ICEI TCP: implicit congestion and explicit interference detection in TCP

Rakesh Raghavan
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
Raghavan, Rakesh, "ICEI TCP: implicit congestion and explicit interference detection in TCP" (2003). Retrospective Theses and Dissertations. 20000.
https://lib.dr.iastate.edu/rtd/20000

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
ICEI TCP - Implicit Congestion and Explicit Interference detection in TCP

by

Rakesh Raghavan

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Arun K. Somani, Major Professor
Manimaran Govindarasu
Johnny Wong

Iowa State University
Ames, Iowa
2003

Copyright © Rakesh Raghavan, 2003. All rights reserved.
This is to certify that the Master's thesis of

Rakesh Raghavan

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
DEDICATION

This small piece of work is dedicated to Achhan, Ammachi, Lakshmi, Madhavi and their unwavering support towards my success.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2 THE NETWORK LAYERS AND THE TCP/IP SUITE</td>
<td>4</td>
</tr>
<tr>
<td>2.1 The OSI Reference Model</td>
<td>4</td>
</tr>
<tr>
<td>2.2 The TCP/IP Protocol Suite</td>
<td>7</td>
</tr>
<tr>
<td>2.3 The Transmission Control Protocol</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 The Slow Start And Congestion Avoidance Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>2.3.2 The Multiplicative Decrease Algorithm</td>
<td>14</td>
</tr>
<tr>
<td>2.3.3 The TCP Timers</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Congestion And Interference Issues In TCP</td>
<td>17</td>
</tr>
<tr>
<td>2.5 TCP Algorithms To Tackle Congestion And Interference</td>
<td>19</td>
</tr>
<tr>
<td>3 RELATED WORK - TECHNIQUES THAT IMPROVE THE PERFORMANCE OF TCP ON WIRELESS AND HETEROGENEOUS NETWORKS</td>
<td>21</td>
</tr>
<tr>
<td>3.1 Pure Link-Level Approaches</td>
<td>26</td>
</tr>
<tr>
<td>3.2 Soft-State Transport Layer Caching Approaches</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1 The Snoop Protocol</td>
<td>27</td>
</tr>
<tr>
<td>3.3 Soft-State Cross Layer Signaling Approaches</td>
<td>28</td>
</tr>
<tr>
<td>3.3.1 Explicit Congestion Notification (ECN)</td>
<td>28</td>
</tr>
</tbody>
</table>
3.3.2 Mobile Aware Interference Robust TCP (MAIRTCP) ........................................ 29
3.3.3 Explicit Bad State Notification (EBSN) .......................................................... 30
3.3.4 Explicit Loss Notification (ELN) ................................................................. 30
3.3.5 Multiple Acknowledgements ....................................................................... 30
3.3.6 Mobility Awareness Incorporated As TCP Enhancement (MAITE) .................. 32
3.4 Hard-State Transport Layer Approaches ......................................................... 32
3.4.1 Indirect TCP (I-TCP) .................................................................................. 32
3.5 Pure End-to-End Transport Level Approaches ................................................. 33
3.5.1 IRTCP ......................................................................................................... 34
3.5.2 FREEZE-TCP .............................................................................................. 35
3.5.3 WTCP ......................................................................................................... 36
3.5.4 TCP-Santa Cruz .......................................................................................... 36
3.5.5 M-TCP ......................................................................................................... 37
3.5.6 TCP Westwood ............................................................................................ 38
3.5.7 Discriminating Congestion Losses From Wireless Losses Using Inter-Arrival Times At The Receiver ................................................................. 39
3.6 Ad-Hoc Network Protocols ............................................................................... 39
3.6.1 TCP-F ......................................................................................................... 40
3.6.2 TCP-BuS .................................................................................................... 41

4 ICEI TCP “ICY” - IMPLICIT CONGESTION AND EXPLICIT INTERFERENCE DETECTION IN TCP ................................................................. 43
4.1 Motivation .......................................................................................................... 43
4.2 The Case For ICEI TCP .................................................................................... 45
4.2.1 Handling Of Congestion In ICEI TCP ......................................................... 47
4.2.2 Handling Of Interference And Hand-off’s In ICEI TCP .............................. 50
4.3 ICEI TCP Algorithms ....................................................................................... 53
4.3.1 At The TCP Source ........................................... 54
4.3.2 At The TCP Sink ............................................. 56
4.3.3 The Working Of ICEI TCP During Congestion And Interference . 57

5 ICEI TCP - IMPLEMENTATION, SIMULATIONS AND RESULTS 58
  5.1 ICEI TCP Implementation Details .................................. 58
  5.2 The Heterogeneous Network Model ................................. 61
  5.3 The Goodput Metric For TCP Performance .......................... 62
  5.4 Simulation Scenario 1 - TCP Flows With Predominantly Interference Losses 63
  5.5 Simulation Scenario 2 - TCP Flows With Both Interference And Congestion Losses ........................................... 68
    5.5.1 Discussions On The Results ................................ 72

6 CONCLUSIONS AND FUTURE WORK ................................. 74
  6.1 Conclusions ..................................................... 74
  6.2 ICEI TCP - A Footnote ......................................... 74
  6.3 Future Work .................................................... 75

BIBLIOGRAPHY ..................................................... 77

ACKNOWLEDGMENTS ................................................ 80
LIST OF FIGURES

Figure 1.1  The Internet And Peripheral End-User Devices .................. 2
Figure 2.1  The Layers Of The OSI Reference Model ....................... 5
Figure 2.2  The Layers Of The TCP/IP Protocol Suite ...................... 7
Figure 2.3  The Protocols Of The TCP/IP Protocol Suite ................... 9
Figure 2.4  Bridges And Gateways In A Network ............................ 11
Figure 2.5  The TCP Slow Start And Congestion Avoidance Algorithm ... 13
Figure 2.6  The TCP Multiplicative Decrease Algorithm ................... 14
Figure 2.7  The Algorithms Implemented In The Different Flavors Of TCP . 18
Figure 3.1  Cellular Wireless Network Architecture ....................... 22
Figure 3.2  Wireless Ad-Hoc Network Architecture ......................... 23
Figure 3.3  Satellite Network Architecture .................................. 24
Figure 3.4  The Different Approaches To Improve The Performance Of TCP On Wireless Networks .................. 25
Figure 3.5  The Snoop Protocol .................................................. 27
Figure 3.6  The State Diagram Of The MAIRTCP Protocol ................ 29
Figure 3.7  TCP With Multiple Acknowledgements ............................ 31
Figure 3.8  Indirect-TCP (I-TCP) ............................................... 33
Figure 3.9  The Pipe Snapshot Algorithm Of The IRTCP Protocol .......... 34
Figure 3.10  The Working Of The MTCP Protocol ............................ 37
Figure 3.11  The Working Of The TCP-F Protocol ........................... 40
Figure 3.12 The Working Of The TCP-BuS Protocol .......................... 42

Figure 4.1 Congestion History In The ICEI TCP Protocol .................. 49
Figure 4.2 Interference Profile In The ICEI TCP Protocol .................. 51
Figure 4.3 The Working Of The ICEI TCP Protocol ......................... 54
Figure 4.4 The Algorithm At The ICEI TCP Source ....................... 55
Figure 4.5 The Algorithm At The ICEI TCP Sink ......................... 56

Figure 5.1 A Heterogeneous Network Model With Wired And Wireless Links 60
Figure 5.2 Simulation Scenario 1 - TCP Flows With Predominantly Interference Losses .................................................. 63
Figure 5.3 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 128 MAC Frames And Average Bad State = 3 MAC Frames ............................................. 64
Figure 5.4 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 100 MAC Frames And Average Bad State = 3 MAC Frames ............................................. 64
Figure 5.5 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 75 MAC Frames And Average Bad State = 3 MAC Frames ............................................. 65
Figure 5.6 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 50 MAC Frames And Average Bad State = 3 MAC Frames ............................................. 65
Figure 5.7 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 25 MAC Frames And Average Bad State = 3 MAC Frames ............................................. 66
Figure 5.8 Results For Scenario 1 - Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods .............. 66
Figure 5.9 Results For Scenario 1 - Confidence Intervals For The Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods ..................................... 67

Figure 5.10 Simulation Scenario 2 - TCP Flows With Both Congestion And Interference Losses .......................................................... 67

Figure 5.11 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 128 MAC Frames And Average Bad State = 3 MAC Frames ........................................... 68

Figure 5.12 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 100 MAC Frames And Average Bad State = 3 MAC Frames ........................................... 69

Figure 5.13 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 75 MAC Frames And Average Bad State = 3 MAC Frames ........................................... 69

Figure 5.14 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 50 MAC Frames And Average Bad State = 3 MAC Frames ........................................... 70

Figure 5.15 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 25 MAC Frames And Average Bad State = 3 MAC Frames ........................................... 70

Figure 5.16 Results For Scenario 2 - Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods ............ 71

Figure 5.17 Results For Scenario 2 - Confidence Intervals For The Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods ............ 71
Computers and data networks have revolutionized communications today. The TCP/IP Protocol suite specifically designed for wired networks, is the predominant protocol suite used in data networks. However, a large number of wireless devices are now part of data networks including the internet.

A TCP flow on a wired network experiences data loss predominantly due to congestion on the network, while a TCP flow on a heterogeneous network, which is a combination of wired and wireless links, may experience data loss due to either congestion (along the wired links) or interference (on the wireless links). Whenever data loss occurs, TCP assumes the loss is due to congestion and reduces its transmission window size to relieve the congestion. This adversely affects the TCP throughput in a heterogeneous or a wireless network where most of the data loss is due to interference.

In this research we propose ICEI TCP - a scheme for Implicit Congestion and Explicit Interference detection in TCP. ICEI TCP is designed based on the following assumptions:

i). The heterogeneous network has a cellular wireless network architecture where all the links of the network are wired links while the last link alone is a wireless link

ii). The wireless host simultaneously runs multiple distributed applications i.e. multiple TCP flows terminate on the same wireless host.

In our scheme each TCP source within a sending host maintains its own Congestion History (CHISTORY), a fixed size queue, which contains the latest values of the Round Trip Time Ratio (RTTR). The RTTR is a ratio of the Round Trip Time (RTT) for each transmission of the TCP flow to the lowest such RTT value computed during the
lifetime of that TCP flow. Rapid variations in the RTTR of successive transmissions, are treated as an indication that the TCP flow is heading towards congestion phase.

Each wireless host on the network also maintains an Interference Profile (IPROFILE), a shared area in memory, which contains the latest details of the times at which packets loss is experienced by different TCP sinks on that wireless host. In case the IPROFILE has entries from different TCP sinks during a time interval i.e. if more than one TCP sink on the same wireless host experiences a packet loss within a time interval, the packet losses for all those TCP flows is interpreted to be due to the effect of interference on their shared wireless link.

In the event of a packet loss, the TCP source in the ICEI TCP scheme now uses the information from the CHISTORY and the IPROFILE to decide if congestion control algorithms need to be invoked. This ensures that congestion control algorithms are not invoked during a phase where interference is causing the data loss, thereby improving the throughput of the TCP flow.
1 INTRODUCTION

Computers have revolutionized human endeavor perhaps more than any other invention in such a short duration. Though they were initially designed for computing and calculations, with the advent of networking and later the World Wide Web, their new found role as communication devices have shrunk the world.

As a natural progression of this phenomenon, non-computing entities like PDA's, mobile phones and other low-power, hand-held devices are slowly breaking into data networks as end-user devices. Already the rate of growth of these wireless and mobile devices is impressive. It is not difficult to imagine a day when these devices become the preferred user tools for computation and communication and the computer that started this revolution goes back to being a programmer's tool.

With the shift in the kind of devices that are finding their way on to communication networks, we see a change to the traditional paradigm of networking. Computer networks are changing from being entirely wired entities connecting different computing devices to wired and wireless entities (see Figure 1.1) that connect both computing devices and other user devices that are less computation oriented and more communication oriented. In such a scenario it is worthwhile to examine some of the assumptions and axioms that we have used to design and run computer networks thus far.

The most preferred protocol stack for reliable end-to-end data communication is the TCP/IP stack, which was essentially built, implemented, tested and accepted on wired networks. However, wireless and heterogeneous networks provide some unique challenges that the TCP/IP protocol suite has to resolve to maintain it's monopoly over data traffic.
of interest to us here, is the packet loss experienced by the network and the usable
data transferred to the TCP layer in a wireless network or a heterogeneous network (a
network which has both wired and wireless links). In this research, we address the twin
problems of congestion and interference. Congestion occurs in both wired and hetero-
genous networks but the problem of interference is a characteristic of heterogeneous or
purely wireless networks.

The issue of congestion in networks has been extensively studied, but the identifica-
tion of interference in networks has merited interest more recently and is an interesting
area of research. In this work we propose an alternate scheme to TCP called ICEI TCP; a scheme for the Implicit Congestion and Explicit Interference detection in TCP flows.

This thesis is organized as follows. In Chapter 2, we provide an overview of the ISO reference model and the TCP/IP protocol suite and the functional specifics of the TCP layer. Next we discuss some related works, proposed in literature for the identification and resolution of congestion and interference in TCP in Chapter 3. In Chapter 4, we propose the ICEI TCP scheme and explain our algorithms. A detailed explanation of the implementation of the scheme in NS (the Network Simulator) along with the simulation results are presented in Chapter 5. We conclude with our observations and future topics of research in Chapter 6.
2 THE NETWORK LAYERS AND THE TCP/IP SUITE

When the idea of a computer data network was nascent, the designers decided to partition their task into smaller and smaller subtasks to make the work easier. This partitioning took the form of layering and today we have protocol suites or stacks which are essentially different functional layers stacked on top of each other to provide data transport.

Once the layers are in place, each layer is capable of providing a service to the layer above it by using one or a combination of services of the layer below it. Though the physical communication was between the different layers of the protocol stack on the same machine, the idea is to use this to establish a logical communication between the same (peer) layer on another machine. In other words, the communication between layer n on machine A and layer n on machine B follow a “protocol” while the communication between layer n and n-1 on both machine A and B happens via an “interface”.

2.1 The OSI Reference Model

In the OSI reference model, the idea of layering was introduced to split the communication hardware and software into specific functional elements. The International Standards Organization (ISO) standardized the model into the OSI (Open systems Interconnection) model and defined its seven layers and their functionalities as depicted in Figure 2.1.

The physical layer provides physical connectivity to the network in the form of
Figure 2.1 The Layers Of The OSI Reference Model

wires, cables, wireless channels or optical fibers. The physical layer only recognizes bit-streams, so it performs the functions of activating, maintaining, and deactivating a physical connection, so that a stream of bits may be transmitted through the physical medium.

The data link layer provides a means to transfer data between network entities. It also provides methods to correct and control transmission errors, schemes to group bit streams into message frames, frame synchronization and flow control. E.g.: HDLC and Ethernet. A sub-layer of the data link layer called the Media Access Control layer (MAC) provides the means to access the physical medium before the data may be transferred.
The network layer provides a logical network connection between different entities on the network. The network layer accepts data from the transport layer and breaks them down into data units called packets, which is a collection of frames. It provides switching and routing capabilities to establish, maintain, and terminate network layer connections and permit the transfer of packets between users. Once the packet has reached the destination, the network layer is responsible for reassembling the packets and passing them on to the transport layer.

The transport layer is the first end-to-end layer in the stack and it provides transparent transfer of data between systems. The transport layer may be either reliable or unreliable but it always provides end-to-end control and information interchange.

The session layer provides mechanisms for organizing and structuring session information in a network. It structures the dialogues between application processes, handles synchronization between the different sessions’ of the same application and has methods for structuring and synchronizing data exchanges between them.

The presentation layer provides an interface, which can handle the different forms of data representation called “presentation contexts” which include syntax, syntax selection and conversion between the different representation schemes. This allows the applications to use any of a wide selection of presentation contexts and yet be able to communicate with other applications which may be using a different presentation context.

The application layer contains the actual application that interacts with the user. It also contains service elements like library routines, which perform inter-process communication, common procedures for constructing application protocols and for accessing the services provided by the servers on a network.
2.2 The TCP/IP Protocol Suite

The OSI model was intended to be a standard. The effort to make the standard robust introduced multiple redundancies in the functioning of the different layers, making it a bit cumbersome to implement and maintain.

![Diagram of the Layers Of The TCP/IP Protocol Suite]

Figure 2.2 The Layers Of The TCP/IP Protocol Suite

A simple functional abstraction on the OSI model is the TCP/IP Protocol suite which crunches the classification into just four layers and eliminates a lot of redundancies present in the OSI model.
The different logical layers (see Figure 2.2) of the TCP/IP protocol stack function more or less as per the specifications of the OSI standard.

The **application layer** handles all non-TCP/IP functionalities and contains application protocols that support different tasks like file transfer and remote system access.

The **transport layer** handles end-to-end data transmission and reception, including session and error control functions and data sequencing.

The **network layer** is responsible for data addressing, data routing, and packet fragmentation and reassembly (IP protocol).

The **network interface and data link** layer is responsible for transmitting data across the network and determines how and when the physical medium may be accessed.

A combination of different protocols (see Figure 2.3) make up the TCP/IP suite.

The **transport layer protocols** provide end-to-end connectivity for the transfer of data between a source and destination on a network. They include:

1). **TCP** (Transmission Control Protocol): TCP is a connection-oriented end-to-end service, which provides a reliable connection from source to destination.

2). **UDP** (User Data-gram Protocol): UDP is a connectionless service. It does not guarantee reliable transfer of data.

The **network layer protocols** help to actually route the data from a source to destination on a network via intermediate hosts. They include:

1). **IP** (Internet Protocol): IP handles the actual data routing from a source entity to a destination entity on a network.

2). **ICMP** (Internet Control Message Protocol): ICMP handles control messages for IP. ICMP is used whenever control information regarding the state of the network has to be passed between hosts in a network.

3). **RIP** (Routing Information Protocol): RIP determines the best routing method for IP to deliver a message. RIP uses the routing information available at each host in a network to determine the best route that a packet may use to reach its destination.
4). **OSPF** (Open Shortest Path First): OSPF is one of the methods that IP uses to deliver packets. OSPF uses the shortest available path from the source to the destination to transfer data.

The **network address protocols** help to identify specific hosts on a network so that data may be transferred to the desired recipient. They include:

1). **ARP** (Address Resolution Protocol): ARP determines the unique numeric hardware addresses of the machines on the network. This unique hardware address enables data transfer to or from any host in the network to any other host served by the same data link (ethernet) layer.
2). **DNS** (Domain Name Service): DNS resolves the numeric addresses from machine names (host names). Hence an application has the option of referring to a destination host either by means of it’s host name or it’s IP address.

3). **RARP** (Reverse Address Resolution Protocol): RARP performs the opposite of the actions performed by the ARP. The RARP service provides the IP address of a host when it’s unique hardware address is known.

The **user-based services** provide the user with applications that my be directly used or be invoked by other applications. They include:

1). **BootP** (Boot Protocol): BootP boots a network by reading information from a central server and sets up the various services required for a network to operate including a gateway service and a name service.

2). **FTP** (File Transfer Protocol): FTP allows the transfer of files across a network. The files may be transferred across different network hosts and they may be resident under different operating systems.

3). **Telnet**: Telnet allows users to remotely log in to another machine. The Telnet session sets up a remote terminal on a machine which can connect to any other machine in the network.

The **gateway-based services** provide the means to route data between different domains and sub-nets (see Figure 2.4) on a network. They include:

1). **EGP** (Exterior Gateway Protocol): EGP governs the transfer of routing information for external networks when the source and the destination are part of different autonomous domains employing their own routing schemes.

2). **GGP** (Gateway-to-Gateway Protocol): GGP enables the routing of information and exchange of control information between different gateways on a network.

3). **IGP** (Interior Gateway Protocol): IGP handles the routing of information for internal networks when the source and the destination are attached to different gateways, which form a part of the same autonomous domain.
4). **BGP** (Border Gateway Protocol): BGP handles inter-domain routing when the source host and the destination host are situated in different domains.

The simplicity of the TCP/IP protocol suite was the main reason for its rapid spread and this model continues to monopolize data transmission systems worldwide.

### 2.3 The Transmission Control Protocol

The Transmission Control Protocol (TCP) forms the transport layer of the TCP/IP protocol suite. TCP is the first end-to-end layer in the stack and provides a connection-oriented service for the transfer of data from the source to the destination. In addition
TCP also performs the functions of flow control, error control, data fragmentation, data re-assembly and sequencing. This makes TCP a reliable transport layer on top of the un-reliable and connectionless IP layer.

TCP is a chaotic protocol, which relies on the random behavior of the different TCP flows in the network to bring about a division of the total bandwidth on shared links. The total amount of data that is sent by TCP is governed by the sending or congestion window size (CWND) and the receiving window size (RWND). The CWND is the maximum size of data that may be transmitted by the TCP source (at the source host) and the RWND is receiving window size i.e. the maximum buffer available at the TCP sink (at the destination host). The actual data transmitted by the TCP source is determined by the maximum of the values of CWND and RWND. TCP continuously tries to increase its CWND as long as no packet loss occurs. When packet losses occur TCP reduces its CWND by half and resumes the transmission.

TCP algorithms for the transmission of data are classified as the slow start and congestion avoidance algorithm and the multiplicative decrease algorithm (see [10], [20]).

2.3.1 The Slow Start And Congestion Avoidance Algorithm

The Slow Start algorithm (see Figure 2.5) causes an exponential increase in the congestion window until data is lost. Initially, the sender’s window (CWND) is one segment. When the ACK for this segment arrives, CWND is increased to two and two new packets are transmitted. When their ACK’S arrive, CWND is incremented by one for each ACK i.e. twice totally to become four, and so on. In this phase the CWND is doubled when ‘CWND’ number of ACK’s are received.

The Slow Start algorithm exponentially increases the CWND until the CWND is equal to the SSTHRESH. After this threshold is reached the TCP flow goes into a period where the growth of the CWND becomes linear. This phase is called the Congestion Avoidance phase. In this phase on the reception of an ACK the CWND is increased
by 1/CWND and so the CWND is increased by 1 when ‘CWND‘ number of ACK’s are received.

Eventually, the congestion window becomes so large that packet loss occurs. At this point the Multiplicative Decrease algorithm takes over.

The slow start and congestion avoidance algorithm has the following steps:

1). When starting data transmission, or when the Effective Window size in the receiver (RWND) is zero, set CWND to one packet.

2). On each acknowledgment for successful data reception, if the current CWND value is less than the SSTHRESH, increase CWND by one, else increase CWND by
3). At any time, transmit packets so that the amount of in-transit data is the minimum of the RWND and CWND.

4). Revert to Multiplicative Decrease on packet loss.

2.3.2 The Multiplicative Decrease Algorithm

The slow start algorithm causes TCP to slowly increases its congestion window size until data is lost; the multiplicative decrease algorithm (see Figure 2.6) causes the congestion window to be halved.

Figure 2.6 The TCP Multiplicative Decrease Algorithm
This helps to drain the excess packets, that are the cause of the congestion, from the network. This pattern of increase and decrease causes a saw-tooth behavior in bandwidth usage over time and continues until equilibrium is achieved. Changes within the network itself may disturb the equilibrium, but the Multiplicative Decrease followed by the Slow Start scheme helps to return data transmission to a near-equilibrium point.

The multiplicative decrease algorithm has the following steps:
1). The sender’s state has a variable CWND (the congestion window) and the SSTHRESH (Slow Start Threshold).
2). When packet loss is detected, set CWND to half the current window size and set SSTHRESH also to this value.
3). On each acknowledgment for successful data reception, increase CWND by $1/CWND$.
4). At any time, transmit packets so that the amount of in-transit data is the minimum of the effective window at the receiver (RWND) and CWND.

2.3.3 The TCP Timers

TCP maintains certain timers to control its operation. The retransmission time-out timer maintains the time that should elapse before those packets, which have not been acknowledged are presumed lost and the whole segment is retransmitted.

Ideally, it is desired to keep the timer to be close to the true round trip (delay) time (RTT). Because the actual round trip time varies dynamically (unlike in the data link layer), using a fixed timer will not serve the purpose.

To cope with widely varying delays, TCP maintains a dynamic estimate of the current RTT:
1). When sending a segment, the TCP source starts a timer.
2). Upon receipt of an acknowledgment it stops the timer and record the actual elapsed delay between sending the segment and receiving its ACK.
3). Whenever a new value for the current RTT is measured, it is averaged into a smoothed RTT or SRTT which is computed using the following formula:

\[ SRTT = (\alpha \times SRTT) + (1 - \alpha) \times RTT \]  \hspace{1cm} (2.1)

where:
\( \alpha \) is the smoothing factor (weight) and typically \((0.8 \leq \alpha \leq 0.9)\) is used.
When \( \alpha = 0 \), \( SRTT = RTT \) (new value of RTT) and
when \( \alpha = 1 \), \( SRTT = SRTT \) (old value of SRTT).

The actual RTT varies between successive transmissions due to normal queuing delays along the path. Hence it would be a mistake to throw out the old RTT and use the new one. The use of \( \alpha \) in the above formula allows the change of the SRTT estimate slowly, so that the TCP source does not overreact to wild fluctuations in the RTT.

The SRTT is only an estimate of the actual delay, and the actual delay varies from packet to packet. Therefore, the actual retransmission timeout (RTO) timer value for a segment is set to be larger than SRTT. To decide on how much larger this value has to be TCP also maintains an estimate of the mean deviation (D) of the RTT. The parameter D, which is the difference between the measured and expected RTT values, provides a close approximation to the standard deviation and is computed by the following formula:

\[ D = (\alpha \times D) + (1 - \alpha) \times |SRTT - RTT| \]  \hspace{1cm} (2.2)

Based on these measurements the value of the retransmission timeout (RTO) timer is set as:

\[ RTO = SRTT + (4 \times D) \]  \hspace{1cm} (2.3)
2.4 Congestion And Interference Issues In TCP

The Internet, which is an aggregation of heterogeneous computer network's is a best-effort network that does not guarantee the quality of service it can provide at any point in time. Congestion occurs when this best-effort network is subjected to a high load. During high network load conditions data may arrive at the input of a router at a rate that is much faster than the rate at which the router dispatches the data. Hence packets of data spend more and more time in the buffers of the router and thereby the buffers at the host and routers may overflow causing packets to be dropped.

Interference or EMI (Electromagnetic Interference) is caused due to the superposition of coherent external transmissions or their harmonics on the actual signal being transmitted. In case the interfering signal is not coherent it is termed as noise. The superposition of the interfering signal changes the amplitude of the original signal and the power associated with the signal. Since interference is a physical phenomenon it may be detected by a measurement of the power of the incoming signal.

Multi-path interference is another physical phenomenon associated with wireless networks. The actual instantaneous power received by the antenna varies since the electromagnetic waves traverse through many different paths to reach the antenna. This makes the detection of these waves difficult and thus cause a loss of data.

Another reason why packet losses occur is due to the phenomenon of hand-off's. In "cellular" mobile wireless networks (see Figure 3.1) multiple Mobile Hosts (MH's) are connected to a wired network through a single Base Station (BS). Each BS is able to communicate with the MH's within a geographical area known as a "cell", which is determined by the transmission strength of the signal at the BS. When a MH moves over a large geographical area it has to leave the "cell" controlled by one particular base station and moves to the next base station so that the data transmission may continue. During this "hand-off" interval when the MH breaks off connection from one BS and
establishes the connection with the other BS packet losses may occur.

Frequency Hopping Spread Spectrum (FHSS) communication actually reduces interference since the transmission frequency of the source is not fixed. The source host transmits data on multiple frequencies that follow a particular pattern which is known to both the source and the sink. The possibility of two source's transmitting on the same frequency and thereby interfering with each other's signal's is thus minimized. However in case a large number of transmissions occur simultaneously in a narrow spread-spectrum bandwidth the chances for interference are high.

In a wired network packet losses occur primarily due to congestion. In heterogeneous networks, packet losses may occur due to congestion, interference or hand-off's, while packet corruption, which again is treated as a packet loss by the TCP layer, occurs due to interference.

<table>
<thead>
<tr>
<th>TCP Flavor</th>
<th>Slow Start</th>
<th>Congestion Avoidance</th>
<th>Fast Recovery</th>
<th>Fast Retransmit</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Reno</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TCP Tahoe</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 2.7 The Algorithms Implemented In The Different Flavors Of TCP
2.5 TCP Algorithms To Tackle Congestion And Interference

Most TCP implementations employ a combination of algorithms (see Figure 2.7), which try to tackle some of the problems that are caused by interference, congestion and handoff losses.

The TCP Tahoe flavor was first implemented in 4.3 BSD UNIX in 1988. It includes the slow start, congestion avoidance and the fast retransmit algorithms. The TCP Reno flavor was an extension of TCP Tahoe and was first implemented in 4.3 BSD UNIX in 1990. It includes the slow start, congestion avoidance, fast retransmit and fast recovery algorithms.

2.5.0.1 Fast Retransmit Approach

The Fast Retransmit approach addresses the issues of TCP performance when communication resumes after a handoff. In TCP the delay caused by a handoff process causes the Retransmission Time-out timer (RTO) at the TCP source to time-out. The TCP source automatically assumes congestion and invokes its congestion control algorithms and retransmits packets to the TCP sink. The retransmissions begin only after the RTO times-out thereby causing a loss in throughput.

In the Fast Retransmit approach, the mobile host rapidly sends a certain threshold number of duplicate acknowledgements (DUPACKS) to the TCP source, after a hand-off, to speed up the process of retransmission. This causes the TCP source to immediately reduce its window size and retransmit packets starting from the first missing one, thereby improving the overall throughput of the TCP flow.

2.5.0.2 TCP Selective Acknowledgments (SACK’s)

The Selective Acknowledgments (SACK) TCP is an extension to the normal algorithms and it allows the TCP source to recover from multiple packet losses within a
single window of transmission. In TCP, multiple packet losses within a single window of data affect TCP throughput in two ways - either because the TCP source waits too long (for an entire Round-Trip Time (RTT)) before re-transmitting the lost packets or because the TCP source re-transmits a lot of packets that have been already been received by the TCP sink.

SACK TCP does not rely on a single coarse timeout to determine packet losses instead SACK's strategy addresses multiple packet losses by enabling the TCP sink to inform the TCP source about all the segments that have arrived successively. The TCP source then, retransmits only those segments that have actually been lost.

2.5.0.3 Fast Retransmit And Fast Recovery

The Fast Retransmit and Fast Recovery scheme, address the problems arising from the loss of a few segments in a TCP transmission. In the Fast Retransmit and Fast Recovery scheme, the TCP source, receives an ACK for each segment received by the TCP sink. In case one or two segments are lost, the TCP sink sends a Duplicate Acknowledgement (DUPACK), with the same ACK number as the last sent ACK. Upon receiving the threshold number of DUPACK's (generally three), the TCP source, retransmits the lost segment/segments, without waiting for the Retransmission Timer (RTO) to time out. The TCP source then resumes the transmission of the unacknowledged data. This eliminates the latency involved with waiting for the RTO time-out and thus improves the throughput of the TCP transmission.
3 RELATED WORK - TECHNIQUES THAT IMPROVE
THE PERFORMANCE OF TCP ON WIRELESS AND
HETEROGENEOUS NETWORKS

Most wireless network architectures employed today may be broadly classified into
three major groups. The following is a brief overview of the three major classifications.

1). Base Station Architecture For Cellular Networks

The cellular network architecture (see Figure 3.1) is the most common form of wireless
network in use today. In cellular networks the Mobile Host (MH) is connected to the
fixed network through a Base Station (BS).

In such networks all links on the fixed network are wired links but the link from the
BS to the MH is wireless. Each BS is able to communicate with all the MH’s within a
geographical area known as a “cell”, which is determined by the transmission strength
of the signal at the BS. Mobile devices such as cellular phones and PDA’s use cellular
networks. Most of the protocols that have been proposed to improve the performance
of TCP over wireless networks assume this architecture.

2). Wireless Ad - Hoc Networks

Wireless ad-hoc networks (see Figure 3.2) are formed by mobile hosts which remain
connected to each other via radio interfaces.

The MH’s are always within a radio “sight” i.e. within radio distance of each other.
The MH’s in ad-hoc networks have a peer-to-peer relationship with each other and
communication is established either through an Access Point (AP) or directly between
Wired and Wireless Nodes Connected via a Base Station

Fixed Hosts connected via wired links to the Base Station

Mobile Hosts connected via wireless links to the Base Station

Figure 3.1 Cellular Wireless Network Architecture
Mobile Hosts connected to Access Point 1 via wireless links

MH1

Mobile Hosts connected to Access Point 2 via wireless links

MH1

MH2

Access points 1 and 2 connected via wired links

AP1

AP2

MH2

MH3

MH3

Figure 3.2 Wireless Ad-Hoc Network Architecture

two MH's. The ad-hoc network architecture is not widely used and so very few solutions have been proposed to improve TCP's performance on this architecture.

3). Satellite Networks

Wireless networks in which a satellite link connects the sender and the receiver (see Figure 3.3) are called satellite networks.

Satellite communication channels are inherently lossy and have very high Bit Error Rates (BER). The large distances to and from the communication satellite that the transmitted signals have to cover cause high transmission and reception latencies in these networks.

Several different protocols have been proposed in literature to improve the perfor-
The different protocols and schemes try to differentiate between the following losses:

1. Losses due to congestion in the network.
2. Losses due to interference or random errors in the network.
3. Losses assumed due to the timeout of the Retransmission Timer (RTO).
4. Losses assumed due to the out of order packets being delivered to the TCP sink causing it to transmit Duplicate Acknowledgement's (DUPACK's) over and above the DUPACK threshold value (generally three DUPACK's).

One method of classifying the various schemes is to differentiate between the network
Figure 3.4  The Different Approaches To Improve The Performance Of TCP On Wireless Networks
layers at which these schemes operate on and to differentiate the kind of state information that these protocols have to maintain. Using this method a classification has been shown in Figure 3.4.

3.1 Pure Link-Level Approaches

The pure link-level approaches abstract the characteristics of the wireless links from the higher layers of the protocol stack. In these approaches reliable link-level protocols are implemented on the wireless link. These protocols employ techniques such as Forward Error Correction (FEC) for error control and Automatic Repeat Request (ARQ) to detect errors along the wireless links and perform local retransmissions independent of the higher-level protocols.

The FEC schemes at the link layer provide error correction capability at the destination host using cyclic codes. On the other hand, the ARQ is an error control protocol whereby the destination host asks the source host to retransmit any lost packets. The higher layers are ignorant of these protocols and hence their performance is not affected. E.g.: The AIRMAIL protocol.

3.2 Soft-State Transport Layer Caching Approaches

The soft-state transport layer caching approaches are aware of the transport layer and maintain a “soft” state or a cache to save the sender from unnecessary invocation of its congestion control mechanisms. The implementation of these protocols is quite involved and they require special algorithms to run at the intermediate nodes like the base station.
3.2.1 The Snoop Protocol

The Snoop protocol (see [8], [9]) is classified as a TCP aware link level protocol, which involves modifications to the network layer (IP) software at the base station (BS) (see Figure 3.5). The snoop module looks at every packet, in either direction, in the TCP connection from the Fixed Host (FH) to the Mobile Host (MH).

![The Snoop Protocol](image)

Figure 3.5 The Snoop Protocol

It maintains a cache of TCP packets sent from the FH that haven't yet been acknowledged by the MH. When a packet loss is detected, snoop retransmits the lost packet to MH if it has the packet cached. Thus, the snoop module at the BS hides the packet...
losses from the FH, by not propagating Duplicate Acknowledgments (DUPACK’s). This prevents the unnecessary invocations of the congestion control mechanisms by the TCP source at the FH.

3.3 Soft-State Cross Layer Signaling Approaches

The soft-state cross layer signaling approaches make the transport layer at the source aware of the wireless link. In these schemes the link layer or the network layer informs the transport layer at the source about specific events in the network so that it can adapt accordingly. The signaling involved may increase the congestion at the network layer and the data traffic at the link layer.

3.3.1 Explicit Congestion Notification (ECN)

Explicit Congestion Notification (ECN) (see [17]) is an extension proposed to the Random Early Detection (RED) scheme. RED is an active queue management mechanism in routers, which detects congestion before the queue overflows and provides an indication of this congestion to the end hosts. A RED enabled router is configured with state information which provides it with a minimum (MINT) and a maximum (MAXT) threshold for its queue length. The RED router drops packets in its queue probabilistically when the average queue size is between MINT and MAXT. This action signals the TCP source and TCP sink that the router is in a state of incipient congestion before the queue actually runs out of buffer space. The TCP source then invokes its congestion control algorithms to relieve some of the network traffic passing through the router.

ECN relies on an extension to RED. ECN works exactly like RED except that an ECN enabled router does not drop packets but marks packets with an ECN bit. The TCP source then checks this bit and if set, it recognizes incipient congestion at the router and invokes its congestion control algorithms.
3.3.2 Mobile Aware Interference Robust TCP (MAIRTCP)

Mobile Aware Interference Robust TCP (MAIRTCP) (see [13]) is a modification on the Interference Robust TCP (IRTCP) scheme described in section 3.5. In MAIRTCP, the TCP sink notifies the TCP source of the state of the wireless channel. If a corrupted packet is received, the channel is assumed to be under the influence of interference, while the loss of a packet altogether denotes congestion in the network. The TCP sink maintains a state machine (see Figure 3.6) which determines the channel state.

![Figure 3.6 The State Diagram Of The MAIRTCP Protocol](image)

If the TCP sink detects congestion, it transmits Duplicate Acknowledgements (DACK’s) which force the "Active Transmitter" implemented at the TCP source to invoke its congestion control algorithms.
In case the TCP sink detects interference, the “Active Receiver” implemented at the TCP sink moves into the fast recovery mode and transmits DACK’s with a bit set to denote interference (IDACK’s) to the TCP source. The “Active Transmitter” at the TCP source then moves into fast retransmit mode and retransmits the lost packets without reducing the size of either its sending window (CWND) or the slow start threshold (SSTHRESH).

3.3.3 Explicit Bad State Notification (EBSN)

Explicit Bad State Notification (EBSN) (see [5]) proposes a mechanism to inform the TCP source of the failure of a wireless link. In case the Base Station (BS) is unsuccessful in transmitting data to the wireless host over the wireless link, it sends an EBSN to the TCP source. The TCP source, thus recognizes the packet loss to be due to interference and hence does not decrease its congestion window.

3.3.4 Explicit Loss Notification (ELN)

Explicit Loss Notification (ELN) (see [9]) adds an option to TCP acknowledgements. Here the TCP sink differentiates between congestion losses and interference losses. When a packet is dropped on the wireless link, all future cumulative Duplicate Acknowledgments (DACK’s) corresponding to the dropped packet are marked with an ELN bit to identify that the loss that occurred was not congestion related. Upon receiving the DACK’s the TCP source retransmits the lost packets without invoking its congestion control procedures.

3.3.5 Multiple Acknowledgements

The multiple acknowledgements (MACK’s) (see [4]) algorithm distinguishes the losses due to congestion or other errors on the wired link from the losses on the wireless link (see Figure 3.7). This method uses two types of acknowledgement’s (ACK’s) to isolate
the wireless links from the fixed network. Partial ACK’s are transmitted from the Base Station (BS) to the Fixed Host (FH) and regular ACK’s are transmitted from the Mobile Host (MH) to the FH.

If a TCP source receives the partial ACK for a packet, but does not receive the regular ACK for the same packet, it deduces that the packet is lost on the wireless link. However, if the TCP source does not receive either the partial ACK or the regular ACK for a packet the loss is deduced to have occurred on the fixed network. Once, the TCP source fixes the reason for the packet losses, it responds accordingly.
3.3.6 Mobility Awareness Incorporated As TCP Enhancement (MAITE)

Mobility Awareness Incorporated as TCP Enhancement (MAITE) (see [15]) is an attempt to use link layer messages to inform the TCP source of high Bit Error Rates (BER’s) on the wireless channel and it’s consequent disconnection. A Supervisory Host (SH) allows the implementation of specific controls over the wireless links. The mobile Host (MH) transmits a beacon signal to the SH to signal the state of the channel. The SH detects channel disconnections by noticing the lack of reception of the beacon signal or data from the MH over a period of time.

The link layer in MAITE informs the SH of a high BER in the channel if the CRC checks that it performs on frames received fail continuously over a period of time.

The transport layer can hence refrain from taking any congestion avoidance measures during a period of high BER or disconnection on the wireless link.

3.4 Hard-State Transport Layer Approaches

The hard-state transport layer approaches encompass different methods that physically split the TCP connection so that the losses on the wired link and the wireless link may be individually identified.

3.4.1 Indirect TCP (I-TCP)

Indirect TCP (I-TCP) (see [1]) splits an end-to-end connection between Mobile Host (MH) and a Fixed Host (FH) into two separate physical connections (see Figure 3.8).

One connection is between the MH and its mobile support router (MSR) at the Base Station (BS) over the wireless medium and another between the MSR and the FH over the fixed network. Data sent to the MH is first received by the MSR at the base station, it then sends an acknowledgement to the FH and then the data is forwarded to the MH.
This indirection helps shield the wired network from the uncertainties of the wireless network and the TCP/IP at the fixed host side need not be changed.

Any loss indicated by the TCP sink at the BS to the TCP source at the FH is treated as a congestion loss and the FH responds accordingly. However, if the MH indicates a loss to the BS, the BS understands this loss to be due to errors on the wireless link and retransmits data from its own cache.

3.5 Pure End-to-End Transport Level Approaches

The pure end-to-end transport level approaches try to solve the problem solely on the end-to-end basis. The advantage of these approaches is that there is no change or extra logic needed at the base station or the any intermediate hosts. But the transport layer of the stack at the source and the sink may need to be modified.
3.5.1 IRTCP

Interference Robust TCP (IRTCP) (see [3], [2]) is an attempt to prevent the loss of TCP throughput when TCP is being affected by interference and not due to congestion in the network. An IRTCP implementation includes the Pipe Snapshot algorithm (see Figure 3.9), the aggressive retransmission scheme at the TCP source, and the active receiver at the TCP sink.

The Pipe Snapshot algorithm is used to preserve the state of the TCP source in a TCP flow in case a packet loss is identified as being due to interference. The state variables at the TCP source that are preserved include the congestion window (CWND), the slow start threshold (SSTHRESH) and the retransmission timer (RTO).

![Flowchart: The Pipe Snapshot Algorithm Of The IRTCP Protocol](image)

Figure 3.9 The Pipe Snapshot Algorithm Of The IRTCP Protocol
The Aggressive Retransmission protocol at the TCP source is invoked to aggressively retransmit the lost data once the interference on the wireless link subsides. The aggressive retransmissions use the state variables that are captured by the Pipe Snapshot algorithm.

The Active receiver protocol is implemented at the TCP sink. When the TCP sink does not send an Acknowledgement (ACK) or a Duplicate Acknowledgement (DACK) the RTO at the TCP source times out. The TCP source then, automatically assumes congestion and invokes its congestion control algorithms. In IRTCP the active receiver at the TCP sink prevents the RTO at the TCP source from timing out by sending DACK’s at the rate of half of the most recent measure of the Round Trip Time (RTT). The active receiver thus helps in fast recovery from the interference mode.

3.5.2 FREEZE-TCP

Freeze-TCP (see [18] is a true end-to-end scheme, which does not require any support from intermediaries along the path. The implementation of Freeze-TCP does not involve any change in TCP code at the TCP source. The algorithm is implemented on the TCP sink at the Mobile Host (MH). This makes Freeze-TCP interoperable with existing networks.

In Freeze-TCP, The TCP sink identifies any impending disconnection of the wireless link, the cause of which may be a potential handoff or fading signal strength. The TCP sink notifies the TCP source at the Fixed Host (FH) of any impending disconnection by advertising an effective receiver window size (RWND) of zero. This Zero Window Advertisement (ZWA) prevents the TCP source to temporarily stop transmission of data without invoking any of its congestion control algorithms. The transmission of data is resumed when the TCP sink notifies the TCP source of a non-zero value of the RWND.
3.5.3 WTCP

WTCP (see [5], [11]) is another approach where the end-to-end semantics of the TCP flow are preserved. It was developed for Wireless Wide Area Networks (WWAN’s) where, the TCP algorithms failed by falsely assuming all packet losses as congestion losses.

The WTCP protocol tries to distinguish random losses from congestion losses by measuring and comparing the packet inter-arrival time with the packet inter departure time. Congestion is determined by calculating the relative delay between the packet inter arrival times and the packet inter departure times as the packets traverse the network.

WTCP uses rate-based transmission control scheme where the packets are transmitted by matching their inter-departure times with the inter-arrival times of the Acknowledgements (ACK’s). This shapes the data traffic and the TCP source never transmits a burst of packet transmissions.

3.5.4 TCP-Santa Cruz

The TCP-Santa Cruz scheme (see [6]) uses an idea similar to the one in WTCP. Here the Round-Trip Time (RTT) is not used by the TCP source as a metric for calculating the retransmission timer (RTO). The TCP source instead, depends on the inter-message delay and explicit notifications (Selective Acknowledgements) for retransmissions. Congestion is determined by calculating the relative delay that a packet experiences with respect to another as it traverses the network.

The relative delay that a packet experiences is used by TCP Santa Cruz as a measure of the queue developing over a bottleneck link along the path from the TCP source to the TCP sink. This measure identifies both the initial stages of congestion and the direction of congestion (whether along the path taken by data packets or the path taken by the acknowledgements). Thus losses may be identified as either being due to congestion or...
random in nature and the appropriate response may be initiated.

3.5.5 M-TCP

M-TCP (see [11]) is designed to work well in the presence of frequent disconnections and low bit-rate wireless links (see Figure 3.10). It maintains end-to-end semantics of the TCP layer.

![Diagram of M-TCP protocol](image)

Figure 3.10 The Working Of The MTCP Protocol

Spurious timeouts are responsible for killing more throughput than both losses due to errors and losses due to small congestion windows. In M-TCP, every TCP connection is split in two at the Supervisory Host (SH). The TCP connection from fixed host (FH) to the SH uses the standard, unmodified version of TCP while the connection between
SH and mobile host (MH) uses the modified version of TCP.

During transfer of data the FH sends a segment, which the SH receives and passes on to the MH. The Acknowledgements (ACK’s) flow back along the same path and the SH saves the latest ACK that it receives from the MH before passing it on to the FH. In case, the MH is disconnected, the SH stops getting the ACK’s and thus knows that the MH has been temporarily disconnected. Then, the SH sends the latest ACK that it received from the MH, to the FH simultaneously advertising the effective receiver window size (RWND) as “zero”.

The TCP source at the FH, upon being notified that the RWND is zero, enters the “persist” mode and freezes all the timers related to the session. The TCP source now starts sending exponentially backed off “persist” packets to the SH. This keeps the TCP connection alive until the SH hears from the MH again and notifies the available RWND to the TCP source at the FH, so that normal transmission may resume.

3.5.6 TCP Westwood

The TCP Westwood scheme (see [16]) relies on bandwidth estimation for enhanced performance over wireless links. The idea here is that by measuring and averaging the rate of the Acknowledgements (ACK’s) returning from the TCP sink to the TCP source, the estimate of the total Bandwidth (BWE) available to that TCP flow may be determined.

In TCP Westwood a the congestion window algorithm at the TCP source is modified, so that the BWE calculated after the arrival of each ACK is used to determine the values of the congestion window (CWND) and slow start threshold (SSTHRESH) during the recovery from an congestion phase.

The BWE captures the nature of the loss. During a congestion loss the arrival rate of ACK’s will decrease and the calculated BWE will be lower, while during an interference loss the arrival rate of ACK’s will increase thereby increasing the calculated BWE. Since
3.5.7 Discriminating Congestion Losses From Wireless Losses Using Inter-Arrival Times At The Receiver

The scheme for discriminating congestion losses from wireless losses using inter-arrival times at the receiver (see [19]), assumes the cellular wireless architecture i.e. only the last link on the path is wireless and this link acts as the bandwidth bottleneck, for the connection.

The TCP source performs a bulk transfer of data. As the wireless link is the bottleneck link in the path the packets arrive at the Base Station (BS) faster than they can be dispatched and they end up being queued at the BS. This ensures that most packets transmitted by the BS to the Mobile Station (MS) over the wireless link are transmitted back to back, at regular intervals.

If an out of order packet arrives at the TCP sink on the MS and the inter-arrival time between this packet and the last packet is consistent with the average inter-arrival time, the loss of packets is assumed to be due to congestion. However, if an out of order packet arrives at the destination and the inter-arrival time varies i.e. is \((n+1)\) times the average inter-arrival time (where \(n\) is an integer), then \(n\) packets are assumed lost due to interference on the wireless link.

3.6 Ad-Hoc Network Protocols

Some ad-hoc network protocols have also been proposed to tackle the problems of interference and congestion in TCP flows.
3.6.1 TCP-F

The TCP-F scheme (see [12]) has been proposed for the multi-hop wireless network architecture (see Figure 3.11) i.e. data is transferred between a source Mobile Host (MH) and a destination MH via other intermediate MH's.

If an intermediate MH detects a route failure, due to which it can’t send the data any further, it sends a Route Failure Notification (RFN) to the source MH. Each intermediate MH that receives the RFN, invalidates all packets travelling through that failed route and prevents more incoming packets. The intermediate MH then tries to find an alternate route to the destination MH. If any alternate path exists, then packets are routed through that path, otherwise RFN is forwarded towards the source.

Figure 3.11 The Working Of The TCP-F Protocol
Upon receiving an RFN, the source MH goes into a “snooze” state i.e. it preserves all state information and waits until it hears an update on the network state from the MH that forwarded the RFN. When an alternate route to the destination has been established, the intermediate routers that forwarded the RFN, send a Route Reestablishment Notification (RRN) packet to the source MH, which then uses the saved state information to resume transmission at the same rate that it used prior to the route failure.

3.6.2 TCP-BuS

The TCP-BuS algorithm (see [7]) is very similar to the TCP-F algorithm and considers a multi-hop wireless network architecture (see Figure 3.12) where data is transferred between a source Mobile Host (MH) and a destination MH via other intermediate MH’s. The basic idea here is to use the buffering capacity of MH’s. TCP-BuS uses a source-initiated on-demand routing protocol for the underlying layer, which in turn uses two control messages the Explicit Route Disconnection Notification (ERDN) and the Explicit Route Successful Notification (ERSN), to maintain the route and to notify the source of route failures and route re-establishments.

The ERDN and the ERSN also differentiate between network congestion and route failure as a result of node-movement. The ERDN message is generated at an intermediate MH upon detection of a route disconnection. When the source MH receives the ERDN message it stops transmitting. Similarly after discovering a new path from the node that initiated the ERDN message the source MH is informed by using the ERSN message. On receiving the ERSN message the source starts retransmission.

The retransmission of lost packets due to congestion relies on a timeout mechanism. Upon the receipt of an ERDN message the source MH increases its Retransmission Timer (RTO) value to avoid retransmission during a disconnection. So, it becomes necessary that the destination MH request the lost packets’, otherwise they will be
retransmitted by the source MH only after the RTO latency.

For the TCP-BuS scheme to work without error the following rules must be followed:

1). All the packets from the intermediate MH that initiated the ERDN message, to the point where the MH previously existed must be flushed upon receiving an ERSN message.

2). All MH's that received the ERDN message for a particular destination MH must stop forwarding those packets, else packets will continue to flow to the MH that initiated the ERDN.

3). All intermediate nodes must time out and retransmit the ERSN message if they do not hear their upstream nodes forwarding the ERSN message, so that the ERSN message is successfully delivered to the source MH.
4 ICEI TCP “ICY” - IMPLICIT CONGESTION AND EXPLICIT INTERFERENCE DETECTION IN TCP

4.1 Motivation

In the preceding sections, we have discussed how the traditional paradigm of network architecture has changed from being a purely wired infrastructure to a combination of wired and wireless links connecting a wide variety of hosts. The wireless elements of the network are poised to outnumber the wired elements of the network in future. Though most of the wireless hosts today are essentially non-computing devices, they are rapidly evolving to provide more and more computing capabilities. We will soon see a phase where they will be able to host multiple applications simultaneously. We thus establish that an effort must be made to welcome wireless devices into the fold of the present network architectures.

We have observed that TCP/IP is the preferred protocol of the networking world. It’s absolute dominance over all other protocols leads us to believe that it will around for a long time to come. It is easier to envisage a scenario where TCP/IP evolves and changes to meet the challenges of the new network paradigm rather than to come up with a replacement for it.

In keeping with this view, we try to analyze the actual working of TCP and propose a few simple changes to it, to improve its throughput and enhance its performance on wireless and heterogeneous networks.

Our approach is motivated by and aims to satisfy the following objectives:
1). We ensure that any modifications made to the TCP layer should be consistent with the standards of the present TCP/IP network model. There should be no violation of the model, i.e. any new services that are provided to the TCP layer should be provided by the IP layer alone and even these should be kept to a minimum. Ideally, we would like to use the data available in the transport layer itself to glean any information that we may need to use. This would make the scheme easy to implement and maintain.

2). We restrict any modifications to the TCP/IP protocol stack to the Transport layer i.e. TCP and the services that are required from lower layers should be minimized. This makes sure that our scheme is compatible with those networks, which use TCP as the transport layer over other network layer protocols like ATM or Bluetooth. This makes our scheme portable across different protocol architectures used in wireless networks.

3). We avoid network overheads like data caching at different hosts and state signalling over the network. The flow of control and signalling packets in the network increase the data traffic and may themselves contribute to congestion in the network. Data caching at intermediate hosts requires more resources from the hosts and affects the scalability of the system.

4). We also make our scheme compatible with the other TCP implementations. Our scheme may be added as a variation of the existing TCP implementation and may also be optionally switched on and off depending on the actual nature of the network.

We develop a simple scheme to provide implicit Congestion and explicit interference detection in TCP to improve its throughput and hence its performance on wireless networks or heterogeneous networks.

The modified TCP is referred to as ICEI (Implicit Congestion and Explicit Interference) TCP pronounced “ICY” TCP.
4.2 The Case For ICEI TCP

The wireless network architecture that we consider while designing the ICEI TCP scheme is the cellular wireless network architecture. The TCP transmissions are assumed to be from one of the fixed hosts (FH’s) on the fixed (wired) network to one of the mobile hosts (MH’s) connected via a Base Station (BS). We assume multiple such TCP transmissions that terminate at an MH. There are two main reasons why we assume this network architecture for ICEI TCP:

1). The cellular network architecture is the most common wireless architecture in use today and are used by the vast majority of wireless devices like cellular phones and PDA’s.

2). We have established that with the increasing popularity and increasing computing power of the wireless devices, it will be commonplace for a wireless device like a cellular phone or a PDA to run multiple user applications simultaneously.

To the best of our knowledge ICEI TCP would be the first scheme to consider the case of multiple TCP transmissions to an MH from different FH’s on the fixed network.

ICEI TCP is designed as a pure end-to-end transport level approach that uses the end-to-end characteristic of the transport layer to function effectively. Like in other pure end-to-end transport level approaches ICEI-TCP does not require any help from the BS or the intermediate hosts to function since the ICEI TCP implementation is restricted to the TCP source at the FH and the TCP sink at the MH.

In designing the ICEI TCP scheme we have tried to make sure that this scheme detects congestion and interference separately and uses the combined information to make a decision regarding the subsequent transmission rate, by adjusting the congestion or sending window size (CWND). An informed adjustment of the CWND ensures a higher throughput for the ICEI TCP scheme and hence enhanced performance on the wireless network.
Congestion is a network characteristic and is caused when TCP sources keep increasing their CWND in an effort to grab more and more of the bandwidth available on network links that they share with other TCP flows. As the rate of transmission of each TCP flow increases its packets spend more and more time queued in the intermediate network routers. This increases the Round Trip Time (RTT) of the packets i.e the time from when a packet is transmitted, by the TCP source to the TCP sink, to the time when its Acknowledgement (ACK) is received by the TCP source. At the TCP source the value of Retransmission Timer (RTO) follows the RTT slowly with an inertia as determined by $\alpha$ (see Equations: 2.1, 2.2, 2.3). The difference in the RTO and the RTT values causes frequent time outs at the TCP source which forces the retransmission of packets. This increased network traffic further pushes the network into congestion.

In ICEI TCP a sender side modification is used to provide the TCP source with information regarding the probability of congestion along the links that are part of the path taken by the TCP flow. Whenever a packet loss event occurs (RTO time out, Arrival of Three DUPACK’s) the TCP source checks the probability of congestion before it adjusts its CWND. The TCP source thus does not react adversely to random errors in the network and thereby enhances the throughput of the TCP flow.

Interference is a physical phenomenon that may be attributed to the location of a wireless receiver. Interference may occur when the wireless receiver is located inside a tunnel, or near a strong radio source like a radio antenna, or even due the presence of other wireless receivers that operate in the same frequency band.

Hand-off’s are fairly random events that form a leading cause of losses in cellular wireless networks. They are caused due to the mobility patterns of the wireless receiver. When a wireless receiver moves from the “cell” covered by one BS to a “cell” covered by another BS, the old BS has to “hand-off” the wireless receiver to the new BS. The resultant disruption in service causes TCP transmission losses.

ICEI TCP uses a receiver side modification at the TCP sink to calculate the prob-
ability of interference along the wireless links since the wireless link losses (interference and hand-offs') are caused due to the location and mobility patterns of the wireless receiver. This interference probability is transmitted to the TCP source as part of the ACK.

### 4.2.1 Handling Of Congestion In ICEI TCP

As each TCP source tries to grab more and more of the available bandwidth on shared links, by increasing its individual sending / congestion window (CWND) size, the actual data transmitted from each source increases. An increase in the data actually translates into an increase in the number of packets that the network layer (IP) has to transport over the network.

As the number of packets increase the time that each packet spends in the queues or buffers of the routers along the path from source to destination increases, thereby increasing the Round Trip Time (RTT) for each packet i.e. the actual time difference between when a packet leaves the TCP source and when the TCP source receives it’s ACK.

The TCP source in ICEI TCP measures the actual RTT whenever it receives an ACK from the TCP sink.

\[
RTT_i = (ToT_i - ToR_i)
\]  

where:

- \(RTT_i\) is the RTT value of the “i” th packet in seconds,
- \(ToT_i\) is the Time of transmission of the “i” th packet in seconds,
- \(ToR_i\) is the Time of reception of the ACK corresponding to the “i” th packet in seconds.

The TCP source also compares the RTT of each ACK’ed packet with the lowest RTT
seen during the life time of that TCP transmission to calculate the Congestion Ratio (CR) using the following formula:

\[ CR_i = \frac{RTT_i}{RTT_{\text{min}}} \]  \hspace{1cm} (4.2)

where:

- \( CR_i \) is the CR of the “i” th packet and is dimension less,
- \( RTT_i \) is the RTT of the “i” th packet in seconds,
- \( RTT_{\text{min}} = \text{MIN} \{RTT_0, RTT_1, RTT_2, ..., RTT_i\} \).

In case the network is either in equilibrium or not bandwidth constrained, we see that the CR for each packet remains almost constant. Wild fluctuations in the CR indicate that the network state is constantly in flux and that multiple TCP flows are trying to grab the bandwidth on shared links in the network.

The interesting thing to note here is that each IP packet might follow a different path to reach the destination. So it may be argued that the information that each subsequent CR provides to the TCP source is actually for a different path in the network. We counter this argument with the following two points:

1). Once a path is established from the source to the destination in a network, a majority of the packets in that TCP transmission use this path.

2). In case certain packets follow an alternate path, this in itself is indicative of the state of the network at that point in time, i.e. the TCP source calculates CR’s which are indicative of congestion along this new path and that is exactly what the TCP source should be concerned with.

Each TCP source in the ICEI TCP maintains a Congestion History (CHISTORY) (see Figure 4.1) i.e. one for each TCP flow. The CHISTORY is a finite size queue, which maintains the latest “N” number of entries of the CR for each TCP flow, where “N” is the size of the CHISTORY. A new CR entry is pushed into the CHISTORY of a TCP
source when it receives an ACK.

Since the CR's are indicative of the congestion in a TCP flow, we use a simple formula to calculate the probability of congestion (CPROB) for each TCP flow as follows:

\[
CPROB_k = 1 - \frac{\gamma}{N}
\]  \hspace{1cm} (4.3)

where:

CPROB\(_k\) is the CPROB of the "k" th TCP flow,
\(\gamma\) = Number of CR's such that:
\([(CR_{mean} - CR_{sdev}) \leq CR \leq (CR_{mean} + CR_{sdev})]\),
\(CR_{mean}\) = Arithmetic Mean of the CR values in the CHISTORY of the "k" th TCP
source,

\[ CR_{sdev} = \text{Standard Deviation of the CR values in the CHISTORY of the "k" th TCP source,} \]

\[ N = \text{Total number of CR's in the CHISTORY of the "k" th TCP source.} \]

To understand the significance of \( CPROB_k \) consider a scenario where the “k” th TCP flow is under the effect of congestion, causing large variations in the RTT’s of successive TCP segments. hence the value of \( \gamma \) is going to be much less than the value of \( N \). It follows that \( \frac{\gamma}{N} \ll 1 \). In this case \( CPROB_k \approx 1 \). Hence \( CPROB_k \) indicates the presence of congestion along the path of the “k” TCP flow.

The CPROB is calculated using data that is implicitly provided to the TCP source by the TCP sink so we call this the implicit detection of congestion in ICEI TCP.

### 4.2.2 Handling Of Interference And Hand-off’s In ICEI TCP

Interference in heterogeneous networks is caused due to the electromagnetic effects on the air-link between the wireless node and the base station. Electromagnetic effects may be very short lived and the air link may move from a bad state to a good state very rapidly. The effects of interference on the air-link or the communication channel between the Base Station (BS) and the Mobile Host (MH) causes a disruption simultaneously in all the TCP flows that are being received by the MH.

Let us now digress a little, to consider the scheme proposed in the RFC 2140, which details TCP Control Block Inter-dependence scheme. This memo makes the case for interdependent TCP control blocks, where part of the TCP state is shared among similar concurrent connections, or across similar connection instances. The TCP state includes a combination of parameters, such as connection state, current round-trip time estimates, congestion control information, and process information. This state is currently maintained on a per-connection basis in the TCP control block, but should be shared across connections to the same host. The goal here is to improve transient transport
performance, while maintaining backward-compatibility with existing TCP implementations.

![Diagram](image)

**Figure 4.2** Interference Profile In The ICEI TCP Protocol

The TCP sink in ICEI TCP is inspired by the TCP control block inter-dependance scheme, but it changes the location of the control block. The TCP sink detects a packet loss whenever an out of order packet arrives at the MH. In ICEI TCP each MH maintains an Interference Profile (IPROFILE) (see Figure 4.2), which is also implemented as a finite length queue. The IPROFILE contains the latest “M” entries of a tuple (I) of information about all TCP flows that terminate on the MH.

\[ I_i = \{I_{is}, I_{id}\} \]  \hspace{1cm} (4.4)
where: \( I_i \) is the \( "i" \) th tuple in the IPROFILE, 

\[ \lambda = \text{Number of I tuples in the IPROFILE such that: } \left[ I_{is} \neq \text{IP address of the TCP source of the } k \text{ th TCP flow} \right] \text{ and } \left[ I_{sd} \neq \text{Port number of the TCP sink of the } k \text{ th TCP flow} \right] \]

It has been observed that in low mobility networks the "good" state i.e. the low interference state when packet loss probability is almost nil lasts for the time duration corresponding to the transmission time of about 100 to 128 link layer frames. In high mobility networks the "good" state lasts for the time duration corresponding to the transmission time of about 6 to 10 link layer frames. In both low mobility and high mobility networks the "bad" state i.e. the high interference state when packet loss probability is almost 1, lasts for a time duration corresponding to the transmission time of about 2 to 3 link layer frames.

So we see that a low mobility network would experience on average at least one "good" state and one "bad" state in a time duration corresponding to the transmission time of about 130 link layer frames. We call this time duration an IPROFILE Unit (IPU).

The entries in the IPROFILE store a history of the losses experienced by TCP sinks on the MH. Though the MH stores a finite number of entries in the IPROFILE, it is not desirable to have the (I) tuples corresponding to an old interference phase to cloud future judgements. Hence the MH periodically resets the IPROFILE (i.e. destroys the old IPROFILE and begins a new one) when it receives "X" packets in order; where X is the number of packets that may be transmitted successfully during a period of 5-10 IPU's.

In ICEI TCP when any of the TCP sink's on an MH detect a packet loss they also check the IPROFILE to see if another TCP sink on the same MH has also lost a packet, in the recent past (corresponding to the current entries in the IPROFILE). In case this happens to be true, then the probability that the loss might have been on the shared
wireless link (due to either interference or hand-off’s) is high.

Since the I tuples in the IPROFILE are indicative of interference or hand-off losses we use a simple formula to calculate the Interference probability (IPROB) for each TCP flow as follows:

\[ IPROB_k = \frac{\lambda}{M} \]  \hspace{1cm} (4.5)

where:

- \( IPROB_k \) is the IPROB of the “k” th TCP flow
- \( \lambda \) = Number of I tuples in the IPROFILE such that \([I_i \neq k]\)
- \( M \) = Total number of I tuples currently in the IPROFILE.

To understand the significance of the value of \( IPROB_k \) consider the scenario when only the “k” th TCP flow on an MH is under the effect of congestion and it has been losing packets. Here the IPROFILE will only have entries from the “k” th TCP flow and so \( \lambda = 0 \). This means that \( IPROB_k \) will now be 0 indicating to the TCP source of the “k” th TCP flow that the wireless link is not under any interference. Hence, the value of \( IPROB_k \) indicates the effect of interference on the wireless link at the MH.

The TCP sink passes this value of the IPROB to the TCP source via the ACK. The TCP source in ICEI TCP relies on this notification of interference, and we thus see that interference is explicitly detected in ICEI TCP.

4.3 ICEI TCP Algorithms

We have seen now that ICEI gathers data to compute two metrics - the Congestion Probability (CPROB) and the Interference Probability (IPROB). The ICEI TCP algorithm now uses these inputs (see Figure 4.3) to decide on the TCP sender’s sending/congestion window size (CWND) and the Slow Start Threshold (SSTHRESH).
4.3.1 At The TCP Source

The ICEI TCP algorithm at the TCP source (see Figure 4.4) is more involved than the ICEI TCP algorithm at the TCP sink.

When the ICEI TCP source receives an ACK it calculates the actual Round Trip Time (RTT) and the total number of bytes transferred by the packet. This is then used to calculate the CR and this information is saved in the CHISTORY. The latest value of CPROB is now obtained from the CHISTORY and the latest value of IPROB is obtained from the ACK.

When an Retransmission Timeout (RTO) occurs, the ICEI TCP source checks the
latest CPROB and IPROB values. If IPROB is high and CPROB is low, the TCP source assumes that the packet loss is due to interference and the value of the Congestion Window (CWND) and Slow Start Threshold (SSTHRESH) are retained. The TCP source then resumes the normal transmission of data.

If the TCP source receives Duplicate Acknowledgement’s (DUPACK’s) in excess of the DUPACK Threshold (DTHSH), it again checks the latest values of CPROB and IPROB. If IPROB is high and CPROB is low, the TCP source assumes that the packet loss is due to interference and the value of the Congestion Window (CWND) and Slow Start Threshold (SSTHRESH) are retained. Further, the Retransmission Timer (RTO)
is not backed-off. The TCP source then resumes the normal transmission of data.

4.3.2 At The TCP Sink

The ICEI TCP algorithm at the TCP sink (see Figure 4.5) is a lot simpler to implement.

When the TCP sink in ICEI TCP receives a packet it checks to see if it is an in-order packet or an out-of-order packet. Whenever an out-of-order packet is received, the TCP sink assumes a packet loss and updates the IPROFILE with the I tuple.

![Figure 4.5 The Algorithm At The ICEI TCP Sink](image_url)

In case the packet is an in-order packet it increments a counter (ICOUNT). If the
value of ICOUNT exceeds ITHRESH (a threshold corresponding to the number of packets that may be transmitted in a time duration of 5-10 IPU’s), the TCP Sink resets the IPROFILE i.e. destroys the current IPROFILE and starts a new one.

For every packet received the TCP sink calculates the IPROB value from the IPROFILE and sends it to the TCP source via the ACK.

4.3.3 The Working Of ICEI TCP During Congestion And Interference

When congestion occurs along the path taken by a TCP flow, packets are dropped. The TCP source recognizes that a packet has been dropped when it experiences an RTO time-out. In this scenario the ICEI TCP relies on the locally available CPROB value and the value of IPROB provided by the latest ACK to detect congestion and hence reduce the CWND at the TCP source.

However, during a phase of interference on the wireless link, all packets travelling to and from the MH (DATA and ACK’s) along the wireless link may be lost. After the interference phase the TCP source receives DUPACKS for all packets that made it to the MH. The TCP source recognizes that interference has caused packet losses when the number of DUPACK’s exceeds the DTHSH. In this scenario TCP relies on the current value of CPROB and the value of the IPROB provided by the latest DUPACK to detect interference and hence retransmit the DATA with the same TCP state parameters.
5 ICEI TCP - IMPLEMENTATION, SIMULATIONS AND RESULTS

5.1 ICEI TCP Implementation Details

The ICEI scheme is compatible with all flavors of TCP. For a performance comparison of the ICEI TCP scheme with any of the TCP flavors, we would have to first implement the ICEI TCP scheme on that flavor of TCP, and then test the performance of that flavor of TCP with and without the ICEI TCP scheme.

As a representative case we have implemented the ICEI TCP scheme on the regular TCP protocol implemented in the ns2 (network simulator) environment and compared the simulation results of the regular TCP algorithm with and without the addition of the ICEI TCP scheme.

The files in ns2 that were modified for the ICEI implementation include:

1) tcp.h

We have added a structure to the tcp.h file called congestion_history. The structure congestion_history contains the data types and functions to create and manage the Congestion History (CHISTORY) queue and to calculate and store the Congestion Probability (CPROB). A variable of this type is declared in the TcpAgent class. Hence each instance of the TCPAgent class will have an object of type congestion_history, making sure that each TCP source has its own CHISTORY. Further the structure hdr_tcp has been modified to include a variable inpoption_, which stores the Interference Probability (IPROB) in the ACK packet and returns the same to the TCP source.
2). tcp.cc

In the file tcp.cc, we have modified the `dupack_action` function in the TcpAgent class to check the IPROB and CPROB of the TCP flow before initiating the action for duplicate acknowledgements. Likewise in the timeout function in the TcpAgent class we have added a check for the IPROB and the CPROB of the TCP flow before initiating the timeout action and the retransmission timer back-off.

3). tcp-sink.h

In the file tcp-sink.h, we have created a structure called the `interference_profile`. The structure `interference_profile` contains the data-types and functions to create and manage the Interference Profile (IPROFILE) and also return the Interference Probability (IPROB) for a TCP sink on a Mobile Host (MH). A global variable of this type is currently declared in the file tcp.cc but since we need to maintain an IPROFILE for each MH, we incorporate the checks within the functions that calculate the IPROB to make sure that we only check the IPROFILE entries of that MH.

We have also defined a structure called the `goodput_profile` has been defined in the tcp-sink.h file. This contains the data-types and functions to manage the goodput profile and it returns the total number of bytes that are transferred to the application in each TCP flow. This enables us to measure the goodput of each TCP flow. A global variable of this type is declared in the file tcp-sink.cc to keep track of all the TCP flows.

4). tcp-sink.cc

Apart from adding the global variables that maintain the IPROFILE and the goodput profile to the tcp-sink.cc file, we have also made changes to the recv function in the TcpSink class to push an entry in to the goodput profile every time a TCP sink passes a set of bytes to it’s application. The function to display the total number of bytes transferred to the application at fixed intervals is also called here. We have also modified the ack function in the TcpSink class to check if a packet loss has occurred (receipt of an out-of-order packet) and if so, push an I tuple entry into the IPROFILE
and also calculate the latest IPROB of the TCP flow. The *inpoption* field of the ACK packet header is updated with the IPROB value and the ACK is sent to the TCP source.

Figure 5.1 A Heterogeneous Network Model With Wired And Wireless Links
5.2 The Heterogeneous Network Model

We have based our simulation studies on a heterogeneous network model (see Figure 5.1) which has a combination of wired and wireless links. The wireless architecture that we have considered is the cellular or Base Station (BS) architecture.

For the purposes of our simulation we have used wired links with a bandwidth of 10Mb. A low bandwidth (1Mb), low delay (1ms) and highly error prone wired link was used to simulate a wireless link. The queues at each of the hosts in the network was assumed to be standard Drop-Tail queue.

To simulate interference on the wireless link we used the two state error model provided in the ns2 environment. The two states corresponding to the “good” state (low interference state) and “bad” state (high interference state) are governed by two exponentially distributed random variables, which form a two state Markov chain.

It has been observed that in low mobility networks the good state i.e. the low interference state when packet loss probability is almost nil lasts for the time duration corresponding to the transmission time of about 100 to 128 link layer frames. In high mobility networks the low interference state lasts for the time duration corresponding to the transmission time of about 6 to 10 link layer frames. In both low mobility and high mobility networks the bad state i.e. the high interference state when packet loss probability is almost 1, lasts for a time duration corresponding to the transmission time of about 2 to 3 link layer frames.

The exponentially distributed random variable corresponding to the duration of the bad state was fixed as the transmission time for 3 link frames. The exponentially distributed random variable corresponding to the duration of the good state was varied as the transmission times of 128, 100, 50, 75 and 25 link frames.
5.3 The Goodput Metric For TCP Performance

When measuring the performance of TCP transmissions regular metrics like packet loss rate and number of packets received at the TCP Sink do not make sense. This is because in a reliable end-to-end protocol like TCP it is possible that the packets that are received at the TCP sink may be duplicate packets, retransmitted by the TCP source, which are going to be discarded anyway by the TCP sink. Further, the packets that arrive at the TCP sink may be a set of out of order packets which are useless to the application layer until all the missing packets arrive.

The metric that we measure as a yardstick of performance for these simulations is the Goodput of the TCP flow. Goodput is defined as the total number of packets that have been transmitted to the application in a given time interval.

\[
\text{Goodput}_{jt} = \frac{\Delta}{t} \quad (5.1)
\]

where:

- \( \text{Goodput}_{jt} \) is the goodput of the “j” th TCP flow at a host at time “t”,
- \( t \) = time in seconds,
- \( \Delta \) = Total number of bytes passed by the TCP sink of the “j” th TCP flow to the application in time “t”.

ICEI TCP assumes multiple TCP flows to an MH over a wireless link hence, the performance metric that we consider for comparison is the total goodput of the wireless link, i.e. the sum of the individual goodput measures of all the TCP flows that share the wireless link.

ICEI TCP is designed to work on wireless and heterogeneous networks. Hence we consider the two simulation scenarios that are most common in such networks.
5.4 Simulation Scenario 1 - TCP Flows With Predominantly Interference Losses

In scenario 1 (see Figure 5.2), we consider a heterogeneous network where the TCP flows predominantly experience interference losses and very low congestion losses.

To simulate scenario 1, we add TCP flows from FH's to the MH in our heterogeneous network model such that none of the TCP flows share any of the wired links with any other flow. All the TCP flows share the wireless link and the loss module is placed on this link so as to simulate interference losses.

Figure 5.2 Simulation Scenario 1 - TCP Flows With Predominantly Interference Losses

We have studied the performance of ICEI TCP in comparison with TCP for varying error probabilities of the wireless link.
Figure 5.3 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 128 MAC Frames And Average Bad State = 3 MAC Frames

Figure 5.4 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 100 MAC Frames And Average Bad State = 3 MAC Frames
Goodput
(Avg Good State = 75 MAC Frames, Avg Bad State = 3 MAC Frames)

Figure 5.5 Results for Scenario 1 - ICEI TCP and TCP Goodput for Average Good State = 75 MAC Frames and Average Bad State = 3 MAC Frames

Goodput
(Avg Good State = 50 MAC Frames, Avg Bad State = 3 MAC Frames)

Figure 5.6 Results for Scenario 1 - ICEI TCP and TCP Goodput for Average Good State = 50 MAC Frames and Average Bad State = 3 MAC Frames
Figure 5.7 Results For Scenario 1 - ICEI TCP And TCP Goodput For Average Good State = 25 MAC Frames And Average Bad State = 3 MAC Frames

Figure 5.8 Results For Scenario 1 - Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods
Average Goodput Increase of ICEI TCP over TCP
(Avg Bad State = 3 MAC Frames)

Duration of Good State (MAC Frames)

Figure 5.9 Results For Scenario 1 - Confidence Intervals For The Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods

Figure 5.10 Simulation Scenario 2 - TCP Flows With Both Congestion And Interference Losses
5.5 Simulation Scenario 2 - TCP Flows With Both Interference And Congestion Losses

In scenario 2 (see Figure 5.10), we consider a heterogeneous network where the TCP flows experience both interference losses and congestion losses.

To simulate scenario 2, we keep all the TCP flows to the MH from the simulation scenario 1 and add a further two more flows to the heterogeneous network. These addition flows are exclusive to the wired links i.e. they flow from one FH to another in the network and these flows compete for bandwidth along the wired links with the other flows from the FH's to the MH.

As in the previous scenario, we have studied the performance of ICEI TCP in comparison with TCP for varying error probabilities of the wireless link.

![Graph showing Goodput comparison between TCP and ICEI TCP with and without error for average good state of 128 MAC Frames and average bad state of 3 MAC Frames.]

Figure 5.11 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 128 MAC Frames And Average Bad State = 3 MAC Frames
Figure 5.12 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 100 MAC Frames And Average Bad State = 3 MAC Frames

Figure 5.13 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 75 MAC Frames And Average Bad State = 3 MAC Frames
Figure 5.14 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 50 MAC Frames And Average Bad State = 3 MAC Frames

Figure 5.15 Results For Scenario 2 - ICEI TCP And TCP Goodput For Average Good State = 25 MAC Frames And Average Bad State = 3 MAC Frames
Average Goodput Increase of ICEI TCP over TCP
(Avg Bad State = 3 MAC Frames)

Figure 5.16 Results For Scenario 2 - Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods

Average Goodput Increase of ICEI TCP over TCP
(Avg Bad State = 3 MAC Frames)

Figure 5.17 Results For Scenario 2 - Confidence Intervals For The Average Increase In ICEI TCP Goodput Over TCP Goodput For Varying Good State Periods
5.5.1 Discussions On The Results

The simulation results for the ICEI TCP and TCP Goodput with increasing Error Probabilities (EP’s) on the wireless link provide us with the following insights:

1). The ICEI TCP solution performs better than TCP in case of scenario 1 where the TCP flow predominantly suffers interference losses on the wireless link. The figures: 5.3, 5.4 and 5.5 show the increase in performance for low mobility networks while the figures: 5.6 and 5.7 show the increase in performance for high mobility networks.

2). The average percentage increase in performance with increasing mobility of the wireless host, in case of scenario 1 (predominantly interference losses), is shown in figure: 5.8 and the confidence intervals for the average percentage increase in goodput is shown in figure: 5.9.

3). The ICEI TCP solution also performs better than TCP in case of scenario 2 where the TCP flows suffer from both interference losses along the wireless link and congestion losses along the wired links. The figures: 5.11, 5.12 and 5.13 show the increase in performance for low mobility networks while the figures: 5.14 and 5.15 show the increase in performance for high mobility networks.

4). The average percentage increase in performance with increasing mobility of the wireless host, in case of scenario 2 (both congestion and interference losses), is shown in figure: 5.16 and the confidence intervals for the average percentage increase in goodput is shown in figure: 5.17.

5). In both scenario 1 and scenario 2 the increase in the performance of ICEI TCP for high mobility networks (where the good state = 50 frames, 25 frames) is higher than the increase in performance of low mobility networks (where the good state = 128 frames, 100 frames, 75 frames). This is observed in the figures: 5.9 and 5.17. In case of high mobility networks, packet losses are very frequent. This causes the constant reduction of the TCP CWND and constant retransmission of packets which drastically reduces the
TCP goodput. However, ICEI TCP is able to detect interference and hand-off losses in a high mobility environment. Hence it does not reduce its CWND and thereby obtains a better goodput.
6 CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

A close examination of the simulation results shows us that the ICEI TCP scheme is able to provide an improvement in the overall goodput of the wireless link when we subject the wireless link to either interference losses alone or both interference and congestion losses.

The identification of congestion and interference in ICEI TCP is probabilistic. In case of a chaotic protocol like TCP a deterministic identification will require the use of explicit control packets and will incur messaging overheads. This may not be a scalable solution for multiple flows.

Also for the ICEI TCP scheme to work, we need to have a minimum of at least two TCP flows that are being received by the wireless host. However with the expected growth of the wireless devices market and with the increase in the number of services provided to wireless customers it won’t be long before it is commonplace to have multiple TCP flows terminating on a single wireless host.

6.2 ICEI TCP - A Footnote

The ICEI TCP algorithm has a few interesting aspects that we shall discuss in this section.

ICEI TCP does not wait until a packet loss occurs to investigate the cause. It con-
tinuously monitors the TCP flow and tries to calculate the probability of the flow being affected by congestion or interference upon the reception of each ACK. This information is then used to decide on the sending window size at the ICEI TCP source, as also the actions to be performed during a timeout or when the number of DUPACKS received are above the set threshold.

The ICEI TCP protocol does not introduce any additional control packets or signalling packets along the network, and thereby it does not add to the network traffic, further causing network congestion. The only additional information that is required at the TCP source is the value of IPROB which is passed to the TCP source from the TCP sink using one of the options fields in the TCP ACK packets’ header.

In the ICEI TCP protocol, all communication and data gathering activities is restricted to the TCP layer alone. All the layers from the network layer downwards may be kept completely ignorant of the functioning of ICEI TCP. This makes the ICEI TCP scheme compatible with existing TCP implementations and also compatible with other protocol suites which may use lower layer protocols other than IP (like Bluetooth or ATM).

6.3 Future Work

We have analyzed the working of the ICEI TCP scheme by comparing it’s goodput with the regular TCP implementation. We have identified some of the following areas for future research in ICEI TCP.

1). Different flavors of TCP implement different algorithms for operation. One aspect of the ICEI TCP scheme that would be interesting to explore is its performance when implemented on the different flavors of TCP.

2). In our experiments to analyze the performance of ICEI TCP we have assumed a cellular wireless network architecture. However, ICEI TCP is a pure end-to-end solution
and as such it is not constrained by a specific network architectures. It would be interesting to investigate the performance of ICEI TCP in other wireless LAN architectures like the multi-hop wireless ad-hoc networks using Bluetooth or 802.11 based wireless LAN’s.

3). As the demand for wireless devices and the quality of applications that are available to wireless customers increases the bandwidth available to wireless devices is bound to increase. Hence another area that would be interesting to explore is the effect of varying the bandwidth of the wireless link on the performance of ICEI TCP.
BIBLIOGRAPHY


ACKNOWLEDGMENTS

My major professor Prof. Arun K Somani has been a very inspiring teacher and guide. I acknowledge with gratitude his constant guidance, direction and assistance both with the technical and non-technical aspects of my research.

I gratefully acknowledge the presence of Dr. Manimaran Govindarasu and Dr. Johnny Wong in my committee. I thank them for their valuable guidance during my research and for their comments towards improving my work.

Anirban Chakraborty from the Real-time Systems and Networking Lab has been a good friend and in many ways my mentor and guide in my work with NS (the Network Simulator). I thank him for all his help towards my research and also for teaching me how to drive! Special thanks are due to ECpE graduate student Hao Li who has helped me out at a few key points during my research with NS.

There have been times during my stay at Iowa State University when my spirits were at an absolute ebb. Through all such times my roommates - Pallab Datta and Murali Ravirala (Studman) and more recently Abu Sebastian (Boo) have always been very encouraging and understanding. I thank them for being my friends and my family in Ames.

I have made some great friends in Ames. I have to specially thank - Pritesh Jasapara (Thakur), Rajesh Ramanujam, Jacob Purackl (Jake), Nikhil Ranade (Randy), Vipul Katyal, Vikas Yadav, Tathagata De (Tat), Amruta Karandikar and Anupreet Minhas (Anu) for making life in Ames fun, interesting and eventful. They have made Iowa State University an unforgettable part of me.