DELAYED RESPONSE PROTOCOLS FOR HIGH-DELAY WIRELESS NETWORKS

A Thesis

by

NAUZAD ERACH SADRY

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2002

Major Subject: Computer Science

DELAYED RESPONSE PROTOCOLS FOR HIGH-DELAY WIRELESS NETWORKS

A Thesis

by

NAUZAD ERACH SADRY

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Donald Friesen (Co-Chair of Committee)	A. L. Narsimha Reddy (Co-Chair of Committee)
Riccardo Bettati (Member)	Donald Friesen (Head of Department)

December 2002

Major Subject: Computer Science

ABSTRACT

Delayed Response Protocols for High-Delay Wireless Networks (December 2002)

Nauzad Erach Sadry, B.E., Bombay University, India

Co-Chairs of Advisory Committee: Dr. Donald Friesen
Dr. A. L. Narsimha Reddy

Wireless networks have become more prevalent over the last few years. These networks introduce different types of errors, which are due to the unpredictable nature of the wireless channel. The congestion control algorithms of existing TCP protocols assume packet losses are due to congestion and hence are not suitable to handle such channel errors. In this thesis, we present a novel approach to handle channel errors. Our approach uses the link-level retransmission capability at the base station and deliberately delays the sender's response to any packet loss. It is expected that this delay would allow the link level retransmissions to overcome the channel errors. The transport layer would only have to handle congestion losses after the delay. We have provided the framework for this class of protocols and examined two cases in particular. We have run simulations on the ns-2 platform to show that these protocols can provide better network throughput under various conditions and network settings.

To my parents

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. Reddy, for having given me a chance to work with him. He has been a source of inspiration to me all along. Throughout my research, he has provided me with immense guidance and encouragement. He has always been very open to all my questions and answered them patiently. His weekly meetings have allowed me to complete my thesis in a timely manner. His informal group meetings created an environment for us to discuss work with students working in similar areas. Thank you, Dr. Reddy, for everything.

I would like to thank Dr. Friesen and Dr. Bettati for their interest in this research and for their invaluable suggestions.

I would like to thank my parents for their blessings and guidance; without them my master's degree would never have been possible. Their dreams will forever be the driving force in all my endeavors. I appreciate the support and cooperation of my family members, particularly Kainaz, for always being the source of inspiration for me. I would like to thank all my friends for their help and assistance.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The Internet over the years has seen a phenomenal growth in the number of users and applications. However the Internet has managed to retain its stability mainly because of the congestion avoidance mechanisms of TCP [1] [2], which is still the most dominant protocol on the Internet. Currently the Internet consists of many diverse networks, wired and wireless. The number of wireless networks is increasing. The network has to provide efficient performance regardless of its physical medium. Wireless and satellite networks have non-negligible channel error rates that can significantly influence the performance of TCP. Traditional TCP considers every packet loss as an indicator of congestion [3], and therefore reduces the rate at which it transmits packets into the network after retransmitting the lost packet. Even if the loss is because of a channel error, the TCP sender reduces its congestion window, which is not necessary. Link level retransmissions are typically employed at the base station to overcome channel errors. However, current TCP sources cannot utilize the availability of link level retransmissions.

By the time the acknowledgement of the lost packet successfully retransmitted through the link layer reaches the sender, the TCP sender has already performed a congestion control action. A typical TCP sender like TCP-Reno (4.3 BSD) performs fast

---

The journal model is *IEEE Transactions on Automatic Control.*

retransmission and halves its congestion window on receiving 3 duplicate acknowledgements. If there is no congestion in the network, or the rate of packet loss because of congestion is much less than that due to channel errors, such congestion responses affect the overall performance of the network. TCP considers any packet loss as a congestion loss and therefore the sender reduces its window size. This mechanism can be very detrimental to a network with little or no congestion.

TCP increases the congestion window additively by one after successful transmission of each window of packets. In networks with large delays, this window gets to the original size after several RTTs. During this time, the TCP sender does not realize full network bandwidth. Hence, incorrectly treating a channel error as a congestion loss can result in significant loss of performance. Fairness is another issue that needs to be considered. Any protocol proclaiming to be TCP-Friendly should not choose to send at a rate higher than a TCP connection under similar conditions of round trip delays and losses.

In this thesis, we propose a Delayed Response Protocol. We study the impact of delaying the reduction of congestion window, in the presence of channel errors at the wireless links with large time delays. Along with this delaying of window reduction, we may or may not delay the retransmission of the lost packet. Assuming that link-level retransmission from the base station to the receiver is successful for a channel error; the proposed protocol allows the sender to continue data transmission without reducing the congestion window.

A. RELATED WORK

Several mechanisms have been put forward in the literature to distinguish congestion losses from channel errors, in order to improve TCP's performance in lossy networks. In the next few pages, we provide a quick review of some of these schemes.

1. Explicit Congestion Notification

In [4], Floyd discusses explicit congestion notification (ECN) as a mechanism for notifying senders of congestion in the network. ICMP Source Quench Messages could be used as one of the ECN mechanisms in TCP/IP networks. However such messages consume critical network bandwidth, and hence are ineffective and unfair. Moreover, BSD TCP implementations use slow-start combined with a small slow start threshold. This makes the use of Source Quench messages very unattractive to large window TCP connections. According to [5], in the DECbit congestion avoidance scheme, the gateway uses a congestion notification bit in the TCP packet headers to provide feedback about congestion in the network. The TCP source uses window flow control, and updates its window once every second roundtrip time.

In its implementation [4] considers IP networks with RED gateways. The gateways monitor the average queue size and during congestion use a probabilistic algorithm to choose which arriving packets to mark or drop. The gateways set the ECN bit in the packet header during the forward packet movement. When the TCP receiver receives the data packet with the ECN bit set, it sets the ECN bit in the next outgoing

ACK packet. The TCP Reno source receives an ECN message at time *t.* If no responses to congestion have been made in the last round trip time, the source halves the value of congestion window and slow-start threshold. Moreover, the source does not respond to succeeding ECN messages until all outstanding packets at time *t* are acknowledged. If the TCP Reno source has already performed Fast Retransmit and Fast Recovery before receiving the ECN message, then the source does not respond to ECN until all outstanding packets at time *t* have been acknowledged. If the source gets 3 duplicate ACKs soon after responding to an ECN message, then it retransmits the lost packet, but does not reduce the values of congestion window or slow-start threshold.

The simulations show that ECN is advantageous for low-bandwidth delay-sensitive interactive traffic such as telnet. However, ECN mechanism is at a disadvantage when non-compliant ECN connections may refrain from making appropriate window decreases in response to packet drops. Moreover, an ECN message may be dropped in a network just like any other packet. However in the implementation of [4], since RED gateways are used, the network does not depend on the sender to respond to congestion. The simulations suggest that ECN mechanisms give a clear performance benefit in compliant networks.

2. Explicit Transport Error Notification

According to [6], the authors discuss Explicit Transport Error Notification (ETEN) as a mechanism to aid a TCP sender in distinguishing congestion losses from channel errors.

Two types of schemes are proposed, (a) per-packet based ETEN that notify the TCP sender for each packet corruption that is detected, and, (b) cumulative based ETEN that notify the TCP senders of aggregate packet corruption statistics.

This study assumes a higher degree of interaction between the link-layer and the network layer. Packet level corruption can be detected at the Link-Layer using the 32-bit Frame Check Sequence (FCS) field in 802.11 MAC header. Other schemes of corruption detection include IP checksum, TCP checksum or Ipsec. The ETEN information is reported to the TCP sources using ICMP messages [7]. Per-Packet based ETEN schemes, include three variations,

i. Oracle ETEN – This mechanism assumes that the source of the flow is notified about packet corruption instantaneously by an intermediate router. This is a theoretical construct and serves primarily as an upper limit to the ETEN performance.
ii. Backward ETEN – This mechanism involves an intermediate node transmitting an ICMP message to the TCP sender upon detecting a corrupted packet.
iii. Forward ETEN – This mechanism involves the intermediate router sending an ICMP message to the destination host on detection of a corrupted packet. The destination host conveys this information to the sender via the subsequent acknowledgement.

Cumulative ETEN is useful when the intermediate routers cannot accurately retrieve the source and destination information from the packet header. The routers

collect the cumulative data by observing either absolute bit / packet error rate within a moving window in time, or, by estimating the probability that a packet survives corruption. It can be collected on a per-hop basis or it can be aggregated over an end-to-end path.

The results show that per-packet based ETEN mechanisms can provide considerable gain in TCP goodput in absence of congestion. However this gain reduces as the level of congestion increases. The ETEN mechanism offers high risk of Internet security vulnerability, like Distributed Denial of Service attacks. This is because the intermediate routers are given more powers to extract information from a packet and ICMP messages are traditionally not secure.

3. TCP Snoop Protocol

TCP Snoop protocol [8] is based on the assumption that the network layer at the base station of a network with wireless links is capable of snooping TCP packets. The base station caches the packets and performs local retransmissions across the wireless links. The protocol intends to preserve the end-to-end semantics of the TCP protocol. The aim is to improve end-to-end performance on networks with wireless links without changing the existing TCP implementations at the end hosts.

The protocol intends to make the vagaries of the wireless link invisible to the TCP sender. Since the local retransmissions take care of the packets lost due to wireless channel errors, the TCP sender does not have to retransmit these lost packets, nor reduce

its sending rate. The snoop module at the base station monitors each packet that passes through the connection in either direction. It caches every unacknowledged TCP packet that goes through it and also keeps track of all the acknowledgements sent from the wireless receiver.

The snoop module detects a packet loss either by the arrival of a duplicate acknowledgement or by a local timeout. It retransmits the cached packet to the receiver. The module drops the duplicate acknowledgements at the base station itself. This shields the sender from such a packet lost due to wireless channel errors and prevents any unnecessary congestion control mechanism invocations at the TCP sender.

The results show performance improvements of up to 20 times over normal TCP/IP for data transfers in a network with mobile hosts across a wide range of high bit error rates. At low error rates, the protocol behaves as if the modification at the base station was not present at all, and this ensures no degradation in performance. The experiments show that the protocol is more robust in comparison to regular TCP when dealing with unreliable links and multiple errors in the same transmission window. As a disadvantage, the protocol involves snooping of TCP packets at the network layer, which is semantically incorrect. If a network supports IPsec, then the protocol may not work. Moreover, since duplicate acknowledgements are dropped, the protocol may not be conducive for wireless links with high delays like satellite links. The TCP sender may timeout in such cases resulting in congestion control procedures and subsequent drop in goodput. Also, the paper does not study the effect of the snoop protocol in presence of congestion.

4. Indirect TCP

Indirect TCP (or I-TCP) is a notion that came into existence because of the poor performance of existing IP-based transport protocols when a wireless host communicates with a fixed network. [9] describes the I-TCP protocol, according to which, the unreliable nature of the wireless link can be shielded from the fixed host. It suggests that any communication between a wireless host and a fixed host can be broken into two separate interactions – wireless interaction between the wireless host and its base station and wired interaction between the base station and the fixed host.

The wired communication can be done using any of the existing TCP protocols. The wireless interaction however uses specialized TCP protocols that support applications in a low speed, high delay and unreliable wireless medium. Such a mechanism separates the vastly different characteristics of the two networks, such as flow control and congestion control. The indirection allows the base station to manage most of the communications overhead for the wireless host such as transmitting acknowledgements. The fixed host just sees an image of its peer wireless host residing on the base station. It is completely unaware of the indirection occurring in the communication. The wireless host on the other hand runs a simple wireless protocol to communicate with the base station.

Experimental results show that I-TCP gives good performance in terms of better throughput, in comparison to regular TCP under similar conditions. However the biggest disadvantage of this protocol is that it breaks the “end-to-end semantics” notion of network communication. According to these semantics, the TCP module at the sender

should communicate only with the TCP module at the receiver. As wireless delays are unpredictable in comparison to wired delays, according to I-TCP, a fixed TCP sender can receive acknowledgements of TCP packets from the base station, even before the packets actually reach the wireless recipient. However the authors proclaim that most of the applications that use TCP for bulk data transfer have some kind of application level acknowledgements and error corrections in-built. Therefore I-TCP would never yield weaker results than regular TCP. Another weakness of I-TCP is that wireless applications running on wireless hosts cannot use normal socket libraries available. They need to be re-linked with the special I-TCP library. Since TCP state is also copied into the base station, I-TCP involves a large overhead

On the whole I-TCP is a solution that is not in conjunction with existing legacy systems. It involves lots of changes at the wireless hosts and the base station, which semantically is not supposed to hold any TCP state.

5. TCP Westwood

TCP Westwood (TCPW) is an end-to-end recovery solution for performance improvement in mixed wired and wireless networks. According to [10], it is a sender-side modification of the TCP congestion window algorithm that results in significant improvement in wireless networks with lossy links. The protocol performs estimation on the cause of the packet loss, either due to congestion or wireless channel error. TCPW performs a mechanism called “faster recovery”, according to which, the TCP sender

selects a slow-start threshold and a congestion window based on the effective end-to-end bandwidth estimate at the time of loss detection.

The rate at which the acknowledgements are returned to the sender is monitored for every connection to estimate this bandwidth. Unlike TCP Reno, the faster recovery mechanism of TCPW avoids overly conservative reductions of the congestion window and slow-start threshold, in case the loss is because of a wireless channel error. The authors discuss a set of algorithms, which are used to determine the effective end-to-end bandwidth and the response to a packet loss.

Experimental results of TCPW show a throughput improvement of roughly 550% over TCP Reno for networks with wireless channel errors. The estimated bandwidth at the instance of detecting a packet loss is only marginally harmed by such a channel error. Fairness of TCPW is at least as good, if not better than that provided by TCP Reno. It is observed that co-existing TCP Reno sources do not starve because of the presence of a TCPW source. However some “unfriendly” behavior is reported due to inherent aggressiveness of TCPW. TCPW involves modification of only the sender side, and not the receiver or any intermediate router, leading to simpler deployment.

However, TCPW may not perform well when the random packet loss rate exceeds a certain limit. The paper acknowledges that mechanisms like TCP Snoop are much more powerful than end-to-end recovery mechanisms as it isolates and corrects the loss at the location of the loss.

6. TCP-Aware

The TCP receiver in [11] is designed to distinguish the cause of a packet loss based on inter-arrival times between packets. The protocol works in case the last hop in the network is a wireless link and is the bottleneck link in the network. The authors argue that making changes in the intermediate routers may not always be a good idea. With cumulative acknowledgements used by TCP, the sender does not exactly know which packets are lost. The receiver may have a better idea about which packets are lost and the cause of the loss. Using this notion, the paper discusses an algorithm at the receiver that estimates the cause of the packet loss, either due to congestion or channel errors in the wireless link. TCP Reno is modified to include this scheme and is known as TCP-Aware.

According to this paper, if T_{min} is the minimum inter-arrival time observed so far by the receiver and T_g denotes the time between an out of sequence packet and the last in-sequence packet, and if

$$(n+1)T_{\min} \le T_g < (n+2)T_{\min}$$

then the *n* missing packets are assumed to be lost due to wireless transmission errors. The experimental results verify the accuracy in discriminating a congestion loss from a channel loss.

On comparing TCP-Aware with TCP-Reno and Ideal TCP, TCP-Aware is shown to perform well when the wireless channel loss is not high and congestion is not heavy.

7. Delayed Congestion Response Protocols

According to [12], a new class of protocols is presented which intend to deliberately delay the response to congestion in a network. The idea is to let this delay serve as an early signal to multimedia applications to make necessary adjustments for impending loss in bandwidth. The application can reduce its sending rate, thereby reduce the number of packets in the network, and subsequently, reduce the congestion in the network. The sender can respond to congestion after a certain time interval expires. The author presents three cases, for which analytical models are developed and conditions are derived under which these protocols can be fair to TCP.

The first case called DCR-I has the congestion window at the TCP sender continuously increasing for a time interval of τ RTT, where τ is an integer constant. After the interval expires the window drops by a constant factor γ. The second case, DCR-D, involves the congestion window to be reduced gradually over a time period of τ RTT, while in the third case, DCR-C, the congestion window is kept constant at the value it had reached just before the packet drop indication occurred. According to the analytical results DCR-C cannot compete fairly with TCP, where as DCR-I and DCR-D show appropriate TCP friendliness. The author also provides results showing the smoothness of the protocols and their responses to dynamic traffic patterns.

This research provides a framework for a new class of protocols. DCR-I with some of the other protocols discussed above can yield different mechanisms to control the congestion window during wireless channel losses. The fairness index of these

protocols is improved with RED. However further work needs to be done in choosing the increasing and/or decreasing functions and parameters like γ and τ.

B. MOTIVATION

Internet traffic due to wireless devices has increased exponentially. The delays associated with wireless networks can be very high - in the range of 300 ms for satellite networks. Moreover, the wireless links are generally unpredictable and fragile. As a result corruption of packets on these links is common. The transport layer at the receiver may receive successive packets after a packet loss due to wireless channel errors. The receiver's TCP layer sends acknowledgements back to the sender for the successively received packets. TCP sends cumulative acknowledgements indicating the next expected packet in sequence. In this case, the acknowledgements corresponding to the packets beyond the lost packet will contain the lost packet number as the next expected sequence number. TCP sender on receipt of three such duplicate acknowledgements realizes that a packet is lost and retransmits it. The sender assumes this loss to be a result of congestion and accordingly reduces its congestion window by half. Meanwhile, the wireless base station retransmits the lost packet based on the acknowledgements at the link layer. Thus the recovery of the lost packet could potentially take place at two layers in the network stack. Most of the research done on TCP protocols for wireless networks does not consider the ability of the base station to perform link-level retransmission. The losses on the wireless links can be retrieved using this mechanism. However, since the duplicate acknowledgements reach the sender before, the sender reduces its congestion

window. This unnecessary reduction of the congestion window severely hampers the throughput of the network.

Imagine a scenario, where the traffic is generated from a single fixed host with regular TCP to a satellite terminal with no cross traffic that can result in congestion. The maximum congestion window size of the TCP sender is kept within bounds of the bottleneck bandwidth of the network to avoid congestion. If the packets are dropped just because of the wireless channel errors, the network throughput is significantly reduced. This is due to the TCP sender reducing its transmission rate on detecting any packet loss and the resulting delay in building the congestion window back to its original level. If we are able to prevent this reduction, and at the same time use the link level retransmission capability of the base station to successfully recover the lost packets, then we will have a network that can provide us with better throughput. Such a scheme must also be able to handle actual congestion losses. If the network is experiencing congestion, the network output should not be worse than that of existing TCP protocols. Moreover the changes involved should be as least as possible so that they can be easily combined with existing TCP implementations.

In this thesis we discuss delayed congestion response (DCR) protocol, which intentionally delays the response to any packet loss. During this delay, the base station can perform successful link level retransmission. According to Fig. 1, if the acknowledgement from the wireless receiver arrives within the delay period T_d, then the TCP sender does not need to reduce the transmission rate since the packet lost due to channel error is already successfully recovered. If an acknowledgement does not arrive

within this delay period, the TCP sender can conclude that the packet loss may be due to congestion and respond accordingly to deal with congestion losses. We investigate DCR protocols by performing simulations under various conditions of congestion and channel errors, and, present their results. We investigate the TCP-friendliness of these protocols for possible deployment in conjunction with existing TCP hosts.

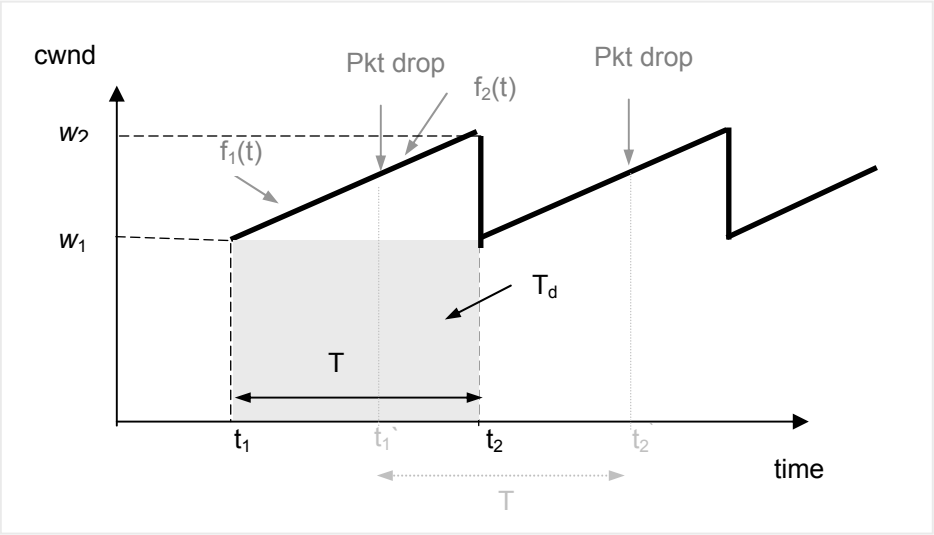

Fig. 1. DCR Behavior on Packet Loss

C. ORGANIZATION

The next few sections are organized as follows. In Chapter II, we study the need for delayed response protocols and discuss the DCR protocol where we delay both the packet retransmission and congestion window reduction. In Chapter III, we analyze and evaluate the results of DCR. In Chapter IV, we present the conclusions and future work.

CHAPTER II

DELAYED RESPONSE PROTOCOLS

A. NEED FOR DELAYED RESPONSE PROTOCOLS

Current research done for TCP involves reducing the transmission rate when a packet loss is detected. Typical TCP protocols like TCP-Reno assume that the cause of every packet loss is congestion. When wireless devices were not common, the ratio of packet losses due to congestion to that due to channel errors was very high. Hence, making such an assumption about the cause of a packet loss was valid. With the increase in the number of wireless devices and wider deployment of wireless networks, the channel errors are becoming more common. A packet is said to be lost due to channel errors when either the packet arrives with its header sequence disordered, or the received packet's check sequence does not agree with the received frame due to noise or other reasons.

In such networks, the TCP sender should not always reduce its transmission rate, when it has experienced a packet loss. Existing base stations have the ability to perform link-level retransmission. The link layer in the base station through link-level acknowledgements can sense that a packet has been lost on the wireless channel and can retransmit this lost packet. Hence a TCP protocol is needed which is aware of such link-level retransmission schemes, and accordingly takes appropriate decisions regarding response to any packet loss.

At the same time, the protocol should respond appropriately when there is actual congestion in the network. The performance of the protocol should not be worse than that of existing TCP protocols like TCP-Reno or TCP-Sack even at high congestion rates. The protocol should avoid violating the "end-to-end semantics" notion of network communication, i.e., the TCP connection should not be broken in between. Moreover, the protocol should be easy to implement and understand. Delaying the response to a packet loss at the sender is a solution that encompasses the requirements presented above and is the central focus of this research.

B. THE DCR PROTOCOL

The research done in this thesis, takes its inspiration from [12] that proposes delayed congestion response protocols. According to the author, the TCP sources retransmit a lost packet but do not reduce the congestion window immediately after the receipt of the notification of the lost packet. The TCP sources wait for a period of τ RTT, where τ is an integer, allowing the application to reduce the number of packets to be sent into the network and subsequently reducing the impact of congestion. The congestion window can increase, decrease or stay the same during this period. When the timer expires, the protocol adjusts the congestion window to its appropriate size.

The idea behind the current research in this thesis is to delay the response to any packet loss by τ RTT. This gives the base station sufficient time to perform link-level retransmission if the packet is lost due to wireless channel errors. In the mean time, the sender continues to increase its congestion window as if no packet loss was detected.

After the acknowledgement for the lost packet reaches the sender (recovered due to link-level retransmissions) it will realize that the detected packet loss was due to wireless channel errors and therefore not take any congestion control action for that packet loss. If the packet loss is because of congestion then the sender will retransmit the packet when the delay expires due to non-receipt of the required acknowledgment. The sender will also reduce the congestion window from its current value as a measure of congestion control. Reducing the congestion window after a delay will have no significant effect on the goodput of the network. In summary, the protocol assumes that the base station of the wireless destination node is capable of performing link level retransmission and therefore does not respond to any packets lost in the wireless link by reducing its transmission rate.

This protocol, known as Delayed Congestion Response Protocol (DCR), can be very effective in satellite networks that generally have large delays in the wireless links. In such networks, the round-trip times (RTT) are very high, resulting in slow additive increase in TCP congestion window. In TCP-Reno if there is a packet loss in the wireless channel, the congestion window is reduced to half of its current value. With such high RTT, it takes a long time for the congestion window to restore back to the original value before the packet loss. This results in significant loss of performance. Moreover, the same packet would be retransmitted twice which is unnecessary. With its loss response mechanism, DCR is likely to provide full network bandwidth if the network has only channel errors.

Hence, DCR is likely to be useful in networks having high delay, high errors and capable of link level retransmission. As the protocol does not retransmit until the delay has expired, in case of channel errors, the lost packet will be retransmitted only once at the base station's link-layer. The end-to-end semantics of TCP are not broken, as there is no break of the TCP connection at the base station. The implementation requires minor changes in existing TCP/IP protocol suites. Analyses in [12] have shown that delayed congestion response protocols are TCP-friendly. Hence, the current protocol is fair to other TCP sources. Detailed analysis and evaluation of the protocol is provided in the subsequent chapter along with its design and implementation.

CHAPTER III

EVALUATION OF DCR PROTOCOL

A. ANALYSIS OF DCR

According to DCR, if a packet loss is because of channel errors, the congestion window at the sender will continue to increase. As long as there is no congestion in the network, the sender can transmit packets to fill up the bottleneck link of the network. The base station of the wireless receiver takes care of the packets lost due to channel errors. Fig. 2

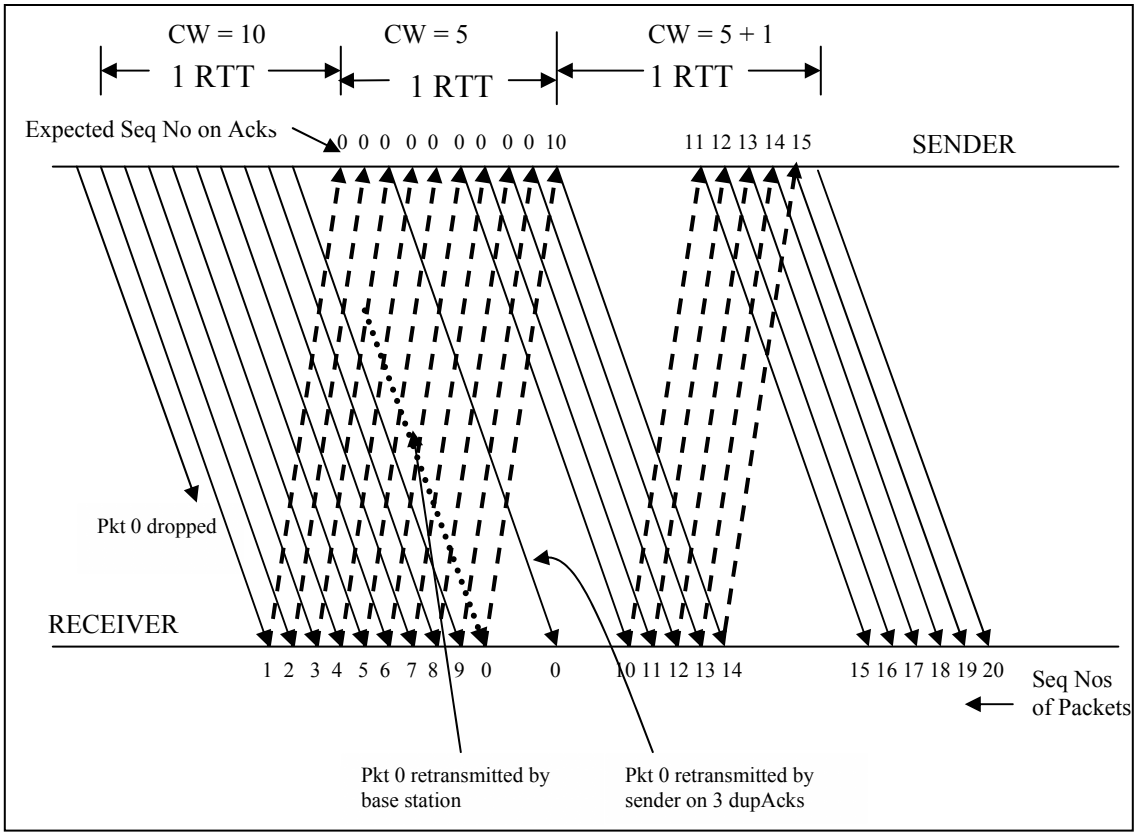

Fig. 2. Packet Traces for TCP-Reno - 1 Packet Lost due to Channel Error

shows how TCP-Reno reacts to a channel error in an example scenario. Packet 0 is dropped in the wireless channel and the base station retransmits this packet using link-level retransmission mechanism. However the TCP-Reno sender on receiving 3 duplicate acknowledgements assumes congestion and retransmits the same packet again. Moreover, it reduces the congestion window to half (=5) of the original window (=10).

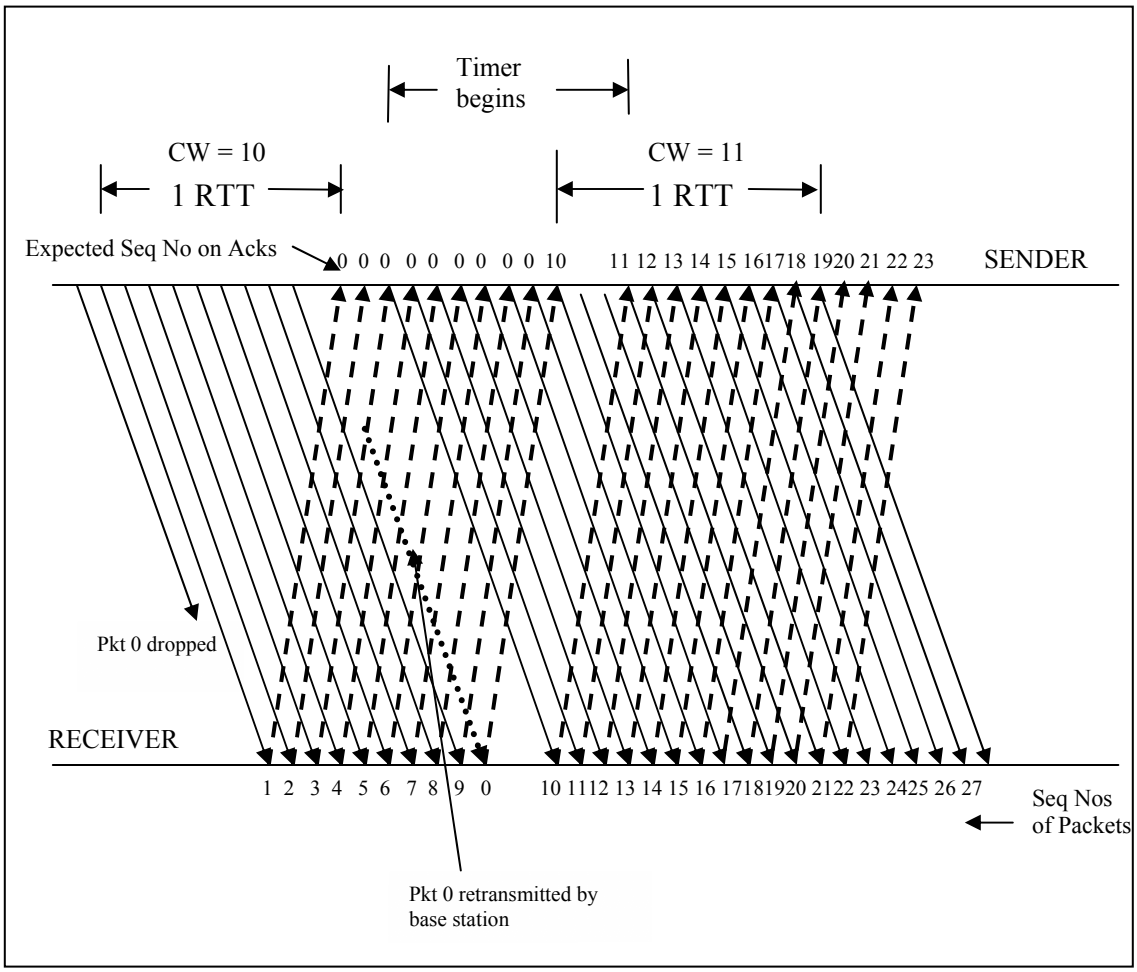

Fig. 3. Packet Traces for DCR - 1 Packet Lost due to Channel Error

Fig. 3 shows the packet traces in the same network situation, but this time with a DCR sender. The DCR sender on receiving 3 duplicate acknowledgements delays the reduction of its congestion window. As the base station successfully retransmits within the delay, the sender receives an acknowledgement within the delayed timer. The DCR sender increases the congestion window and continues transmitting packets. Comparing the traces with Fig. 2, the difference in the total number of packets in the network can be observed.

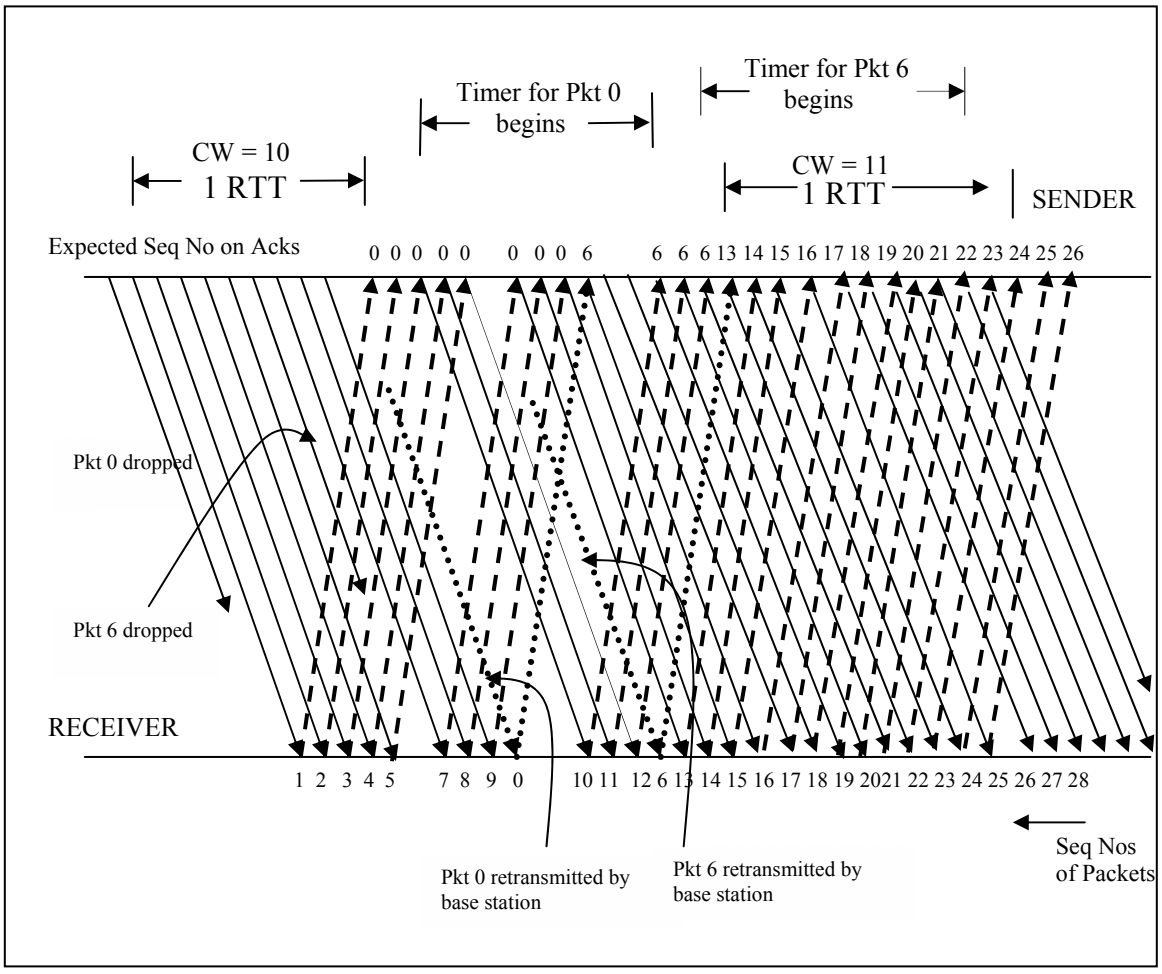

Fig. 4. Packet Traces for DCR - 2 Packets Lost due to Channel Error

Fig. 4 shows the packet traces for a DCR sender if 2 packets are dropped in the same window because of channel errors. Since the cumulative acknowledgements of both the lost packets arrive before their respective timers expire, the DCR sender does not need to reduce its congestion window. As long as there is no congestion in the network, it can grow its congestion window. On the other hand TCP-Reno would have reduced its congestion window twice for both packet losses, further reducing its throughput.

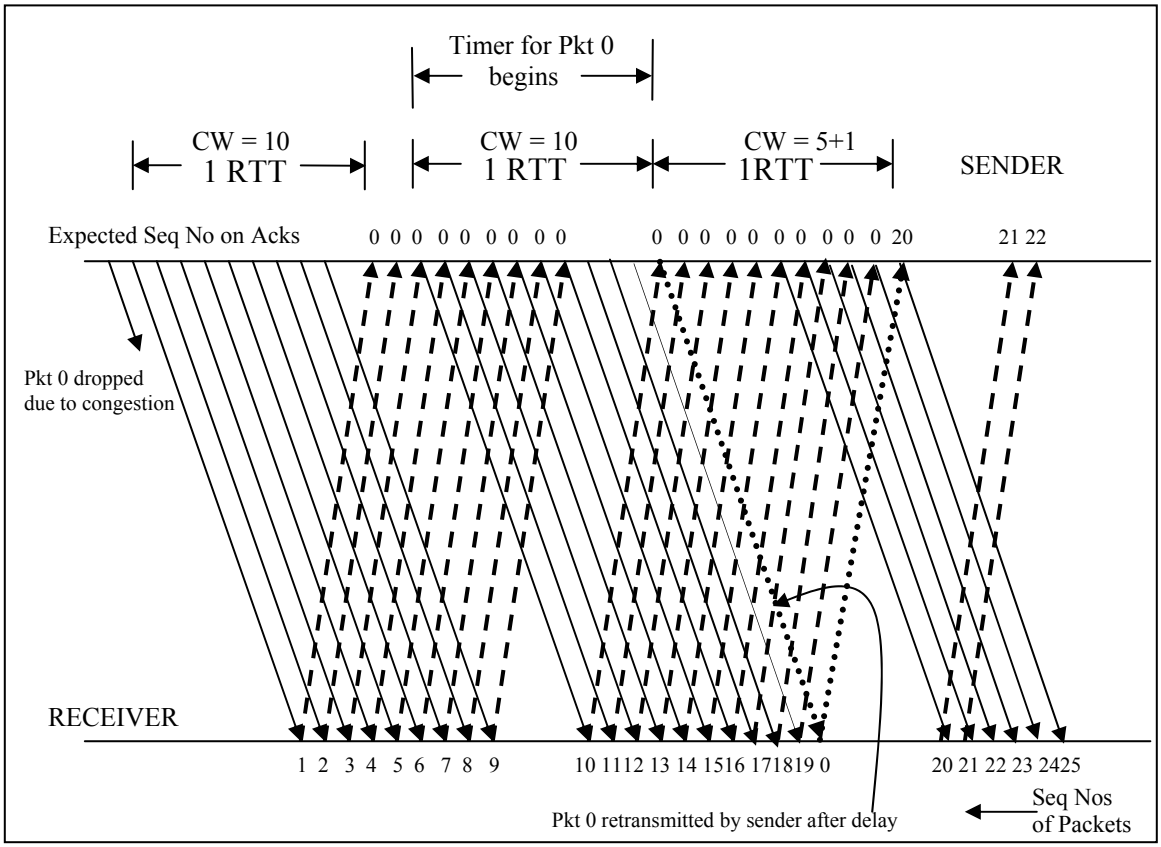

Fig. 5. Packet Traces for DCR - 1 Packet Lost due to Congestion

The above packet traces show that as long as we have packet losses due to channel errors, DCR transmits more packets into the network than TCP-Reno. Fig. 5 shows the packet traces for DCR for a congestion loss. The retransmission of the lost packet takes place after the timer expires. This means that the receiver should have a buffer capacity of 2 * CW_{max} to hold the packets before receiving the lost packet from the sender. Experimental results below show that such a delay in retransmission is not detrimental. Moreover, DCR does not change the timeout mechanism of TCP for detecting lost packets. Arrival of acknowledgements or duplicate acknowledgements resets the timeout timers.

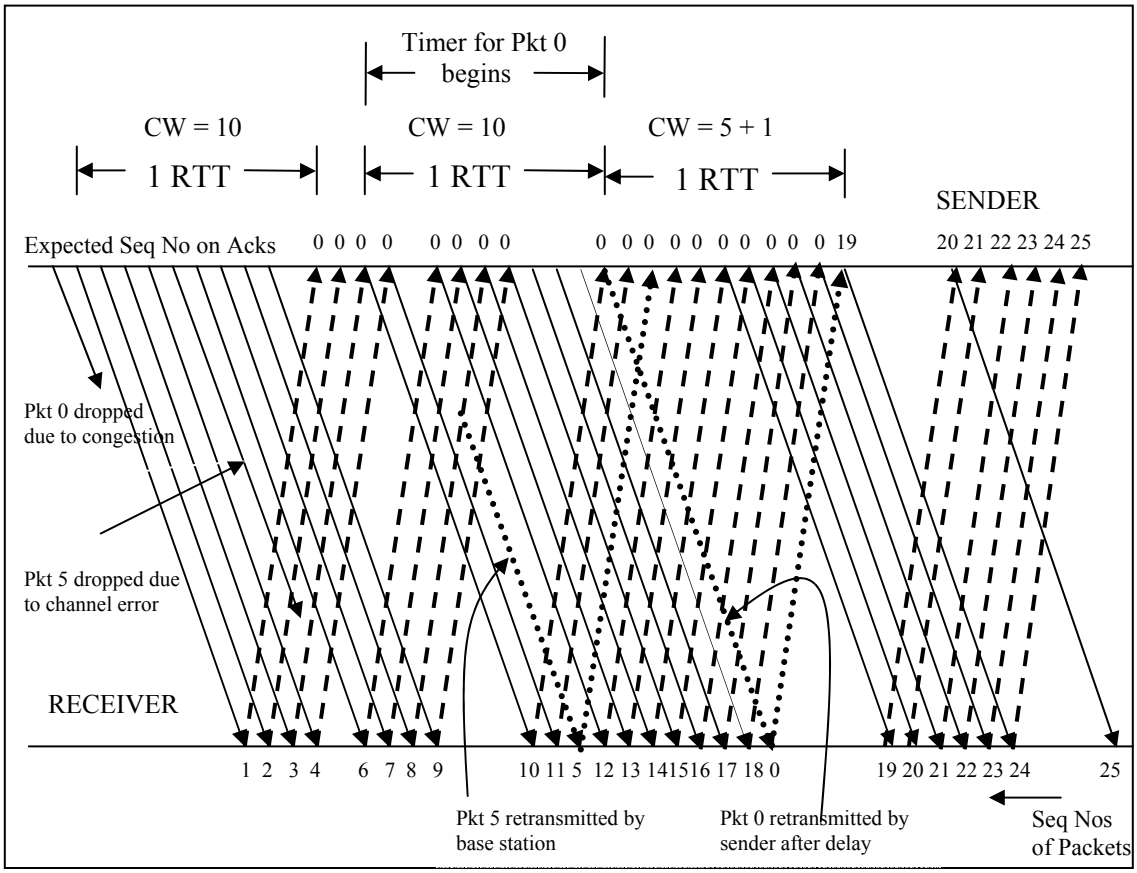

Fig. 6. Packet Traces for DCR - 1 Congestion Loss and 1 Channel Error Loss

Fig. 6 shows the packet traces of DCR, if the network has one congestion loss and one loss due to channel errors. For the congestion loss the packet is retransmitted after the delay expires, whereas base station retransmission takes care of the packet loss on wireless channel. In this case the congestion window needs to be reduced only once with DCR, unlike with TCP-Reno. This shows that DCR can handle packet losses of different nature.

B. IMPLEMENTATION ISSUES

To study the results of delayed response protocols, simulations were performed using ns-2 simulator from Berkeley [13], version 2.1b6. Most of the simulations performed were conducted with the dumb-bell topology as shown in Fig. 7. The wired and wireless environment was created using the wired framework provided by ns-2, setting the wireless part of the network with low bandwidth and high delays. The sources were connected to a router R1 using high bandwidth and low delay links, which in turn was connected to a base station BS-0. Since a wireless channel is a single medium, with no point-to-point links, BS-0 was connected to BS-1 to represent the bottleneck wireless link of low bandwidth and high delay. BS-1 was connected to the remaining supposed wireless nodes using high bandwidth-low delay links.

The routers were configured to have DropTail buffer management. Each *flow* consisted of packets along the link from *Src i* to *Sink i,* i.e. from wired medium to wireless medium. Unless specified, all the simulations in this research were conducted using the above topology with varying values of wired bandwidths and delays, as well as

bottleneck bandwidth and delay. Most of the simulations were conducted for 400 seconds and the results are collected after the first 200 seconds, allowing the simulated network to get into steady state. The window size of each TCP sender was adjusted so that the bottleneck bandwidth was completely utilized. The rate of errors on the wireless channel was configured on the bottleneck link. The value of τ was also specified at each DCR source.

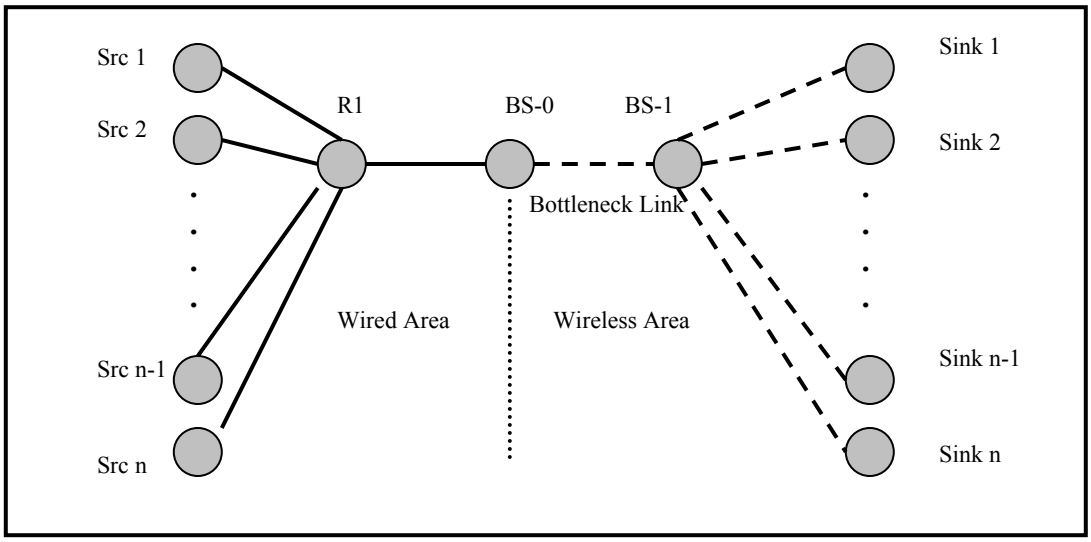

Fig. 7. Network Topology for the Simulations

C. SIMULATION SOFTWARE DESCRIPTION

The network simulator ns-2 was modified to include separate classes to handle the DCR protocol. The original source code of TCP-Reno was modified to build the new protocol. The class Agent/TCP/Dcr was added to support the DCR type senders. In this class the recv() method was changed to incorporate the delay in responding to a packet loss. On

detecting 3 successive duplicate acknowledgements, the time was stored in an array and the congestion window was increased as if nothing had happened. The array is indexed by two indices, *index-in* to keep track of the most recent lost packet and *index-out* to keep track of least recent packet whose loss response is pending. Whenever the recv() method was called on receiving an acknowledgement, if any packet loss response actions were pending, then the current time was compared with the value at *index-out*. If the difference was greater than τ RTT, then the congestion response actions, involving reduction of transmission rate and retransmission of lost packet were performed. Otherwise, if the current acknowledgement had a sequence number larger than that of the last packet acknowledged by the sender, then the timer value at position *index-out* was cancelled. This indicated successful link-level retransmission by the base station. In this way, DCR eliminated retransmission or window reduction if the packet loss was because of channel errors.

The ns- 2.1b6 version does not support any kind of link-level retransmission. Therefore a class named ErrorModel/LLRtx was added to support this feature. This class dropped packets with a probability based on the error rate specified at the bottleneck link. Before dropping the packet, it scheduled the packet on a timer, whose delay was set to twice the delay on the bottleneck link. When the timer expired, the packet was output on the bottleneck link. This is how link-level retransmission was simulated. It was assumed that this retransmission would never fail.

D. PERFORMANCE EVALUATION OF DCR

As explained in Chapter II, the ns-2 simulator was modified to incorporate the DCR protocol as a TCP Agent. Most of the simulations were performed using the topology used in Fig. 7., except otherwise mentioned. In most of the simulations, we compare DCR performance against TCP-Reno. In all simulations error rate refers to the wireless channel error rate. The settings for simulations involving DCR are kept the same as for TCP-Reno to make fair comparisons

1. Goodput v/s Congestion Response Delay

In this simulation, we investigate the performance of DCR for varying values of congestion response delay. Each curve represents a different value of wireless delay, which ranges from 30ms to 300ms. Congestion response delay of zero represents TCP-Reno. The experiments were carried out with single and multiple sources of the same protocol. The wireless bandwidths of 0.5Mbps and 1 Mbps were simulated respectively. The error rate was kept fixed at 5% for all experiments. The wired bandwidth and delay were kept at 0.5Mbps (or 1Mbps) and 3ms respectively. The network was configured such that there was no congestion on the bottleneck link.

The results presented in Fig. 8 shows that DCR performs better than TCP-Reno. The results also show that larger values of τ (>1) don't offer any significant improvement compared to the case when $\tau = 1$. TCP-Reno results are straight lines parallel to the X-axis, values being the same as that of DCR at $\tau = 0$. Therefore they are

omitted from the graph for convenience. The results validate the assumption of DCR that when there is no congestion in the network, the base station can successfully retransmit the lost packets at the link layer. A delay of 1RTT at the sender in responding to the packet loss is sufficient to allow the base station to retransmit. It is observed that almost the complete bottleneck bandwidth is realized by DCR. TCP-Reno, on the other hand, assumes all packet drops due to congestion, and hence reduces its transmission rate every time it encounters a packet loss. The results in Fig. 8 show that higher the wireless delay the higher is the performance improvement with DCR. Fig. 9 shows similar results with 4 sources. We will only consider a congestion response delay of 1 RTT from now.

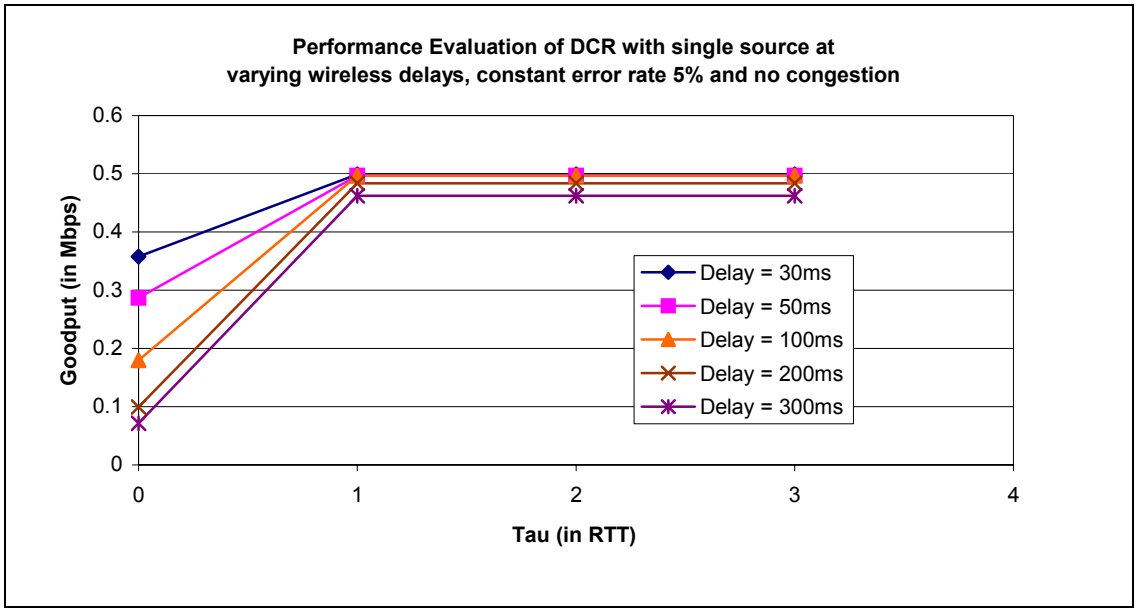

Fig. 8. DCR: Single Source, Constant Error Rate and No Congestion

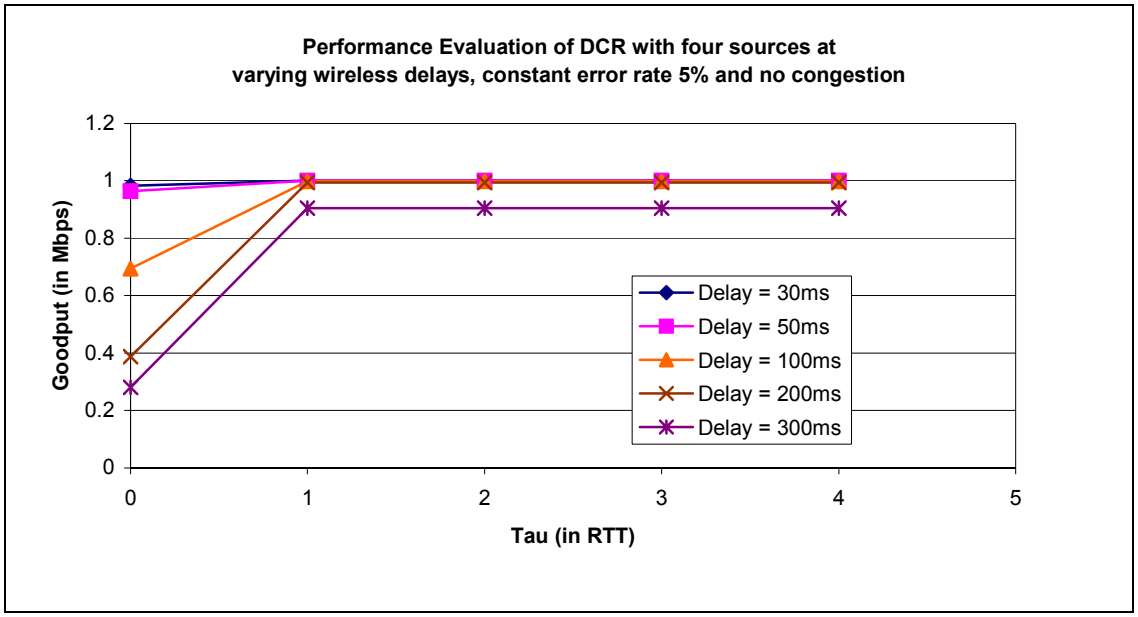

Fig. 9. DCR: Multiple Sources, Constant Error Rate and No Congestion

2. Goodput v/s Channel Error Rate

In this simulation, we investigate the performance of DCR and TCP-Reno against different channel error rates. The experiments were carried out with a single source and the wireless bandwidth was kept at 0.5Mbps. The wireless delay was kept fixed at 100ms for all experiments. The wired bandwidth and delay were kept at 0.5Mbps and 3ms respectively. The network was configured such that there was no congestion on the bottleneck link.

The results presented in Fig. 10, clearly show that DCR offers better throughput compared to TCP-Reno at all channel error rates. TCP-Reno's performance decreases sharply as the error rate increases. This is because it reduces its congestion window on

every packet loss. The results show that larger the wireless channel error rates the better is the performance with DCR.

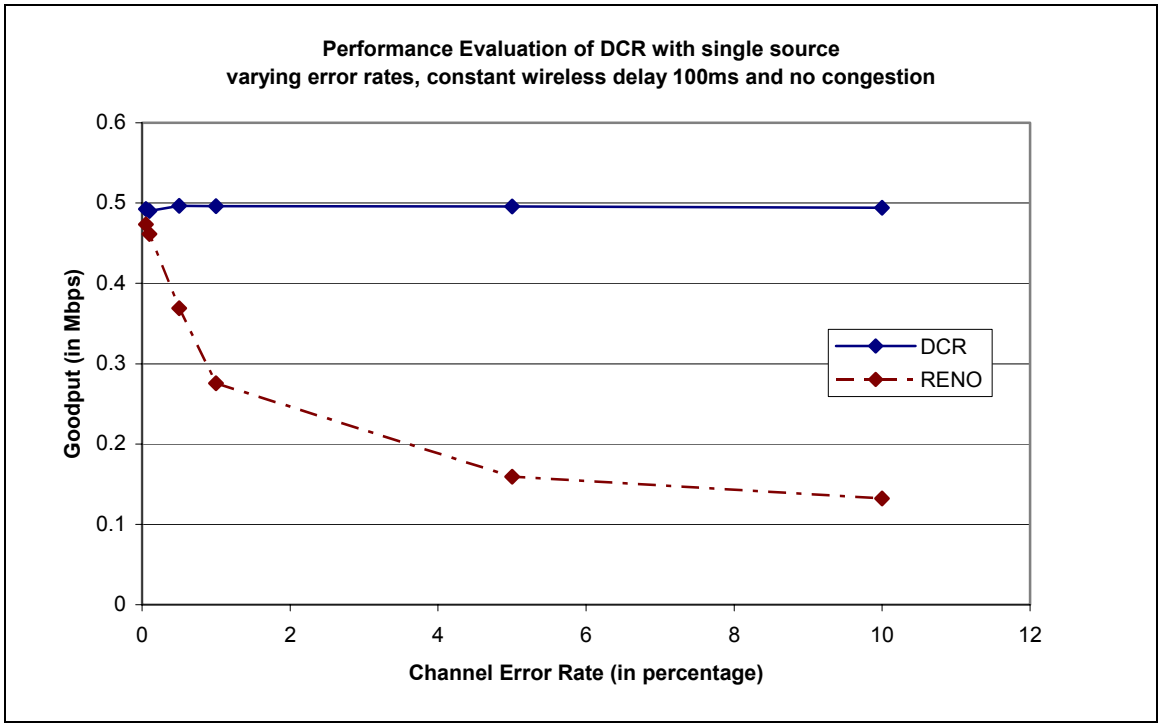

Fig. 10. DCR: Single Source, Constant Wireless Delay and No Congestion

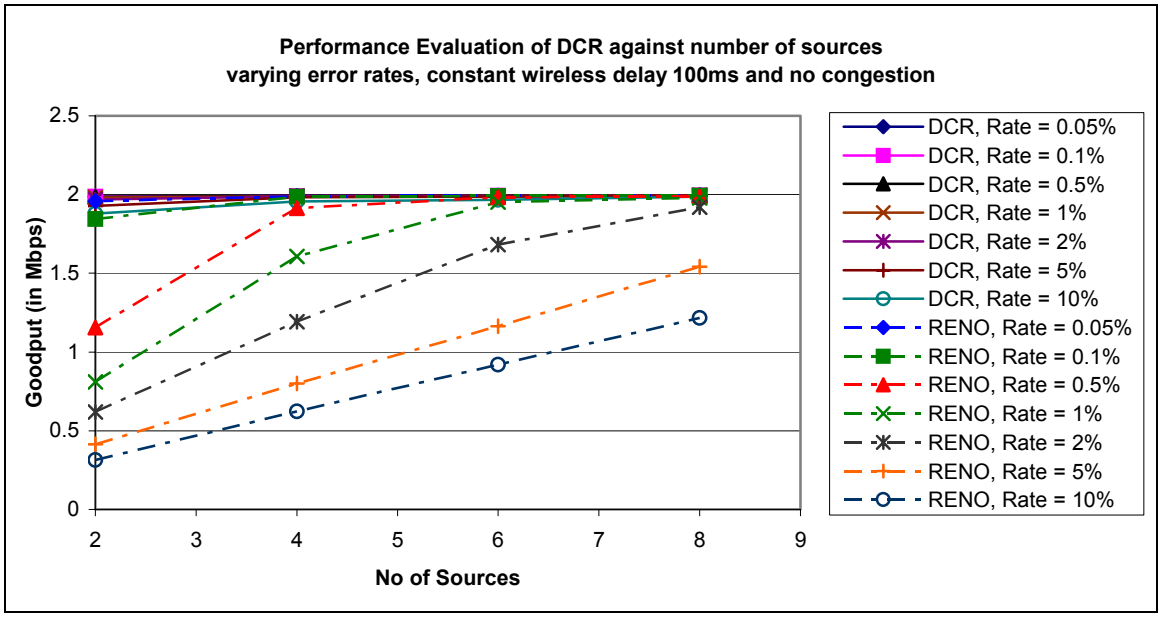

Fig. 11. DCR against Number of Sources - No Congestion

3. Goodput v/s Number of Sources with No Congestion

The aim of this simulation was to compare the performances of DCR and TCP-Reno by increasing the number of sources. Fig. 11 shows the performance of DCR and TCP-Reno at different error rates ranging from 0.05% to 10%. The wireless bandwidth was kept at 2Mbps. The wireless delay was kept fixed at 100ms for all experiments. The wired bandwidth and delay were kept fixed at 2Mbps and 1ms respectively. The total goodput was calculated by adding the individual goodput of each source.

It is observed that DCR sources realize better bottleneck throughput at any channel error rate. As the number of sources is increased, the throughput realized by TCP-Reno sources also increases. On encountering a packet loss, if one source backs

off, the other sources can increase their rate to fill the link. This explains the improved throughput of TCP-Reno as the number of sources is increased.

An analytical model for TCP throughput is developed in [14], which provides the equation for TCP throughput in terms of loss rate and RTT. According to this model, we can have a simplified equation for bandwidth utilization with packet size MSS, and drop probability of p, given by

$$BW = \frac{MSS \times \sqrt{3/2}}{RTT \times \sqrt{p}}$$

Each TCP-Reno source realizes BW bandwidth when channel errors are treated as congestion losses.

In a network with *n* sources, each source can have a fair share of bandwidth given by

$$B_{fair} = capacity / n$$

If $BW << B_{fair}$, then misconstruing channel error rates as congestion losses does not hurt TCP-Reno. As *n* increases, B_{fair} is reduced; allowing TCP-Reno to tolerate higher channel error rates without hurting the overall throughput. In case of DCR, it treats channel errors without reducing congestion window. Therefore it works well for all channel error rates.

4. Goodput v/s Wireless Delay

This simulation compares the performances of DCR and TCP-Reno against increasing wireless delay with no congestion at the bottleneck link. Each curve in Fig. 12 represents a different value of error rate on the wireless channel, ranging from 0.05% to 10%. The wireless bandwidth was kept at 0.5Mbps. The wired bandwidth and delay were kept fixed at 0.5Mbps and 3ms respectively.

The results in Fig. 12 show that TCP-Reno performs poorly as we increase the wireless delay. In satellite networks, where the delays are in the range of 300ms, TCP-Reno is not very efficient. On the other hand, DCR realizes almost complete bottleneck throughput at all the values of channel delays. As TCP-Reno halves its congestion window on any packet loss, it takes longer time to get back to the original congestion window at higher wireless delays. Hence, the throughput degrades as we increase the delay and the channel error rate. The results show that DCR offers improved benefits at higher delays and higher channel error rates.

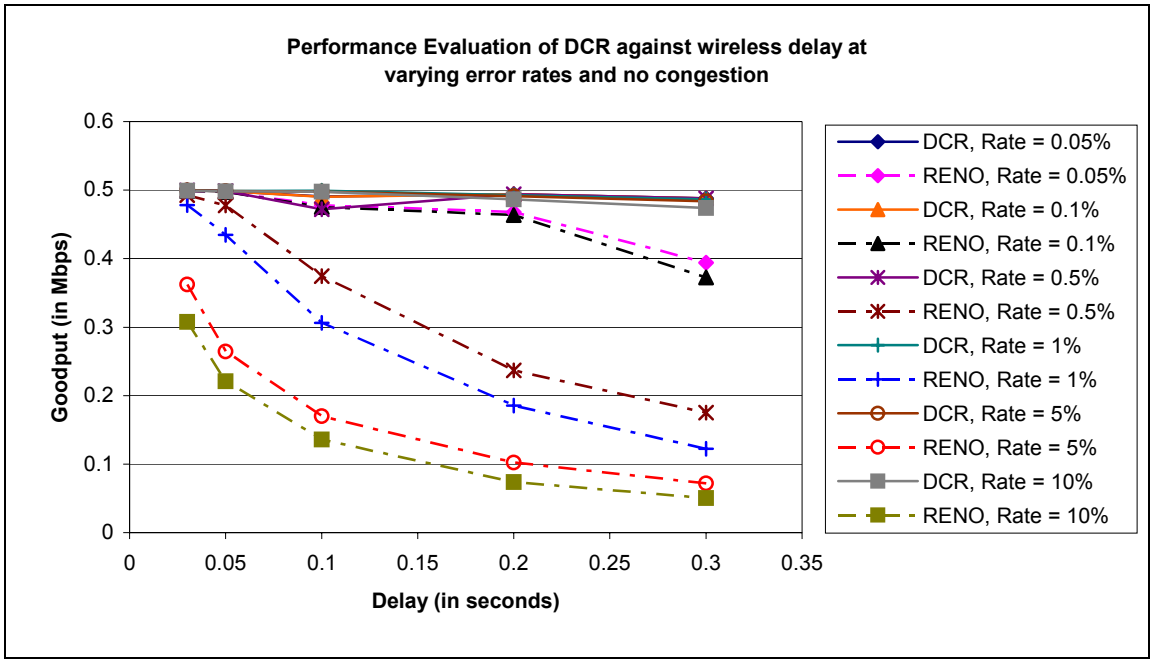

Fig. 12. DCR against Wireless Delay - No Congestion

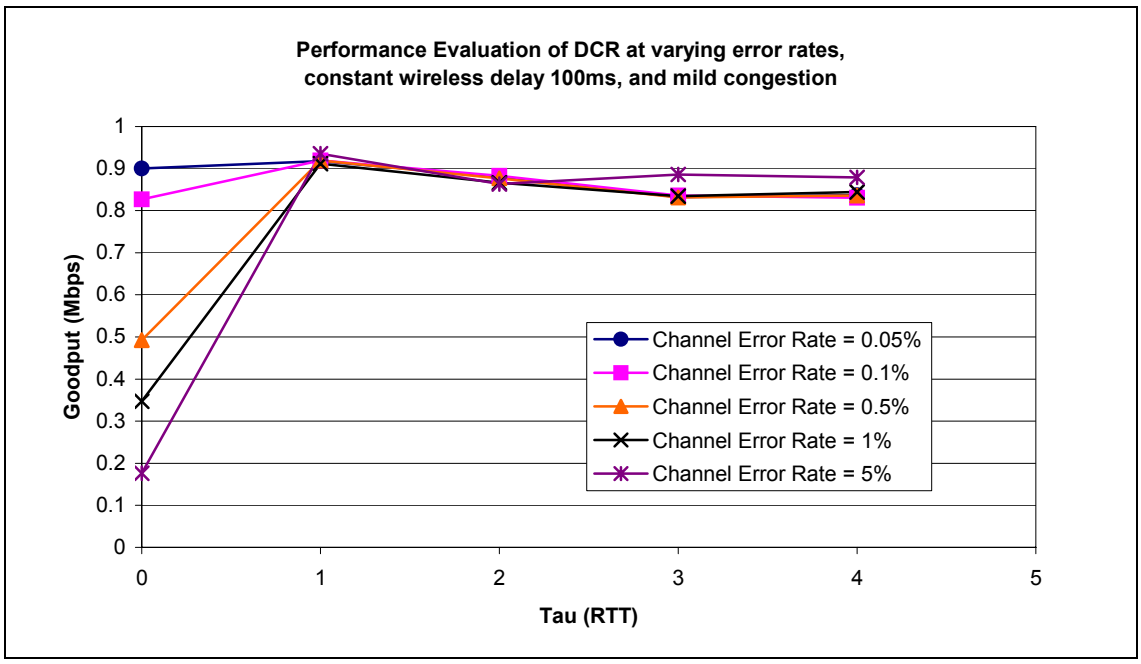

Fig. 13. DCR with Mild Congestion

5. Performance of DCR with Mild Congestion

It can be seen from the above results that DCR performs remarkably when there is no congestion in the network and all the packet losses are due to channel errors. In this simulation, performances of DCR and TCP-Reno are compared in presence of mild congestion. The congestion rate is measured as the number of drops experienced at the base station BS-0 to the number of packets reaching the base station. In the experiment, congestion rate varied from 0.1% to 0.5%. Each curve represents a different value of error rate on the wireless channel, ranging from 0.05% to 5%. The wireless bandwidth was kept at 1Mbps and the wireless delay was fixed at 100ms. The number of sources was kept constant at 8. The wired bandwidth and delay were kept at 10Mbps and 10ms respectively.

The results in Fig. 13 show that when the channel error rate is within the range of the congestion rate, the results of TCP-Reno are comparable with that of DCR. As the value of τ increases we see a small dip in the performance of DCR. This is because when the value of τ increases the delay in the response to congestion correspondingly increases. The base station cannot retransmit the packets lost due to congestion, and only the sender can retransmit these packets. As this retransmission delay at the sender grows, we see a minor drop in the throughput. Even with higher congestion response delays, DCR realizes at least 90% of the network throughput. When the wireless channel error rate is higher than congestion loss rate, we see that DCR performs better than TCP-Reno. Therefore DCR performance is observed to be better than that of TCP-Reno when the congestion is mild.

6. Performance of DCR under Heavy Congestion

In the previous section we saw how DCR reacts to mild congestion and still manages to provide a good performance. In the next set of simulations, we investigate the performance of DCR and TCP-Reno under heavy congestion. Each curve represents a different value of error rate on the wireless channel, ranging from 0.1% to 5%. The congestion rate was increased either by increasing the number of sources or by increasing the maximum window size. The congestion rate varied from 0.2% to 6%. Two sets of experiments were performed; one where number of sources was varied, and another where the number of sources was kept fixed at 15 but the maximum window size was changed to increase the congestion rate. The wireless bandwidth was kept at 1 Mbps and 2 Mbps respectively. The wireless delay was fixed at 100ms. The wired bandwidth and delay were kept fixed at 10Mbps and 1ms respectively.

Fig. 14 shows the results of the first experiment where the goodput of the network is plotted against the number of sources. As the number of sources increase, the performance of TCP-Reno almost reaches the complete network throughput, even in the presence of congestion. On the other hand DCR's performance is in the range of 90% of the total throughput with any number of sources. This minor drop can be attributed to the delay in responding to congestion. It is observed that DCR outperforms TCP-Reno at all values of wireless channel error rates when the number of sources is less than 10. TCP-Reno's improved performance with higher number of sources is explained earlier in 3. (Goodput v/s Number of Sources).

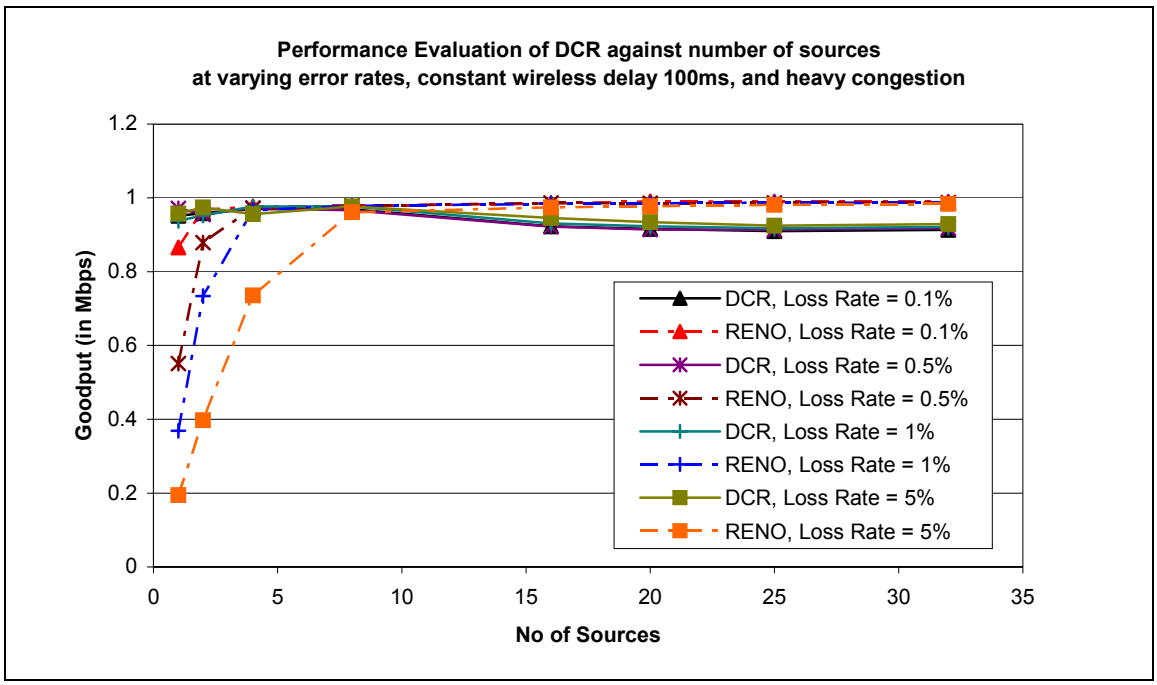

Fig. 14. DCR against Number of Sources - Heavy Congestion

Fig. 15 compares the performance of TCP-Reno and DCR when the number of sources is kept constant at 15 and changing the maximum window size of the sources varies the congestion rate. The graph further strengthens our claim that DCR performance in case of heavy congestion is comparable to TCP-Reno. When the channel error rates are higher than the congestion rate DCR does better than TCP-Reno, but the performance falls by about 5% when the congestion rate dominates. These two graphs show that the performance of DCR is comparable to TCP-Reno even in the presence of heavy congestion.

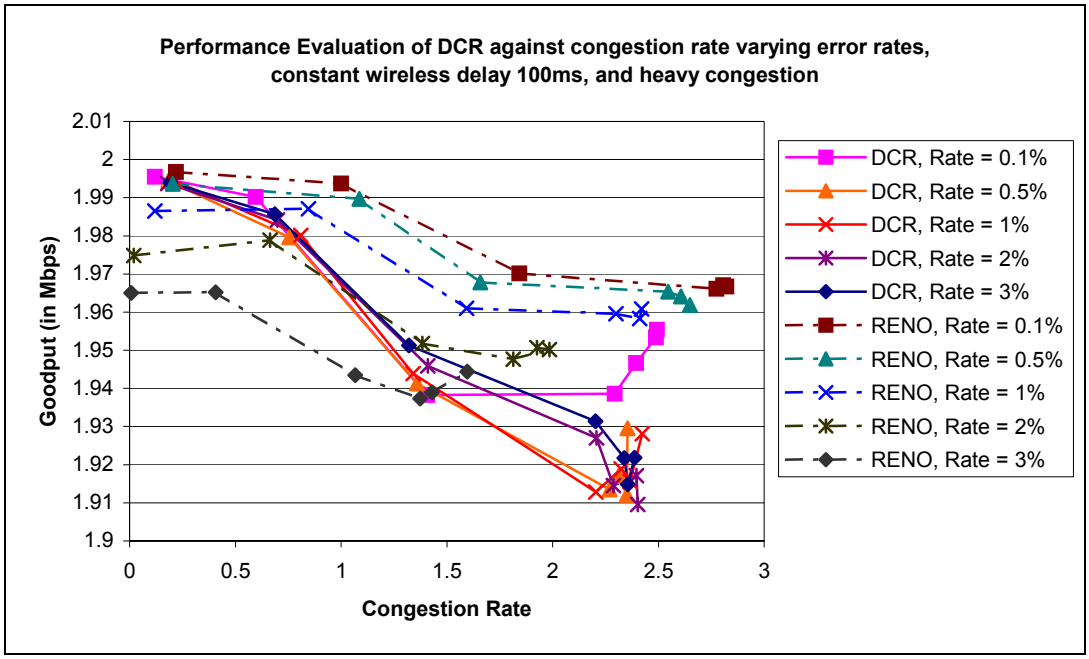

Fig. 15. DCR against Congestion Rate - Heavy Congestion

7. Performance of DCR in the Presence of Cross Traffic

The experiments conducted so far have all the sources connected to the router R1. In this experiment, we study the performance of DCR in presence of cross-traffic across the base station BS-0. The topology of the next two experiments is similar to Fig. 7 except that now there is additional cross-traffic links across BS-0. Fig. 16 shows the topology. Now the base station has many input queues but only one output queue. We study the effects of TCP and UDP cross traffic on DCR performance and compare with that of TCP-Reno in two separate experiments. The wireless channel error rate was varied from 0.5% to 5%. The wireless bandwidth was kept constant at 1Mbps and the wireless delay was kept fixed at 100ms for all experiments. The wired bandwidth and delay were kept

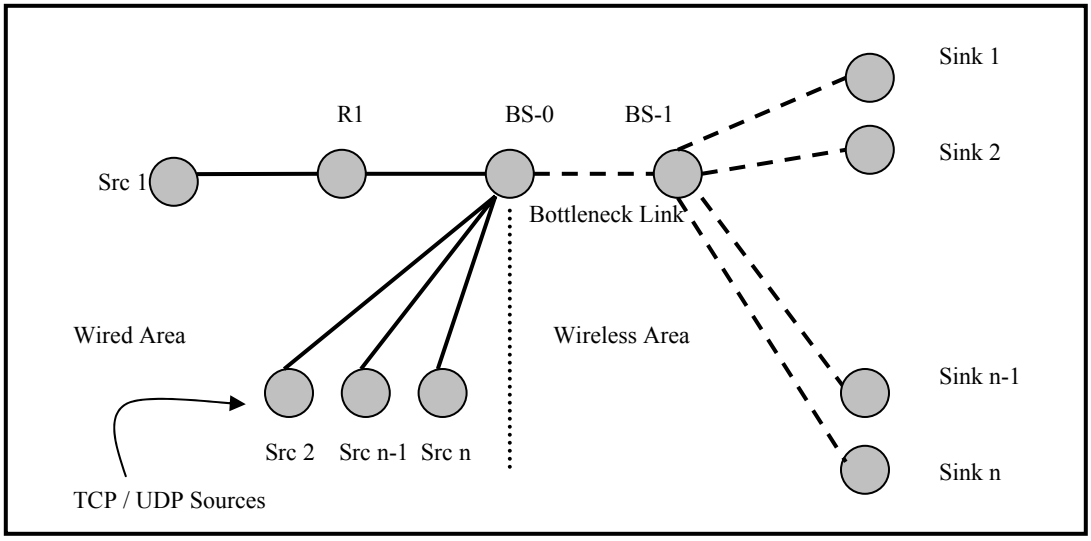

Fig. 16. Network Topology for Cross-Traffic Simulations

at 10Mbps and 1ms respectively. The TCP and UDP cross traffic caused congestion at a rate varying from 0.1% to 5% in both the experiments. The TCP cross traffic was TCP-Reno and the number of total nodes varied from 4 to 20. In case of UDP Cross traffic, we had 2 to 8 UDP Cross Traffic links, each having exponential arrival distribution with a burst time and idle time of 500ms. The rate of transmission was divided between the UDP senders to limit them from completely claiming the bottleneck link bandwidth.

Fig. 17 shows the results of DCR with TCP-Reno cross-traffic through the base station. It allows us to realize how much of the bottleneck bandwidth can a single DCR source utilize in the presence of congestion and channel errors with TCP cross traffic. The results show the usefulness of DCR when the congestion rate is lesser than channel error rate. DCR claims more than its expected share of bandwidth in comparison to TCP-Reno. This is mainly because TCP-Reno (reducing its congestion window for every packet loss) does not claim its share of bandwidth when channel error rates are high. As

the congestion rate increases (by increasing the number of sources), it is observed that under heavy congestion there is not much difference between TCP-Reno and DCR. The results illustrate the TCP-Friendly nature of DCR.

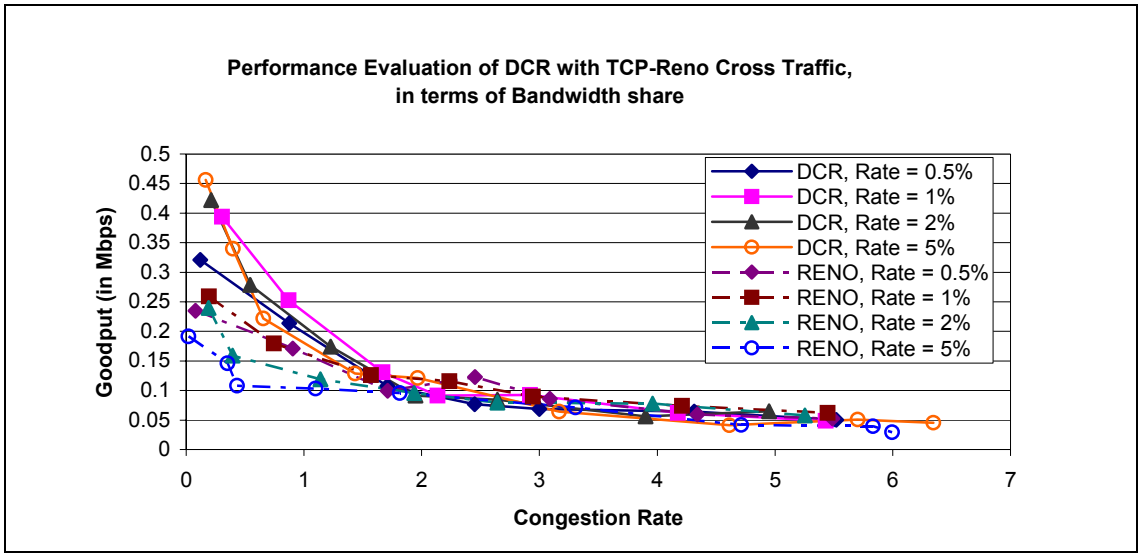

Fig. 17. DCR with TCP-Reno Cross Traffic

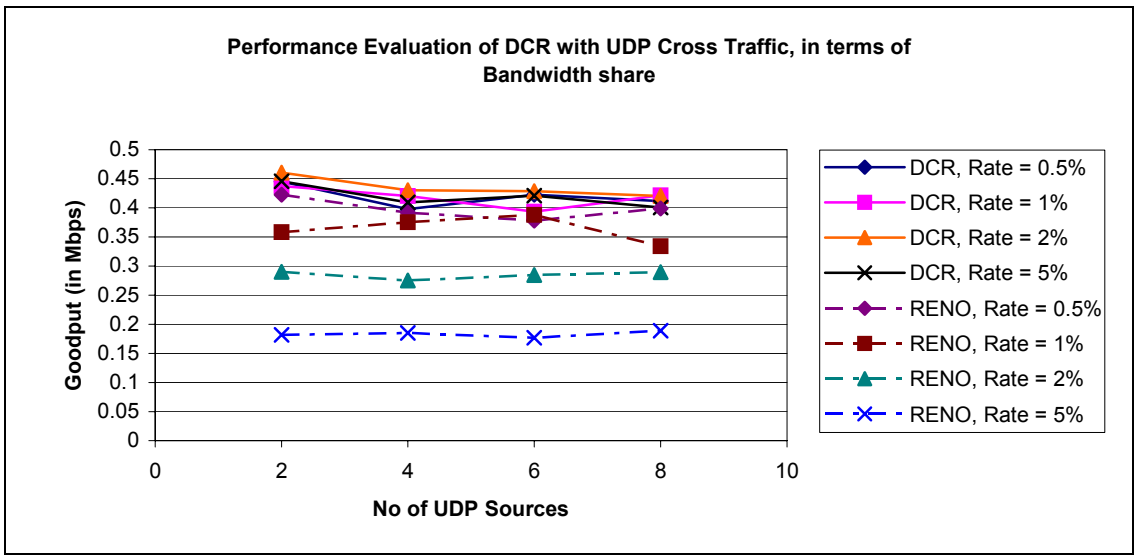

Fig. 18. DCR with UDP Cross Traffic

Fig. 18 shows the performance of DCR and TCP-Reno in the presence of UDP cross traffic. The UDP cross traffic forces higher congestion losses than the TCP cross traffic in Fig. 17. Even at this higher loss rate, DCR outperforms TCP-Reno. TCP-Reno needs to perform a packet loss recovery routine for every packet loss detected. DCR on the other hand needs to only handle the retransmission because of congestion, and therefore, tends to perform better.

8. Comparison of DCR with TCP-Westwood

As discussed in Chapter II, TCP-Westwood [10] is a sender-side modification of the TCP congestion window algorithm that results in significant improvement in wireless networks with lossy links. The experiments performed compare the performance of DCR and Westwood for increasing delays at varying wireless error rates. Each curve represents a different value of wireless error rate, ranging from 0.5% to 10%. The experiments were carried out with a single source. The wireless bandwidth was kept at 0.5Mbps. The wired bandwidth and delay were kept fixed at 0.5Mbps and 5ms respectively. The network was configured such that there is no congestion on the bottleneck link. The TCP Westwood source uses New Reno protocol with the faster recovery mechanism of Westwood.

The results of this experiment are presented in Fig. 19. The graphs give a clear indication that TCP-Westwood cannot match the performance of DCR. DCR sources provide almost full bottleneck bandwidth. Westwood, though much better than TCP-

Reno, loses more than half the throughput when the error rates are very high and the wireless delays reach the magnitude of 300ms. TCP-Westwood reduces its congestion window as a precautionary measure even after estimating the cause of packet loss as wireless channel loss. DCR avoids any window reduction on any wireless channel loss and therefore performs better for larger error rates.

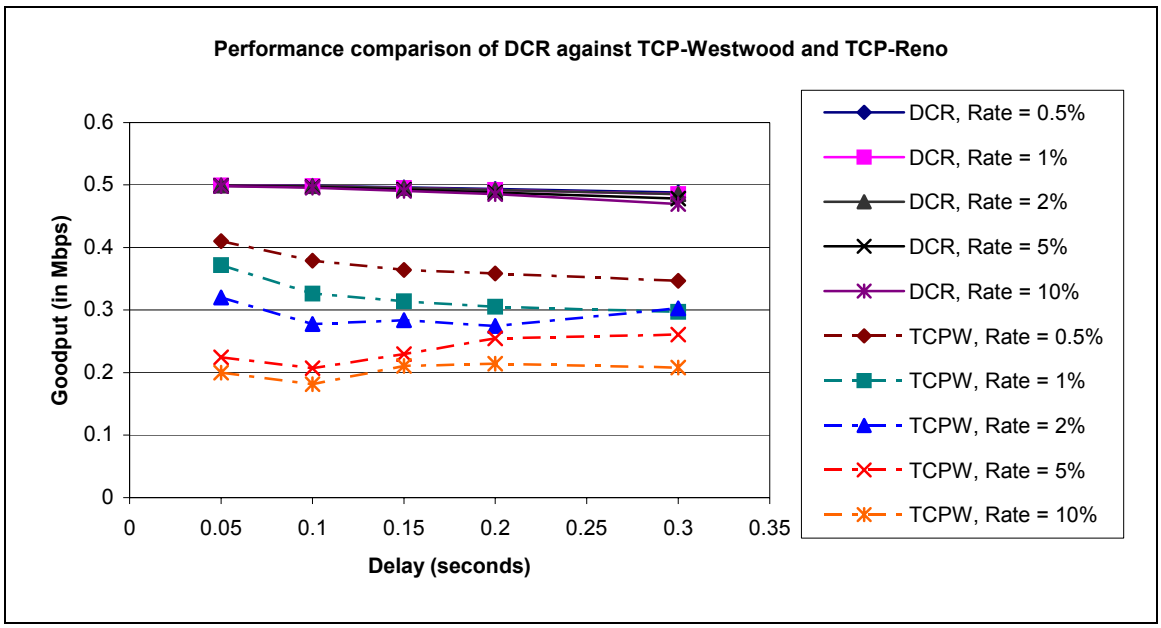

Fig. 19. DCR against TCP-Westwood

9. Comparison of DCR with TCP-Sack

This simulation compares the performances of DCR and TCP-Sack [15] against increasing wireless error rate with no congestion at the bottleneck link. The wireless bandwidth was kept at 10Mbps. The wireless delay was kept at 200ms. The results in Fig. 20 show that TCP-Sack performs poorly as we increase the error rate. In satellite

networks, where the delays are in the range of 300ms, such a protocol is not efficient. On the other hand, DCR gives almost complete bottleneck throughput at all the channel error rates. The objective of TCP-Sack is to overcome the problem of multiple packet losses in a single window. TCP Sack will retransmit and reduce its congestion window, if it does not receive the selective acknowledgements within a period. In high-error networks, by the time selective acknowledgement of the packet retransmitted by the base station reaches the sender, the sender has already reduced its transmission rate. Therefore, it degrades the performance as the error rate increases.

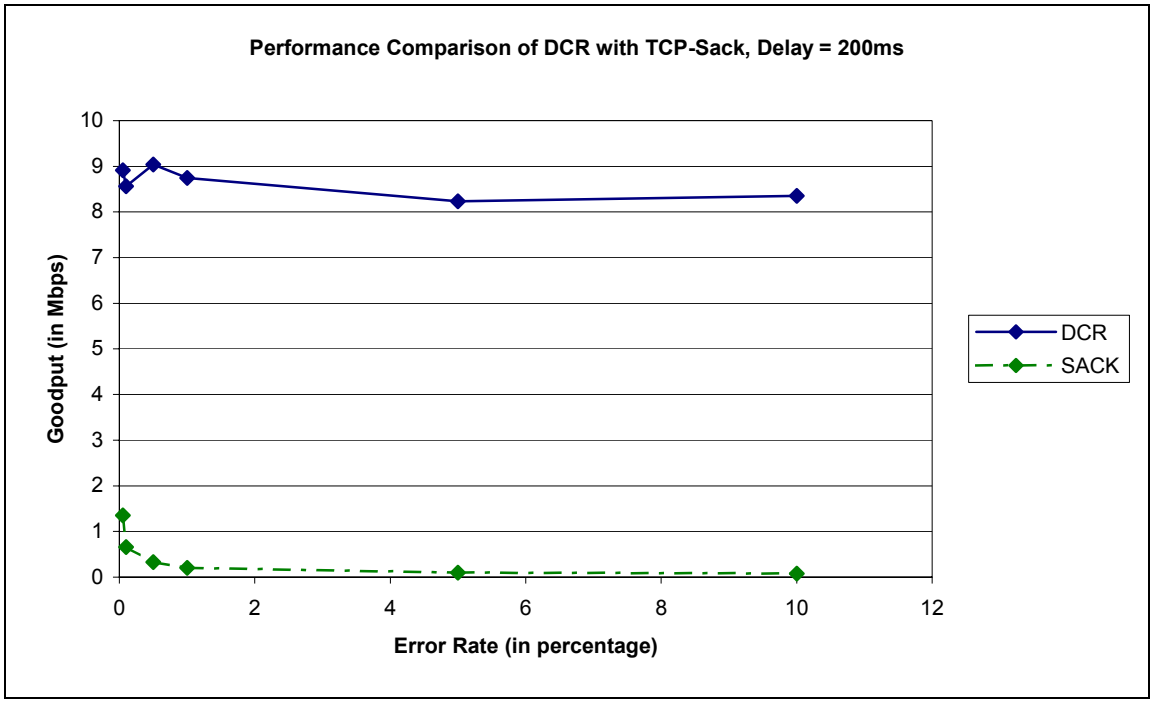

Fig. 20. DCR against TCP-Sack

10. Comparing DCR with Receiver-Based Schemes

Previous results showed above indicated that TCP-Reno lost bandwidth as soon as the sender received three or more duplicate acknowledgements (indicating a packet loss). The receiver could refrain from sending duplicate acknowledgements to allow link-level retransmission at the base station. Then the congestion window at the TCP sender will not reduce because of channel losses. The receiver could delay sending the duplicate acknowledgements for a time equal to the RTT of the wireless delay. A set of simulations was performed using such a receiver and a TCP-Reno sender.

The results in Fig. 21 indicate that as long as there is no congestion in the network, the difference in throughput between TCP-Reno and DCR is proportional to the error rate. Since the TCP-Reno does not receive any acknowledgements, the sender cannot transmit any packets during the delay. Therefore the number of packets that cannot be transmitted into the network by the sender is proportional to the number of packet drops on the wireless channel. On the other hand, DCR is able to send packets when it receives duplicate acknowledgements. Hence DCR realizes better performance.

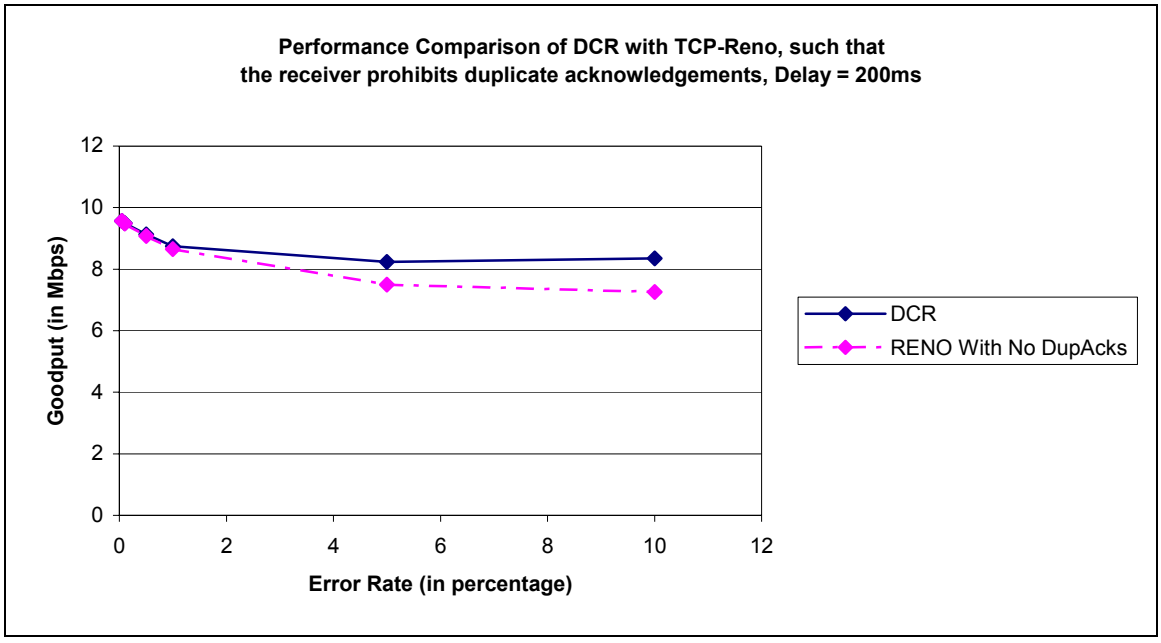

Fig. 21. DCR against TCP-Reno with No DupAcks

E. CONCLUSION

The above experiments show that DCR performs well when the network has no congestion. In the presence of congestion, DCR performs well when the congestion level is low. At higher congestion rates the performance suffers mildly. The performance advantage of DCR (compared to TCP-Reno) also reduces with higher number of sources. DCR proposes that when a network provides link-level retransmission at the base station, there is no need to perform immediate response to a packet loss. The results and analysis show that DCR obtains better network throughput, high-delay and high-loss networks.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

In this thesis, we have proposed a TCP protocol for tolerating wireless channel errors. We have discussed the delayed congestion response protocol (DCR) where the response to a packet loss is delayed for a period of τ RTT. DCR is easy to implement, follows the TCP end-to-end semantics and does not involve inter-layer communication. The results obtained showed that almost complete bottleneck throughput could be achieved in a network with up to 10% channel errors and 300ms wireless delay. DCR protocol can be very useful in satellite networks with high delay and high error rates.

The bandwidth estimation techniques provided in TCP-Westwood can be combined with DCR to distinguish between a congestion loss and a wireless channel loss. Successful channel error recovery rates can be used to estimate probability of channel errors versus congestion losses. This probability may be used for probabilistic retransmission of packets on duplicate acknowledgements in DCR. The parameters we chose in these protocols were heuristics aimed at keeping the implementation simple to understand. Further research needs to be done on parameters like the value of τ and the increasing function of the congestion window when a packet loss is detected. This would allow the protocol to work better under heavy congestion. The TCP-friendliness of DCR needs to be studied in more detail. Instead of depending on the RTT, a combination of ECN [4] and ETEN [6] can be used with DCR to make appropriate congestion control decisions. Research is also needed to take care of link-level retransmission failures.

REFERENCES

[1] J. Postel, "Transmission Control Protocol," Internet Engineering Task Force, *RFC 793*, September 1981.

[2] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of ACM SIGCOMM '88*, Stanford, CA, pp. 314-329, August 1988.

[3] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," Internet Engineering Task Force, *RFC 2581*, April 1999.

[4] S. Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review,* vol. 24, no. 5, pp. 10-24, October 1994.

[5] K.K. Ramakrishnan, and R. Jain, "A Binary Feedback Scheme for Congestion Avoidance," *ACM Transactions on Computer Systems*, vol. 8, no. 2, pp. 158-181, May 1990.

[6] R. Krishnan, M. Allman, C. Partridge and J. P. G. Sterbenz, "Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks," BBN Technologies*, Technical Report No. 8333*, March 2002.

[7] J. Postel, "Internet Control Message Protocol," Internet Engineering Task Force, *RFC 792*, September 1981.

[8] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "Improving TCP/IP Performance over Wireless Networks," in *Proceedings of ACM MOBICOM '95,* Berkeley, CA, pp. 2-11, November 1995.

[9] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Rutgers University*, Technical Report DWR-TR-314*, October 1994.

[10] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, and R. Wang, "TCP Westwood: End-to-end Bandwidth Estimation for Efficient Transport over Wired and Wireless Networks," in *Proceedings of ACM MOBICOM '2001*, Rome, Italy, pp. 287-297, July 2001.

[11] S. Biaz and N. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," Texas A&M University, *Technical Report 98-014*, June 1998.

[12] S. Bhandarkar, "Delayed Congestion Response Protocols," Texas A&M University, M.S. Thesis, August 2001.

[13] "ns-2 Network Simulator," *Dept. of EECS, University of California at Berkeley, CA.* Available at http://www.isi.edu/nsnam/. Accessed June 2001.

[14] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modelling TCP Throughput: A Simple Model and Its Empirical Validation," in *Proceedings of ACM SIGCOMM '98,* Vancouver, Canada, pp. 303-314, September 1998.

[15] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options", Internet Engineering Task Force, *RFC 2018*, October 1996.

VITA

Nauzad Erach Sadry was born on February 11th, 1977 in Bombay, India. He received his Bachelor of Engineering in computer engineering from Bombay University, India, in September 1999. He worked for 1 year at Mahindra British Telecom, Mumbai, India. In August 2000, he started his M.S. in computer science at Texas A&M University. His research at Texas A&M University has been focused on networking. His address is Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112.