UNIVERSITY OF CALIFORNIA
SANTA CRUZ

Efficient TCP enhancements for Hybrid Networks

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER ENGINEERING

by

Ramesh Srinivasan
June 2016

The Thesis of Ramesh Srinivasan
is approved:

_____
Professor J.J. Garcia-Luna-Aceves, Chair

_____
Professor Brad Smith

_____
Professor Pat Mantey

_____
Tyrus Miller
Vice Provost and Dean of Graduate Studies

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Efficient TCP enhancements for Hybrid Networks

By

Ramesh Srinivasan

The original TCP design and implementation still has widespread deployment today among the devices in the Internet, including various end-devices/clients ranging from main-frames, large servers, desktops, laptops, palm-tops, notebooks, notepads, smart-phones, tablets and now various IoT (Internet of Things) devices. The TCP architecture and design, most of it put in place in the 1980s and shortly thereafter prior to the Internet boom, was intended solely for a wired network. One of the main in-built assumptions, due to the above design, is that any packet drops or delays are interpreted as solely due to a congestion in the network and corresponding adjustments/changes are made by the protocol. However, with the advent of wireless networks and their co-existence with existing wired Internet deployment, the legacy TCP implementation ends up inadvertently responding to packet losses and increased latencies, due to wireless-link errors, as if it were due to network-congestion. This results in significantly reduced throughput and efficiencies.

WSC-TCP (Wireless Santa Cruz - TCP) proposed here, is an efficient TCP enhancement for today's hybrid networks, which have both wireless as well as wired-links. It is a culmination of attempts to arrive at a framework to enable the TCP stack to differentiate those delays and losses, which are due to wireless-link-layer issues versus those that are due to actual network-

congestion issues. The effort involved a study of the different mechanisms in the existing TCP design that do not work as per the original intent of responding only to network-congestion, due to its reliance on packet RTT (Round trip time) and related metrics for detecting and sizing the network congestion in today's hybrid networks (wireless and wired links). The simulation tests conducted with some of the proposed enhancements result in an overall improved throughput/performance by 10-15% compared to TCP New-Reno in a hybrid network (wired and wireless links).

DEDICATION

To my parents
Dr. Rajalakshmi Srinivasan and K S Srinivasan

for their unceasing belief and faith in me through the ups and downs and always being there with boundless encouragement and inspiration

To my wife
Srividya Ramesh

for her constant ongoing support through this endeavor and always keeping me focused on making progress and being a very good critic in things and highlighting aspects I might have missed and more importantly bringing a constant refreshing sense of humor and cheer along this journey

To my daughters
Shreya Ramesh and Sathya Ramesh

for putting up with many endless hours when I would have to shut myself off for doing my research work and studies and at the same lending a helping hand by getting involved in simple technical discussions and getting an understanding of the area dad has been spending his late nights and weekends

To my sister

Dr. Jayanti Ravi

for herself excelling in many walks of life and always having a cheerful disposition and encouraging word or two, during our phone conversations, which mean a lot

AND last but not the least

To my maternal grandfather

Late K. Subramanian

for the countless hours he spent with me decades ago when I was in high school working closely with me as I was getting introduced to science and math and instilled in me a passion and a love for quest of knowledge and the belief that yes, I can truly do it and make it happen.

ACKNOWLEDGEMENTS

This work could not have been completed without the help, guidance and support from my advisor Professor J. J. Garcia Luna, whose technical guidance, coupled with his mild and calm demeanor and subtle sense of humor, serves as a tremendous source of motivation and inspiration.

I would like to acknowledge my colleagues at CCRG over the years, who have all been very supportive with their feedback and help as required. I would like to place on record my deep gratitude to Professor Pat Mantey for encouraging me to join the graduate program at UCSC, in addition to reviewing my work closely.

I would like to thank Professor Brad Smith for working with me in a research proposal submission to Cisco Systems few years ago and the other networking faculty at SOE, Professor Katia Obraczka and Professor Cedric Westphal, who provided the inspiration and intellectual stimulation during the many discussions in the Networking Seminar class over the years.

I would like to thank my colleagues at Cisco Systems, where I worked for nearly 14 years during my stint in the networking industry and subsequently my team members at my startup company, Mayosys Solutions for shaping my growth and fascination with networking. This culminated in my choosing computer networks as my focused area of research at the graduate program at UCSC.

Last but not the least, the SOE staff Emily Gregg and Carol Mullane (earlier) at UCSC who have been so amazing in helping, coordinating and putting everything together for us.

1. INTRODUCTION

The Internet today is a global system of interconnected computer-networks that use the standard Internet protocol suite (TCP/IP) to link several billion devices worldwide. It is a *network of networks* that consists of millions of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical-networking technologies. The Internet carries an extensive range of information resources and services, such as the inter-linked hypertext-documents and applications of the World Wide Web (WWW), the infrastructure to support email, and peer-to-peer networks for file sharing and telephony.

Continuous advances in communication, hardware and software technologies, coupled with the ever increasing need for ubiquitous access to the Internet and intranets, have exploded the various different systems, platforms and end-devices which are developed and in-use today. One of the central requirements is the capability to securely access data and computing resources anytime, anywhere in an efficient manner. The proliferation of wireless devices including, but not limited to laptops, tablets, smart-phones and various other devices, has created a new impetus to address some of these requirements.

The latest scale of development and deployment trends for the IoT (Internet of Things) and Machine-to-Machine (M2M) communication space, involving complete usage of wireless technology for last-mile connectivity, coupled with the necessity to be able to transport through the wired-networks to access the cloud and IT infrastructure, mandate the urgent need for a comprehensive review of the design and architecture of the basic transport protocol TCP used ubiquitously. In this work, we propose means/mechanism for optimizing the

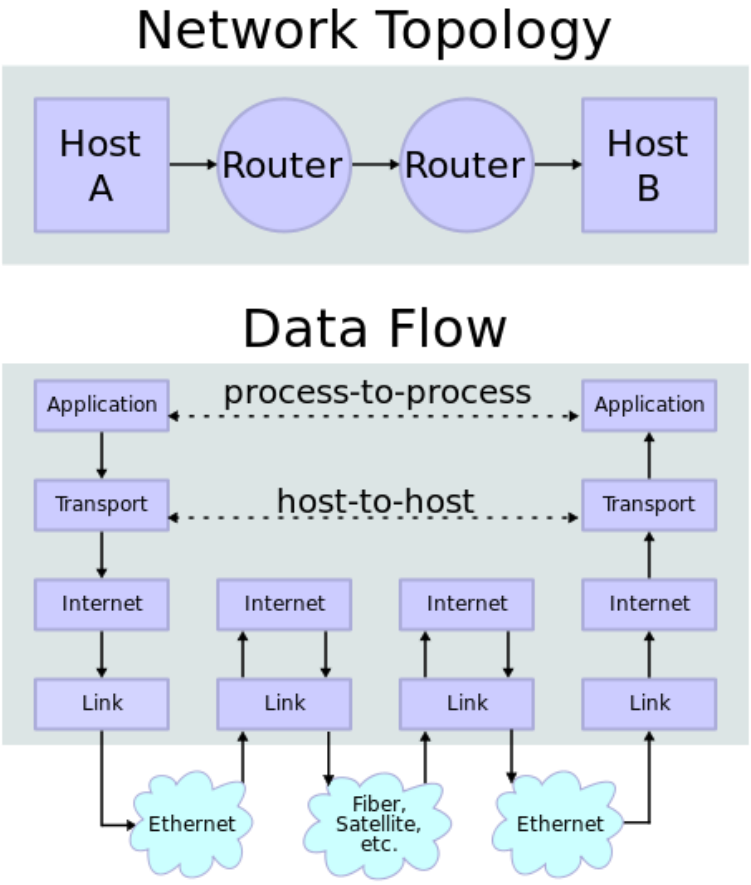
communication and end-end throughput for today's networks which have significant and growing number of wireless end-devices like smart-phone, tablets and IoT devices and thus ensuring the best optimal and efficient use of the networking resources and the Internet.

Traditional host-centric abstractions like Transmission Control Protocol over Internet Protocol (TCP over IP) are not efficient with mobile end-devices which correspondingly bring varying wireless signal-strength and connectivity including, at extreme, scenarios with sporadic breakage of wireless-link-layer last-mile communications. The traditional implementations of TCP have been designed for host-to-host communication in a connected static-network, with short round trip times between hosts, per-packet switching of fixed-length messages, bidirectional connectivity with mostly symmetric rates and low packet loss. In such wired and connected-network scenarios, it can be accurately asserted that if and only if network-congestion occurs, will there be round-trip delays.

TCP-IP (Transmission Control Protocol – Internet Protocol) is the main underlying protocol/framework and is also deemed the backbone of the Internet. The initial Internet architecture and the design of the protocols driving it, including TCP-IP, was based on the model depicted in the picture below (Figure 1) wherein, the last-mile access is assumed to be a wired connection (Ethernet in this example). Thus, it does not consider scenarios of wireless-last-mile links and the corresponding intermittent-wireless-link errors resulting from RSSI (Received Signal Strength Indicator) fluctuations due to transient-temporary-wireless-link conditions.

Figure 1

The original design goal of TCP was to transport packets reliably on an end-end basis, and at



the same time, utilize the bandwidth available to the best and in the most-efficient manner possible. To this end, the bandwidth-delay-product was used to arrive at the overall size of the pipe (end-end connection) and thus, determine the size of the sliding window of packets with outstanding acknowledgements (ACKs).

As part of the above intent, the end-end delay observed and packet drops were used as a metric and a measure of the congestion in the network. This assumption is valid in wired networks wherein, the packet losses or delays are typically only due to congestion. Thus, as stated in earlier section, in traditional-wired networks if-and-only-if there is network congestion, is there an increase in RTT (Round trip time) and delays in packets transportation. Hence, TCP design and implementation in traditional-wired networks correctly interprets delays in packet transportation and corresponding increase in RTT (Round Trip Time), as a sign of congestion in the network.

In today's evolving technological world, with the evolution (miniaturization) of computer technology, along with the rapid adoption and maturing of the wireless technologies, particularly the 802.11 a/b/c/n technologies for the local-wireless LAN, there are innumerable number of wireless laptops, notebooks, tablets, smart-phones and notepads. Their numbers are growing at a rate never before seen in the Information Technology (IT) industry and they are all connecting to the Internet. Thus, the Internet today is a hybrid network with a significant number of the last-mile-wireless links. These trends make it all the more important for us to consider the Internet as a hybrid network. Therefore, we need to take a fresh look into the design of TCP, to ensure that its behavior is consistent with the new Internet, which integrates wired and wireless networks co-existing together, which we call "hybrid networks".

In hybrid networks with last-mile-wireless links, the original design premises used for TCP are no longer accurate. Two approaches could be pursued, to address the operation of TCP in hybrid networks:

a) Rewrite of the TCP implementation from scratch.

b) Identify mechanisms to differentiate the delays caused by the wireless-link-layer issues from those caused by the core-network congestion. The second approach would enable leveraging existing TCP implementations, while ensuring the usage of the new-corrected values/measures for RTT times, which account only for those delays caused by the core-wired network and not those delays introduced by the last-mile-wireless links. This is the approach taken in this work.

This Thesis describes WSC-TCP (Wireless Santa Cruz – Transmission Control Protocol), which consists of the following four mechanisms to make TCP work more efficiently over a hybrid network.

- A new protocol between the link layer and transport layer is proposed, wherein the number of times, a given segment undergoes link-layer retransmission, is communicated back to the TCP layer.
- A mechanism is proposed to dynamically adjust the number of duplicate acknowledgements (Dup-Acks), required to trigger fast retransmits, based on the segments requiring link-layer retransmissions.
- A “best-start” TCP window size is used for each new TCP session to facilitate quick initial convergence to the appropriate window size, for that TCP session.

- A cluster-analysis mechanism is used to identify and address delays which are sporadic, brief, and truly temporary as well as those that are due to link-layer retransmissions.

TCP New-Reno was chosen as the baseline for comparison as it provides better throughputs in hybrid networks, compared to other existing TCP implementations. As observed in our simulation reports, WSC-TCP provides a better over-all throughput compared to TCP New-Reno. This is due to the start of each TCP session with an appropriate initial TCP sliding window size, as well as correctly recognizing the delays, caused by network congestion.

Section 2 discusses prior related work to improving TCP for wireless and hybrid networks and compares those approaches to ours. Section 3 describes the algorithms of our proposed TCP enhancements and shows scenarios of their operation. Section 4 shows via simulation the performance improvements obtained with WSC-TCP over TCP NewReno implementation. Finally section 5 summarizes our results.

2. RELATED WORK

Congestion control for TCP, particularly in the context of wireless networks, has been an area of active research. Outlined below is a survey of the various current deployed versions of TCP and the way they handle the addition/introduction of wireless links:

2.1 Survey of Existing TCP Implementations:

A concise comparative study of the actual approaches used in the different TCP implementations can be found in paper titled "An Overview of Performance Comparison of Different TCP Variants in IP and MPLS Networks" [30].

Here we outline just some of the salient issues with these implementations:

2.1.1 Tahoe

The problem with Tahoe is that it takes a complete-timeout interval to detect a packet loss and in fact, in most implementations, it takes even longer because of the coarse-grain timeout. Also, since it doesn't send immediate ACK, rather sends cumulative acknowledgements, it follows a go-back-n approach. Thus whenever a packet is lost, it waits for a timeout and the pipeline is emptied. This results in a major cost in links with high band-width-delay product.

2.1.2 Reno

Reno performs very well over TCP when the packet losses are small. But when there are multiple packet losses in one window, then RENO doesn't perform very well and its performance is almost the same as Tahoe under conditions of high packet loss. The reason is that it can only detect a single packet loss. If there are multiple packet drops then the first information about the packet loss comes when we receive the duplicate ACKs. But, the

information about the second packet which was lost will come only after the ACK for the re-transmitted first segment reaches the sender after one RTT. Also, it is possible that the CWD is reduced twice for packet losses, which occurred in one window. Another problem is that, if the window is very small when the loss occurs then we would never receive enough duplicate acknowledgements for a fast-retransmit and we would have to wait for a coarse-grained timeout. Thus, it cannot effectively detect multiple-packet losses.

2.1.3 NewReno

NewReno suffers from the fact that it takes one RTT to detect each packet loss. When the ACK for the first re-transmitted segment is received, only then can we deduce which other segment was lost. However, it addresses some of the issues with the Reno approach and eliminates Reno's wait for a retransmit timer when multiple packets are lost within a window by responding to partial-acks, without taking the sender out of fast-recovery.

2.1.4 TCP-SACK

The biggest problem with SACK is that selective acknowledgements are not provided by the receiver. If the TCP changes are localized to the sender side alone, then it is easier to deploy and support and get the needed benefits.

2.2 Issues in the existing implementations

Packet loss in wireless networks may be due to bit errors, handoffs, congestion, and reordering. By design, TCP assumes packet loss is due to congestion. TCP's congestion responses are triggered by wireless-packet losses and the ensuing delay (due to corresponding link-layer retransmission times), which are actually caused by inherent transient issues for packet transmission through a wireless media and not due to a congestion in the network. The corresponding adverse impact, to the effective end-end throughput, is significant.

With the tremendous ongoing increase in number of wireless clients, this is a very significant issue and a good robust solution which is easy to implement and deploy will have a big positive impact on a significant portion of the actual end-users (of laptops, PDA and other devices which use wireless connectivity for access to the network/Internet).

2.3 Survey of the efforts to enhance TCP

The approaches proposed to improve TCP performance over networks with wireless links can be divided into **two major categories**,

i. ones that work at the transport level, and
ii. others that work at the link level.

Transport level proposals include Explicit Bad State Notification (EBSN) [2], Freeze-TCP [8], Indirect-TCP (I-TCP) [1], Snoop [3], fast-retransmission [6], and WTCP [13]

Like other approaches, EBSN uses local retransmission from the base station, to shield wireless-link errors and improve throughput. However, if the wireless link is in error state for

an extended duration, the source may timeout causing unnecessary source retransmission. The EBSN approach avoids source timeout by using an explicit feedback mechanism. The EBSN message causes the source to reinitialize the timer.

Upon detecting a poor signal strength, Freeze-TCP [8] at the mobile host throttles the sender by advertising a receive window size of zero. This causes the sender to enter persist mode and freeze all the timers and window sizes. This way, the mobile host can prevent the sender from taking any congestion control measures

I-TCP splits the transport link at the wireline–wireless border. The base station maintains two TCP connections, one over the fixed network, and another over the wireless-link. This way, the poor quality of the wireless-link is hidden from the fixed network. By splitting the transport link, I-TCP does not maintain end-to-end TCP semantics, i.e., I-TCP relies on application layer to ensure reliability…

The fast-retransmission approach does not address the issue of wireless-link reliability, but reduces the effect of mobile host hand-off. Immediately after completing the hand-off, the IP in the mobile host triggers TCP to generate a certain number of duplicate acknowledgments. This causes the source to retransmit the lost segment without waiting for the timeout period to expire. This requires modification to the TCP code at the mobile host.

Snoop [3] is a well-known link-level protocol. In this scheme, the base station sniffs the link interface for any TCP segments, destined for the mobile host, and buffers them, if buffer space is available. Segments are forwarded to the mobile host only if the base station deems it necessary.

Source-retransmitted segments that have already been acknowledged by the mobile host are not forwarded by the base station. The base station also sniffs into the acknowledgments from the mobile host. If the base station sees a duplicate acknowledgment, it detects a segment loss and locally retransmits the lost segment, if it is buffered and starts a timer. If the retransmitted segment is not acknowledged within twice the round-trip-time of the wireless-link, the segment is again retransmitted. Unlike I-TCP, Snoop does not completely shield the wireless-link losses from the fixed network, and source timeout is still possible. In particular, if acknowledgments are lost on the wireless link, a base station retransmission cannot occur as there are no duplicate acknowledgments, and the source can timeout and source retransmission takes place. The transmission over the wireless link resumes only after the arrival of the source-retransmitted segment. Hence, in the presence of burst losses, the throughput will be poor compared to I-TCP.

In WTCP [13], the base station is involved in the TCP connection. The conceptual view of the transport link is shown in Figure 1. WTCP requires no modification to the TCP code that runs in the mobile host or the fixed host. Based on duplicate acknowledgment or timeout, the base station locally retransmits lost segments. In case of a timeout, by quickly reducing the transmission window, potentially wasteful-wireless transmission is avoided and the interference with other channels is reduced. Also, WTCP hides the wireless-link errors from the source, by effectively subtracting the residence time of the segment at the WTCP buffer from the RTT value computed at the source, thus the RTT computation excludes wireless link-layer retransmission delays. However, this approach requires the base station to know the clock granularity of the source, because the timestamp field in the TCP header contains a

clock tick value, not the real-time clock value. This is an issue because many of the wireless-end nodes might not be in exact clock synchronization with the base station (particularly after a reboot of the end-node) and this is a limitation. In the IoT and M2M domains, wireless end-nodes need not have their clocks synchronized with the network, thus making this approach not applicable.

Thus, WTCP approach has a number of limitations that prevent it being deployed effectively. Additionally, it does not address some of the requirements of IoT/M2M devices, which need high throughput for small bursty traffic.

3. WSC-TCP - PROTOCOL DESCRIPTION

WSC-TCP provides an improvement over existing TCP implementations including TCP NewReno in four major areas.

1. For identifying a true congestion control in a more accurate manner, a new protocol between link-layer and transport layer is proposed. Here, the number of times a given segment undergoes link-layer retransmission is communicated back to the TCP layer. This helps to accurately identify the delays caused due to link-layer issues as against delays caused due to a true congestion in the network. Thus, it avoids inadvertent reduction of TCP window size.

2. Similarly, in current TCP implementations, the fast-retransmit happens when three duplicate acknowledgements (Dup-Acks) are received. This response assumes that network-congestion is the cause of the Dup-Acks. However, a segment undergoing a link-layer retransmission could also result in the generation of multiple Dup-Acks, until it reaches the destination, since subsequent segments (which did not need any link-layer retransmission) would have reached the TCP session end-point before this segment, which required link-layer retransmission(s). To circumvent this issue, in this work, the number of Dup-Acks required is dynamically adjusted based on any pending segment(s) requiring link-layer retransmission(s). This prevents causing inadvertent increase of load and congestion of the network.

3. Another enhancement here is a proposed mechanism for choice of an apt starting TCP window size for each new TCP session to facilitate quick initial convergence to the appropriate window size for this TCP session. This would ensure good throughput and utilization of the network and improved latency, particularly for short duration TCP traffic.

4. We also identify and address network transients (delays which are sporadic, brief and truly temporary) using cluster analysis and thus avoid factoring in some of these one-time transients from impacting the overall response of the TCP state machine. This

helps prevent TCP, from inadvertently reducing its window-size and causing an overall throughput decrease due to an isolated occurrence of a transient delays.

Once a TCP session is established, there are several causes for a low TCP throughput or reductions in TCP throughput in the presence of wireless- link-layer issues and other transient network conditions not related to actual network congestion. These causes include:

Cause A: Increases in round trip time (RTT).

Cause B: The value of the round trip timeout (RTO) being shorter than the observed RTT.

Cause C: Independently of the available bandwidth, TCP initiates the size of the window size of a connection using the slow-start mechanism, which assumes an initial starting window size of one segment.

Cause D: The occurrence of fast-retransmits as a result of three duplicate acknowledgments (ACK) being received.

Cause E: Transient fluctuations in RTT.

The goal of WSC-TCP is to address the negative impact of the above causes of low throughput and increased latency in TCP. WSC-TCP provides an improvement over existing TCP implementations including TCP NewReno in four major areas.

3.1 New Protocol between wireless link-layer and TCP

This mechanism addresses causes (A) & (B) of low throughput and degradation of existing throughput of TCP sessions in current TCP implementations.

The wireless link-layer is enhanced to communicate the fixed link-layer retransmission timeout value to the TCP layer at the beginning of each session. Thereafter, the link-layer

keeps track of the number of link-layer retransmissions for every segment and sends the sequence number and the number of link-layer retransmissions required for that segment back to the TCP layer. This information is used by the TCP layer to accurately estimate the RTT delays for this segment due to network congestion alone. The algorithm used is the following:

1. *Communicate the fixed link-layer retransmission timeout (LL_RTO) value to the TCP layer (one-time per TCP session)*
2. *For every segment/frame transmitted, Keep track of the exact number (n) of link-layer retransmissions required*
3. *Send the Sequence number of successful link-layer (re)transmitted segment, back to the TCP layer along with the exact number of retransmissions (n) required for it.*
4. *TCP Layer uses the above information and computes (n)* LL_RTO*
5. *Corrected RTT (from Network Congestion perspective) for each segment = Actual end-end observed RTT time for that segment – (n)*LL_RTO;*

3.2 Dynamic Dup_Ack calculation

This mechanism addresses cause (D) of low throughput and degradation of existing throughput of TCP sessions, in current TCP implementations.

Instead of the hard-coded response of the TCP implementation to initiate a fast retransmit on receipt of three DUP-ACKs, a method is proposed to calculate the number *(n)* of DUP_ACKs, upon the receipt of which the fast retransmit should be initiated. The approach used to arrive at the appropriate value of *(n)* involves discounting those segments which need retransmission at the wireless link layer.

The Algorithm used to determine the value of *(n)* that should trigger fast-retransmits is the following:

Definitions:

L_ack: Prefix to denote Local Link layer Ack

L_ack_info[]: array indexed by segment numbers and has a value of 0

(if L_ack recd for that segment) OR 1 (if L_ack is pending for that segment).

L_ack_current: segment number of current (latest) link-layer Ack received

L_ack_ max: Greatest segment number of link-layer Ack received, so far

L_ack_cum: Cumulative Segment number, up to which all segments have been transmitted successfully through link-layer

Dup_ack: Number of Dup_acks (End-end) permitted before fast retransmit algorithm kicks in

DEFAULT_DUP_ACK: Default Dup_ack value defined as 3.

Actual Algorithm for dynamic Dup_Ack calculation

Pseudo Code – Dup Ack Computation

```
DupAck_Compute() {
If (L_ack_current > L_ack_max) {
        L_ack_max  L_ack_current;
        L_ack_info[L_ack_current] 0; /* L_ACK recd for this segment */
if (L_ack_current == L_ack_cum +1) {
        dup_ack  = DEFAULT_DUP_ACK;
} else {
        dup_ack ++;
}
} else {
        DupAck_Compute_successful_LL_Retransmission()
} /* End of DupAck_Compute() function */
```

Pseudo Code – DupAck_Compute_successful_LL_Retransmission()

```
DupAck_Compute_successful_LL_Retransmission() {
L_ack_info[L_ack_current] =  0; /* L_ACK recd for this segment */
if (L_ack_current == L_ack_cum +1) {
        indx = L_ack_current;
        while ((L_ack_info[indx] != 1) && (indx < L_ack_ max))
                indx++;
```

```
        dup_ack  = DEFAULT_DUP_ACK + (L_ack_max – indx);
}
} /* End of DupAck_Compute_successful_LL_Retransmission () function */
} /* End of DupAck_Compute() function */
```

3.3 BEST-START window size

This mechanism addresses cause C of low throughput at the start of TCP sessions in current TCP implementations. It is an alternate approach to the current slow-start algorithm in existing TCP implementations. An attempt is made to leverage some of the prior data and statistics of network-congestion observed to predict the current network-congestion and traffic levels. The approach used, looks for any periodicity/patterns in network-congestion levels with definite co-relation to time of the actual day of the week/month in the year and an attempt is made to estimate the anticipated-congestion level (at the start of a new TCP session) by gathering past data/statistics of network-congestion levels; thus, it starts the new TCP session directly with a suitable-appropriate window-size (which we call "best-start" instead of the conventional "slow-start" mechanism used by current TCP implementations). This is intended to facilitate efficient usage of the available throughput, right at the start of a TCP connection. This helps raise the throughput and efficiency of short TCP sessions established by applications that need quick transport of short data burst.

WSC-TCP maintains a database of past TCP sessions with a metric of the congestion-window size achieved at steady state, to select well-known end-points of interest along with the corresponding time, day, week, month, when that session occurred. The intent is to determine any possible statistical co-relation of the observed bandwidth available, at any given time/date for a particular site, and using that to predict the actual bandwidth (one can

expect for a new connection being initiated). Thus, we can determine the correct best-start window size to be used at the time of starting a new TCP session.

3.4 Clustering mechanism for the statistical analysis of RTT times

The above mechanism addresses causes E and B of low throughput and throughput reductions. There are two main objectives of the clustering approach.

The first goal is to detect and recognize any sporadic and random, transient fluctuations in observed RTT values and prevent them from being passed onto to the TCP stack (which might interpret them incorrectly as a sign of network-congestion). The second goal is to detect and eliminate the link-layer retransmissions delays (rLL) from the observed end-end Round-Trip-Transmission time (RTT). The Effective-RTT corresponds to the RTT measured for a packet which has experienced one or more link-layer retransmissions, during its transport from source to destination minus the delays due to link-layer retransmission(s).

We did extensive simulations to evaluate TCP performance in the presence of congestion and wireless losses. This mechanism helped sustain the throughput of an existing TCP session (and not reduce the congestion window size inadvertently, due to wireless link-layer issues). We used the standard clustering library, available from The C Clustering Library [35]. The K-means partitioning algorithm kcluster was selected to find the cluster centers that would correspond to the RTT- rLL namely the RTTs when retransmission occurred in the wireless link-layer. The observed RTTs for each of the segments are stored and significant clusters of the RTT values (if any) are calculated. Any sporadic or random, transient RTT values are filtered out from having an impact on the congestion window.

4. TESTING SIMULATION

We evaluated the performance of WSC-TCP using discrete-event simulation. The NS-2 simulator [11] was used. NS-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. The TCP implementation was modified to account for the delay in retransmission (when three dup-Acks were received). A suitable topology was used for the simulation having multiple-wireless links.

For starting the test, an initial value for rLL (mentioned in section three) is chosen to be 0.5 seconds based on prior results [3] showing that the median delay in a wireless network has been found to be 0.5 seconds. However, this is a tunable parameter, which can be modified for optimal throughput based on topology and deployment considerations.

4.1 The Simulation Scenario

This section describes simulations from four scenarios, two each with

1. Standard TCP NS-2 NewReno
   (i) Using a wireless topology with an almost lossless wireless-link.
   (ii) Using the same wireless topology with a substantial lossy wireless-link at both the wireless end-nodes.

2. Standard TCP NS-2 NewReno implementation modified with our proposed enhancements for networks with wireless end-nodes.
   (i) Using the same wireless topology with an almost lossless-wireless link.
   (ii) Using the same wireless topology with a substantial lossy-wireless link at both the wireless end-nodes.

4.2 Simulation Results

Simulation Topology

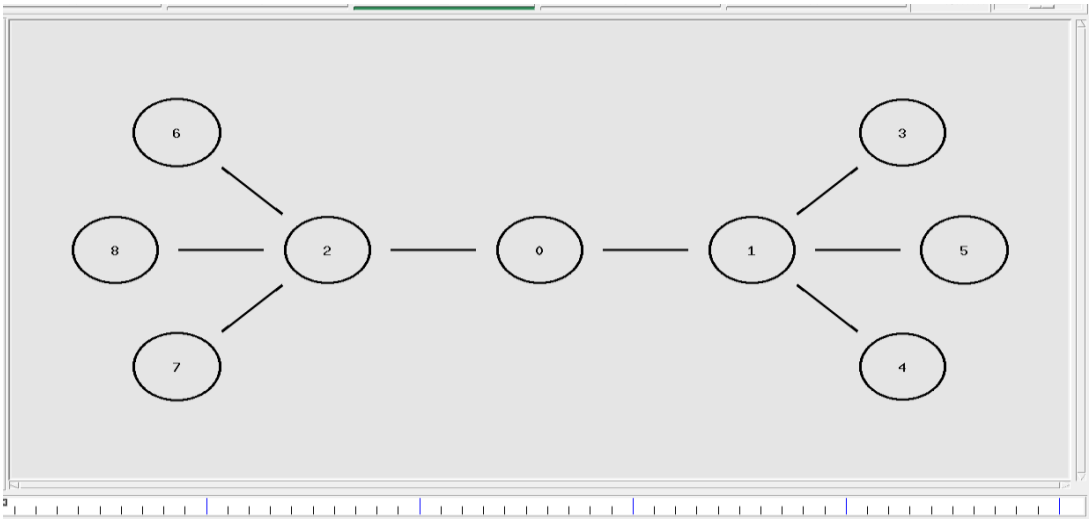


Figure 2

Where nodes (1) and (2) are Access Points and the end nodes are connected to the Access Points (APs) through wireless-links.

There was an improvement in overall throughput around 7% to 22%, observed with the new implementation, especially if transmission errors are introduced in the wireless link (signifying a wireless link with weak signal and/or interference).

Snapshot during a Simulation

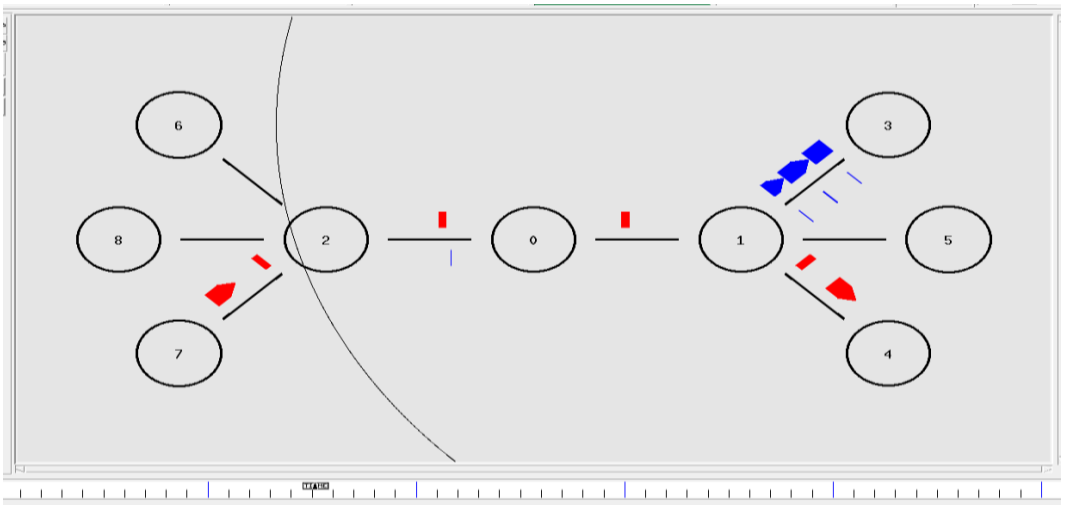


Figure 3

RESULTS IN TABULATED FORM – WHICH SHOW THE *INCREASED THROUGHPUT*

	Flows	**Run Time**	**Offered Load**	**Pkts Sent**	**Pkts Recd**	**Pkts Dropped**	**Collisions**	**Avg. Delay**
Without Changes	3	4.8	0.9883	1184	977	207	0	0.0460
With new Proposed Changes	**3**	**4.8**	**0.9958**	**1193**	**987**	**206**	**0**	**0.0439**

Table 1

	Max Delay	Min Delay	Avg Throughput	Avg Traffic Rate	Aggregate Throughput
Without Changes	0.0698	0.0403	605458	783750	1816374
With new Proposed Changes	**0.0698**	**0.0403**	**653127**	**790402**	**1999381**

Table 2

We observe an overall increase of end-end throughput in the range of 7% to 22%, depending upon the amount of wireless-link-layer-induced retransmissions, without changing the underlying network congestion (by keeping the overall network traffic levels the same).

Our results show that WSC-TCP significantly improves the throughput of TCP connections for

1) Short data-transfer bursts. This is due to its unique feature of starting very close to the available bandwidth, rather than the traditional slow start mechanism in typical TCP implementations as well as the use of proactive transmissions due to wireless-link-layer failures and hiding the wireless-link-layer delays.
2) Longer TCP sessions. This results from the isolation of wireless link-layer issues from network-congestion issues.

In addition, we observe a significant improvement in end-end latency, which would result in a marked improvement in applications like streaming high definition video and similar applications.

3. CONCLUSION AND FUTURE WORK

We presented a set of mechanisms for TCP called WSC-TCP, for a network with wireless-links. While maintaining end-to-end TCP semantics, WSC-TCP runs on the end-clients and achieves improved throughput compared to TCP NewReno. WSC-TCP performs better with a typical performance/throughput improvement of about 15% to 17% (min 10% and max 25%) from the results observed so far. This 15% increase is significant since this substantially reduces the end-end latency experienced by the end-application which is the consumer of the data; thus, significantly improving the performance experienced by an end-user on a mobile device like a smart-phone, tablet or a wireless laptop. This tangible improvement in the end-user experience makes this work relevant and is a quick incremental improvement to the existing deployments without having to rewrite the entire TCP stack.

BIBLIOGRAPHY

[1] **A. Bakre and B. R. Badrinath.** "Handoff and systems support for indirect TCP/IP". Second USENIX Symposium on Mobile and Location-Independent Computing, April 1995.

[2] **B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan.** "Improving performance of TCP over wireless networks". Technical Report 96-014, Texas A & M University, 1996.

[3] **H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz.** "A comparison of mechanisms for improving TCP performance in wireless networks". ACM SIGCOMM Symposium on Communication, Architectures and Protocols, August 1996.

[4] **D. Borman, R. Braden, and V. Jacobson.** "TCP extensions for high performance". RFC 1323 May 1992.

[5] **R. Braden, V. Jacobson, and L. Zhang.** "TCP extensions for high-speed paths". RFC 1185 October 1990.

[6] **R. Caceres and L. Iftode.** "Improving the performance of reliable transport protocol in mobile computing environment". IEEE Journal of Selected Areas in Communications, Volume 13, Issue 5, June 1995.

[7] **C. Casetti, M. Gerla, S. Mascolo, M.Y. Sansadidi, and R. Wang.** "TCP Westwood: End-to-end congestion control for wired/wireless networks". Wireless Networks Journal 8, 2002.

[8] **T. Goff, J. Moronski, and D.S. Phatak.** "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments". IEEE INFOCOM, March 2000.

[9] **C. E. Perkins.** "IP mobility support". IP Mobility Working Group Internet Draft, April 1996.

[10] **T.V. Lakshman and U. Madhow.** "The performance of TCP/IP for networks with high bandwidth-delay products and random loss". IEEE/ACM Transactions on Networking, Volume 5, Issue 3, June 1997.

[11] **S. McCanne, S. Floyd, and K. Fall.** "Network simulator". Public domain software 1995.

[12] **K. Ratnam and I.Matta**. "Effect of local retransmission at wireless access points on the round trip time estimation of TCP". IEEE 31st Annual Simulation Symposium, April 1998.

[13] **K. Ratnam and I. Matta.** "WTCP: An efficient mechanism for improving TCP performance over wireless-links". Third IEEE Symposium on Computer and Communications, June 1998.

[14] **W. R. Stevens.** "TCP/IP Illustrated The Protocols". Addison-Wesley, Reading, MA, USA, 1994.

[15] **G. Wright and W. R. Stevens.** "TCP/IP Illustrated". Addison-Wesley, Reading. MA, USA, 1995.

[16] **K. Ratnam and I.Matta.** "WTCP: An Efficient Mechanism for Improving Wireless Access to TCP Services". International Journal of Communication Systems, Volume 16, Issue 1, February 2003.

[17] **K. Xu, Y. Tian and N. Ansari. "**TCP-Jersey for Wireless IP Communications". IEEE Journal of Selected Areas in Communications, Volume 22, Issue 4, May 2004.

[18] **S. Bhandarkar, N. E. Sadry, A. L. N. Reddy and N. H. Vaidya**. "TCP-DCR: A novel protocol for tolerating wireless channel errors". IEEE Transactions on Mobile Computing, volume 4, Issue 5, August 2005.

[19] **S. Wood**, **H. Sadjadpour**, **J. J. Garcia-Luna-Aceves.** "Social Network coding Rate Control in Information Centric Delay Tolerant Networks". IEEE Global Communications Conference, December 2014.

[20] **A. E. Gamal, J. Mammen, B. Prabhakar and D. Shah. "**Throughput-Delay Trade-off in Wireless Networks". INFOCOM March, 2004.

[21] **C. Parsa and J.J. Garcia-Luna-Aceves**. "Improving TCP Congestion Control over Internets with Heterogeneous Transmission Media". Seventh International Conference on Network Protocols, 1999.

[22] **L. Brakmo and L. Peterson. "**TCP Vegas: End-to-end congestion avoidance on a global Internet". In IEEE Journal of Selected Areas in Communication, October, 1995.

[23] **D. Clark, M. Lambert, and L. Zhang. "NETBLT:** A high throughput transfer protocol." ACM SIGCOMM, Aug. 1987.

[24] **K. Fall and S. Floyd.** "Simulation-based comparisons of Tahoe, Reno, and SACK TCP". Computer Communication Review, volume 26 No. 3, July, 1996.

[25] **S. Floyd and V. Jacobson.** "Random Early Detection gateways for congestion avoidance". IEEE/ACM Transactions on Networking, Volume 4, Issue 1, Aug 1993.

[26] **R. Jacobson, R. Braden, and D. Borman. "**RFC 1323 TCP extensions for high performance". Technical report 1323, May 1992.

[27] **V. Jacobson and S. Floyd.** "TCP and explicit congestion notification". In Computer Communication Review, volume 24 No. 5, October, 1994.

[28] **P. Karn and C. Partridge.** "Improving round-trip time estimates in reliable transport protocols". In Computer Communication Review, volume 17 No. 5, August 1987.

[29] **L. Brakmo, S. O'Malley, and L. Peterson.** "TCP Vegas: New techniques for congestion detection and avoidance". ACM SIGCOMM, Aug./Sept. 1994.

[30] **M. Kazmi, M. Y. Javed, M. K. Afzal.** "An Overview of Performance Comparison of Different TCP Variants in IP and MPLS Networks". Communications in Computer and Information Science pp 120-127, Volume 136, 2011.

[31] **J. Yang**. "Dynamic clustering of evolving streams with a single pass". 19th International Conference on Data Engineering, 2003.

[32] **Y. Feng and G. Hamerly.** "PG-means: learning the number of clusters in data". 20th annual conference on neural information processing systems (NIPS), December 2006.

[33] **W. Lian, D. W. Cheung and S.M. Yiu**. "An Efficient Algorithm for Finding Dense Regions for Mining Quantitative Association Rules". Computers and Mathematics with Applications, International Journal, 2005

[34] **L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha and R. Motwani.** "Streaming-data algorithms for high-quality clustering". 18th International Conference on Data Engineering, 2002.

[35] **M. J. L. de Hoon, S. Imoto, J. Nolan, and S. Miyano.** "Open Source Clustering Software". Bioinformatics, Volume 20, Issue 9, 2004.

END OF DOCUMENT