**In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.**

**While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.**

A Performance Evaluation of the Radio Link Control Protocol in 3G UMTS

by

Jun Li

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements of the degree of

Master of Science (Information and Systems Science)

School of Mathematics and Statistics
Ottawa-Carleton Institute of Mathematics and Statistics
Carleton University
Ottawa, Ontario, Canada K1S 5B6
August 2003

Copyright © Jun Li, 2003

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-88775-8

Our file Notre référence

ISBN: 0-612-88775-8

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Canada

THE UNDERSIGNED RECOMMEND THE FACULTY OF GRADUATE
STUDIES AND RESEARCH THE ACCEPTANCE OF THE THESIS

A Performance Evaluation of the Radio Link Control Protocol in 3G UMTS

Submitted by Jun Li

In partial fulfillment of the requirements for the degree of
Master of Science (Information and Systems Science)

Professor Yiqiang Q. Zhao
Thesis supervisor

Professor Cyril W.L. Garner
Director, School of Mathematics and Statistics

Abstract

With the rapid advances in wireless communications and internetworking, Universal Mobile Telecommunications Systems (UMTS), a main role of the emerging third-generation (3G) wireless networks, have aimed toward the provision of high-speed data integrated with voice services. Since today's data traffic is largely based on Transmission Control/Internet Protocol (TCP/IP) suite, UMTS networks have employed a Radio Link Control (RLC) protocol in support of reliable upper layer protocols such as TCP. RLC employs sophisticated retransmission schemes to perform partial error recovery at the link layer, thus hiding transmission errors from upper layers. Since TCP will treat every segment loss as a sign of congestion, the above mentioned schemes can reduce the chances of a mistaken invocation of the TCP congestion control mechanism, which results in performance degradation. Therefore, it is critical to optimize the RLC protocol to achieve the best performance. The research in this thesis focuses on the RLC performance in terms of the Service Data Unit (SDU) average delay, the RLC throughput, and the RLC goodput in a scenario of bulk data transfer from a Radio Network Controller (RNC) to a User Equipment (UE). Two important retransmission triggers, the Poll Timer and the STATUS Period Timer, have been evaluated under various Block Error Rates (BLERs) using simulation. The performance trends against the Poll Timer and the STATUS Period Timer settings under various BLERs have been analysed and summarized respectively in order to guide the design of optimized 3G UMTS systems for data services. This research builds a process model of the RLC protocol with the OPNET simulator and also extends an existing theoretical work on the RLC SDU average delay of out-of-sequence delivery.

Acknowledgements

I am sincerely grateful to my supervisor Dr. Yiqiang Q. Zhao for his training, advice, and encouragement. He has always offered very helpful guidance to me throughout my thesis research work. I would like to acknowledge the education provided by the School of Mathematics and Statistics, the School of Computer Science, and the Department of System and Computer Engineering, and the help from administrators of the School of Mathematics and Statistics and the Faculty of Graduate Studies and Research, especially Dr. Sam Melkonian and Ms. Diane Smyth.

I would like to thank Dr. Jay Wang and Dr. Delfin Montuno of Nortel Networks for their valuable comments and feedbacks, and for the access to the OPNET simulator.

My thanks also go to my friend Dr. Baoxian Zhang of the University of Ottawa for his useful discussions about research approaches in general, and my research topics in specific.

Finally, I thank my family members, especially my wife, Xuetao Li, for her constant support and encouragement. The excitement from the birth of my son, Jeffrey, has boosted the progress of this research.

Contents

1	Introduction	1
1.1	Motivations	1
1.2	Contributions	4
1.3	Outline	6
2	Background	7
2.1	The Transmission Control Protocol (TCP)	8
2.1.1	Reliability	9
2.1.2	Flow Control	11
2.1.3	Congestion Control	11
2.1.3.1	Slow Start	12
2.1.3.2	Congestion Avoidance	13
2.1.3.3	Congestion Control	14
2.2	TCP over Wireless Networks	15
2.2.1	Overview	15
2.2.2	Main Characteristics of Wireless Networks Compared to Wireline Networks	16

2.2.2.1	Higher Bit Error Rate	16
2.2.2.2	Lower Bandwidth of Wireless Links	16
2.2.2.3	Larger and Varying Delay	17
2.2.2.4	More Disconnections	18
2.3	Schemes of Improving Performance of TCP over Wireless Networks	19
2.3.1	Link-Layer Protocols	20
2.3.2	End-to-End Protocols	22
2.3.3	Split-Connection Protocols	23
2.4	The Third Generation (3G) Wireless Networks	24
2.4.1	Three Generations of Wireless Cellular Networks	24
2.4.2	CDMA2000 Systems	26
2.4.2.1	The System Architecture	26
2.4.2.2	The Physical Layer	27
2.4.2.3	The LAC Sublayer	28
2.4.2.4	The MAC Sublayer	28
2.4.2.5	Radio Link Protocol (RLP)	29
2.4.3	UMTS Systems	31
2.4.3.1	The Protocol Architecture	32
2.4.3.2	The Physical Layer	33
2.4.3.3	The MAC Sublayer	33
2.4.3.4	The RLC Sublayer	34
2.4.3.5	The PDCP Sublayer	35
2.4.3.6	The BMC Sublayer	35

2.5	Performance Evaluation Overview of 3G Wireless Networks	35
2.6	Summary	36
3	The Radio Link Control (RLC) Protocol	38
3.1	Overview	38
3.2	The Three Service Modes	39
3.2.1	Transparent Mode (TM)	40
3.2.2	Unacknowledged Mode (UM)	40
3.2.3	Acknowledged Mode (AM)	41
3.3	PDU Formats and Parameters	43
3.3.1	AMD PDUs	43
3.3.2	STATUS PDUs	45
3.4	Protocol Parameters and Buffers	49
3.5	State Variables and Operations in the Transmitting Side	51
3.5.1	State Variables in the Transmitting Side	52
3.5.2	The Polling Mechanism	54
3.5.3	The RLC SDU Discard Mechanism	57
3.6	State Variables and Operations in the Receiving Side	58
3.6.1	State Variables in the Receiving Side	59
3.6.2	The STATUS Transmission Mechanism	60
3.7	TTI Operations	62
3.8	Summary	63
4	Simulation Support	64

4.1	Simulation Framework	64
4.2	RLC Functions Implemented	66
4.3	Actions on the Process State Transitions	68
4.3.1	Actions on Transition of “IDLE” → “PRO_UPPER”	70
4.3.2	Actions on Transition of “IDLE” → “PRO_LOWER”	70
4.3.3	Actions on Transition of “IDLE” → “TTI_EXPIRE”	71
4.3.3.1	The Procedure <i>receive_data_pdu()</i>	72
4.3.3.2	The Procedure <i>receive_status_pdu()</i>	73
4.3.3.3	The Procedure <i>generate_ack_list_sufi()</i>	75
4.3.3.4	The Procedure <i>produce_new_data_pdu()</i>	76
4.3.3.5	The Procedure <i>send_pdu()</i>	77
4.4	Summary	78
5	Simulation Validations and Experimental Evaluation	80
5.1	Simulation Validations	80
5.1.1	Protocol Parameters Setting for Simulation Validation	80
5.1.2	Metrics Used for Simulation Validation	81
5.1.3	Analysis of the Average Delay	83
5.1.4	Comparison of Theoretical and Simulation Results	86
5.1.4.1	Comparison of SDU Discard Rates	86
5.1.4.2	RLC SDU Average Delays Comparison	86
5.2	Performance Metrics	89
5.3	Simulation Configuration for Performance Evaluation	89

5.3.1	The Traffic Model of the Upper Layer	90
5.3.2	The Transmission Errors of the Physical Layer	90
5.3.3	The Data Rate of the Wireless Channel	90
5.3.4	The Polling Triggers	91
5.3.5	Assumptions of RLC Process	91
5.3.6	Protocol Parameters Setting for the Performance Evaluation	91
5.4	Simulation Results and Discussion	92
5.4.1	Effect of STATUS Period Timer	93
5.4.1.1	The Effect on the SDU Average Delay	93
5.4.1.2	The Effect on the RLC Throughput	97
5.4.1.3	The Effect on the RLC Goodput	99
5.4.2	Effect of Poll Timer	102
5.4.2.1	The Effect on the SDU Average Delay	102
5.4.2.2	The Effect on the RLC Throughput	105
5.4.2.3	The Effect on the RLC Goodput	105
5.5	Summary	108
6	Conclusion	109
6.1	Summary of Contributions	109
6.1.1	Evaluation of the STATUS Period Timer on the RLC Performance .	110
6.1.2	Evaluation of the Poll Timer on the RLC Performance	110
6.1.3	Implementation of the RLC Protocol	110
6.1.4	Improvement of an Analytical Result of the SDU Average Delay . .	111

6.2 Future Work	111
Bibliography	113
Index	121

List of Tables

2.1	The RLP Data Transmission Procedure	31
2.2	Performance Criteria of TCP over 3G Wireless Networks	35
2.3	Factors Affecting the TCP Performance	36
5.1	Simulation Parameters for Validation	82
5.2	Simulation Parameters for Performance Evaluation	92

List of Figures

2.1	(a) The OSI Seven-Layer Model and (b) the TCP/IP Protocol Architecture .	8
2.2	Multiple Duplicate ACKs	10
2.3	Slow Start and Congestion Avoidance	13
2.4	Congestion Control: Time-out	15
2.5	Link-Layer Retransmission	20
2.6	CDMA2000 Systems' Architecture	27
2.7	UMTS Systems' Architecture Overview	33
2.8	The Protocol Architecture of UMTS Systems	34
3.1	Model of Acknowledged Mode of RLC	42
3.2	Format of AMD PDUs	44
3.3	Format of STATUS PDUs	46
3.4	Structure of a Super-Field	46
3.5	Bit Value for Each Type of Super-Field	47
3.6	Super-Field: NO_MORE	47
3.7	Super-Field: ACK	47
3.8	Super-Field: LIST	48

3.9	SUPer-Field: BITMAP	48
3.10	SUPer-Field: MRW	48
3.11	SUPer-Field: MRW_ACK	48
3.12	Variables of the Transmission and Reception Window	53
4.1	Simulation Model	65
4.2	RLC Process Model of Simulation	67
4.3	State Transition Diagram of RLC Process	70
4.4	Actions on the Event of TTI Expiration	72
4.5	Function: receive_data_pdu()	73
4.6	Function: receive_status_pdu()	75
4.7	Function: generate_ack_list_sufi()	76
4.8	Function: send_pdu()	78
5.1	SDU Discard Rates Comparison	87
5.2	SDU Average Delay Comparison with Out-of-sequence Delivery	88
5.3	The Effect on SDU Average Delay with Different BLERs	94
5.4	The Effect on RLC Throughput with Different BLERs	98
5.5	The Effect on RLC Goodput with Different BLERs	100
5.6	The Effect on SDU Delay with Different BLERs	103
5.7	The Effect on RLC Throughput with Different BLERs	106
5.8	The Effect on RLC Goodput with Different BLERs	107

Chapter 1

Introduction

1.1 Motivations

Due to rapid advances in the areas of wireless communications and the global Internet connection, provision of data services for Internet applications over wireless networks is gaining importance. Most of the Internet applications, such as the File Transfer Protocol (FTP) [Bhu72], web browsing (HTTP) [FGM⁺97], telnet [PR83], e-mail (SMTP) [Pos82], are based on the Transmission Control/Internet Protocol (TCP/IP) architecture. Designed originally for wireline reliable transmission networks, TCP [aUoSC81, Jac88] treats every segment loss as a sign of congestion and invokes its congestion control schemes as a response to a segment loss detected. When the Internet stretches over wireless networks for the most convenient access to the end users, however, TCP has to confront two additional sources of packet losses:

- High rate of transmission errors;
- Drops and resumptions of wireless links.

The above two reasons of a segment loss arise from the unique characteristics of wireless networks compared to a wireline network:

1. Higher bit error rate of wireless links because of co-channel interferences, host mobility, and multi-path fading;
2. Low bandwidth of wireless links due to the limited spectrum;
3. More disconnections because of handoffs and coverage limitations.

These unique characteristics of wireless networks were not addressed by TCP, which assumes that the only reason of segment losses is network congestion. This interpretation results in the invocation of the congestion control and avoidance mechanisms by shrinking its transmission window size and dropping the data transmission rate even if there is no congestion in the network. When TCP mistakenly initiates the combined slow start with congestion avoidance algorithm to slow down the traffic sent over the connection as a response to a segment loss due to the reasons existing uniquely in wireless networks, the TCP performance in terms of throughput and end-to-end delay degrades significantly and unnecessarily.

During the last a few years, several schemes were proposed to combat the problem of the TCP performance degradation over wireless networks [BSAK95, APL⁺95, SVSB02, YB94, BB95]. They can be roughly classified into three categories: Link-layer schemes, end-to-end protocols, and split-connection protocols. A link-layer scheme, such as the schemes in [BSAK95] and [APL⁺95], is employed to hide wireless losses from TCP by re-transmitting lost packets over the wireless link using an Automatic Repeat reQuest (ARQ)

protocol. Two accepted roles of the third generation (3G) wireless networks that are aiming at the provision of high-speed cellular data services, UMTS and CDMA2000, are both using link-layer schemes to improve the performance of TCP in such networks. UMTS systems employ the Radio Link Control (RLC) protocol for flow control and error recovery based on the Multiple Rejects ARQ (MR-ARQ) mechanism [YPM00]. RLC hides transmission errors from upper layers, thus it can reduce the chances of mistaken invocations of the TCP's congestion control mechanisms. As a result, the performance of data traffic over the air interface is improved.

In the past two years, analytical studies on the RLC protocol have been conducted. Nicolas Enderle [EL02] performed an analytical study on the upper layer packet average delay with in-sequence delivery disabled, and the MR-ARQ mechanism throughput was analysed in [YPM00]. However, the analyses are based on some simple cases. For instance, the analytical model of the MR-ARQ mechanism throughput is based on the periodic transmission of STATUS reports between the peer RLC entities.

Since the RLC protocol has many features and options that make the protocol very flexible and adaptive for UMTS operators with many parameter settings to choose from in various radio environments, it is of great worth to evaluate the protocol performance over the UMTS specific physical link using simulation. A few studies [ZS02, XXC⁺02, XCX⁺02] have been conducted to optimise the settings of some parameters based on the upper layers' data traffic model with the web browsing behaviours. The performance evaluation of the RLC protocol needs to be further investigated under different data traffic models.

1.2 Contributions

Several contributions presented in this thesis are listed below.

- **Gain of the optimal value setting of the STATUS Period Timer in RLC**

With a large transmission window and reception window size (2047 PDUs in our simulation), our simulation results show that the best RLC performance can be reached when the STATUS Period Timer is set to be slightly larger than the round trip time. In this setting, the SDU average delay is minimised and the RLC throughput and goodput can be maximised. When the timer value is smaller than the optimal value, even though the RLC throughput can be maximum, the SDU average delay becomes larger and the RLC goodput becomes smaller because of unnecessary retransmissions of AMD PDUs. When the timer value passes the optimal value, the SDU average delay becomes larger and the RLC throughput and goodput go down because of more chances of the transmission window blocking.

- **Gain of the optimal value setting of the Poll Timer in RLC**

Simulation results on the Poll Timer are similar to those on the STATUS Period Timer. Based on a large enough transmission and reception window size, our simulation results show that the Poll Timer value should be no smaller than, but close to, the round trip time in order to reach the best RLC performance. With the block error rate increasing, the Poll Timer should be set to be closer to the round trip time, so that the best RLC performance can be reached.

- **Implementation of the RLC Protocol**

To carry out this research on the performance evaluation of the RLC protocol, a process model of the RLC protocol has been developed with the OPNET simulator [Inc02]. According to [rlc02], the functions regarding to the RLC protocol performance, including the simplified version of SDU segmentation/reassembly, transfer of user data, error correction, duplicate detection, sliding window flow control, and SDU discard, are implemented in our RLC process model. Other functions, including concatenation, padding, and ciphering, are not implemented since they are not related to the performance evaluation of the RLC protocol. Meanwhile, the RLC process model has completely built the RLC polling mechanism, the STATUS transmission mechanism, and the SDU discard mechanism. The simulation can be used to explore the system behavior in the presence of very complex parameter settings, and it will be used to conduct studies on the TCP performance when TCP, as an upper layer protocol, is embedded onto the implemented UMTS RLC layer in our future research.

- **Improvement of an Analytical Result on the SDU Average Delay**

In this study, an existing analytical model for SDU average delay of out-of-sequence delivery, introduced by Nicolas Enderle [EL02], is further investigated. Based on the existing results in which the time between the detection of a missing PDU by the receiver and the transmission of a STATUS PDU to the sender is not computable, an approximate computable average delay of out-of-sequence delivery is obtained. By comparing the simulation results with our computable analytical results, it is seen

that they are close to each other, which verifies our approximate computable result on the SDU average delay of out-of-sequence delivery. It will be used to analyse the SDU average delay of in-sequence delivery mathematically in our future studies.

1.3 Outline

This thesis is organised as follows. Chapter 2 provides the relevant background information. Chapter 3 discusses the Radio Link Control (RLC) protocol in details. Chapter 4 describes the simulation support. Chapter 5 validates the simulation model and presents simulation results. Finally, Chapter 6 concludes this thesis and provides directions for future work.

Chapter 2

Background

This chapter provides the background information relevant to the research on the performance evaluation of the Radio Link Control (RLC) protocol reported in this thesis, and it is organised as follows. Section 2.1 describes the Transmission Control Protocol (TCP), the most widely deployed upper-layer protocol that the RLC protocol provides data service to. Section 2.2 discusses challenges for lossy wireless links to carry TCP data. Section 2.3 summarises various approaches for addressing the mentioned challenges. Section 2.4 introduces third generation (3G) wireless networks with the emphasis on their particular schemes to support TCP-like upper-layer protocols. Section 2.5 is an overview of the performance evaluation of 3G wireless networks. Since our main goal in the study of wireless networks is to improve the performance of TCP over the RLC protocol, this thesis research, which conducts the performance evaluation of the stand-alone RLC protocol, also includes some basic concepts about TCP as part of the background work.

Figure 2.1: (a) The OSI Seven-Layer Model and (b) the TCP/IP Protocol Architecture

2.1 The Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) [aUoSC81] is a protocol at the transport layer in the theoretical Open System Interconnection (OSI) reference model, shown in Figure 2.1(a) [Sta00]. The Transmission Control/Internet Protocol (TCP/IP) suite, shown in Figure 2.1(b), has a layered hierarchy similar to the theoretical OSI model. TCP provides a mechanism for the exchange of data between two or more systems, guaranteeing in-sequence delivery of TCP segments and duplicate detection. It is a connection-oriented inter-process communication protocol that provides the functions of flow control, congestion control, and reliable transmission.

2.1.1 Reliability

The TCP reliable transmission of data is implemented by using a positive acknowledgement (ACK) to acknowledge successful reception of a TCP segment. That is, in order to acknowledge the TCP sender a successful reception of a segment, the TCP receiver sends a positive acknowledgement to the sender. Furthermore, TCP employs cumulative acknowledgements to acknowledge a successful reception of multiple segments, instead of only one segment. It works as follows: an ACK with the acknowledgement sequence number n tells the sending side that the receiving side has already received successfully all segments with the sequence number less than n . However, in the event an unordered segment is received, the TCP receiver will send an ACK carrying the sequence number of the missing segment. As shown in Figure 2.2, when multiple segments are received out-of-order, all their ACKs will include the same sequence number indicating the expected segment to be received in sequence, and the duplicate ACKs are received by the TCP sender. If one segment is missing and an ACK carrying the sequence number of the segment is received, the TCP sender may not be able to know the segment is lost. When multiple duplicate ACKs (normally 3) are received, the TCP sender realizes a loss of a segment in the duplicate ACKs, and retransmission of the segment is performed.

Another approach for the TCP sender to notice a segment loss is the expiration of a Retransmission Time-Out (RTO) timer. TCP keeps a RTO timer that starts every time a segment is sent. If no ACK is received when the RTO expires, the segment is considered to be lost, and the TCP sender will retransmit the segment.

The problem of segment losses is critical to TCP because it affects the TCP's interpre-

Figure 2.2: Multiple Duplicate ACKs

tation of segment losses as well as the TCP's detection about the losses. TCP treats every loss of a segment as a sign of congestion because it was originally designed for wireline networks with very high transmission reliability. Upon the detection of a segment loss, TCP will invoke its congestion control mechanism to alleviate the problem, as will be discussed in Section 2.1.3.

2.1.2 Flow Control

TCP uses a variable-sized sliding window scheme to solve the end-to-end flow control problem. The sliding window size, meaning the number of segments that the TCP sender is allowed to send without waiting for further acknowledgements, is the minimum of the receiver's advertised window size and the congestion window size. The receiver's advertised window size is decided by the available buffer space at the receiver, and allows the receiver to control the sender not sending more traffic than the receiver can tolerate. Hence, TCP allows each ACK from the receiver to contain an advertised window size that specifies how many additional bytes of data the receiver prepares to accept. The congestion window size, which is also used in the TCP's congestion control schemes (to be discussed in Section 2.1.3), is calculated dynamically by the TCP sender based on the feedbacks from the network, and it can reduce the chance of overloading in the intermediate hosts. Thus, TCP allows the sliding window size to vary over time. The size of unacknowledged data should be always smaller than the sliding window size.

When old segments are acknowledged, the sliding window moves and new segments will be sent to the receiver.

2.1.3 Congestion Control

The congestion window, which was mentioned in Section 2.1.2, is used by the TCP sender to provide the congestion control over the Internet between the TCP sender and the TCP receiver. At any given time, the TCP sender is not allowed to have the sliding window size (discussed in Section 2.1.2) to exceed the minimum of the receiver's advertised window

size and the congestion window size.

The following algorithms, included in [Ste97], are commonly used in today's TCP implementations.

2.1.3.1 Slow Start

When a TCP connection is established, the TCP receiver's advertised window size is already learned by the TCP sender. However, the network capacity, which decides the congestion window size, is still unknown to the TCP sender. Therefore, TCP uses a slow start mechanism [Jac88] to explore the network capacity and to determine the congestion window size dynamically. At the beginning, the congestion window size is set to be one segment, and the TCP sender sends only one segment to the receiver and then waits for its acknowledgement. If a positive acknowledgement comes back, the TCP sender assumes that the network is able to transmit at least one segment. Then the TCP sender increases its congestion window size by one segment and sends two segments to the receiver. If the two segments' positive ACKs come back to the TCP sender, the TCP sender will increase the congestion window size by two segments and send four segments in the next transmission round to the receiver. Hence, in the slow start phase, the congestion window size increases by one segment for each time at which a positive ACK is received. In other words, the congestion window size doubles for each full window worth of segments that are positively acknowledged. This is referred to as an exponential increase of the congestion window size. The congestion window size increases exponentially until it reaches the value of *ssthresh*, called "slow start threshold." It is a variable maintained by the TCP sender, and usually set to be 64,000 bytes at the beginning of a TCP connection. Because TCP attributes segment

losses to the network congestion, it immediately enters the congestion avoidance stage.

Figure 2.3 [Vai01] is an example of a slow start and a congestion avoidance phases, in which the size of each TCP segment is assumed to be 8,000 bytes.

Figure 2.3: Slow Start and Congestion Avoidance

2.1.3.2 Congestion Avoidance

As soon as the congestion window size exceeds the value of *ssthresh*, it is inappropriate to increase the congestion window size exponentially as in the slow start stage. Instead, a scheme called congestion avoidance with a linear increase of the congestion window size is used to examine the network capacity. In this stage, the congestion window size grows by only one segment for each full window worth of segments that have been positively acknowledged, as illustrated in the example of Figure 2.3.

The slow-start and the congestion avoidance mechanisms can prevent the TCP sender from sending too many segments into an already congested network. As ACKs come back

to the TCP sender, its sliding window size increases and eventually is limited by the receiver's advertised window size.

2.1.3.3 Congestion Control

As we discussed in Section 2.1.1, there are two ways for the TCP sender to detect a segment loss. One is through the RTO's time-out, and the other is by receiving three or more duplicate ACKs. In either the slow start stage or the congestion avoidance stage, if a segment loss is detected through the RTO's time-out, retransmission of the lost segment is performed, the variable *ssthresh* is reset to be half of the current congestion window size, and the congestion window size is reset to one segment. It is followed by a slow start stage with the value of *ssthresh* being half of the congestion window size before it is reset. Figure 2.4 [Vai01] is an example in which a segment loss is detected through the RTO's time-out in the congestion avoidance phase. If a segment loss is detected by the TCP sender through duplicate ACKs in either the slow start stage or the congestion avoidance stage, retransmission of the lost segment is performed, the variable "ssthresh" is reset to be half of the current congestion window size, and the congestion window size is reduced by half. It will be followed by a congestion avoidance phase. The congestion control caused by the reception of three or more duplicate ACKs is more aggressive than that caused by the RTO's time-out.

Figure 2.4: Congestion Control: Time-out

2.2 TCP over Wireless Networks

In this section, we mainly focus on the unique characteristics of wireless networks, and for each of the characteristics we will discuss how it may impact the TCP performance.

2.2.1 Overview

A conventional cellular network, in which one or more mobile hosts are connected directly to a base station that is connected to a wireline core network, is the network model for discussing the impacts of wireless networks' characteristics on the TCP performance. By using the connection between a mobile host and a base station over a wireless channel, all the Internet applications based on the TCP/IP protocol suite, such as web browsing (HTTP) [FGM⁺97], file transfer (FTP) [Bhu72], e-mail (SMTP) [Pos82], telnet [PR83], can run on a mobile host.

2.2.2 Main Characteristics of Wireless Networks Compared to Wire-line Networks

2.2.2.1 Higher Bit Error Rate

A significantly higher bit loss rate of wireless links is an essential difference between wire-line and wireless networks. Reasons of a higher bit error rate in wireless networks can be higher co-channel interferences, host mobility, multi-path fading, or disconnections due to coverage limitations [yHS02]. Experimental studies have reported that the packet error rate in wireless networks could range from 1% up to 10% [yHS02], which cannot be tolerated because of a significant degradation of the TCP performance.

As described in Section 2.1, TCP decreases its congestion window size by half every time at which a segment loss is detected. The quick shrink of the congestion window size is based on the assumption that all segment losses in the network are due to congestion. This is an effective way to avoid system collapse due to congestion. Unfortunately, TCP can not differentiate between congestion-introduced segment losses and random segment losses due to wireless lossy links. Hence, TCP will mistakenly interpret a segment loss due to the wireless lossy link as a sign of network congestion, and cut its congestion window size by half, thus reducing the throughput of the TCP connection unnecessarily.

2.2.2.2 Lower Bandwidth of Wireless Links

Wireless networks are also characterised by the availability of significantly lower bandwidths than that in wireline networks, and the effective bandwidths could be further lower by adding up the high bit error rate characteristics of wireless links. In a wireless network

with low bandwidths, the transmission window size is typically small, which means there are fewer packets on the fly. Because of the higher bit error rate of wireless links, duplicate ACKs could be lost and may not be received by the TCP sender. Consequently, more retransmissions of segments because of the timer RTO's time-out are performed than those due to the reception of three or more duplicate ACKs. This implies that the congestion window size is decreased to one segment more often than it is reduced by half. As discussed in Section 2.1.3.3, the congestion control caused by the reception of three or more duplicate ACKs is more efficient than that caused by the RTO's time-out. As a result, the TCP throughput is reduced more than expected in wireless networks. Even worse, the bursty nature of the wireless links' error models can result in the timer RTO increasing to a large value [yHS02], and the TCP throughput degrading even more.

2.2.2.3 Larger and Varying Delay

The delay along a wireless link for a TCP connection could be affected by the following factors:

- **Low Bandwidths** When the bandwidth of the wireless link is low, the transmission delay, which is equal to the division of the packet size divided by the channel bandwidth, is large, contributing to a large end-to-end delay.
- **Long Latency in the Wireless Components** Data traffic must traverse through more components in the end-to-end wireless systems and more processing time is expected to reach its destination.

- **The Bandwidth Sharing with Voice Traffic** Currently most wireless networks are overlayed on the infrastructures built for voice traffic, which is delay-sensitive. In these circumstances data traffic typically shares the available channels with voice traffic, which has a higher transmission priority than data traffic. So a period during which there are no free channels to allocate data traffic may cause a significant increase in the end-to-end delays of TCP segments, and therefore the delay becomes dependent on the amount of voice traffic.

The delay over wireless links is expected to be large and varying because of the above reasons. Hence, the average Round Trip Time (RTT) maintained in the TCP sender becomes large, and the RTO value, which is calculated based on the average RTT, increases. Since TCP has to wait for a RTO amount of time before retransmitting a lost segment on the absence of duplicate ACKs, which can happen in wireless networks (discussed in Section 2.2.2.2), the TCP performance degrades.

2.2.2.4 More Disconnections

Since users are mobile and cellular networks are composed of overlapping cells, handoffs are needed when a mobile user moves from one cell to another. Mobile locations are not always known, so there could be some areas which cannot be reached by any base station. Hence, a mobile user may experience frequent disconnections with a long period of time because of handoffs and coverage limitations. At the beginning of such a situation, a number of RTO time-outs are received, and retransmissions are performed by the TCP sender. The average RTO maintained by the TCP sender will be increased to a large value. After a period of time, TCP will get into a steady state in which it checks periodically to

determine if the connection is up [yHS02]. When the connection is up, TCP restarts with the slow start phase and with the window size one segment. Hence, such frequent and long-time disconnections result in the TCP throughput degradation.

2.3 Schemes of Improving Performance of TCP over Wireless Networks

So far, we have looked at the key mechanisms of TCP and several unique characteristics of wireless networks. With each characteristic identified, we have also discussed impacts on the TCP performance. In the following, some approaches that attempt to improve the TCP performance over wireless networks are discussed. Due to the restriction of space, we mainly focus on the higher bit error rate characteristic of wireless networks, instead of addressing other issues of wireless networks, such as mobility, large delay, and handoffs.

According to the modifications of the TCP/IP suite needed by the proposed schemes, they can be classified into the three different categories of approaches: (1) Link-layer protocols, (2) end-to-end protocols, and (3) split-connection protocols. For each category of approaches, we will give an overview, consider some properties that each category has, including both advantages and disadvantages, and list one or more proposed schemes falling into that category.

2.3.1 Link-Layer Protocols

The protocols of this class attempt to hide the characteristics of the wireless lossy links from the TCP layer by using some special link layer schemes over the physical layer. Such approaches are typically transparent to the overlaying TCP. Normally data packets are buffered at the Base Station and retransmissions of the packets that are lost due to transmission errors over wireless links are performed between the Base Station and the Mobile Host as the responses to Automatic Repeat reQuest (ARQ) messages, as shown in Figure 2.5 [Vai01]. Consequently, the Fixed Host is exposed mainly to congestion-introduced segment losses.

Figure 2.5: Link-Layer Retransmission

Link layer protocols thus have the following properties:

- They hide a wireless lossy link from the transmission protocol, thus TCP observes a

more reliable wireless link;

- They do not require any protocol modifications in the Fixed Host and the Mobile Host; but additional buffers and the capability of packet retransmissions are required to implement at the Base Station;
- Since there is a packet retransmission scheme in both the TCP layer at the Fixed Host (shown in Figure 2.5) and the Link layer at the Base Station, there may be competitions and redundant retransmissions initiated by the the Fixed Host and the Base Station, when the delay over the wireless link is a significant part of the TCP end-to-end (between the Fixed Host to the Mobile Host) delay. This will result in degradation of the TCP throughput.

In summary, Link-layer retransmission mechanisms could improve TCP performance [CB00, WL99, KQC01, LD02], and there are mainly three factors deciding their effects on the TCP performance. The first one is the number of retransmission attempts. Studies [FL03, CB00] have found that the TCP performance starts to degrade dramatically when TCP observes a segment loss rate higher than 1% or 2%. Therefore, a Link-layer protocol must be persistent enough to recover most of the transmission errors and present to TCP a packet loss rate lower than 1% or 2%. The second factor is packet reordering. The retransmissions of lost packets may result in the reordering of packets and invoke the TCP's congestion control mechanism. The third factor is the retransmission competitions between the TCP layer of the Fixed Host and the Link layer of the Base Station, since the TCP layer of the Fixed Host may timeout before a Link-layer retransmission of the Base Station is performed to recover errors. When both of them attempt to recover the same errors by

retransmission, wireless bandwidths will be wasted and the performance degrades. Therefore, a Link-layer protocol must be persistent enough to recover errors, but does not make the TCP timeout.

Two digital cellular wireless technologies, TDMA and CDMA, are both using such approaches. However, they have a fundamental difference on the error recovery. While the TDMA wireless technology guarantees reliable, in-sequence delivery of link-layer frames [BPSK97], the CDMA technology only makes partial error recovery of link-layer frames and pass the control of un-recovered frames to the TCP layer. Since the two major roles of 3G wireless networks, CDMA2000 and UMTS, are both based on the CDMA technology, they both utilise the link-layer approaches to partially recover link-layer frames with transmission errors by defining different retransmission schemes. We will see those protocols in Section 2.4 and the next chapter. Other proposed schemes such as the snoop protocol [BSAK95] and AIRMAIL [APL⁺95] also belong to this category.

2.3.2 End-to-End Protocols

The end-to-end protocols attempt to make the TCP sender handle segment losses through the following two techniques: (1) An updated form of ACKs, the Selective ACKnowledgements (SACKs) [MMR96] is used to allow the TCP sender to make a recovery of multiple lost segments in a window, but not invoke RTO's time-out; (2) a new TCP scheme called the Explicit Loss Notification (ELN) mechanism [DJ01] is employed to make the TCP sender have the ability to differentiate the segment losses introduced by congestion and other types of segment losses. While such approaches retain the end-to-end data transfer service between the transport layer of the TCP sender and the transport layer of the TCP

receiver, the protocol stacks of both the TCP sender and the TCP receiver have to be modified. Because a more intelligent congestion control mechanism and an updated reliability scheme are used at the TCP sender and the TCP receiver, the TCP performance could be improved to a level that can not be reached by using the other two classes of protocols.

Properties of this category of approaches are summarized as follows:

- The updated congestion control and reliability schemes used by the TCP sender and the TCP receiver are much more complicated and sophisticated, because of the added ability to tell the type of segment losses launched by congestion from other types of losses;
- They may be used to thoroughly and completely solve the problems encountered in the wireless networks, including not only the problem of higher bit error rate of wireless links, but also the problems of handoffs and mobility;
- The congestion-loss detection mechanism, a fundamental to make such protocols work, may fail in some cases [BV99].

WTCP [SVSB02] is a proposed transmission protocol of this category.

2.3.3 Split-Connection Protocols

Basically, the category of split-connection protocols splits each TCP connection between the TCP sender and the receiver into two separate connections at the base station. One is a TCP connection between the sender and the base station, and the other one is a connection between the base station and the receiver, and it may be a specific protocol suitable for the wireless environment, instead of a TCP connection. As a consequence, two simultaneous

connections are maintained at the base station after a TCP connection between the sender and the receiver is established. Hence, more functions are required to add to the base station.

Properties of such class of protocols are summarized as follows:

- The end-to-end data transfer services between the transport layer of the TCP sender and the transport layer of the TCP receiver are destroyed since the end-to-end TCP connection is split at the base station into two connections;
- The schemes of connection management, flow control, and congestion control at the wireless part can be employed independently;
- Since two different protocols at the transport layer are maintained concurrently at the base station, on one hand, operations at the base station are expected to be more sophisticated, and on the other hand, all segments have to be transported to the transport layer at the base station, and go down again through the protocol stack. Thus more overhead and processing delay are introduced into the end-to-end data.

The proposed schemes of this category are the scheme in [YB94] and Indirect-TCP [BB95].

2.4 The Third Generation (3G) Wireless Networks

2.4.1 Three Generations of Wireless Cellular Networks

Wireless networks provide convenience and freedom to the consumer, and have been developing rapidly in the recent two decades thanks to the advances in technology. Commer-

cial wireless networks for mass communication traffic are often cellular systems, which are composed of multiple overlapped small-ranged radio coverage to reuse the frequency resource geographically. So we usually use wireless networks to represent the currently widely used cellular wireless networks in the rest of the thesis.

The first generation (1G) wireless cellular networks are analog systems. Each communication channel is a Frequency Modulated (FM) carrier, each cell typically has only a few carriers due to the scarcity of the available frequency band. The most widely deployed 1G (analog) mobile phone systems include the Advanced Mobile Phone System (AMPS), the Nordic Mobile Telephone (NMT), and the Total Access Communications Systems (TACS).

The second generation (2G) systems are all digital networks. 2G standards include Global System for Mobile Communications (GSM), IS-136 or Digital AMPS (D-AMPS), IS-95 or cdmaOne. GSM and IS-136 are Time-Division Multiple Access (TDMA) systems, while IS-95 are Code-Division Multiple Access (CDMA) systems. TDMA divides the transmission time of a carrier into time slots and allocate to each user unique time slots. Therefore, users transmit in their own time slots without interfering each other. CDMA is a spread-spectrum technology, and a relatively wide frequency band is shared by multiple users. Each user modulates its digital packets with a unique key, or a Walsh code, and its expecting receiver can pick up its signals with that key. The 2G systems focus on voice service, and their data capacity is very limited. General Packet Radio Service (GPRS) is designed to enable circuit-switched GSM and D-AMPS to support packet data. It is often referred to as a 2.5G technology.

By the definition given by the International Telecommunication Union (ITU), the wireless networks whose Radio Frequency (RF) interface can deliver data rates of 144 kbps or

higher are called 3G wireless networks [Leh02]. Currently the Wideband CDMA (UMTS) systems and the CDMA2000 systems satisfy this criterion. Both systems are designed to support a high speed delivery of multimedia data and IP services while providing a reliable and high quality-voice service.

2.4.2 CDMA2000 Systems

CDMA2000 is a 3G wireless networks standard that provides a seamless and evolutionary upgrade of a high data rate for the current users of 2G and 2.5G CDMA technologies based on cdmaOne (IS-95) CDMA standards while supporting 3G services as defined by the ITU's IMT-2000 standards for new users. It deploys the assigned RF spectrum at 800 MHz and 1900 MHz.

2.4.2.1 The System Architecture

Basically, CDMA2000 provides protocols and services that correspond to the bottom two layers of the OSI reference model (Figure 2.1(a)). The two bottom layers are layer 1, the Physical layer, and layer 2, the Link layer. Figure 2.6 [KKLN98] describes the system architecture of CDMA2000 designed to provide voice, packet data, circuit data, and concurrent voice and packet data services. From the point of view of CDMA2000 systems' layered structure, the applications and upper-layer protocols are referred to layer 3 to layer 7 of the OSI reference model, and layer 2 can be further divided into the Link Access Control (LAC) and the Medium Access Control (MAC) sublayers.

Figure 2.6: CDMA2000 Systems' Architecture

2.4.2.2 The Physical Layer

The physical channels (PHCHs) in CDMA2000 can be divided into two classes: Dedicated Physical Channels (DPHCHs), which transmit the information between the base station and a particular mobile host, and Common Physical Channels (CPHCHs), which carry information between the base station and several mobile hosts. The both types of channels can be further divided into multiple sub-channels (Walsh codes) [YH], and each sub-channel

transmits a physical layer frame length in one 20-ms time slot. Hence, multiple frames can be sent out in one time slot, so as to achieve high speed wireless inter-connections. The first phase of CDMA2000 networks, based on IS-2000 specifications, is designed to support data rates up to 316.8 kbps [KY02].

2.4.2.3 The LAC Sublayer

The LAC sublayer is introduced to support highly reliable point-to-point transmission over the air interface for the signalling services and the circuit data services [KKLN98]. It is required to provide a function of guaranteeing a reliable in-sequence delivery of packets. It could be implemented by using an ARQ scheme over the wireless links. The LAC sublayer could be omitted however, if the required services such as in-sequence delivery requirement have been provided by the lower sublayers, such as the Radio Link Protocol (RLP) in MAC sublayer.

2.4.2.4 The MAC Sublayer

The MAC sublayer of CDMA2000 systems can provide complex multimedia services with the capability of satisfying different QoS requirements from each service. It mainly consists of three components: The MAC control states, the Radio Link Protocol (RLP), and the Multiplexing and QoS control [KKLN98]. MAC control states are used to control the data services' access to the physical channel. They can provide congestion control between multiple services for a single user and resolve channel access competitions between multiple users. RLP can guarantee a reliable transmission over a wireless lossy link, thus provide a best effort delivery of packets. Since it is the scheme used in the CDMA2000 systems

to improve the TCP performance over wireless lossy links, we will explain this protocol in details in the next subsection. The multiplexing and QoS control are used to mediate conflicting transmission requests according to their QoS requirements and transmission priorities.

2.4.2.5 Radio Link Protocol (RLP)

The Radio Link Protocol Type 3 [rlp00] in the CDMA2000 systems is a pure Negative Acknowledgement (NAK) radio link protocol based on the selective repeat Automatic Repeat reQuest (ARQ) mechanism [KK00]. The receiver only sends NAKs to request retransmission of RLP data frames that were not received correctly. RLP employs this NAK scheme to partially recover frame errors. The NAK scheme is specified by 3 parameters: *NUM_ROUNDS*, *NAK_COUNT*[], and *ROUND_INTERVAL*. *NUM_ROUNDS* is the number of retransmission rounds tried before abortion of a frame recovery; *NAK_COUNT*[] is an array with (*NUM_ROUNDS* + 1) entries, each of which specifies the number of NAKs to be sent in a given round; and *ROUND_INTERVAL* is the period from the time at which one round starts to the time at which its next one starts. *ROUND_INTERVAL* should be no shorter than the round trip time of the wireless part. When the RLP receiver detects a frame in error, it sends a NAK frame to request for retransmission and starts a timer that is set to be *ROUND_INTERVAL*. Upon receiving a NAK frame, the RLP sender retransmits the requested data frame. When the timer expires, the RLP receiver sends *NAK_COUNT*[1] NAKs to the RLP sender and resets the timer. This process continues until the number of timer expirations reaches *NUM_ROUNDS* (normally 3). Then the RLP receiver will give up the attempt to recover the data frame in error, and passes an un-recovered message of

the frame to the upper layer such as the TCP layer.

At the transmitting side, RLP expects from its upper layer a stream of bits. It formats the bit streams into RLP frames by adding a RLP header on each of them. For each RLP frame, a unique 12-bit sequence number, called *SEQ*, is assigned in its header. And then RLP passes them to the transmission buffer for transmission. At the receiving side, RLP expects to receive RLP frames.

During the data transfer phase, the RLP protocol maintains three counters: $L_V(S)$, $L_V(R)$ and $L_V(N)$. $L_V(S)$ is the *SEQ* of the next frame to be transmitted. $L_V(N)$ is the *SEQ* of the oldest frame needed for in-sequence delivery, and $L_V(R)$ is the *SEQ* of the next frame expected to be received, which is the *SEQ* of the newest frame received plus one. Since the RLP protocol delivers bit streams in sequence to the upper layer, a re-sequencing buffer is needed to store the frames received out-of-order.

When a RLP frame with the sequence number *SEQ* is received, there are a few cases that could happen according to the counters for receiving frames, $L_V(R)$ and $L_V(N)$. If two or more NAKs were sent requesting for retransmission, and the first retransmitted frame is received correctly, then the retransmitted frame requested by the second NAK frame will be an outdated one. It is addressed in Case 1 of Table 2.1. If the re-sequencing buffer is not empty, there must be one or more frames missing. Then if the received one is the oldest frame to be recovered, the action in Case 2 of Table 2.1 happens. If the received frame is a missing one but not the oldest missing frame, the action in Case 4 of Table 2.1 happens. If the received frame is just the next one expected to be received, the action in Case 5 of Table 2.1 happens. However, the re-sequencing buffer may be empty, and the received frame is the next one expected to be received. This is addressed in Case 3 of

Table 2.1. The last possible situation is that the frame is received after one or more frames transmitted for the first time were not received by RLP, in which the action in Case 6 of Table 2.1 is invoked.

Case NO.	Conditions	Actions
1	$SEQ < L.V(N)$ or the frame has been previously received	Discard it
2	$SEQ == L.V(N)$ and $SEQ < L.V(R)$	Update $L.V(N)$ to the SEQ of next oldest missing frame; pass received frames with sequence numbers up to $L.V(N) - 1$ to the upper layer
3	$SEQ == L.V(N)$ and $SEQ == L.V(R)$	Increment both $L.V(N)$ and $L.V(R)$ by 1; pass the received frame to the upper layer
4	$SEQ > L.V(N)$ and $SEQ < L.V(R)$ and it not received before	Store the frame in the re-sequencing buffer
5	$SEQ > L.V(N)$ and $SEQ == L.V(R)$	Increment $L.V(R)$ and store the frame in the re-sequencing buffer
6	$SEQ > L.V(R)$	Send NAKs to request for retransmission of frames with sequence numbers from $L.V(R)$ to $SEQ - 1$, inclusive; set up NAK timers for the next round

Table 2.1: The RLP Data Transmission Procedure

Studies have shown that the NAK scheme of RLP, including the number of rounds of NAK and the number of NAKs per round, is a major role affecting the TCP performance [KKMN00, KY02].

2.4.3 UMTS Systems

Universal Mobile Telecommunications System (UMTS), developed from GSM wireless technologies, is another accepted 3G standard. UMTS is also referred to Wideband CDMA (WCDMA) since the RF channels that UMTS systems require are four times as wide as those required by CDMA2000. The assigned RF spectrum for UMTS is at 1900 MHz and

2100 MHz [Lch02].

2.4.3.1 The Protocol Architecture

The UMTS network architecture consists of three components: The Core Network (CN), the UMTS Terrestrial Radio Access Networks (UTRAN), and the User Equipment (UE). Between the UTRAN and the CN is the general wireline interface called Iu, and between the UTRAN and the UE is the radio interface called Uu. Figure 2.7 [GKSS00] describes an architecture of a UMTS system. There can be several Radio Network Subsystems (RNSs) in a single UTRAN. RNSs can be connected to each other through the Iur interface. The RNS consists of the Radio Network Controller (RNC) and one or more base stations (Node B). A Node B is connected to the RNC through the Iub interface. There could be one or more Node Bs serving a cell. All these components can be found in details in the Third Generation Partnership Project (3GPP) [CMO99].

Instead of bottom two layers' specifications in CDMA2000 systems, UMTS specifies bottom three layers' protocols and services corresponding to the OSI reference model. It includes layer 1, the Physical layer, layer 2, the Link layer, and part of the layer 3, the Network layer, as shown in Figure 2.8 [GKSS00]. Layer 2 includes the Medium Access Control (MAC), the Radio Link Control (RLC), the Broadcast/Multicast Control (BMC), and the Packet Data Convergence Protocol (PDCP) sublayers. Layer 3 of UMTS systems includes the Radio Resource Control (RRC). It only exists in Control plane (C-plane) configuration, instead of User plane (U-plane) configuration. It handles the C-plane signalling of the UTRAN to the UE, and is responsible for setting, modifying and releasing all the lower layer protocol entities [umt03b]. The ellipses in Figure 2.8 represent the Service Ac-

Figure 2.7: UMTS Systems' Architecture Overview

cess Points (SAPs) at the interface between sublayers. For instance, between the Physical layer and the Link layer, the SAPs provide the logical channels.

2.4.3.2 The Physical Layer

The Physical layer is responsible for data or voice transmission by means of radio medium. Its main tasks include transforming the frames coming from layer 2 into radio signals, and translating received radio signals into logical channels to be received by layer 2 [umt02].

2.4.3.3 The MAC Sublayer

The MAC layer is mainly responsible for scheduling the radio bearers with different QoS requirements for data and voice traffic flows, and optimising the use of the radio medium

Figure 2.8: The Protocol Architecture of UMTS Systems

by sharing it dynamically among users [umt03a].

2.4.3.4 The RLC Sublayer

Like the RLP sublayer in CDMA2000 systems, the RLC protocol is responsible for traffic flow control and error control. Due to the unique error-prone characteristics of wireless links compared with wireline links, the RLC protocol needs fine tunes and configurations to make its upper layers such as TCP observe a much more reliable wireless transmission link. We will discuss the RLC protocol in more details in the next chapter.

2.4.3.5 The PDCP Sublayer

The PDCP sublayer is present only in the U-plane, and serves as a convergence layer between the Network layer and the RLC sublayer. It can allow the lower layers' functions and operations to be independent of the Network layer protocols such as IPv4 and IPv6, and provides a header compression function for network protocols such as IPv4 and IPv6 [umt01].

2.4.3.6 The BMC Sublayer

The BMC sublayer only exists in the U-plane, providing the service of cell broadcast. Only one BMC entity can exist per cell [GKSS00].

2.5 Performance Evaluation Overview of 3G Wireless Networks

The primary concerned performance criteria used by studies on the TCP performance over 3G wireless networks are summarized in Table 2.2.

layer	criteria	remarks on existing studies
TCP layer	TCP throughput end-to-end delay	often assume bulk file transfer with FTP often assume web browsing with HTTP
Link layer	link-layer throughput link-layer delay of wireless part upper-layer packet discard rate	can be evaluated for stand-alone link layer can be evaluated for stand-alone link layer can be evaluated for stand-alone link layer

Table 2.2: Performance Criteria of TCP over 3G Wireless Networks

Among those studies, most of them are concerned about the TCP or link-layer throughput assuming a single large file transfer with FTP [BZRO00, Bao96, KKMN00, LV01],

while a few others are concerned with the end-to-end delay of ordered frame delivery or the link-layer delay of wireless part assuming multiple short web page transfers with HTTP [ZS02, XCX⁺02, XXC⁺02]. They all tried to get one or more optimal parameters' settings of the specified link layer protocol when the best performance was reached. Some of them [XCX⁺02, XXC⁺02, ZS02] took the link-layer performance criteria as the point of interests by singling out the specified link layer protocol without accounting for the upper layers, and others [BZRO00, Bao96, KKMN00, LV01] evaluated the TCP performance by overlaying the TCP layer on the implemented link layer protocols.

There are a number of factors in the TCP layer, the link layer, and the physical layer that may affect the TCP performance over 3G wireless networks, and they are summarized in Table 2.3. Evaluations are normally conducted by varying factors of interests and assigning constant values to the others.

Layer	Factor	Remarks
Physical layer	Transmission error rate	Variable with wireless environment
	Error bursty nature	Depending on the error models
	Wireless link latency	Can be fixed
	Data rate	Can be fixed
Link layer	Various parameters	Variable
TCP layer	Congestion window size	Variable
	Round trip time	Variable
	Maximum segment size	Can be fixed

Table 2.3: Factors Affecting the TCP Performance

2.6 Summary

This chapter provided the background information relevant to the research in this thesis on the performance evaluation of 3G UMTS wireless networks. It discussed the TCP flow

control and congestion control schemes, the main characteristics of wireless networks and their impacts on the TCP performance, the proposed protocols of improving the performance of TCP over wireless networks. Finally, the 3G wireless networks were introduced and the performance evaluation of TCP over 3G wireless networks was briefly discussed.

Chapter 3

The Radio Link Control (RLC) Protocol

This chapter describes the RLC protocol in detail. Section 3.1 discusses the purpose of RLC in UMTS systems. Section 3.2 describes the service modes that the RLC can provide. Section 3.3 gives the formats of two types of RLC Protocol Data Units (PDUs). Section 3.4 defines parameters used in the protocol critical for the RLC performance. Section 3.5 and Section 3.6 discuss the operations and variables maintained in the transmitting and receiving sides, respectively. One more set of operations invoked by the expiration of the current Transmission Time Interval (TTI) is described in Section 3.7.

3.1 Overview

The Radio Link Control (RLC) protocol is a layer 2 protocol in UMTS (Figure 2.8), which sits above the Medium Access Control (MAC) sublayer and below the Radio Resource Control (RRC) sublayer and is used for error control and flow control based on the Multiple Rejects Automatic Repeat reQuest (MR-ARQ) mechanism [YPM00]. It is responsible

for the efficient transmission or retransmission under variable data rates of wireless channels, controls the radio link quality and hides a lossy radio link (air interface) from upper layers such as TCP by using the fast and efficient error correction mechanism. It consequently minimises the errors conceived by upper layers such as TCP. The protocol can be configured by upper sublayers such as RRC sublayer (discussed in Section 2.4.3.1) to provide different levels of Quality of Service (QoS) by adjusting the maximum number of retransmissions [rlc02].

3.2 The Three Service Modes

Three types of services are provided by the RLC protocol to upper layers. They are Transparent Mode (TM), Unacknowledged Mode (UM), and Acknowledged Mode (AM). For each of these three service modes, RLC consists of two entities, the transmitting RLC entity and the receiving RLC entity. The transmitting RLC entity is also called the transmitting side, and the receiving RLC entity is called the receiving side. The transmitting side transmits RLC Protocol Data Units (PDUs), and the receiving side receives RLC PDUs. Which service mode is chosen is decided by the type of traffic to be transmitted. For instance, TM should be used when the traffic to be transmitted are voice, and AM will be configured for packet data services. The transmitting and receiving procedures for the three types of service modes, are described as follows.

3.2.1 Transparent Mode (TM)

Transparent Mode (TM) offers service for transmitting upper layer PDUs without adding any protocol information. In this mode, segmentation and reassembly can/can't be configured by upper layers. If segmentation/reassembly is configured by upper layers and the RLC Service Data Unit (SDU) size, which is normally hundreds of bytes, is larger than the Transparent Mode Data (TMD) PDU size (usually dozens of bytes) used by the lower layer for that Transmission Time Interval (TTI), the transmitting side will segment RLC SDUs to fit the TMD PDUs without adding RLC headers. All the TMD PDUs carrying the segments belonging to the same RLC SDU are sent in one TTI, and no segments from another RLC SDU can be sent in that TTI. On the contrary, if the sizes of RLC SDUs are always smaller than the sizes of TMD PDUs and segmentation/reassembly has not been configured by upper layers, then more than one RLC SDU can be sent in one TTI by putting one RLC SDU in one TMD PDU, but all TMD PDUs in the same TTI must have an equal length. The resulting TMD PDUs are submitted to the lower layer and will be transmitted to the destination node through a wireless channel. When the receiving side receives TMD PDUs through the configured logical channel from the lower layer, it reassembles them into RLC SDUs if segmentation/reassembly is configured by upper layers, and delivers RLC SDUs to upper layers.

3.2.2 Unacknowledged Mode (UM)

Unacknowledged Mode (UM) offers service for transmitting higher layer PDUs without guaranteeing delivery to the destination node or the peer RLC entity. If a RLC SDU comes

from upper layers, the transmitting side segments the RLC SDU into Unacknowledged Mode Data (UMD) PDUs of appropriate size, if it is larger than the length of available space in the UMD PDU. So an UMD PDU can contain segmented and/or concatenated RLC SDUs and padding bits to ensure a valid length. Length indicators are used to define the boundaries between RLC SDUs and whether padding is included within an UMD PDU. Ciphering can be configured by upper layers. If it is configured, ciphering happens before an UMD PDU is submitted to the lower layer and the UMD PDU header must not be ciphered. Then UMD PDUs are submitted to the lower layer through a logical channel. When the receiving side receives UMD PDUs, it deciphers them if ciphering is configured, removes RLC headers from received UMD PDUs, and reassembles them into RLC SDUs. Then RLC SDUs are delivered to upper layers by the receiving side.

3.2.3 Acknowledged Mode (AM)

Acknowledged Mode (AM) offers service for transmitting higher layer PDUs and guarantees delivery to the destination host or the peer RLC entity. In this mode, both the transmitting and receiving sides can be configured to utilise one or two logical channels connecting to the lower layer. Since AM is designed to provide reliable transmission of packet data and is the default mode for handling interactive and background applications, we assume that the AM is the RLC operational mode for the following discussion, configuration, and simulation.

Figure 3.1 shows the model for the AM when one logical channel is used. The transmitting operations (to be discussed in Section 3.5) will be invoked when the transmitting side receives RLC SDU(s) from upper layers, the receiving operations (to be discussed in Sec-

tion 3.6) are triggered when the receiving side receives Acknowledged Mode Data (AMD) PDU(s) or STATUS PDU(s) through the configured logical channel. The TTI operations (to be discussed in Section 3.7) are invoked each time at which an old Transmission Time Interval (TTI) expires and a new one starts.

Figure 3.1: Model of Acknowledged Mode of RLC

3.3 PDU Formats and Parameters

In the AM there are five types of PDUs used: AMD PDU, STATUS PDU, Piggybacked STATUS PDU, RESET, and RESET ACK PDU. AMD PDUs contain user data (RLC SDUs). STATUS PDUs and piggybacked STATUS PDUs contain the protocol control information exchanged between the RLC peers, such as retransmission, deletion, or window moving messages. RESET PDU is used to reset all protocol states, variables and timers of the peer RLC entity, and RESET ACK PDU is an acknowledgement to a RESET PDU. Since we mainly use the first two types of PDUs in this study, we mainly focus on the formats of these two types of PDUs.

3.3.1 AMD PDUs

An AMD PDU is used to transfer user data, the STATUS report request, and the protocol control information if a piggybacked STATUS PDU is included in that AMD PDU. The important part of an AMD PDU is the data part (user data). To perform retransmissions of a particular AMD PDU and reassemble AMD PDUs into RLC SDUs, each AMD PDU needs to be identified, so a unique sequence number is assigned to each AMD PDU as a part of its header information. In order to request a STATUS report from the peer RLC entity, a polling bit is allocated in each AMD PDU's header.

Figure 3.2 shows the format of AMD PDUs. The length of the data part must be a multiple of 8 bits. The AMD PDU header consists of the first two bytes of an AMD PDU, including 1-bit D/C field, 12-bit Sequence Number (SN) field, 1-bit Polling bit (P) field, and 2-bit Header Extension Type (HE) field. The D/C field indicates the type of an AM

Figure 3.2: Format of AMD PDUs

PDU, and it is set to be '0' if the type is Control and '1' if the type is Data. SN field has a unique serial number for each AMD PDU used for retransmission and reassembly. the polling bit is used to request a STATUS report from the peer RLC entity. It is set to be '1' if a STATUS report is requested, and '0' otherwise. HE field indicates if the next octet is data or a "Length Indicator" and E bit field. "Length Indicator" and E bit fields are optional, and they are only used for separating the RLC SDUs, Padding, and Piggybacked STATUS PDU. The E bit represents Extension bit used to indicate if the next octet is another "Length Indicator" and E bit field. The next part is Data field containing part of an RLC SDU or one or more RLC SDUs, and the size of Data field can be variable but must be a multiple of 8 bits. The last part is Padding or Piggybacked STATUS PDU. Padding could be used

to make an AMD PDU have a valid length and the RLC receiver will disregard this part. Instead of PAD, a piggybacked STATUS PDU can be placed in this part. It can carry the protocol control information as does a STATUS PDU, and have a variable size to fit available room in the AMD PDU.

3.3.2 STATUS PDUs

STATUS PDUs provide protocol control information exchanged between the RLC peers. In a RLC connection, since retransmissions of lost RLC PDUs need to be performed, STATUS PDUs have to be able to transmit a list of sequence numbers in which the corresponding RLC PDUs are lost due to transmission errors. When AMD PDUs are received and re-assembled successfully by a RLC entity, its peer RLC entity will be notified to move the transmission window forward and delete the previously sent RLC PDUs that are buffered in the retransmission buffer and correctly received. So STATUS PDUs have to transmit such information to acknowledge successful receptions of RLC SDUs. In the case there the number of retransmissions of a RLC PDU reaches the maximum times of retransmission allowed, the recovery of the RLC PDU has to cease, and the peer RLC entity will be notified of the abortion. So STATUS PDUs can transmit the abortion information to advance the reception window of the peer RLC entity.

Figure 3.3 gives the format of STATUS PDUs. '0' is set in the D/C field since it is of Control type. PDU Type field has 3 bits, which indicates the Control PDU type, such as STATUS, RESET, or RESET ACK. SUFI represents Super-Field, which has a variable length according to its type and contents. Figure 3.4 shows the structure of a super-field. The size of its sub-fields can be zero except the "type" sub-field, which is 4 bits long. The

Figure 3.3: Format of STATUS PDUs

Figure 3.4: Structure of a Super-Field

most useful types of super-fields are No More Data super-field (NO_MORE), Acknowledgement super-field (ACK), List super-field (LIST), Bitmap super-field (BITMAP), Move Receiving Window super-field (MRW), and Move Receiving Window Acknowledgement super-field (MRW_ACK). Figure 3.5 gives the bit values for each type of SUFI to be set in the “type” sub-field. NO_MORE SUFI (Figure 3.6) indicates the end of the data part of a STATUS PDU, and must be always placed in a STATUS PDU as the last SUFI if it is included in that STATUS PDU. ACK SUFI (Figure 3.7) indicates the end of the data

Bit	Description
0000	No More Data (NO_MORE)
0001	Window Size (WINDOW)
0010	Acknowledgement (ACK)
0011	List (LIST)
0100	Bitmap (BITMAP)
0101	Relative list (Rlist)
0110	Move Receiving Window (MRW)
0111	Move Receiving Window Acknowledgement (MRW ACK)

Figure 3.5: Bit Value for Each Type of Super-Field

Type=NO_MORE

Figure 3.6: Super-Field: NO_MORE

Type = ACK
LSN

Figure 3.7: Super-Field: ACK

part of a STATUS PDU, and consists of a type identifier (ACK) and LSN, which is the sequence number whose previous AMD PDUs were all received correctly. It is used to notify the peer RLC entity to move the transmission window and delete the previously sent RLC PDUs that are buffered in the retransmission buffer and correctly received. LIST SUFI (Figure 3.8) and BITMAP SUFI (Figure 3.9) are used to request retransmission of AMD PDUs with transmission errors. MRW SUFI (Figure 3.10) is used to request the peer

Type = LIST
LENGTH
SN ₁
L ₁
SN ₂
L ₂
...
SN _{LENGTH}
L _{LENGTH}

Figure 3.8: SUpEr-Field: LIST

Type = BITMAP
LENGTH
FSN
Bitmap

Figure 3.9: SUpEr-Field: BITMAP

Type = MRW
LENGTH
SN_MRW ₁
SN_MRW ₂
...
SN_MRW _{LENGTH}
N _{LENGTH}

Figure 3.10: SUpEr-Field: MRW

Type = MRW ACK
N
SN ACK

Figure 3.11: SUpEr-Field: MRW_ACK

RLC entity to move its reception window and optionally to indicate the set of discarded RLC SDUs. MRW_ACK SUFI (Figure 3.11) is the acknowledgement of an MRW SUFI. Detailed information about how the values of the sub-fields are set is described in [rlc02].

3.4 Protocol Parameters and Buffers

There are nine protocol parameters in RLC, seven of which are used in the transmitting side. They are *MAXDAT*, *POLL_PDU*, *POLL_SDU*, *POLL_WINDOW*, *CONFIGURED_TX_WINDOW_SIZE*, *MAXMRW* and *MAXRST*. Another one, *CONFIGURED_RX_WINDOW_SIZE*, is used in the receiving side of the RLC protocol. The last one, *TTI*, is used in the both sides. The values of the parameters shall be configured by upper layers at the time of a session creation and not be changed during the session. There are three buffers maintained in the RLC protocol. Two of them, which are the transmission buffer and the retransmission buffer, are used to transmit and retransmit RLC PDUs in the transmitting side. The retransmission buffer stores all RLC PDUs transmitted to the peer RLC entity and should be update each time at which a STATUS PDU including an ACK SUFI is received. When one RLC PDU is requested for retransmission, its copy copied from the retransmission buffer is placed in the transmission buffer. As a consequence, the transmission buffer stores not only the new RLC PDUs to be transmitted for the first time, but also the RLC PDUs requested for retransmission. However, it is a priority queue buffer, in which the RLC PDUs requested for retransmission have a higher transmission priority than the RLC PDUs to be transmitted for the first time. The third buffer maintained in RLC is the reception buffer, which stores correctly received RLC PDUs in the receiving side. All the

buffers are empty at the beginning of a connection. The following is a detailed description of each parameter.

1. **TTI** is the time interval between two consecutive transmissions or receptions of AM PDUs. The value of *TTI* shall be set to be a multiple of 10 ms, and all the timers in RLC are in *TTI* units.
2. **MAXDAT** is the maximum number of transmissions of an AMD PDU. It is the upper limit for the state variable $VT(DAT)$ (to be described in Section 3.5). When $VT(DAT)$ equals to the value *MAXDAT*, the SDU discard procedure is initiated.
3. **POLL_PDU** is the upper limit for the state variable $VT(PDU)$ (to be described in Section 3.5), indicating how often the sender should poll the receiver if *POLL_EVERY_PDU_PDU* is configured. When $VT(PDU)$ equals to the value of *POLL_PDU*, a polling bit shall be set in an AMD PDU's header.
4. **POLL_SDU** is the upper limit for the state variable $VT(SDU)$ (to be described in Section 3.5), indicating how often the sender should poll the receiver if *POLL_EVERY_SDU_SDU* is configured. When $VT(SDU)$ equals to the value of *POLL_SDU*, a polling bit shall be set in an AMD PDU's header.
5. **POLL_WINDOW** indicates when the sender shall poll the receiver when *WINDOW_BASED_POLLING* is configured. A poll is triggered for each AMD PDU when the status variable *J* (to be described in Section 3.5) is greater than, or equal to, the value of *POLL_WINDOW*.
6. **MAXRST** is the maximum number of transmissions of a RESET PDU.

7. **CONFIGURED_TX_WINDOW_SIZE** is both the maximum allowed transmission window size and the value for the state variable $VT(WS)$ (to be described in Section 3.5).
8. **CONFIGURED_RX_WINDOW_SIZE** is the reception window size.
9. **MAXMRW** is the maximum times of transmission of an MRW SUFI. It is the upper limit for the state variable $VT(MRW)$ (to be described in Section 3.5). When $VT(MRW)$ equals to the value of **MAXMRW**, the RLC RESET procedure is invoked.

3.5 State Variables and Operations in the Transmitting Side

On the transmitting side, RLC expects bit strings from its upper layers, called RLC SDUs, which may be segmented and/or concatenated into AMD PDUs with a fixed length in RLC. The segmentation happens if the size of a RLC SDU is larger than the length of available space in the AMD PDU. Then, for each AMD PDU to be transmitted for the first time, one copy is placed in the retransmission buffer, and another copy is placed in the transmission buffer. For the RLC PDUs requested for retransmission, their copies from the retransmission buffer are placed in the transmission buffer with the transmission priority higher than the PDUs to be transmitted for the first time. Fields of the AMD PDUs in the transmission buffer will be set. For instance, the polling bit in the header of a AMD PDU may be set based on the values of RLC control parameters. The padding bits may be potentially replaced by piggybacked status information, and ciphering is applied if it is configured. In

the last step, the transmitting side submits the AMD PDUs in the transmission buffer to the lower layer through the logical channel. The PDU processing procedure is also illustrated in the transmitting side of Figure 3.1.

3.5.1 State Variables in the Transmitting Side

There are eight state variables used to keep AM PDUs transmission and retransmission information. As illustrated in Figure 3.12, to maintain the transmission window, the transmission window size $VT(WS)$, the lower bound $VT(A)$ and the upper bound $VT(MS)$ of the transmission window are introduced. To keep the information of the next PDU to be transmitted for the first time, and of how much the transmission window is occupied by RLC PDUs, the variable $VT(S)$ needs to be defined. The variable $VT(DAT)$ is introduced to count the times of transmissions of a particular PDU for the invocation of the SDU discard mechanism (to be discussed in Section 3.5.3). The variable $VT(MRW)$ is introduced to count the number of times a particular MRW SUFI is transmitted for the invocation of the connection RESET procedure. $VT(PDU)$ is used by the polling mechanism to count the number of sent AMD PDUs after the latest polling bit setup, and $VT(SDU)$ is used by the polling mechanism to count the number of sent RLC SDUs after the latest polling bit setup.

1. **VT(WS)** is the transmission window size that is configured at the beginning of a session and can be changed during the session by receiving a STATUS PDU including a WINDOW SUFI.
2. **VT(A)** is the sequence number following the sequence number of the last in-sequence

Figure 3.12: Variables of the Transmission and Reception Window

acknowledged AMD PDU's. It is the lower bound of the transmission window.

3. **VT(MS)** contains the sequence number of the first AMD PDU that can be rejected by the peer RLC entity because of the limited transmission window size. It is the upper bound of the transmission window, and is equal to $VT(A) + VT(WS)$, hence should be updated when either $VT(A)$, or $VT(WS)$, or both are updated.
4. **VT(S)** contains the sequence number of the next AMD PDU to be transmitted for the first time.
5. **VT(DAT)** counts the number of times with which an AMD PDU has been scheduled for transmission, and each AMD PDU is correspondent to a separate $VT(DAT)$. The initial value of $VT(DAT)$ is 0.
6. **VT(PDU)** is used when *POLL EVERY POLL PDU PDU* polling trigger is config-

ured. It is incremented by 1 after the transmission of each AMD PDU including both new and retransmitted AMD PDUs, and when it is equal to the value of *POLL_PDU*, a new polling bit will be set for requesting a STATUS report. The initial value of *VT(PDU)* is 0.

7. **VT(SDU)** is used when *POLL_EVERY_POLL_SDU_SDU* polling trigger is configured. It is incremented by 1 each time at which all AMD PDUs associated with a given RLC SDU have been transmitted at least once. When it is equal to the value *POLL_SDU*, a new polling bit will be set for a STATUS report request. The initial value of *VT(SDU)* is 0.
8. **VT(MRW)** counts the number of times with which an MRW SUFI is scheduled to be transmitted.

3.5.2 The Polling Mechanism

In case of erroneous or lost AM PDUs, retransmissions are conducted by the sender upon reception of one or more STATUS reports from the receiver. Sending a STATUS report can be triggered by either the sender or the receiver. For the polling mechanism, the sender sends the polling request and the receiver responds by sending a STATUS report back to the sender. For the status transmission mechanism (to be described in Section 3.6), sending a STATUS report is triggered by the receiver.

The polling mechanism is implemented by setting the polling bit in one or more AMD PDUs' header. There are eight triggers kept in the transmitting side and used to set the polling bit when the polling mechanism is initiated. They include: (1) *Poll Timer*, (2)

Poll Periodic Timer, (3) *Poll Prohibit Timer*, (4) *Last PDU in buffer*, (5) *Last PDU in retransmission buffer*, (6) *Poll PDU PDU*, (7) *Poll SDU SDU*, (8) *Window based polling*.

Which of the triggers shall be used is/are decided by upper layers.

The buffer-based triggers, *Last PDU in buffer* and *Last PDU in retransmission buffer* can prevent deadlock of the RLC entities [ZS02], so they are important triggers in the polling mechanism. *Poll Periodic Timer* polls the receiver periodically, and the timer value controls the polling frequency. It starts when a session is created. For each time at which the timer expires, a polling is set in the header of an AMD PDU to be transmitted if it is available in the transmission buffer, and the timer is reset. *Poll Timer* starts when a polling contained in an AMD PDU is submitted to the lower layer in the sender, and should be reset each time at which a polling is set and submitted to the lower layer. It thus can guarantee to poll the receiver at least periodically with the period being the *Poll Timer* value. When both *Poll Periodic Timer* and *Poll Timer* are set and the *Poll Periodic Timer* value is less than the *Poll Timer* value, the *Poll Timer* trigger is useless. *Poll Prohibit Timer* controls the frequency of polling the receiver by prohibiting transmissions of polls with a certain period. So when multiple polling bit settings are triggered before *Poll Prohibit Timer* expires, the settings are deferred till *Poll Prohibit Timer* expires. *Poll PDU PDU* will poll the receiver when the number of sent AMD PDUs after the last polling reaches a certain number such as 30 PDUs. *Poll SDU SDU* polls the receiver when the number of sent RLC SDUs after the last polling reaches a certain number such as 5 SDUs. The trigger *Window based polling* polls the receiver each time at which the transmission window occupation of AMD PDUs exceeds a certain percentage such as 70%.

The following lists the value setup principles for the eight triggers.

1. **Poll Timer** starts when a polling contained in an AMD PDU is submitted to the lower layer in the sender, and should be reset each time at which a polling is set and submitted to the lower layer.
2. **Poll Prohibit Timer** starts when a polling, contained in an AMD PDU, is submitted to the lower layer in the sender. From the time at which a polling is triggered until the time at which *Poll Prohibit Timer* expires, polling is prohibited. If one polling is triggered while polling is prohibited, its transmission shall be delayed until the timer expires. If more than one polling was triggered when *Poll Prohibit Timer* expires, only one polling should be transmitted.
3. **Poll Periodic Timer** starts when a session is created. When the timer expires, a polling shall be triggered if an AMD PDU is available for transmission or retransmission, and the timer is reset.
4. **Last PDU in buffer** is the transmission-buffer-based trigger. Each time at which the last AMD PDU to be transmitted for the first time in the transmission buffer is sent, the polling bit in the PDU's header is set.
5. **Last PDU in retransmission buffer** is the retransmission-buffer-based trigger. The polling bit is set if a retransmitted AMD PDU is the last AMD PDU stored in the retransmission buffer.
6. **Poll PDU PDU** triggers a polling bit setup each time at which the value of $VT(PDU)$ is equal to the value of $POLL_PDU$.
7. **Poll SDU SDU** triggers a polling bit setup each time at which the value of $VT(SDU)$

is equal to the value of *POLL_SDU*.

8. **Window based polling** is the transmission-window-based trigger and shall be used when *WINDOW_BASED_POLLING* is configured. There is a status variable *J* expressed in percentage terms maintained in RLC. It is calculated by dividing the number of PDUs sent within the current transmission window by the transmission window size. When value *J* exceeds a certain percentage threshold such as 70, called *POLL_WINDOW*, a polling bit is set.

3.5.3 The RLC SDU Discard Mechanism

The RLC SDU discard mechanism allows the sender to discard the AMD PDUs associated with a SDU from the transmission buffer and the retransmission buffer. It is initiated when the transmission of the AMD PDUs does not succeed for a period of time or for a number of transmissions. This mechanism can avoid buffer overflow in the RLC layer and reduce the maximum transmission delay. There are two RLC SDU discard functions that can be configured according to the QoS requirements: Timer based discard with explicit signalling, and RLC SDU discard after *MAXDAT* number of transmissions.

1. Timer based discard with explicit signalling

This function discards a RLC SDU after its corresponding timer, called *Timer Discard* expires. This makes the SDU discard function insensitive to variations of the channel data rate and error rate, thus controls the maximum delay each RLC SDU may experience. The SDU loss rate, however, increases as RLC SDUs are discarded. The function works as follows. For each RLC SDU received from upper layers, a timer

Timer_Discard shall be started. When *Timer_Discard* of a SDU expires, the sender will discard the SDU, and use explicit signalling to inform the receiver which SDU has been discarded.

2. RLC SDU discard after *MAXDAT* number of transmissions

An alternative RLC SDU discard function is “SDU discard after *MAXDAT* number of transmissions.” It tries to keep the SDU loss rate constant. However, its delay performance is variable and dependent on the channel condition. If $VT(DAT)$ of an AMD PDU reaches the value *MAXDAT*, the sender will discard all RLC SDUs contained in the AMD PDU, and use explicit signalling to inform the receiver the discarded AMD PDU by sending an MRW SUFI.

3.6 State Variables and Operations in the Receiving Side

The receiving side expects to receive AMD PDUs and Control PDUs through the configured logical channel from the lower layer. AMD PDUs are deciphered if ciphering is configured, and then delivered to the reception buffer. The AMD PDUs are placed in the reception buffer until a complete RLC SDU is delivered to upper layers. Then the receiver acknowledges the successful reception or requests retransmissions of the missing AMD PDUs by sending one or more STATUS PDUs to the peer RLC entity through its transmitting side. If a Piggybacked STATUS PDU is found in an AMD PDU, it is delivered to the retransmission buffer at the transmitting side to make copies of the PDUs which need to be retransmitted and put those copies into the transmission buffer. Then those AMD PDUs associated with a complete RLC SDU may be reassembled and delivered to upper layers.

The STATUS PDUs are delivered to the retransmission buffer for purging the buffer of positively acknowledged AMD PDUs, and finding which AMD PDUs need to be retransmitted. RESET and RESET ACK PDUs are delivered to the RLC Control Unit for processing. The PDU processing procedure is also illustrated in the receiving side of Figure 3.1.

3.6.1 State Variables in the Receiving Side

There are four state variables used to keep the information of received AM PDUs in the receiving side. They are $VR(R)$, $VR(H)$, $VR(MR)$, and $VR(EP)$.

As shown in Figure 3.12, to maintain the reception window, the lower bound $VR(R)$ and the upper bound $VR(MR)$ of the reception window are introduced. To count which part of the reception window is occupied by PDUs and which part is not, the variable $VR(H)$ is defined. The variable $VR(EP)$ is introduced to count the number of AMD PDUs, which require retransmissions as a consequence of the transmission of the latest STATUS report. It is used in the STATUS transmission mechanism (to be discussed in Section 3.6.2).

The value setup principle for each state variables is described as follows.

1. **VR(R)** contains the sequence number following that of the last in-order AMD PDU received. It is the lower bound of the reception window and shall be updated upon the receipt of the APD PDU with sequence number equal to $VR(R)$.
2. **VR(H)** contains the sequence number following the highest sequence number of any received AMD PDU. It is the largest sequence number of the next AMD PDU expected to be received.
3. **VR(MR)** is the maximum sequence number of the first AMD PDU that will be re-

jected by the receiver. It is equal to $VR(R) + CONFIGURED_RX_WINDOW_SIZE$.

So it is the upper bound of the reception window and should be updated when the value of $VR(R)$ is changed.

4. **VR(EP)** is called the Estimated PDU Counter, containing the number of AMD PDUs whose retransmission is still expected as a consequence of the transmission of the latest STATUS report. At the end of each TTI, it is decremented by the total number of retransmitted AMD PDUs that were recovered, and it is used in the STATUS transmission mechanism.

3.6.2 The STATUS Transmission Mechanism

Upon reception of a polling request, the receiver always sends a STATUS report back the sender as a response to the request. Another way of sending a STATUS report is driven by the receiver, instead of the sender. Such mechanism can allow the receiver to send STATUS reports more aggressively. The trigger *detection of missing PDU(s)* can make the receiver send a STATUS report to request for retransmission when one or more missing PDU is learned by the receiver. It is an effective trigger to shorten the RLC SDU delay. The trigger *STATUS Period Timer* asks the receiver to send STATUS reports back to the sender periodically. The EPC mechanism makes the receiver send STATUS reports more actively by estimating the time needed to recover the erroneous AMD PDUs included in the latest STATUS report. The timer *Status Prohibit Timer* controls the frequency of sending STATUS reports to the sender. All of the triggers of the STATUS transmission mechanism are set optionally in the RLC protocol, and the EPC mechanism and *Status Prohibit Timer*

can be configured at the same time.

1. The Detection of Missing AMD PDU(s)

Once the receiver detects one or several missing AMD PDUs, a STATUS report is sent to the sender, requesting retransmission of all erroneous AMD PDUs.

2. The Timer-based STATUS Transmission

The receiver sends STATUS reports to the sender periodically. The timer *STATUS Period Timer* controls the time period of sending STATUS reports. This timer shall be started when the RLC entities are created, and when it expires, the transmission of a STATUS report will be performed and the timer is reset.

3. The EPC Mechanism

The EPC mechanism prevents excessive exchanges of STATUS reports. *EPC Timer* is the period of scheduling the transmissions of STATUS reports in this mechanism, and it starts each time at which the first STATUS PDU of a STATUS report is submitted to the lower layer. Then, the state variable $VR(EP)$ is set to be the number of AMD PDUs to be recovered. If not all AMD PDUs requested for retransmission have been received when $VR(EP)$ equals to zero, a new STATUS report is sent by the receiver. However, if another transmission of STATUS report is triggered while $VR(EP)$ is not equal to zero, the STATUS report will be delayed until it is decremented to zero.

4. State Variable Status Prohibit Timer

The timer *Status Prohibit Timer* prohibits the receiver from sending consecutive STATUS reports. It starts each time at which the first STATUS PDU of a STATUS

report is submitted to the lower layer. The receiver is not allowed to transmit a STATUS report before the timer expires.

5. **Both the *EPC Mechanism* and *STATUS Prohibit Timer* Configured**

In case of both the *EPC mechanism* and the *STATUS Prohibit Timer* are configured, a transmission of a STATUS report must be delayed until the maximum of $VR(EP)$ and *Status Prohibit Timer* expires. If more than one transmission of a STATUS report is triggered before the maximum of $VR(EP)$ and *Status Prohibit Timer* expires, only one STATUS report will be transmitted when it expires.

3.7 TTI Operations

The TTI operations are invoked each time at which an old TTI expires and another one starts. The operations include sending STATUS PDUs and AMD PDUs to the lower layer, decrement all related timers, and check each timer if it expires. When sending AM PDUs to the lower layer, the Control PDUs have the highest priority, then the AMD PDUs requested for retransmission, and the AMD PDUs transmitted for the first time have the lowest transmission priority. Before sending AMD PDUs, either newly created or retransmitted, the polling mechanism needs to be initiated to check if a polling bit needs to be set. After it is sent out, all the corresponding state variables in the transmitting side need to be updated, such as $VT(S)$, $VT(DAT)$, $VT(PDU)$, $VT(SDU)$, etc.

In the procedure of decrementing timers, all corresponding timers need to be decreased by 1. These timers could be *EPC Timer* or $VR(EP)$, *STATUS Prohibit Timer*, *STATUS Period Timer*, *Poll Timer*, *Poll Periodic Timer*, *Poll Prohibit Timer*, *Discard Timer* for each

AMD PDU, and *MRW Timer*. After decrement of the timers, checking their expirations will be handled by the polling mechanism, the STATUS transmission mechanism, and the RLC SDU discard mechanism correspondingly. For instance, checking *EPC Timer* or *VR(EP)*, *Poll Timer*, *Poll Periodic Timer*, and *Poll Prohibit Timer* will be in the polling mechanism, check of *Status Prohibit Timer*, and *Status Periodic Timer* will be in the STATUS transmission mechanism, and check of *MRW Timer*, and *Discard Timer* for each AMD PDU is taken in the RLC SDU discard mechanism.

3.8 Summary

RLC can be quite complicated with a large number of parameters that need fine-tuning. This makes the protocol very flexible and adaptive for UMTS operators with many parameter settings to choose from in various radio environments. However, optimising these parameters leads to significant challenges because of complex interactions among them, and inappropriate settings could even cause RLC performance degradation or deadlocks [XXC⁺02]. For these reasons, the evaluation of the various features, options, and functions of the UMTS RLC protocol using simulation becomes critical.

Chapter 4

Simulation Support

This chapter discusses the simulation support for the research reported in this thesis on the RLC protocol performance evaluation. It is organised as follows. Section 4.1 discusses the simulation framework, and Section 4.2 and 4.3 the RLC process design used in the simulation.

4.1 Simulation Framework

The simulation framework mainly consists of three components: the Radio Network Controller (RNC), the User Equipment (UE), and the radio interface (Uu), as shown in Figure 4.1. RNC exchanges a bulk data with UE through Uu employing the W-CDMA digital communication technique. The protocol stacks configured on the both sides are the same, including the upper layer, the RLC layer, and the physical layer. The RLC layer in RNC and UE is configured with the RLC process model (to be discussed in Section 4.2), but with different parameter settings. The upper layer is responsible for generating data (RLC

SDUs) and sending them to the RLC layer in RNC, and for receiving data and destroying them in UE. The physical layer in both sides is responsible for sending RLC PDUs coming from the RLC layer without adding any additional physical layer information, and receiving RLC PDUs coming from the radio interface Uu and delivering them to the RLC layer.

Figure 4.1: Simulation Model

The simulation model was developed with OPNET Modeler 9.0 from OPNET Technologies Inc. [Inc02], which is one of the most widely used discrete event-driven network simulator in the areas of telecommunications and networking for system development. The goal of this simulation setup is to evaluate the RLC protocol performance under different parameters' settings, thus to observe the trend of its parameters' setting such as Poll Timer, when the best RLC performance is reached. The interesting performance metrics in this research are the SDU average delay of in-sequence delivery, the RLC throughput,

and the RLC goodput. The definitions of these performance metrics will be discussed in Section 5.2.

4.2 RLC Functions Implemented

The RLC process implemented in our simulation is illustrated in Figure 4.2. Compared with the functions of the RLC protocol specified in [rlc02], SDU concatenation, padding, and ciphering were not implemented since they are not related to the performance evaluation of the RLC protocol. The function of SDU segmentation/concatenation is simplified (not in the simulation validations to be discussed in Section 5.1), and all other functions concerning to the RLC performance have been fully implemented in our RLC process model. The basic RLC functions implemented are described as follows.

1. Segmentation and/or concatenation and reassembly

Higher layer PDUs or RLC SDUs are segmented into RLC PDUs according to the pre-configured RLC PDU size. At the receiver, the RLC PDUs containing fragments of a RLC SDU are reassembled and the SDU is delivered to upper layers. Since our RLC process model is simplified based on the assumption that each RLC SDU corresponds to one, and only one, RLC AMD PDU in the performance evaluation of the RLC protocol (to be discussed in Section 5.3), the function of SDU segmentation/reassembly has been simplified for that purpose.

2. Transfer of user data

RLC transfers higher layer PDUs (RLC SDUs) to the lower layer. The QoS of user data can be realized by applying different parameter settings.

Figure 4.2: RLC Process Model of Simulation

3. Error correction

MR-ARQ [YPM00] is used for transmission errors' recovery in the event of erroneous or missing RLC PDUs. The receiver sends a STATUS report (one or more STATUS PDUs) including LIST SUFI(s) to the sender when it is polled or detects the missing AMD PDU(s) (optional). In the LIST SUFI(s), all sequence numbers of those AMD PDUs incorrectly received or missing are included. At the end of LIST SUFI(s), an ACK SUFI is attached. The ACK SUFI includes the sequence number

whose previous AMD PDUs are all received correctly, which is the value $VR(R)$.

4. In-sequence delivery of RLC SDUs

This function can preserve the order of RLC SDUs, which are submitted for transmission by RLC from the sender to the receiver.

5. Duplicate detection

This function detects duplicate RLC PDUs received correctly and ensures the re-assembled SDUs is delivered to upper layers only once.

6. Sliding-window flow control

This function allows the receiver to control the rate at which the sender may send RLC PDUs. The reception window size and the transmission window size are negotiated between the receiver and the sender at the time of a session creation, and are fixed during the session.

7. RLC SDU discard

Details of this function have been discussed in the RLC SDU discard mechanism.

4.3 Actions on the Process State Transitions

Figure 4.3 describes the State Transition Diagram (STD) used to implement RLC process developed with OPNET simulator. For the RLC process, there are three types of events that can happen. The first type of events is the RLC SDU(s) coming from upper layers, the second type of events is the RLC PDU(s) received from the lower layer, and the third type of events is the expiration of one Transmission Time Interval (TTI). Base on the three types of

events that the RLC process deals with, five states in the RLC state machine are drawn. The initial state is the “INIT” state, in which all of the pre-configured parameters are initiated, the timers, such as *Timer_Status_Periodic*, *Timer_Poll_Periodic* and *Timer_Poll_Prohibit*, are started if they are configured, and an event of TTI expiration is scheduled in one TTI time. Then it transits to the “IDLE” state unconditionally. When some event happens, it will transit to one of the other three states according to the type of the event. The event of RLC SDU(s) coming from upper layers can make the RLC process transit from the “IDLE” state into the “PRO_UPPER” state. The event of RLC PDU(s) received from the lower layer will make the RLC process transit from the “IDLE” state into the “PRO_LOWER” state. The event of the expiration of one TTI makes the RLC process transit from the “IDLE” state into the “TTI_EXPIRE” state. For each state transition, the corresponding actions are executed. The actions of the state transition from “IDLE” to “PRO_UPPER” will be discussed in Section 4.3.1, the actions of the state transition from “IDLE” to “PRO_LOWER” are to be discussed in Section 4.3.2, and the actions of the state transition from “IDLE” to “TTI_EXPIRE” will be described in Section 4.3.3. After the actions in a state transition are executed, the process will return back to the “IDLE” state unconditionally.

All of the RLC basic functions, including transfer of user data, error correction, in-sequence delivery of RLC SDUs, duplicate detection, flow control, and RLC SDU discard, have been implemented in the actions of the state transition from “IDLE” to “TTI_EXPIRE” (to be discussed in Section 4.3.3). Compared with the TTI operations described in Section 3.7, a few functions belonging to the transmitting and receiving side are moved into the actions of the state transition from “IDLE” to “TTI_EXPIRE”, which makes implementation of the RLC protocol easier.

Figure 4.3: State Transition Diagram of RLC Process

4.3.1 Actions on Transition of “IDLE” → “PRO_UPPER”

The action on the reception of upper layers’ SDU(s) is to place the received RLC SDU(s) in a temporary buffer, *t_sdu_buf*.

4.3.2 Actions on Transition of “IDLE” → “PRO_LOWER”

When receiving a RLC PDU from the lower layer, the PDU type, either Data or STATUS, needs to be learned by accessing the “D/C” field in the header of the RLC PDU. If the PDU is a STATUS PDU, it is put in a temporary buffer, *r_report_buf*, otherwise it is put in a different temporary buffer, *r_data_buf*.

4.3.3 Actions on Transition of “IDLE” → “TTI_EXPIRE”

The actions of the state transition from “IDLE” to “TTI_EXPIRE” is summarized in Figure 4.4 in pseudocode. All basic functions of RLC are implemented in the transition actions. These actions include: checking *MRW Timer* for invoking retransmission of a STATUS PDU including an MRW SUFI; checking the *VT(DAT)* for each AMD PDU in the retransmission buffer for initiation of the RLC SDU discard function; addressing the AMD PDUs and STATUS PDUs received from the lower layer during the last TTI time; generating ACK and LIST SUFIs; dealing with RLC SDUs received from upper layers during the last TTI time; sending RLC PDUs to the lower layer; decrementing configured timers; and scheduling next event of TTI expiration. The procedure of addressing the AMD PDUs received during the last TTI is to be discussed in Section 4.3.3.1, in which the functions of in-sequence delivery of RLC SDUs and duplicate detection are implemented. The error correction function has been implemented in the procedure of addressing the STATUS PDUs (to be discussed in Section 4.3.3.2). In the procedure of generating ACK and LIST SUFIs (to be described in Section 4.3.3.3), LIST SUFIs and ACK SUFIs are produced for implementing the function of error correction. The function of segmentation and concatenation will be implemented in the procedure of dealing with RLC SDUs (to be described in Section 4.3.3.4), in which RLC SDUs are transformed to RLC PDUs by attaching a RLC AMD PDU header for each RLC SDU and the function of segmentation/reassembly is skipped (not in the simulation validations to be discussed in Section 5.1) in the performance evaluation of the RLC protocol since we assume that one RLC SDU corresponds to one, and only one, RLC PDU. This function implementation will be included in our

future studies. The function of sliding window flow control has been implemented in the procedure of sending RLC PDUs (to be discussed in Section 4.3.3.5). The procedure of decrementing configured timers has been discussed in details in Section 3.7.

```

if (one MRW SUFI sent and MRW_ACK SUFI not received yet) do
  if (Timer_MRW expires) do
    if ( $VT(MRW)$  is equal to the value  $MAXMRW$ ) do
      call the RESET procedure;
    if ( $VT(MRW)$  not equal to the value  $MAXMRW$ ) do
      increase  $VT(MRW)$  by 1;
      reset the timer Timer_MRW;
      retransmit the MRW SUFI;
  check the State Variable  $VT(DAT)$  for each RLC PDU in the retransmission buffer for
  initiation of RLC SDU Discard Mechanism if there is/are AMD PDU(s) whose value
  of  $VT(DAT)$  is equal to the value  $MAXDAT$ .
  call the procedure receive_data_pdu();
  call the procedure receive_status_pdu();
  call the procedure generate_ack_listSufi();
  call the procedure produce_new_data_pdu();
  call the procedure send_pdu();
  call the procedure decrementTimers();
  schedule the next event of TTI expiration after TTI time;

```

Figure 4.4: Actions on the Event of TTI Expiration

4.3.3.1 The Procedure *receive_data_pdu*()

The procedure *receive_data_pdu*() is a function of the receiving side. It addresses the RLC AMD PDUs that are received and stored in the buffer *r_data_buf* during the last TTI time. For each AMD PDU, its sequence number needs to be accessed at first. Whether it is a PDU requested for retransmission or transmitted for the first time will be identified by its sequence number. Then it will either be destroyed or placed in another buffer, *r_pdu_buf*, and all corresponding State Variables in the receiving side, such as $VR(R)$, $VR(H)$, $VR(MR)$, need to be updated. Figure 4.5 illustrates the procedure in pseudocode.


```

// assume STATUS PDUs always stand alone rather than piggybacked
for (each AMD PDU  $x$  in  $r\_data\_buf$ ) do
  remove  $x$  from the buffer  $r\_data\_buf$ ;
   $seq :=$  sequence number of PDU  $x$ ;
  if (Polling bit of  $x$  is set) then
    set variable  $polled$  to be TRUE;
  if ( $seq$  is less than  $VR(R)$  OR  $seq$  greater than  $VR(MR)$  OR
    there is one copy with sequence number being  $seq$  in  $r\_pdu\_buf$ ) then
    destroy the received PDU  $x$ ;
  else
    insert the PDU  $x$  into the buffer  $r\_pdu\_buf$ ;
    if ( $seq == VR(R)$ ) then
      if ( $num\_pending\_retr > 0$ , meaning number of PDUs requested for retrans-
        mission since last status report transmission is greater than 0) then
        decrease  $num\_pending\_retr$  by 1;
       $VR(R) :=$  the next smallest sequence number of missing PDUs in  $r\_pdu\_buf$ ;
       $VR(MR) := VR(R) + CONFIGURED\_RX\_WINDOW\_SIZE$ ;
      if ( $VR(R) > VR(H)$ ) then
         $VR(H) := VR(R)$ ;
      else if ( $seq == VR(H)$ ) then
        increase  $VR(H)$  by 1;
      else if ( $seq > VR(H)$ ) then
         $pdu\_missing := 1$ ;
         $VR(H) := seq + 1$ ;
      else //  $seq > vr\_r \ \&\& \ seq < vr\_h$ 
        if ( $num\_pending\_retr > 0$ ) then
           $num\_pending\_retr --$ ;
    end if
  end for loop
  remove headers of all PDUs with the sequence numbers less than  $VR(R)$ , reassemble them into
  SDU(s) if possible and transmit them to upper layer;

```

Figure 4.5: Function: receive_data_pdu()

4.3.3.2 The Procedure receive_status_pdu()

The procedure *receive_status_pdu()* is another function in the receiving side. It deals with RLC STATUS PDUs that are received and stored in the buffer *r_status_buf* during the last TTI time. For each received STATUS PDU, there are five types of SUFIs that may be included in the STATUS PDU that is implemented in our simulation. The five types of SUFIs can be ACK, WINDOW, MRW, MRW_ACK, and LIST SUFI. For each type

of SUFI accessed, different actions will be taken. If a non-empty ACK SUFI is accessed, LSN sub-field in that SUFI (discussed in Section 3.3.2) is accessed, then the retransmission buffer is purged, and the transmission window state variables, such as $VT(A)$ and $VT(MS)$, are updated and the window moves forward. If a non-empty WINDOW SUFI is accessed, the transmission window state variables, such as $VT(WS)$ and $VT(MS)$, are updated. If a non-empty MRW SUFI is accessed, the reception buffer and the reception window are all updated according to the sequence number of the PDU to be discarded. If a non-empty MRW_ACK SUFI is accessed, the timer for retransmission of that MRW SUFI is cancelled, and, if any, another MRW SUFI will be sent out. If a non-empty LIST SUFI is accessed, a copy of the RLC PDU will be made. This will correspond to each sequence number included in that LIST SUFI from the retransmission buffer. It will be put in the transmission buffer, and the corresponding state variable $VT(DAT)$ will be updated. The procedure is illustrated in pseudocode in Figure 4.6.

```

// assume all STATUS PDUs always stand alone rather than piggybacked
for (each STATUS PDU  $x$  in buffer  $r\_report\_buf$ ) do
  remove  $x$  from buffer  $r\_report\_buf$ ;
   $ack\_sufi :=$  stripped ACK SUFI of  $x$ ;
   $window\_sufi :=$  stripped WINDOW SUFI of  $x$ ;
   $mrw\_sufi :=$  stripped MRW SUFI of  $x$ ;
   $mrw\_ack\_sufi :=$  stripped MRW_ACK SUFI of  $x$ ;
   $list\_sufi :=$  stripped LIST SUFI of  $x$ ;

  // address the ACK SUFI
  if ( $ack\_sufi$  is not empty) then
     $newVT(A) :=$  LSN sub-field of  $ack\_sufi$ ;
    if ( $newVT(A) > VT(A)$ ) then
      inform the upper layer those PDUs received by the peer;
      update  $VT(A)$  and  $VT(MS)$ ;
      check if an MRW SUFI has been sent and has been acknowledged;
      delete PDUs up to  $(VT(A) - 1)$  from the transmission buffer;
      delete PDUs up to  $(VT(A) - 1)$  from the retransmission buffer;
      delete  $VT(DAT)$  and  $Timer\_Discard$  correspondent to the deleted PDUs;
      destroy  $ack\_sufi$ ;
    end if for addressing ACK SUFI

```

```

// address the WINDOW SUFI
if (window_sufi is not empty) then
     $VT(WS) := \min(\text{value of } ws \text{ subfield of } window\_sufi, CONFIGURED\_TX\_WINDOW\_SIZE);$ 
     $VT(MS) := VT(A) + VT(ws);$ 
    destroy window_sufi;
end if for addressing WINDOW SUFI

// address the MRW SUFI
if (mrw_sufi is not empty) then
    seq := the largest sequence number of the PDUs to be deleted;
    delete AMD PDUs up to seq from r_data_buf;
    delete AMD PDUs up to seq from r_pdu_buf;
    delete all SDUs in the buffer r_sdu_buf;
    inform the upper layer which SDUs are discarded;
    update  $VR(R)$ ,  $VR(MR)$ , and  $VR(H)$  if necessary;
    create an MRW_ACK SUFI and put it in buffer t_sufi_buf for transmission to
    the lower layer;
    destroy mrw_sufi;
end if for addressing MRW SUFI

// address the MRW_ACK SUFI
if (mrw_ack_sufi is not empty) then
    set mrw_on_going being FALSE;
    update  $VT(A)$ ,  $VT(MS)$ ;
    remove the MRW SUFI just acknowledged from buffer t_mrw_sufi;
    check if there are pending MRW SUFIs needed to send, and send one to the
    lower layer if any;
    destroy mrw_ack_sufi;
end if for addressing MRW_ACK SUFI

// address the LIST SUFI
if (list_sufi is not empty) then
    retransmit each AMD PDU requested, increase its  $VT(DAT)$  by 1;
    destroy list_sufi;
end if for addressing LIST SUFI
end for loop

```

Figure 4.6: Function: receive_status_pdu()

4.3.3.3 The Procedure *generate_ack_list_sufi()*

In the procedure *generate_ack_list_sufi()*, one or more LIST SUFI and an ACK SUFI are generated and placed in the buffer *t_sufi_buf* waiting for transmission. The procedure is

illustrated in Figure 4.7.

```

list_sufi_needed := 0;
if (polled == TRUE OR
    (MISSING_PDU_INDICATOR configured AND pdu_missing == TRUE) OR
    Timer_Status_Periodic expires OR
    (VR(EP) == 0 AND num_pending_retr > 0) OR
    (VR(EP) == 0 AND delayed_list_report == TRUE) OR
    (Timer_Status_Prohibit expires AND delayed_list_report == TRUE)) then
    list_sufi_needed := 1;
else
    stop and return;

// check timer_epc and prohibit
if ((EPC_MECHANISM configured AND VR(EP) > 0) OR
    (STATUS_PROHIBIT configured AND Timer_Status_Prohibit not expire)) then
    delayed_list_report := 1;
else
    delayed_list_report := 0;
    create a LIST SUFI, insert it into t_sufi_buf;
    create a ACK SUFI, insert it into t_sufi_buf;
    if (EPC_MECHANISM configured) then
        reset the value of VR(EP);
        reset the value of num_pending_retr;
    if (STATUS_PROHIBIT configured) then
        restart timer Timer_Status_Prohibit;
if (Timer_Status_Periodic expires) then
    restart timer Timer_Status_Periodic;

```

Figure 4.7: Function: generate_ack_list_sufi()

4.3.3.4 The Procedure *produce_new_data_pdu()*

The procedure *produce_new_data_pdu()* is a function in the transmitting side. It deals with RLC SDUs received and stored in the buffer *t_sdu_buf* during the last TTI. For each RLC SDU removed from the buffer *t_sdu_buf*, transform the RLC SDU into one AMD PDU by attaching an AMD PDU header. In the AMD PDU header, a unique sequence number and a PDU type are assigned, and the polling bit will be set at the time of transmission to the lower layer. Meanwhile, the State variables *VT(DAT)* and *Discard Timer* are assigned and

stored for that AMD PDU.

4.3.3.5 The Procedure *send_pdu()*

The procedure *send_pdu()* is another function in the transmitting side. It sends STATUS PDUs if there is/are SUFI(s) generated and stored in the buffer *t_sufi_buf* during the last TTI time, and AMD PDUs, including those either to be transmitted for the first time or requested for retransmission. Before sending each AMD PDU, the polling mechanism needs to be invoked and the polling bit may be set in that AMD PDU's header. After sending out that AMD PDU, its corresponding state variables in the transmitting side such as $VT(S)$ and $VT(PDU)$, and the related timers like *Poll Prohibit Timer* and *Poll Timer*, should be updated if the polling bit is set in that AMD PDU. Figure 4.8 shows the function in pseudocode.

```
// construct and send STATUS PDUs first
// assume all STATUS PDUs stand alone rather piggybacked
numPdusToSend := NUM_PDUS_PER_TTI;
while (t_sufi_buf is not empty AND NUM_PDUS_PER_TTI > 0) do
    remove SUFIs from t_sufi_buf, construct them into STATUS PDU and send to
    the lower layer;
    decrease numPdusToSend by 1;
end while loop

// schedule a dedicated POLLING PDU if necessary
no_data_pdu := FALSE;
polling_triggered_POLL_TIMER := FALSE;
polling_triggered_TIMER_BASED_POLLING := FALSE;
polling_prohibited := FALSE;
if (t_pdu_buf is empty OR the smallest sequence number in t_pdu_buf greater than
     $VT(MS)$ , the upper bound of transmission window size) then
    no_data_pdu := TRUE;
if (POLL_TIMER is configured AND Timer_Poll expires AND  $VT(A) <$ 
    poll_time_vt_ms, meaning there is still at least one AMD PDU not received
    correctly polled for retransmission at the last poll) then
    polling_triggered_POLL_TIMER := TRUE;
if (TIMER_BASED_POLLING conf. AND Timer_Poll_Periodic expires) then
    polling_triggered_TIMER_BASED_POLLING := TRUE;
if (POLL_PROHIBIT is configured AND Timer_Poll_Prohibit not expire) then
    polling_prohibited := TRUE;
```

```

if (no_data_pdu is TRUE AND (polling_triggered_POLL_TIMER is TRUE OR
    polling_triggered_TIMER_BASED_POLLING is TRUE) AND
    polling_prohibited is FALSE) then
    make a copy of last new transmitted AMD PDU in the retransmission buffer and
    insert it into t_pdu_buf, the transmission buffer;

// send each pdu to lower layer
for (each PDU x in t_pdu_buf) do
    seq := sequence number of x;
    if ((seq <= VT(MS) OR seq == VT(S) - 1) AND
        NUM_PDUS_PER_TTI > 0) then
        remove x from t_pdu_buf;
        decrease NUM_PDUS_PER_TTI by 1;
        check all seven polling triggers, if any is active, set polling_triggered TRUE;
        polling_sent := FALSE;
        set polling bit of x being 0;
        if (polling_triggered == TRUE OR polling_delayed == TRUE) then
            if (POLL_PROHIBIT conf. AND Timer_Poll_Prohibit not expire) then
                polling_delayed := TRUE;
            else
                polling_delayed := FALSE;
                set polling bit of x being 1;
                polling_sent := TRUE;
            end inner if
        send x to the lower layer;
        if (x is transmitted for the first time) then
            increase VT(S) by 1;
        if (polling_sent == TRUE) then
            reset timer Timer_Poll_Prohibit;
        if (polling_sent == TRUE AND POLL_TIMER configured) then
            resart Timer_Poll;
            poll_time_vt_s := VT(S);
        if (EVERY_POLL_PDU_PDU configured) then
            reset or increase State Variable VT(PDU) by 1;
            check POLL_EVERY_POLL_SDU_SDU and change value of VT(SDU);
        if (TIMER_BASED_POLLING conf. AND Timer_Poll_Periodic expires) then
            restart Timer_Poll_Periodic;
        end if
    end for loop

```

Figure 4.8: Function: send_pdu()

4.4 Summary

This chapter described our simulation model and the RLC process used in our simulation. The simulation framework consists of RNC, UE, and Uu. Both RNC and UE include

the RLC protocol process at layer 2 in their protocol stacks. The RLC process model is implemented for this research and its basic functions related to the protocol performance were described in this chapter.

Chapter 5

Simulation Validations and Experimental Evaluation

This chapter validates the simulation and presents the simulation results of this study. Section 5.1 validates the simulation by comparing analytical and simulation results. Section 5.2 defines the interesting performance metrics of this research. Section 5.3 describes the simulation parameters' settings to evaluate the RLC performance. The simulation results are presented and analysed in Section 5.4.

5.1 Simulation Validations

To justify our simulation model, we take further studies on some existing theoretical results and compare them with our simulation results.

5.1.1 Protocol Parameters Setting for Simulation Validation

In order to apply theoretical results to validate our simulation, the out-of-sequence delivery is configured and the STATUS reports are sent periodically in the simulation. The upper

layers' traffic inter-arrival time is set to be a relatively large constant value so that, when the system is in a stable state, all RLC SDUs coming in one TTI can be sent to the lower layer when that TTI expires. The Block Error Rate (BLER) is the probability for a RLC PDU to be incorrectly received. It is a Link-layer parameter of UMTS that shows the characteristics of wireless lossy links. In order to compare with analytical results, we assume that no transmission error happens on RLC STATUS PDUs, and the error rate of RLC AMD PDUs is configured before starting a simulation and cannot change during the simulation. To simplify the problem, no Piggybacked STATUS PDUs are used, and no connection RESET procedure is invoked during the simulation. Upper layers' packets are considered a fixed size, which corresponds to a fixed number of RLC AMD PDUs.

The parameters used in the validation are summarized in Table 5.1.

5.1.2 Metrics Used for Simulation Validation

Two metrics are used to validate our simulation: The SDU discard rate and the SDU average delay of out-of-sequence delivery.

1. **The SDU Discard Rate** is defined to be the percentage of the RLC SDUs incorrectly received after the RLC layer error recovery, when out-of-order delivery of RLC SDUs is configured. It is the division of the number of RLC SDUs un-recovered by RLC layer divided by the total number of RLC SDUs sent out from the sender, in percentage term. Now, we analyse the theoretical result for the SDU discard rate. We define the BLER of RLC PDU as p , the number of RLC PDUs per SDU is N , and the maximum times of transmission allowed for a RLC PDU is k_{max} . Then, by assuming that a RLC PDU being incorrectly received is independent of other RLC PDUs, the probability of a RLC PDU is recovered by RLC after k_{max} times of transmissions is

Parameters	Value Options	Value Chosen for Validation
RLC SDU size	Variable	1216 bytes (or 32 RLC PDUs)
SDU traffic model	Bulk data transfer or web browsing	Web browsing with a constant inter-arrival time 0.08 sec
RLC SDU delivery	In-sequence or out-of-sequence delivery	Out-of-sequence delivery
RLC PDU size	multiple of 8 bits	40 bytes
TTI	multiple of 10 ms	20 ms
Physical layer data rate	no less than 144 kbps	384 kbps (or 24 PDUs per TTI)
PDU one-way latency	variable	100 ms
SDU discard function	Timer-based discard or max. number of transmissions based discard	Max. number of transmissions based discard
Max. number of transmissions	Variable	3 - 7
SDU discard time	Variable	N/A
Block error rate	variable	0%, 10%, or 25%
Transmission window size	Variable	2047 PDUs
Receiving window size	Variable	2047 PDUs
Polling mechanism	7 polling triggers	Disable all polling triggers
STATUS report transfer	4 triggers	Sending periodically
Period of STATUS report sending	Variable	300 ms

Table 5.1: Simulation Parameters for Validation

$(1 - p^{k_{max}})$. The probability of a SDU that corresponds to N PDUs correctly received after RLC layer recovery is $(1 - p^{k_{max}})^N$. Obviously, the SDU discard rate is,

$$sdr = 1 - (1 - p^{k_{max}})^N \quad (5.1)$$

2. **SDU Average Delay of Out-of-sequence Delivery** has the same definition as in [EL02], which is the division of the summation of delays for the correctly received RLC SDUs, including correctly received for the first time transmission and by re-transmissions, divided by the number of correctly received RLC SDUs. Since it is still hard to compute this value using the theoretical result given by Nicolas Enderle

[EL02], we will use the lower bound and the upper bound, which lead to an approximation of the mean average delay. Then, we will use the three theoretical results (to be analysed in Section 5.1.3) to compare with our simulation results.

5.1.3 Analysis of the Average Delay

We use the same definitions of all delay parts as in [EL02], including T_{proc} , T_{lub} , TTI , and T_{ack} . T_{proc} is the processing delay of one RLC SDU in the RLC sender; T_{lub} is the latency on the Iub interface (RNC - Node B), which is independent of the transport block size; TTI is the Transmission Time Interval at RNC; T_{ack} is the time between the detection of a missing PDU by the receiver and the transmission of a STATUS PDU to the sender. Three variables are analysed: D_1 , the time for SDU to be transmitted once; T_1 , the time between the detection of an erroneous RLC PDU on the receiving side and the reception of its retransmission by the receiving side for one time of transmission; and D_k , the time between the detection of an erroneous RLC PDU on the receiving side and the reception of its retransmission by the receiving side for k times of transmission. These are all analysed in [EL02], and their results are the following:

$$D_1 = T_{proc} + T_{lub} + mxTTI \quad (5.2)$$

where,

$$m = \lceil N/N_l \rceil \quad (5.3)$$

where N is the number of RLC PDUs per RLC SDU, N_l is the number of RLC PDUs per TTI, and $\lceil \cdot \rceil$ rounds up to the nearest integer.

$$T_1 = T_{ack} + 2x(TTI + T_{lub}) \quad (5.4)$$

$$D_k = D_1 + (k - 1)xT_1 \quad (5.5)$$

where k is the number of transmissions experienced by a RLC PDU.

Based on the assumption that a RLC PDU being incorrectly received is independent of other RLC PDUs, the probability mass function of the discrete random number D is,

$$p(D = D_k) = (1 - p^k)^N - (1 - p^{k-1})^N, k = 1, 2, \dots, k_{max} \quad (5.6)$$

If it includes the delay of un-recovered RLC SDUs, then the revised mean delay of out-of-sequence delivery $\bar{D}$ is,

$$\bar{D} = \sum_{k=1}^{k_{max}} D_k x p(D = D_k) + [1 - (1 - p^{k_{max}})^N] x [D_1 + (k_{max} - 1)xT_1] \quad (5.7)$$

where the second term is the delays of those un-recovered RLC SDUs, which is missing from the original result given in [EL02].

Since the RLC protocol is required to recover most of the RLC SDUs with transmission errors (and otherwise the allowed maximum times of transmission should be increased), the last part in equation 5.7 can be neglected. Then the average delay becomes the same as the one given in [EL02], which is,

$$\bar{D} = \sum_{k=1}^{k_{max}} D_k x p(D = D_k) \quad (5.8)$$

Since T_{ack} is the time between the detection of a missing PDU by the receiver and the transmission of a STATUS PDU to the sender, it is not fixed. For a particular RLC PDU, the value of its T_{ack} could be at least one TTI in the case of the PDU reaching the receiver just at the TTI in which a STATUS PDU is sent to the sender, or at most the period of STATUS PDUs sending in the event of the RLC PDU arriving at the receiver just after the TTI in which a STATUS PDU is sent to the sender. Hence we have an upper bound T_{ackmax} and a

lower bound T_{ackmin} of T_{ack} . Also, we can have the mean value of T_{ack} , which is uniformly distributed between one TTI and the period of STATUS PDUs sending in the case of a RLC PDU incorrectly received. Then the average T_{ack} is $(1 + STATUS_PDU_PERIODIC)/2$ $TTIs$. We have,

$$T_{ackmax} = 1xTTI \quad (5.9)$$

$$T_{ackmin} = STATUS_PDU_PERIODICxTTI \quad (5.10)$$

$$T_{ackmean} = (1 + STATUS_PDU_PERIODIC)xTTI/2 \quad (5.11)$$

in which $STATUS_PDU_PERIODIC$ is the time between two consecutive sendings of STATUS reports.

Then we can get the lower bound, the upper bound, and the approximate results of the mean delay by taking equation 5.10, equation 5.9, and equation 5.11 into equation 5.8.

$$\overline{D_{min}} = \sum_{k=1}^{k_{max}} [T_{proc} + T_{lub} + mxTTI + (k-1)(3xTTI + 2xT_{lub})]xp(D = D_k) \quad (5.12)$$

$$\overline{D_{max}} = \sum_{k=1}^{k_{max}} \{T_{proc} + T_{lub} + mxTTI + (k-1)[STATUS_PDU_PERIODICxTTI + 2x(TTI + T_{lub})]\}xp(D = D_k) \quad (5.13)$$

$$\overline{D_{app}} = \sum_{k=1}^{k_{max}} \{T_{proc} + T_{lub} + (k-1)[(1 + STATUS_PDU_PERIODIC)xTTI/2 + 2x(TTI + T_{lub})]\}xp(D = D_k) \quad (5.14)$$

Now, we will use these theoretical results of the SDU average delay with out-of-sequence delivery to do comparisons with our simulation results.

5.1.4 Comparison of Theoretical and Simulation Results

5.1.4.1 Comparison of SDU Discard Rates

Figure 5.1 describes the SDU discard rate comparison from the analytical results and simulation results based on the BLERs being 0%, 10%, 25%, respectively, and different maximum times of retransmission from 2 to 6 times. The analytical values come from equation 5.1 and all other protocol parameters in this simulation conform to the specified values in Table 5.1.

Figure 5.1 shows that the simulation results agree with their corresponding theoretical results, from which the implementation of the RLC protocol based on the Multiple Rejects ARQ (MR-ARQ) [YPM00] scheme is justified. If it is required for the upper layer packet discard rate that is perceived by the TCP layer to be less than 2% [FL03, CB00], then by observing the SDU discard rate with BLER being 10% in Figure 5.1, the maximum times of retransmission for a RLC PDU should be no less than 3 when BLER does not exceed 10%. And by observing the SDU discard rate with BLER 25%, the maximum times of retransmission for a RLC PDU should be no less than 5 when BLER is between 10% and 25%. In other words, the State Variable *MAX_DAT* (meaning the allowable maximum number of transmissions for a RLC PDU) should be set to be no less than 4 when BLER is less than 10%, and no less than 6 when BLER is between 10% and 25%.

5.1.4.2 RLC SDU Average Delays Comparison

Figure 5.2 describes the average delays comparison from the calculated and simulated results when in-sequence delivery is disabled. The calculated lower bound delay results come

Figure 5.1: SDU Discard Rates Comparison

from equation 5.12, and the upper bound results from equation 5.13, and the approximate average delay results come from equation 5.14. For all the results, BLER is set to be 10%. The analytical results we used are based on the condition that RLC has recovered most of the transmission errors, because otherwise, the second term of the mean delay result in equation 5.7 cannot be neglected, and the comparison will be flawed. By the SDU discard rate comparison in Figure 5.1, the maximum number of retransmission of a RLC PDU should be at least 3 when block error rate is no greater than 10%. Hence we perform the SDU delay comparison starting the maximum number of retransmissions being at 2 times.

From Figure 5.2, it is observed that the curve of our simulated SDU average delay with

Figure 5.2: SDU Average Delay Comparison with Out-of-sequence Delivery

out-of-sequence delivery is in between of the curve of the lower bound delay and the curve of the upper bound, which again verifies our RLC process model implementation. Furthermore, it is much closer to the curve of our approximate SDU average delay values, from which it shows that our computable approximate analytical result on SDU average delay is a good approximation and could be used to guide the approximation of SDU average delay with in-sequence delivery.

5.2 Performance Metrics

In our study of the RLC performance evaluation, we focus on the three RLC performance metrics defined as follows.

1. **The RLC SDU delay of in-sequence delivery** is the time at which a RLC SDU reaches the RLC layer of the sender till the time at which it is correctly reassembled and delivered by the RLC of the receiver to the RLC upper layer. It includes three parts: The queueing delay at RNC, the out-of-sequence delivery delay (analysed in Section 5.1), and the re-sequencing delay at UE.
2. **The RLC throughput** is the total data bits transmitted on the forward link at a unit of time. It includes transmissions of data PDUs, either transmitted for the first time or for retransmissions, and STATUS PDUs such as the PDUs including MRW SUFIs.
3. **The RLC goodput** is the total useful information bits received correctly and delivered to the upper layer by the receiver. So neither duplicate AMD PDUs nor STATUS PDUs are counted for the RLC goodput.

5.3 Simulation Configuration for Performance Evaluation

We would like to obtain results using simulation to evaluate the RLC protocol performance under different parameters' settings, thus to observe the trend of its parameters' setup such as the Poll Timer and the STATUS Period Timer when the best RLC performance is reached. To reach the goal, the simulation and the RLC parameters are discussed as follows.

5.3.1 The Traffic Model of the Upper Layer

The traffic arrival process model and traffic statistics are the important factors for considering the RLC performance and parameters' settings [ZS02]. In this study, we use the traffic model for bulk data transfer (heavy FTP) to study the performance of the RLC protocol. The traffic coming from the upper layer of RNC is modeled as a determinate process with a constant mean packets. The speed of upper layer packets coming is set to be the same as the data rate of the wireless forward link.

5.3.2 The Transmission Errors of the Physical Layer

The physical layer of UMTS introduces the transmission errors with the Block Error Rate (BLER) depending on the link characteristics. BLER is the probability for a RLC PDU being incorrectly received. It is configured before starting a simulation and cannot be changed during the simulation. To simplify the problem, this study assumes that a RLC PDU being incorrectly received is independent of the status of other PDUs and the error distribution is uniform. More sophisticated and realistic transmission error models with bursty nature will be investigated in our future studies.

5.3.3 The Data Rate of the Wireless Channel

The IMT-2000 standard requires the data rate of 3G wireless networks is no less than 384 kbps capability for pedestrian (micro-cell) and low speed vehicular environment, and 2 Mbps for indoor office using wide-ban 1.6 MHz carrier [GG00]. In this study, we use 384 kbps as the data rate of the physical channel to simulate a micro-cell wireless environment.

The data rate of 2 Mbps for indoor office communications will be included in our future research work.

5.3.4 The Polling Triggers

To prevent system deadlock, two polling triggers, “Last PDU in buffer” and “Last PDU in retransmission buffer,” are set on when testing the effect of the Poll Timer [ZS02].

5.3.5 Assumptions of RLC Process

To simplify our simulation, four assumptions on the RLC protocol are made as follows:

1. Each AMD PDU corresponds to one, and only one, RLC SDU. In other words, no segmentation or concatenation of RLC SDUs happens in our simulation.
2. No transmission error happens on any STATUS PDU. That is, only RLC data PDUs can be lost because of transmission errors.
3. No Piggybacked STATUS PDUs are used. In other words, STATUS PDUs are always transmitted in stand-alone PDUs rather than piggybacked in AMD PDUs.
4. No RESET procedure is invoked during a simulation.

5.3.6 Protocol Parameters Setting for the Performance Evaluation

The parameters used in the RLC protocol performance evaluation are summarized in Table 5.2.

Parameters	Value Options	Value Chosen for Evaluation
RLC SDU size	Variable	38 bytes (or 1 RLC PDU)
SDU traffic model	Bulk data transfer or web browsing	Bulk data transfer with FTP of a constant traffic speed
RLC SDU delivery	In-sequence or out-of-sequence delivery	In-sequence delivery
RLC PDU size	multiple of 8 bits	40 bytes
TTI	multiple of 10 ms	20 ms
Physical layer data rate	no less than 144 kbps	384 kbps (or 24 PDUs per TTI)
PDU one-way latency	variable	100 ms
SDU discard function	Timer-based discard or max. number of transmissions based discard	Max. number of transmissions based discard
Max. number of transmissions	Variable	7 times
SDU discard time	Variable	N/A
Block error rate	variable	0%, 10%, 15%, or 25%
Transmission window size	Variable	2047 PDUs
Receiving window size	Variable	2047 PDUs
Polling triggers	7 polling triggers	Set Last_PDU_in_buffer, Last_PDU_in_retx_buffer on when testing Poll_Timer
Status report transfer	4 triggers	Test STATUS_Period_Timer trigger disable other 3 triggers
Period of status report sending	Variable	40ms - 1000ms

Table 5.2: Simulation Parameters for Performance Evaluation

5.4 Simulation Results and Discussion

We have done intensive simulations to evaluate the RLC protocol performance by varying the values of two timers: The Poll Timer and the STATUS Period Timer under different block error rates. In this section, we will present the effects of these two timers on the performance of the RLC protocol.

5.4.1 Effect of STATUS Period Timer

To evaluate the effect of the STATUS Period Timer, we put on this trigger and turn off all others, including “detecting missing AMD PDU(s),” “EPC mechanism” and “STATUS prohibit” triggers.

5.4.1.1 The Effect on the SDU Average Delay

Figure 5.3 shows the effects of the STATUS Period Timer on the SDU average delay under various block error rates. Usually, the in-sequence delivery delay a SDU experiences includes three parts: The queueing delay in the transmission buffer at the sending side, the out-of-sequence delivery delay as we defined and analysed in Section 5.1.3, and the re-sequencing delay in the reception buffer at the receiving side.

The values of the STATUS Period Timer affect the interval between the retransmissions of lost PDUs that can influence the SDU delay of out-of-sequence delivery, which is the second part of a SDU delay of in-sequence delivery as described above. When the BLER is zero, no PDUs are lost. All RLC PDUs are transmitted only once, so no effect of the STATUS Period Timer is observed. Since no retransmission is performed, and each RLC SDU can be delivered to upper layers as soon as its segmented PDUs are received completely, there is no re-sequencing delay of a RLC SDU at UE (the receiver). Meanwhile, the upper layers’ traffic has a constant rate with 24 SDUs per TTI, which can be transferred to the lower layer during the same TTI due to no RLC PDUs requested for retransmission. Then no queueing delay happens for any RLC PDU in the transmission buffer of RNC. Hence, SDU delay is the summation of the PDU one-way latency time and one TTI time (since one

Figure 5.3: The Effect on SDU Average Delay with Different BLERs

SDU can be delivered in one TTI). As shown in Figure 5.3, the simulated result of SDU delay with BLER 0% is 120 ms, which perfectly agree with our expected result analysed above.

When the block error rate is non-zero, the SDU average delay is not a fixed value. By observing the SDU average delay curves under different BLERs in Figure 5.3, there is a same trend regardless the block error rate values. At small STATUS Period Timer values such as 40 ms, RLC SDU delays are very large. As the STATUS Period Timer values

increases to a threshold, the RLC SDU delay decreases. The threshold values are 500 ms for BLER being 10%, 400 ms for BLER being 15% and 300 ms for 25% BLER. When the STATUS Period Timer is at these values mentioned above, SDU average delays reach the lowest for the three curves, respectively. After those values, SDU delays begin to increase. The reason for such a trend of SDU delay is discussed as follows.

A small STATUS Period Timer value makes the receiver send STATUS PDUs too frequently, which results in too many RLC PDUs being transmitted for the first time blocked at the transmission buffer of the sender. When the STATUS Period Timer value is so small that it is less than the round trip time, the STATUS Period Timer expires soon after a STATUS report is sent from the receiver. No associated RLC PDUs will arrive within the period of time since it is shorter than the round trip time. Then there must be one or more same PDUs on the fly when the receiver sends a STATUS report requesting retransmission of the same PDU as those being transmitted on the fly. If the PDU(s) on the fly is/are received correctly, the requested retransmission becomes unnecessary. However, the receiver is not given enough time to be aware if the last one or more sent STATUS reports can recover the erroneous PDU(s). Hence, more bandwidths of the wireless forward link will be used to unnecessarily transmit the PDUs requested for retransmission. Since the RLC PDUs requested for retransmission have a higher transmission priority than those to be transmitted for the first time, lots of new RLC PDUs are blocked at the transmission buffer of the sender, which results in a higher queueing delay RLC SDUs experience at the sender. As a consequence, the SDU average delays are extremely large when the STATUS Period Timer value is small. With the increase of that value, fewer STATUS PDUs are sent to the sender, and less bandwidths of the wireless forward link is used for unnecessary retransmissions.

Then the blocking of RLC PDUs to be transmitted for the first time in the transmission buffer at the sender is alleviated, and the SDU average delays decrease. When the period of sending STATUS reports is slightly greater than the round trip time, say 260 ms or 300 ms (the round trip of RLC PDU in our simulation is 240 ms), fewer unnecessary PDUs are requested for retransmission, and the timing of sending STATUS reports is optimal. Consequently, the SDU average delays are the smallest around those points. With BLER increasing, the STATUS Period Timer value at which SDU delay reaches the lowest is closer to the round trip time. The reason is, comparing with a relatively large STATUS Period Timer value, such as 400 ms, a smaller STATUS Period Timer value, such as 300 ms, will decrease the re-sequencing delay that a RLC SDU experiences in the receiver because of faster recovery of transmission errors, but causes slightly larger queueing delay in the sender due to more wireless bandwidths used to transmit RLC PDUs requested for retransmission. The two effects will compromise each other on the total SDU delay. For a larger block error rate like 25%, the effect of decreasing the re-sequencing delay is more obvious than the other effect, so the STATUS Period Timer value at which SDU average delay is the smallest is closer to the round trip time than the optimal timer value for a smaller block error rate like 10%.

On the contrary, a large STATUS Period Timer value asks the receiver to infrequently send STATUS PDUs to the sender, which results in taking more time for SDUs to be delivered to upper layers at the receiver. After the STATUS Period Timer value passes the optimal value of the STATUS Period Timer at which SDU delays are the smallest, the duration between two times of sending STATUS PDUs are longer, then a RLC SDU with transmission errors will wait for longer time to be recovered. Consequently, all of its fol-

lowing SDUs cannot be delivered to upper layers until it is recovered by retransmission or replaced by a failed recovery message. This results in a larger re-sequencing delay that a RLC SDU will experience at the receiver. Hence, the SDU average delay increases as the STATUS Period Timer value increases.

5.4.1.2 The Effect on the RLC Throughput

The effect of the STATUS Period Timer on the RLC throughput is illustrated in Figure 5.4. By the definition of the RLC throughput we made in Section 5.2, the maximum possible RLC throughput is $(Sx(1 - p))$, where S is the physical layer data rate and p is the block error rate of RLC PDUs. As we know, the RLC throughput can always reach its possible maximum value so long as the transmission window of the sender is not blocked.

When no transmission error happens on RLC PDUs, the possible maximum value of the RLC throughput is equal to the wireless link data rate. As we can observe from the curve of the RLC throughput with BLER 0%, for all STATUS Period Timer values, the RLC throughput is equal to 384 kbps, which is the defined physical layer data rate. As described in Section 3.3, a STATUS PDU sent periodically includes an ACK Super-Field that acknowledges the sequence number whose previous PDUs have been correctly received, and when the sender receives such a SUFI, the transmission window can move forward. Then, for the small values of the STATUS Period Timer, the transmission window is not blocked as a result of the higher frequent ACK SUFIs included in STATUS PDUs. When the timer value increases to the maximum tested value 1000 ms, which is equal to 50 TTIs (each Transmission Time Interval or TTI is 20 ms), the sender's transmission window can be occupied by a maximum of 1200 PDUs (during each TTI a maximum of 24 PDUs could

Figure 5.4: The Effect on RLC Throughput with Different BLERs

be sent), which need to be acknowledged. The transmission window size is set to be 2047 PDUs so that it is large enough to ensure no window blocking occurs at RNC (the sender) for the maximum tested value 1000 ms when BLER is 0%. Consequently, the curve of the RLC throughput with BLER 0% keeps a straight line at 384 kbps as observed in Figure 5.4.

However, for the other three curves of the RLC throughput with non-zero block error rates, there are differences. When the STATUS Period Timer values are small, there is

no chance of the transmission window being blocked, as described above. So the RLC throughput for each curve is much close to $(Sx(1 - p))$. As the timer value increases to a threshold, which has a different value with different BLERs, the RLC throughput starts decreasing. The larger the block error rate is, the smaller the threshold. The reason is, for a fixed STATUS Period Timer value, the transmission window with the bigger BLER is more likely to be blocked due to more RLC PDUs being retransmitted. As observed from Figure 5.4, the threshold value is bigger than the round trip time when BLER is smaller than 25%. After passing the threshold, the RLC throughput starts to decrease the STATUS Period Timer value increases. The reason for decrementing the RLC throughput is the sender's transmission window blocking. Blocking occurs because of a low speed of moving the transmission window due to a low frequency of receiving ACK SUFIs, included in STATUS PDUs sent from the receiving side. In the event of no STATUS PDU being received for a long time, less protocol control information is exchanged between the sender and the receiver. Then, the sender has fewer chances to update its transmission window because of lack of protocol control information. Therefore, the transmission window at the sender will be blocked eventually due to its limited window size. At such a situation, no data are transferred to the wireless link, and the wireless link bandwidths are not fully utilised. As a consequence, the RLC throughput decreases.

5.4.1.3 The Effect on the RLC Goodput

Figure 5.5 shows the effect of the STATUS Period Timer on the RLC goodput. The curve of the RLC goodput keeps a straight line when BLER is 0%, and for a non-zero block error rate, the RLC goodput has a peak value as observed in Figure 5.5.

Figure 5.5: The Effect on RLC Goodput with Different BLERs

As discussed in Section 5.4.1.2, there is no transmission window blocking or RLC PDUs retransmitted when BLER is 0%. Therefore, all bandwidths of the wireless link are used to transmit new RLC PDUs that are all delivered to upper layers. Each RLC PDU excluding the 2-bytes PDU header is the useful information for calculations of the RLC goodput. Then the RLC goodput value with BLER 0% is $384 \times 38 / 40 = 364.8$ kbps (a two-byte header in each RLC PDU excluded). As we observed, the straight line in Figure 5.5

is the simulated RLC goodput when BLER is 0%, which perfectly matches the analysed result.

Figure 5.5 shows that a peak value of the RLC goodput is reached with a non-zero BLER at a value of the STATUS Period Timer that is slightly larger than the round trip time of a RLC PDU. Before this value, the smaller the STATUS Period Timer value is, the smaller the RLC goodput. After this value, the larger the STATUS Period Timer value, the smaller the RLC goodput.

As we discussed in Section 5.4.1.1, even though for small values of the STATUS Period Timer, there is no chance for the transmission window of the sender to be blocked and the RLC throughput can reach its possible maximum value, a significant portion of the wireless link bandwidths are used to transmit unnecessary RLC PDUs requested for retransmission. Hence, a small STATUS Period Timer value may not produce any benefit to, or probably even degrade the RLC goodput because of an excess of unnecessary retransmissions of RLC PDUs. The smaller the STATUS Period Timer value is, the greater the degradation of the RLC goodput. However, if the STATUS Period Timer value is much larger than the round trip time, there will be more chances of the transmission window blocking due to fewer STATUS PDUs received by the sender, as described in Section 5.4.1.2. This causes fewer RLC PDUs to be transmitted for the first time sent by the sender, which results in less useful information received by the receiver. Consequently, the RLC goodput degrades. The larger the STATUS Period Timer value is, the higher the chance of blocking the transmission window, and the lower the RLC goodput.

5.4.2 Effect of Poll Timer

In order to put Poll Timer in effect, the Poll Timer trigger for polling STATUS reports needs to be enabled in the sender. Two other triggers in addition to this one, “Last PDU in buffer” and “Last PDU in retransmission buffer,” also need to be enabled in the sender to prevent protocol deadlock [ZS02]. All other triggers for polling STATUS reports, including *Poll Prohibit Timer*, *Poll Periodic Timer*, *Every Poll PDU*, *Every Poll SDU*, and *Window Based Polling*, are disabled. The timer starts when an AMD PDU containing a polling in the PDU header’s polling bit, which is triggered by other polling triggers such as the submission of “Last PDU in buffer” and “Last PDU in retransmission buffer,” is submitted to the lower layer in the sender. Since then, each time at which an AMD PDU containing a polling is submitted to the lower layer, the Poll Timer value is reset. The polling can be triggered by either the expiration of the Poll Timer, or “Last PDU in buffer” or “Last PDU in retransmission buffer” triggers in our simulation. Hence, STATUS reports are polled with a least frequency of the period time with the Poll Timer value. They may be polled at a time less than the period, which is invoked by the triggers other than the expiration of the Poll Timer.

5.4.2.1 The Effect on the SDU Average Delay

Figure 5.6 shows the effects of the Poll Timer on the SDU average delay. When the block error rate is zero, the SDU average delay keeps a same value. For non-zero block error rates, however, at a small Poll Timer value such as 40 ms, the SDU average delay is very large. As the Poll Timer values increase until 260 ms or 300 ms, the SDU average delays

decrease. When the Poll Timer is around such a value, the SDU average delay reaches the smallest. After that, the SDU average delay begins to increase.

Figure 5.6: The Effect on SDU Delay with Different BLERs

When block error rate is zero, no transmission error happens on any RLC PDU, and no retransmissions of PDUs are performed. The RLC PDUs to be transmitted in each TTI are all new PDUs generated in that TTI time, and there is one STATUS report received in each TTI due to the trigger, “Last PDU in buffer.” Then there is no queuing delay for RLC

PDU in the transmission buffer of RNC, or re-sequencing delay at UE. All delay RLC SDUs experience is the PDU one-way latency time (since one RLC PDU corresponds to one, and only one, RLC SDU) and one TTI time (since one SDU can be delivered in one TTI). As shown in Figure 5.6, the simulated result of the SDU average delay with BLER 0% is 120 ms, which agrees with the expected result analysed above.

Since the Poll Timer trigger functions very similar to the trigger STATUS Period Timer when the block error rate is non-zero, the SDU average delay is very large when the Poll Timer values are small (because of too many unnecessary retransmissions of RLC PDUs that blocks new PDUs to be transmitted for the first time). Then the SDU average delay experiences the lowest value when the Poll Timer value is slightly larger than the round trip time, say 260 ms or 300 ms (the round trip time in our simulation is 240 ms). The reason for the SDU average delay being the lowest at such values is timely transmission of STATUS reports and less unnecessary retransmissions of RLC PDUs. As the Poll Timer values increase, the SDU average delay starts to increase for a bigger block error rate such as 25%, or stay with a same value for a lower block error rate such as 10%. The reason is that a SDU may experience a larger re-sequencing delay at the receiver due to the longer duration between two times of sending STATUS reports. For a lower block error rate, other triggers, “Last PDU in buffer” and “Last PDU in retransmission buffer,” may be involved in polling bit setups more often than for a bigger block error rate. Hence, it is shown that SDU delay almost keeps the same value when the Poll Timer values are between 360 ms and 1000 ms when BLER is 10%. This shows that the importance of the Poll Timer decreases when its value increases to a larger one because of more often invocations of other triggers.

5.4.2.2 The Effect on the RLC Throughput

The effect of the Poll Timer on the RLC throughput is illustrated in Figure 5.7. As discussed in Section 5.4.1.2, the maximum possible RLC throughput is $(Sx(1 - p))$. As mentioned in Section 5.4.2.1, when the block error rate is zero, there is one STATUS report received for each TTI due to the trigger “Last PDU in buffer.” Then there is no chance of transmission window blocking, and the RLC throughput can always reach the maximum value regardless of different Poll Timer values. When the block error rate is a non-zero value, the RLC throughput for different Poll Timer values has a trend similar to that generated by different STATUS Period Timer values. The reason is that the Poll Timer can be considered as a trigger of polling STATUS reports periodically if it is the only trigger put on and has already started initially in the sender. Since there are two more triggers set in the simulation of testing the Poll Timer (the STATUS Period Timer is the only trigger put on in the simulation of testing the STATUS Period Timer), there is more chance of STATUS PDUs transmitted from the receiver to the sender for a same timer value for the Poll Timer and the STATUS Period Timer. Then the decrease of the RLC throughput for the Poll Timer values may be slower than that for the STATUS Period Timer values when they are between 360 ms and 1000 ms. This can be confirmed by comparing the curves of the RLC throughput in Figure 5.4 and Figure 5.7 with the same timer value and block error rate.

5.4.2.3 The Effect on the RLC Goodput

Figure 5.8 illustrates the effect of the Poll Timer on the RLC goodput. When the block error rate is zero, the RLC goodput reaches its maximum values for all Poll Timer values. The reason is that no transmission window blocking or RLC PDUs’ retransmissions

Figure 5.7: The Effect on RLC Throughput with Different BLERs

happen (discussed in Section 5.4.2.1), and all bandwidths of the wireless link are used to transmitted new RLC PDUs to be delivered to the upper layer. For the non-zero block error rates, the RLC goodput is low when Poll Timer values are small. Then it increases to a peak value and starts to decrease with the increase of Poll Timer values. The Poll Timer value at which the RLC goodput reaches its maximum value is about 300 ms, which is slightly

Figure 5.8: The Effect on RLC Goodput with Different BLERs

larger than the round trip time value. The reason is that the Poll Timer can function similar to the STATUS Period Timer once it starts. Then the sender polls STATUS reports at least periodically with the Poll Timer value as the period. As we discussed in Section 5.4.1.3, the RLC goodput can reach its maximum value with the polling period slightly larger than the round trip time.

5.5 Summary

This chapter made validations for our simulation model and presented the experimental results obtained by simulation. The impacts of the STATUS Period Timer and the Poll Timer on the SDU average delay, the RLC throughput, and the RLC goodput were discussed and analysed. The trends of the two timers settings were drawn by the simulation results and could be used to provide guidance on the design of optimised 3G UMTS systems for data services.

Chapter 6

Conclusion

The research in this thesis has investigated the performance of the UMTS RLC protocol in a scenario of a bulk data transfer from a RNC to a UE. Various design options have been evaluated under different wireless environment using simulation. Simulation results were employed to study the effects of different protocol configurations on the RLC performance in terms of the SDU average delay, the RLC throughput, and the RLC goodput. To support this study, a process model of the RLC protocol has been developed with the OPNET simulator [Inc02].

6.1 Summary of Contributions

The research in this thesis has made several contributions that are discussed in the following sections.

6.1.1 Evaluation of the STATUS Period Timer on the RLC Performance

Our simulation results suggest that, based on a large enough transmission and reception window size, the RLC protocol can have the best performance when the value of STATUS Period Timer is set to be slightly larger than the round trip time. At such a value of the STATUS Period Timer, the SDU average delay can be minimum, and the RLC goodput can reach the maximum value. When the timer is smaller or larger than this value, the SDU average delay becomes larger and the RLC goodput becomes smaller because of unnecessary retransmissions of AMD PDUs at a smaller value of the STATUS Period Timer, and more chances of the transmission window blocking at a larger STATUS Period Timer value.

6.1.2 Evaluation of the Poll Timer on the RLC Performance

Based on a large enough transmission and reception window size, our simulation results show that the Poll Timer value should never be less than the round trip time in order to reach the best RLC performance. The larger is the block error rate, the closer is the value of Poll Timer to the round trip time.

6.1.3 Implementation of the RLC Protocol

To carry out this study, the RLC protocol was implemented with the OPNET simulation environment. The polling mechanism, the STATUS transmission mechanism, and the SDU discard mechanism have been completely built in the implemented RLC process, in addition to the basic RLC functions described in Section 4.2. The simulation is capable of

exploring the system behavior when complex parameter settings are present. It will be used to investigate the TCP performance when the TCP, as an upper layer protocol, is overlaid on 3G UMTS systems in our future studies.

6.1.4 Improvement of an Analytical Result of the SDU Average Delay

In this study, an existing analytical model of the SDU average delay of out-of-sequence delivery was further investigated. Based on the given theoretical results in which the time between the detection of a missing PDU by the receiver and the transmission of a STATUS PDU to the sender is not computable, a lower bound, an upper bound, and an approximate computable average delay of out-of-sequence delivery were obtained. Then, the simulation results were compared with the analysed results, and they are agreeable to each other. The approximate computable analytical result of the SDU average delay of out-of-sequence delivery will be used to analyse the SDU average delay of in-sequence delivery mathematically.

6.2 Future Work

This study has only evaluated two of the RLC retransmission triggers on the RLC performance independently from each other. As described in Chapter 3, there are more triggers that can invoke transmission of a STATUS report, and two or more triggers can be set at the same time. Other interesting issues, such as mobility, block-out (when behind a building) of end users, etc. were not addressed in this thesis. Our future studies will focus on the optimization of two or more triggers which are simultaneously active after the function of SDU

segmentation/reassembly is completely implemented under the wireless environments with mobility and block-out natures.

As mentioned in Section 5.3, the RLC performance can be affected by both different higher layer traffic models and transmission error models of wireless links. Our future research will also be conducted on the optimization of the retransmission triggers of the RLC protocol under different higher layer traffic models as well as error models with bursty nature in the Physical layer.

As our major goal of the wireless network studies, the performance of TCP over UMTS networks and the interactions between RLC and TCP, will be more deeply investigated when TCP is used as an upper layer protocol for data service.

Bibliography

- [APL⁺95] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin. AIR-MAIL: A link-layer protocol for wireless networks. 1:47–60, February 1995.
- [aUoSC81] Information Science Institute at University of Southern California. RFC 793: Transmission control protocol. Technical report, Arlington, Virginia, September 1981. Web page: www.rfc-editor.org/rfc/rfc793.txt.
- [Bao96] Gang Bao. Performance evaluation of TCP/RLP protocol stack CDMA wireless link. In *International Conference on Communication Technology Proceedings (ICCT'96)*, volume 2, pages 710–713, Beijing, China, May 1996.
- [BB95] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proceedings of 15th International Conference on Distributed Computing Systems (ICDCS)*, volume 1, pages 136–143, Vancouver, Canada, May 1995.
- [Bhu72] A. Bhushan. RFC 354: The file transfer protocol (FTP). Technical report, July 1972. Web page: www.globecom.net/ietf/rfc/rfc354.html.
- [BPSK97] H. Balakrishnan, Venkata N. Padmanabhan, S. Seshan, and R.H. Katz. A comparison of mechanisms for improving TCP performance over wireless links.

In *IEEE/ACM Transactions on Networking (TON)*, volume 5(6), December 1997.

- [BSAK95] H. Balakrishnan, S. Seshan, E. Amir, and R.H. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the first annual international conference on Mobile computing and networking*, volume 1, Berkeley, CA, December 1995.
- [BV99] S. Biaz and N. H. Vaidya. Discriminating congestion losses from wireless losses using interarrival times at the receiver. In *Proceedings of IEEE Asset'99*, volume 1, pages 10–17, Richardson, TX, March 1999.
- [BZRO00] Yong Bai, Pengfei Zhu, Ashok Rudrapatna, and Andy T. Ogielski. Performance of TCP/IP over IS-2000 based CDMA radio links. In *Proceedings of IEEE 52nd vehicular Technology Conference*, volume 3, pages 1036–1040, Boston, MA, september 2000.
- [CB00] A. Chockalingam and Gang Bao. Performance of TCP/RLP protocol stack on correlated fading DS-CDMA wireless links. In *IEEE Transactions on Vehicular Technology*, volume 49 NO. 1, pages 28–33, January 2000.
- [CMO99] P. Chaudhury, W. Mohr, and S. Onoe. The 3GPP proposal for IMT-2000. 37 No 12:72–81, December 1999.
- [DJ01] Wenqing Ding and A. Jamalipour. A new explicit loss notification with acknowledgement for wireless TCP. In *Proceedings of IEEE 12th Interna-*

- tional Symposium on Personal, Indoor and Mobile Radio Communications*, volume 1, pages B-65 – B-69, San Diego, CA, USA, September 2001.
- [EL02] Nicolas Enderle and Xavier Lagrange. Radio link control-acknowledged mode protocol performance modeling in UMTS. In *Proceedings of IEEE 4th International Workshop on Mobile and Wireless Communications Network*, volume 1, pages 332–336, September 2002.
- [FGM⁺97] R. Fileding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners. RFC 2068: Hypertext transfer protocol. Technical report, January 1997. web page: www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html.
- [FL03] Cheng Pend Fu and S. C. Liew. TCP veno: TCP enhancement for transmission over wireless access networks. 21(2):216–228, February 2003.
- [GG00] Vijay K. Garg and Mridu Garg. Role of MAC/RLC in achieving higher-data rates and QoS in third generation (3G) UMTS system. In *IEEE International Conference on Personal Wireless Communications*, volume 1, pages 459–463, Hyderabad, India, December 2000.
- [GKSS00] Christina Gessner, Reinhard Kohn, Jorg Schniedenharn, and Armin Sitte. UTRA TDD protocol operation. In *Proceedings of 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 1226–1230, London, UK, September 2000.
- [Inc02] OPNET Technologies Inc. Optimised network engineering tools (OPNET) release 9.0. Technical report, 2002.

- [Jac88] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM Computer Communication Review*, volume 18 (4), August 1988.
- [KK00] Jeong Geun Kim and Marwan M. Krunz. Delay analysis of selective repeat ARQ for a Markovian source over a wireless channel. In *IEEE Transactions on Vehicular Technology*, volume 49 NO. 5, pages 1968–1981, September 2000.
- [KKLN98] Douglas N. Knisely, Sarath Kumar, Subhasis Laha, and Sanjiv Nanda. Evolution of wireless data services: IS-95 to CDMA2000. 36(10):140–149, October 1998.
- [KKMN00] F. Khan, S. Kumar, K. Medepalli, and S. Nanda. TCP performance over CDMA2000 RLP. In *IEEE 51st Vehicular Technology Conference Proceedings (VTC2000-Spring)*, volume 1, pages 41–45, Tokyo, Japan, May 2000.
- [KQC01] Zoran Kostic, Xiaoxin Qiu, and Li Fung Chang. The performance of the WWW traffic due to interactions between TCP and RLP protocols in a cellular system. In *Proceedings of IEEE 54th Vehicular Technology Conference*, volume 2, pages 1158–1162, Atlantic City, NJ, October 2001.
- [KY02] F. Khafizov and M. Yavuz. Analytical model of RLP in IS-2000 CDMA networks. In *IEEE 56th Vehicular Technology Conference Proceedings (VTC2002-Fall)*, volume 1, pages 487–491, September 2002.

- [LD02] Haitao Lin and Sajal K. Das. TCP performance analysis of CDMA systems with RLP and MAC layer retransmissions. In *Proceedings of IEEE 10th International Conference on MASCOTS*, volume 1, pages 313–320, October 2002.
- [Leh02] Harvey Lehpamer. *Transmission Systems Design Handbook for Wireless Networks*. 2002. Artech House, Inc., Norwood, MA, ISBN: 1-58053-243-8.
- [LV01] Fabienne Lefevre and Guillaume Vivier. Optimizing UMTS link layer parameters for a TCP connection. In *IEEE 53rd Vehicular Technology Conference*, volume 4, pages 2318–2322, Rhodes, Greece, May 2001.
- [MMR96] M. Mathis, J. Mahdavi, and A. Romanow. RFC 2018: TCP selective acknowledgement options. Technical report, April 1996. web page: www.ietf.org/rfc/rfc2018.txt.
- [Pos82] J. B. Postel. RFC 821: Simple mail transfer protocol (smtp). Technical report, August 1982. Web page: www.faqs.org/rfcs/rfc821.html.
- [PR83] J. Postel and J. Reynolds. RFC 854: telnet protocol specification. Technical report, May 1983. Web page: www.ietf.org/rfc/rfc0854.txt.
- [rlc02] Radio link control (RLC) protocol specification (release 5). Specification 3GPP TS 25.322 V5.3.0, 3rd Generation Partnership Project (3GPP), December 2002. Web page: www.3gpp.org.
- [rlp00] Data service options for spread spectrm systems: Radio link protocol type 3. Specification C.S0017-0-2.10 Version 2.0, 3rd Generation Partnership Project 2 (3GPP2), August 2000. Web page: www.3gpp2.org.

- [Sta00] William Stallings. *Data & Computer Communications*. 2000. sixth edition, Prentice-Hall Inc., Upper Sanddle River, New Jersey, ISBN 0-13-084370-9.
- [Ste97] W. Stevens. RFC 2001: TCP slow start, congestion avoidance, fast retransmit and fast recovery algorithms. Technical report, January 1997. Web page: www.faqs.org/rfcs/rfc2001.html.
- [SVSB02] P. Sinha, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: a reliable transport protocol for wireless wide-area networks. In *Proceedings of Wireless Networks*, volume 8 (2/3), Seattle, WA, March 2002.
- [umt01] Packet data convergence protocol (PDCP) protocol specification. Specification 3GPP TS 25.323 V4.2.0, 3rd Generation Partnership Project (3GPP), September 2001. Web page: www.3gpp.org.
- [umt02] Services provided by the physical layer. Specification 3GPP TS 25.302 V4, 3rd Generation Partnership Project (3GPP), 2002. Web page: www.3gpp.org.
- [umt03a] Medium access control (MAC) protocol specification. Specification 3GPP TS 25.321 V3.15.0, 3rd Generation Partnership Project (3GPP), March 2003. Web page: www.3gpp.org.
- [umt03b] Radio resource control (RRC) protocol specification. Specification 3GPP TS 25.331 V3.14.0, 3rd Generation Partnership Project (3GPP), March 2003. Web page: www.3gpp.org.
- [Vai01] Nitin H. Vaidya. Tutorial on TCP for wireless and mobile hosts. Technical report, 2001. Web page: <http://www.crhc.uiuc.edu/nhv>.

- [WL99] Jackson W.K. Wong and Victor C.M. Leung. Improving end-to-end performance of TCP using link-layer retransmission over mobile internetworks. In *Proceedings of IEEE International Conference on communications*, volume 1, pages 324–328, Vancouver, Canada, June 1999.
- [XCX⁺02] Hua Xu, Yi-Chiun Chen, Xiao Xu, Eren Gonen, and Peijuan Liu. Performance analysis on the radio link control protocol of UMTS system. In *IEEE 56th Vehicular Technology Conference*, volume 4, pages 2026–2030, September 2002.
- [XXC⁺02] Xiao Xu, Hua Xu, Yi-Chiun Chen, Eren Gonen, and Peijuan Liu. Simulation analysis of RLC timers in UMTS systems. In *Proceedings of the 2002 winter simulation conference*, volume 1, pages 506–512, December 2002.
- [YB94] R. Yavatkar and N. Bhagvat. Improving end-to-end performance of TCP over mobile internetworks. In *Mobile 94 Workshop on Mobile Computing Systems and Applications*, volume 1, pages 146–152, December 1994.
- [YH] K. Yen and L. Hanzo. Third-generation wireless systems. In *Wireless Radio Communications*, pages 897–963. 2nd edition, Edited by R. Steel and L. Hanzo, John Weley & Sons, Inc., New York, 1999, ISBN: 0-471-97806-X.
- [yHS02] Hung yun Hsieh and Raghupathy Sivakumar. Transport over wireless networks. pages 289–308, 2002. Edited by I. Stojmenovic, John Wiley & Sons, Inc., New York, ISBN: 0-471-41902-8.

- [YPM00] Uooyeol Yoon, Seongsoo Park, and Pall S. Min. Performance analysis of multiple rejects ARQ at RLC(radio link control) for packet data service in W-CDMA system. In *Proceedings of IEEE GLOBECOMM '00*, volume 1(27), pages 48–52, San Francisco, CA, December 2000.
- [ZS02] Qinqing Zhang and Hsuan-Jung Su. Performance of UMTS radio link control. In *Proceedings of IEEE International conference on Communications*, volume 5, pages 3346–3350, 2002.

Index

2.5G, 25

Acknowledged Mode (AM), 39, 41

Acknowledged Mode Data (AMD), 42

acknowledgement (ACK), 9

Acknowledgement super-field (ACK), 46

Advanced Mobile Phone System (AMPS),
25

AMD PDU, 43

Automatic Repeat reQuest (ARQ), 2, 20,
29

base station, 15

bit error rate, 2, 16

Bitmap super-field (BITMAP), 46

Block Error Rate (BLER), 81, 90

Broadcast/Multicast Control (BMC), 32

bulk data transfer, 90, 109

CDMA2000, 3, 26

Code-Division Multiple Access (CDMA),
25

Common Physical Channels (CPHCHs),
27

congestion avoidance, 2, 13

congestion control, 2, 3, 11

congestion window, 11–13

Control plane (C-plane), 32

Core Network (CN), 32

cumulative acknowledgements, 9

data services, 3, 39

data traffic, 18

Dedicated Physical Channels (DPHCHs),
27

discrete event-driven network simulator,
65

duplicate ACKs, 14, 17, 18

Duplicate detection, 68

end-to-end delay, 2

end-to-end protocols, 22

Error correction, 67

Explicit Loss Notification (ELN), 22

exponential increase, 12

first generation (1G), 25

Fixed Host, 21

flow control, 11, 68

Frequency Modulated (FM), 25

FTP, 1, 15, 35, 90

General Packet Radio Service (GPRS), 25

Global System for Mobile Communica-
tions (GSM), 25

HTTP, 1, 15, 36

In-sequence delivery, 68

in-sequence delivery, 65, 86, 88, 89, 93

International Telecommunication Union (ITU),
25

IS-136 or Digital AMPS (D-AMPS), 25

IS-95 or cdmaOne, 25

Iu, 32

linear increase, 13

Link Access Control (LAC), 26

Link-layer protocols, 19

List super-field (LIST), 46

Medium Access Control (MAC), 26, 32,
38

mobile host, 15, 27

Move Receiving Window Acknowledge-
ment super-field (MRW_ACK), 46

Move Receiving Window super-field (MRW),
46

Multiple Rejects ARQ (MR-ARQ), 3

- Negative Acknowledgement (NAK), 29
- network congestion, 2
- No More Data super-field (NO_MORE), 46
- Node B, 32
- Nordic Mobile Telephone (NMT), 25
- Open System Interconnection (OSI) reference model, 8
- OPNET, 5, 65, 68
- out-of-sequence delivery, 81, 84, 85, 88, 89, 93
- Packet Data Convergence Protocol (PDCP), 32
- performance criteria, 35
- performance evaluation, 3, 66, 71, 89, 91
- performance metrics, 89
- Piggybacked STATUS PDU, 43
- Poll Timer, 4, 89, 92, 102–106, 108, 110
- polling mechanism, 54
- protocol parameters, 49
- Quality of Service (QoS), 39
- Radio Frequency (RF), 25
- Radio Link Control (RLC), 3, 6, 7, 32, 38
- Radio Link Protocol (RLP), 28
- Radio Link Protocol Type 3, 29
- Radio Network Controller (RNC), 32, 64
- Radio Network Subsystems (RNSs), 32
- Radio Resource Control (RRC), 32, 38
- receiving side, 40, 58
- reception window, 45, 59
- RESET ACK PDU, 43
- RESET PDU, 43
- Retransmission Time-Out (RTO) timer, 9
- RLC goodput, 66, 89, 99–101, 105, 107, 109
- RLC performance, 38
- RLC Protocol Data Units (PDUs), 38, 39
- RLC Service Data Unit (SDU), 40
- RLC throughput, 65, 89, 97–99, 101, 105, 108, 109
- Round Trip Time (RTT), 18
- SDU average delay, 5, 65, 81, 85, 87, 93–97, 102–104, 109, 111
- SDU discard, 68
- SDU discard mechanism, 57
- SDU discard rate, 81, 86
- second generation (2G), 25
- segmentation/reassembly, 40
- Selective ACKnowledgements (SACKs), 22
- Service Access Points (SAPs), 33
- simulation framework, 64
- simulation results, 80, 83, 85
- slow start, 2, 12, 13
- SMTP, 1, 15
- split-connection protocols, 23
- ssthresh, 12–14
- State Transition Diagram (STD), 68
- state variables, 52, 59
- STATUS PDU, 43, 45, 46
- STATUS Period Timer, 4, 89, 92, 110
- STATUS report, 43
- STATUS transmission mechanism, 60
- SUPer-Field, 45
- TCP performance, 2, 5
- TCP reliable transmission, 9
- TCP throughput, 17, 19, 21
- TCP/IP, 1, 8, 15
- telnet, 1, 15
- theoretical results, 80, 83, 85
- third generation (3G), 3, 7
- Third Generation Partnership Project (3GPP), 32
- throughput, 2
- Time-Division Multiple Access (TDMA), 25
- Total Access Communications Systems (TACS), 25
- traffic model, 3
- Transfer of user data, 66
- Transmission Control Protocol (TCP), 7
- Transmission Time Interval (TTI), 38, 40, 68
- transmission window, 52
- transmitting side, 40, 51

- Transparent Mode (TM), 39, 40
- Transparent Mode Data (TMD), 40
- TTI operations, 62
- UMTS Terrestrial Radio Access Networks (UTRAN), 32
- Unacknowledged Mode (UM), 39, 40
- Unacknowledged Mode Data (UMD), 41
- Universal Mobile Telecommunications System (UMTS), 31
- User Equipment (UE), 32, 64
- User plane (U-plane), 32
- Uu, 32, 64
- voice traffic, 18, 33
- Wideband CDMA (WCDMA), 31
- Wireless networks, 16
- wireless networks, 1–3, 16, 18, 19, 22, 23, 25, 26, 36
- wireline networks, 10