.

Choose the appropriate security level for your system.

High Security - By choosing **High Security**, your system will not accept connections that are not explicitly defined by you. By default, only the following connections are allowed:

- DNS replies

Firewall Configuration

Please choose your security level:

High [✓ Medium] No firewall

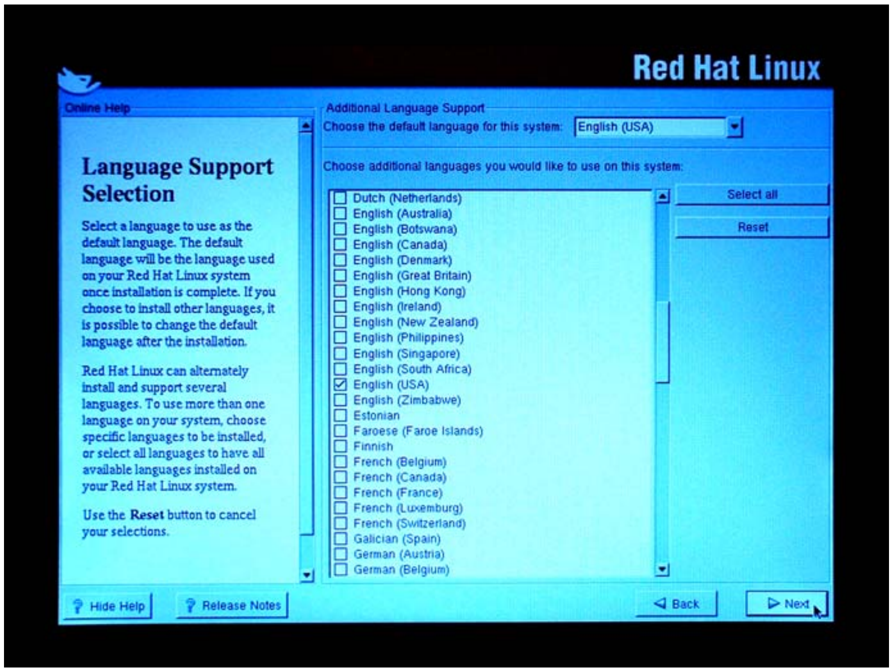
Use default firewall rules [✓ Customize]

Trusted devices: ☐ eth0

Allow incoming:
☑ DHCP
☐ SSH
☐ Telnet
☐ WWW (HTTP)
☐ Mail (SMTP)
☐ FTP

Other ports:

Figure 7.15: Red Hat installation process (Firewall Configuration)



Language Support Selection

Select a language to use as the default language. The default language will be the language used on your Red Hat Linux system once installation is complete. If you choose to install other languages, it is possible to change the default language after the installation.

Red Hat Linux can alternately install and support several languages. To use more than one language on your system, choose specific languages to be installed, or select all languages to have all available languages installed on your Red Hat Linux system.

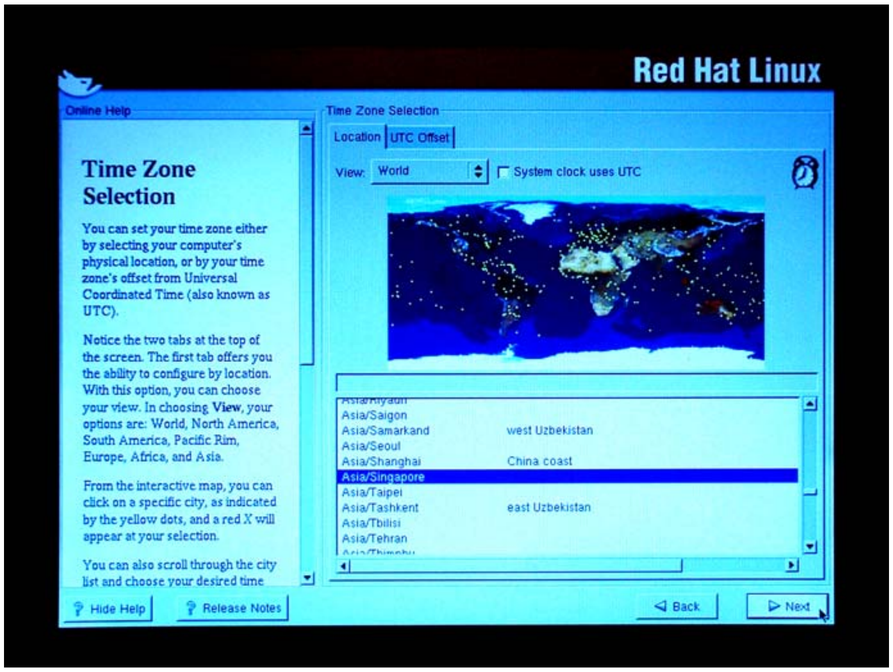
Use the **Reset** button to cancel your selections.

Choose the default language for this system: English (USA)

Choose additional languages you would like to use on this system:

- ☐ Dutch (Netherlands)
- ☐ English (Australia)
- ☐ English (Botswana)
- ☐ English (Canada)
- ☐ English (Denmark)
- ☐ English (Great Britain)
- ☐ English (Hong Kong)
- ☐ English (Ireland)
- ☐ English (New Zealand)
- ☐ English (Philippines)
- ☐ English (Singapore)
- ☐ English (South Africa)
- ☑ English (USA)
- ☐ English (Zimbabwe)
- ☐ Estonian
- ☐ Faroese (Faroe Islands)
- ☐ Finnish
- ☐ French (Belgium)
- ☐ French (Canada)
- ☐ French (France)
- ☐ French (Luxemburg)
- ☐ French (Switzerland)
- ☐ Galician (Spain)
- ☐ German (Austria)
- ☐ German (Belgium)

Figure 7.16: Red Hat installation process (Language Support Selection)



Time Zone Selection

You can set your time zone either by selecting your computer's physical location, or by your time zone's offset from Universal Coordinated Time (also known as UTC).

Notice the two tabs at the top of the screen. The first tab offers you the ability to configure by location. With this option, you can choose your view. In choosing **View**, your options are: World, North America, South America, Pacific Rim, Europe, Africa, and Asia.

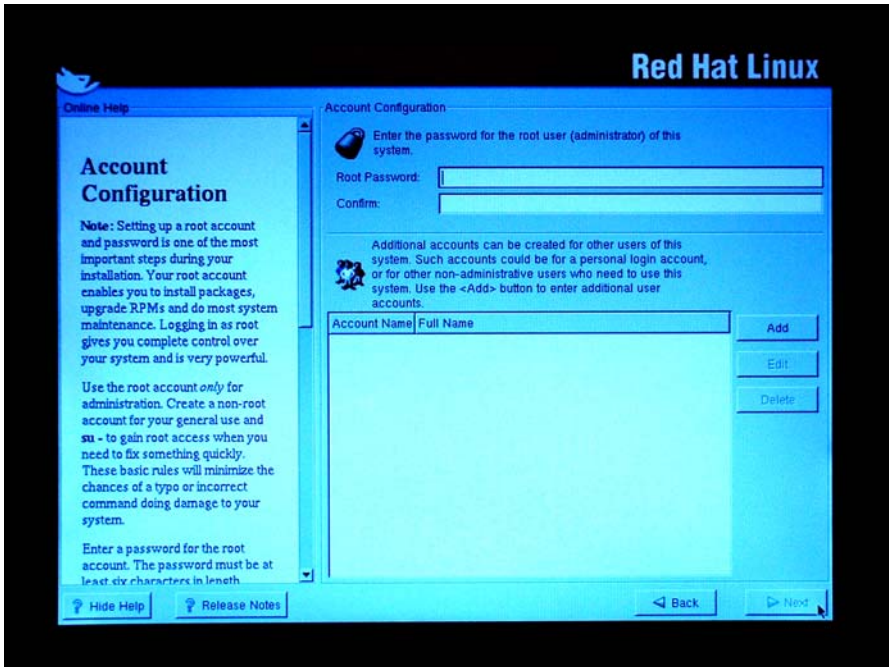
From the interactive map, you can click on a specific city, as indicated by the yellow dots, and a red X will appear at your selection.

You can also scroll through the city list and choose your desired time

View: World ☐ System clock uses UTC

Asia/Saigon	
Asia/Samarkand	west Uzbekistan
Asia/Seoul	
Asia/Shanghai	China coast
Asia/Singapore	
Asia/Taipei	
Asia/Tashkent	east Uzbekistan
Asia/Tbilisi	
Asia/Tehran	

Figure 7.17: Red Hat installation process (Time Zone Selection)



Account Configuration

Note: Setting up a root account and password is one of the most important steps during your installation. Your root account enables you to install packages, upgrade RPMs and do most system maintenance. Logging in as root gives you complete control over your system and is very powerful.

Use the root account *only* for administration. Create a non-root account for your general use and **su -** to gain root access when you need to fix something quickly. These basic rules will minimize the chances of a typo or incorrect command doing damage to your system.

Enter a password for the root account. The password must be at least six characters in length

Account Configuration

Enter the password for the root user (administrator) of this system.

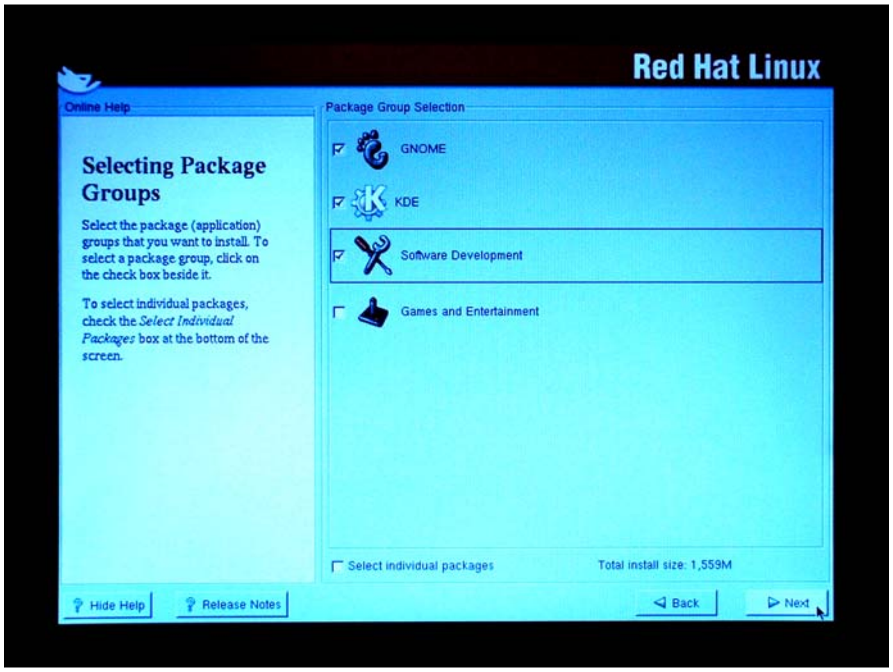
Root Password:

Confirm:

Additional accounts can be created for other users of this system. Such accounts could be for a personal login account, or for other non-administrative users who need to use this system. Use the <Add> button to enter additional user accounts.

Account Name	Full Name

Figure 7.18: Red Hat installation process (Account Configuration)



Selecting Package Groups

Select the package (application) groups that you want to install. To select a package group, click on the check box beside it.

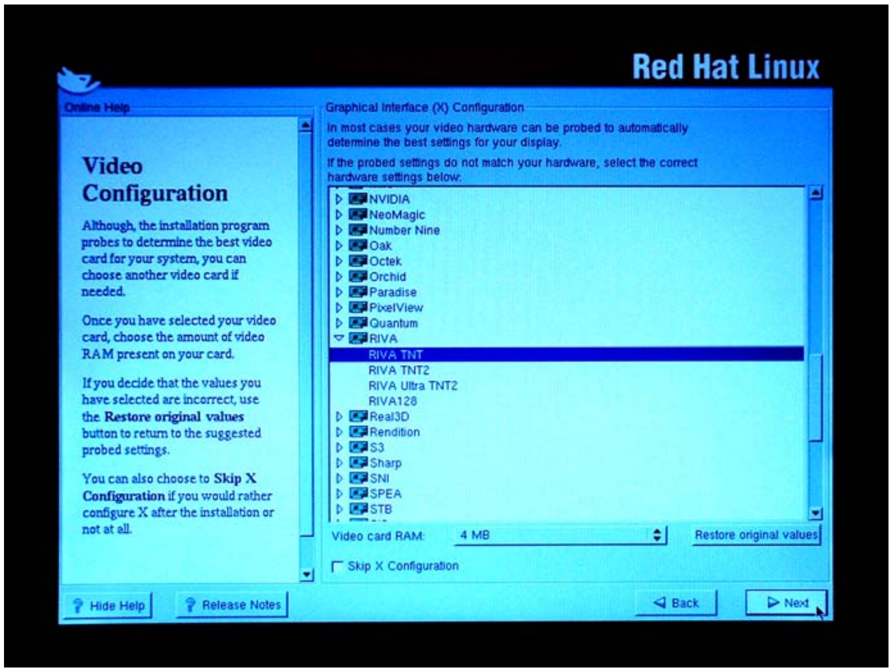
To select individual packages, check the *Select Individual Packages* box at the bottom of the screen.

Package Group Selection

- [✓] GNOME
- [✓] KDE
- [✓] Software Development
- [ ] Games and Entertainment

[ ] Select individual packages   Total install size: 1,559M

Figure 7.19: Red Hat installation process (Selecting Package Group)



Video Configuration

Although, the installation program probes to determine the best video card for your system, you can choose another video card if needed.

Once you have selected your video card, choose the amount of video RAM present on your card.

If you decide that the values you have selected are incorrect, use the **Restore original values** button to return to the suggested probed settings.

You can also choose to **Skip X Configuration** if you would rather configure X after the installation or not at all.

Graphical Interface (X) Configuration

In most cases your video hardware can be probed to automatically determine the best settings for your display.

If the probed settings do not match your hardware, select the correct hardware settings below:

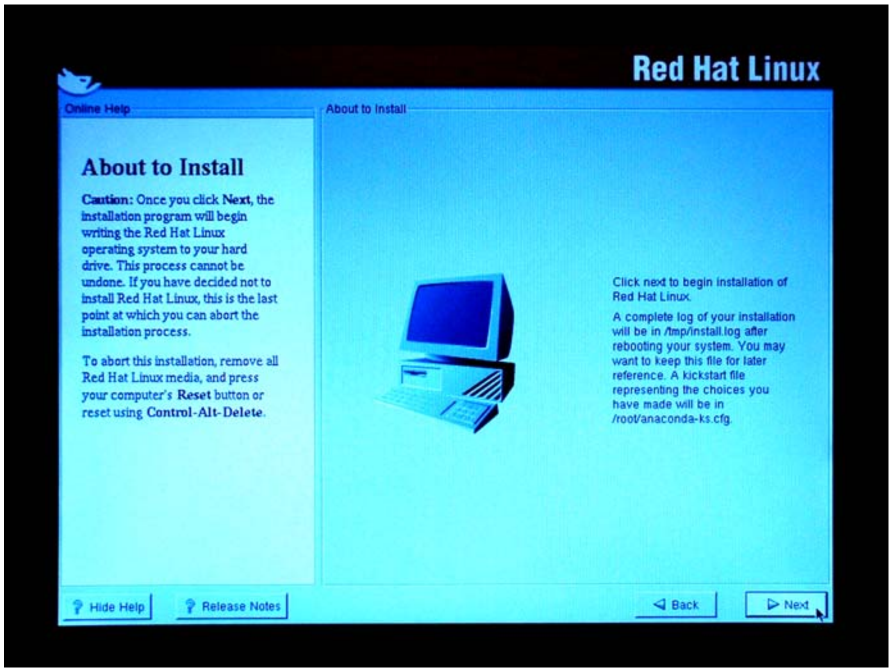
```
▷ NVIDIA
▷ NeoMagic
▷ Number Nine
▷ Oak
▷ Octek
▷ Orchid
▷ Paradise
▷ PixelView
▷ Quantum
▽ RIVA
    RIVA TNT
    RIVA TNT2
    RIVA Ultra TNT2
    RIVA128
▷ Real3D
▷ Rendition
▷ S3
▷ Sharp
▷ SNI
▷ SPEA
▷ STB
```

Video card RAM: 4 MB

Restore original values

☐ Skip X Configuration

Figure 7.20: Red Hat installation process (Video Configuration)



About to Install

Caution: Once you click **Next**, the installation program will begin writing the Red Hat Linux operating system to your hard drive. This process cannot be undone. If you have decided not to install Red Hat Linux, this is the last point at which you can abort the installation process.

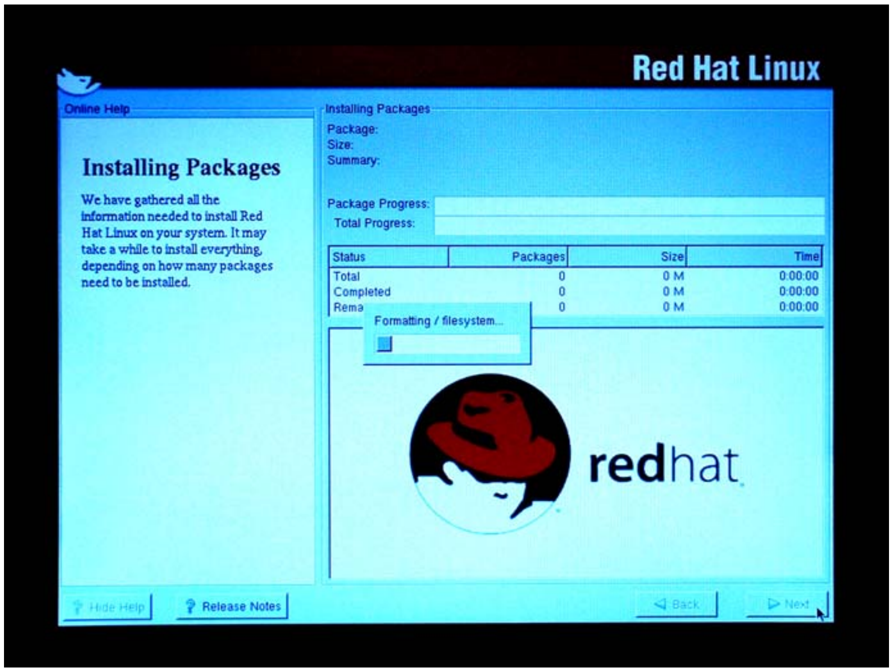
To abort this installation, remove all Red Hat Linux media, and press your computer's **Reset** button or reset using **Control-Alt-Delete**.

About to Install

Click next to begin installation of Red Hat Linux.

A complete log of your installation will be in /tmp/install.log after rebooting your system. You may want to keep this file for later reference. A kickstart file representing the choices you have made will be in /root/anaconda-ks.cfg.

Figure 7.21: Red Hat installation process (Installing Package part 1)



Installing Packages

We have gathered all the information needed to install Red Hat Linux on your system. It may take a while to install everything, depending on how many packages need to be installed.

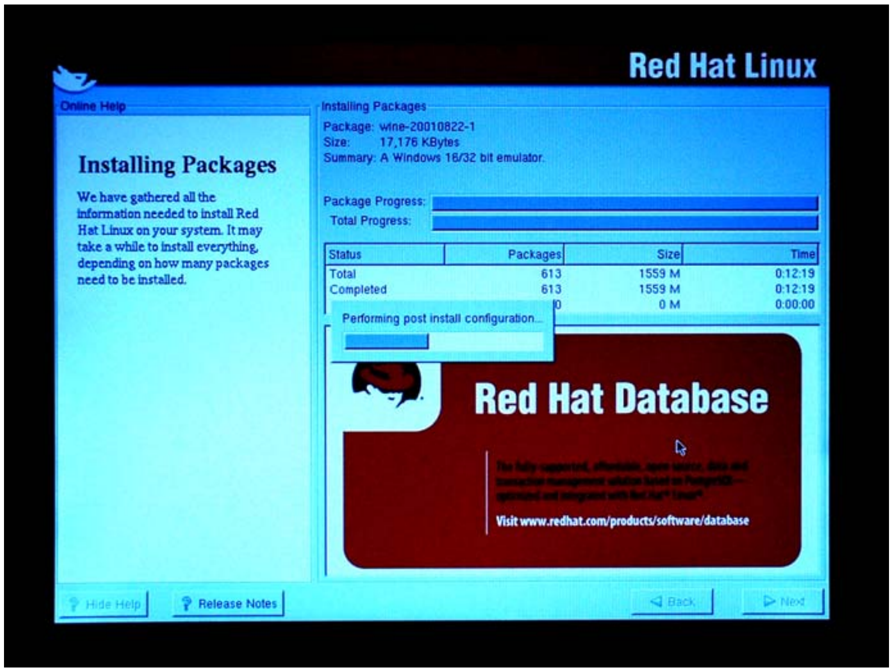
Installing Packages
Package:
Size:
Summary:

Package Progress:
Total Progress:

Status	Packages	Size	Time
Total	0	0 M	0:00:00
Completed	0	0 M	0:00:00
Rema	0	0 M	0:00:00

Formatting / filesystem...

Figure 7.22: Red Hat installation process (Installing Package part 2)



Installing Packages

We have gathered all the information needed to install Red Hat Linux on your system. It may take a while to install everything, depending on how many packages need to be installed.

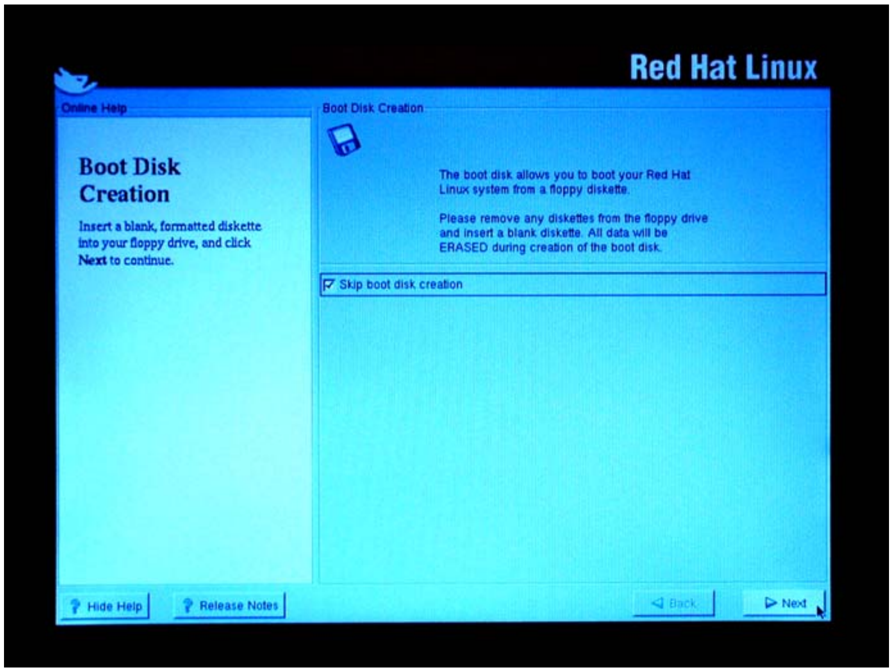
Installing Packages
Package: wine-20010822-1
Size: 17,176 KBytes
Summary: A Windows 16/32 bit emulator.

Package Progress:
Total Progress:

Status	Packages	Size	Time
Total	613	1559 M	0:12:19
Completed	613	1559 M	0:12:19
	0	0 M	0:00:00

Performing post install configuration...

Figure 7.23: Red Hat installation process (Installing Package part 3)



Boot Disk Creation

Insert a blank, formatted diskette into your floppy drive, and click **Next** to continue.

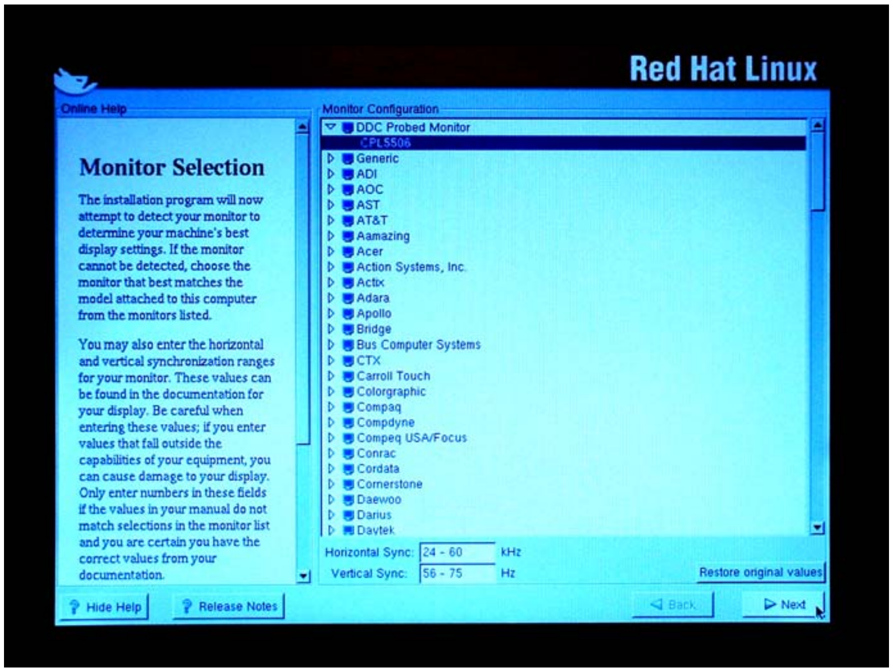
Boot Disk Creation

The boot disk allows you to boot your Red Hat Linux system from a floppy diskette.

Please remove any diskettes from the floppy drive and insert a blank diskette. All data will be ERASED during creation of the boot disk.

☑ Skip boot disk creation

Figure 7.24: Red Hat installation process (Boot Disk Creation)



Monitor Selection

The installation program will now attempt to detect your monitor to determine your machine's best display settings. If the monitor cannot be detected, choose the monitor that best matches the model attached to this computer from the monitors listed.

You may also enter the horizontal and vertical synchronization ranges for your monitor. These values can be found in the documentation for your display. Be careful when entering these values; if you enter values that fall outside the capabilities of your equipment, you can cause damage to your display. Only enter numbers in these fields if the values in your manual do not match selections in the monitor list and you are certain you have the correct values from your documentation.

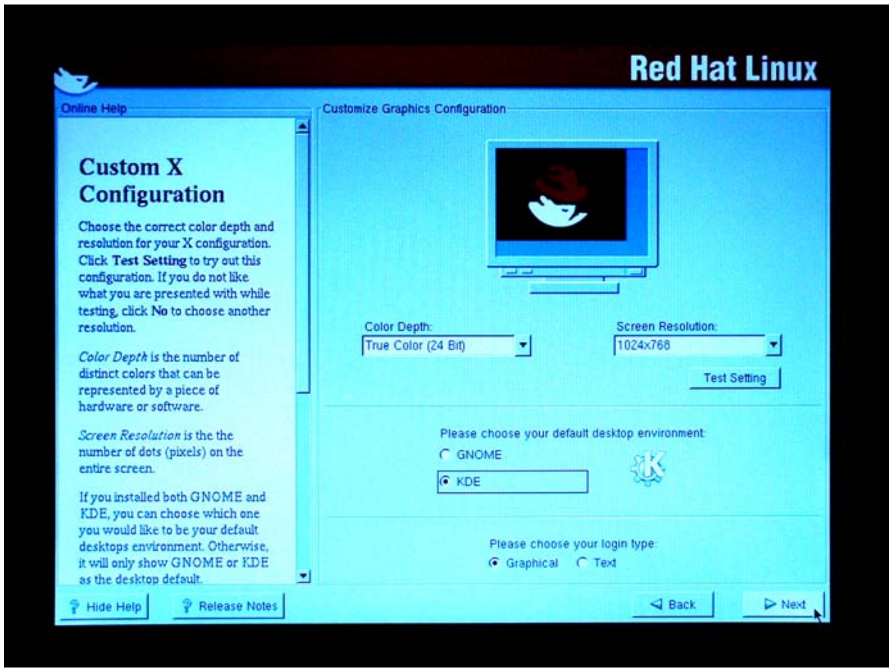
Monitor Configuration

- DDC Probed Monitor
  - CPL5506
- Generic
- ADI
- AOC
- AST
- AT&T
- Aamazing
- Acer
- Action Systems, Inc.
- Actix
- Adara
- Apollo
- Bridge
- Bus Computer Systems
- CTX
- Carroll Touch
- Colorgraphic
- Compaq
- Compdyne
- Compeq USA/Focus
- Conrac
- Cordata
- Cornerstone
- Daewoo
- Darius
- Daytek

Horizontal Sync: 24 - 60 kHz

Vertical Sync: 56 - 75 Hz

Figure 7.25: Red Hat installation process (Monitor Selection)



Custom X Configuration

Choose the correct color depth and resolution for your X configuration. Click **Test Setting** to try out this configuration. If you do not like what you are presented with while testing, click **No** to choose another resolution.

Color Depth is the number of distinct colors that can be represented by a piece of hardware or software.

Screen Resolution is the the number of dots (pixels) on the entire screen.

If you installed both GNOME and KDE, you can choose which one you would like to be your default desktops environment. Otherwise, it will only show GNOME or KDE as the desktop default.

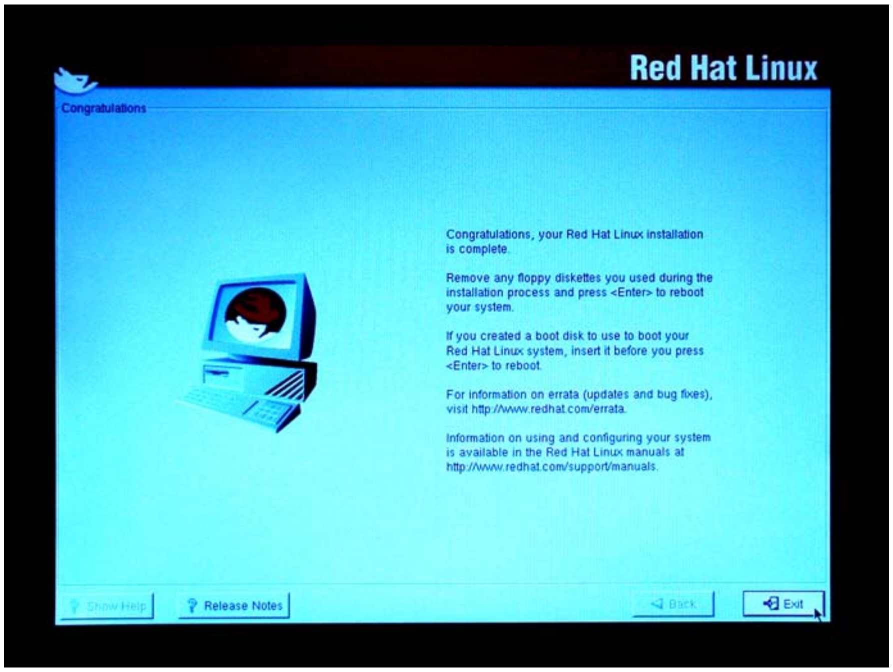
Color Depth: True Color (24 Bit)

Screen Resolution: 1024x768

Please choose your default desktop environment: GNOME [✓ KDE]

Please choose your login type: [✓ Graphical] Text

Figure 7.26: Red Hat installation process (X Configuration)



Congratulations, your Red Hat Linux installation is complete.

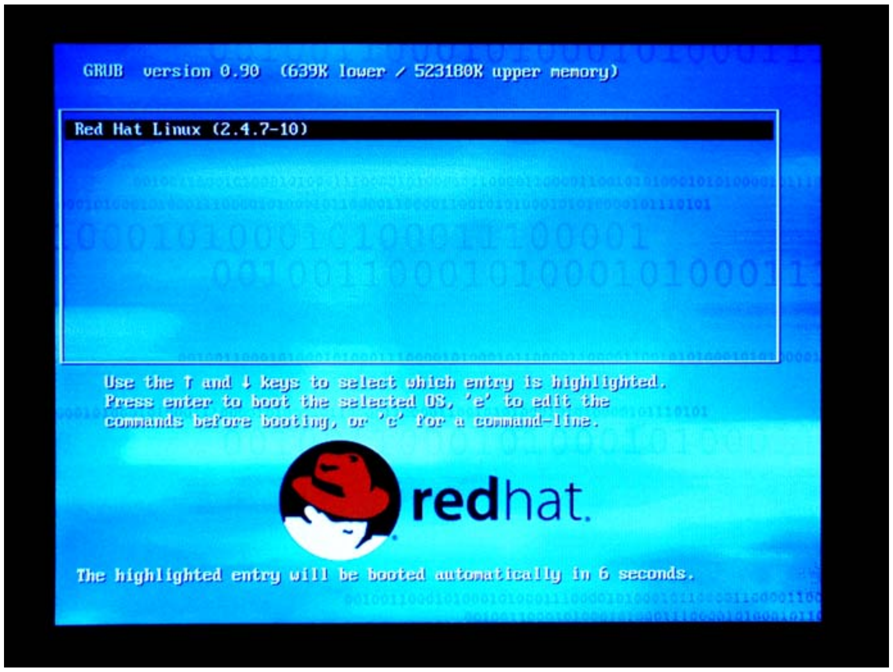
Remove any floppy diskettes you used during the installation process and press <Enter> to reboot your system.

If you created a boot disk to use to boot your Red Hat Linux system, insert it before you press <Enter> to reboot.

For information on errata (updates and bug fixes), visit http://www.redhat.com/errata.

Information on using and configuring your system is available in the Red Hat Linux manuals at http://www.redhat.com/support/manuals.

Figure 7.27: Red Hat installation process (Congratulations-Linux has been installed)



```
GRUB  version 0.90  (639K lower / 523180K upper memory)

Red Hat Linux (2.4.7-10)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 6 seconds.
```

Figure 7.28: Red Hat installation process (Graphical Boot Loader Prompt)

If a user is added, user can login using their username and password. Once at the shell prompt, type ‘startx’ to load the graphical interface. Gnome should start at this time.

7.2 Network Simulator 2 Installation Process

NS2 software can be downloaded at www.isi.edu/nsnam/ns. NS2 can be built from pieces or all at once. Click on ‘Download and Build ns’. Under ‘Getting everything at once’ there is an ns-allinone package (current release is 2.27) available for the download. It requires about 250MB of disk space to build. The path for the download directory in this project is download_directory = /home/peh.

To install and build NS2, execute the following commands:

- cd download_directory
- mkdir ns
- download ns-allinone-2.27.tar.gz in download_directory/ns directory
- gunzip ns-allinone-2.27.tar.gz
- tar xvf ./ns-allinone-2.27.tar
- cd ns-allinone-2.27
- ./install

After successful installation of the ns-allinone package, a message will be generated:

```
***************************Message-start******************************

Nam has been installed successfully.
Please compile your gt-itm & sgb2ns separately.
Ns-allinone package has been installed successfully.
Here are the installation places:
tcl8.4.5:      /home/peh/ns/ns-allinone-2.27/{bin,include,lib}
tk8.4.5:            /home/peh/ns/ns-allinone-2.27/{bin,include,lib}
otcl:          /home/peh/ns/ns-allinone-2.27/otcl-1.8
tclcl:         /home/peh/ns/ns-allinone-2.27/tclcl-1.15
ns:            /home/peh/ns/ns-allinone-2.27/ns-2.27/ns
nam:    /home/peh/ns/ns-allinone-2.27/nam-1.10/nam
xgraph: /home/peh/ns/ns-allinone-2.27/xgraph-12.1

--------------------------------------------------------------------------------
```

```
Please put /home/peh/ns/ns-allinone-2.27/bin:/home/peh/ns/ns-allinone-
2.27/tcl8.4.5/unix:/home/peh/ns/ns-allinone-2.27/tk8.4.5/unix
into your PATH environment; so that you'll be able to run itm/tclsh/wish/xgraph.

IMPORTANT NOTICES:

(1) You MUST put /home/peh/ns/ns-allinone-2.27/otcl-1.8, /home/peh/ns/ns-allinone-2.27/lib,
into your LD_LIBRARY_PATH environment variable.
If it complains about X libraries, add path to your X libraries into LD_LIBRARY_PATH.
If you are using csh, you can set it like:
 setenv LD_LIBRARY_PATH <paths>
 If you are using sh, you can set it like:
export LD_LIBRARY_PATH=<paths>

(2) You MUST put /home/peh/ns/ns-allinone-2.27/tcl8.4.5/library into your TCL_LIBRARY
environmental variable. Otherwise ns/nam will complain during startup.

(3) [OPTIONAL] To save disk space, you can now delete directories tcl8.4.5 and tk8.4.5. They are now
installed under /home/peh/ns/ns-allinone-2.27/{bin,include,lib}

After these steps, you can now run the ns validation suite with
cd ns-2.27; ./validate
For trouble shooting, please first read ns problems page
http://www.isi.edu/nsnam/ns/ns-problems.html. Also search the ns mailing list archive
for related posts.
```

***************************Message-end******************************

The environment variables in the .cshrc file inside Linux needs to be set. This allows the variables to be automatically set, every time the user log into the account. The following commands can be used.

- cd
- open the .cshrc file for editing
- add the following lines:

```
set  path=($path download_directory/ns/ns-allinone-2.27/bin)
set  path=($path download_directory/ns/ns-allinone-2.27/tcl8.4.5/unix)
set  path=($path download_directory/ns/ns-allinone-2.27/tk8.4.5/unix)
set  LD_LIBRARY_PATH=download_directory/ns/ns-allinone-2.27/otcl-
1.8:download_directory/ns/ns-allinone-2.27/lib:/usr/openwin/lib
set   TCL_LIBRARY=download_directory/ns/ns-allinone-2.27/tcl8.4.5/library
```

- save the changes
- source .cshrc

Perform the validation tests:

- cd download_directory/ns/ns-allinone-2.27/ns-2.27
- ./validate

The validation tests will take about four hours to execute. After the successful completion of the tests the output should contain the following lines:

```
****************Message-start************************

Test output agrees with reference output
All test output agrees with reference output.
These messages are NOT errors and can be ignored:
warning: using backward compatibility mode
This test is not implemented in backward compatibility mode

validate overall report: all tests passed

****************Message-end*************************
```

After this point the installation of NS2 is successfully completed.

Chapter 8

Conclusions and Further Work

8.1 Conclusions

TCP has been performing well over the traditional wired networks where packet losses are usually caused by network congestion. However in wireless networks, this assumption would be inadequate. As the reason of packet loss in wireless networks is caused by the high bit error rate over the wireless link, thus TCP performance is degraded under these new conditions. The main reason of TCP poor performance is due to the fact that TCP cannot distinguish between packet losses due to wireless errors from those due to congestion. This will significantly degrades TCP end-to-end delay performance, as there is an increase in delay of re-transmitting of the lost packets.

In this project, the performances of improved mechanisms are evaluated. Experimental results show that the mechanisms such as Snoop protocol, Explicit Loss Notification (ELN) and Explicit Congestion Notification (ECN) can improve both delay performance and throughput of TCP in wireless networks significantly.

In Snoop Protocol, Biswas (2003) found out that although TCP performance deteriorates with increased delay over the wireless hop, Snoop still managed to obtain a better throughput than the normal TCP. The performance improvement is close to two times that of normal TCP. Biswas (2003) also found out that when packets were corrupted over the wireless link, Snoop still managed to give a consistent performance higher than normal TCP.

The advantage as noted by Balakrishnan *et al* (1998) is that Snoop mechanisms improved the performance of the connection in both directions, without sacrificing any of the end-to-end semantics of TCP, modifying host TCP code in the fixed network or re-linking existing applications. However, West and Vaidya (1997) found out that one of the greatest disadvantages of Snoop protocol was that it requires the ACKs to follow the same path as the data in order to shield the sender from losses. This was not a problem for network topologies containing a single wireless path, which every packet must traverse. However it did became a problem when multiple wireless paths were possible, or with asymmetric links where the sender used a high bandwidth, high delay path (such as a satellite link) to send the data and the receiver used a low bandwidth terrestrial path to return the ACKs. Snoop had no method of informing the sender when the base station experiences a period of high errors and this could lead to unnecessary time out, which invoke congestion avoidance procedures.

In ELN, Ding and Jamalipour (2001) found out that the throughput performance of the Snoop and ELN-ACK protocols remains very close until the packet loss rate of 1%. Beyond that, the ELN-ACK outperforms the Snoop protocol. ELN-ACK protocol was able to improve the throughput performance even more by sending information on the reason of packet loss to the TCP sender whereas the Snoop protocol tries to handle all wireless-related losses at its snoop agent located in the base station. This goes to show that, ELN-ACK protocol was able to add extra features to the Snoop protocol and immunes all packet loss even when the packet loss rate is high and the snoop agent cannot handle them.

The main advantage of ELN was that it helped to maintain a large TCP congestion window even when wireless error rates were high, reacting only to congestion. However the disadvantage of ELN mechanism was that it required TCP-stack modifications at all endpoints. Therefore ELN mechanism required standardization of modifications in TCP followed by widespread acceptance of these changes.

In ECN, Pentikousis and Badr (2003) found out that ECN network allows for a fairer allocation of resources. ECN can convey congestion information in a timely manner and can diminish packet drops, thus increasing the delivered-to-dropped packet ratio. It showed that aggressive ECN was successful in reducing packet drops, promoting a fairer environment, increasing network efficiency, and delivering higher and more even performance to individual connections.

The advantage noted by Pentikousis and Badr (2003) was that many network operators charge their customers by the amount of traffic they carry through their routers and dropping a packet could be significant in many cases. Therefore, network costs could be reduced with aggressive ECN, which yields high goodput efficiency and fewer packet drops, and, hence, higher operating margins. On the other hand, Floyd (1994) noted that there were two disadvantages with ECN concerning non-compliant ECN connections and the potential loss of ECN messages in the network. A non-compliant TCP connection could set the ECN field to indicate that it was ECN-capable, and then ignore ECN notifications. For a network that uses only packet drops for congestion notification, a non-compliant connection could also refrain from making appropriate window decreases in response to packet drops. A non-compliant connection interested in reliable delivery cannot ignore packet drops completely, but in the absence of monitoring and controls, a non-compliant connection could cause congestion problems in either an ECN or a non-ECN environment. Network with ECN messages that had no counterpart with packet drops could drop the messages and the congestion notification could fail to reach the end node. Even with the use of ECN fields in packet headers, it could not guarantee that the TCP source would receive each notification of congestion.

Network Simulator 2 (NS2) was chosen for this project, as it is an event-driven network simulator, which is popular with the networking research community. It includes numerous models of common Internet protocols including several newer protocols, such as reliable multicast and TCP selective acknowledgement. Additionally, different levels of configuration are present in NS2 due to its open source nature, including the capability of creating custom applications and protocols as well as modifying several parameters at different layers. On top of that, NS2's code source is split between C++ for its core engine and OTcl, an object oriented version of TCL for configuration and simulation scripts. The combination of the two languages offers an interesting compromise between performance and ease of use. A highly dynamic newsgroups and source codes are also available on the web to provide assistance to most of the problems encounter while using NS2.

8.2 Further Work

Implement the representative mechanisms (Snoop, ELN and ECN) using NS2 and compare the performance of TCP Reno with the representative mechanisms under various conditions. Propose and implement possible solutions to improve the representative mechanisms.

In the simulation, the mobility issues are not considered. The focus is on the impact of link loss on the performance of TCP Reno and the improvement mechanisms. Hence, in the simulation environment, a link with the error rates of 0.1%, 1.0% and 10.0% is used to represent wireless link. The simulation environment also consists of a 10 Mbps, 20 ms delay wired channel and a 2Mbps wireless channel with a negligible delay. The packet size is to be set at 1024 bytes and the simulation time is 150sec. The maximum possible window size for the connection is to be set at 64 Kbytes.

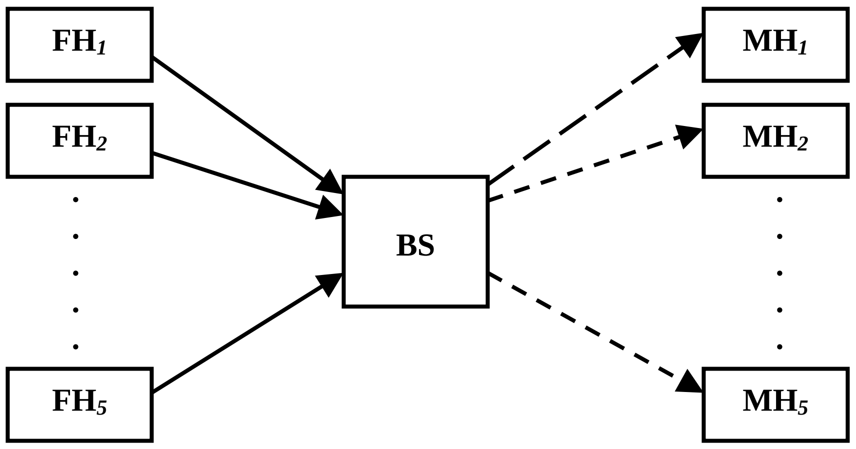


Figure 8.1: Network topology 1

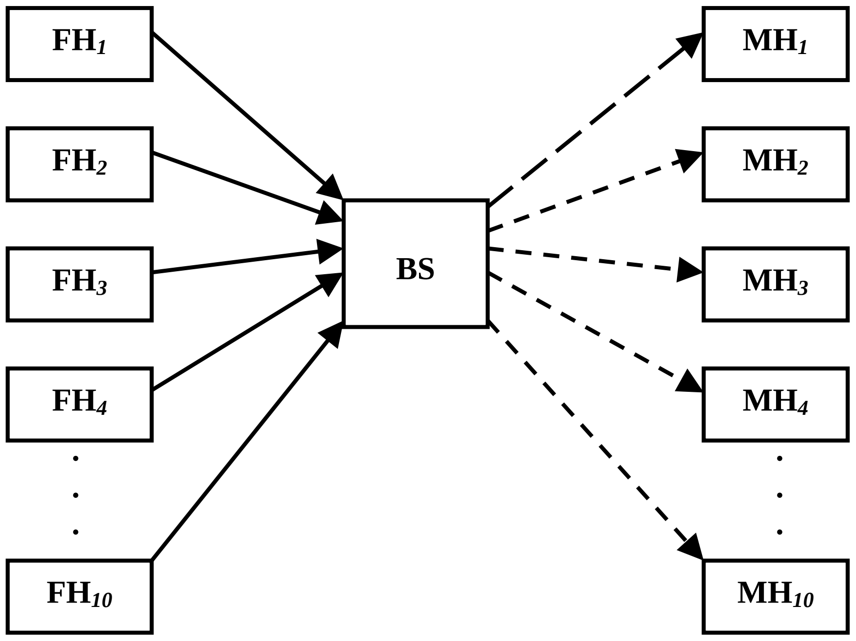


Figure 8.2: Network topology 2

With reference to Figure 8.1 and 8.2, the wired-cum-wireless topology is used in this project. In network topology 1, five Fixed Hosts (FHs) are directly attached to the Base Station (BS) through a wired network and five Mobile Hosts (MHs) are one hop away from the wireless access point. Whereas in network topology 2, ten FHs are directly attached to the BS through a wired network and ten MHs are one hop away from the wireless access point. Both the network topologies are to be simulated using the NS2.

Six simulation models will be investigated in the experiment:

1) TCP Reno only
2) TCP Reno with Snoop
3) TCP Reno with ECN
4) TCP Reno with Snoop and ECN.
5) TCP Reno with ELN
6) TCP Reno with Snoop and ELN.

Each of the six simulation models will be investigated in scenario 1 and 2, with the simulation environment remain unchanged. During each run, the throughput, goodput, delay, fairness performance and changes in congestion window size of the models will be examined.

References

Balakrishnan, H. and Katz, R. H. (1998), Explicit Loss Notification and Wireless Web Performance. *Proc. IEEE Globecom Internet Mini-Conference, Sydney, Australia* [Online], November 1998.
http://nms.lcs.mit.edu/~hari/papers/globecom98/
current May 2004

Balakrishnan, H., Seshan, S., and Katz, R. H. (1998), Improving Reliable Transport and Handoff Performance in Cellular Wireless Network [Online], November 1998.
http://nms.lcs.mit.edu/~hari/papers/globecom98/
current May 2004

Biswas, I. (2003), Undergraduate Research Opportunity Program (UROP) Project Report Enhancing, and Implementing the SNOOP Protocol in Linux [Online]
http://www.cir.nus.edu.sg/research/software/snoop/linux-snoop-thesis.pdf
current May 2004

Brakmo, L. and Peterson, L. (1995), TCP Vegas: End-to-end congestion Avoidance on a global Internet [Online]
http://netlab.snu.ac.kr/lecture/computer_networks/2004Spring/TCP-congestion%20control.ppt
current May 2004

Cavin, D., Sasson, Y. and Schiper, A. (2002),
On the Accuracy of MANET Simulators [Online]
http://lsewww.epfl.ch/Documents/acrobat/CSA02b.pdf
current May 2004

Chung, J. and Claypool, M., NS by Example [Online],
http://nile.wpi.edu/NS/
current Nov 2003

Deshpande, N. (1999), TCP Extensions for Wireless Networks [Online]
http://www.cse.ohio-state.edu/~jain/cis788-99/ftp/tcp_wireless/index.html
current May 2004

Ding, W. and Jamalipour, A. (2001), Delay Performance of the New Explicit Loss Notification TCP Technique for Wireless Networks [Online]
http://csl.ee.iastate.edu/~cpre543/paper/wenq01glo.pdf
current May 2004

Durresi, A., Sridharan, M., Liu, C., and Jain, R. (2002), Improved Explicit Congestion Notification for Satellite Networks [Online]
http://www.cse.ohio-state.edu/~jain/papers/ftp/itcom01.pdf
current May 2004

Ewerlid, A. (2001), Reliable Communication over Wireless Links [Online], April 2001
http://www.signal.uu.se/Publications/pdf/c0100.pdf
current Sept 2004

Floyd, S. (1994), TCP and Explicit Congestion Notification [Online], October 1994
http://www-nrg.ee.lbl.gov/papers/tcp_ecn.4.pdf
current Sept 2004

Flow-Control Mechanisms [Online], 2001
http://www.linktionary.com/f/flow_control.html
current May 2004

Hassan, M. and Jain, R. (2001), TCP Performance in Future Networking Environments. *Guest Editorial*, *IEEE Communications Magazine* [Online], April 2001, pp. 51.
http://www.cis.ohio-state.edu/~jain/papers/tcp_ed.htm
current May 2004

Information Sciences Institute (1981), *TRANSMISSION CONTROL PROTOCOL, DARPA INTERNET PROGRAM, PROTOCOL SPECIFICATION*, University of Southern California. [Online]
http://www.ietf.org/rfc/rfc0793.txt
current May 2004

Kinicki, R. and Zheng, Z. (2001), A Performance Study of Explicit Congestion Notification (ECN) with Heterogeneous TCP Flows [Online], July 2001
http://www.cs.wpi.edu/~rek/ICN01talk.ppt
current Sept 2004

Kristoff, J. (2003), The Transmission Control Protocol [Online], February 2003
http://condor.depaul.edu/~jkristof/technotes/tcp.html
current Sept 2004

Lucio, G. F., Paredes-Farrera, M., Jammeh, E., Fleury, M. and Reed M. J. (2002), OPNET Modeler and Ns-2: Comparing the Accuracy Of Network Simulators for Packet-Level Analysis using a Network Testbed [Online]
http://esewww.essex.ac.uk/~fleum/weas.pdf
current May 2004

Moraru, B., Copaciu, F., Lazar, G. and Dobrota, V. (2002),
Practical Analysis of TCP Implementations: Tahoe, Reno, NewReno [Online]
http://conference.iasi.roedu.net/site/conference/papers/MORARU_B-Practical_Analysis_of_TCP_Implementations_Tahoe_R..pdf
current May 2004

Nishida, Y. (2003), TCP and Congestion Control (Day 2) [Online]
http://www.soi.wide.ad.jp/class/20020032/slides/15/index_31.html
current May 2004

NS-2 simulation tool home page [Online]
http://www.isi.edu/nsnam/ns/
current Nov 2003

Pentikousis, K. and Badr, H. (2003), An Evaluation of TCP with Explicit Congestion Notification [Online], 2003
http://www.cs.sunysb.edu/~kostas/art/ecneval.pdf
current Sept 2004

Pilosof, S., Ramjee, R., Raz, D., Shavitt Y. and Sinha, P. (2002), Understanding TCP fairness over Wireless LAN [Online]
http://www.bell-labs.com/user/ramjee/papers/tcpfair03.pdf
current May 2004

Ramakrishnan, K., Floyd. S., and Black, D. (2001), RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP [Online], September 2001
http://www.faqs.org/rfcs/rfc3168.html
current Aug 2004

Red Hat Linux 7.2 The Official Red Hat Linux x86 Installation Guide [Online]
http://www.redhat.com/docs/manuals/linux/RHL-7.2-Manual/install-guide/
current Dec 2003

Sharma, N. K. and Hu, F. (2002), Enhancing Wireless Internet Performance. *IEEE Communications Surveys & Tutorials* [Online], December 2002
http://www.comsoc.org/livepubs/surveys/Public/2002/Dec/hu.html
current May 2004

SY22 Computer Networks [Online]
http://www.comp.leeds.ac.uk/sy22/web_pages/113/WirelessTCP.html#top
current Sept 2004

Tutorial for Network Simulator “ns” [Online]
http://www.isi.edu/nsnam/ns/tutorial/
current Nov 2003

Ubik, S. and Klaban, J. (2003), Experience with using simulations for congestion control research [Online], December 5, 2003
http://www.ten.cz/doc/techzpravy/2003/congestion/congestion.pdf
current Aug 2004

West, S. M. and Vaidya, N. H. (1997), TCP Enhancements for Heterogeneous Networks [Online], April 1997
http://www.crhc.uiuc.edu/~nhv/old.papers/mobile-computing/97-003.ps.Z.
current Sept 2004

Wireless Overview [Online], (2002)
http://wireless.ittoolbox.com/pub/wireless_overview.htm
current May 2004

Appendix A

Project Specification

University of Southern Queensland
Faculty of Engineering and Surveying

ENG 4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	**Peh Wee Liang**
TOPIC:	Investigation of TCP performance over Wireless Internet.
SUPERVISOR:	Dr. Hong Zhou
SPONSORSHIP:	Faculty of Engineering, USQ
PROJECT AIM:	The project seeks to investigate the TCP performance over Wireless Internet and that of a few representative mechanisms that are, Snoop protocol, Explicit Loss Notification (ELN) and Explicit Congestion Notification (ECN).
PROGRAMME:	**Issue B, 1st October 2004**

1. Study protocols and representative mechanisms: TCP/IP, Snoop protocol, Explicit Loss Notification (ELN) and Explicit Congestion Notification (ECN).

2. Compare the performance of regular TCP with the representative mechanisms.

3. Identify the advantages and disadvantages of the representative mechanisms.

4. Study simulation tool, Network Simulator (NS2) and Linux.

As time permits

5. Implement the representative mechanisms in NS2.

6. Propose and implement possible solutions to improve the representative mechanisms.

AGREED: ________________ (Student) ________________ (Supervisor)

_____ / _____ / _____ _____ / _____ / _____