1-1-2016

Techniques for End-to-End Tcp Performance Enhancement Over Wireless Networks

Bong Ho Kim
University of Pennsylvania, kimbongho@yahoo.com

Techniques for End-to-End Tcp Performance Enhancement Over Wireless Networks

Abstract
Today's wireless network complexity and the new applications from various user devices call for an in-depth understanding of the mutual performance impact of networks and applications. It includes understanding of the application traffic and network layer protocols to enable end-to-end application performance enhancements over wireless networks. Although Transport Control Protocol (TCP) behavior over wireless networks is well known, it remains as one of the main drivers which may significantly impact the user experience through application performance as well as the network resource utilization, since more than 90% of the internet traffic uses TCP in both wireless and wire-line networks. In this dissertation, we employ application traffic measurement and packet analysis over a commercial Long Term Evolution (LTE) network combined with an in-depth LTE protocol simulation to identify three critical problems that may negatively affect the application performance and wireless network resource utilization: (i) impact of the wireless MAC protocol on the TCP throughput performance, (ii) impact of applications on network resource utilization, and (iii) impact of TCP on throughput performance over wireless networks. We further propose four novel mechanisms to improve the end-to-end application and wireless system performance: (i) an enhanced LTE uplink resource allocation mechanism to reduce network delay and help prevent a TCP timeout, (ii) a new TCP snooping mechanism, which according to our experiments, can save about 20% of system resources by preventing unnecessary video packet transmission through the air interface, and (iii) two Split-TCP protocols: an Enhanced Split-TCP (ES-TCP) and an Advanced Split-TCP (AS-TCP), which significantly improve the application throughput without breaking the end-to-end TCP semantics. Experimental results show that the proposed ES-TCP and AS-TCP protocols can boost the TCP throughput by more than 60% in average, when exercised over a 4G LTE network. Furthermore, the TCP throughput performance improvement may be even superior to 200%, depending on network and usage conditions. We expect that these proposed Split-TCP protocol enhancements, together with the new uplink resource allocation enhancement and the new TCP snooping mechanism may provide even greater performance gains when more advanced radio technologies, such as 5G, are deployed. Thanks to their superior resource utilization efficiency, such advanced radio technologies will put to greater use the techniques and protocol enhancements disclosed through this dissertation.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Insup Lee

Keywords
End-to-end Performance, Network Performance, Split-TCP, TCP, Wireless Network

Subject Categories
Computer Sciences | Engineering

TECHNIQUES FOR END-TO-END TCP PERFORMANCE ENHANCEMENT OVER WIRELESS NETWORKS

Bong Ho Kim

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2016

Supervisor of Dissertation

_____

Insup Lee

Cecilia Fitler Moore Professor of Computer and Information Science

Graduate Group Chairperson

_____

Lyle H. Ungar, Professor of Computer and Information Science

Dissertation Committee:

Boon Thau Loo, Professor of Computer and Information Science

Jonathan M. Smith, Olga and Alberico Pompa Professor of Computer and Information Science

Andreas Haeberlen, Professor of Computer and Information Science

Doru Calin, Director at Bell Labs, Nokia and Adjunct Professor at Columbia University

TECHNIQUES FOR END-TO-END TCP PERFORMANCE ENHANCEMENT OVER WIRELESS NETWORKS

ACKNOWLEDGMENT

I would like to express my special appreciation and thanks to my advisor, Professor Insup Lee, for the continuous support of my PhD study, for his patience, motivation, and immense knowledge. I also would like to thank you for allowing me to continue my PhD study by reinstating my PhD student status after the multiple and long leave of absents related to my job.

I would also like to thank my committee members, Professor Boon Thau Loo, Professor Jonathan Smith, Professor Andreas Haeberlen, and Dr. Doru Calin for serving as my committee members even in hardship. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

I would especially like to thank Dr. Doru Calin for encouraging and supporting me to continue my PhD study, for in depth discussions on my dissertation, for devoting his precious time to review my dissertation and for proofreading of my papers and dissertation. I could not have imagined achieving my degree without your tremendous support.

I will forever be thankful to my former college research advisor, Professor Jungyul Na. You remain my best role model for a scientist, mentor, and teacher. I still think fondly of my time as an oceanographer in the lab with the rotating table.

A special thanks to my family. Words cannot express how grateful I am to my mother (Yoon Jin Moon), and father (Rae Seok Kim) for all of the sacrifices that you've made on my behalf. At the end, I would like express appreciation to my beloved wife Soo Jung Lim, who spent sleepless nights with and was always my support without any doubt. Thank you.

ABSTRACT

TECHNIQUES FOR END-TO-END TCP PERFORMANCE ENHANCEMENT OVER WIRELESS NETWORKS

Bong Ho Kim

Insup Lee

Today's wireless network complexity and the new applications from various user devices call for an in-depth understanding of the mutual performance impact of networks and applications. It includes understanding of the application traffic and network layer protocols to enable end-to-end application performance enhancements over wireless networks. Although Transport Control Protocol (TCP) behavior over wireless networks is well known, it remains as one of the main drivers which may significantly impact the user experience through application performance as well as the network resource utilization, since more than 90% of the internet traffic uses TCP in both wireless and wire-line networks. In this dissertation, we employ application traffic measurement and packet analysis over a commercial Long Term Evolution (LTE) network combined with an in-depth LTE protocol simulation to identify three critical problems that may negatively affect the application performance and wireless network resource utilization: (i) impact of the wireless MAC protocol on the TCP throughput performance, (ii) impact of applications on network resource utilization, and (iii) impact of TCP on throughput performance over wireless networks. We further propose four novel mechanisms to improve the end-to-end application and wireless system performance: (i) an enhanced LTE uplink resource allocation mechanism to reduce network delay and help prevent a TCP timeout, (ii) a new TCP snooping mechanism, which according to our experiments, can save about 20% of system resources by preventing unnecessary video packet transmission through the air interface, and (iii) two Split-TCP protocols: an Enhanced Split-TCP (ES-TCP) and an Advanced Split-TCP (AS-TCP), which significantly

improve the application throughput without breaking the end-to-end TCP semantics. Experimental results show that the proposed ES-TCP and AS-TCP protocols can boost the TCP throughput by more than 60% in average, when exercised over a 4G LTE network. Furthermore, the TCP throughput performance improvement may be even superior to 200%, depending on network and usage conditions. We expect that these proposed Split-TCP protocol enhancements, together with the new uplink resource allocation enhancement and the new TCP snooping mechanism may provide even greater performance gains when more advanced radio technologies, such as 5G, are deployed. Thanks to their superior resource utilization efficiency, such advanced radio technologies will put to greater use the techniques and protocol enhancements disclosed through this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

4G	Fourth Generation
5G	Fifth Generation
ACK	Acknowledgement
ARQ	Automatic Repeat reQuest
BSR	Buffer Status Report
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
CDMA-1xRTT	Code Division Multiple Access-1x (single-carrier) Radio Transmission Technology
CDN	Content Distribution Networks
DC	Data Center
DL	DownLink
EEN	Early Error Notification
EH	Enhanced Split-TCP Host
eNB	Evolved Node B
ELN	Explicit Loss Notification
EPC	Evolved Packet Core
ES-TCP	Enhanced Split-Transport Control Protocol
EU	European Union
FCHL	Fundamental Channel
FEC	Forward Error Correction
FH	Far Host
FTP	File Transfer Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPSEC	IP Security
I-TCP	Indirect TCP
ITU	International Telecommunication Union
LFN	Long Fat Network
LTE	Long Term Evolution
M2M	Machine-to-Machine
MAC	Medium Access Control
MAPI	Messaging Application Programming Interface
MH	Mobile Host
MME	Mobility Management Entities
MP-TCP	Multipath TCP
MSR	Mobility Supporting Router
M-TCP	Mobile TCP
OPNET	OPNET Technologies, Inc
P_BSR	Periodic BSR
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PDU	Protocol Data Unit
PGW	Packet Data Network Gateway
PHY	Physical Layer
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
ReTx_BSR_Timer	Retransmit BSR Timer

RFC	Request For Comment
RLC	Radio Link Control
RNC	Radio Network Controller
RRC	Radio Resource Control
RTO	Retransmission Timeout
RTT	Round Trip Time
RTTVAR	Round Trip Time Variation
SACK	Selective ACK
SCH	Supplemental Channel
SDU	Service Data Unit
SH	Split Host
SINR	Signal to Interference and Noise Ratio
SGW	Serving Gateway
SRTT	Smoothed Round Trip Time
TCP	Transport Control Protocol
TCP-RST	TCP Reset
UE	User Equipment
UL	Uplink
UMTS	Universal Mobile Telecommunications System
VoIP	Voice over IP
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access

CHAPTER 1

Introduction

Mobile wireless systems are in rapid evolution due to the continuous increase in user's demands for seamless connectivity, high throughput, and stringent quality of service (QoS) requirements to match the experience of broadband fixed networks. In order to provide the necessary performance and system capacity, standardization activities for a next generation wireless technology (i.e. 5G) have already begun [52]. The global mobile traffic growth forecast estimates that traffic will increase nearly ten times by 2020, and more than 50% of the total mobile traffic will be from 4G wireless network by 2016 [21]. In addition to the human users, objects and machines are also becoming increasingly connected. Machine-to-Machine (M2M) communications, for example in the Internet of Things (IoT) and sensor-based networks are additional drivers for wireless network traffic growth. The Future Internet drivers are all kinds of services and applications that require a wide range of throughput rates, a wide range of latency values, and are running on a variety of devices. The new data services and applications are key success factors for the wide deployment of the mobile broadband networks. Hence, the demands for supporting the evolving mobile applications and for improving application and network performance become increasingly critical to end users and to network operators as well. There are various technical areas to support and enhance the application performance.

The structured nature of network protocol layers, such as application, transport, network, MAC, and PHY layer, allow for each protocol layer to be updated and enhanced independently. Particularly, the performance of the TCP protocol is critical to the end-to-end application performance, since more than a decade of internet traffic analysis consistently shows that more than 90% of the internet traffic uses TCP in the network [27][29]. The internet traffic analysis results in [16] and [37] show that the amount of TCP traffic is even more than 95% of the total

internet traffic. As a higher layer protocol, TCP traffic flows across both wired and wireless networks. It is why with the wireless traffic volume growing exponentially, the optimization of the TCP performance over the wireless network is of critical importance.

Because of the wide range of usage and popularity of the current internet protocol (TCP/IP), it is very likely that it will be used even more in the future, as the Internet Engineering Task Force (IETF) standardization community continues to improve the TCP protocol. For example, one of the activities in the TCP research community for the past several years has been targeting a Multipath TCP (MPTCP) solution [28] that establishes multiple parallel TCP connections with multiple addresses, instead of a single TCP connection. This requires the TCP protocol stack to be modified, but it still runs over the traditional TCP protocol, since each of the TCP sub-flows is operated as a traditional TCP connection. Thus, the TCP performance improvement with a single TCP connection is still critical for the end-to-end application performance enhancement.

As the fourth generation LTE [1] high data rate wireless networks are widely deployed, and the LTE-Advanced and 5G wireless technology with even higher wireless data rates are being expected in the near future, optimizing the TCP performance over these advanced wireless networks will have an even more significant impact on application and network performance.

In fact, TCP is known to have a poor performance over unreliable wireless network where packet losses due to air-link transmission errors are significantly higher than the packet losses due to the network congestion. Many research materials that try to improve TCP performance over wireless networks may be found in the research communities for the past two decades.

1.1 Challenges

The cross-layer network protocol interactions introduce a high degree of complexity in terms of understanding and enhancing the performance of various applications and the utilization of wireless networks. Having as main objective to improve the performance of applications and

network utilization, possible enhancements in TCP performance may contribute significantly to this goal, but they should be carried out considering the cross protocol layer behaviors because of the mutual interactions.TCP performance in particular could be significantly affected by other network layer behaviors under various wireless network conditions. The network research communities are not under-estimating a potential performance impact caused by the cross layer protocol interactions, but it is generally more common to target isolated parts of the network protocol stack.

Consequently, this dissertation has identified the following three critical problems that may negatively affect the application performance and wireless network resource utilization.

- **Impact of the wireless MAC protocol on the TCP throughput performance:** From the analysis of the MAC scheduling protocols in LTE, we found out that the LTE uplink bandwidth scheduling may be stalled temporarily and cannot allocate uplink resources to a User Equipment (UE) until a timer is expired. Consequently, the uplink packet delivery could be delayed from hundreds of milliseconds to above 10 seconds. This interruption could trigger a TCP Retransmission Timeout (RTO) which would significantly degrade the application performance.
- **Impact of applications on wireless network resource utilization:** From the analysis of HTTP-based video streaming traffic, we found out that a significant volume of video data that is downloaded over the air-interface ends up being discarded by the end device, which results in an inefficient use of wireless network resources. This problem can be mitigated using the TCP header information.
- **Impact of TCP on throughput performance over wireless network:** The poor TCP performance over wireless networks is a known problem, and the Split-TCP mechanism is a good candidate for enhancing TCP performance. However, this mechanism breaks the end-to-end semantics of TCP, and this may impact the fundamental behavior of the protocol.

1.2 Objectives

This dissertation is focused on designing an enhancement in the LTE uplink resource scheduling protocol, a new TCP snooping mechanism, and two novel Split-TCP mechanisms, the ES-TCP and the AS-TCP. These contributions are listed below:

- **An enhancement LTE uplink resource scheduling protocol:** It reduces the uplink packet delay and reduces the volume of control signals. This helps prevent the TCP to time out, hence eliminating events which may be cause for throughput degradation.
- **A new TCP snooping mechanism:** This consists in monitoring the TCP Acknowledgement (ACK) packets and prevents unnecessary video content transmission over air-interface which may end up not be consumed by the end user. The proposed mechanism brings various advantages as well, such as: (i) causing no interruption on the end-to-end connections between client and server, (ii) improving the air-link bandwidth utilization for both uplink and downlink, including for uplink signaling control resources, (iii) not requiring modifications on the existing TCP or application layer, and (iv) allows for easy integration in the existing base stations, since it may be implemented as an independent module with little interaction with the other system modules.
- **Two novel Split-TCP protocols, the ES-TCP and the AS-TCP:** The ES-TCP and AS-TCP solutions maintain the end-to-end protocol semantics without losing the performance gain attributed to the Split-TCP. To evaluate the TCP throughput performance gain using ES-TCP and AS-TCP, we implememented them on top of the end-to-end LTE network simulation platform [14] that can estimate the application performance reflecting various protocol interactions and network congestion conditions, including air-link impairments.

1.3 Outline of Dissertation

The previous section has presented three well-defined objectives, and. the work performed towards each objective is described in corresponding chapters. The outline of this dissertation is as follows:

Chapter 2 contains the background of the TCP protocol and sources of the challenges for improving application and network performance. We firstly describe the fundamental characteristics of the TCP protocol and congestion control, followed by an example of cross-layer protocol interactions that negatively impact the application performance, by analyzing network packet traces over a commercial CDMA-1XRTT network [40]. Secondly, we describe some issues with the TCP protocol over wireless networks. Lastly, several existing TCP performance enhancements are described.

In Chapter 3, we analyze the performance of the LTE MAC layer packet transmission protocol. We show that the uplink bandwidth scheduling may be temporarily stalled and cannot allocate uplink resources to a UE until a timer is expired. Consequently, this uplink resource allocation interruption could trigger a TCP Retransmission Timeout (RTO) which would significantly reduce the transmission rate at the traffic source. To prevent this problem, we propose a LTE link layer protocol enhancement that reduces the uplink packet transmission interruption time [14][39].

Chapter 4 provides a mechanism that improves the wireless network resource utilization using a Snoop TCP mechanism. We have analyzed a few popular HTTP-based video streaming services over 3G, 4G-LTE and Wi-Fi networks. We found that the online video traffic behavior depends on various factors, such as the type of mobile device (iOS and Android), multimedia applications running on the devices, and wireless network (3G, 4G-LTE, and Wi-Fi) conditions. Furthermore, we found out that a significant volume of video content downloaded through a busy network may end up being discarded by the end device. This results in an inefficient use of wireless network resources. We propose a mechanism to prevent an eNB transmitting the video traffic that will be

discarded by the user device [42][43]. This enhancement is effective not only for the video traffic, but also effective for any type of applications with similar traffic behavior.

In Chapter 5, we propose two Split-TCP enhancement mechanisms, the ES-TCP and the AS-TCP, which maintain the end-to-end protocol semantics without losing the performance gain attributed to the Split-TCP. To evaluate the TCP throughput performance gain using ES-TCP and AS-TCP, we implemented them on top of the end-to-end LTE network simulation platform [14] that can estimate the application performance, reflecting various protocol interactions and network congestion conditions, including air-link impairments.

Finally, Chapter 6 states the main conclusions drawn through this research dissertation.

1.4 Publications

The work developed during this dissertation made possible the publications to the following journal and conferences. We also filed patents for a part of the proposed solutions in this dissertation.

Publications

[1] D. Calin and **B. Kim**, “LTE Application and Congestion Performance” Bell Labs Technical Journal, vol. 18, no 1, June 2013, pp. 5-25.

[2] H. Nam, **B. Kim**, D. Calin, and H. Schulzrinne, “Mobile Video is Inefficient: A Traffic Analysis,” December, in the Proc, of IEEE GlobeCom Control Techniques for Efficient Multimedia Delivery Workshop 2013, pp. 512 - 517, Atlanta, GA, December 9 – 13, 2013.

[3] H. Nam, K.H Kim, **B. Kim**, D. Calin, and H. Schulzrinne, Towards A Dynamic QoS-aware Over-The-Top Video Streaming in LTE, in the Proc, of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) 2014, Sydney, Australia, June 16 – 19, 2014.

Submitted papers

[1] **B. Kim** and D. Calin, “On the Split-TCP Performance over Real 4G LTE and 3G Wireless Networks,” submitted to IEEE Communications Magazine, 2016

[2] **B. Kim**, D. Calin, and I. Lee, “Enhanced Split TCP with End-to-End Protocol Semantics over Wireless Networks,” submitted to IEEE GlobeCom Mobile and Wireless Networks Symposium, 2016.

[3] **B. Kim**, D. Calin, I. Lee, “Advanced Split TCP with End-to-End Protocol Semantics over LTE Wireless Network,” submitted to IEEE GlobeCom Communication QoS, Reliability and Modelling Symposium, 2016.

Patents

[1] **B. Kim** and D. Calin, “Method and Apparatus for Controlling Buffer Status Report Messaging,” US Patent 8,630,202, January 14, 2014 (granted).

[2] **B. Kim** and D. Calin, “Method And Apparatus For Preserving End To End Semantics And Performance Enhancement Of Split-TCP Protocols,” US Patent filed 14/943128, November, 17, 2015

CHAPTER 2

Background

This section contains following sub-sections, TCP overview, congestion control mechanism, challenges of TCP over wireless network, related works to overcome these challenges, and in the last sub-subsection we provides the comparison of these TCP performance enhancement mechanisms. The fundamental TCP protocol and congestion mechanism show how those challenging wireless network environment would degrade the TCP throughput performance, and the comparison section highlights a weakness from the existing TCP enhancement mechanism that should be strengthened.

2.1 TCP Overview

The purpose of TCP is to provide a reliable, connection oriented service to the application layer. Therefore, its end-to-end reliability between a host and a data network is an important design criterion. TCP was originally designed for wired links, where the packet error rate is typically low and packet losses are due to congestion in the network. Every data byte that a TCP sender transmits has a logically associated with a sequence number starting from a randomly selected number. When a TCP endpoint receives a packet containing data, it sends back an acknowledgment (ACK) with a sequence number that is 1 byte larger than the last received sequence number indicating the next expected sequence number. When a sender receives an acknowledgment for data in its window, it can remove that data from the transmission buffer, since it has been successfully delivered to the receiver. This is the most important property of TCP, the end to end semantics of TCP. Since either the packet or its ACK packet could be lost in the network, a sender starts a timer, which is a function of the estimated round-trip time (RTT) when it transmits a packet. If the timer expires before an ACK packet is received, then the sender retransmits the outstanding packets assuming that the packet had been lost. This is called a

retransmission timeout (RTO), which is estimated each time that a new round-trip time is measured. The round-trip time (RTT) is based on the time difference between a data packet transmission and ACK packet corresponding to the data packet. RFC 6298 [51] suggests that the initial RTO before any RTT measurement should be set either at 1 or 3 seconds and how RTO should be estimated.

In the late 1980's a congestion control mechanism (RFC 2581 [4] and RFC 2914 [24]) was added to the TCP and, since then, a number of optimizations to this mechanism, such as TCP NewReno (RFC 3782 [25]) and TCP SACK (RFC 2018 [45] and RFC 2883 [26]) have been proposed and standardized. The TCP congestion control mechanism is to improve the performance when there is network congestion indicated by a packet loss. This became a main limitation of the classic TCP's congestion control mechanism when a packet loss is not directly caused by the network congestion.

2.1.1 TCP Congestion Control Algorithms

The congestion control algorithm implemented in TCP is divided in two major parts: a “slow start” region and a “congestion avoidance” region. These two regions are governed by independent algorithms and follow different objectives. They are based on a congestion window (*cwnd*) parameter at the TCP sender, which is used to adapt the sender transmission rate to the congestion status of the link. Another important parameter is a slow start threshold (*ssthresh*), which separates the slow start phase from the congestion avoidance phase: the former allows the *cwnd* size to grow exponentially, while the latter limits it to a linear increase.

Figure 1 illustrates how a typical TCP (i.e. NewReno) flow control works and reacts to a packet loss. The x-axis captures the time dimension and each tick indicates a Round Trip Time (RTT) value. The y-axis indicates the number of TCP segments transmitted by the TCP transmitter as a function of the RTT, which is referred as the congestion window (*cwnd*) value. The TCP maintains an estimate of how much unacknowledged data is in-transit in the network, without causing

congestion. The actual amount of in-transit data is limited by the minimum *cwnd* size at the transmitter and by the receiver's advertised window (*rwin*) size [20][22][57]. When a TCP connection is established, it starts conservativelly with a very low transmission speed because it does not know the network condition yet. That is why the beginning phase is called "Slow Start". In this diagram in Figure 1 the the initial *cwnd* is set to 1. Each time an ACK is returned from the targeted destination, the sender increases its *cwnd* by 1 during the slow start phase. This actually increases the *cwnd* size exponentially until it hits the *ssthresh*. Some implementations set the initial *ssthresh* value to the receiver window size, which is controlled by a TCP receiver, while some implementation uses infinite. If infinite is used for the initial *ssthresh* value, then the slow start will continue until a packet loss occurs. In this example diagram, the *ssthresh* is set to 32. Once the *cwnd* reaches *ssthresh*, the congestion avoidance phase starts. The difference between the slow start phase and the congestion avoidance phase is the rate of the *cwnd* size growth. Instead of doubling the *cwnd* size per RTT, it increases only 1 segment per RTT. Whenever a receiver receives a packet after a missing packet, it sends an ACK packet with the same expected sequence number as included in the immidiatly prior ACK packet. This is called a duplicate ACK, and when three duplicate ACKs are received at the sender, it immediately retransmits the potentially missing packet without waiting for Round Trip Timeout (RTO) to be expired. This is called "fast retransmission." In Figure 1, the *cwnd* size increase is stopped after seeing three duplicate ACKs that indicates a segment was not acknowledged (at RTT = 6). After seeing three duplicate ACKs, the sender immediately sets the *ssthresh* to half of the current *cwnd*; this effectively halves the transmission rate and starts the congestion

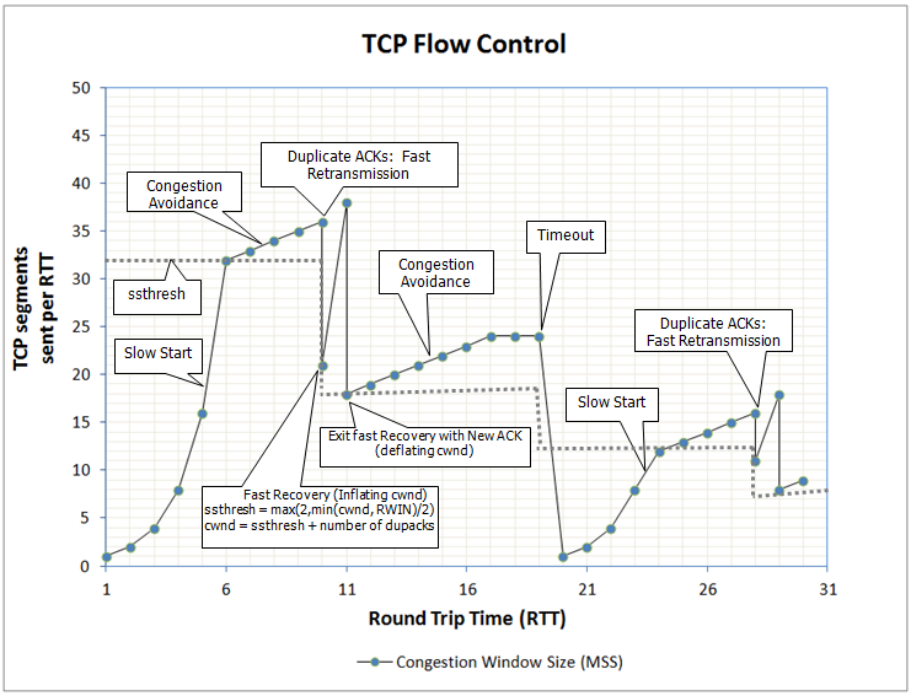


Figure 1: TCP flow control mechanism

avoidance phase instead of entirely closing the *cwnd* to one, and sets the *cwnd* size to *ssthresh* plus the number of duplicate ACKs. The reason of adding "number of duplicate ACKs" is that the transmitter has received 3 duplicate ACKs, which indicates that at least 3 segments have been transferred successfully, thus allowing the increase of *cwnd*. This is indicated as "inflating *cwnd*" at the 10th RTT. The *cwnd* increased by one when another ACK is received for all packets transmitted in that window and a packet loss was observed. After all packets in that window are acknowledged, *cwnd* is deflated to the *ssthresh*, which was set at the time of fast retransmission start. This is indicated as "deflating *cwnd*" at the 11th RTT. This mechanisms are called fast retransmission (for "inflating *cwnd*") and fast recovery (for "deflating *cwnd*"), respectively.

The retransmission scheme relies on a RTO timer to expire before a lost segment is retransmitted. However, the fast retransmit algorithm avoids this problem and the fast recovery algorithm prevents the *cwnd* to entirely collapse after a single retransmission. This algorithm is

able to recover from one lost segment within one RTT. as a second lost segment cannot be detected due to the lack of corresponding duplicate ACKs. If a second segment is lost within one RTT, the RTO timer will expire and the *cwnd* will collapse to one segment. However, if the packet loss was not detected by three duplicated ACKs, and instead by the RTO timer expiry, the *cwnd* size is set to one and starts with a slow start phase and a congestion avoidance afterwards [4] [56]. If the RTO timer expires again, the RTO value increases exponentially until it reaches 64 seconds. The RTO timeout event is observed at the 19th RTT in Figure 1. One can notice that in case of a packet loss event over the air caused by poor airlink conditions, the TCP sender reduces its transmission rate by 50% and increases the transmission rate very slowly. Hence, the TCP interprets the packet errors over wireless channels as network congestion, which may trigger frequent bandwidth underutilization events, in particular when the wireless propagation conditions are challenging.

2.1.2 TCP Packet Trace Analysis

Figure 2 and Figure 3 shows the detailed behavior of TCP and how the transmission rate is dynamically adjusted to the network condition during the lifetime of a connection over the LTE network. In Figure 2, the X-axis represents the time of packet capture, the Y-axis represents the TCP sequence number, and the slope of the graph gives the TCP throughput over time. Figure 3 shows the data transmission rate via outstanding data in bytes during the TCP session. The outstanding data is the amount of the unacknowledged data that can be used to estimate the congestion window at the sender.

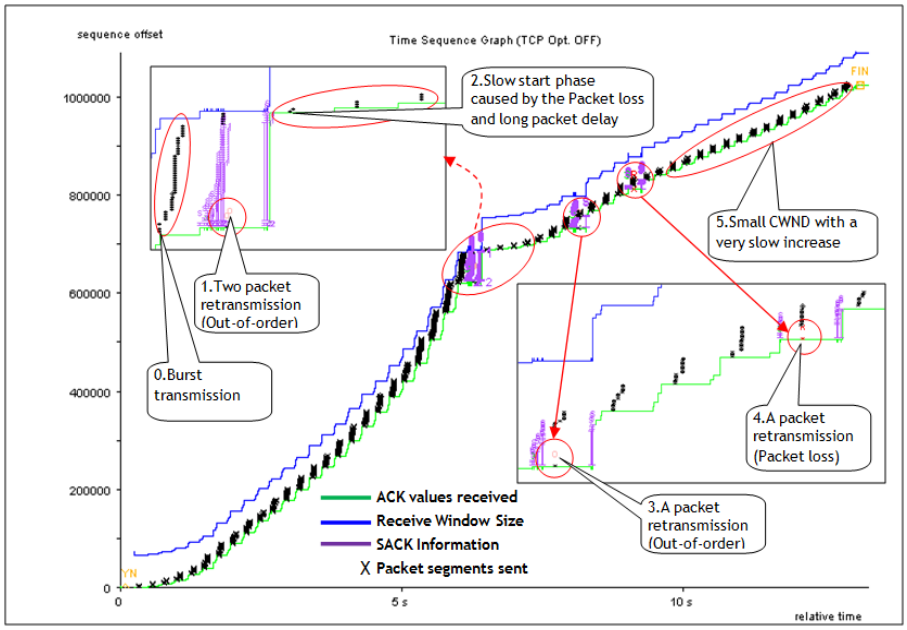


Figure 2: TCP trace over LTE network: TCP sequence number (server to UE)

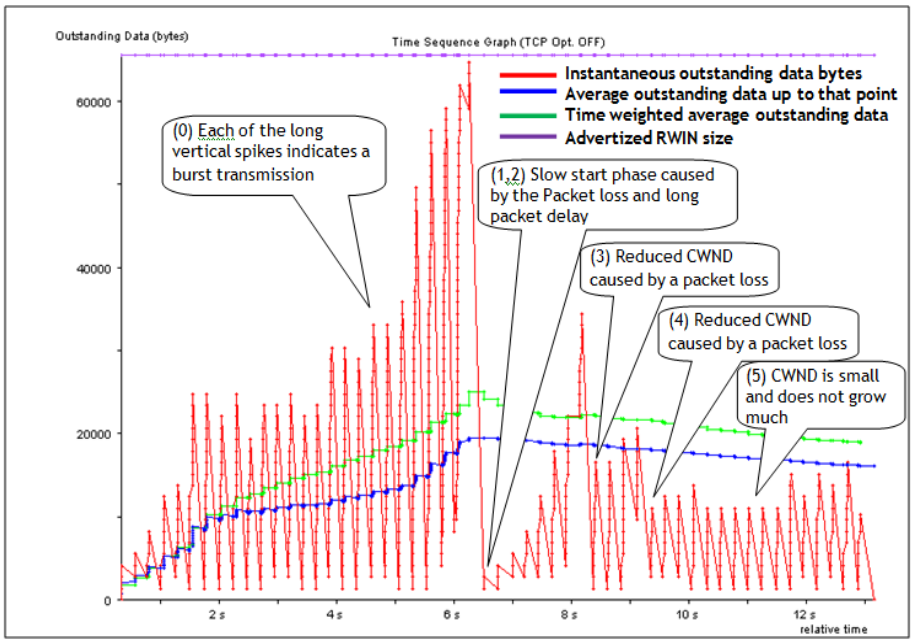


Figure 3: TCP trace over LTE network: outstanding data bytes (estimated *cwnd*)

- The Red Line represents instantaneous outstanding data bytes during the lifetime of the connection.
- The Blue Line tracks the average outstanding data up to that point. The average outstanding unacknowledged data (in bytes) calculated from the sum of all the outstanding data (in bytes) divided by the total number of samples.
- The Green Line tracks the time weighted average outstanding unacknowledged data up to that point.
- The Yellow Line tracks the window advertised by the opposite end-point i.e., the receive window.

Several important factors are marked in the dialog box with an index in Figure 2 and Figure 3

- Dialog Box (0): The TCP transmits a burst of data when the packet transmission opportunity opens in a short period of time as shown in the dialog box (0). When the *cwnd* size is increased, the burstiness may also increase and eventually may lead a long network queuing delay, a packet drop from a network or many duplicate ACKs if there was a packet loss in the network.
- Dialog Box (1): This burstiness may increase and may eventually lead a long network queuing delay, a packet drop from a network or many duplicate ACKs if there was a packet loss in the network. Two packet retransmissions caused by the three duplicate ACKs and a long RTT is observed. The combinations of these events seem to cut down the slow start threshold (*ssthresh*) value and trigger the slow start phase which would set the *cwnd* to one.
- Dialog Box (2): Shows that the amount of data transmission per RTT is very small (small *cwnd*) and is gradually growing.
- Dialog Box (3): Another packet retransmission is observed in Figure 2 and Figure 3 which shows that the *cwnd* size is reduced to half of the previous *cwnd* size.

- Dialog Box (4): Soon after the 3rd packet retransmission, the 4th packet retransmission is observed, which shows that the *cwnd* size is reduced to half of the previous *cwnd* size again.
- Dialog Box (5): The continuous packet loss reduced the slow start threshold (*ssthresh*) value to very low and started a congestion avoidance phase with a very low *cwnd* value which would reduce the *cwnd* growth rate. Figure 3 clearly shows that the *cwnd* value stays very low until the TCP connection is terminated.

2.1.3 Problems of TCP over Wireless Network

The TCP provides connection oriented, reliable service to the application layer. It is intended for use as a highly reliable end-to-end protocol between hosts in a data network. It was originally designed for the wired links where the error rate is low and assumes that packet losses are due to congestion in the network. However, hosts in wireless networks move frequently while communicating and as they share the media for communication they experience a lot of interference from the environment [10]. The wireless systems suffer from different environmental properties like radio signals related with fading, shadowing, interference, mobility, handovers and variable bandwidth, which dramatically affect the TCP performance. Following are the four types of the typical problems associated with wireless networks and TCP protocol.

- **High bit error rate and loss:** The error behaviors of wireless links suffer from multi-path fading due to terrestrial obstructions. Mobility constantly changes the fading and interference characteristics of the link. The interference caused by external sources increases their error rates significantly and introduces unpredictability in their performance. Unfortunately, when packets are lost in networks for reasons other than congestion, these measures result in an unnecessary reduction in end-to-end TCP throughput, since it is misinterpreted by a TCP sender as congestion. TCP regards all data loss as a notification of network congestion and lowers its sending rate accordingly.

- **Long end-to-end delay and delay variation:** In general, a typical Radio Access Network (RAN) adds a longer network delay compare to the WAN. A part of the additional delay is in order to improve the performance of the wireless link, and it is from the link layer that applies to the ARQ and FEC. In general, the link layer ARQ and FEC can provide an almost error free packet service to the upper layer traffic. However, the retransmission by ARQ introduces latency and jitter to the link layer flow. Typical RAN delay in 3G is about 100 ms and 4G LTE is about 30 ms to 60 ms. It is, however, not rare to observe a longer than 100s ms RAN delay in the LTE network as well. This extended RTT itself significantly degrades the TCP performance.
- **Packet transmission Interruption:** The TCP performance over wireless network suffers from significant throughput degradation and very high interactive delays [12] caused by intermittent connectivity due to a handover as an example. The handover procedure for instance often results in a variation in packet delays or in packet losses that can lead to disconnections lasting from a few tens of milliseconds to a lot longer than a few seconds.. If the status of the moving TCP connection has a large *cwnd* prior to handover and the bandwidth allocation from the new wireless connection is narrow, then this may lead to a long packet delay for the packets already in-flight. This may shutdown the *cwnd* to one if RTO occurs. The TCP interprets these disconnections as network congestion and invokes a congestion control mechanism, which is unnecessary [9]. This results in the TCP timeout and lowering its congestion window, thus reducing the efficiency of the transmission.
- **Wireless Link Rate Variability:** To maintain the transmission quality under various air-link conditions, channel condition based scheduling is used in the wireless network. This means that the transmission bandwidth over the air is not fixed but dynamically fluctuates. While channel condition based scheduling improves overall throughput, it also increases the rate variability. While link layer retransmission protocol improves TCP throughput and

channel state based scheduling algorithms improves a cell throughput, this improvement comes at the expense of increased delay and rate variability.

The common factors that significantly affecting TCP throughput performance over the wireless network from these four types of the typical problems are the network RTT delay and the packet loss rate. Since the TCP transmitter requires receiving acknowledgements from the receiver for every window of data packets sent, a standard TCP throughput is inversely proportional to the end-to-end Round Trip Time (RTT_{e2e}) and the square root of end-to-end packet error rate (P_{e2e}) as shown in (1) [49]. Thus, the distance and the packet error rate between the server and the end user becomes a true bottleneck factor of the TCP throughput and hard to overcome unless the server is relatively close to the end user.

$$Throughput_{e2e} \sim \frac{1}{RTT_{e2e} * \sqrt{P_{e2e}}} \quad (1)$$

2.2 Related Works

In this Section we briefly discuss a set of selected works related to the topic focused in this dissertation, including a cross layer interaction research that affect the application performance and three different categories of TCP performance enhancement strategies.

2.2.1 Cross Layer Interaction Affecting Application Performance

The research [40] identifies that the interaction between the application traffic behavior and the wireless MAC scheduler causes application performance degradation. It also indicates the importance of considering the interaction with an application traffic behavior on designing the wireless MAC scheduling algorithm. The authors of [40] analyzed the application traffic over CDMA-1XRTT and dialup network using some of typical applications (i.e. Web, FTP, and Outlook Email). The network measurement results show that the CDMA-1XRTT system provides a throughput of 81 kbps on the average for FTP, and 53 kbps throughput for the Web browsing,

which are 3.3 times and 2.5 times respectively higher than the Dialup connection. Email application throughput result, however, shows only 1.5 times higher than the dialup connection.

The critical issue resulting in this unequal application throughput is because the CDMA-1XRTT MAC scheduler is designed to select link rate dynamically depending on the application packets buffered in the MAC layer. Thus, if an application (i.e. outlook email in this case) has a certain packet transmission pattern and if it is not aligned with the MAC scheduler well then this application may experience performance degradation regardless of the network resource availability. The proposed MAC scheduler, that considers the application traffic pattern, selectively improves the application throughput. The application experienced the worst throughput improves the most with the adaptive MAC scheduler proposed in [40].

2.2.2 TCP Protocol Enhancement Solutions

The effort improving TCP can be categorized into three groups: (i) End-to-end TCP solutions with various TCP flavors and Explicit Loss Notification (ELN), (ii) Link layer solutions, such as link layer ARQ and Snoop TCP, and (iii) Split-TCP that divides a TCP connection in to multiple TCP connections.

A. End-to-End TCP Solutions

Some of the TCP performance enhancement mechanisms do require TCP protocol stack modification and the changes to the TCP protocol is on the sender and receiver sides to distinguish wireless losses from the network congestion. One of three main strategies for the end-to-end TCP solution is to optimize the TCP *cwnd* size or enhance the congestion control algorithms. They are often referred to as "TCP flavors". The other strategy is to enhance the congestion control enhancement further. It can be improved with additional information called “Explicit Loss Notification”, which explicitly let the sender know it is a wireless loss. Using ELN information, TCP can provide a targeted congestion control for the wired and wireless network differently. It prevents the sender from reducing the congestion window size when it is a wireless

loss. The last strategy is using the Selective Acknowledgement (SACK), which allows the sender to recover from multiply packet losses without timeouts. Both ELN and SACK can be used together with one of the TCP flavors to increase the TCP performance over the wireless network.

TCP Flavors

Optimizations preferably are placed at the end-hosts to avoid adding complexity to the network. The intermediate node need not be TCP-aware. Some of the TCP variations listed below are certainly not exhaustive: Vanilla TCP, Tahoe, TCP Reno/New Reno, TCP SACK, Freeze TCP [31][32], TCP Westwood [15][44], CUBIC [33], and Multi-path TCP (MPTCP) [28]. TCP Reno is the most widely deployed version of TCP while TCP SACK is becoming deployed increasingly with the global spread of the wireless network. MPTCP is one of the latest TCP enhancements that utilize multiple network connections at the same time as most of the mobile devices have the capability of maintaining multiple network connections; those network environments are growing in the market with the wide spread use of Wi-Fi hot spots. These common TCP flavors are briefly described in this Section since most of the TCP enhancements are on top of these fundamental TCP protocols.

- Vanilla TCP: This is a standard TCP flavor that implements slow start and congestion avoidance, which is thus compliant to RFC 793 [55].
- TCP Reno/New Reno: The TCP "Reno" uses fast retransmits as defined in RFC 2001 once the node receives the n^{th} duplicate acknowledgement and then it enters the fast recovery phase. After receiving the lost packet, the receiver quickly recovers the congestion window (fast recovery algorithm). This enhancement allows recovering from one lost segment within one RTT [57].
- Freeze-TCP: This uses the Zero Window Acknowledgement (ZWA) and the Zero Window Probe (ZWP). A ZWA packet sets the *rwin* size to zero to pause the TCP sender from transmitting packets and A ZWP packet probes the TCP receiver to check if the *rwin* size is increased [31][32].

- TCP Westwood: TCP Westwood is a sender-side modification of the TCP congestion window algorithm. The improvement is most significant in wireless networks with lossy links, since TCP Westwood relies on end to end bandwidth estimations to discriminate the cause of packet loss. TCP Westwood performs poorly when random packet loss rates exceeds a few percent [15][44].
- CUBIC: TCP CUBIC uses a modified congestion control algorithm for high bandwidth networks with high latency as known as Long Fat Networks (LFN). It uses a unique congestion window (*cwnd*) algorithm. This algorithm tries to find the maximum where to keep the window at for a long period of time. It does not rely on the receipt of ACK packet to increase the window size. CUBIC's *cwnd* size is dependent only on the last congestion event, and the growth function is defined in real-time so that its growth will be independent of RTT [33].
- MPTCP: Multipath TCP (RFC 6824) provides the ability to simultaneously use multiple paths between sender and receiver, unlike the current TCP that restricts it to a single path per connection [28]. Nowadays, most of the mobile devices have the capability of maintaining multiple network connections and those network environments are growing in the market with the wide spread of Wi-Fi hot spots. Each TCP sub-flows use unmodified TCP stacks and multiple TCP sub-flows are aggregated at the MPTCP layer.

Explicit Loss Notification

The Explicit Loss Notification (ELN) mechanism is to inform the exact cause of packet loss to the TCP sender and prevents unnecessary congestion control actions using a bit in the TCP header [8][53]. The ELN requires TCP-snoop capability in the wireless network. Another mechanism under same category of ELN is Early Error Notification (EEN) [11]. The EEN mechanism is implemented in between wireless MAC layer and TCP layer. It notifies a packet loss to the TCP layer if a packet loss over the wireless link is determined even after the recovery process (i.e. ARQ) in the MAC layer so that the TCP layer can retransmit the lost packet instead of slowing

down the transmission rate. It focuses only on the uplink TCP connection over wireless network, and this also requires TCP-snoop capability.

Selective ACK

Selective acknowledgments (SACKs) allow the sender to more efficiently retransmit lost or delayed packets. During the fast retransmission phase, the sender first retransmits all suspected lost packets only before sending new ones. This allows recovery from several lost segments within one RTT [45].

B. Link Layer Solutions

Link Layer protocols are another alternative for improving the poor performance of TCP over the wireless link. In those methods, usually automatic repeat request (ARQ), TCP snooping, and forward error correction (FEC) methods are used to improve the performance. TCP is an end-to-end protocol that deals with global congestion problems, and as the problem of high bit error rate is a wireless link problem, this is a reasonable solution to localize the problem by providing faster local retransmission without informing the TCP layer [59]. This means that the TCP is not involved in handling wireless losses and the errors over wireless links, which are instead recovered by the link layer mechanisms.

Link Layer ARQ

In the majority of the current wireless system, the use of Automatic Repeat Request (ARQ) scheme in a Radio Link Protocol (RLP) layer allows recovery from wireless link errors and provides relatively reliable transfer of packets to the upper layers. The ARQ and forward error correction schemes are used at the link layer to obtain local data reliability and make the wireless link appear to TCP as a reliable link, although with a longer and variable delay [7][19][60][38]. This approach has the advantage of acting independently of TCP and not requiring any modification to the TCP implementation. On the other hand, the TCP sender cannot be completely shielded from the wireless link, because of both the interactions between the TCP and

the ARQ timers [10] and the reordering of the TCP segments performed by the link layer at the receiver. The interaction between the MAC ARQ and TCP may result in poor performance due to spurious retransmissions. The incompatible values for timers from TCP and ARQ can lead to competing retransmissions and therefore to poor performance. Moreover, ARQ has no support when link disconnection event occurs, that is to say that this solution can do little to prevent TCP from a timeout when an ACK packet does not arrive on time. The independent timer reaction at the link layer and transport layer may result in unnecessary retransmissions, and large RTT variations, which are considered as major problems with the link layer approaches. The extensive local retransmission may also increase the network delay variation, and the air-link condition based scheduling may also increase the link rate variability by dynamically changing the link data rate. This rate and delay variability introduces bursty ACK packet arrivals, which is also called ACK compression at the TCP sender

Snoop-TCP

The fundamental concept of Snoop-TCP mechanism is to read the TCP headers and take an appropriate action locally without TCP layer involvement. With the Snoop-TCP mechanism, the end-to-end semantics of TCP is maintained, but the problem is that it needs to read the TCP header of the packets at the network element (i.e. base station) so it cannot be applied to encrypted traffic such as IPSEC traffic that is readable only at the final destination or outside of the secure tunnel.

The Snoop protocol in [10] and the Rate-adaptive Snoop in [46] are a cache-based local recovery mechanism. These enhancement mechanisms are located in a base station to observe and cache TCP packets going out to the mobile host as well as acknowledgments coming back. The snoop agent determines what packets are lost on the wireless link and schedule a local link layer retransmission. Another Snoop-TCP based enhancement mechanism is “ACK Regulator” in [18]. It is designed for improving the downlink TCP performance in the presence of bandwidth and

delay variation. It mitigates the link rate variation to reduce the ACK compression problem that can be caused by the link variation.

C. Split TCP Solutions

As aforementioned, a standard TCP throughput is inversely proportional to the end-to-end Round Trip Time (RTT_{e2e}) and the square root of the end-to-end packet error rate (P_{e2e}) as shown in (2) and in Figure 4 [49]. Thus, the distance and the packet error rate between the server and the end user becomes bottleneck factors for the TCP throughput and are hard to overcome, unless the server is relatively close to the end user. The Split-TCP is one of mechanisms that may significantly improve the TCP throughput performance under challenging network conditions, by segmenting the RTT_{e2e} across different network segments and by isolating the packet error events to those network segments which are the root cause for problems, if there are multiple network segments that have different network characteristics. The conceptual Split-TCP diagram is illustrated in Figure 4, where the TCP_{RAN} is the connection between a Mobile Host (MH) and a Split-TCP Host (SH), and the TCP_{WAN} is the connection between a Far Host (FH) and the SH.

Split-TCP divides an end-to-end TCP_{e2e} connection characterized by the delay RTT_{e2e} and packet error rate P_{e2e} into two independent TCP connections: a TCP_{RAN} for the Radio Access Network (RAN) with corresponding delay (RTT_{RAN}) and packet error rate (P_{RAN}) and a TCP_{WAN} for the Wide Area Network (WAN) with corresponding delay (RTT_{WAN}) and packet error rate (P_{WAN}), as depicted in Figure 4 and represented in equation (2). This has the effect of isolating the packet errors to the network segment that is responsible for errors occurrence. Hence, the effective end-to-end TCP throughput is the minimum throughput across the individual TCP connections resulted from splitting the link, and the improvement is expressed in equation (3). One should remark, however, that the TCP throughput gain depends on the split point in the network, which is defined by the RTT_{e2e} split ratio and by the packet error rates in the TCP_{RAN} and TCP_{WAN} segments.

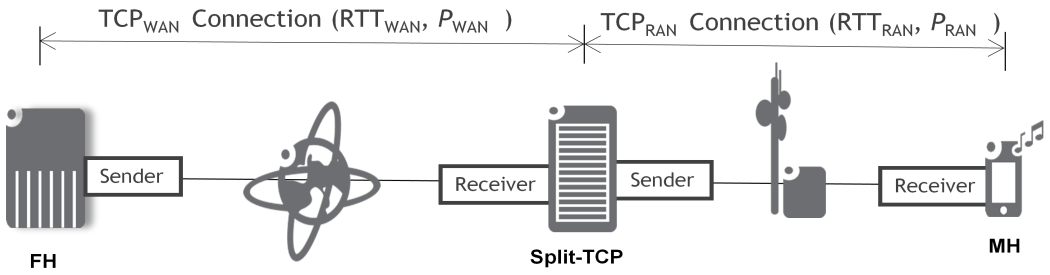


Figure 4 Conceptual diagram of Split-TCP

$$Thp_{e2e} \sim \frac{1}{\mathrm{RTT}_{\mathrm{e2e}} * \sqrt{P_{\mathrm{e2e}}}} \sim \frac{1}{(\mathrm{RTT}_{\mathrm{RAN}} + \mathrm{RTT}_{\mathrm{WAN}}) * \sqrt{P_{\mathrm{WAN}} + P_{\mathrm{RAN}}}} \quad (2)$$

$$Effective\ Thp_{e2e} \sim \min\left(\frac{1}{(\mathrm{RTT}_{\mathrm{WAN}}) * \sqrt{P_{\mathrm{WAN}}}}, \frac{1}{(\mathrm{RTT}_{\mathrm{RAN}}) * \sqrt{P_{\mathrm{RAN}}}} \right) \quad (3)$$

However, its major drawback is that acknowledgments received by the sender do not mean that all the packets have been received by the actual intended destination. When the MH moves away to another cell, or in the event that the SH crashes, some data packets may not be successfully delivered to the MH. If this is the case, the sender may well be informed that all data packets were delivered over the TCP$_{WAN}$ segment, but in reality, some of them may not be delivered to the MH over the TCP$_{RAN}$ segment. Since the end-to-end protocol semantics is violated, the data transmission is not reliable. This is damaging the layering structure of the internet protocols.

There are various types of Split-TCP mechanisms, including Indirect-TCP (I-TCP) [7], Aggregate TCP (ATCP) [17], TCP for Mobile Cellular Networks (M-TCP) [13], M-TCP+ [48], and Pre-Acknowledgement Mode Split-TCP (PAM Split-TCP) [5]. Among these Split-TCP mechanisms, M-TCP and M-TCP+ are the only options that maintain the end-to-end protocol semantics. The M-TCP and M-TCP+, which is a modified M-TCP, aim at improving the TCP efficiency while the wireless link suffers disconnections by preventing the sender from closing down the congestion window (*cwnd*) to the minimum. The SH using the M-TCP and M-TCP+ mechanism freezes the packet transmission from the FH when an air-link disconnection is detected [13][48]. The PAM

Split-TCP is also a modified M-TCP mechanism. It pre-acknowledges TCP packets to prevent packet transmissions from the FH when a long air-link delay is detected, but it does not maintain the end-to-end semantics [5].

The common goal of these three mechanisms is to handle the wireless link disconnection issues that may arise through a handover process or due to air-link failures. They don't address problems related to high packet error rate links or to significant RTT_{e2e} delays in the network [5][6].

While the M-TCP and M-TCP+ mechanisms maintain the end-to-end protocol semantics and isolate the disconnection issue in the wireless network, they present a few significant drawbacks. These mechanisms still propagate the packet losses on the wireless network segment into the wired network, since the ACK packets from a MH are rather forwarded to the FH. Holding on an acknowledgement from SH to FH until a data packet arrives at the destination, obviously maintains the end-to-end protocol semantics, but at the same time one looses the main benefit of the Split-TCP, which is shortening the RTT delay for TCP. Shortening RTT by itself could increase the TCP throughput performance significantly, even without link condition changes. Because of this pitfall, the performance improvement, if any, of these protocols would be limited.

2.3 Comparison of the TCP Performance Enhancement Mechanisms

Table 1 summarizes the capabilities of the various TCP enhancement mechanisms, which is intended to solve the TCP problems over the wireless network environment, and it is self explanatory. The comparison criteria that have been adopted are the capability of preserving end-to-end protocol semantics, the resiliency of handling long disconnection, the resiliency of handling high bit error rate, isolating packet error, splitting RTT_{e2e}, etc. Table 1 show that some of these mechanisms are complementary solutions, so the combination of the solutions would probably be an interesting way to improve the TCP over the wireless networks.

The M-TCP, M-TCP+ and PAM out of the split-TCP based solutions do not isolate or distinguish the wireless network from a wired network and do not split the RTT_{e2e} which is one of the key barrier for the TCP performance. The RLP, Snoop, and ELN mechanisms out of the non-split TCP based solutions neither have the capability of splitting the RTT_{e2e}. I-TCP solution has capability for these key items but it does not maintain the end-to-end protocol semantics which we address as a critical problem that needs to be resolved.

Table 1: Comparison of TCP performance enhancement mechanisms

	Non-split TCP Solution			Split TCP Solution			
	RLP	**Snoop**	**ELN**	**I-TCP**	**M-TCP**	**M-TCP+**	**PAM**
Maintain End-To-End Protocol Semantics	Yes	Yes	Yes	No	Yes	Yes	No
Require TCP Modification	No	No	Yes	Yes	Yes	Yes	No
Isolate/Distinguish Wireless Network	No	Yes	Yes	Yes	No	No	No
Split Long Network Delay (Split RTT)	No	No	No	Yes	No	No	No
Resilient to Lengthy Disconnection(Handoff)	No	Yes	Yes	Yes	Yes	Yes	Yes
Capable of Data Pre-fetch	No	No	No	Yes	No	Yes	No
Resilient to High BER	Yes	Yes	Yes	Yes	No	No	No
Resilient to High Delay Variation	No	Yes	No	Yes	No	No	No
Resilient to Link Rate Variation	No	No	No	Yes	No	No	No
Aware Of TCP	No	Yes	Yes	Yes	Yes	Yes	Yes

CHAPTER 3

Link Layer Enhancement via Resource Allocation Protocol

As [40] presented, the interaction between application traffic and the wireless MAC layer behavior may lead to an unexpected application performance and wireless network resource utilization. We have analyzed the performance of LTE MAC layer packet transmission protocols and identified that the LTE uplink bandwidth scheduling could be temporarily stalled and cannot allocate the uplink resources to user equipment (UE) until a timer associated with the resource scheduler is expired. Consequently, the uplink packet transmission could be delayed from hundreds of milliseconds to 10.24 seconds for some configurations and application traffic behaviors. This LTE uplink resource allocation interruption could trigger a TCP Retransmission Timeout (RTO) which significantly drops the TCP transmission rate at the traffic source. To prevent this problem, we have proposed a uplink resource allocation protocol enhancement that reduces the uplink packet transmission interruption time over LTE network [14][39]. This is done by removing an unnecessary interruption condition in the resource allocation process without introducing additional control traffic overhead.

3.1 Analysis of the Default LTE Uplink Resource Allocation Protocol

In dealing with applications, a conventional LTE network makes use of Buffer Status Report (BSR) messages. The Buffer Status reporting procedure is used to provide the serving eNB with information about the amount of data available for transmission in the uplink (UL) buffers of the UE. Radio Resource Control (RRC) controls BSR reporting by configuring two timers:

- A periodic BSR timer (e.g., “periodicBSR-Timer” and also called as “BSR periodic_timer”), which indicates the periodic timing of a BSR message.

- A retransmit BSR timer (e.g., “retxBSR-Timer” also called as “BSR retx_timer”), which indicates a retransmission wait period for retransmission of a BSR message upon transmission of the BSR message.

Figure 5 represents an example flowchart illustrating the default LTE protocol for sending BSR messages from a UE. At step 1 the UE is known to have an empty buffer. An empty buffer indicates that there is no data available for transmission in the UL buffers of the UE. At step 2, after packet arrival from the upper layer (i.e., application layer), the UE sends a regular BSR message which indicates the total buffer size to an eNB. The UE also starts the periodic BSR timer and the retransmit BSR timer. The UE then proceeds to wait for a grant message at step 3. While waiting for the grant, if the retransmit BSR timer times out before the receipt of the grant, the UE loops back to step 2 and again sends a regular BSR message, and restarts the periodic BSR timer and the retransmit BSR timer. While waiting for the grant, if the grant is received before the retransmit BSR timer times out, the UE proceeds to step 4, where it determines whether the UE buffer size is greater than the grant size.

If the buffer size is not greater than the grant size, at step 5, the UE transmits data with the data size equal to the buffer size, and terminates the periodic BSR timer and the retransmit BSR timer. After these activities, the UE loops back to step 1, at which point the buffer is once again known to be empty. If the buffer size is greater than the grant size, the UE determines whether the periodic BSR timer has expired at step 6. The value of the periodic BSR timer indicates a period of time to wait between transmissions of BSR messages. If the periodic BSR timer has not expired, the UE transmits a data message with a data size equivalent to the grant size at step 7, and then returns to step 3 to await another grant so as to be permitted to transmit additional portion(s) of the data remaining in the UL buffers at the UE. If the periodic BSR timer has expired, at step 8, the UE prepares a periodic BSR message to report the buffer size as the buffer size minus the difference of the grant size plus the BSR message size, transmits a data message with data equivalent to the grant size minus the BSR message size, and restarts the periodic BSR

timer and the retransmit BSR timer. The UE then returns to step 3 to await a grant to be able to transmit additional portion(s) of the data remaining in the UL buffer(s) at the UE.

Additional description for the BSR is found in the 3GPP document TS 36.321 [2] and the potential values for a periodic BSR timer (e.g., "periodicBSR-Timer") and a retransmit BSR timer (e.g., "retxBSR-Timer") are provided in the RRC document, namely 3GPP TS 36.331 [3]. The “MAC-Main Config” topic of Section 6.3.2 in 3GPP TS 36.331 lists those values for both timers. Some default values are suggested in Section 9.2.2 of the same document. The periodic BSR timer "periodicBSR-Timer" (default = infinity) is optional and the values are sf5, sf10, sf16, sf20, sf32, sf40, sf64, sf80, sf128, sf160, sf320, sf640, sf1280, sf2560, and infinity. For example, "sf5" means subframe 5 and it is equivalent to 5 msec; "sf10" means subframe 10 and it is equivalent to 10 msec, and so on. The retransmit BSR timer "retxBSR-Timer" (default = sf2560) values are sf320, sf640, sf1280, sf2560, sf5120, and sf10240.

When utilizing BSR messaging according to LTE standards, we noticed that performance degradation may occur prior to the expiration of the periodic BSR timer if the total grant size is equal to or greater than the reported size of the last BSR message sent. In such cases, it is very possible to encounter scenarios where the UE buffer continues to increase due to new packets created by applications on the UE (e.g., video, picture, document, etc.) but, the UE is unable to send another BSR message until the BSR retransmission timer expires. As a result, the eNB has no knowledge of the most recent UE buffer conditions and does not allocate any resources until it receives a new BSR request. Eventually, resources will be allocated after the BSR retransmission timer expires and a BSR message is forwarded by the UE to the eNB. However, the delay until expiration of the BSR retransmission timer may result in increased access delay and jitter beyond what can be tolerated per application. For instance, the VoIP packet can only tolerate a delay budget which is typically 200 ms to 250 ms end-to-end; beyond this delay

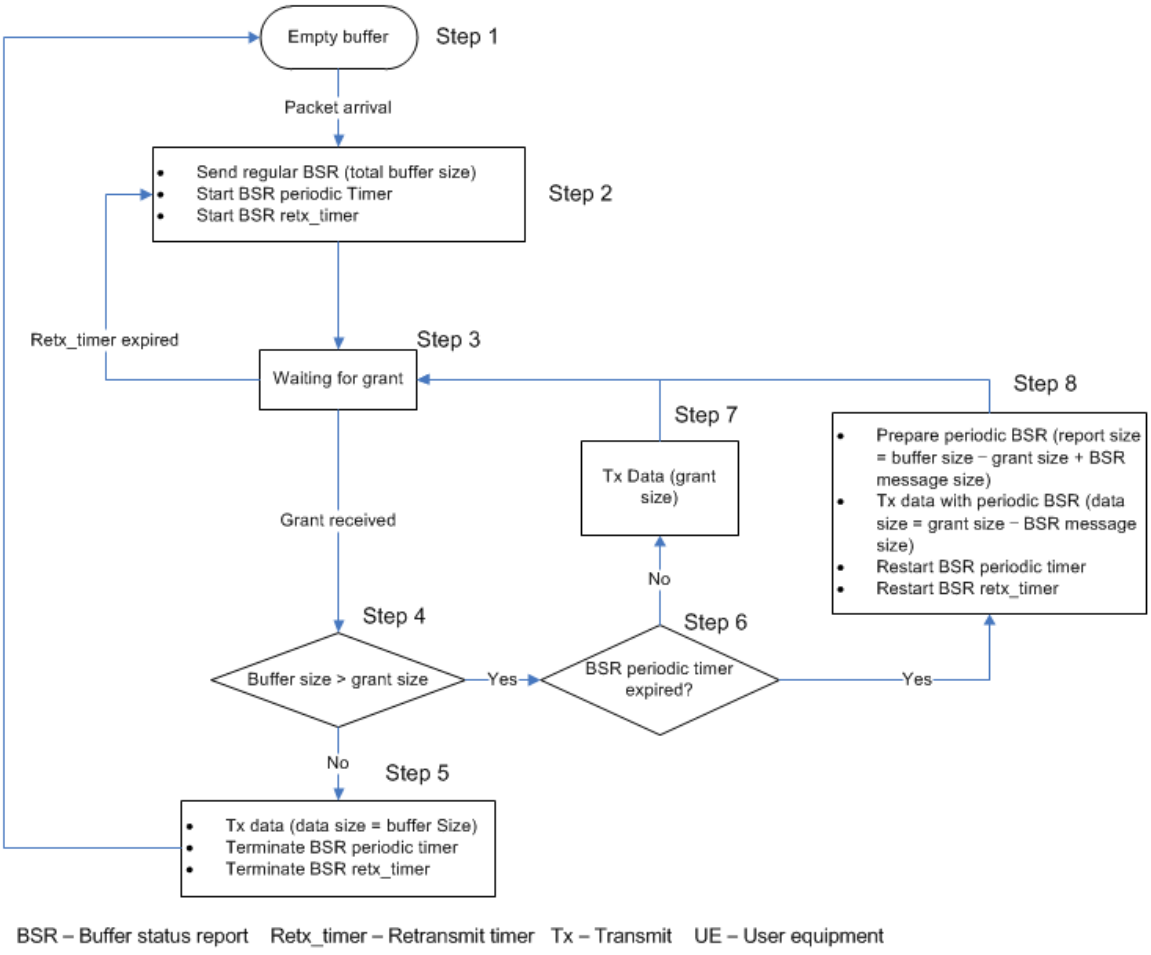


Figure 5: Default LTE protocol sending BSR messages

threshold VoIP packets may be dropped resulting in a reduced quality of service or quality of experience. In addition to the imminent delay problem, it may cause the TCP to timeout.

If the resource allocation is interrupted longer than the TCP Retransmission timeout (RTO), then the TCP will start retransmission and perform a slow start phase. This will significantly reduce the TCP transmission rate and the transmission rate may experience a very slow increase because of the reduced TCP slow start threshold (*ssthresh*) value.

Example Packet Flow: Uplink Resource Allocation Interruption with Default Protocol

Figure 6 is an example of communication flow illustrating packet transmissions between UE and eNB with packet delay performance degradation caused according to the default protocol for

sending BSR messages detailed in Figure 5. In the illustrated example of Figure 6, the periodic BSR timer is set to 10 subframes and the retransmit BSR timer is set to 2560 subframes. Time, denoted by times instances numbered T99, T100… T2669 runs from the top to bottom of Figure 6. While both the periodic BSR timer and the retransmit BSR timer are utilized, activity associated with the expiration of the retransmit BSR timer is in action/invoked in the illustrated example.

Referring to Figure 6, at time T99, 1500 bytes are placed in the UL buffer(s) of the UE. At time T100, a regular BSR message is transmitted from the UE to the eNB. The BSR message is characterized as regular since it is the initial BSR message. The BSR message indicates that there are 1500 bytes in the buffer of the UE. At time T102, an additional 300 bytes are placed in the UE's UL buffers, so that there are now 1800 bytes available for transmission. At time T104, the eNB transmits a first grant to the UE; the UE receives the first grant of 1500 bytes from the eNB. Since the buffer size is greater than the grant size and the periodic BSR timer not yet having expired (Figure 5: step 4, step 6), at time T108, the UE transmits a first data message of 1500 bytes (Figure 5: step 7).

Thereafter, the periodic BSR timer expires at T110 and the UE is ready to prepare a periodic BSR message (P_BSR). At this time, 300 bytes remain in the UE buffer. However, the UE has to wait (Figure 5: step 3) until the retransmit BSR timer expires before transmitting any additional data. At this point, the UE cannot send any BSR messages specifically because the eNB has already granted the 1500 bytes requested by the UE. Even though the periodic BSR timer has expired and the UE is ready to send a periodic BSR message for the remaining 300 bytes, the UE has to wait until it is granted permission to send another BSR message. A regular BSR message cannot be sent because the retransmit BSR timer is still running and must first expire before a regular BSR message can again be sent. During this interval until the retransmit BSR timer expires, packet delay and jitter performance degradation occurs.

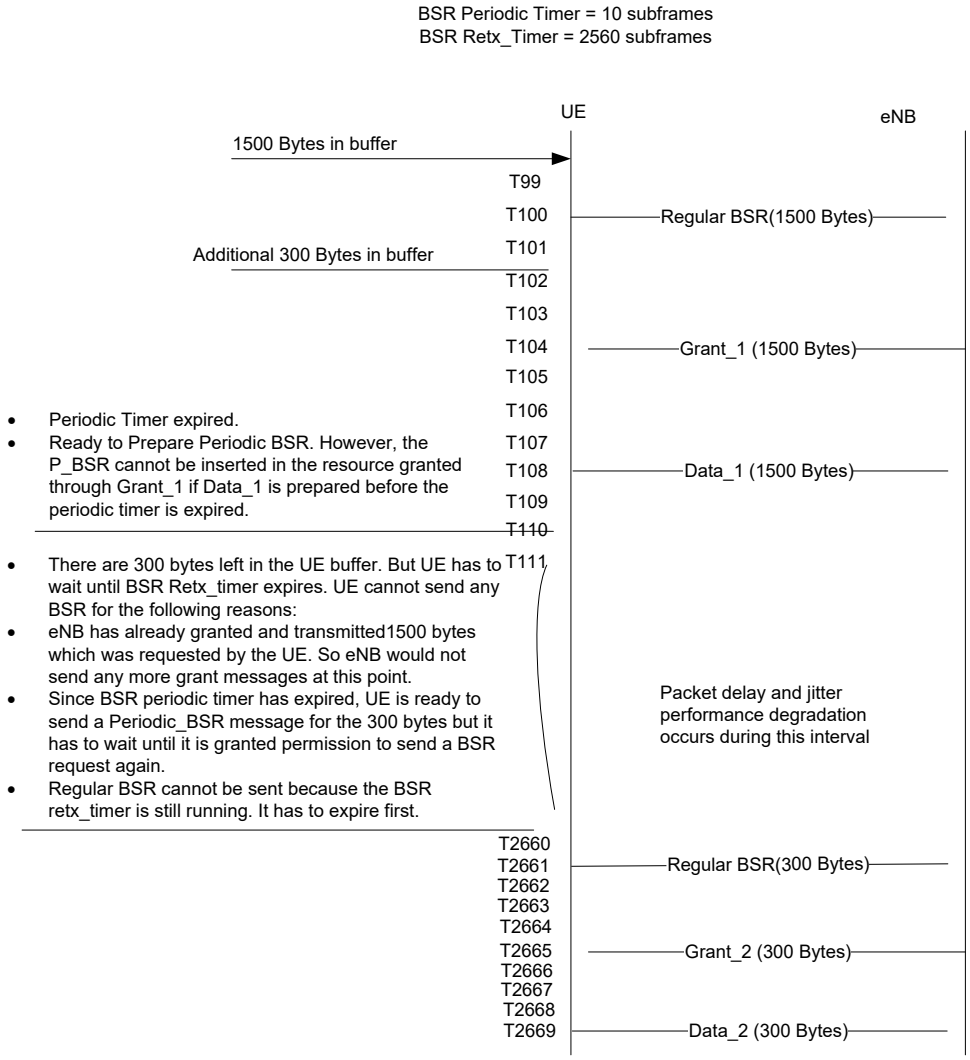


Figure 6: An example for packet delay and jitter performance degradation scenario with the default buffer management protocol. Periodic BSR is not in action and retx_timer is in action.

At time T2660, the retransmit BSR timer expires and accordingly, at time T2661 a regular BSR message is transmitted from the UE to the eNB to indicate that there are 300 bytes in the UE's UL buffer(s). At time T2665, the eNB transmits a second grant to the UE; and the UE receives the second grant for 300 bytes from the eNB. The buffer size is not greater than the grant size (Figure 5: step 4), so at time T2669, the UE transmits a data message of 300 bytes (Figure 5: step 5).

Note that a periodic BSR message (i.e., P_BSR) cannot be inserted in the resource granted and received before the periodic BSR timer expired. A P_BSR cannot follow a grant message which is received before the periodic BSR timer expires and the uplink data PDU was already prepared before a BSR trigger. This is because when the grant is received, a MAC Protocol Data Unit (MPDU) packet was already created for the transmission without considering insertion of periodic BSR message since the corresponding timer has not yet expired.

3.2 Known Approaches to Minimize the Problem

One remedy for the problem of packet delay performance degradation as described above is to set the BSR transmission and BSR retransmission timers such that the packet delay problem is minimized. The periodic BSR timer (e.g., "periodicBSR-Timer") can be set to a very small value so that the UE will inform the network in a timelier manner regarding the UE's current status. This action would alter the network to a potential UE data backlog, which would allow appropriate actions to be taken when possible. However, one drawback with a solution based on a small periodic timer value is that, the UE will generate far many more periodic BSR transmit requests, which in turn will result in an increase in signaling overhead. Since the control channels are the bottleneck for thin applications, increasing the signaling overhead further is not a desirable solution. Furthermore, the periodic BSR transmission interacts with the bandwidth grant message, but not the actual time, unlike how it is named.

Example Packet Flow: Uplink Resource Allocation Signaling Overhead Increase

Figure 7 is an example communication flow illustrating UE to eNB packet transmissions with signaling overhead increase according to the default protocol for sending BSR messages detailed in Figure 5, when the periodic BSR timer value is decreased. In the illustrated example of Figure 7, the periodic BSR timer is set to 5 subframes and the retransmit BSR timer is set to 2560 subframes. Time, denoted by times instances numbered T99, T100 … T120 runs from the top to

the bottom of Figure 7. While both the periodic BSR timer and the retransmit BSR timer are utilized, activity associated with the expiration of the retransmit BSR timer is not in action or invoked in the illustrated example.

At time T99, 10000 bytes are now in the buffer of the UE. At time T100, a regular BSR message is transmitted from the UE to the eNB. The BSR message is characterized as regular since it is the initial BSR message. The BSR message indicates that there are 10000 bytes in the buffer of the UE (i.e., 10000 bytes available for transmission in the UL buffer(s) of the UE). At time T101, an additional 1000 bytes are placed in the UL buffers of the UE by the application, so that there are now 11000 bytes available for transmission. At time T102, the eNB transmits a first grant to the UE, and the UE receives the first grant for 1000 bytes from the eNB. Since the buffer size is greater than the grant size and the periodic BSR timer has not yet expired (Figure 5: step 4, step 6), at time T106, the UE transmits a first data message of 1000 bytes (Figure 5: step 7). At time T105, the periodic BSR timer expires and the UE is ready to prepare a periodic BSR message.

At time T107, the UE receives a second grant for 1000 bytes from the eNB. The buffer size is now 10000 bytes (10000 initially + 1000 added – 1000 transmitted = 10000). Since the buffer size is greater than the grant size and the periodic BSR timer has expired (Figure 5: step 4, step 6), at time T111, the UE prepares and transmits a periodic BSR message and a second data message. The periodic BSR message transmitted reports the buffer size as 9010 bytes, which is the buffer size (10000 bytes) minus the grant size (1000 bytes) plus the BSR message size (e.g. 10 bytes). The transmitted data message has a data size equal to 990 bytes, which is a size equivalent to the grant size (1000 bytes) minus the BSR message size (10 bytes).

At time T116, the eNB transmits a third grant to the UE; the UE receives the third grant for 1000 bytes from the eNB. Since the buffer size is greater than the grant size and the periodic BSR timer has expired (Figure 5: step 4, step 6), at time T120, the UE transmits a third data message.

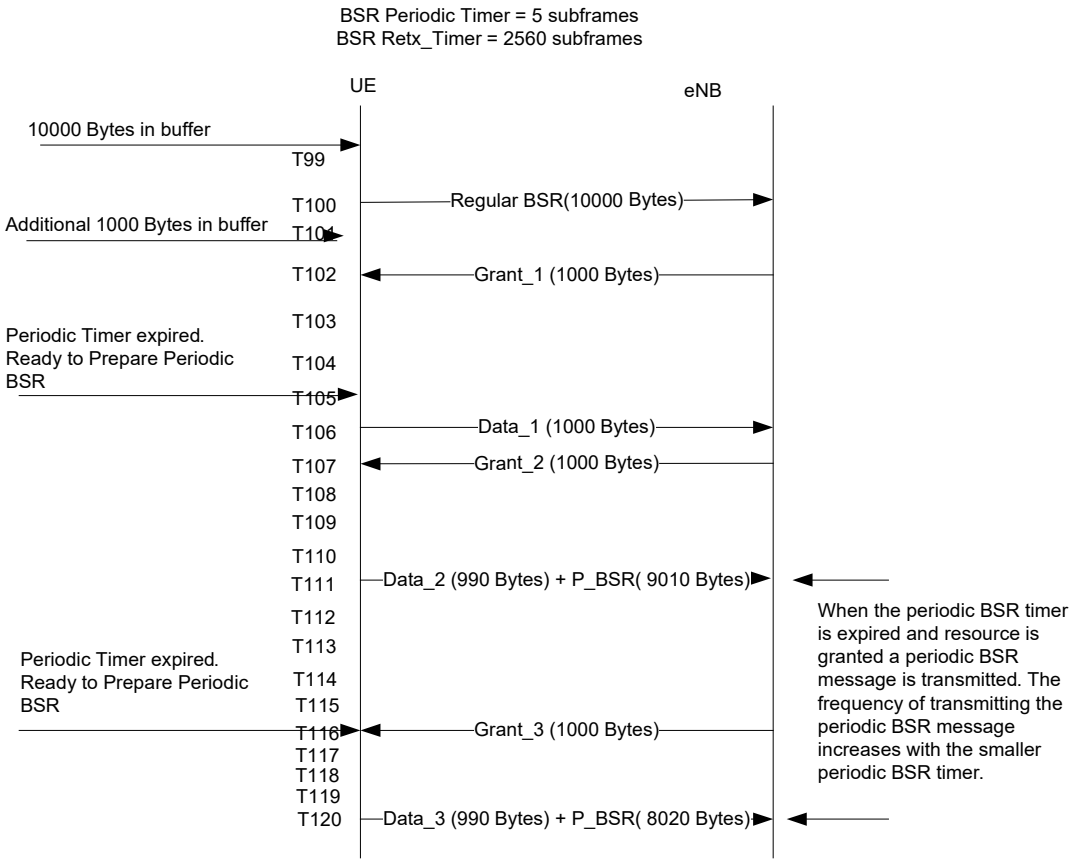


Figure 7: An example of signaling overhead increase for a normal scenario with the default protocol. Periodic BSR is in action and retx_timer is not in action.

The buffer size is now 9010 bytes; since the buffer size is greater than the grant size and the periodic BSR timer has expired, at time T120, the UE prepares and transmits a periodic BSR message and a third data message. The periodic BSR message transmitted reports the buffer size as 8020 bytes, which is the buffer size (9010 bytes) minus the grant size (1000 bytes) plus the BSR message size (e.g., 10 bytes). The transmitted data message has a data size equal to 990 bytes, which is a size equivalent to the grant size (1000 bytes) minus the BSR message size (10 bytes). Note that when the periodic BSR timer is expired and the resource is granted, a periodic BSR message is transmitted. The periodic BSR messages are transmitted with

increasing frequency as the value for the periodic BSR timer is decreased, and these results in an increased signaling overhead.

In addition to the increase in signaling overhead, the remedy of decreasing the time value for the periodic BSR timer may still fail to overcome the packet delay and jitter performance degradation since the scenario described in Figure 6 may still occur even with the smallest periodic BSR timer (i.e., 5 ms) specified in the 3GPP document (TS 36.331). Furthermore, setting the retransmit BSR timer (e.g., "retxBSR-Timer") to the minimum value specified by the 3GPP will not solve the problem either, since the inter-packet delay access can still be higher than 320 ms, which is not acceptable for all applications. Moreover, choosing any set of timer values for the periodic BSR timer and the retransmit BSR timer fails to optimize the access delay for all applications. For the commercial LTE product point of view the periodicBSR-Timer may not be implemented because it is an optional feature in the standard. Even if it is implemented in the system, it is very unlikely to set to the smaller value because of the signaling overhead increase. For this reason the default timer value for the periodicBSR-Timer in the standard is set to infinity.

3.3 Uplink Resource Allocation Protocol Enhancement

We designed a mechanism that would allow packets to exit the waiting process as soon as the buffer size reported in the last BSR message is satisfied. We did so by tracking the total grant size since the last BSR transmission (see Figure 8). This responds to the very specific needs of all applications. As such, the packet access delay and jitter could be minimized regardless of the application type. As soon as the total grant size is equal to or greater than the reported buffer size, the UE is able to send a regular BSR without waiting for the BSR retransmission timer to expire.

Figure 8 represents an example flowchart illustrating an enhanced protocol for sending Buffer Status Report (BSR) messages from a UE. At step 1, the UE is known to have an empty buffer. At step 2, after the packet arrival from the upper layer (e.g., application layer), the UE sends a regular BSR message to an eNB which indicates the total buffer size, starts the periodic BSR

timer and starts the retransmit BSR timer. The UE then proceeds to wait for a grant message at step 3.

While waiting for a grant message, if the retransmit BSR timer expires before the grant is received, the UE loops back to step 2 and again sends a regular BSR message and restarts the periodic BSR timer and the retransmit BSR timer. Thus, the enhanced procedure will wait for a grant message for up to a period of time equivalent to the value of the retransmit BSR timer before resending a regular BSR message. While waiting for a grant message, if the grant is received before the retransmit BSR timer expires, the UE proceeds to step 4 where it is determined whether the buffer size is greater than the grant size. That is, it determines whether the amount of data available for transmission in the UL buffers of the UE is larger than the grant size.

If the buffer size is not greater than the grant size, at step 5, the UE transmits data with the data size equal to the buffer size, and terminates the periodic BSR timer and the retransmit BSR timer. After these activities, the UE loops back to step 1, at which point the buffer is once again known to be empty. If the buffer size is greater than the grant size, the UE determines whether the periodic BSR timer has expired at step 6. The value of the periodic BSR timer indicates a period of time to wait between transmissions of BSR messages. If the periodic BSR timer has not expired, the UE prepares to transmit a data message with a data size equivalent to the grant size at step 7 and proceeds to step 9 to determine whether the total grant size since the last BSR is greater than or equal to the last BSR size.

If the total grant size since the last BSR is not greater than or equal to the last BSR size then the prepared data message is transmitted and the process proceeds to step 3 to await another grant of permission to transmit additional portion(s) of the data remaining in the UL buffer at the UE.

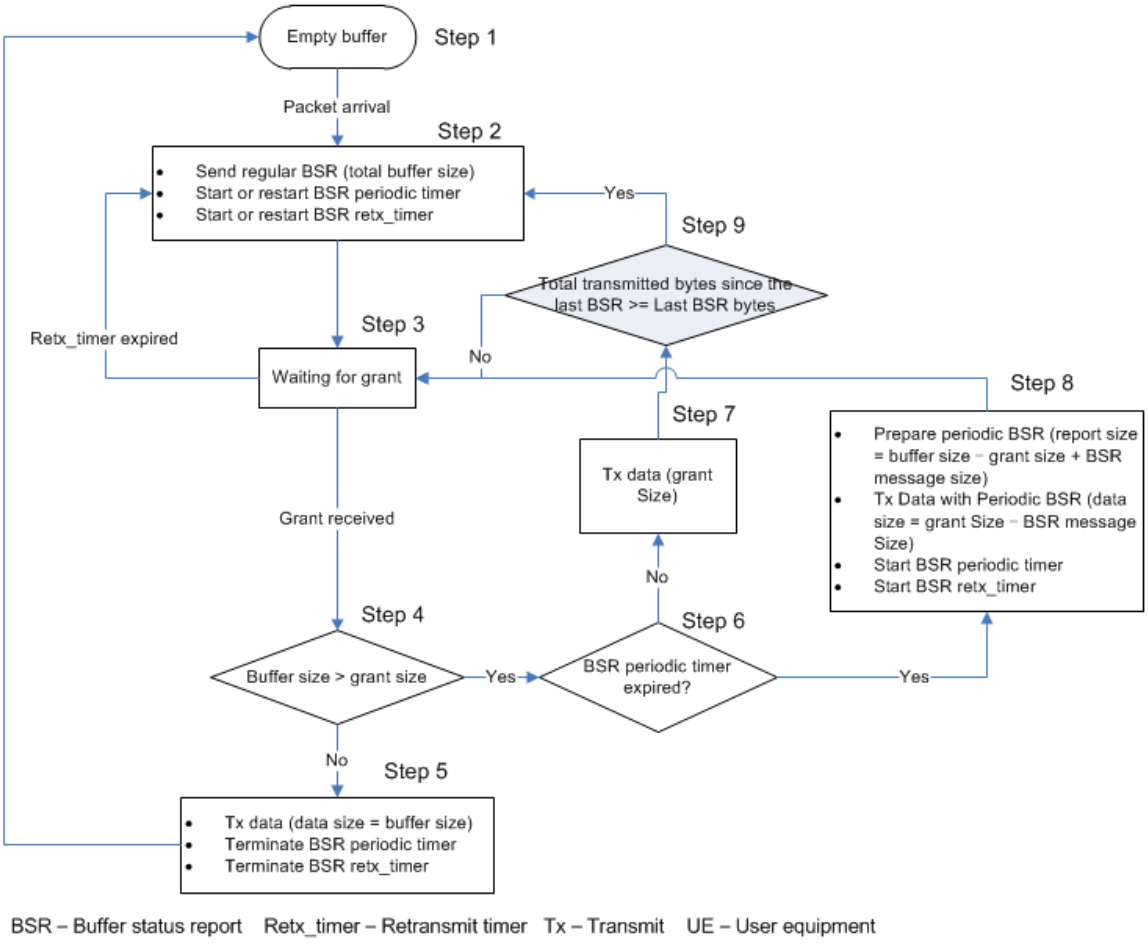


Figure 8: Enhanced UE packet transmission protocol

If the total grant size since the last BSR is greater than or equal to the last BSR size, then the process proceeds to step 2 to send a regular BSR message indicating the buffer size and restarts the periodic BSR timer and the retransmit BSR timer. Adjustment in the size of the prepared data message is also made to account for the transmission of a BSR message within this same grant.

Returning to step 6, if the periodic BSR timer has expired, at step 8, the UE prepares a periodic BSR message to report the buffer size as the buffer size minus the difference of the grant size plus the BSR message size, transmits a data message with data equivalent to the grant size

minus the BSR message size, and restarts the periodic BSR timer and the retransmit BSR timer. Thus, after receiving a grant that is not larger than the buffer size (i.e., the amount of data available for transmission in the UL buffers of the UE) or the expiration of the periodic BSR timer, the illustrated procedure sends a periodic BSR message and a portion of the data in the UL buffers of the UE to the eNB. The UE then returns to step 3 to await a grant to be able to transmit additional portion(s) of the data remaining in the UL buffers at the UE.

Thus, as soon as the total grant size is equal to or more than the reported buffer size, a UE according to the methodology herein is able to send a regular BSR message without waiting for the retransmit BSR timer to expire. This robust mechanism enables the reduction of congestion in wireless networks and increases the user quality of experience (in particular by reducing application access and jitter) without greatly increasing signaling overhead as comparing to prior solutions for reducing congestion caused by BSR messaging.

Example Packet Flow: Uplink Resource Allocation with Proposed Enhancement

Figure 9 is an example communication flow illustrating UE to eNB packet transmissions according to the enhanced protocol for sending BSR messages detailed in Figure 8 In the illustrated example of Figure 9, the periodic BSR timer is set to 10 subframes and the retransmit BSR timer is set to 2560 subframes. Time, denoted by times instances numbered T99, T100… T119 runs from the top to bottom of Figure 9. While both a periodic BSR timer and a retransmit BSR timer are utilized, activity associated with the expiration of the retransmit BSR timer is not in action/invoked in the illustrated example. At time T99, 1500 bytes are now in the buffer of the UE. At time T100, a regular BSR message is transmitted from the UE to the eNB. The BSR message is characterized as regular since it is the initial BSR message. The BSR message indicates that there are 1500 bytes in the buffer of the UE.

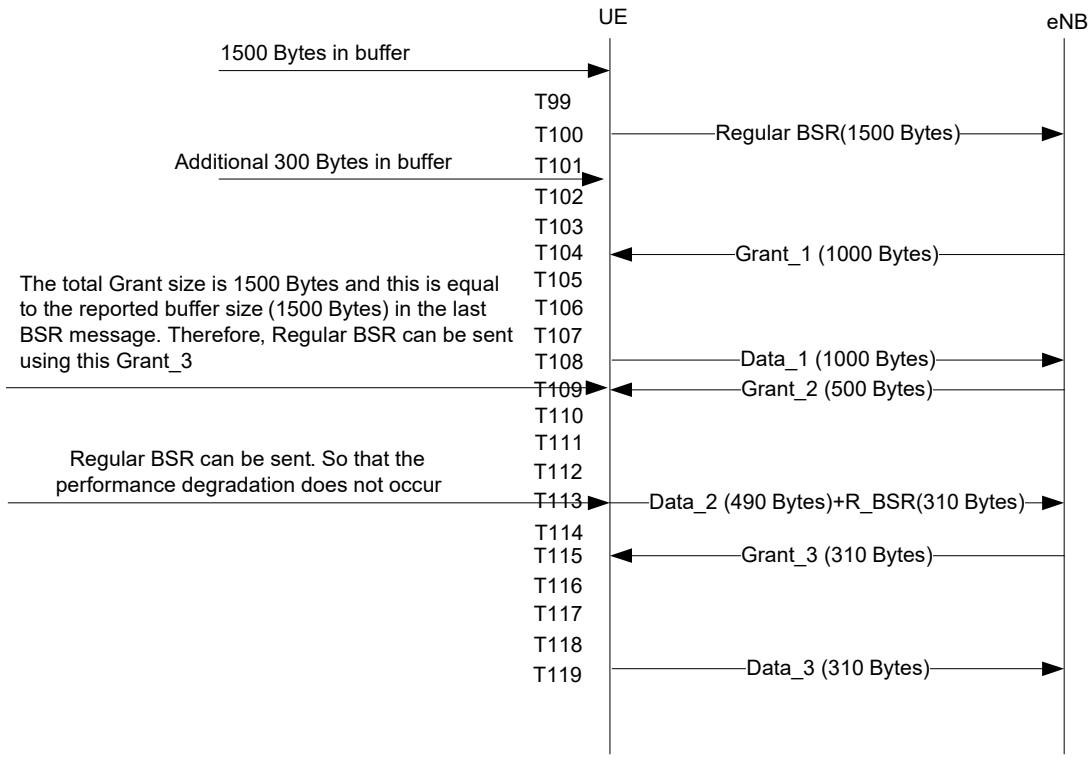


Figure 9: Example in Figure 8 with proposed solution (packet delay performance degradation has been eliminated).

At time T102, an additional 300 bytes are placed in the UL buffers of the UE. At time T104, the eNB transmits a first grant message to the UE and the UE receives the first grant for 1000 bytes from the eNB. Since the buffer size is greater than the grant size and the periodic BSR timer not yet expired (Figure 8: step 4, step 6), at time T108, the UE transmits a first data message of 1000 bytes (Figure 8: step 7). At this point in time, the total grant size since the last BSR (1000 bytes) is not greater than or equal to the last BSR size (1500 bytes) so the UE waits for the next grant (Figure 8: step 9, step 3). At time T109, the eNB transmits a second grant to the UE and the UE receives the second grant for 500 bytes from the eNB.

At this point, the total transmitted size including the latest grant size since the last BSR (1500 bytes) is greater than or equal to the size of last BSR (1500 bytes). Therefore, a regular BSR can be sent (Figure 8: step 9, step 2).

A regular BSR message can be sent without waiting for retransmission timer expiration so that the performance degradation is minimized and/or avoided. The size of the prepared data message is adjusted (Figure 8: step 7) to account for the transmission of a BSR message within this same grant. Accordingly, the size of the data message transmitted at T113 is adjusted for the size of the BSR message transmission (Data message = grant size (500 bytes) - BSR message size (10 bytes) = 490 bytes) and the BSR message reflects the remaining size of the UL buffers (390 bytes = UL buffer size (800 bytes) – data message size (490 bytes)).

At time T115, the eNB transmits a third grant to the UE and the UE receives the third grant for 310 bytes from the eNB. Since the buffer size is not greater than the grant size (Figure 8: step 4), at time T119, the UE transmits a third data message with a data size equivalent to the grant of 310 bytes (Figure 8: step 5).

3.4 Machine-to-Machine Traffic Performance Analysis over LTE

To further analyze the protocol parameters analyzed in the previous section, Machine-to-Machine (M2M) telemetry was used in the simulation. Each M2M application session is consist of one 50 bytes downlink packet and one 1000 bytes uplink packet via TCP connection. Each UE runs one telemetry session. The total traffic load is varied from two UEs up to 200 UEs per eNodeB. The M2M sessions start simultaneously and operate under error-free conditions over the wireless channels to create stress-test conditions on the protocol's performance. The BSR periodic timer is set to one of the values 5, 20, or 320 sub-frames. The BSR retx_timer is set to one of the values 640 or 2560 sub-frames.

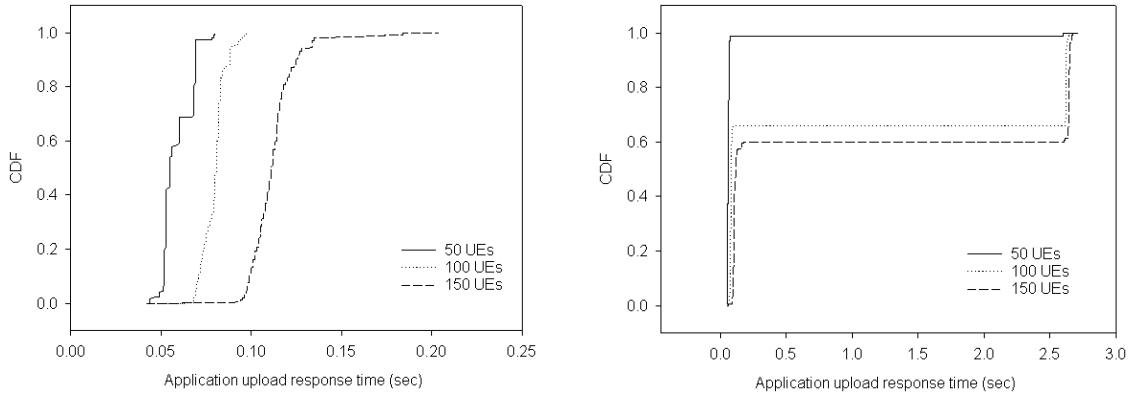
Impact of BSR Periodic_Timer and Retx_Timer:

The issue of the conventional mechanism is visible further in the Cumulative Distribution Function (CDF) of the packet delay in Figure 10 and Figure 11, which shows the strong dependency of the LTE layer packet delay for uplink traffic (UE to eNB) on the periodic timer (higher delays with 320 ms compared to the 5 ms setting) and BSR retx_timer (higher delays with 2560 ms compared to the 640 ms setting).

The only difference in configuration between Figure 10 (a) and Figure 10 (b) consists of the value of the BSR periodic_timer. 50, 100, and 150 UEs were considered in both settings. The BSR retx_timer is set to 2560 ms in both cases. With the BSR periodic_timer set to 5 ms (Figure 10.a), the application upload response time is under 150 ms in all cases. Increasing the BSR periodic_timer from 5 ms to 320 ms (Figure 10.b) dramatically increases the application upload response time to beyond the value of the BSR retx_timer. These findings are consistent with the UE buffer management described in the protocol analysis sections above (fix later).

Figure 11 illustrates the strong dependency between the BSR retx_timer (higher delays at 2560 ms compared to the 640 ms setting) and LTE layer packet delay for uplink traffic. The application response time is noticeably impacted by the BSR retx_timer, and a longer response time is observed with higher values of the BSR retx_timer. This is due to protocol behavior, and it can be improved only by understanding and tweaking the protocol parameters. These findings are consistent with the UE buffer management analysis described in the protocol analysis sections.

With the proposed enhancement mechanism the delay spike cause by the BSR retx_timer can be removed and does not require to shortening those timer values that increases the signaling traffic and only mitigates the occurrence of the timeout instead of removing them.



(a) (b)

Figure 10: CDF of application upload response time (sec): (a) BSR timer (periodic = 5 ms, retx = 2560 ms), and (b) BSR timer (periodic = 320 ms, retx = 2560 ms)

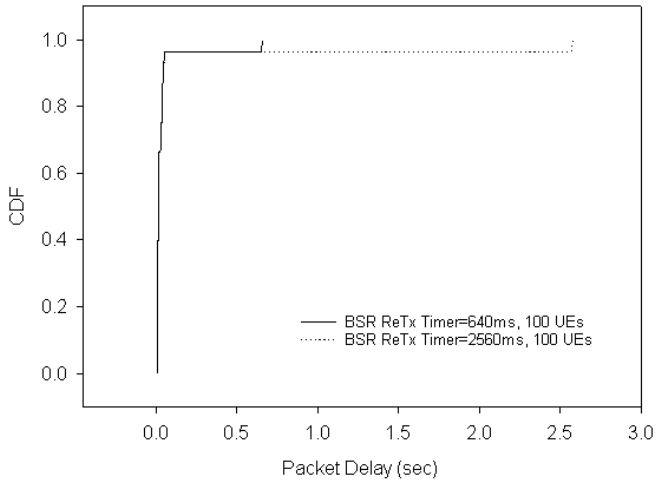
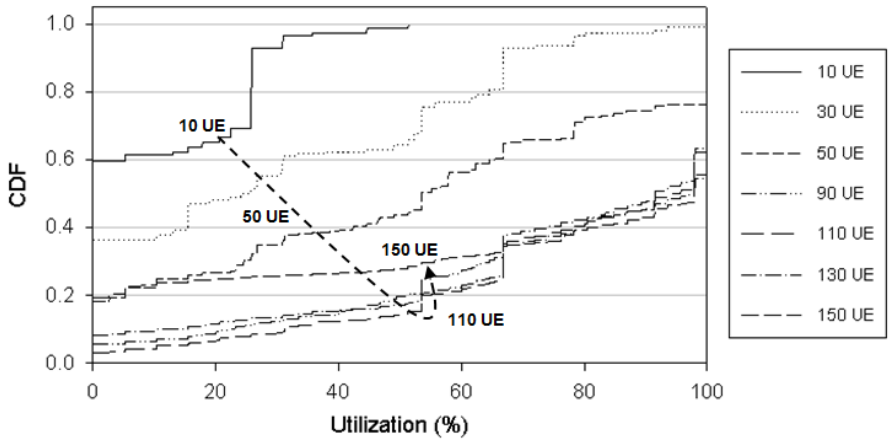


Figure 11: CDF of uplink LTE packet delay (UE to eNB). BSR timer (periodic = 20 ms, retx = 640 and 2560 ms)

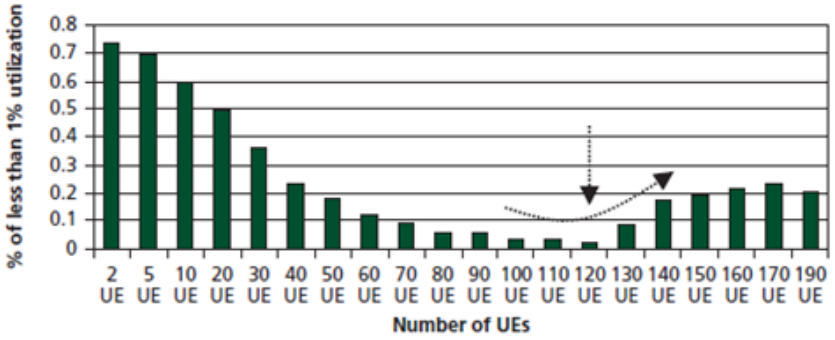
Physical Downlink Control Channel Utilization during M2M Traffic Sessions:

Figure 12 illustrates Physical Downlink Control Channel (PDCCH) utilization under different protocol settings with number of UEs from two to 190 (only selective curves are shown for readability). The BSR periodic_timer was set to 5 ms and the BSR retx_timer was set to 2560 ms. The probability of low PDCCH utilization (i.e., less than one percent) decreases when the number of UEs increases to 120 (Figure 12.b). This indicates that the system can still absorb additional incoming traffic up to a given threshold (e.g., 120 UEs). At higher loads, the probability of low PDCCH utilization (i.e., less than one percent) begins to increase approaching the value for the 50 UE scenario, and remains quite flat with the load increase as specified in these tests (Figure 12.b). This behavior can be explained by the rapidly increasing application response time when the number of UEs is higher than 120 (Figure 12.d and Figure 12.e). The higher the ratio of the increase in response time means lower traffic activity per subframe, which leads to lower PDCCH usage. What is more, the increase in application response time is deeply rooted in the interaction with the upper layer protocols such as TCP, which are modeled within the LTE application platform in conjunction with the LTE lower layer protocols and buffer management protocols.

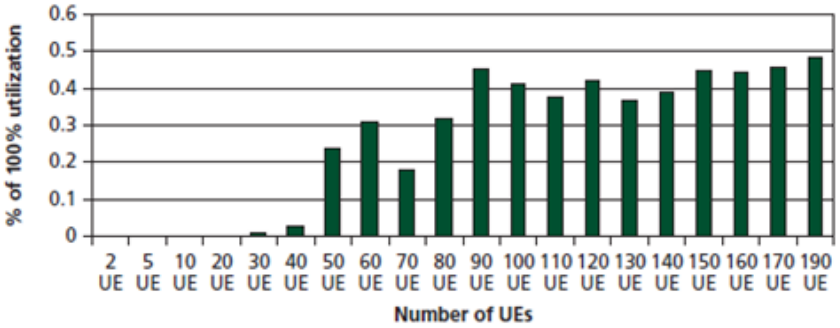
Transmission Control Protocol (TCP) flow control prevents a linear increase in system load along with the increase in traffic. Figure 12 (c) provides details on utilization of PDCCH resources on a percentage basis, and shows that there is quite a sudden increase as the load increases from 40 UEs to 50 UEs. It appears possible to push the number of highly utilized PDCCH resources up to a certain limit: for example, with 90 UEs, PDCCH is 100 percent occupied 45 percent of the time. Pushing the load further does nothing to increase PDCCH usage, which indicates that the system is congested, and that TCP flow control is regulating access.



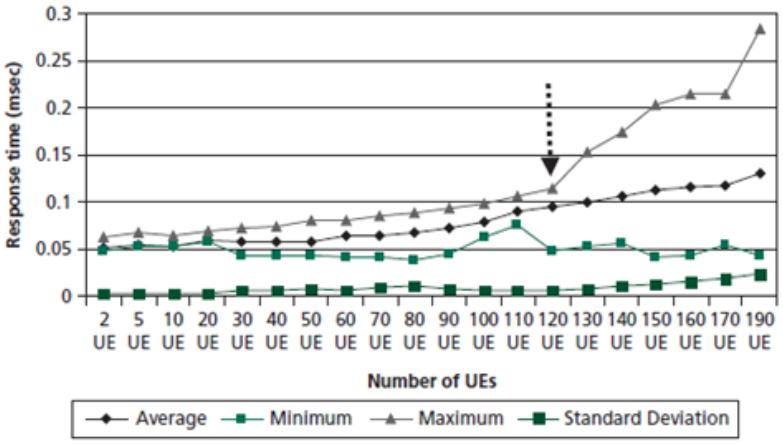
(a) PDCCH utilization during traffic session (CDF), BSR timer (periodic=5 ms, retx=2560ms).



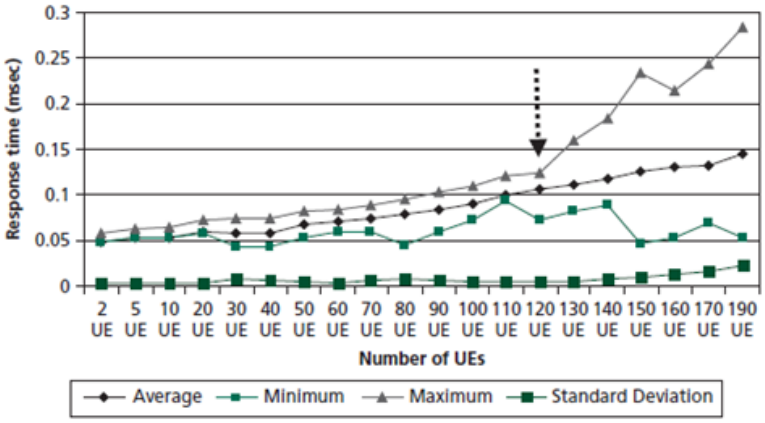
(b) % of less than 1% PDCCH utilization. BSR timer (periodic=5ms, retx=2560ms).



(c) % of less than 100% PDCCH utilization. BSR timer (periodic=5ms, retx=2560ms).



(d) Downlink response time statistics



(e) Uplink response time statistics

Figure 12: PDCCH utilization for various BSR settings. Number of UE= {2 through 190}, BSR timer (periodic=5ms, retx=2560ms)

3.5 Conclusion

We analyzed and reported the impact on the applications performance cause by the packet transmission protocols on the LTE network. We further propose an enhanced buffer management protocol [39], which enables the reduction of congestion which may cause a TCP timeout in

wireless networks and increases the user quality of experience (in particular by reducing application access and jitter) without greatly increasing signaling overhead as compared to prior solutions for reducing congestion caused by BSR messaging. Through the M2M performance analysis, we were able to validate the insights we offered into the UE buffer management. In particular, the upper layer protocols, such as the TCP, may slow down the traffic activity because of the increased packet delay caused by the PDCCH channel congestion. This causes the application response time to increase and it lowers the air resource utilization.

The PDCCH channel congestion may interfere with the BSR message exchange, so that the uplink resource allocation may be deferred beyond the BSR retransmission timer, which may cause a larger application response time. A shorter BSR periodic timer and a BSR retransmission timer could be used to mitigate the performance degradation. Such settings (within the limit permitted by the standard) would allow the network to be aware of the potential UE data backlog and to take appropriate actions when possible. However, a recognizable drawback is that with small BSR timer values, the UE may generate far many more periodic BSR transmit requests, which in turn may result in signaling overhead increase. Since the control channels are the bottleneck for thin applications, increasing the signaling overhead further is not a desirable solution. Moreover, choosing any set of timer values for the periodic BSR timer and the retransmit BSR timer fails to optimize the access delay of all applications.

The proposed protocol enables escape from a waiting state as soon as the reported buffer size in the last BSR message is satisfied, by tracking the total grant size since the last BSR transmission. As soon as the total grant size is equal to or more than the reported buffer size, a UE, according to the proposed methodology, is able to send a regular BSR message without waiting for the retransmit BSR timer to expire.

CHAPTER 4

Wireless Link Bandwidth Saving via Snoop-TCP

Due to the explosive growth of the increasing video demands, online video streaming services have been receiving more attention than ever. The growth of video streaming is on the rise. Global mobile data traffic has increased by 74% in 2015, and mobile video traffic increased to 55% of a total mobile network traffic in 2015 and 75 % of the world's mobile data traffic will be video by 2020 [21]. Especially the video streaming traffic is transmitted via TCP protocol.

We analyzed the network traffic behaviors of the two most popular HTTP-based video streaming services: YouTube and Netflix over Wi-Fi, 3G and LTE networks. We found that the video traffic behaviors rely on various environments, such as the types of user devices, multimedia applications running on the devices and network conditions. Unlike prior studies that focused more on analyzing video streaming usage patterns, popularity, duration of videos, file sizes, network traffic pattern per user device capabilities, and how clients access the video contents,[23], [30], [34], [35], [41], [54], and [63], we focus on the video packet drops at the user device.

As a result, we found that a large amount of video data is being discarded after being downloaded over the air-link. Some of the measurements show that the amount of discarded video traffic could exceed 20% of the total video content.

4.1 Mobile Video Traffic Measurement over Wireless Networks

For the video streaming and mobile application traffic analysis, we setup a test-bed to capture the live network traffic over various wireless access networks (i.e. Wi-Fi, 3G and LTE) using real user devices such as an iPhone, Android Phone, iPad and Android Tablet, as shown in Figure 13.

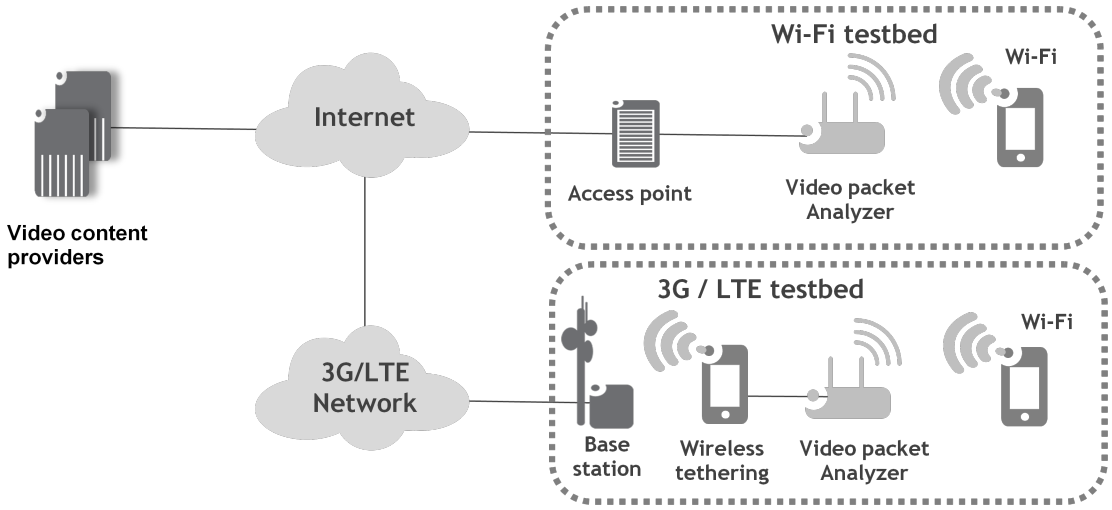


Figure 13: Testbed for the video traffic measurement

To analyze a various types of video contents, we selected twenty one diverse video titles from the most popular video content providers, in terms of genre, length and video quality. We measured and analyzed HTTP and TCP packets, while displaying the videos on the iPhone, Android and Tablets over wired and wireless networks. This was done by using a client application installed in the user device. Table 2 shows the list of the sample video titles from YouTube.

4.2 Detection of the Wasteful Video Traffic

Figure 14 is a simplified video traffic flow from one of the video streaming traffic traces. When a subscriber connects to a video streaming server, the client sends an HTTP GET message (packet flow #1). The text/html file includes a unique id of the requested video, user-agent information (OS and application types running on the device) and cookie information (i.e., recently watched video list stored on the device).

Table 2 Test video titles from YouTube

No.	Length	Title on YouTube	Type
1	7:19	E3: FarCry 3 - Demo Game play Walkthrough (HQ 360p)	Game Demo1: Action
2	10:43	Ghostbusters - First 10 Min HD HQ GeForce GTS 250	Game Demo2: Action
3	10:27	Wheelman - First 10 Min HD HQ - GF9800GT	Game Demo3: Action
4	10:53	This Corrosion (Extended 10 min) -Sisters of Mercy [HQ]	Music video
5	10:13	Minions Banana Song	Animation
6	9:10	Judas Priest - Rapid Fire (With Technical Problems)	Live Concert
7	10:24	Manchester United vs Hangzhou Greentown HQ	Sports
8	10:08	Vince Mcmahon You're fired 10 min (HQ)	WWE Show
9	10:00	The Fireplace Video - Widescreen HD Download Available!	General recording 1
10	8:11	Saturn V Launch Views - High Speed Cams 480p	General recording 2
11	1:59:00	Reign of Assassin	Action
HD 1	1:00:19	Those relaxing sounds of waves - full hd film 1080p 1h long video	Nature
HD 2	20:34	Planet Earth seen from space Full HD 1080p Original	Nature
HD 3	8:23	Power of mother nature 1080p HD	Nature
HD 4	10:29	Beautiful nature scenery(1080p HD)	Nature
HD 5	4:54	Toshiba Camelio H20 HD(1080) Nature Close-up	Nature
HD 6	6:54	Full HD 1080p nature wallpapers-Download these wallpapers ~	Nature
HD 7	13:29	Planet Earth amazing nature scenery 1080p HD	Nature
HD 8	9:59	Planet Yosemite HD 1080	Nature
HD 9	8:14	Beautiful nature scenery 2 (1080p HD)	Nature
HD 10	Not avail.	Nature 1080p HD	Nature

Then, the server responds back to the client with the list of the Content Distribution (CDN) servers which contain the video file and image servers where the client will download background images from (packet flow #2). The client transmits a set of HTTP GET messages in parallel to downloading background images. The images mostly consist of Web-page images and the snap shots of the requested video content (packet flow #3 and #4). When the client clicks the play button on the application, it starts downloading the video data from one of the CDN servers by sending HTTP GET messages (packet flow #7, TCP port 5000). Based on our measurements, the client for the targeted video provider under the study requests the entire video at once. However, the client frequently terminates the TCP connection and establishes another TCP connection, followed by another HTTP GET message to continue receiving the video content

(packet flow #11, TCP port 5001). The frequency of this behavior varies with a different OS in the user device and the network conditions. Transmitting several HTTP GET messages via new TCP connections is even noticeable when the network is busy.

Based on our analysis, we found that a significant amount of video content that had been already delivered to the mobile device through the precious air-link was discarded between the HTTP GET messages that were requested through a new TCP connection. This behavior can be found in Figure 14 between packet flow #7 and #15. The packet flows #8, #9 and #10 are acknowledged by the client and before receiving packet flow #12, the client transmitted another HTTP GET message via a new TCP connection with a port number 5001 (packet flow #11). Establishing a new TCP connection for the video content means that the client decided not to use the previous TCP connection anymore. Thus, the client sends a TCP reset (RST) packet to the server every time it receives a video packet via the previous TCP connection. The TCP RST packet is used to stop the server from further sending the video data through the closed TCP port and prevents the server from being left in the wait state, awaiting further transmissions. During this process, the client may still receive the video packets through the terminated TCP connection if the server already had transmitted before it noticed that the client had terminated the connection. These packets are not accepted by the client application since the TCP connection had been already terminated. In other words, the video data are simply thrown away without even being played. As a result of this behavior the packet flow #12, #13, #14, #19, #20, and #21 would be discarded by the client.

In addition to the wasted air-link bandwidth for the downlink traffic, there is unnecessary uplink TCP ACK packets and multiple uplink resource allocation related control traffics which would have not occurred if there was no wasteful downlink traffic. This uplink control traffic would consume the uplink air-link signaling resources which could cause the wireless system performance bottleneck while the wireless bandwidth for user data is still available, as described in Chapter 3.

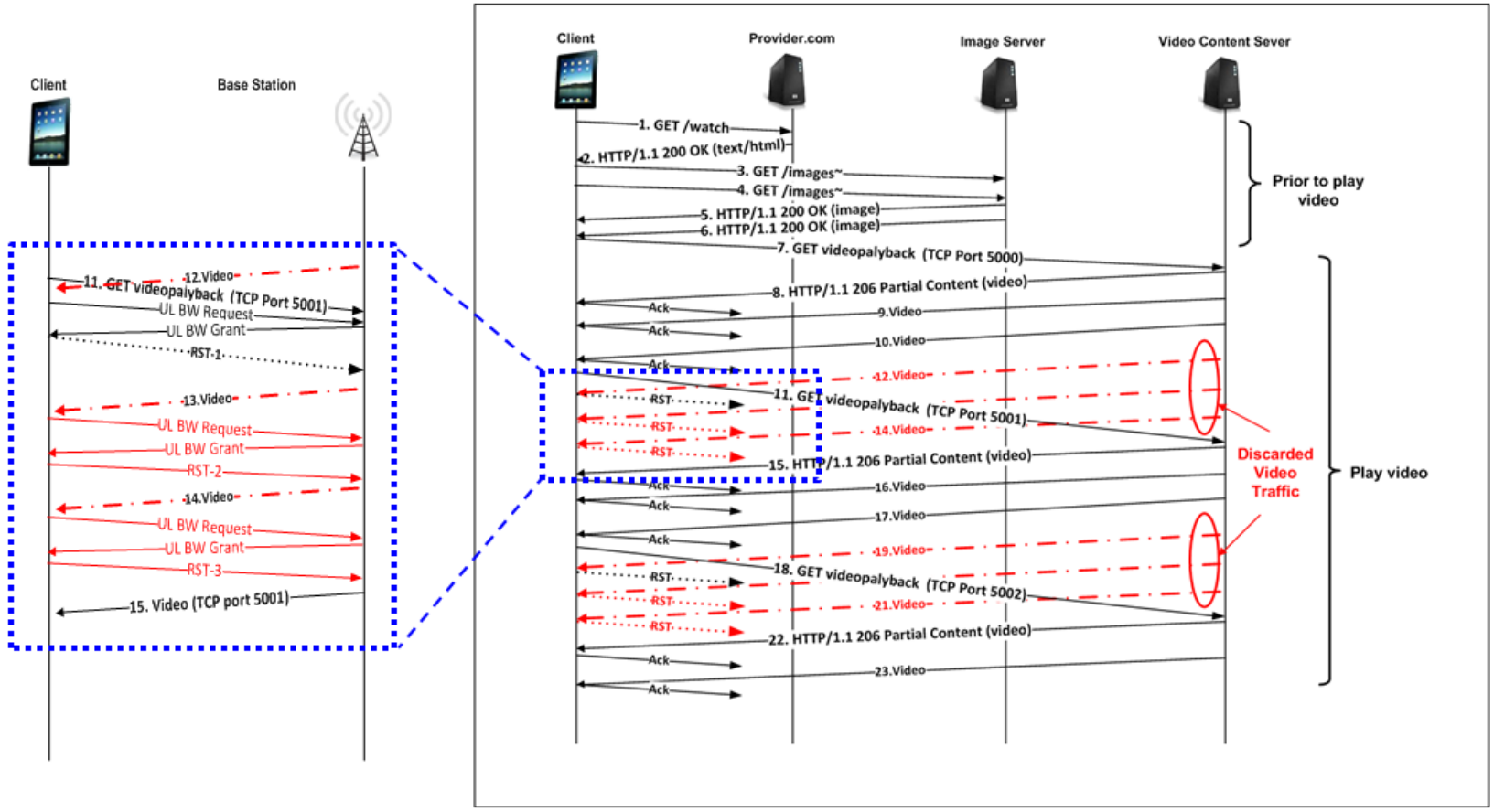


Figure 14: Video traffic flows without the TCP RST tracker mechanism

4.3 Video Traffic Analysis

4.3.1 YouTube Video Traffic Analysis

Figure 15 shows the TCP receive window size with the iPhone and Android phone. The maximum receive window size is 130 Kbytes for the iPhone and 82 Kbytes for the Android. As shown in Figure 15, the iPhone utilizes a higher maximum receive window size and mostly stays above 60 Kbytes. However, the receive window size trace from the Android is significantly different from the iPhone. It uses a lower maximum receive window size, and also repeatedly drops to 0 bytes. The consequence of this behavior would be from controlling the data transmission by the server.

As shown in Figure 16, we also measured the amount of in-flight traffic by analyzing the sender's TCP header packets. Similarly, the amount of in-flight traffic also indicates that the link utilization of the YouTube application on the iPhone is higher than the one on the Android.

For further analysis, we performed measurements over 3G and Wi-Fi access networks using the iPhone and Android phone. Figure 17,Figure 18, Figure 19, and Figure 20 show the TCP throughputs and TCP sequence number graphs using the iPhone and the Android Phone over 3G and Wi-Fi networks. The measurements clearly show the distinctive behavior between iPhone and Android. The distinction is even noticeable with a Wi-Fi network. Figure 17 and Figure 19 show that the video server pushes data to the iPhone device as long as the network bandwidth is allowed on the link without pausing transmission. On the other hand, the data transmission to the Android device periodically halts and resumes as shown in Figure 18 and Figure 20.

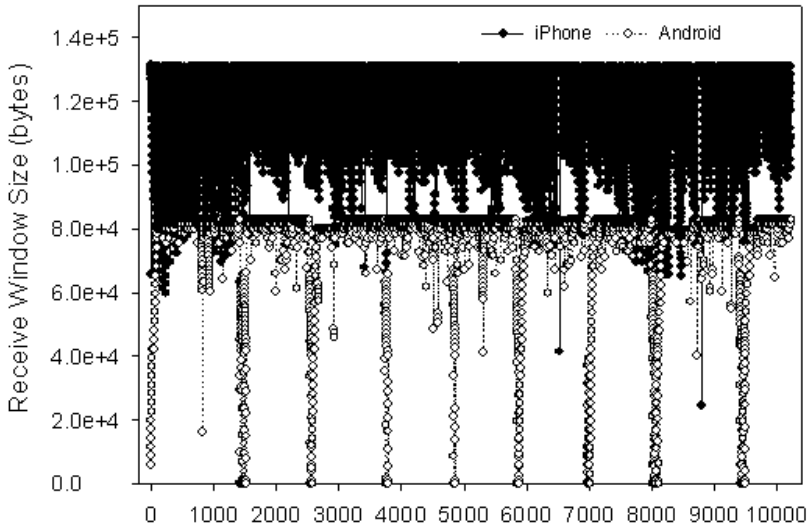


Figure 15: Receive window size (bytes) trace with the iPhone and Android phone

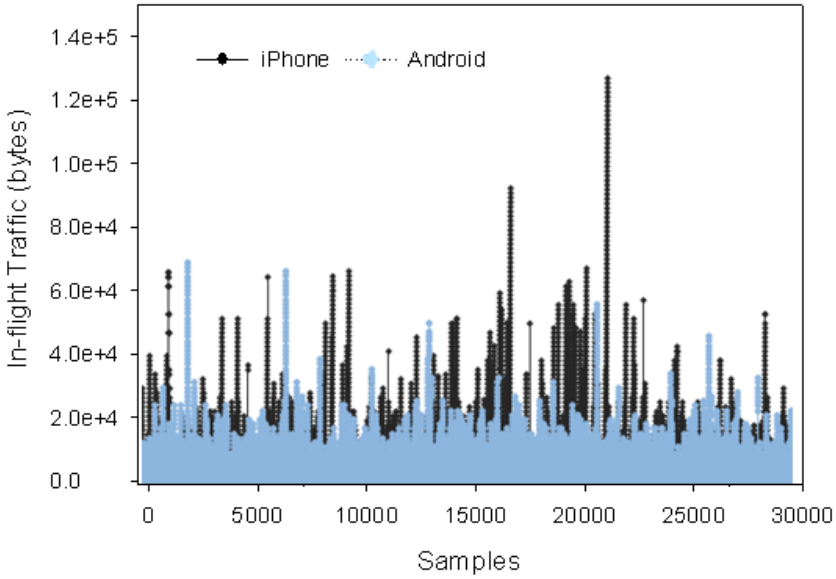
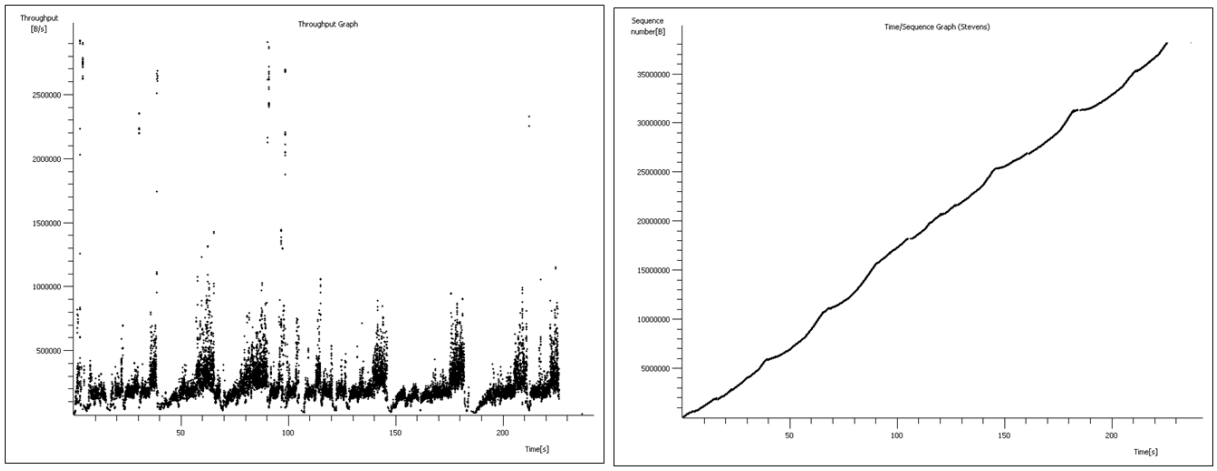


Figure 16: In-flight traffic with iPhone and Android phone

The YouTube client on the Android device tends to use only one TCP connection for the entire video streaming, unlike the behavior on the iPhone. Hence, the YouTube client on the Android device is actively controlling the video traffic transmission from the server by controlling the data consumption from the TCP receiver buffer. The buffer size at the application layer may be different or dynamically adjusted by the application, depending on the link condition or the capability of the Android device. As an example, Figure 20 shows traffic surges (represented as vertical lines in the graph) about every 50 seconds, and not much traffic activity is noticed in between these traffic surges. The client receives about 4.5 Mbytes of video content as quickly as possible by updating the receive buffer size, so that the video server can send data. Once so much data is received, the client reduces the receive buffer size to zero, so that the server stops sending data. While the client maintains the receive buffer size to zero byte, the server periodically sends TCP keep-alive messages to check if the connection is still alive. There are small amounts of data transmission within the 60 seconds interval (maximum window size multiply by three bytes) and another 4.5 Mbytes of transmission occurs.

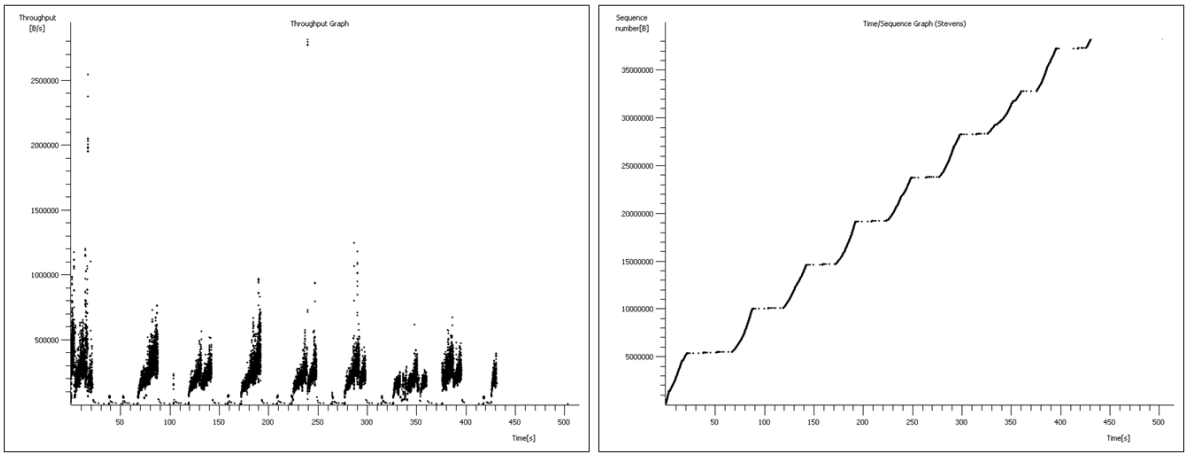
This is a different way of controlling content transmission to reduce the unnecessary transmission and system resource consumption, just in case a client decided to stop playing a video.

- Netflix: A client requests video content piece by piece every 10 seconds (see Section 4.3.2).
- YouTube with Android device: A client requests the entire video content normally, but the TCP transmission is controlled with the TCP *rwin* size report.
- YouTube with iPhone: The video traffic tends to be transmitted as fast as it can.



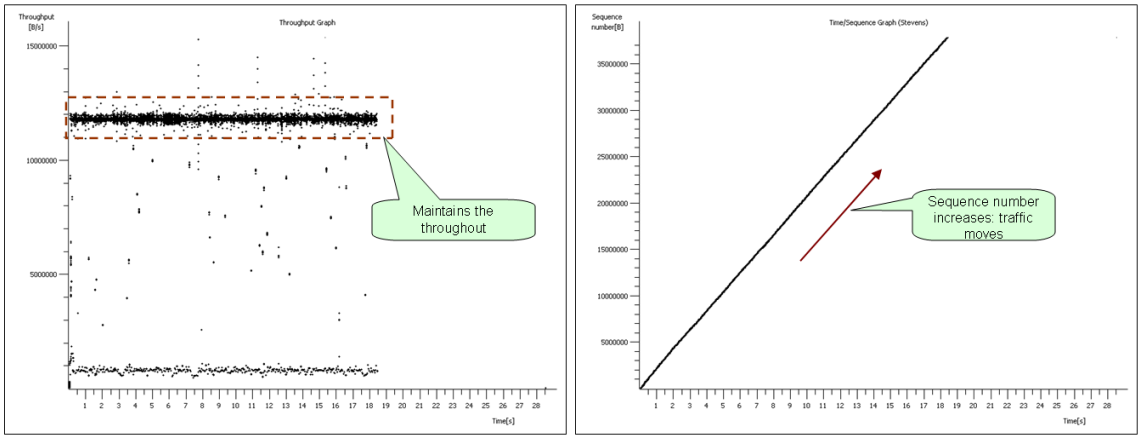
(a) (b)

Figure 17: (a) TCP throughput and (b) TCP sequence number trace using iPhone over 3G



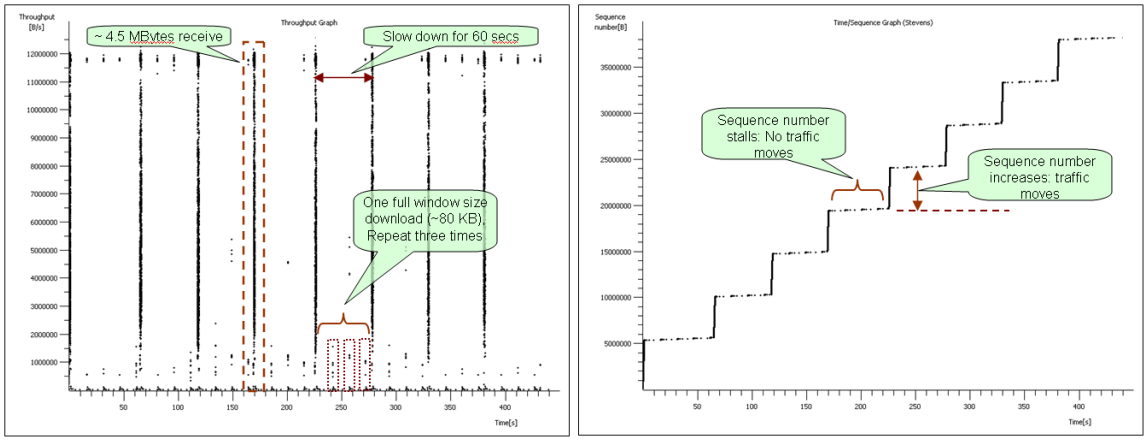
(a) (b)

Figure 18: (a) TCP throughput and (b) TCP sequence number trace using Android over 3G



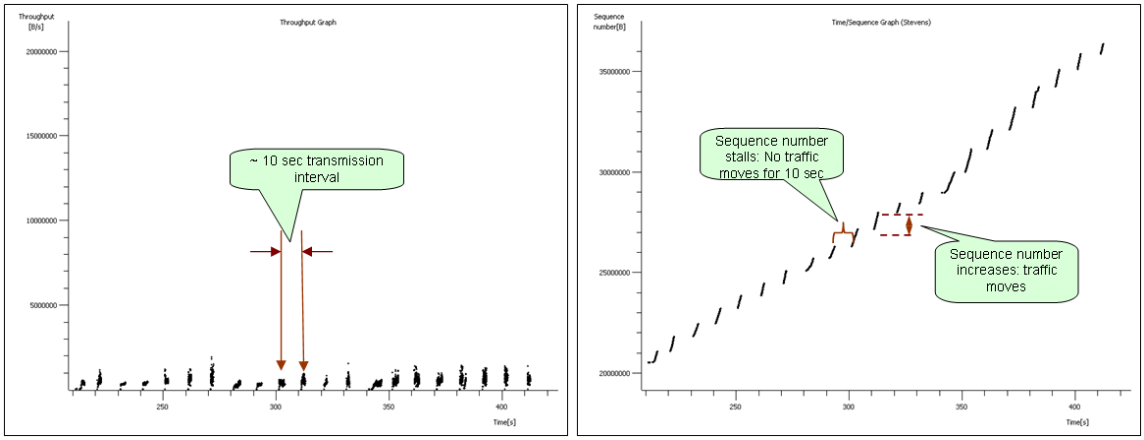
(a) (b)

Figure 19: (a) TCP throughput and (b) TCP sequence number trace using iPhone over Wi-Fi



(a) (b)

Figure 20: (a) TCP throughput and (b) TCP sequence number trace using Android over Wi-Fi



(a) (b)

Figure 21: (a) TCP throughput and (b) TCP sequence number trace using iPhone over LTE (Netflix)

4.3.2 Netflix Video Traffic Analysis

The TCP packet trace in Figure 21 was captured using the Netflix server and an iPhone over the LTE network. Since the Netflix content is encrypted and the traffic pattern with the Netflix server is rather simple compared to the YouTube, an extensive traffic analysis has not been performed.

The TCP sequence number graph in Figure 21 (a) shows that traffic bursts every 10 seconds. The TCP throughput graph also shows a 10 second interval clearly. The measurement also clearly shows that the Netflix client application use only one TCP connection for the entire video session, regardless of the device type or network conditions, unlike the YouTube as described in the Section 4.3.1.

4.3.3 HTTP GET Message Analysis

Figure 22 shows the number of HTTP GET messages in 500 second duration of video traffic using iPhone over the 3G network. We analyzed 9 different video titles from HD1 to HD4 and HD6 to HD10 that are listed in Table 2. Figure 23 represents the duration of the HTTP GET messages on average. The average duration of the HTTP GET session was as little as around 2.5 seconds up to as high as about 30 seconds. Figure 22 and Figure 23 show a different traffic behavior per video content. However, this could be caused by the network condition changes at the time of the measurements, since the measurements were not under a fully controlled test environment, but instead a public network was used. Figure 24 and Figure 25 show the CDF of the total amount of traffic delivered to the client per HTTP GET segment and the CDF of the amount of the discarded traffic by the receiver per HTTP GET segment. It shows that the amount of video traffic delivered per HTTP GET segment is about 320 Kbytes on average and about 52 Kbytes of them are discarded by the receiver (i.e. client device) on average.

Table 3 shows that iOS devices experience a much higher discard ratio compared to the Android devices. That is mainly because the YouTube video player on the iOS device sends more HTTP GET messages through new TCP connections than the video player on the Android device.

These results in the higher discard ratio compared to the case with an Android device. The discard ratio seems to be also affected by the hardware capabilities, such as memory size and display resolution, of the client device. The highest average discard ratio out of the tested client devices was observed from the iPhone 3G, and it exceeds 13% on average with a standard deviation of about 9%.

Table 3 Average and STD of discard Ratio (%) for YouTube video over LTE, 3G and Wi-Fi networks

Devices	Avg. (STD) Discard Ratio (%)
iPad 3	11.77% (1.23)
iPhone 4S	11.25% (1.22)
iPhone 3G	13.01% (9.02)
Nexus 7	1.79% (0.45)
Nexus S 4G	9.23% (1.81)

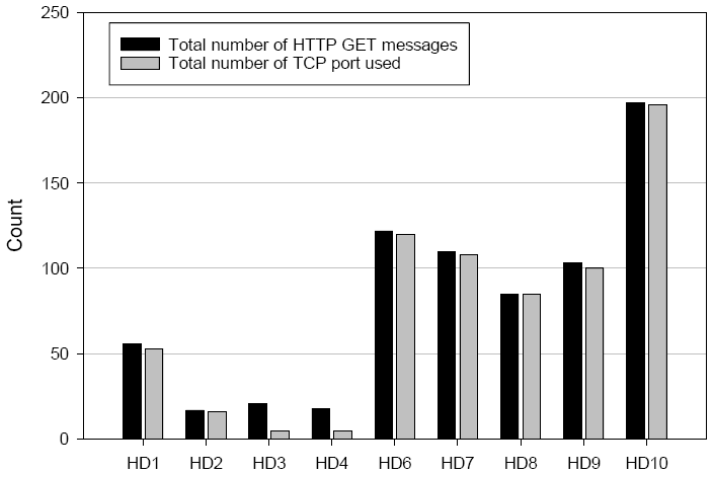


Figure 22: Number of HTTP GET messages and TCP connections in 500 sec using 3G-iPhone 4S

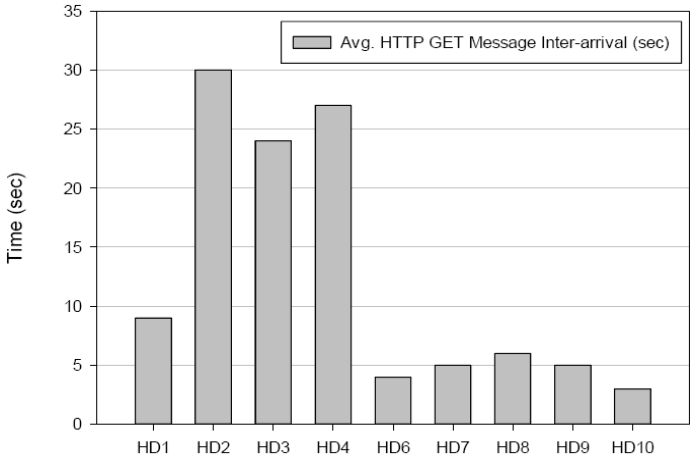


Figure 23: Average HTTP GET message inter-arrival time (sec)

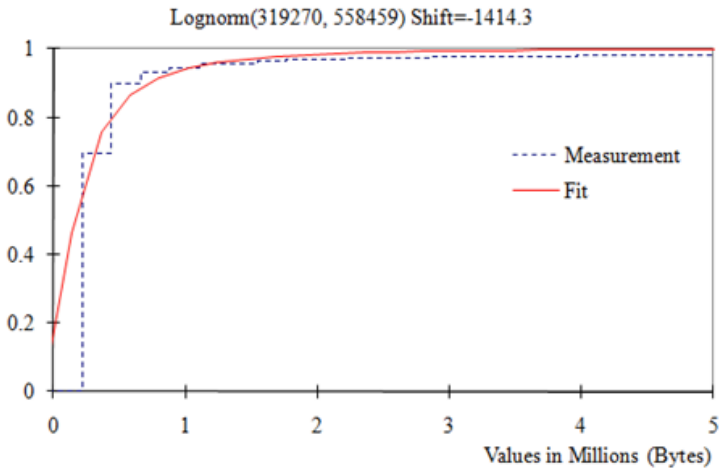


Figure 24: CDF of total bytes received per HTTP GET transaction

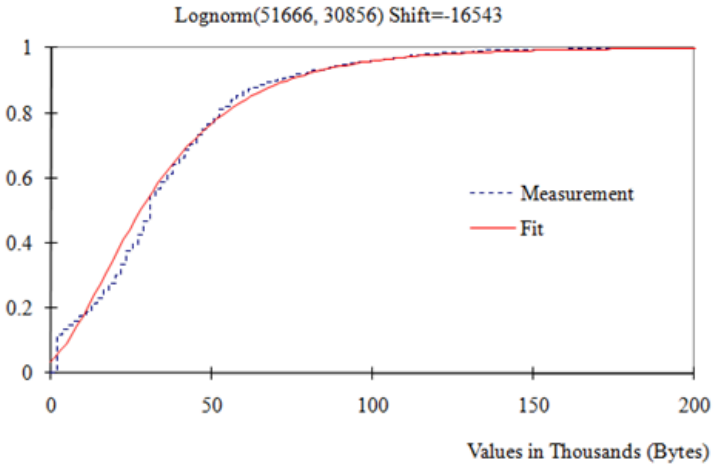


Figure 25: CDF of discarded traffic bytes per HTTP GET transaction

4.4 Known Approaches to Minimize the Problems

The best remedy for this problem is to not even generate wasteful traffic from the beginning. To accomplish this goal, the video application may establish a persistent TCP connection for the video traffic instead of establishing a sequence of TCP connections during a single video streaming session. This mechanism exists for the non-progressive video streaming, and it has even worse disadvantages in terms of the possibility of wasting video traffic, because the entire video content may be delivered to the client as fast as possible and it may be wasted when a user stops watching the video.

The progressive and HTTP Adaptive Streaming (HAS) methods, which are very common nowadays, segmentize the video files and are controlled by the application layer to support the "progressive video functionality", and to prevent transmitting excessive amounts of video traffic too early before the play time, since a user may stop playing the video and waste the early downloaded video content. Hence, even though a single TCP connection is used, the application layer may still interact with the TCP layer for the transmission. Indeed, some video content providers use a single TCP connection with an application level flow control in addition to the TCP layer flow control, to avoid generating wasteful traffic. However, the highest concern for some content providers and end users may be the application performance and convenience of the application usage, not the increased network utilization. In fact, the most popular video content providers use video traffic transmission mechanisms that generate wasteful traffic. This is completely up to the content provider's decision and there is no standard or regulation to prevent it. The important fact here is that those entities who create the issue (i.e. content providers) and those who have a keen interest to resolve it (i.e. wireless network providers & equipment vendors) are different. Those content providers may not have a concern about the network traffic being wasted; however preserving the scarce wireless network resource is a top concern for the wireless network providers.

4.5 TCP Reset Tracker

We propose a TCP Snoop based solution called TCP Reset Tracker (TCP RST Tracker) to prevent transmitting the wasteful network traffic. The proposed solution is to monitor the TCP reset (RST) message originated from the client and discard the buffered and incoming packets from the network that have the same 5-tuple TCP flow (i.e. source and destination IP addresses, source and destination port numbers, and protocol number) at the base station (see Figure 26).

When a base station receives a MAC Protocol Data Unit (PDU) from a mobile station, it checks the 5-tuple information in the received packet. This is important because if the first MAC PDU has enough information to identify the TCP session and the TCP reset information, then the system does not need to wait until the entire MAC Service Data Unit (SDU) is reassembled so that can be more efficient in terms of the operational delay. If the MAC PDU is a segment of a MAC SDU and does not have all the information to identify the 5-tuple information, then the MAC PDU should be stored until the segment is reassembled as a MAC SDU to extract the TCP information. Once the 5-tuple information is identified, the system checks if the received segment includes the TCP reset information. If it is not a TCP acknowledgement packet, then no action is needed by the proposed "TCP RST Tracker" which monitors a TCP reset message. If a packet includes TCP Reset information, then the TCP RST Tracker searches the downlink buffer for the specific mobile station. All packets that belong to the same TCP session will be discarded from the buffer. This monitoring may continue for a few seconds because some packets may still be coming to the base station, and to ensure that subsequent packets will not be transmitted over the air. The duration of the packet searching process should be configurable with a timer since the amount of in-flight packets varies depending on the network condition, TCP configuration parameters, etc. The video content packet flow #12 in the packet flow diagram, Figure 27, arrives at the mobile station after transmitting a new HTTP GET message via a new TCP connection. Thus, the mobile station sends a TCP Reset message to the content server notifying that the previous TCP

connection has been aborted. As soon as the first TCP Reset message is detected by the base station, the base station actively monitors and discards the packets arriving from the network

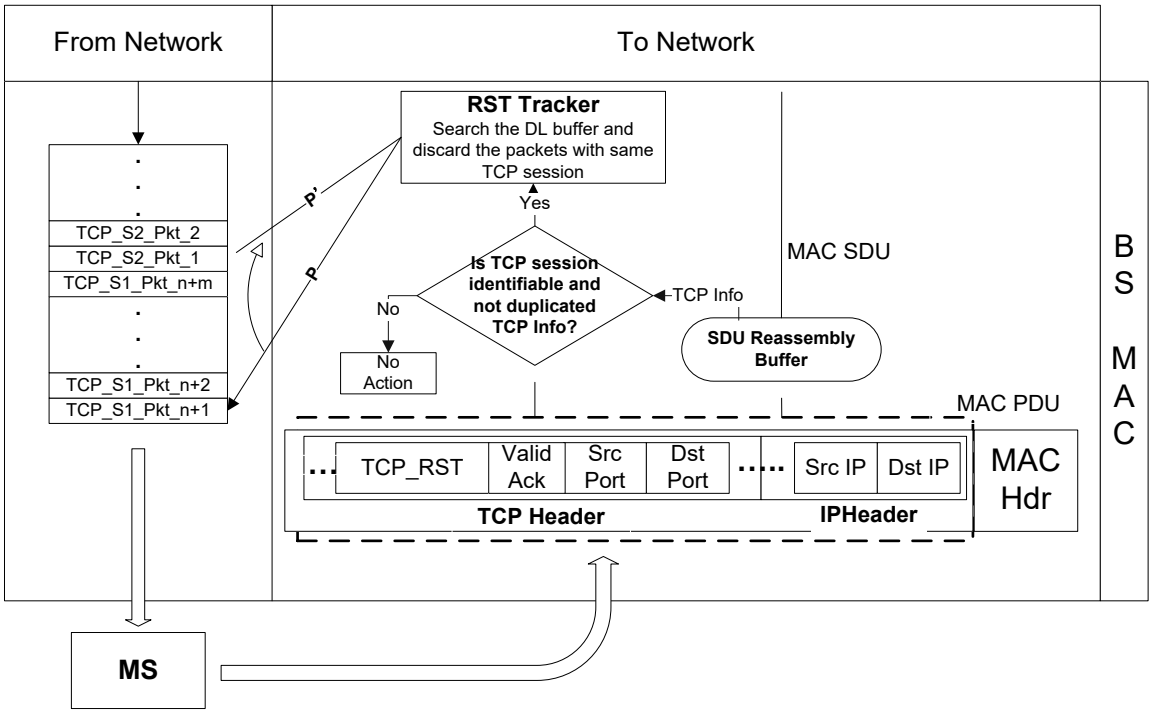


Figure 26: TCP RST tracker at the base station

through the aborted TCP connection. So the video traffic #13' and #14' will be discarded by the base station before being transmitted to the mobile station. As a consequence of this action, the mobile station does not need to respond to the discarded video traffic, and the packet flows in blue in the diagram will not be generated using the proposed mechanism, TCP RST tracker.

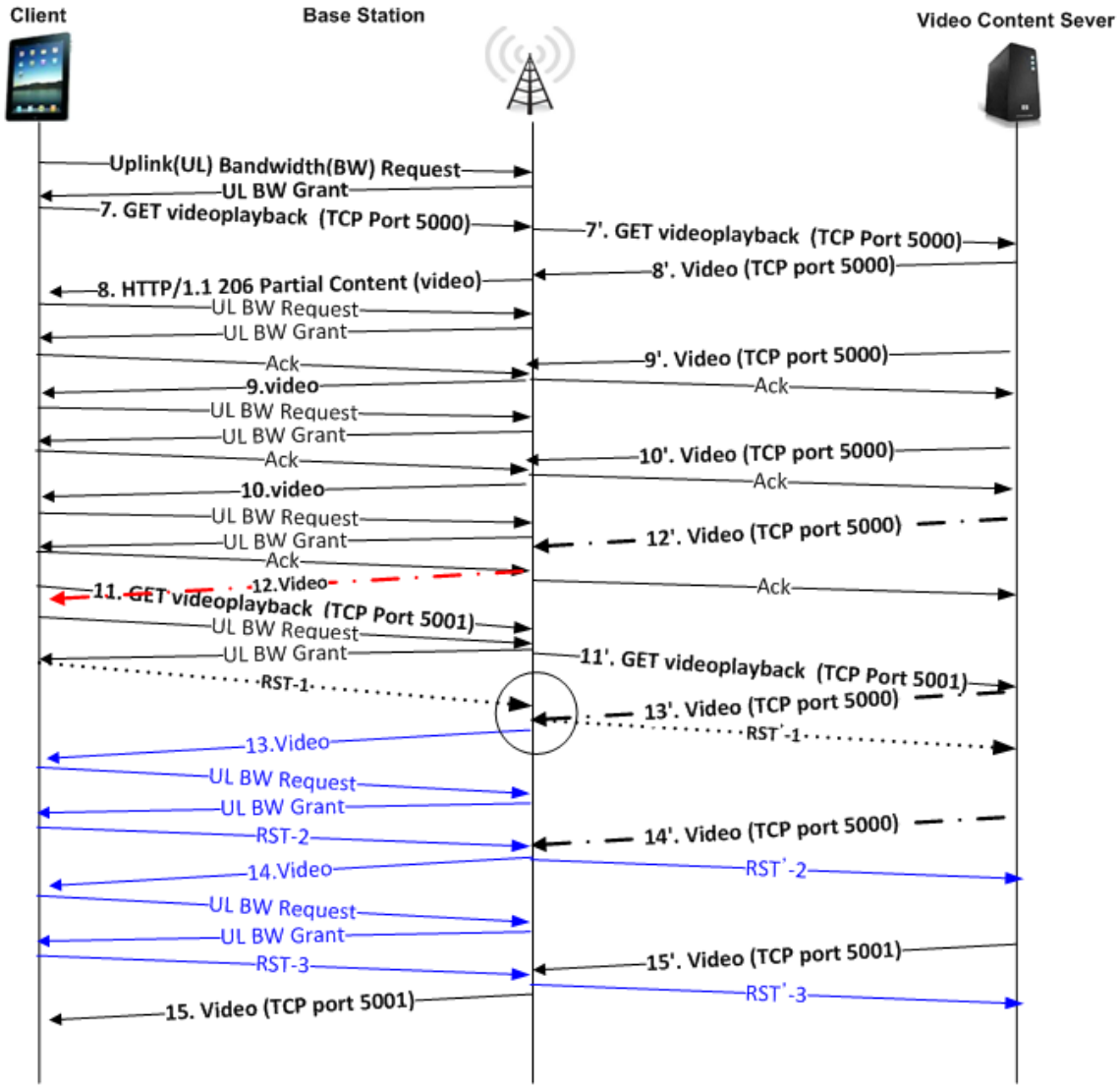


Figure 27: With TCP optimization at the base station using TCP reset message

4.6 Analysis on TCP Reset Tracker

The proposed mechanism describes a simple, yet robust mechanism to eliminate wasteful video content traffic without transmitting it over the precious air-link. The estimated amount of air-link bandwidth saving could be up to 35% and about 10% on average of the video traffic over the air-link based on the mobile video traffic usage information from the market. The proposed mechanism is handled at the base station, not at the user device. Thus, the enhancement is achievable without affecting the existing TCP or application layer. The air-link bandwidth saving would also be reflected to the billing system; hence this mechanism would be beneficial to the mobile subscribers as well, in terms of monthly data usage. The proposed solution should work for all applications which use the TCP protocol with the similar application behavior, not only for the video application.

We expect to have the following advantages with the proposed mechanism.

1. No interruption on the end-to-end connections between clients and server: The solution does not manipulate any HTTP connection originated from the client and does not generate any new messages to clients and servers in order to deal with the data.
2. Improve efficiency of bandwidth usage on air interface for both downlink and uplink: by discarding the wasteful traffic from the server to the mobile station, the proposed mechanism can also eliminate the TCP RST packet that would have been sent by the mobile station to the server. The elimination of unnecessary packets in the uplink has further benefits on top of the uplink bandwidth saving, since transmitting uplink traffic involves more control message exchanges with the base station.
3. Not affecting the application behavior: It does not change any behavior of the application. The application does not send any TCP RST message for the downloaded content, unless it receives video content from a closed TCP connection. With this solution, the number of TCP RST messages is expected to be significantly reduced.

4. This mechanism can be integrated in the existing base stations without disruption, since it could be implemented as an independent module with a little interaction with the other modules, such as mirroring the TCP information from the received packet before and after the reassembly buffer at the base station. Also there would not be any side effect on the performance since the enhancement module does not hold any part of the packet flows in the system.

4.7 Conclusion

This chapter explored and analyzed the two most popular HTTP-based video streaming services in terms of video traffic behavior in the network, while displaying the videos on mobile devices (iOS and Android) over the wireless networks (Wi-Fi, 3G and LTE). In the experiments, we identified that the network traffic behavior of watching videos on-line depend on hardware performance, software running on the devices and network conditions. The measurements show that when a client requests a video, the resolution is selected based on the device types, regardless of the OSs on the devices or access networks. We observed that a noticeable amount of video content is being discarded after the successful delivery to the client device. The content discarding occurs when a TCP connection is repeatedly terminated and established. In such cases, the video packets that arrived at the client through the terminated TCP connection are discarded. The measurements indicate that the video packet loss may exceed 35% of its complete content. This causes the misuse of the limited network resources. Considering the increasing tendency of video traffic download by the mobile users and the scarcity of the network bandwidth, understanding the application traffic behavior is crucial in order to develop an effective video delivery mechanism.

CHAPTER 5

Split TCP with End-to-End Protocol Semantics

5.1 Problems of Split-TCP Solution

As described in the TCP overview Section in Chapter 2, the performance enhancement for the applications that uses TCP protocol is even more seriously considered by the network operators and content distribution networking service providers. One of the main reasons for this is the poor TCP performance over the wireless network while wireless data rates are rapidly increasing. The TCP over the wireless network frequently underutilizes the available network link bandwidth. Because the TCP requires receiver acknowledgements for every window of data packets sent, throughput (when using a standard TCP) is inversely related to network latency or round trip time (RTT). Thus, the distance between the server and the end user becomes the true bottleneck factor in download throughput and hard to overcome, unless the server is relatively close by to the end user.

The Split-TCP proxies are already deployed in an operational world-wide network by the cloud service operators and Content Distribution Networks (CDNs), operators such as Google and Akamai [47][50][58]. A study indicates that a vanilla TCP splitting solution deployed at the satellite DCs reduces the 95th percentile of latency by as much as 43% when compared to serving queries directly from the mega DCs [50]. In addition to the cloud service and CDN operators, some of the major wireless network operators are also deploying Split-TCP mechanisms in their latest wireless network to enhance the user perspective download time. It is clear that the Split-TCP mechanism is being implemented and deployed in a commercial networks and must be enhanced to resolve the critical drawback, lack of the end-to-end semantics [62][36].

5.2 Known Approaches to Minimize the Problems

There are various types of Split-TCP mechanisms including Indirect-TCP (I-TCP) [7], Aggregate TCP (ATCP) [17], Mobile TCP (M-TCP) [13], M-TCP+ [48], and Pre-Acknowledgement Mode Split-TCP (PAM Split-TCP) [5]. Among these variations I-TCP, ATCP and PAM Split-TCP do not maintain the end-to-end semantics even though it improves TCP throughput performance. On the other hand M-TCP and M-TCP+ overcomes the drawback of the vanilla split-TCP, the lack of end-to-end semantics. M-TCP and M-TCP+ mechanisms, however, lost some fundamental benefit of Split-TCP because this mechanism does not shorten the round trip time (RTT) while it improves the TCP efficiency from the wireless link disconnection problem. Because of this pitfall, the performance improvement would be limited or does not have any improvement if the wireless network is stable.

5.3 On the Split-TCP Performance Over Real 4G LTE and 3G Wireless Networks

5.3.1 Wireless Network Environment and Measurement Scenarios

We deployed a Split-TCP system in commercial 4G LTE and 3G networks in the North American market, in order to evaluate the end-to-end TCP throughput performance and the perceived end user experience. The network is characterized by seven independent geographical regions (i.e. R1 through R7), as listed in Table 4, and separate Split-TCP systems are deployed in each of the seven regions; this table summarizes the combination of the measurement scenarios considered for this analysis. Since the network delay (RTT_{e2e}) observed during file downloads depends on the relative location of servers and clients, we have selected two extreme locations for the servers. The two servers, identified as S1 and S2, are more than 2000 miles apart from each other. As for the location of clients, we identified one cell site per region and collected measurements while the number of connected clients is representative for a typical use in a moderately loaded cell. For example, the peak number of connected subscribers is about or

below 100 during the measurement in the selected cell site. Furthermore, we have selected three client locations within the tested cell sites, which are referred as Near-Cell (NC) for good radio conditions, Mid-Cell (MC) for average radio conditions and Far-Cell (FC) for poor radio conditions. The guidelines for selecting these three locations are specified in Table 4 with respect to Received Signal Reference Power (RSRP) zone thresholds. The motivation for these placements is to account for the fact that the maximum available bandwidth depends on the client's radio conditions and to get a fair sampling of the perceived performance within a cell. Furthermore, we employed five different file sizes through these experiments: 0.5MB, 1MB, 5MB, 10MB and 20MB. With these conditions, we collected TCP packet traces at the network interface in the Split-TCP system toward the RAN by repeating 20 independent file download experiments for each file size, with and without TCP optimization (i.e. Split-TCP) in order to allow performance comparison. All test scenarios were executed over both 3G and 4G LTE networks.

A massive volume of packet traces has been collected from the field using the Wireshark packet capture tool, and it was followed by an in-depth analysis. Prior analysis, we first corrected some of the packet fields in the Wireshark captured files because we frequently observed that the TCP traces included invalid frame sizes and duplicate TCP port numbers, which caused tangling of multiple TCP test sessions into a single TCP session. We have developed an automated TCP PCAP analysis tool to handle the massive number of TCP sessions. The complete data process is illustrated in Figure 28: it separates out each TCP session from the input PCAP file and classifies each TCP session per radio technology, per region, per test client sites, and per file size. The detailed TCP packet information is stored in a database for further analysis and for displaying various TCP graphs.

Table 4: Measurement criteria

Two server locations	S1 and S2 (>2000 miles apart)
Seven cell sites	R1, R2, R3, R4, R5, R6, R7
Three client locations per cell site	Near Cell(NC): RSRP > -80 dBm Mid Cell(MC): -100dBm < RSRP < -80dBm Far Cell(FC): RSRP < -100 dBm
File sizes for 4G LTE	0.5MB, 1MB, 5MB, 10MB, 20MB
File sizes for 3G	0.5MB, 1MB

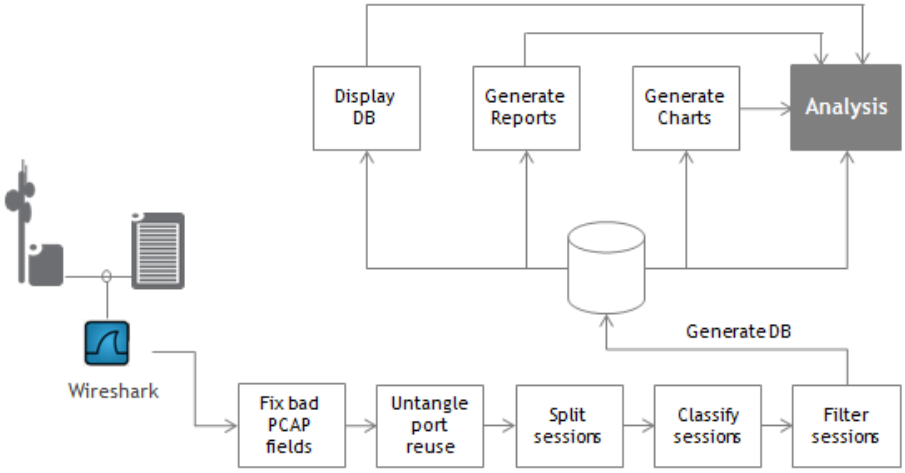


Figure 28: TCP Packet Trace Analysis Process

5.3.2 Measurement Results and Analysis

We extracted various information from the TCP packet traces including download time, download throughput, the ratio of retransmitted data bytes, RTT measurement, duplicate ACK packet ratio, number of out-or-order packets, and amount of in-flight bytes. In this paper we show the TCP throughput gain and the ratio of retransmitted data bytes. We have noticed that the throughput

results for the baseline (i.e. without TCP optimization) vary widely for some test sites, and in particular for the FC locations. This variability is likely caused by the fluctuation in the network condition. Since the measurements were done over an uncontrolled live network, some of the performance gain comparisons may not be sufficiently reliable with a small number of measurements.

Figure 29 shows the TCP throughput performance gain over the LTE network, contrasting the performance with and without TCP optimization. The percentage values in the associated tables are obtained from averaging across 20 repetitive measurements performed per measurement scenario according to Table 1. The seven measurement regions are indexed from R1 through R7 and the test client locations (i.e. NC, MC, and FC) are clustered together. Figure 29 (a) shows the measurement results using server S1 and Figure 29 (b) shows the results using server S2. The throughput gain range is widely spread and it was as high as 228%,which means that the TCP optimization provides throughputs that are more than three times faster over the baseline performance (with no TCP optimization). Because of the wide range of performance gains, we set two criteria to calculate an aggregated weighted throughput gain per measurement region:

- Weighted contributions per file size: 45% (file sizes <= 1 MB), 30% (5 MB file size), and 25% (file sizes >= 10MB)
- Weighted contributions per RF conditions: 25% (NC), 30% (MC), and 45%(FC)

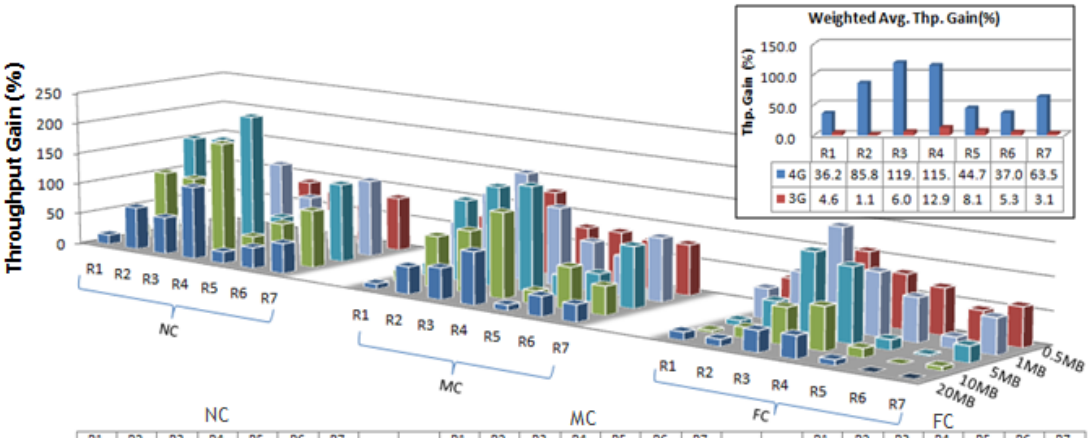
The weighted average graphs in Figure 29 (a) are based on these weighted average criteria. These graphs show the average throughput gains for the seven regions over both 4G LTE and 3G networks, even though the detailed measurement results over 3G network are not shown in this paper. Referring to the server S1, the region R3 shows about 120% of throughput gain, while the overall weighted average throughput gain across all the seven regions is about 72% over the LTE network and about 6% over the 3G network. Similar information is displayed in Figure 29 (b) for the server S2: the overall weighted average throughput gain across all the seven regions is about 43% over the 4G LTE network and about 3% over the 3G network. One of the reasons for

the performance gain difference between these two server locations is the relative distance between servers and clients. Five out of the total seven measurement regions are closer to the server S2 comparing to the server S1. This is described in more details later in this section.

Figure 30 (a) and (b) show the percentage of retransmitted bytes per measurement scenario using (a) server S1 and (b) server S2 over the 4G LTE network. Overall, the packet retransmission ratio is noticeably low; even 0% packet retransmission has been observed quite often, and most of the time it is below 0.5% and seldom gets larger than 1%. These results are collected from the end-to-end TCP connections which span across both wireless and wire-line networks.

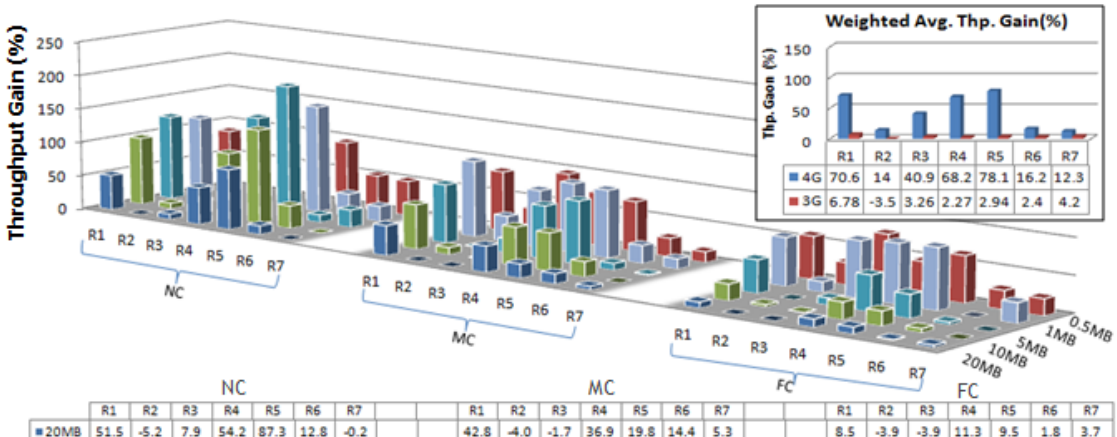
Figure 31 (a) represent the TCP sequence number graphs obtained from all individual TCP traces that are collected from the NC locations (good radio conditions) in one of the test regions while downloading a 10 MB file from the server S1 over the 4G LTE network. The group of TCP sequence graphs in the upper part of Figure 31 (a) is collected without TCP optimization while the group of TCP sequence graphs in the lower part of Figure 31 (a) is collected with TCP optimization. This indicates that the retransmission rate is zero percent, otherwise there would be red dot(s) indicating packet retransmissions and duplicate ACK packets. Furthermore, the throughput gain for this scenario is around 200%, which means that the TCP optimization renders the download speeds three times faster over the baseline.

On the other hand, the TCP traces in Figure 31 (b) corresponds to a scenario with about 0.8% of packet retransmission; in additions about 10% of ACK packets are duplicate ACKs, and are illustrated as red dots. The throughput gain for this scenario is about 33%, which is noticeably lower compared to the measurements shown in Figure 31 (a). One of the reasons for the lower performance gain is the frequent RTO expiration with and without TCP optimization.



	NC							MC							FC						
	R1	R2	R3	R4	R5	R6	R7	R1	R2	R3	R4	R5	R6	R7	R1	R2	R3	R4	R5	R6	R7
20MB	15.8	68.3	59.3	116.4	18.8	33.5	47.9	8.3	43.2	49.4	82.3	9.3	31.2	27.3	12.6	10.8	32.7	34.4	8.7	-11.0	-4.9
10MB	21.7	117.6	114.7	178.9	34.2	62.5	91.0	18.0	80.9	97.2	133.5	20.5	65.3	44.9	4.2	16.9	55.8	65.9	14.1	0.6	6.5
5MB	40.9	166.6	170.6	215.4	56.6	35.0	124.9	19.6	128.0	156.5	165.0	33.5	45.3	95.4	8.6	45.8	127.0	113.4	15.8	3.1	24.9
1MB	78.6	134.0	148.6	128.3	80.1	64.0	121.9	76.0	129.4	169.5	122.3	76.7	61.5	98.7	46.1	78.7	153.9	95.6	66.8	17.7	54.7
0.5MB	77.1	87.3	114.3	90.1	80.1	64.4	85.3	76.9	89.0	131.0	82.0	82.6	76.9	79.0	51.0	82.2	108.3	82.8	70.7	45.9	60.1

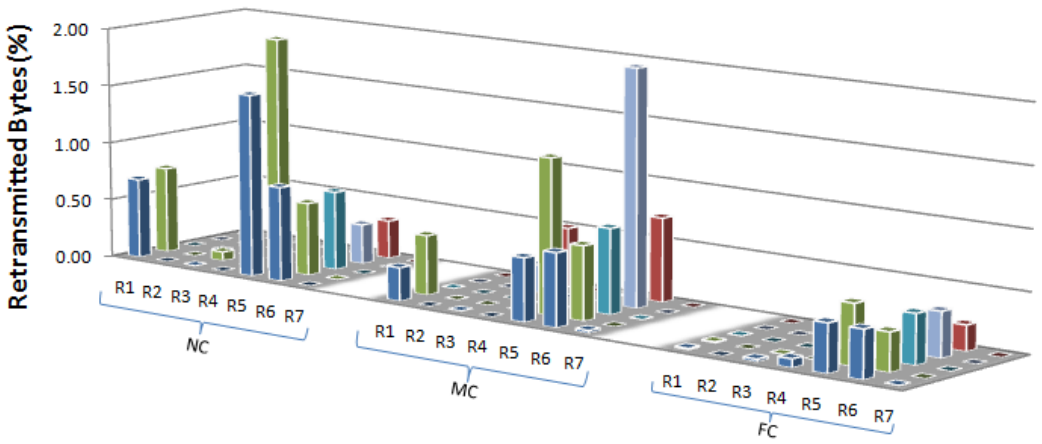
(a) 4G LTE Throughput Gain(%): TCP_Opt_OFF vs. TCP_Opt_ON (Server S1)



	NC							MC							FC						
	R1	R2	R3	R4	R5	R6	R7	R1	R2	R3	R4	R5	R6	R7	R1	R2	R3	R4	R5	R6	R7
20MB	51.5	-5.2	7.9	54.2	87.3	12.8	-0.2	42.8	-4.0	-1.7	36.9	19.8	14.4	5.3	8.5	-3.9	-3.9	11.3	9.5	1.8	3.7
10MB	99.6	8.3	18.5	96.4	137.5	33.3	0.6	63.7	10.4	1.8	54.1	53.7	21.6	-3.4	23.0	3.8	1.4	23.7	20.1	5.8	0.1
5MB	123.8	4.4	23.2	141.5	194.3	11.1	25.6	83.8	1.6	19.9	75.3	89.4	9.8	2.2	46.1	-2.7	9.9	49.5	32.2	4.1	-0.2
1MB	114.1	43.0	92.9	116.7	156.8	33.6	22.5	109.0	37.1	81.2	98.3	95.9	25.6	14.6	67.2	14.8	77.9	82.2	84.3	-3.6	27.9
0.5MB	87.1	45.1	110.6	55.9	95.4	52.5	51.1	86.0	39.0	96.5	76.2	71.4	26.6	15.3	59.9	30.7	77.7	47.9	64.4	25.9	23.1

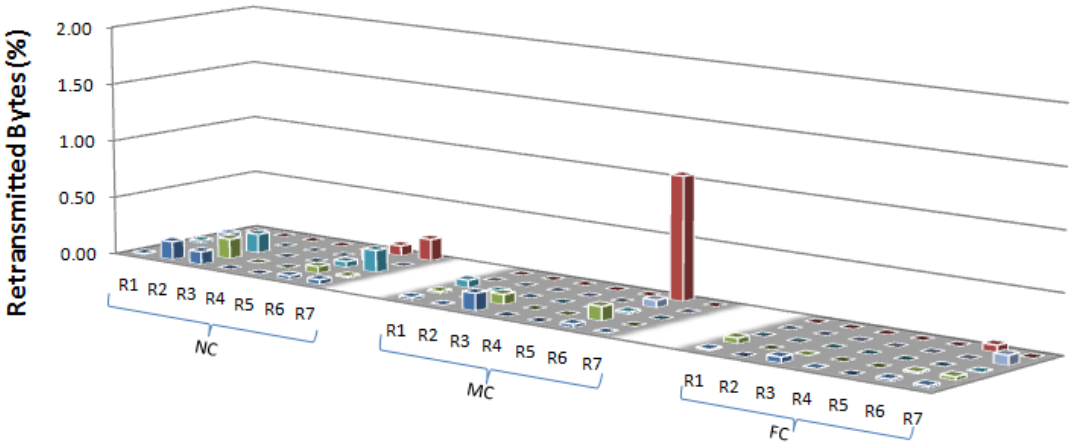
(b) 4G LTE Throughput Gain(%): TCP_Opt_OFF vs. TCP_Opt_ON (Server S2)

Figure 29: 4G LTE throughput gain(%) comparison



	R1	R2	R3	R4	R5	R6	R7			R1	R2	R3	R4	R5	R6	R7			R1	R2	R3	R4	R5	R6	R7
20MB	0.68	0.00	0.01	0.00	1.56	0.80	0.00			0.28	0.00	0.01	0.00	0.53	0.62	0.02			0.02	0.00	0.02	0.07	0.40	0.40	0.01
10MB	0.74	0.00	0.08	0.00	2.14	0.62	0.00			0.50	0.00	0.00	0.00	1.31	0.62	0.00			0.01	0.00	0.00	0.00	0.50	0.32	0.00
5MB	0.00	0.00	0.00	0.00	0.00	0.67	0.00			0.00	0.00	0.00	0.00	0.00	0.72	0.01			0.01	0.00	0.00	0.00	0.00	0.41	0.00
1MB	0.00	0.00	0.00	0.00	0.00	0.34	0.00			0.00	0.00	0.24	0.00	0.00	2.07	0.00			0.00	0.00	0.01	0.00	0.00	0.38	0.00
0.5MB	0.00	0.00	0.00	0.00	0.00	0.33	0.00			0.00	0.00	0.50	0.00	0.00	0.71	0.00			0.00	0.00	0.00	0.00	0.00	0.21	0.00

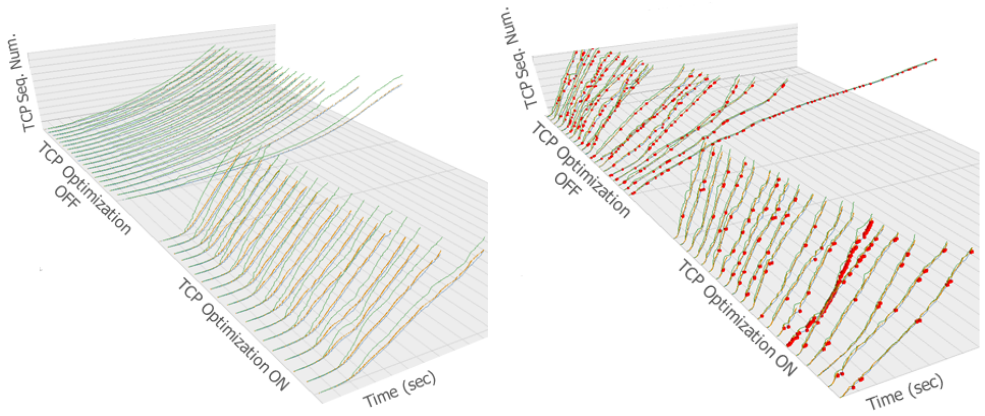
(a) Retransmitted bytes (%): TCP_Opt_OFF v.s. TCP_Opt_ON (Server S1)



	R1	R2	R3	R4	R5	R6	R7			R1	R2	R3	R4	R5	R6	R7			R1	R2	R3	R4	R5	R6	R7
20MB	0.01	0.16	0.11	0.00	0.00	0.03	0.04			0.03	0.01	0.15	0.00	0.00	0.03	0.00			0.01	0.00	0.05	0.01	0.00	0.02	0.02
10MB	0.05	0.00	0.17	0.00	0.00	0.06	0.02			0.02	0.00	0.09	0.00	0.00	0.12	0.00			0.05	0.00	0.00	0.00	0.00	0.02	0.03
5MB	0.02	0.00	0.17	0.00	0.00	0.05	0.19			0.07	0.01	0.00	0.00	0.00	0.02	0.00			0.00	0.00	0.00	0.00	0.00	0.00	0.01
1MB	0.04	0.00	0.00	0.00	0.00	0.09	0.00			0.00	0.00	0.00	0.00	0.00	0.07	0.00			0.00	0.00	0.00	0.00	0.00	0.00	0.09
0.5MB	0.00	0.00	0.00	0.00	0.00	0.09	0.19			0.00	0.00	0.00	0.00	0.00	1.06	0.00			0.00	0.00	0.00	0.00	0.00	0.06	0.00

(b) Retransmitted bytes (%): TCP_Opt_OFF v.s. TCP_Opt_ON (Server S2)

Figure 30: Retransmissino bytes ratio(%) over LTE network (a) Server S1 and (b) Server S2



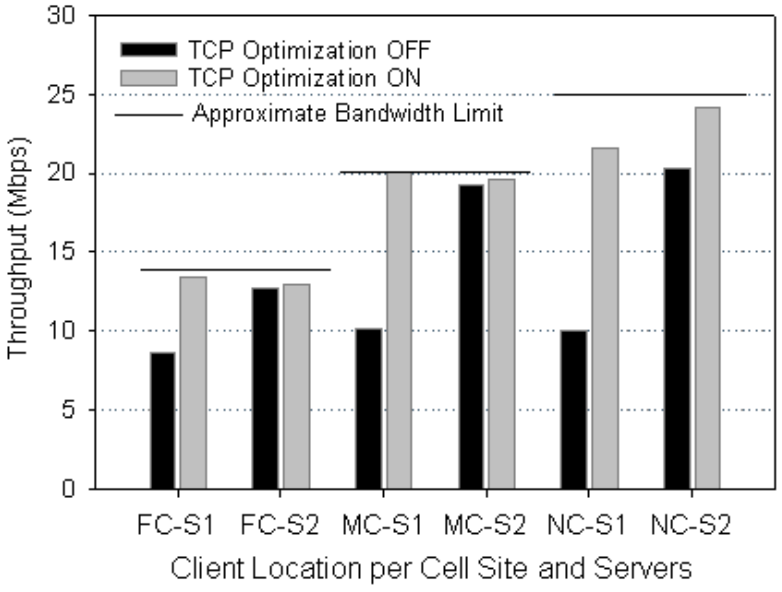
(a) (b)

Figure 31: TCP sequence number graph with (a) 0% retransmission and duplicate ACKs, and (b) 0.8% retransmission and 10% duplicate ACKs

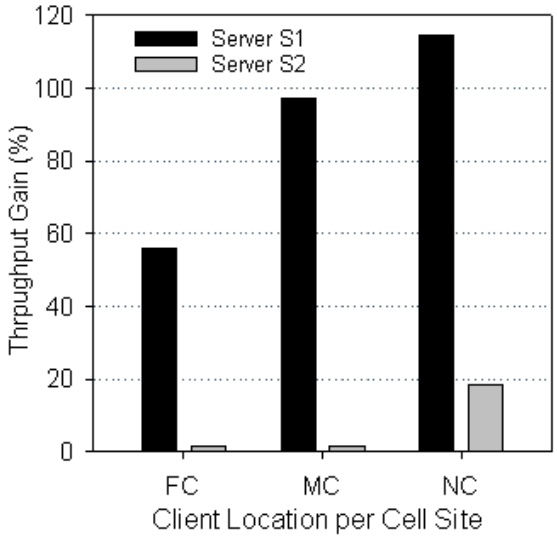
Figure 32 illustrates the distinctive performance gain differences between the server S1 and the server S2. These measurements have been collected from one of the regions that is very close to the server S2. Noticeably, the measurements taken with reference to the server S1 show high performance gains, while the measurements with reference to the server S2 show smaller gains. This is because the TCP performance gains depend on the available bandwidth in the network and network delays. If the network bandwidth is underutilized, higher TCP optimization gains are expected; on the other hand, if the network bandwidth is already saturated, there is no much room for gains through TCP optimization.

Before taking TCP throughput measurements in Figure 32, we tested the end-to-end network delays and the maximum available bandwidth using speed tests taken in the proximity of the clients, presumably for measuring the available air-link bandwidth. The average PING delay from a client to the server S2 is about 40 ms and to the server S1 is about 130 ms on average. The three speed test results identify the approximate bandwidth limits perceived at the locations labeled as NC (i.e. 25 Mbps), MC (i.e. 20 Mbps) and FC (i.e. 14 Mbps). Hence, these measurements point out to the reference upper bounds per test site, and are marked as

horizontal floor lines in Figure 32 (a). The speed test tool is available in [61], and it gauges how much inbound traffic can be handled consistently through a connection, determining its Maximum Sustainable Throughput (MST). In other words, it provides a reference to benchmark how mobile carriers perform at specific test locations. If the TCP throughput for the baseline configuration (without optimization) is lower than the reference benchmark, one gets a clear indication that the link is underutilized and hence there is room for TCP throughput gains, as enabled through TCP optimization. Referring to Figure 32 (b), a 10MB file size download at a NC location results in an approximate 117% throughput gain resulting from TCP optimization with respect to file downloads from server S1 (which is further away from client), while the gains are capped to 19% with respect to file downloads from the server S2 (which is closer to the client). This is because for the baseline configuration (in absence of TCP optimization), the actual TCP throughput measured through file downloads from the server S1 was limited to 9 Mbps, which is significantly lower than 25 Mbps measured via speed tests, while the TCP throughput measured through file downloads from the server S2 was 21 Mbps, which is much closer to the 25 Mbps benchmark. Post TCP optimization, the TCP throughput with the server S1 has increased from 9 Mbps to 21 Mbps, which translated into the 117% gain since the reference baseline for the server S1 was significantly lower in comparison to the server S2.



(a) Throughput comparison: Server S1 vs. S2



(b) Throughput gain comparison per cell sites

Figure 32: **(a) Throughput comparison: server S1 v.s S2 per test site: (b) Throughput gain (%) comparison per cell site**

The Split TCP mechanism clearly shows a significant TCP throughput improvement over a commercial 4G LTE network. On the other hand, 3G networks operate at lower data rates compared to 4G LTE networks, and hence the gains through TCP optimization are expected to

be modest. Two server locations were considered with respect to reference clients during file download operations: a far distance server and a much closer one. Regardless, the overall throughput gains enabled via TCP optimization for both server locations were about 60% in average for the 4G LTE network and about 4% in average for the 3G network. Significantly larger throughput gains (e.g. in excess of 200%) that are attributed to the TCP optimization were noticed through individual measurements. These field experiments indicate that the performance gains enabled through a Split-TCP optimization may be maximized when the RTT_{e2e} between a client and a server is large and when the network bandwidth is not saturated. Thus, we expect that the throughput gains enabled via TCP optimization may be even greater when advanced radio technologies, such as 5G, are deployed. For these reasons, Split-TCP has the potential to be widely deployed in both current and next generation networks

5.4 Enhanced Split TCP with End-to-end Protocol Semantics

It is challenging to find solutions that can all benefit from Split-TCP and maintain the end-to-end semantics at the same time. We propose a novel-mechanism, Enhanced Split-TCP (ES-TCP) that can be used with the current TCP implementation without modification.

The Enhanced Split-TCP (ES-TCP) host (EH) is located along the path between a mobile host (MH) and a far host (FH), and the location should be a single point where all TCP packets must pass through to reach the MH and FH. In case of an LTE network, PGW would be a sufficient candidate. The TCP protocol at the FH or MH does not need to be modified to use ES-TCP. Thus, the description for the ES-TCP mechanism in this Section is focused on the EH only and for the simplicity download from the FS to MH through the EH is assumed. EH consists of the EH receiver, which receives packets from the FH sender on behalf of the MH and the EH sender, which transmits the received packets to the MH on behalf of the FH as depicted in Figure 33.

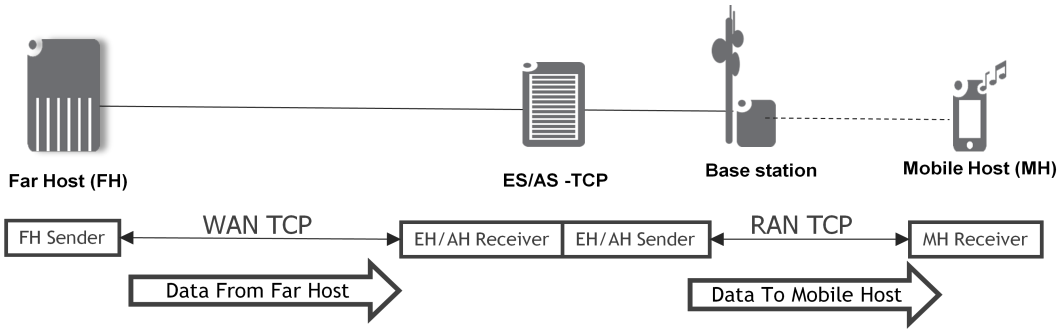


Figure 33: ES-TCP and AS-TCP host location

5.4.1 ES-TCP Operation

ES-TCP Host (EH) which operates with the ES-TCP sends a TCP ACK packet to the sender (FH) as a typical TCP does, but with the following rules:

- The sequence number in the ACK packet is K bytes less than a typical ACK sequence number calculation (i.e. ACK_seq_num = seqNumber + data length – K bytes). The byte count (K) for reservation should be at least 2 bytes to work properly. So, K=2 is assumed throughout this document.
- 1 byte out of the reserved K (= 2) bytes is to freeze or unfreeze the FH TCP
- The last 1 byte out of the reserved 2 bytes is to maintain the end-to-end semantics between the FH and the MH. The last byte is acknowledged back to the FH after it is received by the MH.
- When the buffer of the EH is empty and the last ACK packet is received by the EH from MH, the final ACK_seq_num is transmitted to the FH so that the FH may terminate the TCP connection.
- The *cwnd* for the WAN side TCP connection (between the FH and the EH) can grow independent of the RAN side TCP connection.

The received data bytes from the FH are transmitted by the EH transmitter to an MH as a typical TCP does. Since a TCP_{RAN} connection (between the EH and MH) is independent to the TCP_{WAN},

the air-link issue in the TCP_{RAN} connection is isolated in the RAN and does not propagate to the TCP_{WAN} connection.

Figure 34 depicts a high level operational flow diagram for ES-TCP at the EH, which is different from a typical TCP protocol. The ES-TCP_{WAN} part is responsible for communicating with the FH, and the ES-TCP_{RAN} part is responsible for communicating with the MH. Since the TCP protocol is not modified at either the FH or at the MH, the FH and MH components are not shown in this diagram.

The ES_Rx_0 in The ES_Rx_0 is a main state that receives a packet from the FH and is triggered by the EH_Tx when it receives an ACK packet from the MH. The only explicit communication between the ES-TCP_{WAN} and the ES-TCP_{RAN} is when an ACK packet is received from the MH. When a data packet is received by the EH_Rx from the FH, it is stored in a buffer which is shared with the EH_Tx, as depicted in the slide window diagram in Figure 35).

After both TCP sessions (FH to EH and EH to MH) are established, the FH starts transmitting packets to the MH via the EH. There are five key operations which are described below:

1. When new data bytes (i.e. packet) are received from FH, EH_Rx checks the length of the new data bytes. If the sum of this new data length and the reserved ACK bytes which were received earlier but not yet acknowledged is smaller than 3 bytes, then it simply stores the received data in a buffer without send an ACK packet to FH. The small packets will be acknowledged by MH. This means that the ES-TCP optimization will not be applied for the continuous small packets (smaller than 3 bytes data length) transmission because the main purpose of small TCP packet transmissions is not to achieve high transmission throughputs.
2. If the sum of the new data length and the reserved ACK bytes which were received earlier but not yet acknowledged is larger than or equal to 3 bytes, then send an acknowledgement with a sequence number that is 2 byte less than the actual expected next sequence number. This

is to purposely not acknowledge the last 2 bytes for later use. This is indicated as "Received but not yet Acknowledged (2 bytes)" in Figure 35. One of the main reasons is to preserve the end-to-end semantics by not acknowledging all the received data until MH receives all data bytes. When an ACK packet is to be sent to FH, *rwin* size should be included as a typical TCP would do. Depending on the *rwin* size, a certain operation is selected for the next step.

3. If *rwin* >0, then estimate TCP retransmission timeout (RTO) for the reserved ACK bytes which FH TCP would be ticking for retransmission if acknowledgement for the reserved ACK sequence number is not received before RTO is expired. RTO expiration may occur if there is not a new packet from FH and MH is not sending acknowledgement for the reserved ACK sequence number during the timer activity. If RTO expires at FH, then FH will retransmit for the reserved ACK sequence number which was already received by EH. To prevent the unnecessary retransmissions, EH runs an estimated RTO for FH which should be smaller than the actual RTO at FH. If estimated RTO expires before receiving retransmission from FH, then EH sends an acknowledgement packet that acknowledges 1 byte out of the reserved 2 bytes with *rwin* size equal to 0. This acknowledgement packet freezes FH to prevent retransmission. If, however, a retransmission is received from FH, then EH immediately sends an acknowledgement packet that acknowledges 1 byte out the reserved 2 bytes with *rwin* size equal to 0. This acknowledgement packet freezes FH to prevent further retransmissions. Since the last 1 byte is not acknowledged yet, the end-to-end semantics is still maintained.
4. If *rwin* = 0, then an ACK will be sent and it freezes FH and stop transmission until EH unfreezes with window update. When EH receive buffer becomes available the EH sends an acknowledgement packet that acknowledges 1 byte out of the reserved 2 bytes with the updated *rwin* size. This acknowledgement packet with window update information unfreezes FH to resume packet transmission. Since the last 1 byte is not acknowledged yet, the end-to-end semantics is still maintained.

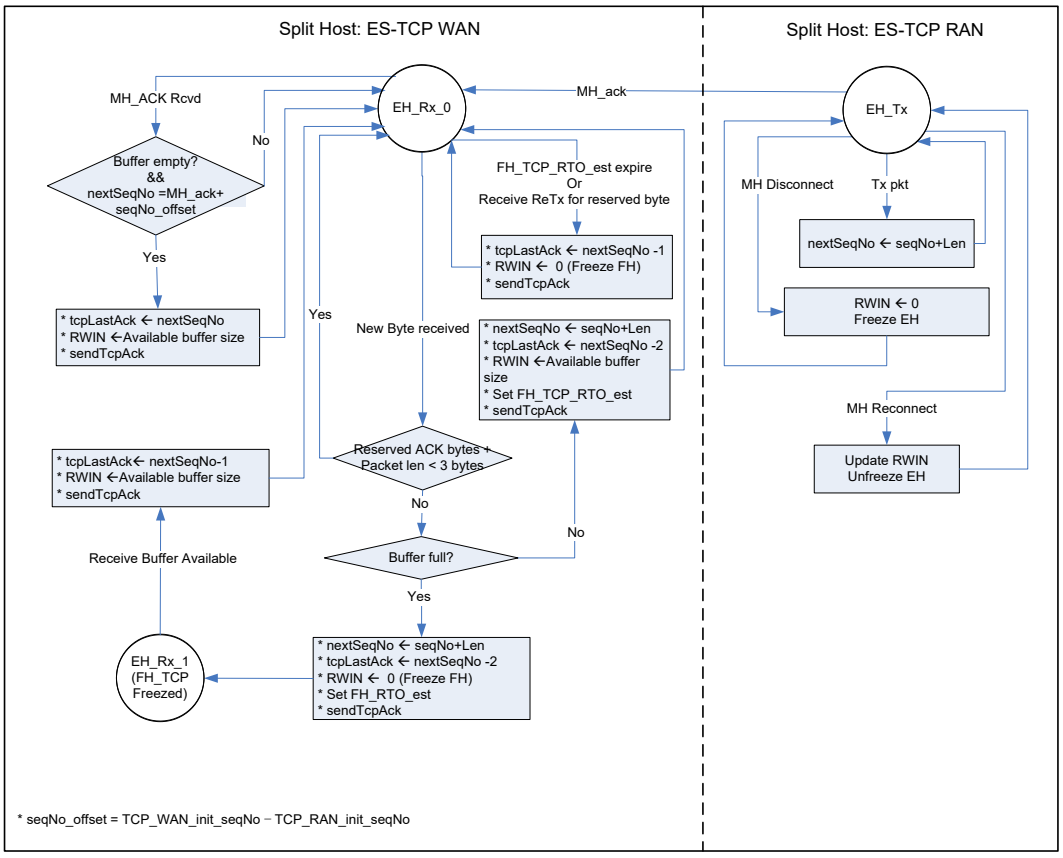


Figure 34: Operation flow diagram for ES-TCP receiver and ES-TCP sender

5. At anytime an ACK from MH is received, EH checks if the ACK confirms that the last byte is received by MH and there is no pending data bytes to be transmitted to MH, then EH needs to send ACK for the last reserved ACK byte to FH, which let the FH know that all data bytes have been received by MH.

Figure 35 illustrates a conceptual diagram of the sliding window for both the congestion window and receive window that is using as a shared buffer at EH. The packets from FH are added to the left side of the diagram and transmitted packets to MH are illustrated on the right side of the diagram. The values in the buffer space represent sequence numbers.

For data being transmitted to MH, there are four transmit categories, as described from the right most diagrams (see Figure 35):

1. Transmit Category #1: Sent And Acknowledged from MH (i.e., byte range: 0 to 19)
2. Transmit Category #2: Sent But Not Yet Acknowledged from MH (i.e., byte range: 20-29)
3. Transmit Category #3: Not Yet Sent but Recipient Is Ready. This represents the *rwin* size at MH (i.e., byte range: 30 to 34)
4. Transmit Category #4: Not Yet Sent and Recipient Is Not Ready. This falls outside of the available *RWIN* space at MH (i.e., byte range: 35 to 41)

For data being received from FH, there are also four receive categories.

1. Receive Category #1: Received from FH and Acknowledged to FH. Some bytes may or may not have been transmitted to MH (i.e., byte range: 20 to 41 in Figure 35).
2. Receive Category #2: Received from FH but not yet acknowledged to FH. Up to 2 bytes are reserved (i.e., byte range: 42 to 43 in Figure 35). The second last byte reserved is to Freeze or Unfreeze and the last byte reserved is to maintain the end-to-end semantics.
3. Receive Category #3: Bytes Not Yet Received but Transmitter (FH) is permitted to transmit (i.e., byte range: 44 to 50 in Figure 35).
4. Receive Category #4: Bytes Not Yet Received but Transmitter (FH) is not permitted to transmit (i.e., byte range: from 51 to onward in Figure 35).

Immediately after the two TCP connections are established the "Right edge of Send window" and the "Right edge of Receive window" are aligned, and if the transmission from FH to EH is faster than the transmission from EH to MH then the "Right edge of Receive window" would be slide to the left direction, away from the "Right edge of Send window." At some point if these two edges are aligned again, then this indicates that all bytes from FH have been transmitted to MH. Further, if the "Receive Next Pointer" is also aligned with these two edges, then it indicates that all bytes have been delivered to MH and acknowledged to FH as well.

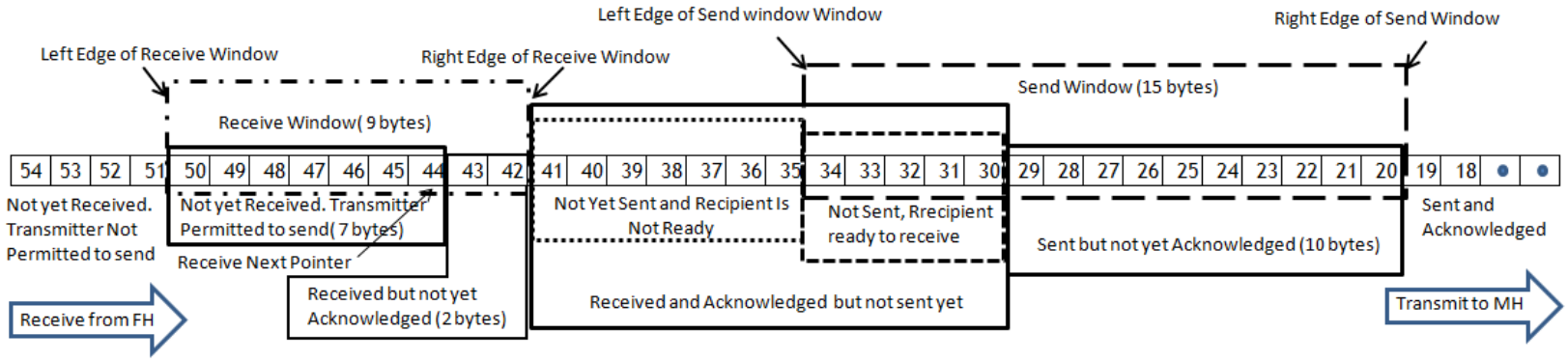


Figure 35: Sliding window diagram for send and receive window at ES-TCP host (EH)

5.4.2 ES-TCP Packet Flow Example

Figure 36, Figure 37, and Figure 38 show step by step ES-TCP operation through packet exchanges between a FH and a EH, and assume a total data transmission of 3000 from a FH to a MH. Figure 36 shows packet transmission sequences with an event of zero window messages from the EH to the FH that freezes FH, and the detailed description per packet flow is as following. The packet flow from 1 through 11 in between the FH and the EH occurs in parallel with the packet flow from 1' to 9' in between the EH and the MH, except for the packet #8 on the FH side TCP connection and the packet #6' in the MH side TCP connection.

1. Packet #1: The FH transmits 1000 bytes length packet to EH.
2. Packet #2: The EH sends an ACK packet corresponding to the packet #1 with acknowledgement sequence number equals to 998. The last 2 bytes received are not acknowledged yet.
3. Packet #3: Before receiving the ACK packet (packet #2), FH transmits another 1000 bytes length packet to the EH starting from sequence number equals to 1000.
4. Packet #4: The EH sends an ACK packet corresponding to the packet #3 with acknowledgement sequence number equal to 1998. The previously reserved 2 bytes are now acknowledged as a part of this ACK packet, but again the last 2 bytes received are not acknowledged yet. One difference between the previous ACK packet, packet #2, and this ACK packet (packet #4) is the *rwin* size. Packet #4 sets *rwin* to zero and it freezes the FH since the receive buffer at the EH is not available for now.
5. Packet #5: When receive buffer becomes available, the EH sends a window update packet with 1 byte increased acknowledge sequence number (=1999) from the unacknowledged 2 bytes that were reserved. This will unfreeze the FH and continue transmission.
6. Packet #6: The FH transmits the last 1000 bytes.

7. Packet #7: The EH sends ACK packet with ACK sequence number equal to 2998 which is again 2 bytes less than the received data bytes. Since the FH has transmitted all 3000 bytes, it waits for acknowledgement of the last two bytes after receiving this ACK packet.
8. Packet #8: When the EH receives acknowledgement for the last byte from MH (packet # 6'), it sends an acknowledgement (ACK=3000) for all received data to the FH. If the packet #6' is not received from the MH for any reason, the packet #8 cannot be transmitted from the EH to the FH.
9. Packet #9: After receiving an ACK packet for the last data byte from EH, the FH sends a FIN packet to terminate the TCP connection.
10. Packet #10: The EH sends FIN_ACK to FH.
11. Packet #11: The FH sends the last ACK packet to the EH and complete the three way handshake for terminating the TCP connection

The step 8 is a key procedure that maintains the end-to-end semantics. It prevents the EH from sending the final ACK packet to the FH until confirming that the last data byte is successfully received by the MH. The diagram in Figure 37 is the same as Figure 36, except for the packet sequence number 8 in Figure 37. After sending the ACK packet (#7), the EH runs an estimated RTO timer for the unacknowledged 2 bytes to prevent retransmission from the FH if an acknowledgement from the MH or a new packet from the FH is not received before the estimated RTO timer expires. If this timer expires, then the EH sends a zero window message to the FH with 1 byte acknowledgement to freeze the FH. When the EH receives acknowledgement for the last byte (packet #6') from the MH, it sends an acknowledgement (ACK=3000) for all received data to FH (packet #9) and continue process described in Figure 36. The diagram Figure 38 is the same as in Figure 37, except for the packet sequence numbers 8 and 9 in Figure 38. After sending the ACK packet (#7), the EH runs an estimated RTO timer for the unacknowledged 2 bytes to prevent retransmission from the FH if an acknowledgement from the MH or a new packet from the FH are not received before the timer expires. The FH, however, retransmits (packet #8), the unacknowledged data byte before the estimated RTO timer at the EH is expired, then the EH

immediately sends a zero window message (packet #9) to the FH with 1 byte acknowledgement to freeze the FH. When the EH receives acknowledgement for the last byte (packet #6') from the MH it sends an acknowledgement (ACK=3000) for all received data to the FH (packet #10) and continues the process described in Figure 37.

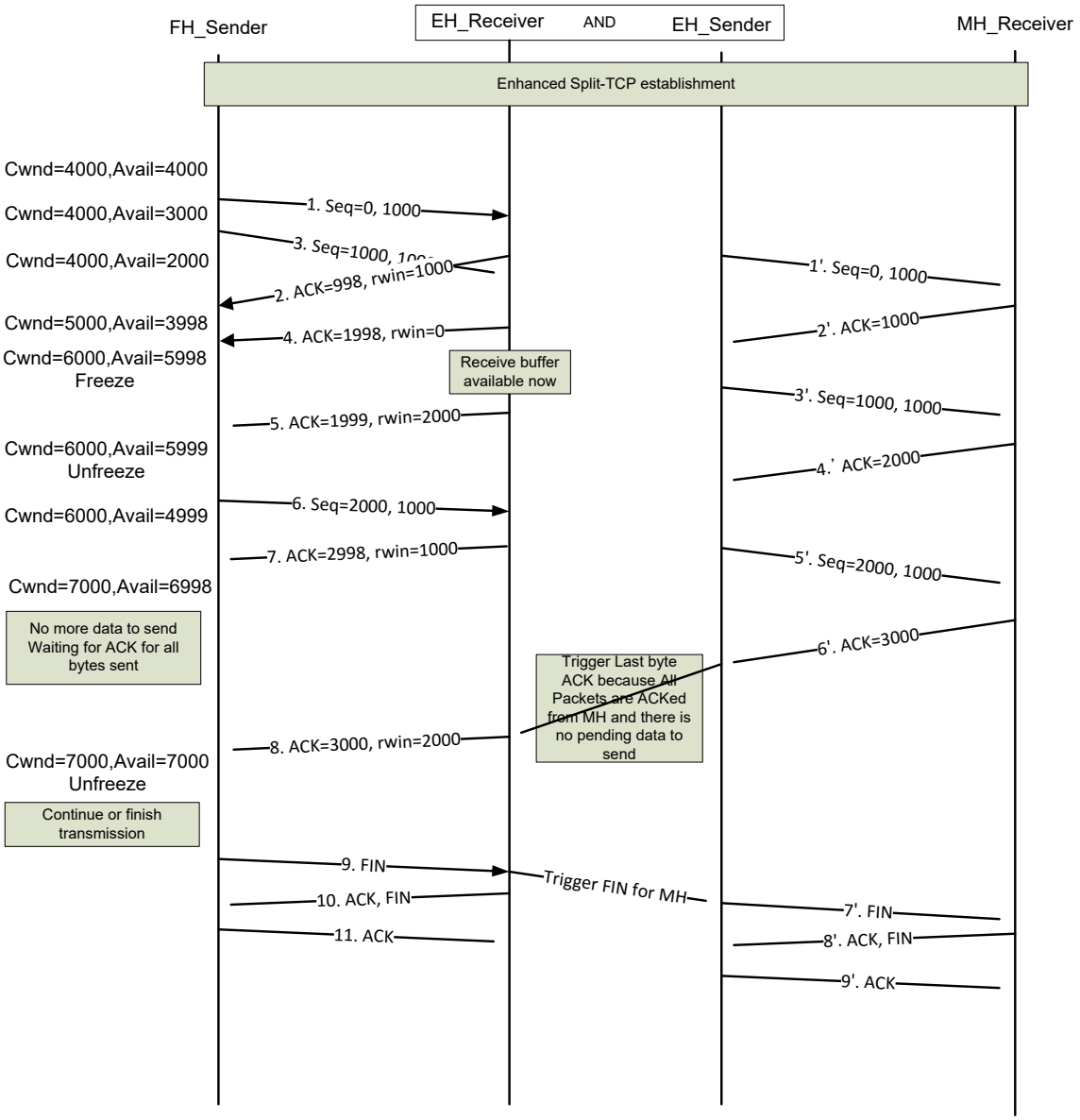


Figure 36: ES-TCP packet flow diagram: normal operation

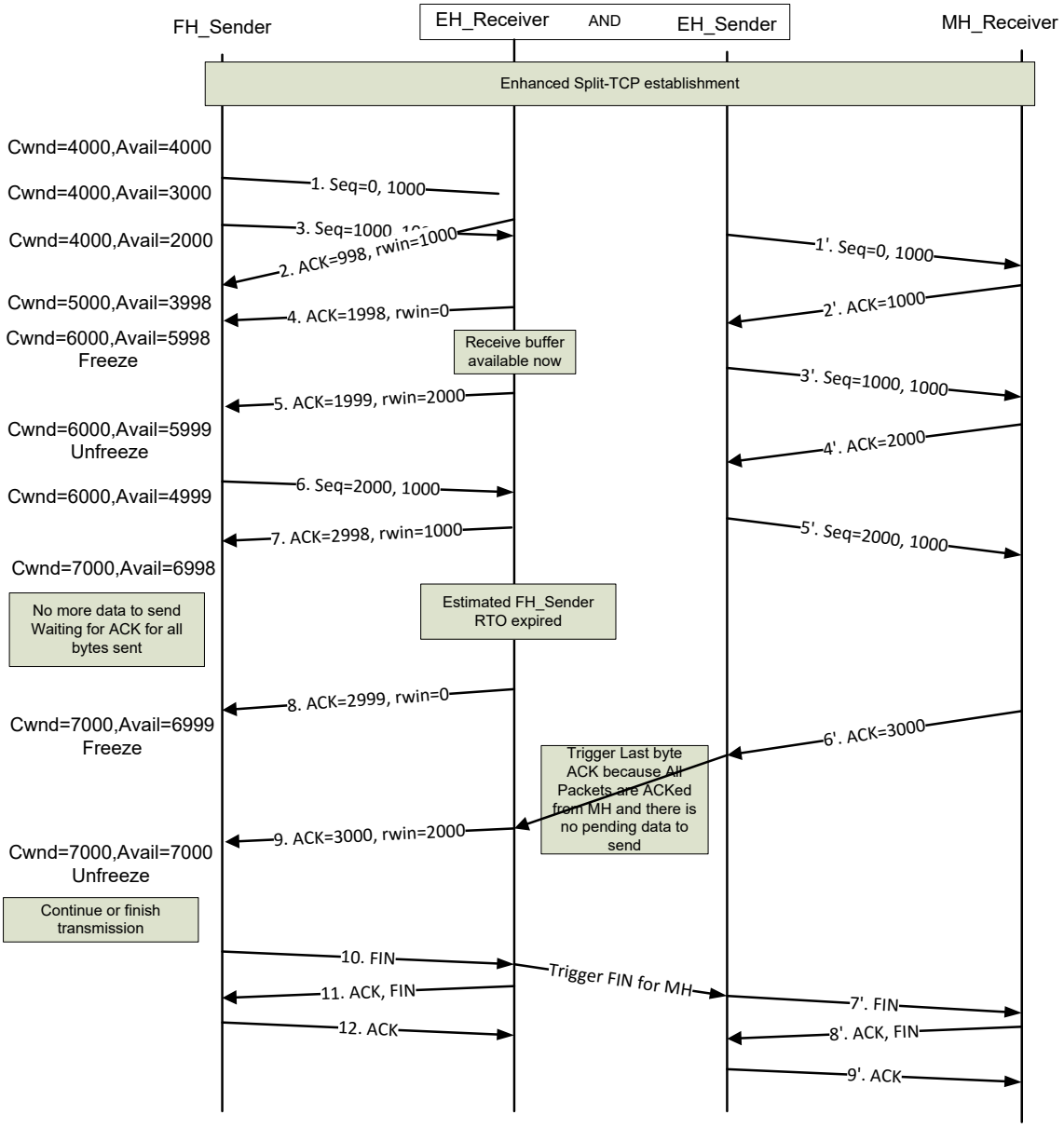


Figure 37: ES-TCP packet flow diagram: freeze FH to prevent retransmission

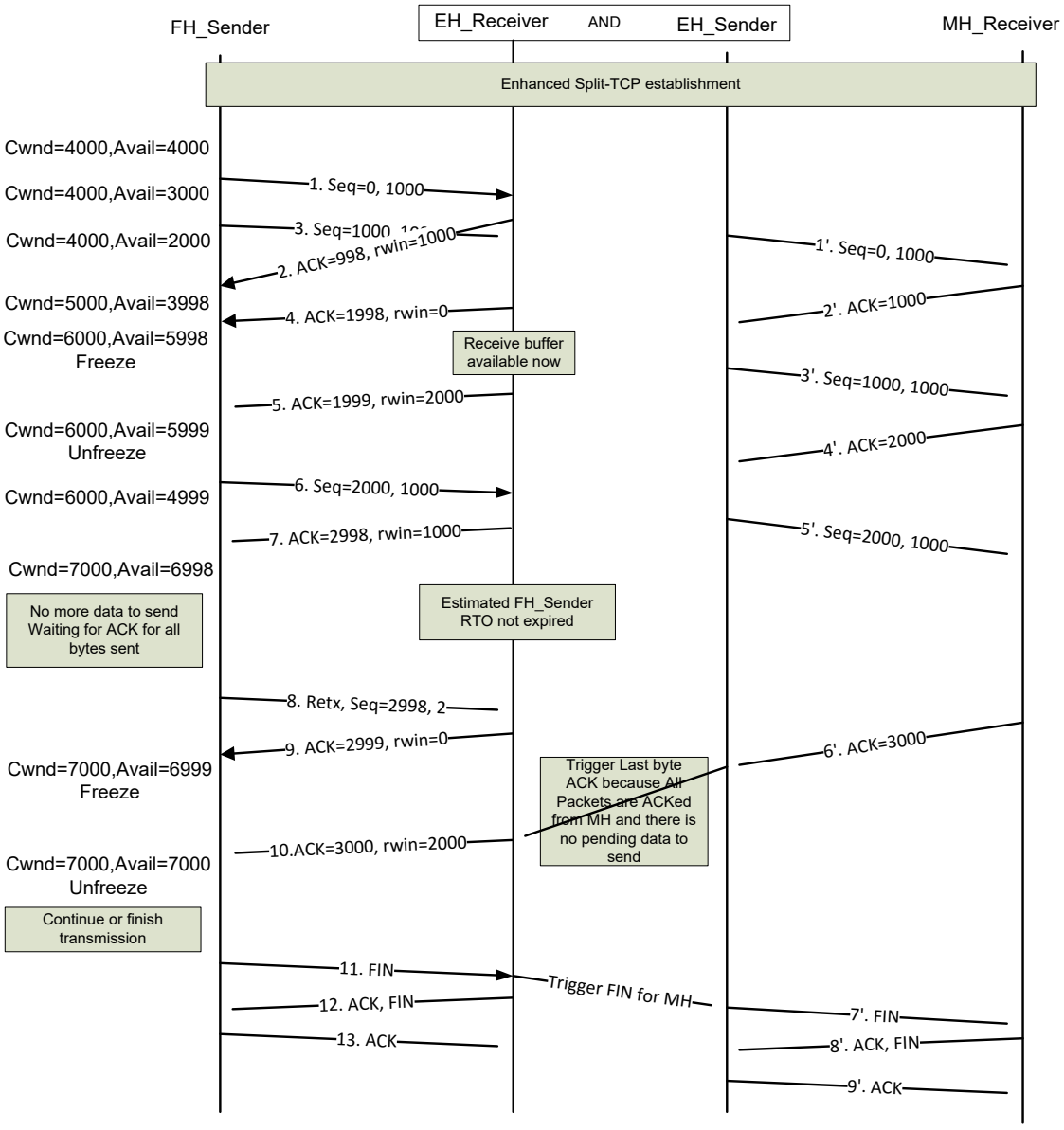


Figure 38: ES-TCP packet flow diagram: freeze FH_Sender after receiving retransmit packet for reserved bytes

5.5 Advanced Split TCP with End-to-end Protocol Semantics

The Advanced Split-TCP (AS-TCP) designed in this section complements the ES-TCP with the added goal to preserve the end-to-end TCP protocol semantics down to the byte level. As opposed to the ES-TCP, AS-TCP does require a TCP protocol extension. The ES-TCP and AS-TCP work independently of each other. It is further contemplated that depending on the capability of the FH, one may enable either ES-TCP or AS-TCP.

The "Advanced" mode (AS-TCP) designed in this paper requires to use a TCP header extension, as detailed below. This proposed TCP header requires 6 additional header size for the following three fields:

- Option-Kind: One byte length; it is the only mandatory field, which indicates the option kind.
- Option Length: One byte length; it is an optional field, which specifies the length of the option field.
- Option Data: 4 bytes length; it carries acknowledgement sequence number from MH that notifies successful data delivery to the destination.

The TCP receiver at the Advanced Split-TCP Host (AH) includes an acknowledgement number in the acknowledgement sequence number field, as the regular TCP does. The congestion window sliding at the FH is based on the said ACK sequence number found in the main TCP header. The sender (FH), however, empties the TCP transmit buffer based on the MH acknowledgment sequence number (MH_ack) inserted in the TCP Option field (see Figure 39). The byte by byte level of end-to-end semantics is preserved, even if an application uses multiple transactions per TCP connection. It relies on the TCP buffer status for a message delivery confirmation because the TCP transmit buffer at the sender entity (FH) is emptied only after a successful delivery to the final destination (MH).

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 000				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20	160	Option Kind=n										Option Length=6							Option Data = MH Acknowledgement number														
24	192	Option Data = MH Acknowledgement number																															

Figure 39: TCP extension field for the AS-TCP enhancement

The TCP receiver at the AS-TCP Host (AH) places an ACK sequence number received from a MH (MH_ack) in the MH Acknowledgement sequence number field of the TCP optional header, as shown in Figure 39. Subsequently, the FH empties it's transmit buffer based on the said MH_ack, since it was notified that the said MH has received data up to the said MH acknowledged sequence number (MH_ack). Note that the ACK sequence number received from the said MH must be mapped to a corresponding ACK sequence number used by the TCP connection between FH and AH because the initial sequence number of the TCP_{WAN} and TCP_{RAN} connections could be different. The congestion window sliding, however, at the FH is based on the ACK sequence number retrieved from the regular TCP header field. Once the MH_ack number is equal to the last byte transmitted, the FH recognizes that all transmitted bytes have been received by the destination (MH). This ensures preserving the end-to-end semantics down to the byte level.

5.5.1 AS-TCP Operation

The AS-TCP in the AH operates in exactly the same way as a conventional TCP, except for the fact that when an ACK packet is received from the MH. The AH forwards an acknowledgement sequence number from the MH to the FH using the proposed TCP option header. The TCP option header is not included if no ACK packet has been received from the MH since the latest "MH Acknowledgement sequence number" transmission, or if there has been a duplicate ACK packet received from the MH.

Figure 40 depicts a high level operational flow diagram for the AS-TCP at the AH and the FH, which is different from the conventional TCP protocol. The AS-TCP_{WAN} part is responsible for communicating with the FH, while the AS-TCP_{RAN} part is responsible for communicating with the MH. The MH is not shown in the diagram, since the traditional TCP protocol is used at the MH. We assume a file download from a FH to a MH to describe the operation depicted in Figure 40 . The steps below describe three key operations:

1. When data packets are received at the receiver entity at the AH (AH_Rx) from the FH, the received data bytes are buffered and ACK packets are transmitted to the FH, similarly to a conventional TCP. This ACK packet is called AH_ack, and is independent of the ACK packet from the MH. Therefore, the FH is allowed to continue transmitting new data, as it receives ACK packets from the AH. The AH_ack packets are delivered to the FH via the "Acknowledgement number" field in the regular TCP header.
2. The AH_Rx_0 is the main state in the AS-TCP-WAN that is triggered by the AH_Tx when the AH_Tx receives an ACK packet (MH_ack) from the MH. This is the only explicit communication between the AS-TCP-WAN and AS-TCP-RAN entities. Further, the AH delivers the MH_ack to the FH via the "MH acknowledgement number" field found in the TCP option header. Whenever the AH receives a non-duplicate ACK packet from the MH, the AH constructs a MH_ack and sends it to the FH to convey the sequence number of the data bytes received by the MH. This allows the FH to immediately clear its

transmission buffer up to the sequence number indicated through the MH_ack just received. For the sake of reducing the number ACK packets from the AH to the FH, the MH_ack may be simply piggybacked to a regular ACK packet for the TCP_{WAN} connection, providing that a regular ACK packet is available within a specified timer value.

3. The AH_Tx may self-freeze by setting the *rwin* size to zero each time there is a link interruption (e.g., link disconnection). Freezing the state of the AH_Tx preserves the TCP status. Hence, when the transmission resumes, the TCP transmission rate from the AH to MH corresponds to the rate stored in the freezing state of the AH_Tx.

The left side diagram (FH-AS-TCP) in Figure 40 runs at the FH supporting the AS-TCP and communicates with AH_Rx at the AH. The two main differences with respect to a conventional TCP protocol are the following:

1. The acknowledgement sequence number in the main TCP header received at the FH from the AH is used to slide the TCP send window, just as through a conventional TCP operation. A departure from a conventional TCP consists in the fact that the FH-AS-TCP does not delete the acknowledged data bytes from the FH transmission buffer because the acknowledgement sequence number only indicates the data bytes received by the AH, but not by the MH. On the other hand, if a MH_ack is included in a TCP option header received at the FH from the AH, the data bytes in the FH transmission buffer may be removed up to the MH_ack sequence number, since this notifies that those data bytes were successfully received by the MH.
2. Since there are two types of acknowledgement in a TCP ACK packet and either one or both may be present, one needs a procedure for identifying a duplicate ACK at the FH. FH-AS-TCP considers that an ACK packet is duplicated only if both the acknowledgement in the main TCP header and the MH_ack in the TCP option header have the same value as the previously received ACK packet. Otherwise the TCP ACK packet is not a duplicate packet. Furthermore, the TCP ACK packet is not a duplicate if

the rwin value in the ACK packet is larger than 0 and the FH state is frozen. Such a TCP ACK packet is considered as a "window update" request packet.

Figure 41 illustrates a slide window buffer that is used at both receiver (AH_Rx) and transmitter (AH_Tx) using a shared buffer at the AH. The packets from the FH are buffered to the left side of the diagram, while the packets to be transmitted to the MH are shown on the right side of the diagram. The values in the buffer space represent packet sequence number.

For data being transmitted to the MH, there are four transmit categories, which are described starting from the right side of the diagram:

1. Transmit Category #1: "Sent And Acknowledged" from the MH (i.e. byte range: 0 to 19)
2. Transmit Category #2: "Sent But Not Yet Acknowledged" from the MH (i.e. byte range: 20 to 29)
3. Transmit Category #3: "Not Sent, Recipient Is Ready To Receive". This represents the *rwin* size at the MH (i.e. byte range: 30 to 34)
4. Transmit Category #4: "Not Yet Sent and Recipient Is Not Ready". This indicates the content outside of the available *rwin* space at the MH (i.e. byte range: 35 to 41)

For data being received from the FH, there are four receive categories:

1. Receive Category #1: "Received from the FH and Acknowledged to the FH". Some bytes may or may not have been transmitted to the MH (i.e. byte range: 20 to 41)
2. Receive Category #2: "Received from the FH but not yet acknowledged to the FH". The number of deferred acknowledgement bytes depends on the conventional TCP dynamics. This category is not shown in Figure 41.
3. Receive Category #3: "Bytes Not Yet Received but the Transmitter (FH) is permitted to transmit" (i.e. byte range: 42 to 50)
4. Receive Category #4: "Bytes Not Yet Received and the Transmitter (FH) is not permitted to transmit" (i.e. byte range: from 51 onwards)

Immediately after the TCP connection is established, the "Right edge of the Send window" and the "Right edge of the Receive window" are aligned. If the transmission from the FH to the AH is faster than the transmission from the AH to the MH, the "Right edge of the Receive window" would be slide to the left, away from the "Right edge of the Send window." At some point, if these two edges are aligned again, it means that all the data from the FH has been transmitted to the MH. Further, if the "Receive Next Point" is also aligned with these two edges, it means that all the data has been delivered to the MH and acknowledged to the FH as well.

Figure 42 illustrates a conceptual diagram of the transmission sliding window at the FH. The packets from the FH are transmitted to the AH in order, as illustrated from right side of the diagram. The values in the buffer space represent packet sequence numbers and are aligned with the sequence numbers in Figure 41.

For data being transmitted from the FH to the AH, there are five transmit categories:

1. Transmit Category #1: “Sent and Acknowledged by the AH” (i.e. byte range: 20 to 41)
2. Transmit Category #2: “Sent but Not Yet Acknowledged by the AH”. Since this diagram assumes all data bytes have been acknowledged, this category is not shown.
3. Transmit Category #3: “Not Yet Sent but Recipient Is Ready”. This represents the *rwin* size at the AH (i.e. byte range: 42 to 50).
4. Transmit Category #4: “Not Yet Sent and Recipient Is Not Ready”. This indicates the content outside of the available *rwin* size at the AH (i.e. byte range: 51 to onward).

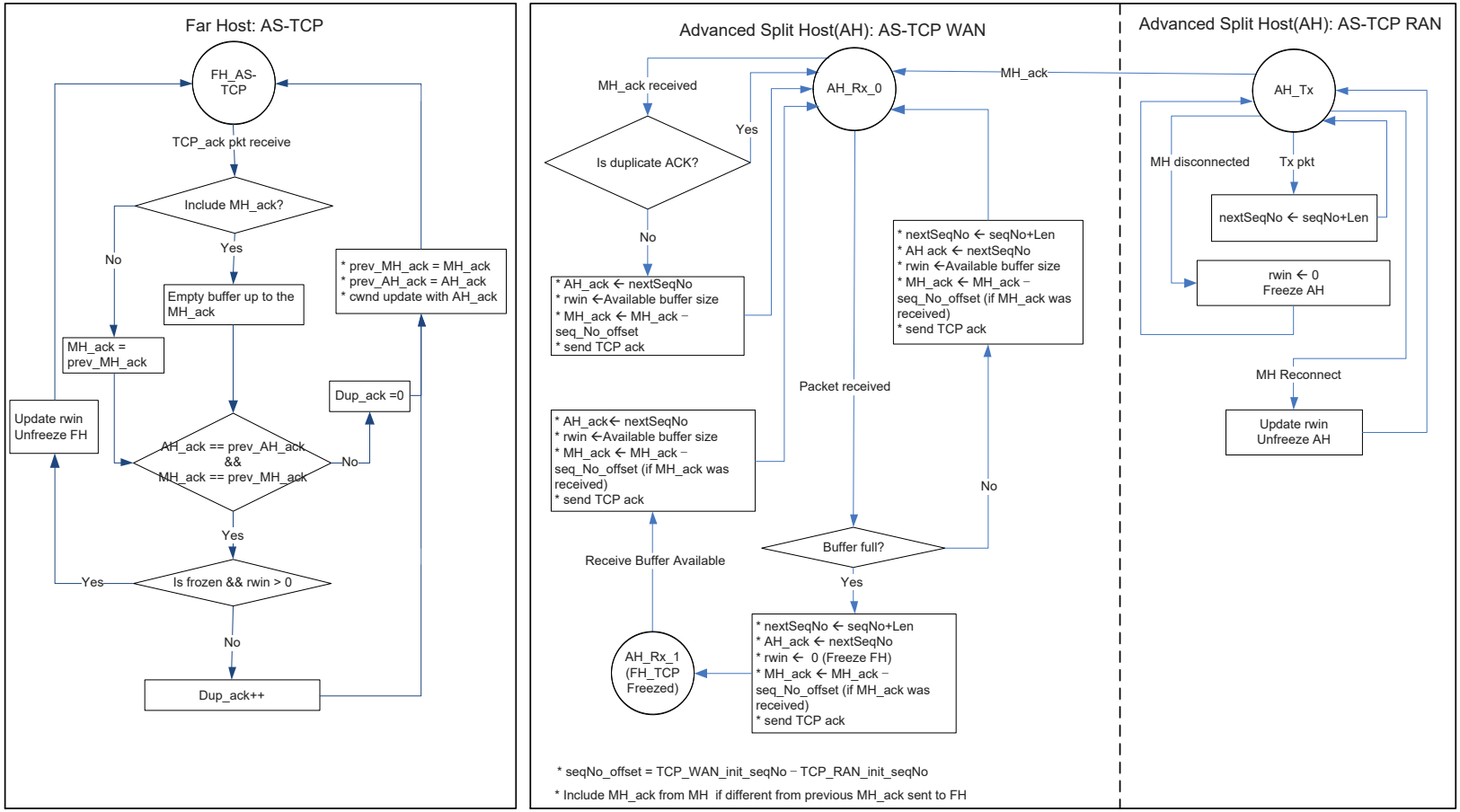


Figure 40: Operation flow diagram for AS-TCP at AH and FH

5. Transmit Category #5: “Acknowledged by the MH”. This indicates that the data bytes delivered to the MH (i.e. byte range: 0 to 19). Thus, the buffer up to the sequence number 19 can be cleared at the FH.

The transmit category #5 is the main addition to the AS-TCP for preserving the end-to-end semantics at byte level. The FH does not delete the data bytes from the 20th byte to the 41st byte, even though they have been successfully delivered to the AH. The send window, however, is still able to slide and hence, it allows to transmit new data bytes, regardless of the acknowledgement status from the MH, as long as the AH permits the transmission.

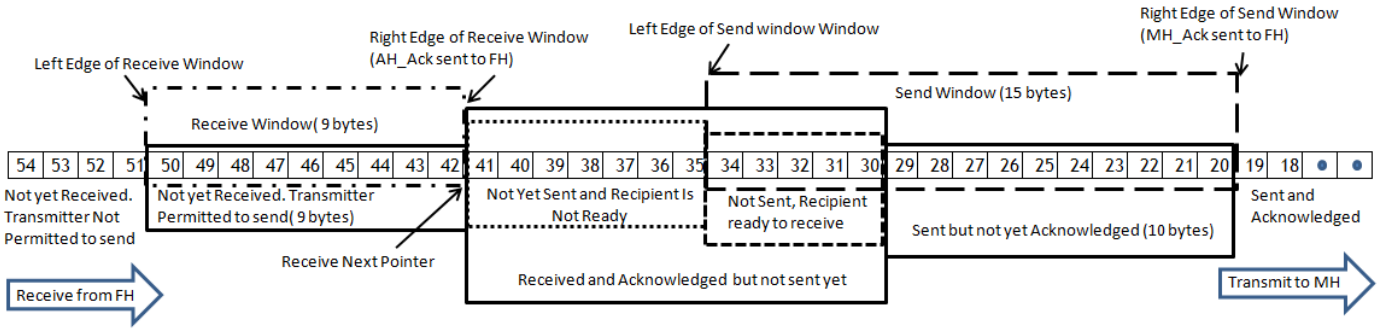


Figure 41: Sliding window diagram for send and receive window at AS-TCP host (AH).

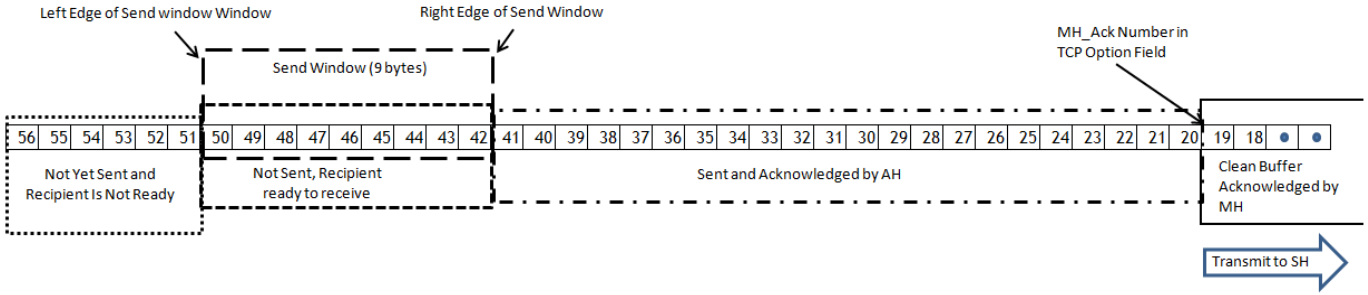


Figure 42: Sliding window diagram for send window at FH supporting AS-TCP.

5.5.2 AS-TCP Packet Flow Examples

Figure 43 and Figure 44 show step by step AS-TCP operations through packet exchange between the FH, AH and MH entities. Figure 43 also shows the interaction between the AH_Sender and the AH_Receiver. It assumes a total data transmission of 3000 bytes from the FH to the MH.

Figure 43 shows packet transmission sequences with an event of ZWA from the AH to the FH that freezes the FH, and a detailed description per packet flow is as following.

1. Packet #1" and #1: The MH sends a TCP Sync packet to the FH to establish a TCP connection. In the middle of the network, however, a AH intercepts the TCP sync packet and requests a TCP connection to the FH on behalf of the MH with an “AS-TCP option” request.
2. Packet #2 and #2": The FH recognizes that this TCP Sync packet is from an AS-TCP host and responds with an “AS-TCP option” capability (#2). The AH sends a TCP Sync ACK to the MH once it receives the TCP Sync ACK from the FH. Now there are two TCP connection established: a TCP_{WAN} connection between the AH and the FH and another conventional TCP_{RAN} connection between the AH and the MH.
3. Packet #3: The FH transmits a 1000 bytes length packet to the AH, starting from the sequence number equal to zero.
4. Packet #4 and #3": The AH sends an ACK packet corresponding to the packet #3 with the acknowledgement sequence number equal to 1000 (packet #4). The AH also sends the received 1000 bytes data packet to the MH (packet #3”).
5. Packet #5: Before receiving an ACK packet (packet #4), the FH transmits another 1000 bytes length packet to the AH, starting from the sequence number equal to 1000. This is possible because the current congestion window size (*cwnd*) is 4000 bytes.
6. Packet #6: The AH sends an ACK packet corresponding to the packet #5 with the acknowledgement sequence number equal to 2000. The *rwin* size value in this ACK

packet is zero. This packet #6 carries a ZWA from the AH, and it freezes the FH transmission since the receive buffer at the AH is not available for now.

7. Packet #4": A MH_ack packet is received at the AH from the MH and it triggers an ACK packet from the AH to the FH, indicating that the first 1000 bytes of data have been successfully received by the MH.
8. Packet #7: A new ACK packet is received at the FH from the AH; it is triggered by the ACK packet (ACK=1000) received from the MH. The acknowledgement sequence number from the MH is sent to the FH through the MH_ack in the TCP option header. Other fields, such as the ACK sequence number in the main TCP header and the *rwin* size are the same as for the previous packet #6. The TCP at the FH may now remove the data bytes up to the sequence number 1000 from the transmit buffer.
9. Packet #8: When the receive_buffer_status becomes available at the AH, the AH sends a window update packet. This unfreezes the FH and it allows the FH to resume the transmission.
10. Packet #9: The FH transmits the last 1000 bytes.
11. Packet #6": The MH sends a MH_ack indicating that up to the first 2000 bytes have been received at the MH after receiving the packet #5".
12. Packet #10: AH sends an ACK packet with the ACK sequence number equal to 3000. This ACK packet also includes the MH_ack sequence number (=2000) received from the MH. Since the FH has transmitted all the 3000 bytes, it awaits for the acknowledgement of the last 1000 bytes from the MH after receiving this ACK packet #10. The TCP at the FH may now remove the data bytes up to the sequence number 2000 from its transmit buffer. Even though all 3000 bytes transmitted have been acknowledged by the AH, the FH knows that the delivery of the last 1000 bytes has not been confirmed by the MH yet.
13. Packet #7" and #8": The MH sends a MH_ack indicating that the total of 3000 bytes has been received (#8") after receiving the third segment of 1000 bytes from the AH (#7")

14. Packet #11: When the AH receives the packet #8” from the MH, it sends an acknowledgement (MH_ack=3000) for all received data to the FH. The TCP at the FH now received all the ACK sequence numbers, indicating that all data has been transmitted successfully to the MH. FH may now remove all data from the transmit buffer.
15. Packet #12: After receiving an ACK packet for the last data bytes delivered to the MH, it sends a FIN packet to terminate the TCP connection.
16. Packet #13: Send a FIN_ACK from the AH to the FH.
17. Packet #14: Complete the three way handshake for terminating the TCP connection.
18. Packet #9", #10", and #11": Perform TCP termination between the AH and the MH. Either one of these two TCP connections may initiate the TCP termination.

The data transmission scenario depicted in Figure 44 diagram is the same as the scenario described in Figure 43, except that there are multiple AS-TCP hosts between the FH and the MH. The diagram in Figure 44 assumes two AS-TCP hosts and shows three TCP connections: a first one between the FH and the AS-TCP host #1, a second one between the AS-TCP host #1 and the AS-TCP host #2, and a third one between the AS-TCP host #2 and the MH. Regardless of the number of AS-TCP hosts, all TCP connections behave similarly to a conventional TCP protocol, except from supporting the MH_ack delivery through the TCP option header.

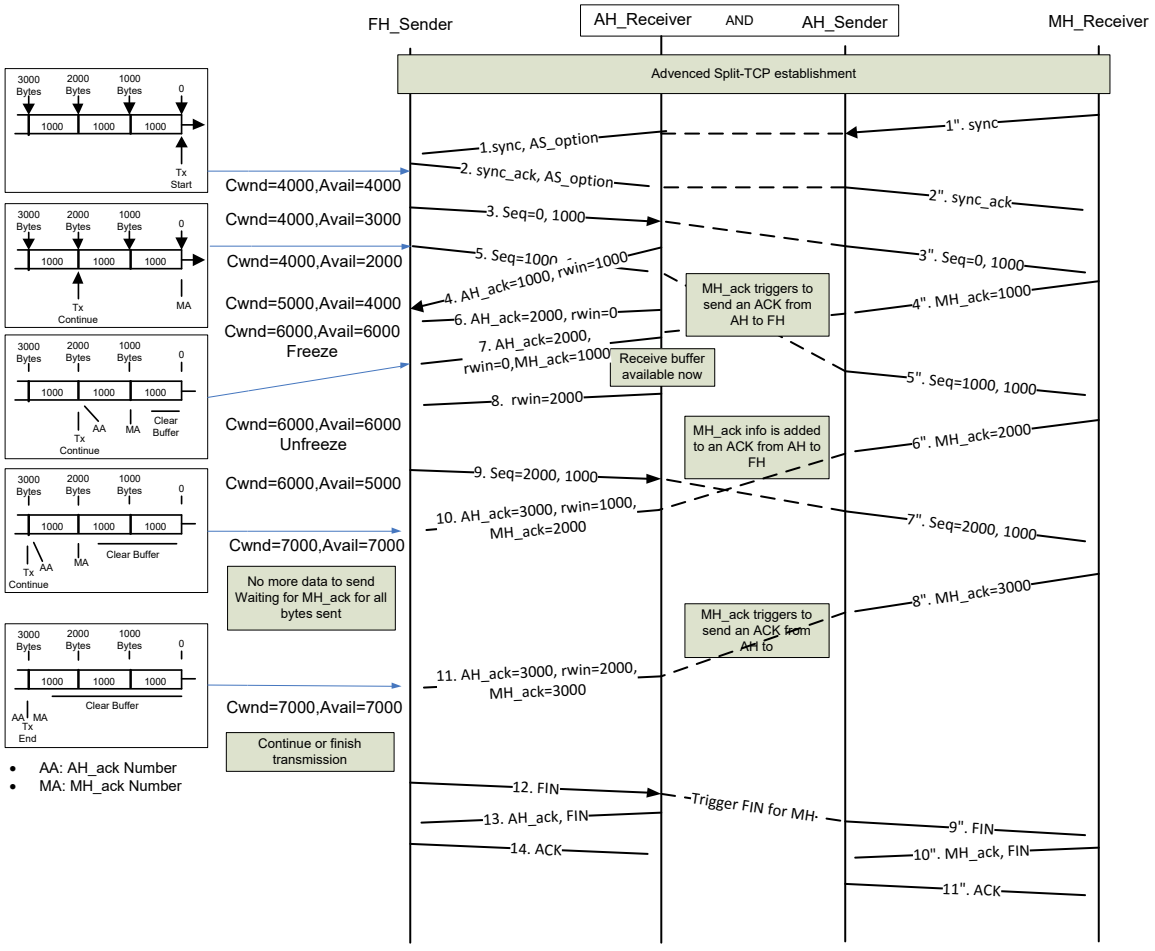


Figure 43: AS-TCP packet flow diagram with a single AS-TCP host.

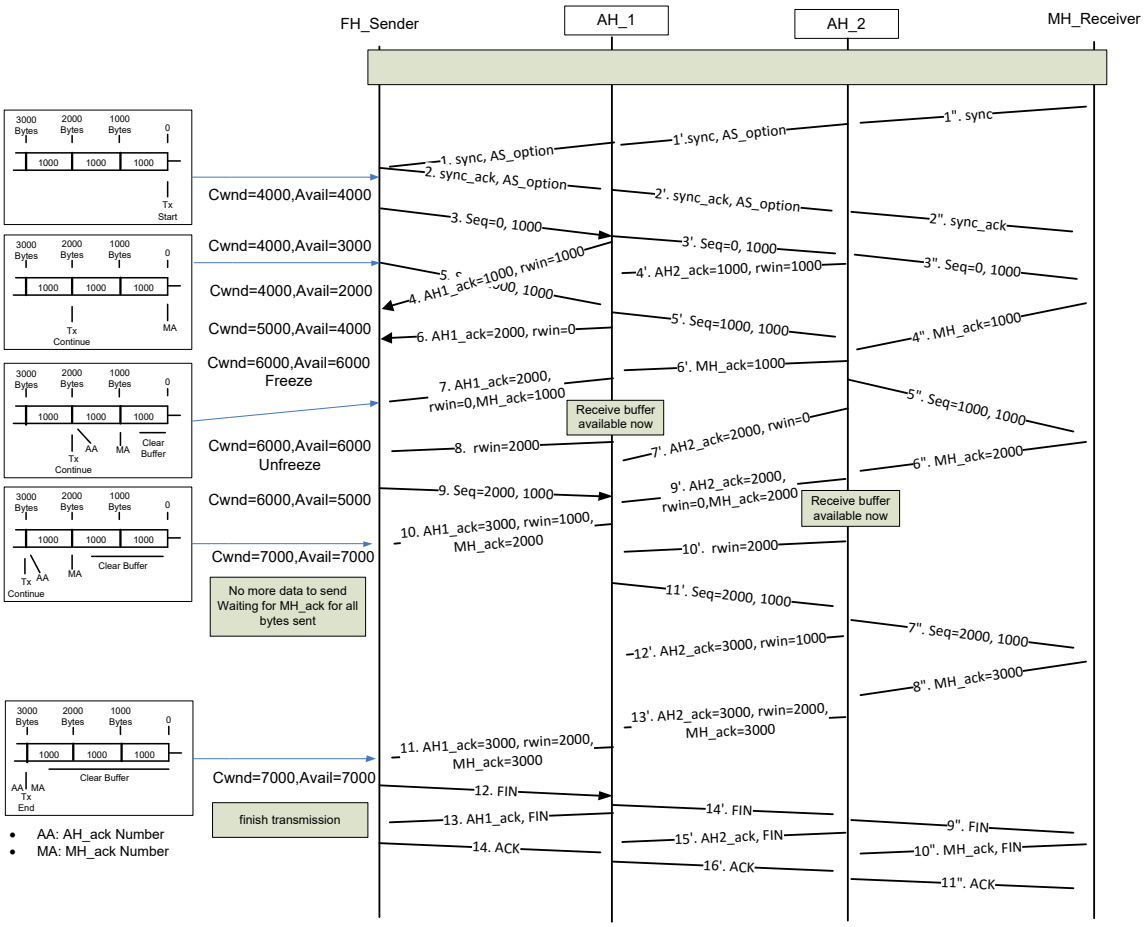


Figure 44: AS-TCP packet flow diagram with multiple AS-TCP hosts

5.6 Performance Evaluation: ES-TCP and AS-TCP

5.6.1 End-to-end Application Performance Simulation Platform

This section we validate the proposed solutions and evaluate the TCP throughput gains achieved through ES-TCP and AS-TCP. Since the application performance is critically dependent on the mutual interaction across the network protocol layers (PHY, MAC, and higher protocol layers), the actual performance estimation would be significantly different if done in isolation, without considering such cross layer interactions. In particular, the dynamic nature of the TCP protocol makes it difficult to estimate and analyze the performance of applications that rely on TCP as a transport layer over wireless networks. To tackle this complexity, we implemented the ES-TCP and the AS-TCP on top of a unique end-to-end LTE network and application performance simulation platform [14], which allows assessing the mutual impact of lower layer protocols and higher layer protocols , along with wireless link characteristics and network delay/congestion effects on the TCP performance (see Figure 45). This simulation platform provides a flexibility to change the network topology and network configurations so that the simulations under various network scenarios can be performed. Figure 46 depicts the protocol stacks implemented in the MH, EH/AH, and FH. Other network elements in the simulation platform are built similar way as depicted in Figure 46. The MH and FH include application layer, transport layer, IP layer, link layer (i.e. LTE MAC layer for the MH and Ethernet MAC layer for the FH), and abstraction of the LTE physical layer for the MH. The EH/AH is a key component of the proposed ES-TCP and AS-TCP enhancements. When a file download request arrives at the Split-TCP host from the MH, it forwards the application request to the FH on behalf of the MH and creates two TCP legs; one for TCP_{RAN} and the other one for TCP_{WAN}. The TCP adaptation layer in the EH/AH inter-connects these two TCP connections so that the packet from the TCP_{WAN} receive buffer is forwarded to the TCP_{RAN} transmit buffer and the ACK sequence number received by the TCP_{RAN} is used by the TCP_{WAN} entity in EH/AH.

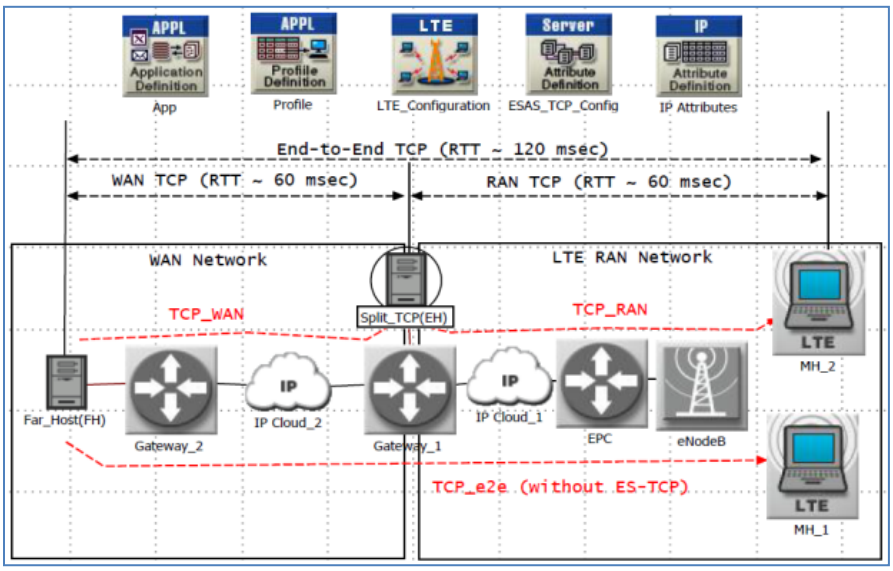


Figure 45: LTE application performance simulation platform(simulation network architecture and configuration) with ES-TCP/AS-TCP

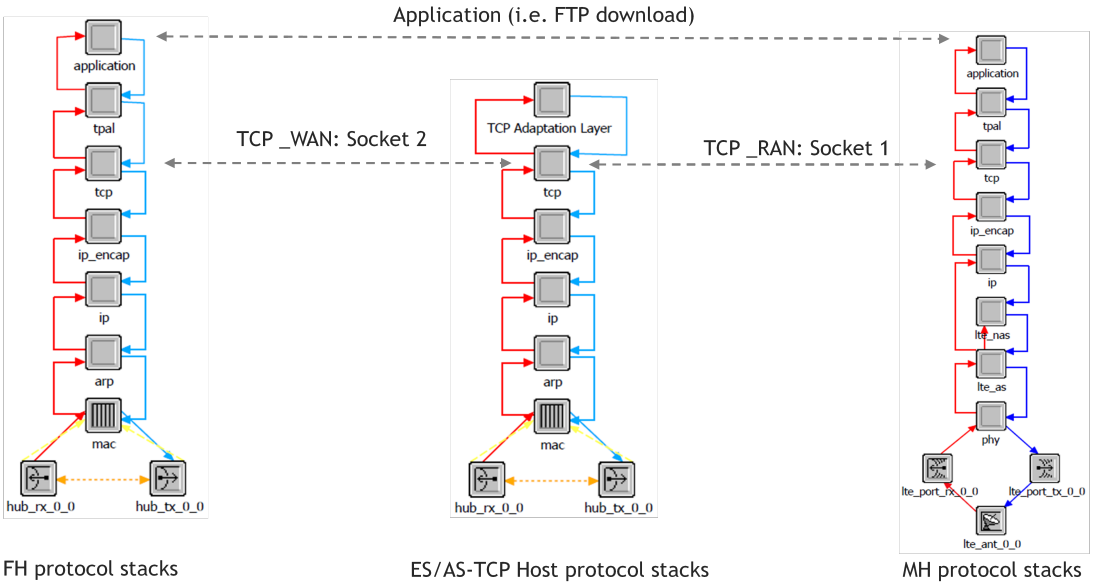


Figure 46: Protocol stacks in the host device models

5.6.2 Simulation Scenarios

We perform a file download from a FH to a MH via the Split-TCP host (i.e. EH for ES-TCP, and AH for AS-TCP), with various combinations of parameters listed in Table 5. For each scenario, each data point was recorded by averaging the results over 100 simulation runs. The packet error rate ranges are selected based on internal measurement evaluation from a commercial LTE network. The same test scenarios are repeated per Split-TCP enhancement mechanism: ES-TCP or AS-TCP.

5.6.3 Results and Analysis

Prior to the efficiency of the proposed Split-TCP based enhancements, we validates three key expected capabilities from both ES-TCP and AS-TCP enhancements: 1) the TCP session end-to-end semantics is maintained, (2) the long network delay is split across the TCP_{WAN} and TCP_{RAN} segments, and (3) network impairments such as packet losses over a network segment are isolated and handled by the corresponding TCP connection.

A. ES-TCP Enhancement

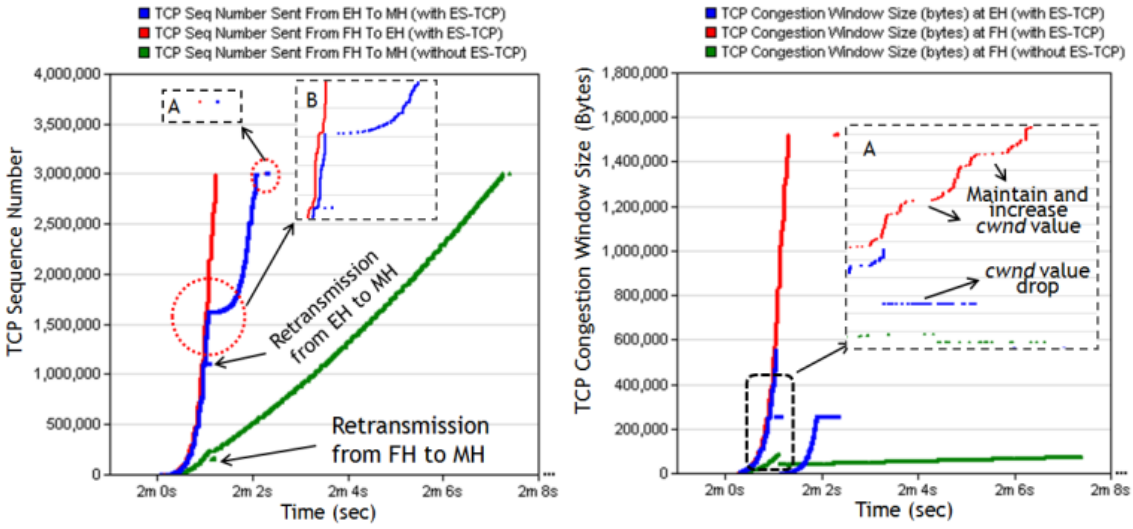
Figure 47 (a) illustrates the three TCP sequence number graphs while downloading a 3 Mbytes file size: a green curve from FH to MH (TCP_{e2e}) when the ES-TCP is disabled, a red curve from FH to EH (TCP_{WAN}) and a blue curve from EH to MH (TCP_{RAN}) when the ES-TCP is enabled. Figure 47 (b) shows the three TCP *cwnd* size curves for each of the TCP connections from the same simulation. It indicates that the download time without ES-TCP is 7 sec, and it is reduced to 3 sec when our proposed ES-TCP is enabled. During this simulation experiment, multiple packet drops were enforced in the RAN segment of the network around the 121^{st} sec of the simulation time, in order to emulate stress test conditions for TCP. Such conditions may often happen in operational wireless networks, and these simulation results emphasize the importance of designing resilient network protocols.

Table 5: Simulation parameters

End-to-End Delay	120 ms	
Location of EH and AH	(35 ms:85 ms), (60 ms:60 ms), and (85ms:35ms) Note: Parameters in the parentheses indicate (RTT for TCP_{RAN}, RTT for TCP_{WAN})	
Packet Error Rate	RAN	0.01%, 0.05%, 0.1%, 0.5%, 1.0%
	WAN	0%
File Size (Mbytes)	0.5MB, 1MB,5MB,10MB,20MB	
LTE Parameters	Frequency Bandwidth	10 MHz
	MIMO Transmission Technique	Transmit Diversity
	Modulation Coding Scheme(MCS)	5, 10, 15, 20, 25
	HARQ Error Rate	10%
	Max HARQ Retransmission Count	4
TCP Parameters	TCP Flavor	New Reno
	Receiver Buffer Size(Bytes)	64KB, 128KB, 512KB
	Delayed ACK Count	2
	Max ACK Delay	200 ms

The TCP sequence graphs correspond to the TCP packet retransmission: the green curve is between the FH and the MH without ES-TCP and the blue curve is between the EH and the MH with ES-TCP. One can see that the TCP_{WAN} sequence graph (the red curve) has not been affected by the packet loss occurred in the RAN segment, as intended by the ES-TCP.

Furthermore, the *cwnd* graph (Figure 47.b) clearly shows that the *cwnd* size of the TCP_{WAN} (the red curve) is increasing continuously, while the *cwnd* size of TCP_{RAN} (the blue curve) was halved first and was further reduced later to the initial *cwnd* size. The enlarged graph (block-A) in Figure 47 (b) also illustrates that the FH stopped transmitting during the packet transmission interruption in the TCP_{RAN} segment and maintains the *cwnd* size of TCP_{WAN} as intended. The enlarged block-A in Figure 47 (a) indicates that the TCP_{WAN} closes the TCP connection after all data bytes are delivered to the MH because the EH holds the last acknowledgement byte until the last acknowledgement packet is received by the EH from the MH after the MH receiving the last data byte. This validates that with ES-TCP: (1) the TCP session end-to-end semantics is maintained, (2) the long network delay is split across the TCP_{WAN} and TCP_{RAN} segments, and (3) network impairments such as packet losses over a network segment are isolated and handled by the corresponding TCP connection. Because of the space limitation, only selected data points



(a) (b)

Figure 47: Comparison of (a) TCP sequence number sent and (b) *cwnd* size

that represent overall trend out of more than 12000 data points are presented in this dissertation.

Figure 48 shows the amount of in-flight bytes in TCP_{WAN} and TCP_{RAN} when ES-TCP is enabled and compares to the in-flight bytes in TCP_{e2e} without ES-TCP enhancement. For this experiment, 10 Mbytes file download was performed with 0% packet error in the wired network and both RTT_{WAN} and RTT_{RAN} are set to 60 ms.

The 10 Mbytes file download without ES-TCP takes about 7 sec and the in-flight bytes in the TCP_{e2e} connection is about 512 Kbytes (see green curve in Figure 48), which is same as the maximum *rwin* size at the destination (i.e. MH). This indicates that one of the throughput bottleneck reasons could be the maximum *rwin* size. However, the download time with ES-TCP enhancement is reduced from 7 sec to 4.5 sec. The TCP throughput gain is about 56%. The in-flight byte in TCP_{WAN} and TCP_{RAN} is about 450 Kbytes on average for the duration of the TCP sessions. Since these two TCP connections run in parallel, the total end-to-end in-flight byte with ES-TCP is 900 Kbytes. Thus the throughput performance gain could be achieved.

We also measured the amount of the buffer occupancy in the Split-TCP host if a buffer starvation may occur. The buffer occupancy in Figure 49 is measured from the same experiment for the information in Figure 48. The maximum buffer size configuration at the Split-TCP host was set to 1 Mbytes which includes the maximum *rwin* size for the TCP_{WAN} connection. The graph indicates that there is no buffer starvation and the buffer occupancy is consistently maintained at about 600 Kbytes level. The maximum buffer size allocation in the Split-TCP host should be as small as possible to save the system resource because patching more data with a larger buffer would not increase the throughput performance gain as long as the buffer starvation does not occur.

Figure 50 shows the TCP throughput performance gain with various file sizes and radio conditions when the RTT_{e2e} is split into (RTT_{RAN}: RTT_{WAN}) according to the following scenarios: (35ms:85ms), (60ms:60ms) and (85ms:35ms). The *rwin* is set to 128 Kbytes and the packet loss rate in the RAN is set to 0.01% in all these experiments.

Comparing the performance of the proposed ES-TCP against regular TCP for all these scenarios, the highest throughput performance gain in favor of ES-TCP is 94%, and corresponds to the scenario with the TCP Split configuration (60ms:60ms), 20 Mbytes file size, and best radio conditions assumed here (MCS=25); this is illustrated in Figure 50 (b). The performance gain, however, varies significantly over a wider range of file sizes and radio conditions. Figure 50 (a) corresponding to the TCP Split configuration (35ms:85ms) shows performance gains in the order of 20% to 50%, and Figure 50 (c) corresponds to the TCP Split configuration (85ms:35ms) and shows more limited performance gains in the order of 20% to 40%. These are more modest gains in comparison to the previous two configurations. This is due to the fact that the RAN delay is dominant and the RTT_{e2e} splitting point is relatively closer to the traffic source (FH). Figure 50 (d) is a comparison of the scenarios (a) and (b) by overlapping these two graphs together.

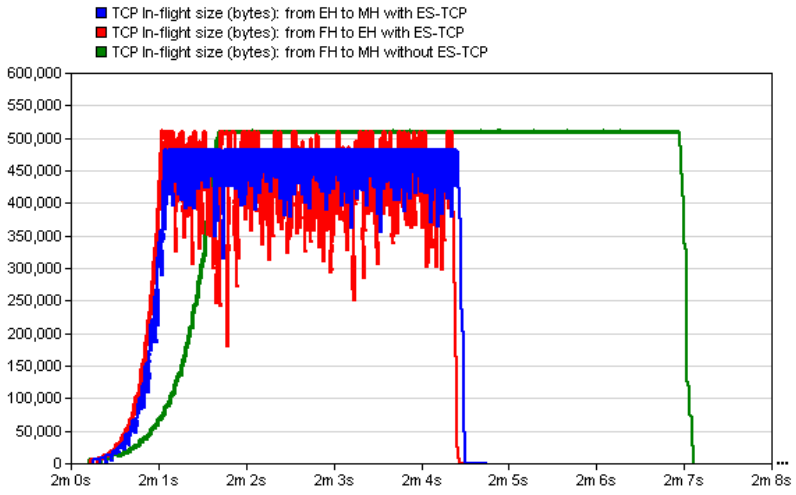


Figure 48: In-flight data size (bytes) in TCP_{e2e}, TCP_{RAN} and TCP_{WAN}

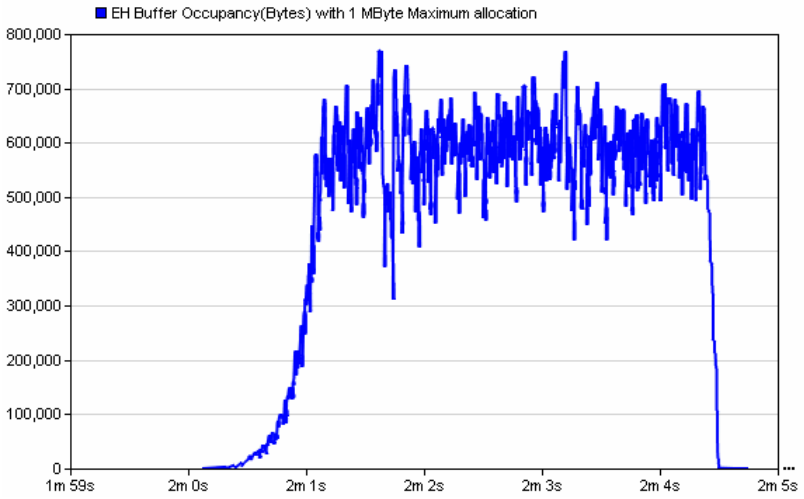


Figure 49: Buffer occupancy in the Split-TCP host (bytes)

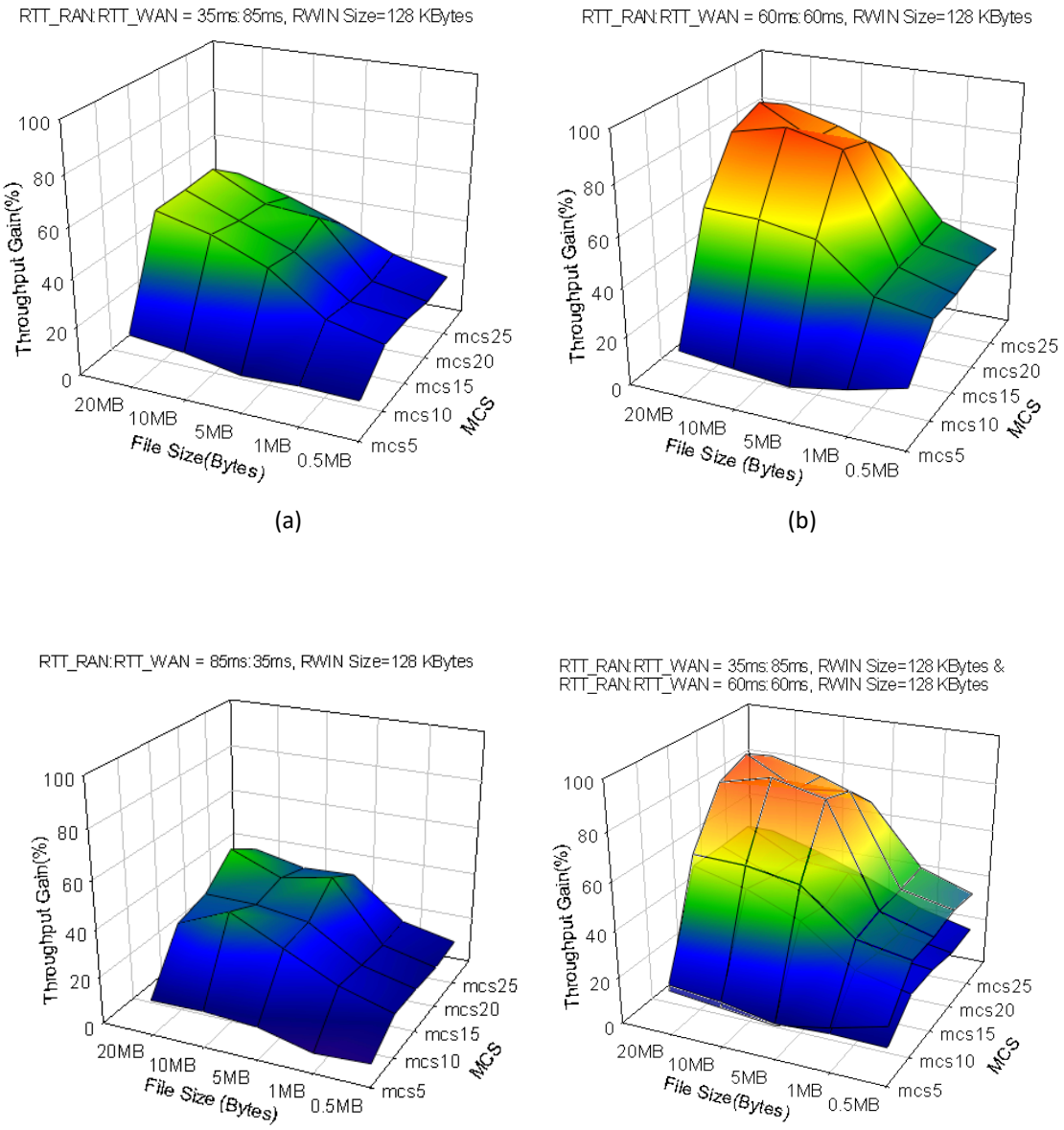


Figure 50: ES-TCP: Throughput gain as a function of radio conditions, file sizes and TCP Split configuration: (a) (35ms:85ms), (b) (60ms;60ms), (c) (85ms:35ms), (d) comparison of scenarios (a) and (b)

B. AS-TCP Enhancement

The main difference between the AS-TCP and the ES-TCP mechanisms is the way of maintaining the end-to-end semantics, and separating TCP_{e2e} in to TCP_{WAN} and TCP_{RAN} are same as the ES-TCP. The AS-TCP host forwards a part of the acknowledgement information from MH to FH to maintain the end-to-end protocol semantics instead of holding a few byte from the acknowledgement sequence number.

Figure 51 (a) illustrates the three TCP cwnd size graphs while downloading a 2 Mbytes file size: a green curve from FH to MH (TCP_{e2e}) when the AS-TCP is disabled, a red curve from FH to AH (TCP_{WAN}) and a blue curve from AH to MH (TCP_{RAN}) when the AS-TCP is enabled. To evaluate if the proposed mechanism clearly isolates the packet losses within the TCP segment which caused the errors, we induced multiple packet losses in the TCP_{RAN} connection around the 121st sec of the simulation time. One can see that the TCP_{WAN} *cwnd* size (the red curve) has not been affected by the packet losses occurred in the RAN segment and continues to grow, as intended by the AS-TCP, but the *cwnd* size of the TCP_{RAN} (the blue curve) is halved and further reduced down to the initial *cwnd* size. Figure 51 (a) also depicts that the TCP_{WAN} connection is maintained until the all data bytes are successfully delivered to the MH. The point P1 indicates that the FH has completed the transmission to the AH at round 121.6^{th} sec of the simulation time, but the red curve is extended to the point P2 that is aligned with the point P3, which in turn indicates the completion of all data bytes delivery from the AH to the MH.

Figure 51 (b) shows the TCP ACK sequence number reported from the MH to the AH (the blue curve), along with both types of ACK sequence numbers from the AH to the FH (the green and the red curves) while downloading a 2 Mbytes file size. The ACK sequence number on the green curve and on the red curve, as pointed by the arrow headed line ‘A’, are delivered in a single ACK packet from the AH to the FH: one in a regular TCP header and the other one in a TCP option header. The ACK sequence number in a regular TCP header is used for the TCP_{WAN} congestion

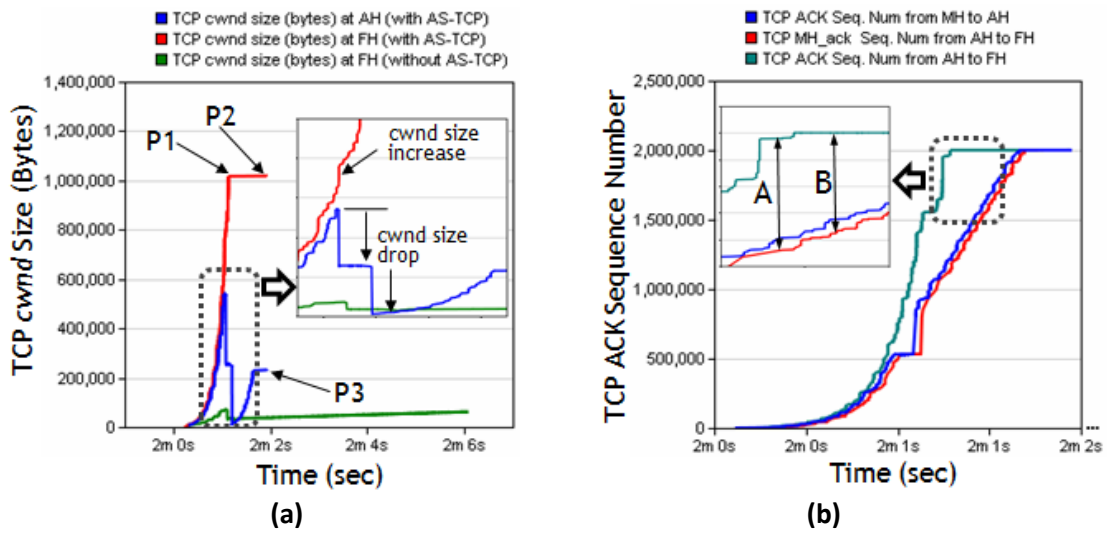


Figure 51: TCP ACK sequence number and MH ACK sequence number in the TCP option header from AS-TCP Host

control, while the ACK sequence in the TCP option header is to maintain the byte level end-to-end protocol semantics. The ACK sequence number from the MH is inserted in to the ACK packets belong to the TCP_{WAN} when available. This means that not all ACK sequence number from MH may be forwarded to the FH. If there was no ACK packet to be transmitted from AH to FH, then a timer based transmission is used. The ACK sequence number on the top flat green curve, where the vertical arrow headed line 'B' is pointing in Figure 51 (b), indicates that the AH uses the same ACK sequence number, while the MH_ack sequence number (the red curve) in the TCP option header may change. This is because while all data bytes have been delivered to the AH from the FH, the MH may still receive data from the AH. This notifies the FH to maintain the status of the byte level of data delivery to the MH, even after finishing all data transmission to the AH.

Figure 52 shows the TCP throughput performance gain with various file sizes and radio conditions when the RTT_{e2e} is split into (RTT_{RAN}: RTT_{WAN}) according to the following scenarios: (35ms:85ms), (60ms:60ms) and (85ms:35ms). The *rwin* is set to 128 Kbytes and the packet loss

rate in the RAN is set to 0.01% in all these experiments. This is exactly same scenarios used for the ES-TCP evaluation in the previous section.

As expected, these results in Figure 52 are very similar to the results with the ES-TCP mechanism as depicted in Figure 50, because the only difference between the ES-TCP and the AS-TCP mechanism proposed in this dissertation is the solution for preserving the end-to-end protocol semantics.

Comparing the throughput gains of the proposed AS-TCP against regular TCP, Figure 52 (a) corresponding to the TCP Split configuration (35ms:85ms) shows performance gains in the order of 20% to 55%. The highest throughput performance gain in favor of AS-TCP is 93%, and it is observed in Figure 52 (b) which is under the scenario with the TCP Split configuration (60ms:60ms), 20 Mbytes file size, and best radio conditions (MCS=25). The performance gain, however, varies significantly over a wider range of file sizes and radio conditions. Figure 52 (c) corresponds to the TCP Split configuration (85ms:35ms) and shows more limited performance gains in the order of 20% to 40%. This modest gain in Figure 52 (c) is due to the fact that the RAN delay is dominant and the RTT_{e2e} splitting point is relatively closer to the traffic source (FH). Figure 52 (d) is a comparison of the scenarios (a) and (b) by overlapping these two graphs together similar to Figure 50 (d).

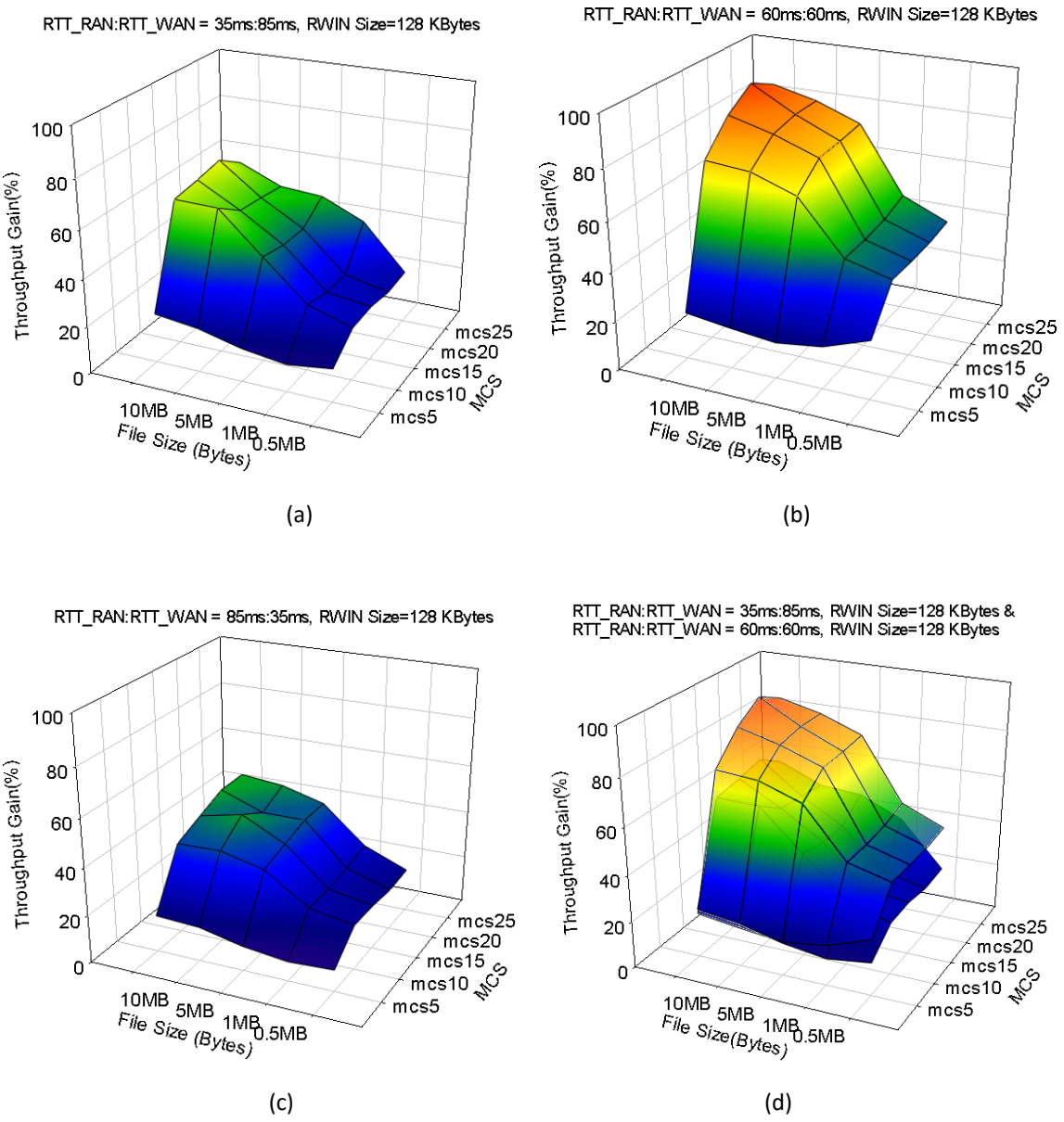
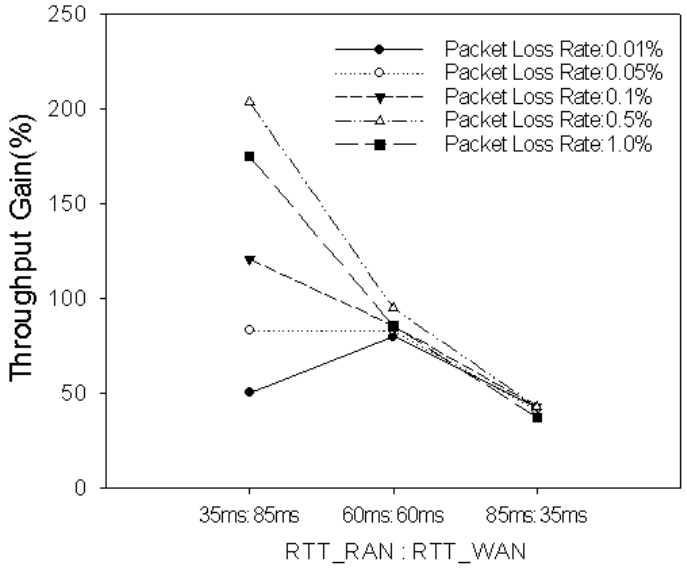


Figure 52: AS-TCP: Throughput gain as a function of radio conditions, file sizes and TCP Split configuration: (a) (35ms:85ms), (b) (60ms;60ms), (c) (85ms:35ms), (d) comparison of scenarios (a) and (b)

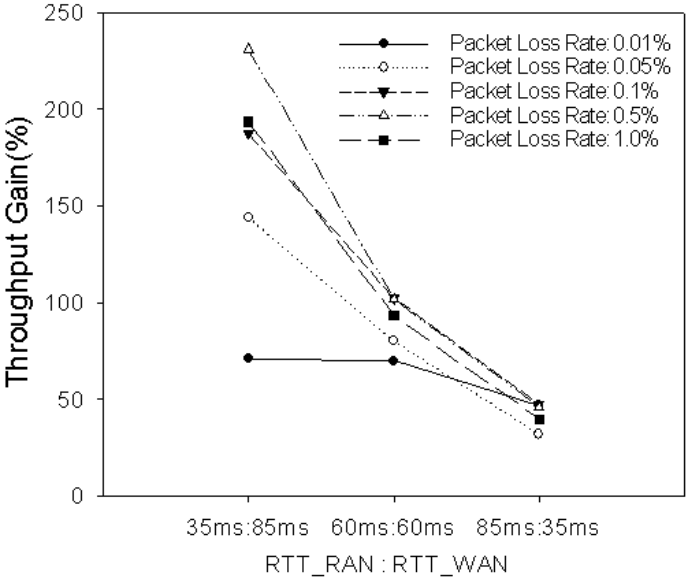
C. Throughput gain over wide range of network conditions

Figure 53 shows the throughput gain as a function of the packet loss rate in the RAN segment (0.01%, 0.05%, 0.1%, 0.5%, and 1.0%) and the three TCP Split configurations with ES-TCP enhancement. The result with AS-TCP is expected to be very similar. The throughput gain increases with the increase of packet loss rate in the RAN segment up to 0.5%, and then it gets decreased for the 1% case. The TCP packet traces indicate that higher packet error rates cause the TCP to time out in the TCP_{RAN}, which stalls the TCP transmission. As the stall duration becomes dominant with the higher packet error rate, the TCP throughput gain is reduced. The largest gain is achieved under the following configurations: (i) the TCP Split configuration (35ms:85ms), which indicates the preference for a RTT_{e2e} splitting point that is closer to the eNB, (ii) *rwin* size of 512 Kbytes, which indicates the preference for a larger *rwin* size, and (iii) MCS=25, which indicates the preference for the best radio conditions. The gain is visibly reduced when the RTT_{e2e} splitting point is closer to the FH.

This analysis clearly indicates that the TCP performance gains are dependent on several variables which may change in real-time (e.g., as a function of user location), such as user specific radio conditions and RTT_{e2e} split ratio. Thus, it is critical to select a robust TCP enhancement mechanism that provides solid gains over various network conditions. The proposed Split-TCP based enhancement mechanism provides at least 40% throughput gain, regardless of the network and usage conditions, as long as the RTT_{e2e} splitting point is not too close to the traffic source (FH). Furthermore, throughput gains larger than 200% can be achieved in the presence of larger packet error rate conditions in the TCP_{RAN} segment unless the RTO events stall the TCP connection.



(a)



(b)

Figure 53: ES-TCP: Throughput gain as a function of packet loss rates and TCP Split configuration: (a) MCS = 5 and *rwin* = 64 Kbytes, (b) MCS = 25 and *rwin* = 512 KBytes

5.7 Comparison of TCP Enhancement Mechanisms

Table 6 compares the ten key capabilities or requirements of the various TCP performance enhancement mechanisms. It is separated in two categories: non-split TCP based solution and split-TCP based solution. This Table 6 is same as Table 1 but comes with the two proposed split-TCP based solution in the last two columns: ES-TCP and AS-TCP. The Table 6 is self explanatory, but we highlight a few key differences of our proposed solutions from the existing enhancement solutions.

The M-TCP, M-TCP+ and PAM out of the split-TCP based solutions do not isolate or distinguish the wireless network from a wired network and do not split the RTT_{e2e} which is one of the key barrier for the TCP performance. This incapability is represented as incapability of supporting resiliency of high BER, high delay variation and dynamic link rate variation. The RLP, Snoop, and ELN mechanisms out of the non-split TCP based solutions neither have the capability of splitting the RTT_{e2e}. I-TCP solution has capability for these key items but it does not maintain the end-to-end protocol semantics which we address as a critical problem that needs to be resolved. On the other hand, the proposed solutions, ES-CTP and AS-TCP does maintain the end-to-end protocol semantics and it also effectively split the TCP_{e2e} in to two TCP_{WAN} and TCP_{RAN} so it can achieve the true performance gain from the Split-TCP mechanism.

In terms of comparison between these two proposed solutions, they are different in the following two aspects: (1) ES-TCP maintains the end-to-end protocol semantics at the TCP session level while the AS-TCP does it at the data byte level which is the same level as a regular TCP does, and (2) while ES-TCP solution does not require to modify the regular TCP implementation, the AS-TCP requires to extend the TCP to accommodate a TCP option to carry ACK sequence number from the MH to the FH.

Table 6: Comparison of TCP performance enhancement mechanisms including the proposed solutions

	Non-split TCP Solution			Split TCP Solution					
	RLP	**Snoop**	**ELN**	**I-TCP**	**M-TCP**	**M-TCP+**	**PAM**	**ES-TCP**	**AS-TCP**
Maintain End-To-End Protocol Semantics	Yes+	Yes+	Yes+	No	Yes+	Yes+	No	Yes*	Yes+
Require TCP Modification	No	No	Yes	Yes	Yes	Yes	No	No	Yes
Isolate/Distinguish Wireless Network	No	Yes	Yes	Yes	No	No	No	Yes	Yes
Split Long Network Delay (Split RTT)	No	No	No	Yes	No	No	No	Yes	Yes
Resilient to Lengthy Disconnection(Handoff)	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Capable of Data Pre-fetch	No	No	No	Yes	No	Yes	No	Yes	Yes
Resilient to High BER	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Resilient to High Delay Variation	No	Yes	No	Yes	No	No	No	Yes	Yes
Resilient to Link Rate Variation	No	No	No	Yes	No	No	No	Yes	Yes
Aware Of TCP	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

*: Maintain end-to-end protocol semantics at the TCP session level

+: Maintain end-to-end protocol semantics at the TCP data byte level

5.8 Conclusion

In this section we provided two Split-TCP based solutions, named ES-TCP and AS-TCP to resolve all three issues aforementioned: maintain the end-to-end semantics, isolate the link error in the wireless network, and split the long network delay. The ES-TCP maintains the end-to-end protocol semantics at the TCP session level but not at the every byte level which is done by the regular TCP. Thus, to maintain the end-to-end semantics flawlessly, we further proposed the AS-TCP that has more functions over the ES-TCP. We also quantified the TCP throughput performance improvement with the proposed TCP enhancements using the end-to-end application performance simulation platform for the LTE network.

Both of the proposed enhancements, significantly increase the TCP throughput while maintaining the end-to-end protocol semantics, unlike other solutions that either break the end-to-end semantics while improving throughput performance or degrade throughput performance while maintaining the end-to-end semantics.

Simulation results show that the throughput performance gain could be over 200% compare to the baseline TCP, depending on the network conditions and usage scenarios. It indicates that the throughput gain is increasing with the increase of packet loss rate in the RAN segment. However, the throughput gain significance start reducing when the packet loss rate is high because the high packet error rate may cause a TCP timeout in the TCP_{RAN}, and it stalls the TCP transmission. As the stall duration becomes dominant with the high packet error rate, the throughput gain starts reducing. The simulation also shows that the smaller throughput gain is expected under poor radio conditions because there may be no much unused bandwidth to improve on. Thus it is important maintaining the packet loss below a certain level to not to cause a significant TCP timeout event and utilizing network bandwidth efficiently so that can maximize the performance improvement using the ES-TCP and AS-TCP enhancement mechanisms.

CHAPTER 6

Conclusion

In this dissertation, we identified three critical problems that may negatively affect the application performance and wireless network resource utilization via application traffic measurement and packet analysis over a commercial LTE network and via LTE network simulation: (i) impact of the wireless MAC protocol on the TCP throughput performance, (ii) impact of applications on network resource utilization, and (iii) impact of TCP on throughput performance over wireless networks. We further proposed four enhancement mechanisms which improve the end-to-end application and wireless system performance. The work was carried out considering cross-layer protocols behaviors, due to the mutual impact of network protocol layers. To evaluate and quantify the TCP throughput performance gains using ES-TCP and AS-TCP, we implemented them on top of an end-to-end LTE network simulation platform that can estimate the application performance reflecting various protocol interactions and network congestion conditions, including air-link impairments.

The two novel Split-TCP mechanisms, the ES-TCP and the AS-TCP, improve significantly the TCP throughput without breaking the end-to-end TCP semantics. Experimental results show that the proposed ES-TCP and AS-TCP can boost the TCP throughput by more than 60% in average when exercised over a commercial 4G LTE network. Furthermore, the TCP throughput performance improvement may be even over 200%, depending on network and usage conditions. The simulation results show that the performance enhancement with the proposed solutions is significant and the improvement is noticeable even with a higher air-link data rate available. While some prior solutions maintain the end-to-end semantics and isolate the connectivity issue in the wireless network, they present a few significant drawbacks: the packet loss on wireless network is propagated into the wired network instead of isolating the problem to the wireless network segment; most of all, those solutions lost the critical benefit of Split-TCP, which consists in

shortening the round trip time for TCP connections. Shortening the RTT by itself improves the TCP throughput performance significantly, even without link condition changes.

The advantage of the proposed ES-TCP and AS-TCP consists in the fact that one does not lose any performance benefits which are inherent to the Split-TCP mechanism, while preserving the end-to-end protocol semantics. Therefore, they make the Split-TCP mechanism reliable over wireless media. However, we also identified two important conditions to maximize the performance improvement using the ES-TCP and AS-TCP enhancement mechanisms: (1) maintaining a low TCP timeout event and (2) utilizing network bandwidth efficiently.

The proposed LTE uplink resource allocation mechanism in chapter 3 could reduce network delay and potentially prevent TCP timeout events, while the novel TCP snooping mechanism proposed in Chapter 4 could save about 20% of wireless network resources by preventing unnecessary packet transmission through air interface, taking as reference video traffic downloads. Considering the amount of video traffic over wireless network (i.e. about 60% of total wireless traffic), the overall bandwidth saved in downlink is about 12%. The saved air-link bandwidth may be utilized through the performance enhancements attributed to ES-TCP and AS-TCP. We expect that these proposed Split-TCP enhancement mechanisms, together with the novel uplink resource allocation enhancement and the novel TCP snooping mechanism may provide even greater performance gains when advanced radio technologies, such as 5G, are deployed.

BIBLIOGRAPHY

[1] 3GPP, “Long Term Evolution – The Mobile Broadband Standard”, http://www.3gpp.org/LTE

[2] 3rd Generation Partnership Project, “Evolved Universal Terrestrial Radio Access (E-UTRA), Medium Access Control (MAC), Protocol Specification,” 3GPP TS 36.321, <http://www.3gpp.org/ftp/Specs/html-info/36321.htm>

[3] 3rd Generation Partnership Project, “Evolved Universal Terrestrial Radio Access (E-UTRA), Radio Resource Control (RRC), Protocol Specification,” 3GPP TS 36.331, <http://www.3gpp.org/ftp/Specs/html-info/36331.htm>

[4] M. Allman, V. Paxson, and W. Stevens, "RFC 2581: TCP Congestion Control," April 1999

[5] M. Amjad, B. Aslam, and C. C. Zou, “Transparent Cross-Layer Solutions for Throughput Boost in Cognitive Radio Networks,” IEEE CCNC 2013.

[6] F. Anjum and R. Jain, “Performance of TCP over lossy upstream and downstream links with link level retransmissions,” in Proc. ICON, 2000, pp. 3–7.

[7] A. Bakre and B. R. Badrinath, “I-TCP: Indirect TCP for mobile hosts”, In Proc. of the 15th IEEE International Conference on Distributed Computing Systems, pp. 136-143, Vancouver, BC, May 1995.

[8] H. Balakrishnan and R. Katz, “Explicit loss notification and wireless web performance,” In Proceedings Globecom Internet Mini-Conference, Sydney, Australia, November 1998.

[9] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, “A comparison of mechanisms for improving TCP performance over wireless links,” ACM SIGCOM’96, August 1996.

[10] H. Balakrishnan, S. Seshan, and R. H. Katz, “Improving reliable transport and handoff performance in cellular wireless networks,” ACM/Baltzer Wireless Networks J., vol. 1, no. 4, Dec. 1995.

[11] R. Bestak, P. Godlewski and P. Martins, “A TCP Connection over Uplink UMTS Radio Access Bearer in RLC Acknowledgement Mode,” IEEE Conf. on Vehicular Technology (VTC 2002 Spring), May 2002

[12] H. Bischl, H. Brandt, A. Dreher, M. Emmelmann, A. Freier, B. Hespeler, F. Krepel, E. Lutz,W. Milcz, L. Richard, P. Todorova, and M. Werner, “ATM-sat: ATM-based multimedia communication via LEO-satellites – system architecture report,” German Aerospace Agency (DLR), Tech. Rep., 2000.

[13] K. Brown, and S. Singh, “M-TCP: TCP for mobile cellular networks,” Computer Communication Review (a publication of ACM SIGCOMM), volume 27(5), October 1997

[14] D. Calin, and B. Kim, “LTE Application and Congestion Performance” Bell Labs Technical Journal, vol. 18, no 1, June 2013, pp. 5-25

[15] C. Casetti, M. Gerla, S. Mascolo, M.Y. Sanadidi and R.Wang, “TCP Westwood: Bandwidth estimation for enhanced transport over wireless links,” In Proceedings of ACM Mobicom, pp. 287-297, Rome, Italy, July 2001.

[16] Center for Applied Internet Data Analysis (CAIDA) 2013, “Analyzing UDP usage in Internet traffic,” accessed 1 July 2015, < https://www.caida.org/research/traffic-analysis/tcpudpratio/>

[17] R Chakravorty, S Katti, J Crowcroft, and I Pratt, "Flow Aggregation for Enhanced TCP over Wide-Area Wireless," Proceeding of I_FOCOM 2003, Apr. 2003, pp. 1754-1764.

[18] M. C. Chan and R. Ramjee, “TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation,” in Proc. ACM Mobicom 2002

[19] A. Chockalingam, M. Zorzi, and V. Tralli, “Wireless TCP performance with link layer FEC/ARQ,” in Proc. ICC, 1999, pp. 1212–1216.

[20] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, “RFC 6928: Increase initial window,” April 2013

[21] Cisco, “Visual Networking Index: Global Mobile Data Traffic Forecast Update 2015–2020,” Feb., 2016

[22] N. Dukkipati, et al, “An Argument for Increasing TCP’s Initial Congestion Window (with appendix)”: https://developers.google.com/speed/articles/tcp_init*cwnd*_paper.pdf

[23] A. Finamore, M. Mellia, M. M. Munaf`o, R. Torres, and S. G. Rao, “YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience,” in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, ser. IMC, Berlin, Germany, Nov. 2011.

[24] S. Floyd, "RFC 2914: Congestion Control Principles," September 2000

[25] S Floyd, T. Henderson, and A. Gurtov, “RFC 3782: The NewReno Modification to TCP’s Fast Recovery Algorithm,” April 2004

[26] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, “RFC 2883: An Extension to the Selective Acknowledgement (SACK) Option for TCP,” July 2000

[27] M. Fomenkov, K. Keys, D. Moore, and K, claffy, "Longitudinal study of internet traffic in 1998-2003," in WISICT, 2004.

[28] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, “RFC 6824: TCP Extensions for Multipath Operation with Multiple Addresses,” January, 2013

[29] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, “Packet-level traffic measurements from the sprint ip backbone,” Network, IEEE, vol. 17, no. 6, pp. 6–16, 2003.

[30] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, “YouTube Traffic Characterization: A View From the Edge,” in Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ser. IMC, San Diego, California, USA, Oct. 2007.

[31] T. Goff, et.al, “Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile

environments," IEEE JOURNAL, INFOCOM'2000.

[32] T. Goff, J. Moronski, D. Phatak, and V. V. Gupta, "Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments," In Proceedings of IEEE INFOCOM, March 2000.

[33] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," ACM Vol. 42 Issue 5, July 2008

[34] M. A. Hoque, M. Siekkinen, J. K. Nurminen, and M. Aalto, "Investigating Streaming Techniques and Energy Efficiency of Mobile Video Services," omputing Research Repository - arXiv:1209.2855, 2012.

[35] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in Proceedings of the 2012 ACM conference on Internet measurement conference, ser. IMC, Boston, Massachusetts, USA, Nov. 2012.

[36] R. Jain, and Teunis J , "Design and Implementation of Split TCP in Linux Kernel," IEEE NJIT 2006

[37] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in ACM IMC, 2007

[38] F. Khan, S. Kumar, K. Medepalli and S. Nanda, "TCP Performance over cdma2000 RLP," Proc. IEEE Conf. on Vehicular Technology (VTC 2000 Spring), May 2000

[39] B. Kim and D. Calin, "Method And Apparatus For Controlling Buffer Status Report Messaging," US Patent 8630202, 2014.01.14.

[40] B. Kim, I. Lee and K. Chu, "End-to-end Application Performance Impact on Scheduler in CDMA-1XRTT Wireless System," VTC 2005 spring

[41] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A Comparative Study of Android and iOS for

Accessing Internet Streaming Services," in Proceedings of the 14th international conference on Passive and Active Measurement, ser. PAM, Hong Kong, China, Mar. 2013.

[42] H. Nam, B. Kim, D. Calin, and H. Schulzrinne, "Mobile Video is Inefficient: A Traffic Analysis," December, GlobeCom 2013

[43] H. Nam, K.H Kim, B. Kim, D. Calin and H. Schulzrinne, "Towards A Dynamic QoS-aware Over-The-Top Video Streaming in LTE," WoWMoM 2014

[44] S. Mascolo, et.al. "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links". IEEE2000.

[45] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "RFC 2018: TCP selective acknowledgement options," October 1996

[46] J. Moon et.al, "Rate-Adaptive Snoop: A TCP Enhancement Scheme over Rate-Controlled Lossy Links". IEEE 2002.

[47] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," SIGOPS Oper. Syst. Rev., 44(3):2–19, August 2010.

[48] M. Omueti and L. Trajkovic, "M-TCP+: using disconnection feedback to improve performance of TCP in wired/wireless networks," in Proc. SPECTS 2007, San Diego, CA, USA, July 2007, pp. 443-450.

[49] T. Ott, J. H. B. Kemperman, and M. Mathis, "The stationary behavior of ideal TCP congestion avoidance," August 1996.

[50] A. Pathak, Y. A.Wang. A., C. Huamg, A. Greenburg., C. Hu, R. Kern, J. LI, and K.W. Ross, "Measuring and evaluating TCP splitting for cloud services," PAM (2010).

[51] V. Paxon, M. Allman, J. Chu, and M. Sargent, "RFC 6298: Computing TCP's Retransmission Timer," June 2011.

[52] L. Philippe, "EU Initiative On 5G Status Report," Oct., 2013.

[53] K. Ramakrishnan, and S. Floyd, “RFC 2481: A Proposal to add Explicit Congestion Notification (ECN) to IP,” January 1999.

[54] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, and W. Dabbous, “Network Characteristics of Video Streaming Traffic,” in Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies, ser. CoNEXT, Tokyo, Japan, Dec. 2011.

[55] RFC 793: “Transmission control protocol,” Internet Engineering Task Force (IETF), September 1981.

[56] W. R. Stevens, “TCP/IP Illustrated: The Protocols,” Addison-Wesley Professional Computing Series. Massachusetts, USA: Addison Wesley Longman, Inc., 1994, vol. Volume 1.

[57] W. R. Stevens, “TCP Slow start, congestion avoidance, fast retransmit and fast recovery algorithms, RFC 2001,” January 1997

[58] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, “Answering What-If Deployment and Configuration Questions with WISE,” In: ACM SIGCOMM. (August 2008)

[59] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro, “Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless,” Technical Report, Computer Science Dept., Texas A&M University, February 1999.

[60] J.W. K.Wong and V. C. M. Leung, “Improving end-to-end performance of TCP using link-layer retransmissions over mobile internetworks,” in Proc. ICC, 1999, pp. 324–328.

[61] www.speedtest.net

[62] W. Wei, C. Zhang, H. Zang, J. Kurose, and D. Towsley, “Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks,” Proc. of the ACM PAM, 2006.

[63] M. Zink, K. Suh, Y. Gu, and J. Kurose, “Watch Global, Cache Local: YouTube Network

Traffic at a Campus Network – Measurements and Implications," Computer Science Department Faculty Publication Series. Paper 177, 2008.