Sub-segment based transport layer protocol for wireless medium

Som Sengupta

New Jersey Institute of Technology

ABSTRACT

A SUB-SEGMENT BASED TRANSPORT LAYER PROTOCOL FOR WIRELESS MEDIUM

by
Som Sengupta

This thesis discusses the techniques to improve the TCP over wireless. The loss-intensive wireless communication results in high retransmission rates to recover lost packets and bandwidth consumption. In addition, the retransmitted segments have significant chance of being dropped. To make the retransmission process more granular, large segments at the transport layer (540 bytes, by default) can be subdivided into smaller sub-segments. This document introduces a split TCP based solution and describes how to produce a series of smaller-sized segments that share the same transport layer header. A new header format is introduced to support the transmission of smaller segments and to conserve bandwidth during transmission. The NACK-based message exchange method is adopted with a special windowing protocol to achieve reliability, flow-control, and efficient buffer handling. The simulation results indicate that use of sub-segments improves latency, throughput, bandwidth, power and other measurements. In addition to that, an analytical model is developed to study the influence of sub-segmentation over latency and throughput.

A SUB-SEGMENT BASED TRANSPORT LAYER PROTOCOL FOR WIRELESS MEDIUM

by

Som Sengupta

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical Computer Engineering

May 2003

APPROVAL PAGE

A SUB-SEGMENT BASED TRANSPORT LAYER PROTOCOL FOR WIRELESS MEDIUM

Som Sengupta

[illegible] 5/9/2003

Dr. N. Ansari, Thesis Advisor — Date
Professor, Electrical and Computer Engineering, NJIT

[illegible] 5/9/2003

Dr. D. Karvelas, Committee Member — Date
Special Lecturer, Telecommunications, NJIT

[illegible] 5/9/03

Dr. S. Tekinay, Committee Member — Date
Assistant Professor, Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Som Sengupta
Degree:	Master of Science
Date:	May 2003
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education:

- Master of Science in Computer Engineering
  New Jersey Institute of Technology, Newark, NJ, 2003

- Bachelor of Science in Civil Engineering
  Jadavpur University, Calcutta, India, 1994

Major:	Computer Engineering

To my family

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to Professor Nirwan Ansari, who not only served as my research supervisor, providing valuable insights but also constantly gave encouragement and reassurance as well. Special thanks are given to Dr. Dionissios Karvelas and Dr. Sirin Tekinay for actively participating in my committee.

TABLE OF CONTENTS

TABLE OF CONTENTS

(Continued)

Chapter **Page**

TABLE OF CONTENTS

(Continued)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objectives

Wireless access to information systems and the Internet is becoming mandatory for businesses that provide online services. Most applications, whether designed for use over wired or wireless networks, rely on the TCP protocol for reliable transport[1]. Businesses require that the existing online services and applications be available to any wireless client. Since these applications are TCP-based, wireless clients need to support some type of reliable transport to interface with these applications seamlessly.

TCP is designed for operation in wired networks, where random packet losses due to transmission errors are negligible. It is well known that TCP performs poorly over wireless links, which suffer from packet losses due to the error-prone and idiosyncratic nature of the wireless medium [2], [3], [4]. TCP does not adapt well to wireless communications, as it interprets the packet losses as being caused by network congestion[1]. In addition, acknowledgement and retransmission schemes used in TCP unnecessarily consume network bandwidth. This thesis outlines the problems associated with wireless communications over TCP and current solutions, and proposes a new TCP segment structure that enables successful wireless communications.

1.2 Motivation

The current thesis work is formulated on the foundation of few unresolved problems associated with the retransmission in the MAC layer. The retransmission at the MAC layer helps to identify the corrupted PDUs with a very good granularity. Those specific PDUs can be retransmitted to recover from the packet losses due to high error rate. However, retransmission in MAC layer causes unwanted jitter and bandwidth waste. To leverage the benefits of smaller size segments, a reliable transport protocol with smaller sub-segments has been proposed. This helps to achieve the granularity in retransmission without any introduction of jitter. The sub-segmentation solution controls the retransmission intensity and jitter factor by only retransmitting selective sub-segments.

The potential of Split TCP approach has influenced the protocol architecture in a significant extent. The split TCP approach has provided the freedom to adjust the reliable transport protocol for the lossy wireless channel while maintaining the backward compatibility with the wire-line network.

CHAPTER 2

PROBLEMS ASSOCIATED WITH WIRELESS-TCP

2.1 Overview

TCP is a well-suited protocol for reliable transmission over the wired networks. Currently, TCP does not support wireless communications for the following reasons.

2.2 Excessive Retransmission

Wireless communications experiences packet losses due to packet corruption over a noisy channel. A Bit Error Probability may vary from 0.001% up to 10% [5]. Damaged packets are retransmitted to recover erroneous frames before the TCP timer expires [6]. This retransmission increases the effective latency and decreases the effective throughput of the transmission. The noisy wireless channel causes TCP segment corruption. Retransmission of the corrupted packets consumes wireless bandwidth; packet losses during the retransmission and acknowledgement propagations are also possible. This situation becomes significantly worse over a noisy channel with a higher frame error rate (FER).

2.3 TCP Slow Start

In wired communication networks, TCP performs well because the channel error rate is low and packet loss occurs only when there is congestion. In wireless communications, where the channel error rate is high, any packet loss is interpreted as congestion by TCP and responded by reducing the transmission window size. This results in an unnecessary reduction in throughput [6].

2.4 Power Consumption

Hand-held devices coupled with wireless network interfaces are emerging as a new way to achieve seamless connectivity while adhering to stringent power and size constraints. The measurement-based results clearly indicate that the power drained by the network interface constitutes a large fraction of the total power used by the PDA. Simulation results show that the predominant cost comes not from the number of packets sent or received by a particular transport protocol, but from the amount of time the network interface is in an idle state. The energy cost can increase significantly in the presence of wireless losses since a receiver must wait for a TCP sender to recover from packet losses [7].

2.5 Memory Consumption

Wireless handsets are constrained by memory size and processing powers. Large buffer size and high code complexity may cause problems for the wireless TCP implementation.

CHAPTER 3

CURRENT SOLUTIONS

3.1 Indirect-TCP

Indirect TCP separates the flow and congestion control functionality on the wireless link from that on the fixed network. A separate transport protocol for the wireless communication link can support notification of events—such as disconnections, moves, and other characteristics including bandwidth availability—to the higher layers. This transport protocol can be used by *link aware* and *location aware* mobile applications.

Indirection allows the base station or mobile support router (MSR) to manage much of the communication overhead for a mobile host (MH). An MH that runs a very simple wireless protocol to communicate with the MSR, such as a small palmtop, can still access fixed network services, such as *WWW,* that may otherwise require a full TCP/IP stack running on the mobile. The Indirect TCP protocol model for an MH suggests that any interaction from the MH to a machine on the fixed network (FH) should be split into two separate interactions—one between the MH and its MSR over the wireless medium and another between the MSR and the FH. This model provides an elegant method for accommodating the special requirements of an MH in a way that is backward-compatible with the existing fixed network. All the specialized support that is required for mobile applications and low-speed, unreliable wireless communication medium can be built into the wireless side of the interaction, while the fixed side is left unchanged [9].

A transport protocol must be defined for the wireless link to handle the special characteristics of that physical medium. At the same time, an efficient mapping between the fixed network wired TCP and the Transport protocol over the wireless medium needs to be developed. The proposed TCP segment structure, described in section IV, satisfies the need for reliable Transport protocol over wireless communications.

3.2 Link Layer Retransmission

Reliability of TCP demands retransmission of lost segments. Depending on the error rate of the wireless channel, the retransmission of lost segments may cause higher BDP (Bandwidth-Delay Product) and bandwidth consumption. If the retransmission can occur in the RLC layer, the smaller size PDUs get retransmitted. This can improve the bandwidth degradation and loss rate (FER) of the communication channel. The data link layer will appear to have less loss rate and a little higher BDP [8].

Retransmission in the link layer is advantageous for the following reasons:

1. Persisted ARQ at the link layer saves all packets including UDP packets. Retransmission of the UDP packets is not required. Consequently, the buffer space at the MH and the wireless bandwidth is wasted.
2. Retransmission at the link layer causes jitter and increases the BDP [8].
3. TCP CRC catches errors in the segments even after the error checking at the link layer is complete [9], indicating that the retransmission in the link layer is not adequate to handle all the data losses.
4. A significant number of packet corruption may occur in the layers above the link layer.
5. The header compression mechanism in the link layer may impose more errors, which cannot be detected by the link layer CRC [9].

The proposed TCP segment structure, described in section IV, resolves the issues mentioned above since it uses sub-segmentation for transport layer messages.

3.3 Snoop TCP

The usual TCP problems over wireless links are given as motivation for this work. Interruptions during handoff worsen TCP performance. The base-station routing code is modified by adding a network-level snoop module to transfer the data from a fixed host (FH) to a mobile host (MH). Snoop keeps a cache of unacknowledged FH->MH packets. When a packet loss is detected (duplicate acknowledgement or local timeout), snoop retransmits the lost packet, hiding the FH from duplicate acknowledgements, thereby preventing congestion control from kicking in. If a new, in-sequence packet arrives from the FH, it is cached and forwarded to the MH. If an old packet re-arrives, it is a sender retransmission, and is forwarded to the MH, and the snoop retransmits counter reset. If a new, out-of-sequence packet arrives, it is marked as having experienced congestion loss, and the forwarded snoop retransmits the lost packet, hiding the FH from duplicate acknowledgements, thereby preventing congestion control from kicking in. A new ACK (Acknowledgement) from the MH allows snoop to clean its cache, update its RTT estimate, and relay the ACK to the sender. A spurious ACK is ignored. A duplicate ACK for a packet not in the snoop cache or marked as retransmitted by the sender is forwarded to the fixed host, since the FH TCP stack maintains state based on this ACK. A duplicate ACK that snoop does not expect indicates loss on the wireless link - the lost packet is retransmitted with high priority. A duplicate ACK that snoop does expect (based on the highest expected received sequence number) is discarded; such expected duplicate ACKs occur after snoop has retransmitted a packet but before the MH sequence numbers reflect the retransmitted packet.

Modification to the TCP state at the MH was done to process TCP negative acknowledgements (NACKs), which are generated at the BS for packets lost within a transmission window. The MH uses these NACKs to selectively retransmit lost packets.

A mobile-IP like strategy is used. The major difference is that each MH is assigned a home address and a temporary IP multicast address, instead of a home address and a foreign address. Packets are forwarded from the HA to the MH's primary BS, which forwards them to the MH. Secondary BSs that are identified as potential handoff targets also are asked to join the multicast group, and buffer packets for the MH in case the MH hand-offs to its cell. Data to the MH during a handoff are thus delivered directly from the new primary's buffered packets instead of causing data loss or forwarding delays. The new primary BS determines the first packet to forward based on the first ACK received from the MH. The BS also maintains a finite-sized FIFO queue of buffered packets instead of keeping all of them; the size of the queue is determined by the expected handoff delay.

The new primary BS's snoop agent may not have cached all packets since not all packets may have been buffered due to losses. Since snoop does not change TCP semantics, it is resistant to such gaps.

Performance is strictly improved using these schemes, both in terms of handoff delay and throughput. This method solves the wireless TCP problems, and does so without changing TCP semantics, inducing significant processing overhead, or requiring recompilation of clients or modifications of fixed host 's TCP stacks.

Invariably, some overhead is incurred by a scheme such as snoop, although it does perform without extra packet copying. State and resources must be maintained by secondary BS's to make the handoff routing work. IP multicast must be used for the handoff routing to work. The TCP stack on the MH has to be modified to get the fully snoop optimized.

CHAPTER 4

THE PROPOSED SOLUTION

4.1 Overview

The proposed solution focuses on the sub-segmentation of the transport layer segments. This section discusses the advantage of using the sub-segmentation approach over the wireless medium, within the context of the problems associated with the TCP. An improved algorithm is used to overcome the buffer handling limitation. A special windowing protocol is designed to add a flow control mechanism suitable for wireless communications.

4.2 Overview of the "Split –TCP" Approach Used In The Proposed Solution

The wireless section of an end-to-end communication uses a special TCP implementation along with an adapter layer between the IP and the special TCP implementation. On the other hand, the wired network section uses the standard TCP/IP implementation. A gateway protocol must be implemented to perform the mapping between the wireless-based transport protocol and the wired TCP. The advantage to this approach is that the standard TCP/IP implementations running at the FH can seamlessly communicate with any wireless-friendly TCP implementation. The base station, where the mapping occurs, should have two different receiving queues to support the two different physical networks. That way, a particular network can isolate itself from other type of connected networks. Otherwise, the mismatch in latencies between the two types of networks may create unnecessary contention in the receiver queue of the wired network. Figure 4.1 illustrates the Split TCP approach.

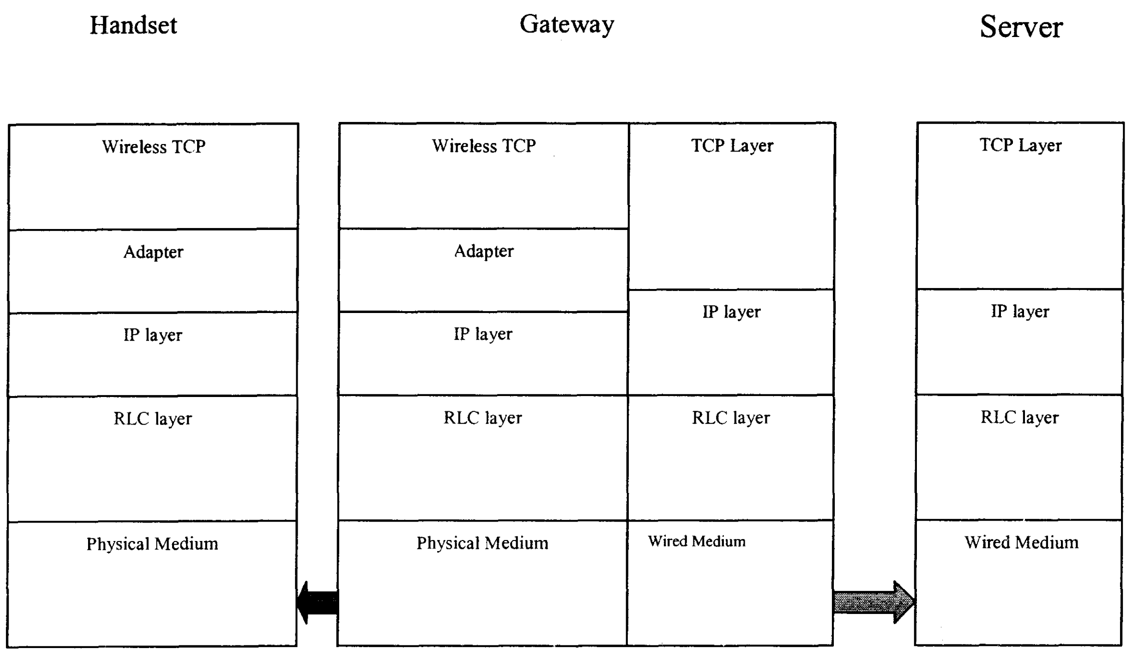

Figure 4.1 Split TCP approach.

4.3 Sub-Segmentation

The Transport layer protocol that is specially designed for the wireless medium uses the concept of sub-segmentation—the Transport layer segment is divided into a header and a series of sub-segments. Each sub-segment is as small as one RLC PDU payload (roughly 40 bytes for WCDMA)[12] and carries the:

1. Checksum of the TCP header as correlation ID to associate the transport layer header with a sub-segment. Checksum is unique for each PDU in the context of destination, source, sequence number, and so on.

2. Checksum for itself to detect the corruption of the sub-segment. Corresponding sub-segment number, which is unique in the context of a segment.
3. Data payload.
4. The overhead size of each sub-segment is less than 10%.

The rationale for selecting this approach is as follows. A standard TCP packet (each approximately 540 bytes in size, by default) must undergo the time-consuming fragmentation and reassembly process over the wireless medium (each PDU is roughly 40 bytes in size). At the same time, if one of the PDUs carrying the TCP segment is corrupted, standard TCP retransmits the entire TCP segment to recover from the error The probability that the retransmitted segment will be dropped is relatively high. This results in latency estimation of the TCP segment and unnecessary bandwidth consumption.

Though the sub-segmentation approach has time delays associated with the reassembly of smaller sized sub-segments, it improves the latency and bandwidth. Since the sub-segments are smaller in size, they do not have to go through further fragmentation and reassembly at the IP layer. In addition, since the Transport layer protocol only retransmits specific sub-segments to recover from the losses, the associated time delay to recover from the losses is less compared to the standard TCP. This approach contributes to decreased latency and bandwidth consumption.

4.4 New Protocol Stack with Relevant Message Structures

Figure 4.2 illustrates the proposed protocol stack and the corresponding PDU structures.

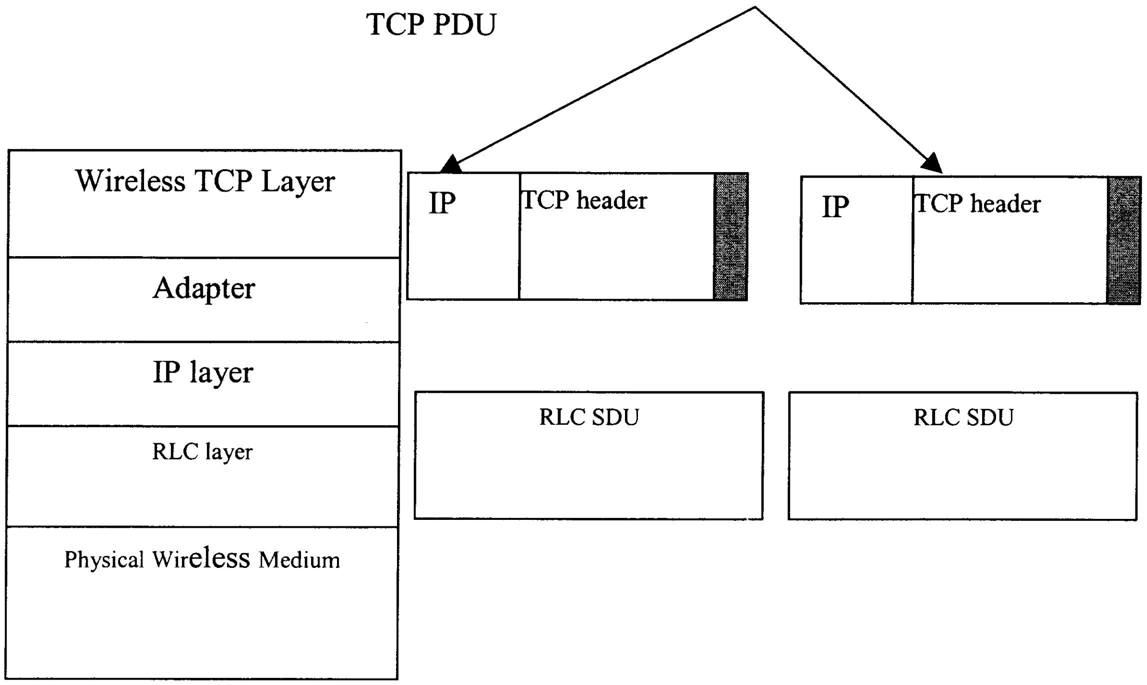

Figure 4.2 Proposed protocol stack.

The new protocol stack (Figure 4.2) consists of the following layers:

1) Physical Wireless Medium: Carries the headers and data from upper layers in the form of an electro-magnetic wave.

2) RLC Layer: Encapsulates the headers and data from the upper layer into a link layer frame structure, and is used in "Forward Error Correction."

3) IP Layer: IP layer is responsible for routing the PDUs attached with IP headers. This layer keeps track of the presence of any adapter layer. If the adapter layer is present, IP layer calls back the adapter layer appropriately. Otherwise, the packets in IP layer will move up to the transport layer once it is done with the IP layer.

4) Adapter Layer: Identifies the appropriate IP header associated with a transport layer sub-segment. The adapter layer caches the IP and the transport layer related header of a segment until its successful arrival is complete. This layer uses the correlation ID associated with each sub-segment to determine the corresponding IP header and the Transport layer protocol header. Shaded boxes indicate that the correlation ID across the sub-segments. Checksum of TCP header is used as correlation ID. The adapter layer allows any other transport layer protocols (such as UDP) to work directly on the standard IP layer protocol. IP layer communicates with the adapter layer via callback and the adapter layer has to be pre-registered with the IP layer to make the proposed protocol work.

5) Wireless TCP Layer: Provides reliability, duplicate handling, message ordering, and flow control, based on the use of sub-segments.

Note: The adapter layer helps the protocol implementation to live with other transport layer protocols, e.g. TCP, UDP. Section 4.6 and 4.7 discuss the issue in detail. The location of the adapter layer is selected carefully. As an alternative, if the adapter layer sits over RLC layer, the IP layer will attach the IP header to all the packets passing through it. The adapter layer has to read the IP header again to delete the attached IP header if the IP packet carries the proposed transport layer PDUs. This is an unnecessary overhead for all other protocols.

4.5 Header Format

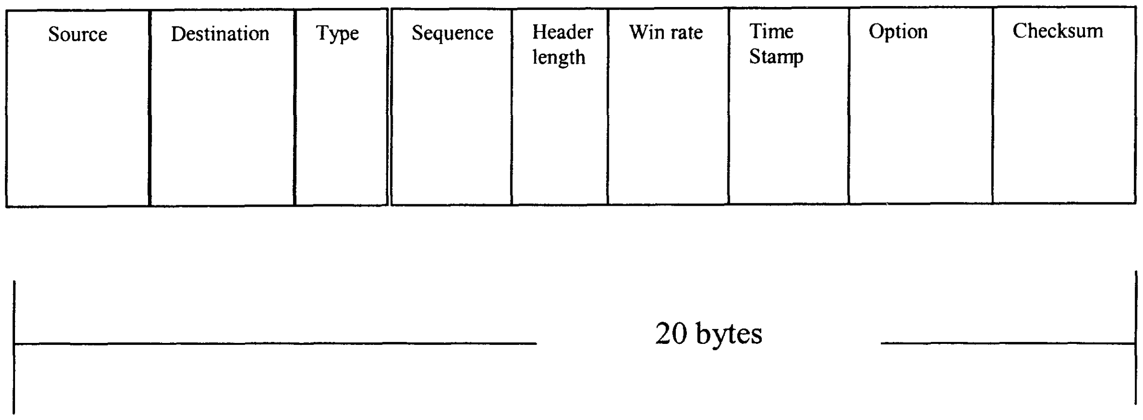

Figure 4.3 Request header.

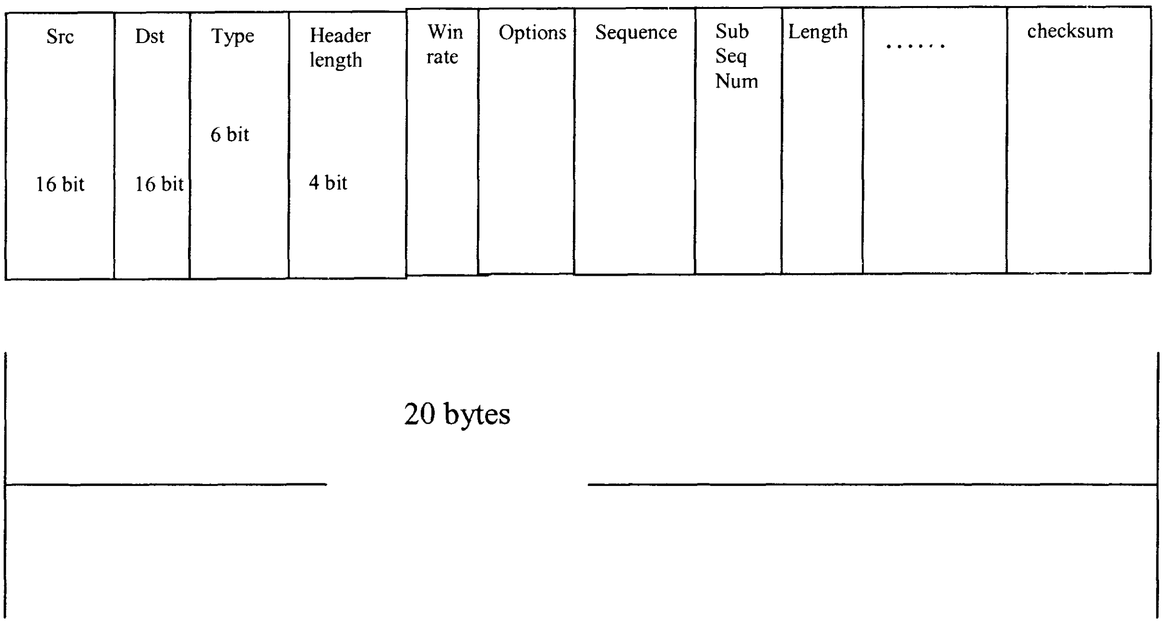

Figure 4.4 Acknowledgement header.

Figure 4.3 and Figure 4.4 present the request header and the acknowledgement header respectively. Few header fields are explained below. Fields are as follows:

1. "Type" field may contain one of three values "ACK","RETX","PSH".

2. “Sequence Number” has a size of 32 bits.
3. “Sub-Sequence Number” has a size of 4 bits.
4. “Length” has a size of 8 bits indicating the range of lost sub-segments.
5. “Time Stamp” has a size of of 8 bits indicating the origination time of the header.
6. “CheckSum” includes only the header checksum. Size is 16 bits.
7. The size of “Option” may vary but can increase the size of the header beyond 20 bytes.

The non header sub-segment structure is presented in Figure 4.5.

Payload	Correlation ID Size:16 bits	Segment # Size:32 bits	Sub-Segment# Size:4bits	Checksum Size:16 bits

Figure 4.5 Sub-segment structure.

4.6 Scenario Describing The Sending Process

Action sequence involved with the process of sending transport layer PDU are as follows.

1. Connection establishment.

2. Transport layer divides the total payload in series of sub-segments. Twelve sub-segments together form a virtual transport layer segment. Since each sub-segment is equivalent to a RLC PDU (40 bytes), twelve of them will create a virtual transport layer segment (Approximately 540 bytes). This virtual segment is same in size with the standard TCP segment and this will help in mapping the proposed protocol to the standard TCP.

3. For each segment the Transport layer creates a transport header of 20 bytes and passes it to the adapter layer. [For each segment, the header fields can be different, e.g., windowing rate.]

4. The adapter layer passes the Transport header to the IP layer to attach an IP header with it. Those two headers together form a payload for the RLC layer.

5. For other sub-segments, the adapter layer tunnels through the IP layer and directly passes the sub-segment to the link layer.

4.7 Scenario Describing The Receiving Process

Action sequence involved with the process of receiving transport layer PDUs includes:

1. When a packet reaches the IP layer, IP validation gets done if the checksum of IP header is correct.

2. If validation is successful, it calls back the adapter layer.

3. If the "protocol field" of the IP header indicates other protocols, the adapter layer returns the control to the IP layer.

4. Otherwise, the adapter layer confirms the checksum of the sub-segment, caches the IP header and the transport header and does not return the control to IP layer.

5. If validation in the IP layer fails, instead of dropping the packet, it calls back the adapter layer.

6. The adapter layer passes it to the transport layer.

7. If the checksum is confirmed, the Transport layer verifies the correctness of the packet assuming that it is a non header sub-segment.

8. If verification fails, the packet will be dropped and the control will not be returned to the IP layer.

9. If verification is successful, the adapter layer associates the cached headers to the sub-segment and the control will not be returned to the IP layer.

CHAPTER 5
PROTOCOL DEFINITION

5.1 Overview

The protocol defined within this section is a version of TCP that is suitable for a lossy wireless communication link. It ensures reliability, duplicate handling message ordering, and flow control at the transport layer communications. The required features of the proposed protocols include:

5.2 NACK-Based Interaction

A three-way handshaking, like TCP, should be used for the connection establishment .The flow, based on the proposed protocol, is as follows.

1. Sender sends a transport layer PDU once the receiver window allows.
2. Sender stores the sent PDU in a buffer.
3. Receiver receives it.
4. Receiver periodically finds the out-of-sequence PDUs in the queue and sends a NACK to inform the sender about the lost PDU. Loss of NACK can be handled via this mechanism. The time period should be selected carefully. In practice, this parameter should be the service rate of the receiver application provided the sum of the time required to send a NACK from the receiver to the sender and the time required to send a retransmitted segment in response to that is smaller than the service rate. Otherwise, this interval should be twice the propagation delay. The time stamp present in each header can notify the changes in the round trip time mentioned above. This helps the protocol to adapt to the changes related to the round trip time. Usually, It may vary due to delay in channel allocation.
5. Sender retransmits the lost PDU.
6. Receiver periodically sends a notification carrying a list of sequence numbers. The sequence numbers indicate the set of PDUs received at the receiving end.

Once the notification reaches the sender, the sender should update its storage buffer by deleting the PDUs identified in the notification. The frequency at which notifications are sent should be configurable depending on the buffer space available at the sender. This frequency must be significantly higher than the frequency of sending NACK.

7. The sender periodically checks its storage buffer. If a particular PDU remains in the buffer for a period of time that is longer than the time required to receive two successive notifications, a retransmission should take place. This step handles the loss of the last/first PDU even in the case when the notification is lost. The time period of checking the sender's buffer is the same as the time required to receive two successive notifications at the sender's end.

8. A positive ACK will be sent to the receiver in response to the notification. In the absence of this positive ACK, receiver should retransmit the notification.
   When the first PDU arrives, a notification should be sent immediately to avoid the unnecessary delay caused by the loss of the first segment. Otherwise, since this protocol needs the message ordering, all the successive PDUs will be stuck at the receiver buffer. This protocol provides total reliability, equivalent to the standard TCP protocol.

Use of NACK is required for the fast recovery of the lost segment. Notification is required for cleaning up the occupied storage buffer at the sender side and guarantee of the receipt of arrival from the receiver side.

This is a windowing protocol. Each sender sends only a limited number of PDUs (constrained by the available window space), thus achieving the flow control.

5.3 Buffer Management

To provide a better buffer management, the following scheme is proposed.

1. A separate variable should maintain the identifier of the next missing sub-segment, including the sequence number and the sub-sequence number.

2. A second variable should be created to keep track of the sequence and sub-sequence number of the last sub-segment removed by the application from the receiver TCP buffer.

3. The process of receiving transport layer segments and the reading of application data from the receiver buffer are carried out simultaneously. An application process reads the data from the receiver buffer starting from the sub-segment that is next to the segment indicated by the second variable mentioned. The application process continues reading until it reaches the sub-segment directly before the sub-segment referred to in the first variable mentioned. The read operation also frees up the buffer. This ensures that the receiver process will sort a smaller sized receiver buffer to identify the missing sub-segments. The variables have to be updated when a read operation is done.

4. To further improve the sorting process, every block of consecutive sub-segments can be mapped to a single logical entity after the sorting takes place on a comparatively smaller set. Each entity should have a lower bound and upper bound corresponding to the first and last sub-segment of the block. When more sub-segments arrive, the next sort will occur over those few logical entities and newly arrived sub-segments. A new sub-segment should be greater or less than a logical entity (as mentioned before) if it is greater or less than the upper and lower bound of the entity, respectively. The sub-segment is duplicated if it falls between the lower and upper bound.

5.4 Windowing Mechanism

The proposed windowing mechanism is specifically designed for the wireless medium. Instead of a sliding window protocol, where the sender's window carries the unacknowledged messages, this mechanism separates the flow control of the unacknowledged messages from those that are about to be sent. This windowing mechanism helps to avoid a reduction in message flow rate at the sender end due to the loss of NACK from the receiver side. The loss of NACK does not necessarily imply that the sent PDUs are lost, and reduction in the message flow rate is not always appropriate. On the other hand, if the retransmission rate is neglected in comparison to the transmission rate, transport layer PDUs have to wait unnecessarily at the receiver end buffer until the missing transport layer PDUs arrive. The sender's windowing rate should only be impacted by the service rate of the receiver's application.

The window message protocol is a rate-based windowing protocol. The receiver indicates its current service rate through the NACK header. If the sender application sends the data at a higher rate, a sender side queue of finite size should maintain the data to reduce the sending rate to the receiver's service rate. Even the transport layer PDUs that are unacknowledged and need to be retransmitted must go through the same process of the rate-based flow control. The following summarizes the features of this protocol:

1. The windowing rate does not change for packet corruption.
2. The windowing rate changes when a receiver becomes slow (not for the packet corruption).
3. The change in the rate gets propagated via the NACK headers, which carry a field for window rate and a field indicating the proportion of the service rate granted for the original transmission.
4. A logically separate buffer space should be dedicated for the unacknowledged messages. This storage supplies the transport layer PDUs for the retransmission.
5. The windowing algorithm steps are provided below:
    - Sender posts a Transport layer PDU onto the queue designated for the original PDUs at the sender side.
    - Sender determines the receiver's servicing rate and the proportion granted for original Transport layer PDUs from the acknowledgement headers received from the receiver.
    - Sender reads the PDUs from the queue and sends the data to the receiver at the rate indicated by the acknowledgement headers from the sender.
    - Steps 1 through 3 are repeated for the retransmitted messages.
    - The sender process round robin between two sender side queues to avoid any unnecessary delay either in the transmission queue or in the retransmission queue.

CHAPTER 6

THEORETICAL CHARACTERIZATION

6.1 Scenario

The protocol defined in the previous section introduces a specific sequence of interactions between a sender and a receiver. The scenario is as follows:

1. Sender sends a transport layer PDU.
2. The PDU may wait in the send queue before it gets sent.
3. The PDU departs from the send queue, once the receiver window is ready.
4. The receiver receives the PDU or it is dropped.
5. If the PDU is dropped, the receiver sends a NACK to the sender notifying the loss of a PDU. This NACK is one of the periodically sent NACKs.
6. Sender waits before the retransmission after it receives the NACK, if there is some delay involved due to the windowing mechanism.
7. Sender retransmits the PDU.
8. If the NACK is missing, the receiver will continually retransmit NACKs at given intervals.

6.2 Modeling Assumptions

The following assumptions are made to model the proposed protocol:

1. The probability of a transport layer PDU getting dropped is p.
2. Receiver window size is expressed as service rate, e.g., μ.
3. Both the sender's transmission queue size and the retransmission queue size are n.
4. Network queue sizes are very large. Therefore, packet drop and congestion delay from the network layer are ignored.

6. Transmission time is Δt_x.

7. Time to traverse between the sender and receiver is Δt_l. It includes the time taken to traverse the link and the time required to validate the packet in the link layer and transport layer.

8. $t = \Delta t_l + \Delta t_x$

9. Piggy-backing of NACK is required. For every $2t$ time, a NACK will be sent if there are missing PDUs. This NACK will indicate that all the PDUs are missing.

6.3 Markov Chain Based Model

A Markov chain-based model is introduced here to theoretically characterize the scenario identified above. Sender and receiver jointly form a system which represents the "TCP PDU transmission" of each Transport layer PDU. A total of seven recurrent states are identified for the system mentioned above. These are as follows.

Sent(S),

Received(R),

NACK Received(NR),

NACK Sent(NS),

NACK Not Received (NNR),

Wait at Send Queue(W),

Wait at the Retransmission Queue(WR).

The state transition sequences are as follows:

1. When a PDU is sent from the sender, the state "S" is reached. The time involved with state "S" before reaching the state "W" is 0.
2. The PDU will move to the wait state "W" with probability 1.
3. From the state W, if the PDU is not dropped, the state "R" is reached. The transition probability from the state "W" to "R" is "1-p". "R" is an absorbing state. The time involved in state "W" before reaching the next state "R" is $T_e' + t$ where expected T_e' is the waiting time in the transmission queue and t is the time which includes propagation time of a PDU and the time involved in the validation of the PDU. (See the assumptions.)
4. If the PDU is dropped, the state "NS" is reached from the state "W". The transition probability from the state "W" to "NS" is "p". The PDU gets dropped only after it reaches the receiver. It goes through the "link layer validation" and gets dropped if the PDU is corrupted. Time involved with the state "W" is $2t + T_e' + t$ where the average time to wait for the next

NACK is $2t$, the expected waiting time in the transmission send queue is $T_e^{'}$, and time to send a PDU to the receiver is t.

5. If the PDU is not dropped, the state "NR" is reached. The transition probability from the state "NS" to "NR" is "1-p". The time involved with the state "NS" in this case is t.

6. Otherwise, if the PDU is dropped, the state "NNR" is reached. The transition probability from the state "NS" to "NNR" is "p". Since NACK sending is a periodical process, the same NACK will be repeatedly sent after each round trip time is passed. The rationale is that the time involved in sending a NACK followed by a response from the sender should be greater than at least a full roundtrip time. Before it is understood that the NACK is lost and the system is at the NNR state, a minimum of a round trip time is required. This time is equivalent to $2t$. In this case, this is the time involved with the state "NS". This particular event may occur repeatedly.

7. From the state "NNR", a transition may happen directly to the state "NS". The transition probability is 1. Ignoring the processing time at the sender end, the time involved with the state "NNR" in this case is 0, as sending of NACK occurs only when the "NNR" state is reached.

8. From the state "NR", a transition to the WR state happens with probability 1. The time involved with the state "NR" in this case is 0.

9. From "WR", a transition to the state "S" always happens. The transition probability is 1. The time involved with the state "WR" is $T_e^{''}$.

Figure 6.1 illustrates the state transition diagram.

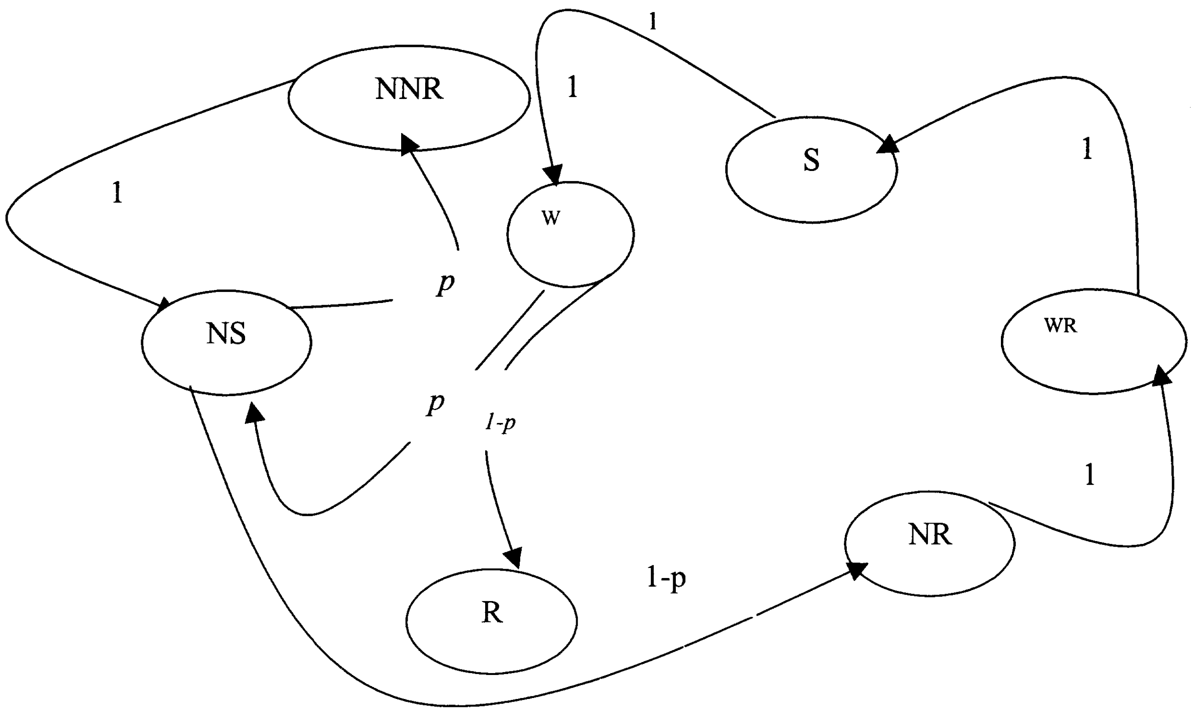

Figure 6.1 State transition diagram for NACK based protocol.

The transport layer protocol proposed here resembles a state machine. The states are discrete over a range of discrete times. At the same time, each state is memoryless. The probability of being in each state solely depends on its last state but not on any previous state. Thus, in this case, the Markov chain suits well as a modeling foundation.

6.4 Expected Number of Retransmission of PDUs

Let say N number of PDUs are transmitted from the transport layer. A few of those PDUs may get lost and be retransmitted. Even few retransmitted PDUs may get dropped, and this process may continue. Let consider a system where states of the system correspond to the number of transmitted PDUs. Figure 6.2 indicates the state transition.

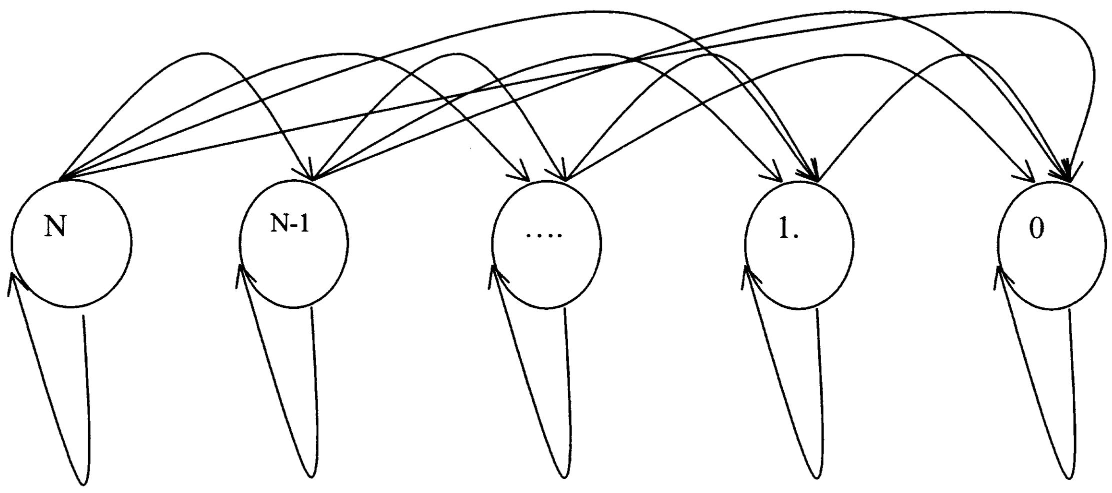

Figure 6.2 State transition diagram for retransmission.

The state transition matrix P for the transient states should be as follows.

$$\begin{vmatrix} p00 & p01 & p02 & .. & p03 & p0N \\ p10 & p11 & p12 & .. & p1N-1 & p1N \\ p20 & p21 & p22 & .. & p2N-1 & p2N \\ .. & .. & .. & .. & .. & .. \\ pN-10 & pN-11 & pN-12 & .. & pN-1N-1 & pN-1N \\ pN0 & pN1 & pN2 & .. & pNN-1 & pNN \end{vmatrix}$$

where $p_{jk} = \binom{j}{j-k} (1\text{-p})^{k} (\text{p})^{j--k}$ for $j \geq k$,

p_{jk} =0 for j<k,

The value of j indicates the state from where the state transition begins and value of $j-k$ represents the destination of the state transition. The probability that the system will reach the state k at any point of time starting from the state N is calculated follows.

Let $R(i,j)$ be the expected number of visits to state j starting from state i, and $F(i,j)$ be the probability of ever reaching the state i. Let D denote the set of all transient states and let Q and S be the matrices obtained by deleting all the rows and columns corresponding to the recurrent state:

$Q\,(i,j)=P(i,j)$ for $i,j \ge 1$ and $S=(I-Q)^{-1}$.

The potential matrix of Markov chain [11] is

$$R = \left|\begin{array}{c|cccc} \infty & \infty & \infty & \cdots & \infty \\ \hline \infty & p_{11} & p_{12} & \cdots & p_{1N} \\ \infty & & & \cdots & \\ \cdots\cdots & & & \cdots & \\ \infty & p_{N1} & p_{N2} & \cdots & p_{NN} \end{array}\right|$$

Thus, $F(j,j)$ =1-1/ $R(j,j)$; $F(i,j) = R(i,j)\,/\,R(j,j)$ [11], and the expected number of retransmitted PDUs is equal to $N_e = \sum_{k=1}^{N} k\ F(N,k)\ +\ \ N\,.\,p_{N0}$ (1)

6.5 Inter-arrival Time of Retransmitted PDUs to the Retransmission-Queue.

The inter-arrival time is the sum of the expected time consumed in waiting for the next NACK and the expected time to send a NACK successfully. The value of t is defined earlier. The frequency of sending NACKs is assumed to be $2t$ considering that μ (the service rate at the receiver end) is smaller than $2t$, as explained previously. The expected time to send a NACK successfully is

$$T_1 = (1-p)t + p(1-p)(t+2t) + p^2(1-p)(t+t++t+2t) + \ldots \infty$$

$$= (1-p)t\,[(1+2p)/(1-p) + 2p^2/(1-p)^2].$$

The inter arrival time is equal to $t_e = T_1$. (2)

6.6 Expected Waiting Time in the Retransmission Send-Queue

Every t_e time, a retransmission request arrives at the sender's end, if there is any PDU loss is detected at the receiver side.

Using Equation (2) a conclusion can be made that the "Send-Queue" dedicated only for retransmission should have a input rate of $\lambda' = N_e / t_e$.The service rate for this queue is μ.The expected waiting time in the "retransmission send queue" is

$$T_e'' = ((1-\rho)\rho / 1-\rho^{n+1})(1-(n+1)\rho^n + n\rho^{n+1})/(\lambda' * (1-\rho)^2), \quad (3)$$

where $\rho = \lambda' / \mu$.

6.6 Expected Waiting Time in the Send- Queue

Consider that the service rate at the receiver side is μ. If we consider the input rate at the send queue is λ and the queue size is n, using M/M/1/K queuing model and Little's theorem, the expected waiting time in the queue is as follows:

$$T_e' = \sum_{r=0}^{n+1} (1-(\lambda/\mu)) * (\lambda/\mu)^r * r / \lambda * (1-(\lambda/\mu)^{n+1}) \quad (4)$$

The value of t is defined in the assumption.

6.7 Latency Calculation

The state transition diagram in Figure 6.1 has a starting point and an ending point. The starting point is "S" and the ending point is "R".

Assume the states are denoted with the following numbers.

(S) ->1,

(W)->2,

(NS)->3,

(NR)->4,

(NNR)->5,

(WR)->6,

(R)->7

The "transition probability matrix" for each state is presented below.

$$\begin{vmatrix} p11 & p12 & p13 & p14 & p15 & p16 & p17 \\ p21 & p22 & p23 & p24 & p25 & p26 & p27 \\ p31 & p32 & p33 & p34 & p35 & p36 & p37 \\ p41 & p42 & p43 & p44 & p45 & p46 & p47 \\ p51 & p52 & p53 & p54 & p55 & p56 & p57 \\ p61 & p62 & p63 & p64 & p65 & p66 & p67 \\ p71 & p72 & p73 & p74 & p75 & p76 & p77 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 1-p \\ 0 & 0 & 0 & 1-p & p & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} = P$$

The probability that the system starts at state j and reaches the state k for the first time at the nth step (or after nth transition) is f_{jk}. The probability that starting with state j the system will ever reach state k is clearly

$$\text{F(jk)} = \sum_{n=1}^{\infty} f_{jk}^{(n)}$$ by using Chapman-Kolmogorov equation [11].

Using the "transition probability matrix" provided above,

$F(jk)= \sum_{n=1}^{\infty} \mathbf{P}_{jk}^{(n)}$, where $\mathbf{P}$ is the "transition probability matrix".

Thus, $F_{jk} = \mathbf{P}_{jk}^{(1)} + \mathbf{P}_{jk}^{(2)} + \mathbf{P}_{jk}^{(3)} + \mathbf{P}_{jk}^{(4)} + \ldots\ldots\ldots$ (5)

The state transition diagram presented in Figure 6.1 demonstrates that we are interested in determining the transition probability from state 1 to state 7, that is, the value of F_{17}.

Then,

$$F_{17} = \mathbf{P}_{17}^{(1)} + \mathbf{P}_{17}^{(2)} + \mathbf{P}_{17}^{(3)} + \mathbf{P}_{17}^{(4)} + \ldots\ldots\ldots \quad (6)$$

Now, from the assumptions considered previously, each transition step produces different transition times, as follows:

$t_1 =$ n/a, $t_2 = t + T_e'$; $t_3 =$n/a, $t_4 =$n/a, $t_5 =$n/a, $t_6 =$n/a, $t_7 = 4t + 2T_e' + T_e''$, $t_8 =$n/a, $t_9 = 6t + 2T_e' + T_e''$, $t_{10} =$n/a, etc.

Missing terms of this matrix do not exist, as probabilities of those transitions are zero. The expected time required to reach the receiver from a sender is the mean time required to move from state 1 to state 7.

Let say the mean time is μ_{17},

Then $\mu_{17} = \sum_{n=1}^{\infty} t_n f_{17}^{(n)}$.

$$= t_1 \mathbf{P}_{17}^{(1)} + t_2 \mathbf{P}_{17}^{(2)} + t_3 \mathbf{P}_{17}^{(3)} + t_4 \mathbf{P}_{17}^{(4)} + \ldots\ldots\ldots \quad (7)$$

6.8 Expected Receiver Side Throughput

Figure 6.1 represents the state transition diagram of a system, which includes both the sender and the receiver taking part in message exchange. The incoming message rate is given as λ i.e., λ number of PDUs enter the system in one second. Thus, every $1/\lambda$ second, one PDU enters the system. Each of these PDUs, spends μ_{17} seconds (on average) before it departs from the system and becomes available to the application at the receiver end. This concludes that every ($\mu_{17}+1/\lambda$) seconds, one PDU departs from the system and becomes available to the receiver side application, and the throughput of this system is $1/(\mu_{17}+1/\lambda)$ PDUs per second.

The bandwidth consumption can be measured as $N/(N+N_e)$ which is the same as the system efficiency.

CHAPTER 7

SUB-SEGMENTATION VS STANDARD APPROACH

7.1 Normal Segment Approach

A transport layer segment is assumed to contain virtual sub-segments equivalent to RLC PDUs (approximately 338 bits). Assume that the channel error (FER)=e and the number of virtual sub-segments = $m \leq 12$ (because, by default, the transport layer segment size is approximately 540 bytes). A normal transport segment will be dropped if any one of the constructing PDU/virtual sub-segments is dropped with probability

$p = 1-(1-e)^m$.

7.2 Sub-Segment Based Approach

A transport layer sub-segment is assumed to be equivalent to a RLC PDU. Let say that the channel error (FER)=e. The number of sub segments = $m \leq 12$ [because the default TCP segment size is approximately 540 bytes]]. Thus, a transport layer sub-segment will be dropped with a probability e.

7.3 Latency Comparison

The expression of μ_{17} (Expected Effective Latency Time) is presented in the previous section (Equation 8). It is apparent that μ_{17} is a monotonically increasing function of p (the probability of a transport layer PDU is dropped). For a normal sized TCP segment, $p= 1-(1-e)^m$ and in the case of sub-segmentation based approach, p=e. Since in a wireless medium, e is less than 10%, and therefore the dropping probability of a normal size TCP segment is always greater than that of the sub-segmentation based TCP PDUs. Thus, the "Expected Effective Latency Time" calculated from the large segment based approach is higher than that calculated from the sub-segmentation approach, provided the code complexity is the same in both cases.

7.4 Throughput Comparison

In the previous section, effective throughput is calculated. Since it is a decreasing function of p, it is apparent that the throughput in the case of the large segment based approach is less than that of the sub-segment based approach.

CHAPTER 8

SIMULATION RESULTS

8.1 Setup

Application modules are written to compare the effect of the sub-segmentation with large segment-based transmission. A NACK-based approach is adopted to implement the reliable transmission.

Application modules simulate the RLC frame transmission of a wireless medium and reflect the random Frame Error characteristic. UDP packets of very small size (as small as 40 bytes) are used over Ethernet to mimic the RLC frames of a wireless channel. Size of each UDP packet is the same as the size of one RLC SDU. A software module is introduced to generate the random packet drop. This module randomly drops the previously mentioned UDP packets while sending a given number of packets. Different channel error rates contribute to different number of packet drops.

An experiment is designed to conduct the simulation. Each experiment uses UDP-based socket-APIs to implement a sender and a receiver taking part in the reliable transmission. The sender sends one thousand small UDP packets at a rate, which depends on the available windowing rate of the receiver. The experiment continues until all the packets arrive the receiver end. At the end of the experiment, the average latency and throughput of the transmission will be measured. For our experiment, one thousand small UDP packets are transmitted to mimic the RLC frame transmission. Each experiment uses a random seed to generate random numbers. To eliminate any bias involved with the random seed selection, one hundred randomly selected seeds are used to repeat the experiment for one hundred times. The average value of those hundred trials are selected as the final result for each experiment. The goal of this experiment is to compare the performance of two protocol schemes in an environment characterized by small sized PDU support and random error rate. It is not our goal to accurately predict the real time performance over the wireless medium like CDMA etc.

Two sets of applications are developed to run the simulation. The first set of application code implements a large segment-based reliable transmission protocol. UDP-based socket-APIs are used to implement a sender and receiver taking part in the reliable transmission. To mimic the fragmentation and reassembly that takes place at the IP layer (a time-consuming operation), a module is introduced to fragment and reassemble the large transport layer segments. A large segment is considered dropped if one of the fragmented pieces (40 bytes) of the large segment is dropped. In case of retransmission, a full large segment (540 bytes) gets retransmitted.

The second set implementation is a sub-segment-based reliable transmission protocol. UDP-based socket-APIs are used to implement a sender and receiver taking part in the reliable transmission. A separate module for the fragmentation and reassembly at the IP layer is not needed as only small sub-segments are arriving at the IP layer in this case. The creation and assembly of sub-segments at the receiver end is implemented at the Transport layer over UDP sockets. In the event of retransmission, a small sub-segment gets retransmitted.

8.2 Results

The results are collected against a given set of parameters to characterize the wireless channel. input rate from the sender as follows.

1. Forward channel rate is 200Kbps.
2. Reverse channel rate is 64kbps.
3. Forward delay in the channel is 20ms.
4. Buffer size is 10 PDUs.

Table 8.1 Comparison of Throughputs Derived From Analysis

Error Rate	Throughput/Sub-Segment	Throughput/Segment
1	34.2005	26.89452
2	33.4796	19.769651
3	32.7439	14.016796
4	31.9860	9.960003
5	31.1939	7.343242
6	30.3516	5.736482
7	29.4472	4.7862988
8	28.4930	4.262448
9	27.5342	4.030479
10	26.6221	4.018432

Table 8.2 Comparison of Throughputs Derived From Simulation

Error Rate	Throughput/Sub-Segment(Kbps)	Throughput/Segment(Kbps)
1	33.2005	25.89452
2	32.4796	18.769651
3	31.0439	13.016796
4	29.9860	8.960003
5	27.1939	6.343242
6	25.3516	4.736482
7	24.4472	3.7862988
8	22.4930	2.962448
9	20.5342	1.930479
10	19.06221	1.018432

Table 8.3 Comparison of Latency Derived From Analysis

Error Rate	Latency/Sub-Segment	Latency/Segment
1	0.00775	0.010298
2	0.00795	0.014586
3	0.008172	0.021229
4	0.008404	0.030528
5	0.008658	0.041977
6	0.008943	0.054183
7	0.009266	0.065257
8	0.009630	0.073474
9	0.010021	0.077795
10	0.010420	0.078033

Table 8.4 Comparison of Latency Derived From Simulation

Error Rate	Latency/Sub-Segment	Latency/Segment
1	0.00815	0.0142
2	0.00845	0.01789
3	0.009117	0.02423
4	0.009401	0.03553
5	0.009901	0.0482
6	0.01	0.06218
7	0.01201	0.07226
8	0.01273	0.07747
9	0.013391	0.0812
10	0.014002	0.08633

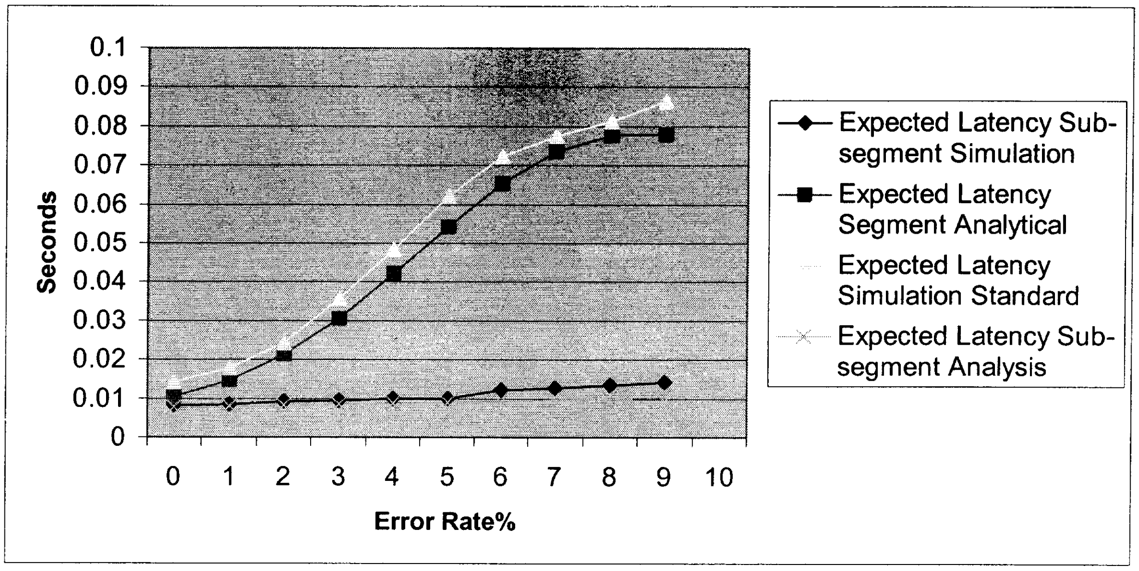

Figure 8.1 Latency comparison.

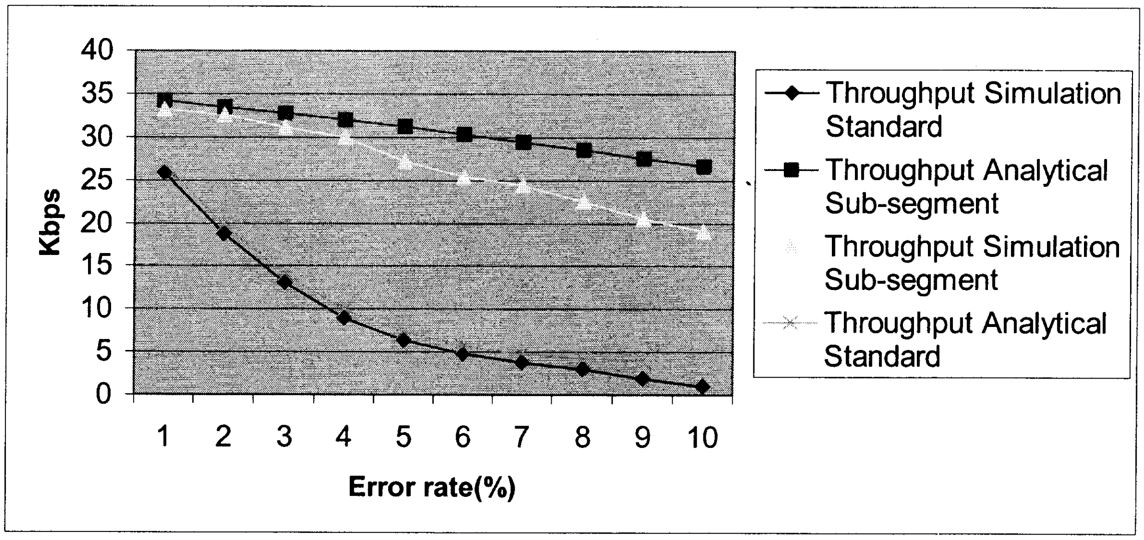

Figure 8.2 Throughput comparison.

Figure 8.1 (supported by Table 8.3 and Table 8.4) indicates that both the simulation results and the analytical results confirm the better performance of sub-segmentation based approach in the context of latency. The throughput comparison is presented in Figure 8.2 (supported via Table 8.1 and Table 8.2) to show the effect of sub-segmentation. Deviation in the values between the analytical results and the simulation-based results is due to the code complexity involved with the protocol implementation.

CHAPTER 9

LIMITATIONS OF THE PROPOSED SOLUTION

The proposed solution introduces code complexity involving intensive searching operations. The code complexity demands a large amount of processing power. Use of low power processors should be able to control the power consumptions. An efficient buffer handling mechanism also helps to reduce the size of the data set to be searched.

CHAPTER 10

CONCLUSION

In this thesis, a new Transport layer reliable protocol has been introduced to solve the problem associated with TCP over wireless communications. The sub-segmentation scheme solves the problems associated with the previous schemes where the standard TCP is used over the wireless link and a link layer retransmission is introduced to manage the packet loss. Simulation results demonstrate the superior performance of the proposed scheme over a wireless communication medium of any loss rate. Logical reasoning has been applied to illustrate how this scheme helps to keep the jitter low and to achieve better QoS .

Additional work is required to solve the issues related to hand-off, mapping between the wired and wireless protocol, and the scalability of the real deployments. A test run over an actual live wireless communication channel may provide interesting observations.

APPENDIX

RANDOM PACKET LOSS GENERATION

In this appendix, a segment of codes used to simulate the random packet loss is presented.

```
count=0;
//drop packet

if(typeError.compareTo("RANDOM")==0)
{

_rand[0]= 1;

for(int kk=1; kk<=2;kk++)
_rand[kk]=2;

for(int kk=3; kk<=6;kk++)
_rand[kk]=3;

for(int kk=7; kk<=11;kk++)
_rand[kk]=4;


for(int kk=12; kk<=17;kk++)
_rand[kk]=5;


for(int kk=18; kk<=24;kk++)
_rand[kk]=6;


for(int kk=25; kk<=32;kk++)
_rand[kk]=7;


for(int kk=33; kk<=41;kk++)
_rand[kk]=8;

for(int kk=42; kk<=51;kk++)
_rand[kk]=9;

for(int kk=52; kk<=62;kk++)
_rand[kk]=10;
```

```
for(int kk=63; kk<=99;kk++)

_rand[kk]=11;


for(int rr=0;rr<TCPHeader._numSeg*(TCPHeader._numSubSeg+1);rr++)
{
        int random =_rrand.nextInt(100);
        if(_rand[random]==_errorRate )
        {
                _dropped++;
                System.out.println("dropped seg "+rr+"
"+_sW.dropSubSegment((new Integer(rr)).toString()));
                errorOP._dropped++;
        }
}
```

REFERENCES

[1] F. M. Anjuman and L.Tassiulas, "On the behavior of different tcp algorithms over a wireless channel," *ACM Sigmetrics*, Atlanta, May 1999, pp. 155-165.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz,"A comparison of mechanisms for improving tcp performance over wireless links," *IEEE/ACM Transactions on Networking*, June 1997, vol. 5, pp. 120-146.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM Wireless Networks Journel,(WINET)*, vol. 14, pp. 30-56, Decemeber 1995.

[4] R. Caceres and L. Iftode, "Improving reliable transport protocols in mobile computing environment," *IEEE JSAC*, pp. 130 –145, June 1995.

[5] M.J. Juntti, and M. Latvaho, "Bit –Error Probability Analysis of Linear Receivers for CDMA Systems in Frequency-Selective Fading Channels," *IEEE Transactions on Communications,* vol. 47, pp. 110-122, December 1999.

[6] M. Stemm and R.H. Katz, "Measuring and reducing Energy Consumption of Network Interfaces in Handheld Devices," *IEICE Transactions on Communications,* pp. 110-122, August 1997.

[7] G. Sekin, and R. Brooks, "Challenges of Wireless Media Streaming," Infrastructure for Mobile And Wireless System, pp. 164-172.

[8] A. Bakre and B.R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," *15th Int'l Conf. on Distributed Computing Systems (ICDCS),* pp. 46-75, May 1995.

[9] J. Stone and C. Patridge," When The CRC and TCP Checksum Disagree," *ACM SIGCOMM*, pp. 68-84, September 2000.

[10] Erhan Cinlar, "An Introduction To Stochastic Processes", Prentice-Hall, Inc, Chapter 6, Year 1975.

[11] "WAP-225-TCP-20010331", Version 31-March-2001, Wireless Application Protocol Forum, Ltd. All Rights Reserved. Terms and conditions of use are available from the WAP Forum Web site. (http://www.wapforum.org/what/copyright.htm).