Modeling and Performance Analysis of TCP traffic in a Bluetooth Scatternet

A thesis presented

by

Rajasekaran Venugopal

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Master of Science

in the subject of

Computer Science

The University of Manitoba

Winnipeg, Manitoba

August 2006

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

Modeling and Performance Analysis of TCP traffic in a Bluetooth Scatternet

BY

Rajasekaran Venugopal

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

MASTER OF SCIENCE

Abstract

Bluetooth is a wireless communication technology for ad-hoc networks operating at 2.4 GHz ISM (Industrial, Scientific, Medical) band with the initial goal of cable replacement. Now it is used as a communication medium for personal-hand held devices such as laptops, PDA's, mobile phones, printers etc., thereby forming a Wireless Personal Area Network (WPAN). This results in a large amount of TCP traffic over the Bluetooth Network. The smallest unit of the Bluetooth network is a piconet comprising one master and up to seven active slaves. Larger networks are implemented by interconnection of piconets.

When TCP traffic is carried over a Bluetooth network, complex interactions are expected between congestion control mechanisms and the data link layer. In this research, these interactions were modeled and analyzed in a Bluetooth piconet running TCP new Reno and later the analysis was extended to a scatternet environment (two piconets connected through a bridge). Modeling was implemented using the Artifex simulation tool. The research was performed both in bit error free and error-prone environment. Modeling of the error-prone environment (Bluetooth RF link) was done using a two-state markov channel. The performance metrics TCP goodput, RTT (round-trip time), congestion window size, number of fast retransmits and timeouts were studied in response to a number of data link parameters including buffer size and scheduling algorithms.

The results indicate that in an error free environment satisfactory TCP performance can be obtained by proper dimensioning of the Bluetooth data link parameters. However, in an error-prone environment, TCP performance deteriorates drastically.

It was also found that the error-affected packets are mostly retransmitted only by means of timeout.

Keywords: Bluetooth, Piconet, Scatternet, TCP

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisor Dr. Jelena Mišić, for her continued support and guidance. I would like to thank Dr. Vojislav B. Mišić for his help in learning Artifex simulation software.

I am very grateful to thank Dr. Ekram Hossain and Dr. Neil Arnason for taking the time to review my thesis and for being a part of my defense committee.

A special thanks to Mr. Gilbert Detillieux for his help in solving the multiple access problem in Artifex.

I would like to thank all my friends and colleagues in Winnipeg for their friendship and cooperation

Dedication

This thesis is dedicated to my parents, my brothers, and to my sister.

Chapter 1

Introduction

Bluetooth [2] is an emerging communication standard for wireless Personal Area Networks (WPAN) [3]. Bluetooth was originally presented in 1998 by SIG (Special Interest Group) with its five promoters, namely, Ericsson, Nokia, IBM, Toshiba, and Intel. But now SIG has more than 4000 members in it. The name Bluetooth is taken from the 10'Th century Danish King Herald Blatand (Harold Bluetooth in English) who was instrumental in uniting warring factions. Bluetooth devices organize themselves to form a piconet, which consists of one master and up to seven active slaves. Networking capabilities of Bluetooth is enhanced when two or more piconets combine to form a scatternet through bridging devices.

Key features of Bluetooth include

- Robustness,
- Low cost,
- Low power consumption, and

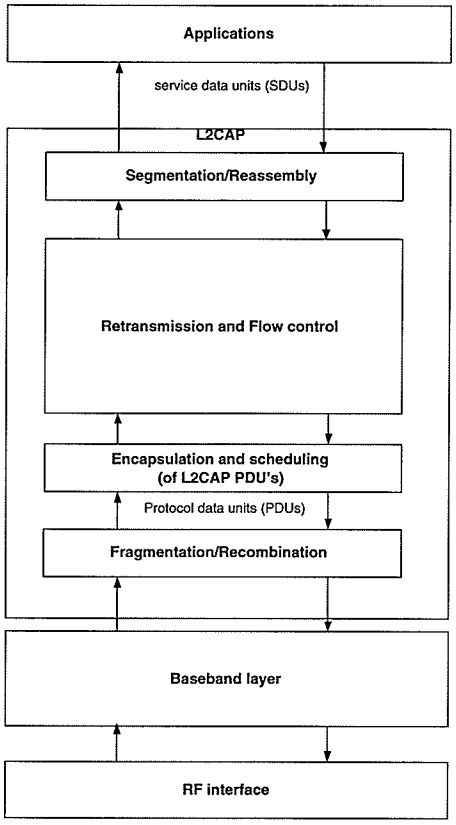
- Ability to handle both synchronous (e.g. voice) and asynchronous (e.g. data) traffic simultaneously.

1.1 Motivation

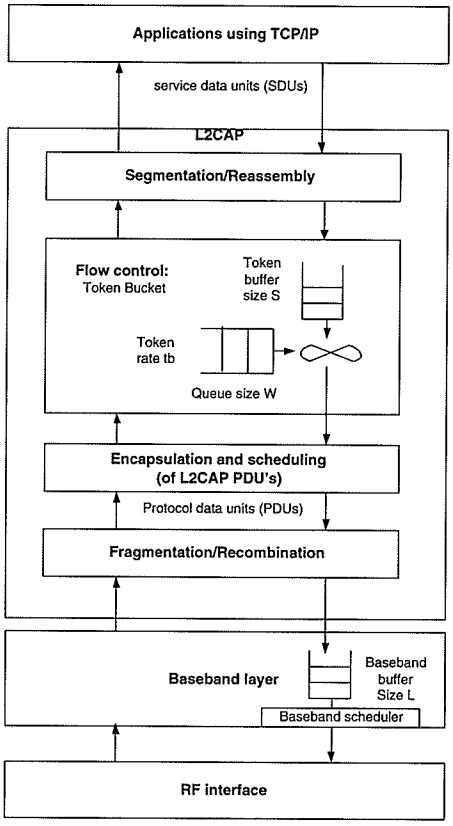
Bluetooth, whose initial goal was to replace cable technology, is now used as a communication medium for personal hand-held devices such as laptops, mobile phones, PDA's, etc. This results in a large amount of TCP traffic.

The performance analysis of data traffic over Bluetooth which was carried out in ideal conditions based on scheduling techniques (the manner in which the master visits the slaves and exchange data with it) at the data link layer in [4; 5; 6; 7] are no longer applicable to TCP traffic. By ideal conditions it is meant that the data (TCP/UDP) segments created by the application units in slaves are passed to a L2CAP layer (layer that provide connection oriented and connection-less data services to the upper layer protocol) where they are segmented into baseband packets and are placed in an infinite size buffers (baseband buffer). The baseband packets are then transmitted into the wireless radio channel according the scheduling policy. Since the buffers are infinite, TCP congestion window increases thereby increasing the transmission rate which increases the end-to-end delay resulting in timeouts more or less independently of the scheduling scheme.

In order to cater to such TCP applications, the recent Bluetooth 1.2 standard [1] allows each L2CAP channel to operate in one of the following modes: basic L2CAP mode [8] (the mode that was supported in the previous Bluetooth 1.1 version [8]), Flow control [1], or Retransmission mode [1]. Although, all the three modes offer



(a) Original Specification from [1]



(b) Flow control implemented using token bucket adapted from [9]

Figure 1.1: Architectural blocks of the Bluetooth L2CAP layer

segmentation, the latter two control buffering and flow rate through protocol data unit (PDU) sequence numbers and limiting the required buffer space [1]. Figure 1.1(a) shows the architectural blocks [1] of the L2CAP layer.

When TCP traffic is carried over Bluetooth, complex interactions are expected between TCP congestion control mechanisms and data link layer (L2CAP flow control and the scheduling schemes). To the best of my knowledge, the only research in which

these interactions were studied was performed by Mišić et al. [9] using TCP Reno [10]. However in [9], analysis was restricted to a piconet and was carried over a Perfect Physical Layer [PPL] where there is no packet loss in the wireless medium (error free environment).

This research is an extension of the work performed by Mišić et al. [9]. The interactions taking place between TCP congestion control mechanisms, and data link layer (L2CAP flow control and the scheduling schemes) when TCP traffic is carried over Bluetooth network are modeled and analyzed over a Bluetooth piconet, running TCP new Reno [11]. Performance analysis is performed both in error free and error-prone [Imperfect Physical Layer (IPL)] environment through simulations using Artifex simulation software [12]. Later, the analysis is extended to a scatternet involving two piconets connected by a bridge. System model is explained in section 3.1. Finally, the results of PPL and IPL was compared and analyzed.

1.2 Contributions of the Research

This research, addresses the interactions that occurs between TCP congestion control mechanisms, and data link layer (L2CAP flow control and the scheduling schemes) when TCP traffic is carried over Bluetooth network. The cross-layer analysis performed in this research can be used to improve the network performance.

In this research a comparative analysis of PPL and IPL, when TCP traffic is carried over Bluetooth, is performed. The analysis of this research can be used to study the performance of TCP over wireless Bluetooth networks.

1.3 Road map of the Thesis

Rest of the thesis is organized as follows

- In Chapter 2 background information and theory related to Bluetooth and TCP along with the work previously performed in this research area is discussed.
- Chapter 3 describes in detail the system model considered along with the approach adopted, and finally covers with the materials used for this research.
- Chapter 4 explains design of the simulator along with the method adopted to check the packet errors during transmission.
- Chapter 5 describes the code validation performed to confirm the integrity of the code.
- Chapter 6 presents the results followed by the discussion.
- Finally in Chapter 7 the thesis is concluded with future work.

Chapter 2

Background

2.1 Bluetooth Technology Overview

Bluetooth is a short range wireless communication technology for ad hoc networks. Bluetooth implements the specification standard for first two layers in the OSI (Open System Interconnection) model [13] namely physical and data link layer. In this section, the communication protocols in Bluetooth 1.2 specification [1] which are essential for this study is explained.

2.1.1 RF and Baseband

The functions that correspond to the PHY (physical layer) of the IEEE 802.11 wireless protocol stack are performed by the RF unit. Bluetooth operates in the unlicensed 2.4 GHz ISM (Industrial, Scientific, Medical) band. There is a total of 79 RF frequencies. The RF transceiver hops randomly over the available channel 1600 times/second to combat interference. The baseband layer is a layer, which is

responsible for packet transport over the physical link.

Bluetooth Network (Piconets and Scatternets)

Piconet: A group of devices that are synchronized to a common clock and hopping sequence share the same radio channel to form a small network piconet. Bluetooth provides both point-to-point and point-to-multipoint connection. In a piconet, one device takes the role of master (typically the device that initiates the connection) and all others (maximum of 7 active devices at a given time) act as slaves. No two slaves can communicate directly. All communications must be routed through the master. Data communication among Bluetooth devices is explained in subsequent sections.

Scatternet: Two or more piconets combine to form a large network namely the scatternet through shared device/devices. The shared device (bridge) acts as a gateway to transfer packets among Bluetooth devices in a scatternet. The bridge can be a master in one piconet and a slave in all other piconets denoted as Master/Slave (MS) bridge, or it can be a slave in all the piconets to which it is associated denoted as Slave/Slave (SS) bridge. The bridge has only one transceiver thereby making it active in only one piconet at a time. Bluetooth specification allows the functioning of the bridge by operating the slave devices in power saving modes such as sniff or hold mode (explained in detail in 2.1.3). Figure 2.1 shows a typical scatternet formed by three piconets in which piconets 1 and 2 are connected by a Master/Slave bridge and piconets 2 and 3 connected by a Slave/Slave bridge.

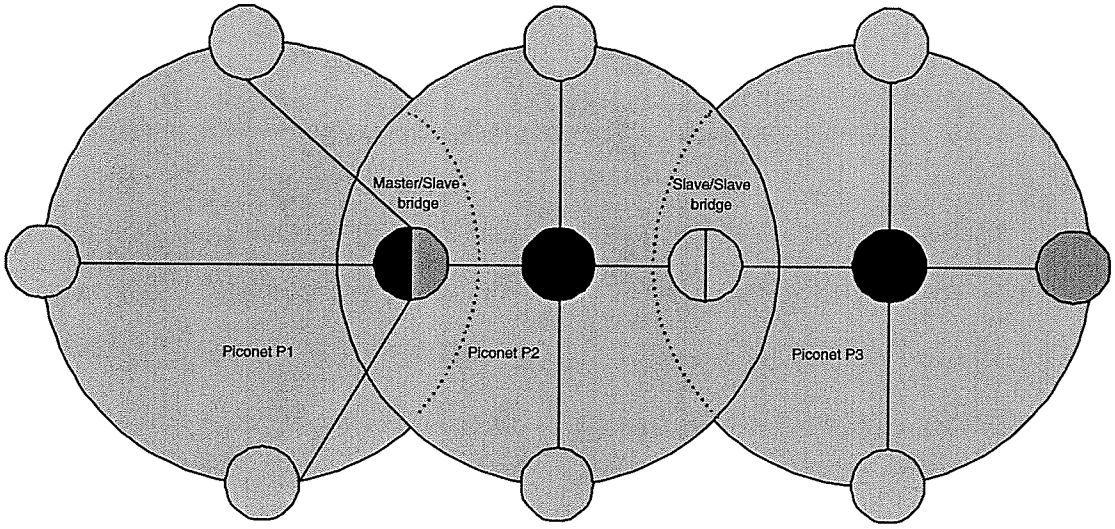


Figure 2.1: Scatternet formed with Master/Slave bridge and Slave/Slave bridge

Data Communication in Bluetooth

Bluetooth supports both voice and data traffic through Synchronous Connection Oriented (SCO) link and Asynchronous Connection-Less (ACL) links respectively. SCO link is a symmetric point-to-point link established between the master and a slave in a piconet by reserving the slots at regular intervals. SCO link is not further discussed as the focus of this research is in analyzing data traffic only. ACL link is a point-to-multipoint link between the master and all the slave devices used for data transmission.

In a Bluetooth network, slave devices cannot communicate directly i.e. slave devices can communicate only through the master. Data are transmitted among Bluetooth devices in packets (described in subsequent sections). A slave is allowed to transmit a baseband packet from its baseband buffer (also called as slave buffer or uplink queue) only if the master sends a packet from its baseband buffer (also

called as master buffer or downlink queue) to it in the previous time slot. The master maintains a number of downlink queues, one per each active slave.

Piconets (Polling/intra-piconet Scheduling)

In a piconet, data communication between the master and slave devices is through polling. Since master polls up to 7 active slaves, a schedule of polling has to be constructed. In polling, the master polls the slave by sending a data packet from its downlink queue or a POLL packet (packet with zero payload) if there is no data packet and in turn the slave responds by sending a data packet from its uplink queue (slave buffer) or a NULL packet (packet with zero payload) if the uplink queue is empty. Figure 2.2 shows the Queuing Model of a single piconet.

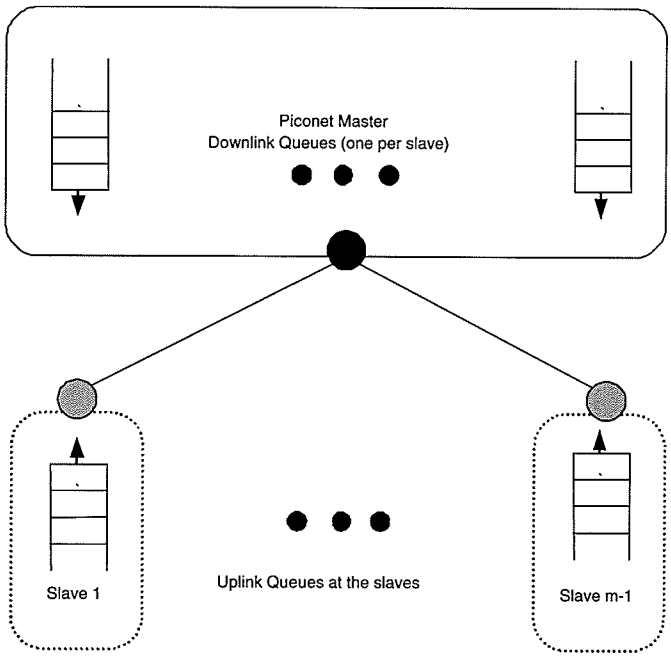


Figure 2.2: Queuing Model of a piconet

The packets experience queuing delay because it has to wait at the slave and mas-

ter before reaching the destination. Thus the intra-piconet scheduling (the manner in which the master visits the slaves and exchanges data with it) is an important criterion in the performance of data traffic. Section 2.3.1 describes the different polling schemes.

Scatternets (inter-piconet/bridge Scheduling) In a scatternet, data transmission from one piconet to an other is through bridging devices. Since the bridging device can be active in only one piconet at a given time, the bridge shares its time among the piconets it belongs to in a sequence. The packets for destination in other piconets are queued by the master and are delivered to the bridge during its residence (the period during which the bridge is present in a piconet). Figure 2.3 shows the Queueing model of a single piconet with a bridge.

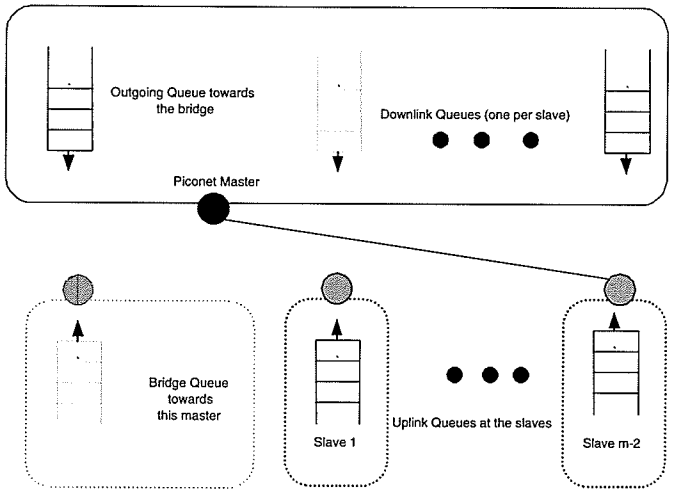


Figure 2.3: Queueing model of a single piconet with a bridge

The bridge experiences two synchronization delays when switching between piconets, namely, clock synchronization delay and frame synchronization delay. Clock Synchronization depends on the phase difference of the corresponding clocks with

value between of 0 to 2T (T-Bluetooth Time Slot = 625 microsecs). After the clock synchronization, frame synchronization takes place as there might be a frame transmission in progress, which can take a maximum of 8T but whose mean value depends on mean packet length. However, these synchronization delays are small compared to end-to-end delay in the scatternet and the effect is not noticeable if the switching is not made too often.

The bridge scheduling policy (the scheduling of the presence of bridge in a piconet, and the manner in which it is going to polled by the master during its residence) plays a vital role in the performance of data traffic in a scatternet. The absence of bridge during polling can significantly increase the end-to-end delays and overload the buffers in the master, thereby causing deterioration in the network performance. In section 2.3.2 various bridge scheduling policies are discussed.

Packet types in Bluetooth

Bluetooth defines many packet types related to the physical link. Since this research focuses on data traffic only, we only discuss packets defined for ACL link. Six ACL packets are defined in the Bluetooth 1.2 specification, namely, Data-Medium (DM) and Data-High (DH) rate packets each of which may occupy either 1, 3 or 5 slots. The use of FEC (Forward error correction) scheme by DM packets distinguishes itself from DH packets. FEC scheme is a technique for detecting and correcting errors by intentionally adding redudant information in the data payload. DM packets uses 2/3 FEC while DH packets don't have any FEC in data payload. 2/3 FEC is a (15,10) shortened Hamming code. Multi-slot ACL packets have larger payload capacity and

hence can carry more data resulting in better network performance provided there is less noise in the wireless medium (increase in number of slots makes it more vulnerable to noise). Table 2.1 shows the characteristics of the various ACL packets. In addition to these packets, Bluetooth also defines POLL and NULL packets used in master-slave communication (discussed already in polling).

Table 2.1: ACL Packets characteristics

Type	Slots	User Payload (bytes)	FEC	Asymmetric Max Rate (kbps total)
DM1	1	17	2/3 FEC	217.6
DH1	1	27	No	341.6
DM3	3	121	2/3 FEC	516.2
DH3	3	183	No	692
DM5	5	224	2/3 FEC	514.1
DH5	5	339	No	780.8

Error correction schemes

Bluetooth specification supports error correction through FEC and Cyclic Redundancy Check (CRC). The FEC in the data payload reduces the number of retransmission in the expense of reduced payload. In an error-free environment it is just overhead. Bluetooth specification also supports ARQ (Automatic Repeat Request) procedure. In ARQ scheme, packets are retransmitted until an acknowledgment is received.

2.1.2 Logical Link Control and Adaptation Protocol (L2CAP)

L2CAP layer (layer that resides over baseband layer) provides connection-oriented and connectionless data services to the upper layer protocol. Data from upper level

protocols are transmitted and received through the L2CAP layer. The L2CAP layer provides functions like multiplexing, segmentation and reassembly (SAR), optional flow control (in recent Bluetooth 1.2 specification [1]) and group abstraction (permits implementations for mapping groups on to a piconet). By multiplexing, the L2CAP is able to identify the upper-layer protocols. The SAR provides way for transmitting larger packets (TCP/IP packets) in a Bluetooth network. In section 2.3.3 various SAR mechanisms are explored.

2.1.3 Operational Procedure and modes

Operational procedures enable piconet formation for data communication to take place subsequently. The operational mode refers to how a Bluetooth enabled device stays connected with other Bluetooth devices in a piconet and exchanging data with them.

Piconet formation: Inquiry and Paging procedure

For Piconet formation, first the nearby Bluetooth enabled devices need to be discovered. This is done using an Inquiry procedure. The device that does it is referred to as an inquiring device. In the inquiry procedure, the inquiring device sends inquiry requests to the discoverable devices and gets responses from them. Once the nearby devices are discovered, a paging procedure is followed to enable piconet formation so that data communication can take place. In the paging procedure, the paging device (Master) sends a page signal and the slave device respond. Thereafter, the slave device synchronizes with the master clock and hopping sequence.

Connected Modes

In the connection state (where a connection has been established and packets can be sent back and forth), the slaves are in active mode listening to all the master transmissions. If the slave does not want to listen to all the master transmissions it can switch itself to one of the following power saving modes: sniff, hold, or park mode. Since there are up to 7 active slaves in a piconet, an unique piconet address is provided to each active slave. Sniff and hold mode retain the piconet address whereas in park mode the active address mode is released. In Sniff mode, the slave listens to the master only during periodic time slots called sniff slots. In hold mode, the slave detaches itself from the piconet for a specified time period, referred to as hold timeouts. The slaves in the inactivity period (both in sniff and hold mode) is free to perform other activities such as scanning, paging or even joining other piconets. Figure 2.4 schematically shows the connection state and modes.

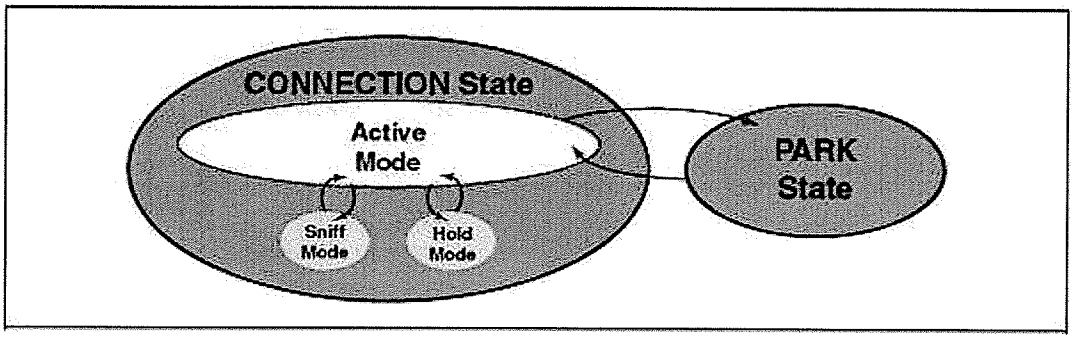


Figure 2.4: Connection State [1]

In this research, it is assumed that the Bluetooth network is already formed through discovery and paging procedure and hence subsequent discussion is focused

on performance of TCP traffic over Bluetooth network.

2.2 Transmission Control Protocol (TCP)

TCP [14] is a data transfer protocol that guarantees in-order delivery of packets from sender to receiver. Apart from providing reliability, TCP also provides congestion-control mechanism by adjusting the size of the sender's state variable congestion window (CWND). If TCP perceives that the network is free from congestion it increases the sender's rate by increasing the CWND. If the network is congested then sender's rate is decreased by reducing CWND. At any instant of time flight size (amount of unacknowledged data in the network) is minimum (CWND, RWND)where RWND is the latest advertised receiver window. How CWND is increased / decreased is based on TCP variant and state (explained in subsequent sections). TCP uses a retransmission timer to ensure data delivery in the absence of feedback from the receiver. The duration of this period is referred to as retransmission timeout (RTO). On case of a RTO, the packet will be retransmitted and CWND will be set to 1 MSS (Maximum Segment Size). More on RTO algorithm can be found in [15]

2.2.1 TCP variants

There are various version of TCP namely Tahoe, Reno [10], Vegas [16], Selective Acknowledgement (SACK) [17], and New Reno [11] all differing by the way in which congestion control is handled. The following three states are common to all the TCP variants: Slow start, Congestion Avoidance, Fast Retransmit.

Slow Start: In this state the growth of CWND is exponential. For each ACK

received, CWND is increased by one. Once the CWND equals threshold (THRESH), TCP shifts to Congestion Avoidance state. THRESH is a sender state variable which determines whether sending state is Slow Start or Congestion Avoidance. In Slow Start, the network is free from congestion. CWND in Slow Start starts at 1 MSS and THRESH is kept at a very high value of 64 MSS.

Congestion Avoidance: In this state, for each successful ACK, CWND is increased by MSS*MSS/CWND.

Fast retransmit: Packet loss is detected by TCP by means of triple duplicate acknowledgements or by means of timeouts. On receiving three duplicate acknowledgements, the segment with Sequence Number (SN) given by 3 duplicate acknowledgements is retransmitted and the system is entered into Fast retransmit state. THRESH and CWND are reset as follows: THRESH = max (flight size/2, 2 * MSS) & CWND = 1 MSS. On receiving a new data ACK, transition is set to Slow Start. During the Fast retransmit state, no action in done on receiving further duplicate acknowledgements.

Tahoe and Reno use a cumulative acknowledgement number field indicating the receiver has received all of the data up to the indicated byte. Tahoe and Reno both support fast retransmit. The addition of fast recovery in Reno differentiates it from Tahoe. During the fast recovery phase, CWND is set to THRESH + 3 * MSS and any duplicate ACK received during this phase CWND is increased by 1 MSS. When new data ACK is received, CWND is set to THRESH instead of 1 MSS (Tahoe case). Fast recovery phase prevents pipe from emptying after fast retransmit.

Reno has the ability to detect only a single packet loss. However, when multiple

packets get lost, the retransmission of lost segments can only happen by means of timeouts which is a very costly operation. SACK and New Reno have the ability to address multiple packet loss without timeouts happening. SACK can acknowledge out-of-order packets received by the sender. New Reno reduces the number of timeouts without SACK. On reception of a partial acknowledgement (acknowledges some but not all packets outstanding at start of Fast Retransmit), New Reno immediately retransmits the missing segment and continues to stay in fast recovery thereby avoiding timeouts. Thus when N packets are lost from a window, New Reno can be in fast recovery for N round trip times, retransmitting one more dropped packet each round trip time. Table 2.2 shows the summary of TCP variants.

2.2.2 TCP Over wireless

TCP was originally designed for wired networks where the error rate is very low and hence packet loss is due to congestion in the network. However, in a wireless transmission medium such as Bluetooth, packet loss also occurs due to interference in the wireless medium. Interference in the wireless link can occur for two reasons: the presence of multiple piconets [18; 19], and the presence of IEEE 802.11 devices in the same frequency band [19]. TCP is not aware of packet loss due to interference and considers it as an indication of congestion and decreases the congestion window, which in turn reduces the data traffic sent to the network. The limited network bandwidth in Bluetooth is thus under-utilized, thereby causing deterioration in the network performance.

Table 2.2: Summary of TCP Variants

	Tahoe	Reno	New Reno
Response to triple duplicate ACK	Do fast retransmit & enter Slow start	Do fast retransmit & enter fast recovery	Do Fast retransmit & enter fast recovery
Response to duplicate ACK after triple duplicate ACK	Do nothing	CWND = + MSS	CWND = +MSS
Response to partial ACK of fast retransmit	Enter Slow Start	Exit fast recovery, set CWND = THRESH & enter Congestion Avoidance * Leads to timeouts	Deflate CWND by amount of data acknowledged by partial ACK, retransmit the segment with SN given by partial ACK, remain in fast recovery * Prevents timeouts
Response to full ACK	Enter Slow Start	Exit fast recovery, set CWND = THRESH and enter Congestion Avoidance	Exit fast recovery, set CWND = THRESH and enter Congestion Avoidance

2.3 Related Work

Some of the works previously carried out in this research area are given in this section. To be more precise, the research performed by others is explained in terms of polling, bridge scheduling, and SAR schemes individually. Later the work performed by others involving TCP is explained in terms of network scenario, TCP version,

SAR algorithm, L2CAP flow control, baseband scheduling scheme, wireless bit error modeling with their limitations.

2.3.1 Polling

An effective polling scheme should maintain fairness while having low end-to-end packet delays. Polling schemes are classified as follows, based on the order in which the master visits the slaves, and the number of frames exchanged during a single visit to the slave:

- traditional polling schemes (fixed ordering of slaves),
- dynamic reordering of slaves
- Adaptive polling

Traditional polling schemes

In traditional polling schemes, the order in which the master visits the slave is fixed and the variable parameter is duration of the visit to each slave device. Mišić et al. [20] studied the performance of Bluetooth piconets using E-limited scheduling scheme [21] as intra-piconet scheduling scheme. In this scheme, the master stays with each slave for at most "M" frames (one downlink packet from master to slave and one uplink packet from slave to master), or until there are no more packets to be exchanged between the master and the slave device before moving to the next slave. The master visits the slaves in a round robin fashion. For $M = 1$, the E-limited service becomes 1-limited service, in which the master stays with each slave

for exactly one frame before moving to the next slave. For $M = \infty$, the E-limited service becomes an exhaustive service, in which the master stays with the slave until all the packets are exchanged between the master and the slave, and only then moves to the next slave. Many of the polling schemes [4; 5; 6; 7] proposed are just a variation of 1-limited and exhaustive service polling. Mišić et al.[20] showed that the E-limited scheme offers better performance than the 1-limited and exhaustive service while maintaining inherent fairness.

Dynamic reordering of slaves

Stochastically Largest Queue (SLQ) policy [22] and Longest Downlink Queue [LDQ] policy are based on dynamic reordering of slaves. In SLQ the Master serves the slave which has a highest sum of master and slave queues. Since the master does not know the status of slave queues, this scheme is possible only if the queue information is exchanged between the master and the slave explicitly. By contrast in LDQ, the master serves the slave which has the longest downlink queue. The master can adopt 1-limited, exhaustive or E-limited polling once the master finds the queue to be served. The problem with dynamic reordering of slaves is that fairness cannot be guaranteed.

Adaptive polling

In adaptive polling schemes, the duration of master visiting each slave is adjusted dynamically based on the current or historical traffic information.

The Fair Exhaustive Polling [FEP] scheme [6] uses a poll reduction technique. Polling is done using 1-limited service. When POLL/NULL is encounterd (no data in

both master and slave queue), that particular slave is not polled for some time. The inactive slave is restored once the master has some data to send or when the entire pool is reset.

Approaches similar to FEP were explored in Limited and Weighted Round Robin [LWRR] [4]. The difference is that instead of 1-limited service, E-limited service is adopted in LWRR. LWRR uses a weighted scheme by setting a Maximum Priority (MP) to each slave initially. If there is data exchange, MP is increased and if there no exchange MP is reduced. If MP is reduced to 1, then the slave has to wait MP-1 cycles to be polled again.

Pseudo-Random Cyclic limited slot-Weighted Round Robin [23] uses both poll reduction and reordering of slaves. In this scheme, the slaves are polled in a pseudo random order and the inactive slaves are not polled for certain time slots.

2.3.2 Inter-piconet scheduling

The inter-piconet scheduling schemes [9; 24; 25; 26; 27] described in this section can be broadly classified into two categories based on the concept of rendezvous points [28]. Rendezvous points are time slots at which the bridge should exist in a piconet to exchange data with the master.

Inter-piconet scheduling with rendezvous points

In [24; 25] sniff mode was used to support inter-piconet scheduling. Sniff modes is one of the baseband modes used for power consumption. The master agrees to transmit packets only at certain regular intervals and it is enough for the slave to

listen for packets only at that interval. The slave is active only in that interval thereby consuming power. A credit scheme is used to allocate the communication events in a link and also ensures fairness among the slaves under the same traffic load. However, in the case of non equal load and bursty traffic an additional overhead is needed. Moreover, the treatment of bridge and ordinary slaves is not differentiated and the methods to find the sniff intervals for bridge nodes are not discussed.

Pseudo random coordinated scheduling scheme (PCSS) [26]: In PCSS scheme, peer nodes assign meeting points and the sequence of meeting points follows a pseudo random sequence. Since this is a randomized scheme, there are possibilities to miss the communication with the peer node. The possibility of missed communication grows with the number of nodes in the scatternet. Moreover, in PCSS scheme the communication periods are not adjusted to traffic intensity and burstiness.

Locally coordinated scatternet scheduling scheme (LCS) [27]: In this scheme, based on the traffic of the piconet members, the duration and spacing between communication events on the link is adjusted dynamically. LCS requires the piconets to be hierarchically ordered and schedules them according to this order (priority). Since LCS scheme is based on priorities of the neighboring piconets, this scheme can leave gaps between communication events, thereby reducing the overall efficiency of this scheme.

The problem with the rendezvous based inter-piconet scheduling is that the construction and maintenance of scheduling these rendezvous points becomes time consuming. As Bluetooth technology is mainly used for short range ad hoc networks, there will be changes in the topology which is unavoidable and this further increase

the complication in scheduling the rendezvous points. Moreover, as Bluetooth devices are low power, it is not feasible to use such complicated scheduling schemes.

Inter-piconet scheduling without rendezvous points

Mišić et al. [29] proposed the walk-in bridge scheme. This scheme does not require scheduling of rendezvous points and the bridges switch between piconets at will. Therefore, a piconet can have several bridges and a bridge can visit several piconets (to which it belongs) in a sequence. In the walk-in bridge scheduling, the master polls its slaves including the bridges using a chosen intra-polling scheme. If the bridge is present, the exchange between the master and the bridge takes place. Otherwise, the master continues to poll the next slave. The walk-in bridge tells when a communication should start and does not specify when it should end. The simplest solution is that bridge can exist in a piconet in a limited exchange mode or complete exchange mode. In limited exchange mode, the bridge is present only for a specified number of piconet cycles and in complete exchange mode, the bridge is present until all the packets are exchanged.

The walk-in bridge does have certain drawbacks. If the bridge is unavailable during polling, some time slots [2T] (T-Bluetooth Time Slot = 625 microsecs) are wasted which is very small compared to end-to-end delay in the scatternet. The absence of a bridge for many piconet cycles can deteriorate the network performance drastically. In walk-in approach the bridge is not going to be polled immediately by the master after it joins the piconet; rather it will be polled based on the polling scheme. Therefore, the performance of the walk-in scheme is mainly dependent on

the chosen intra-piconet polling scheme. The study of the walk-in bridge scheduling scheme is restricted to data link layer and the effect of it in the transport layer is yet to be analyzed.

2.3.3 Segmentation and Reassembly

SAR in the L2CAP layer makes provisions for large packets (TCP/IP packets) to transmit over the Bluetooth network. In SAR scheme, packets can be chosen either based on channel condition [30] or by using Best Fit (BF) algorithm [5] or by using Optimum Slot Utilization (OSU) technique [5].

The complex computations involved in choosing packets based on channel condition makes it not suitable for Bluetooth devices which are low power, battery operated devices. Whereas, in BF and OSU there is no such complex computations involved. In BF the higher level packets are segmented to baseband packets using best fit strategy whereas OSU reduces the queuing delay of baseband packets by segmenting it to a minimum number. In [5], the authors have shown that OSU outperforms BF in terms of throughput, link utilization and end-to-end delay.

2.3.4 TCP over Bluetooth

Johansson et al. [31] analyzed the performance of TCP/IP over the Bluetooth wireless ad-hoc network. Their model involved only two nodes in a piconet and studied performance using TCP Vegas. The TCP segments and acknowledgements were by default segmented into DH1 packets. Bit error loss was modeled using constant probability. The impact of Scheduling algorithm was not discussed. Moreover, they

assumed that packet loss is due to bit errors that occur only due to interference in the wireless link and did not considered packet loss due to network congestion.

Das et al. [5] proposed two Segmentation and Reassembly (SAR) algorithms: Best Fit (BF), and Optimum Slot Utilization (OSU); along with some Medium Access Control (MAC) scheduling algorithms to enhance the performance of transport layer. They modeled the bit error loss in wireless link by a two state markov chain model and studied the performance using TCP Tahoe, TCP Reno, TCP New Reno and TCP Sack. Similar to the work presented by Johansson et al. in [31], the entire study was restricted to intra-piconet level only. Moreover the analysis by Johansson et al. in [31] and by Das et al. in [5] does not include the L2CAP flow control, possibly because the provision was given only in the recent Bluetooth specification.

In [9], Mišić et al. used the E-limited scheme as a intra-piconet scheduling scheme to test the performance of TCP traffic over Bluetooth network. The interactions that took place between TCP congestion control, L2CAP flow control and scheduling techniques were studied both analytically and through simulations. The TCP segment was segmented into minimum number of baseband packets before passing it to the token buffer. However, the authors assumed a perfect physical layer model with no packet losses due to interference in the wireless link and tested it against TCP Reno. Moreover, their study was restricted to a single piconet.

2.4 Overview of the Thesis

In this research, the work performed by Mišić et al. [9] is extended to a scatternet involving two piconets connected through a SS (Slave/Slave) bridge and studied both

in PPL (Perfect Physical Layer) and IPL (Imperfect Physical Layer). The system performance was studied using TCP new Reno. For bridge scheduling, a walk-in bridge scheme was used. The reason to choose walk-in bridge scheduling and using TCP new Reno is justified in 3.1.1 and 3.1.2 respectively.

Chapter 3

Methods and Materials

3.1 System model

The system was modeled first for a piconet and then for a scatternet (two piconets joined by a bridge). TCP connections created among slaves are as follows. In a piconet for each slave i, for i = 1..n creates a TCP connection with slave j =(i+1) mod n (If j = 0 then slave j will be n) creating n identical TCP connections where "n" is the number of active slaves. In the case of a scatternet(2 piconets connects through a SS bridge) it is P-1-S-n (Piconet one slave n) and P-2-S-n establishing independent TCP connection with each other.

Once the TCP connections are established the application layer in slave devices sends messages of 1460 bytes at a rate λ. The generated messages can be sent in a single TCP segment provided the congestion window is not full. The TCP segment is then passed to the L2CAP layer after passing through the IP layer. The SAR in L2CAP layer is assumed to produce minimum number of baseband packets i.e. in

this case it is 4 DH5 and 1 DH3. It is also assumed that the ACK is sent using an empty TCP segment carrying only the ACK bit, which can be accommodated in a single DH3 packet. The segmented packets are placed in the token buffer S. Figure 1.1(b) [9] shows the L2CAP flow control implemented using a token buffer filter scheme [14]. Token buffer scheme controls the flow rate of the packets into the network by adjusting the token rate *tb*. The segmented baseband packets are placed in the token bucket buffer S . For successful transmission, each packet from the token bucket buffer S has to be associated with a token. If a baseband packet cannot find a token then it has to wait in the token buffer. Since the length of the token bucket buffer is finite, some packets can be rejected. Too low "tb" will increase the number of packets in the the buffer S leading to packet losses and too high "tb" will overload the baseband buffer L. Also the token rate capacity decides the maximum burst to the baseband buffer L. In order to improve the TCP performance proper dimensioning of the token arrival rate "tb", token bucket buffer S, and the token rate capacity is required.

Packets that pass the flow control are stored in a baseband buffer of the baseband layer. Packets which are accepted by the baseband buffer L are then transmitted into the wireless radio channel by piconet scheduler according to the given baseband scheduling policy. When the data is received by the receiver, the received baseband packets are reassembled and an acknowledgment is sent back to the sender. If any of the buffers along the path from source to destination slave are full, or if the wireless link is too noisy, then the TCP segment or acknowledgment can be lost resulting in the degradation of TCP performance.

3.1.1 Choice of intra- and inter-piconet scheduling scheme

For intra-piconet scheduling scheme, E-limited scheduling scheme was used because of its inherent fairness with QoS in spite of being simple to implement. For inter-piconet scheduling scheme, the walk-in bridge scheme was used because of its low computational overhead using the HOLD mode. Since the absence of bridge for many piconet cycles in walk-in can deteriorate the network performance drastically, the bridge will be in a piconet only for one piconet cycle. Within one piconet cycle, all the exchange will be done with the bridge i.e. if Ms is the number of frames to be transferred with a slave in E-limited scheduling scheme then number of frames that exchanged with the bridge Mb will be n*Ms where "n" is the number of active slaves in that piconet.

3.1.2 Choice of TCP

The analysis was based on TCP new Reno [11] as it is the latest version available in the literature. TCP new Reno gives better performance compared to RENO when multiple packets are lost from a single window of data, which are of specific interest to Bluetooth devices with small buffers. TCP new Reno was build over the artifex version of TCP. Chapter 4 explains in detail how TCP new Reno was build. TCP parameters are tuned as shown in Table 3.1.

3.1.3 Physical layer model using a two state Markov chain Model

The modeling of the Imperfect Physical Layer [IPL] was performed using a two state Markov chain Model as in [5]. Such a model was enough to analyze the behavior of TCP when packets are subject to bursty packet losses. In this thesis, modeling of the physical layer using a two state markov chain model was considered for each master-slave connection as wireless channels are distinct and time varying for each user. Figure 3.1 shows the transition structure of the Markov Loss Model that is been considered in my thesis. The channel was modeled with two states, namely, Good and Bad. In Good state the BER (Bit Error Rate) is P_G and in Bad state it is P_B with $P_G << P_B$ with these BER depending on the characteristics of the propagation environment and the transition modulation scheme. In this thesis, it was assumed that time spent in each state is exponentially distributed with different mean values. Therefore, the average time between state transitions can be expressed as $\rho_G = 1/\mu_G$ and $\rho_B = 1/\mu_B$. The parameter values are $P_G = 6.8979 * 10^{-5}$,$P_B = 1.263 * 10^{-3}$, $\rho_G = 437.5ms$, $\rho_B = 55.8ms$ taken from [5].

3.2 Approach Adopted

Mathematical analysis of the performance of TCP traffic on a Scatternet level is too complex and hence simulation method was adopted. However, both analytical modeling and simulation of TCP traffic in a single piconet with Reno, L2CAP flow control and E-limited scheduling over a PPL is available in [9]. Simulations were

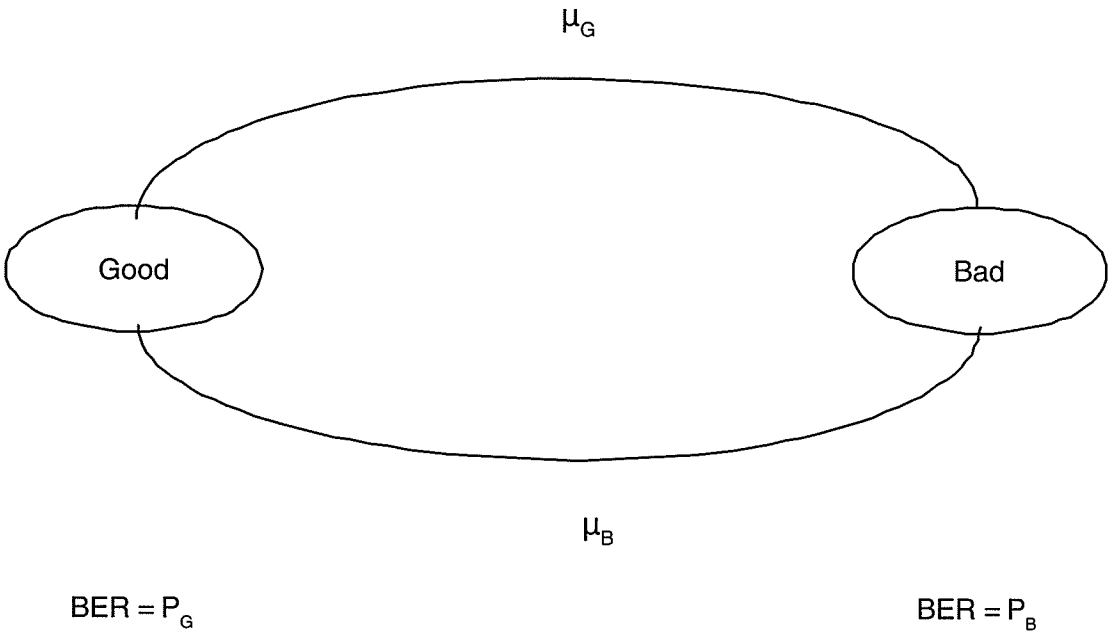


Figure 3.1: Transition structure of the Markov Loss Model adopted from [5]

conducted using Artifex simulation software [12].

3.2.1 Simulation Technique adopted

The simulations were run for 8 minutes in order to get stable results. Moreover, a warm-up period of 1 minute was included for the system to get stable and during this period no values are recorded. For consistency, each simulation was repeated ten times by changing the seed and the average was taken.

3.2.2 Performance metrics

The performance metrics TCP goodput, round trip time (RTT), congestion window, number of timeouts and fast retransmits/sec were studied in this research. TCP

goodput is an indication of the amount of data received by the user per second. RTT is time taken in transmitting a TCP segment from the sender to the receiver and getting an acknowledgement back to the sender and is calculated using RTT time stamp algorithm. Congestion window (cwnd) is a state variable that limits the amount of data TCP can send. At any instant of time the amount of unacknowledged data cannot exceed the minimum of cwnd and rwnd (the most recent advertised receiver window). It is this congestion window variable that is changed during slow start, congestion avoidance, fast retransmit and recovery stage. Congestion window is calculated during the moments of acknowledgement arrivals and packet loss events. Timeouts and fast retransmits are an indication of congestion and error rate (wireless medium) in the network.

3.2.3 Assumptions and Design Considerations

Listed below are the assumptions and design considerations made in this research

- The Bridge device acts only as a slave in all the piconets in which it is a member.
- The Master and bridge device in a scatternet do not produce any traffic. They are only used to route the traffic (i.e. there is no TCP connections starting and ending at the master and the bridge) which makes the analysis simpler.
- The master buffer size is larger than the slave and bridge buffer, and hence the packet losses in it can be ignored. Such an assumption will prevent devices like wireless headsets, mouse, etc., with smaller buffer sizes acting as master which will result in poor performance of the network

- The segmentation algorithm in L2CAP always produces minimum number of baseband packets (i.e. 4 DH5 and 1 DH3 packets) and for acknowledgement one DH3 packet was used as in OSU. In [5]the authors have shown that OSU outperforms BF in terms of throughput, link utilization and end-to-end delay.
- Each application in the slave device produces identical TCP traffic even though this assumption does not reflects the real time scenario, which is too complex to analyze.
- The application unit in slave devices generates segments of 1460 bytes at a rate λ (poisson arrival)
- The bridge stays in each piconet only for one piconet cycle before switching to other (bridge staying for too many piconet cycles might increase the end-to-end delays thereby degrading the TCP performance)
- Polling mode for bridge = n*M = bridge buffer size where "n" is the number of slaves in a piconet and "M" is the scheduling parameter in E-limited scheduling scheme. (To transmit all the packets during a single visit by the master and to prevent packet loss in the bridge)
- There is no ARQ scheme at the link level and the packet loss detection and re-transmission is taken care of by TCP.

3.2.4 Code Validation

Before carrying out with the simulation and analyzing the results, code validation is performed to show that the implementation is correct. Code Validation is explained

in detail in Chapter 5

3.2.5 Analysis strategy

The performance metrics was calculated and analyzed by varying the variable parameters (shown in table 3.1). The results were depicted using 3D graphs and the analysis was performed in three phases. In this research, optimal token buffer size, scheduling parameter M, and token rate *tb* are found. By optimal I mean that, to what values these parameters needs to be tuned to achieve maximum TCP goodput. To achieve this goal, the variable parameters were varied as follows:

- Initially optimal token buffer size S was found by varying the offered load against token buffer S fixing *tb* (matching to the system capacity) and M = 5 (ability to transmit one TCP segment without interleaving).
- Next, the optimal scheduling parameter was found by varying the offered load against scheduling parameter M from 5 to 25 [*tb* and S (found before) fixed]
- Having found the optimal M and S, optimal *tb* was found by varying the offered load against *tb* [S and M fixed at optimal values]

Phase I and II was carried out in PPL. Conducting the experiments in the PPL, the packet loss due to congestion alone was focused. In phase I, the simulation was performed on a single piconet. Initially, the experiment was carried out with just two active slaves establishing independent TCP connection with each other and later was increased to four and six active slaves. Phase II was performed in a scatternet with two piconets and one bridge. Initially, there was only one active slave in each piconet

Table 3.1: Model parameters

Fixed parameters
TCP parameters MSS (Maximum Segment Size) = 1460 RcvBuffer (Receiver Buffer) = 8MSS Threshold = 64MSS Bluetooth Parameters Token Buffer Capacity = 3KB Slave Buffer Size L = 20 [large enough so that the research can be focused on flow control] $P_G = 6.8979 * 10^{-5}$,$P_B = 1.263 * 10^{-3}$, $\rho_G = 437.5ms$, $\rho_B = 55.8ms$ taken from [5] Master buffer size [large enough so that the effect of it is ignored] Scheduling Parameter for bridge = n*M = Bridge buffer size [To transmit all the packets to the bridge during a single visit by the master and to prevent packet loss at the bridge buffer] where "n" is the number of active slaves in a piconet.
Variable Parameters
Token buffer S Token rate *tb* (kbps) Slave offered load (kbps) Scheduling parameter M in E-limited scheduling scheme

establishing independent TCP connection with each other. Later the experiment were repeated with two, four and six active slaves in each piconet.

In phase III, the IPL was included for simulation in addition to first two phases. The analysis of phase III with IPL made this research more realistic where packet losses take place not only due to over flow of buffers but also due to the interference in the wireless transmission media.

Finally, the results of phase III were compared to phase I and II. Such a comparison helped in studying the performance of TCP congestion control mechanisms in the wireless networks which were initially designed for wired networks.

3.3 Materials Used

Object-oriented Petri-Net-based simulation engine, Artifex, by Artis Software Inc. [12] was used for simulation.

The salient features of Artifex simulation software are

- The dynamic behavior of the model can be viewed by using its intuitive graphical language feature.
- Advanced networking toolkit (with drag and drop features) encompassing the TCP/IP protocol provided by Artifex was readily used in my research model. TCP new Reno was build over it. Chapter 4 simulator design explains in detail what artifex provided and how new Reno was build over it.

In this research, the code of slave, master, and bridge designed by K. L. Chan [29] using Artifex simulation software was used as a base. Highlighted below are the list of modifications been done over it and in chapter 4 they are explained in detail.

Master

- Implementation of wireless medium

Slave

- Implementation of Application layer,

- Modification of Artifex TCP layer to new Reno standards, single timer implementation based on [15] (Chapter 4 explains in detail these modifications),
- Implementation of Segmentation and Reassembly unit,
- Implementation of wireless medium,
- Implementation of flow control using token buffer scheme and
- Implementation of finite buffer

Bridge

- Implementation of wireless medium, and
- Implementation of finite buffer

Chapter 4

Simulator Design

My research model was build using Artifex Simulation Software [12]. Artifex Petri nets provide an intuitive graphical language to build a model. Objects are defined in Artifex using elements like transition, place, link etc. The user writes the code in the transitions (processing units). Transitions are denoted by rectangles. Places denoted by circles acts like buffer where it stores tokens (communication unit). The transitions are connected to the places and are triggered when the tokens are received in places. Links connect the transition with places. Objects interact with each other through interfaces (set of input and output places). The output of one object is connected to the input of an other object. Further details about the operation of Petri nets can be found in Artifex manuals.

4.1 Scatternet

The designed scatternet consists of two piconets ($P1 : PICONET and P2 : PICONET$) connected through a SS bridge ($B1 : SS\_BRIDGE$). Figure 4.1 shows the developed scatternet model. PICONET and BRIDGE class implements the piconet and bridge functionalities (explained in subsequent sections). START place has one token by default to trigger the simulation. In the initial actions of the code, the simulation time is given. When the simulation time ends the packet HALT transition is triggered. The $xx\_halt()$ function present in the action of HALT transition ends the simulation.

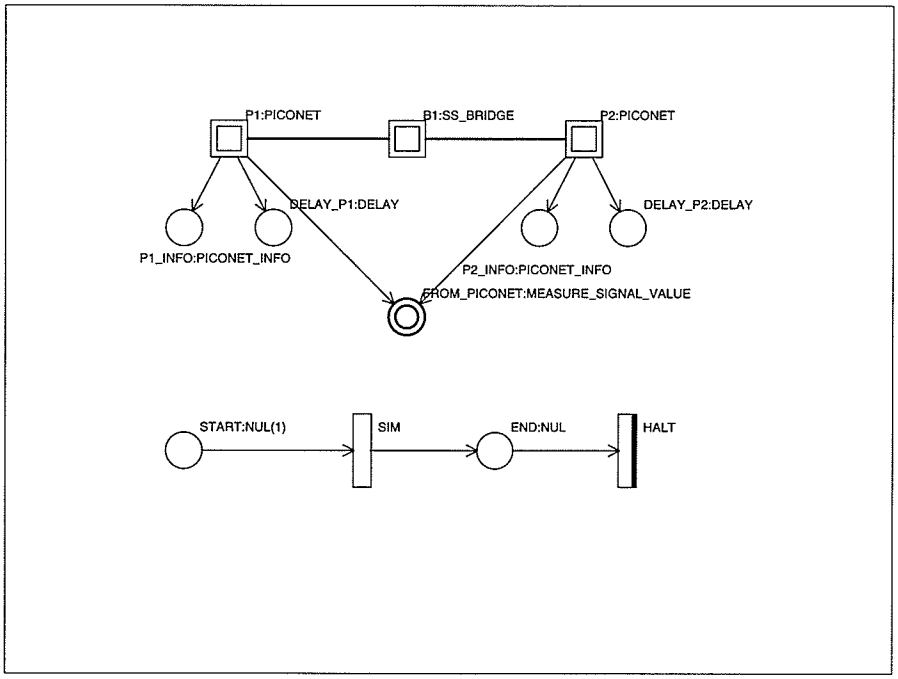


Figure 4.1: Scatternet

Information is exchanged between the piconet master and the bridge in the form of tokens by establishing a link set between them. The metrics: TCP goodput (kbps),

RTT, congestion window size, fast retransmits/sec and timeouts/sec are collected from both the piconet and the average is taken.

The number of slaves active in a piconet, number of piconets present, simulation time, warm-up time etc., in set only in the top level scatternet parameters. Required parameters are passed to other sub classes like piconet, bridge through parameter passing in initialization phase. The parameters are further passed to slave and Master through the piconet using the same parameter passing technique. This method of parameter passing helps in setting the parameters of the simulation easily during simulation in batch runs.

4.1.1 Piconet

Piconet class consists of one master and up to seven active slaves. Figure 4.2 shows this piconet model. In the case of a scatternet (2 piconets are present with bridge enabled), the maximum number of active slaves can be only 6 and the bridge takes the slave id n+1 where "n" is the number of active slaves. The master and slave devices are connected through link sets for data exchange. The metrics from individual slaves are collected and passed to the scatternet where the average is found.

Also present in the piconet class is the fading channel (channel that maintains the state of the wireless medium). Figure 4.3 shows the implementation of fading channel in the piconet. Each master slave channel modeled has independently exponentially distributed good and bad durations. This state variable is later used by master, bridge, and slave devices in detecting corrupted packets (described in section 4.2).

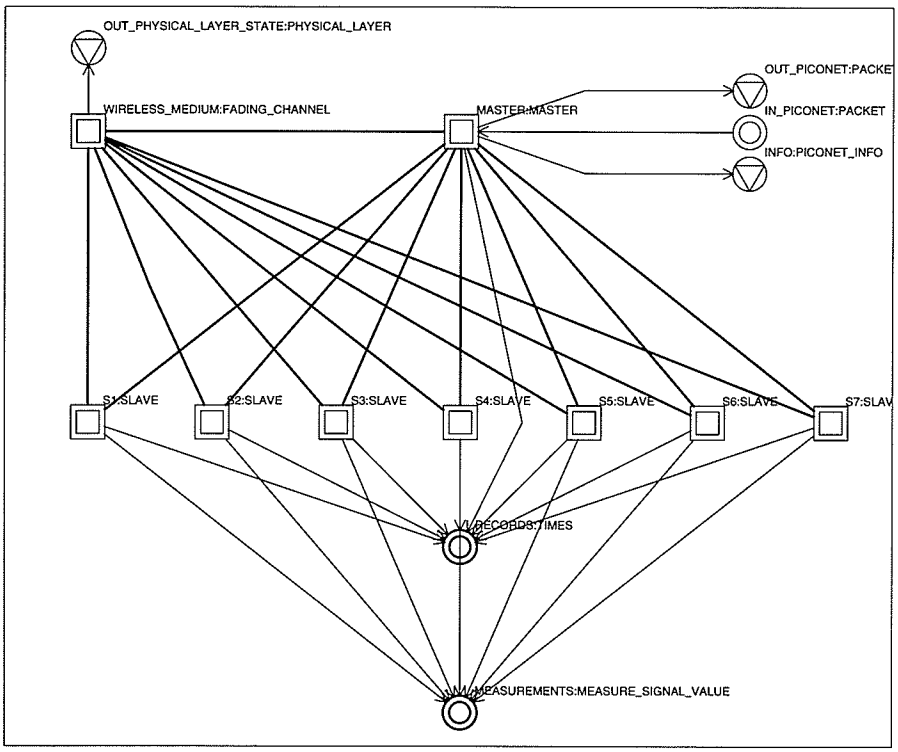


Figure 4.2: Piconet

Master

The baseband packets that reach the master are investigated for an erroneous packet (described in section 4.2). Figure 4.4 shows the implementation of this packet loss investigation. The packets after investigation are separated based on local (intra-piconet) or non local (inter-piconet) data and passed to the respective downlink queues. Figure 4.5 shows the master downlink queues. Then the master removes packets from these queues and transmits into the wireless medium based on E-limited scheduling policy. Figure 4.6 shows the E-limited scheme implemented.

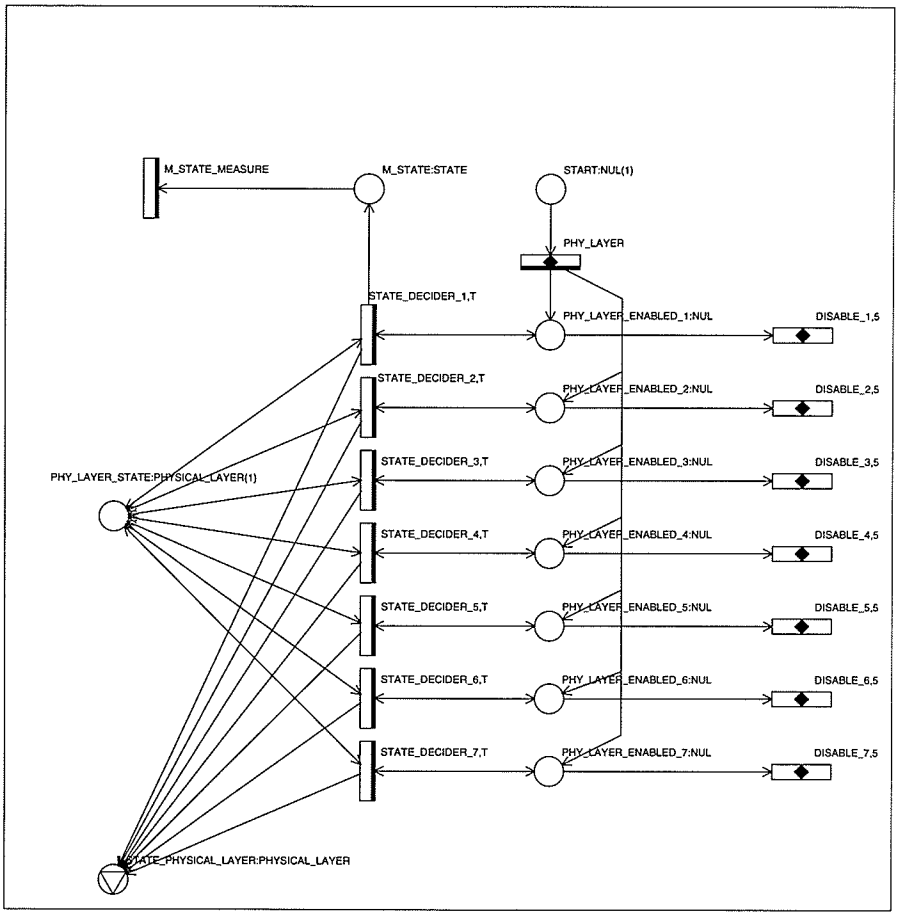


Figure 4.3: Fading Channel

Slave

Slave devices are the ones that establish independent TCP connections with other slave devices and start data exchange between them.

Application layer

The application layer is the one that generates traffic to other slave devices. In this model, based on the number of piconets, TCP connections are initiated by the application layer of the slave devices. When there is only one piconet, independent

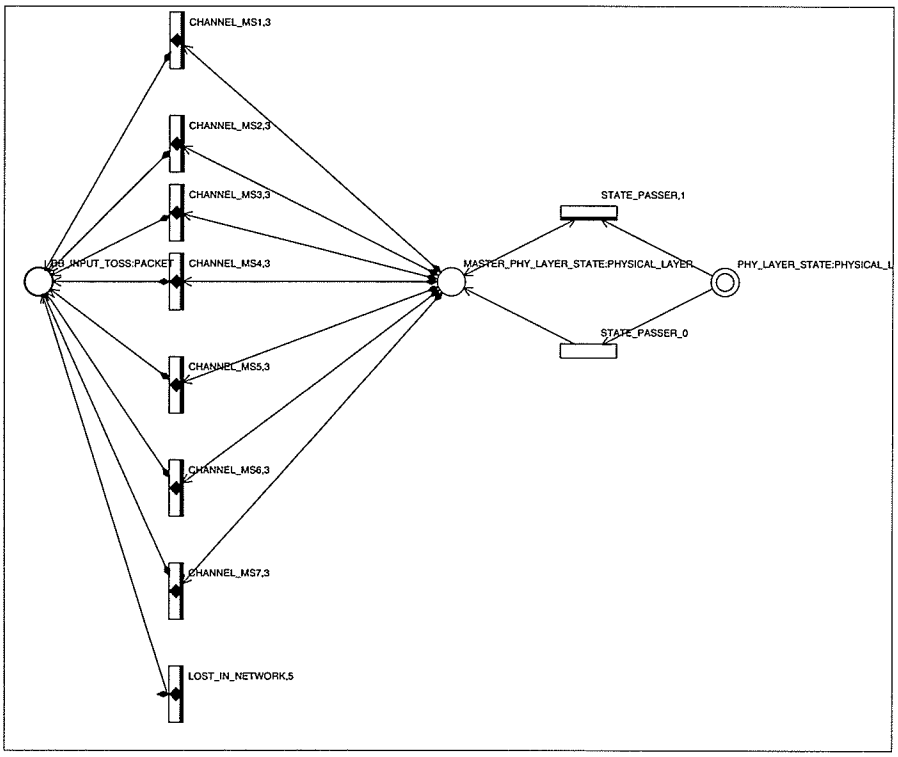


Figure 4.4: MASTER: Physical layer

TCP connections are created as follows: for each slave i, for i = 1..n creates a TCP connection with slave j = (i+1) mod n (If j = 0 then slave j will be n) creating n identical TCP connections where "n" is the number of active slaves. For only one piconet the bridge is disabled. In the case of a scatternet it is P-1-S-n (Piconet one slave n) and P-2-S-n establishing independent TCP connection with each other. Figure 4.7 shows the application layer implementation

The application layer generates segments of 1460 bytes at a rate λ and sends it to the TCP/IP layer for data transmission and the packets that are received by the TCP/IP layer are read back by the application layer. The application layer is designed so fast that it instantly retrieves the messages from the TCP buffer.

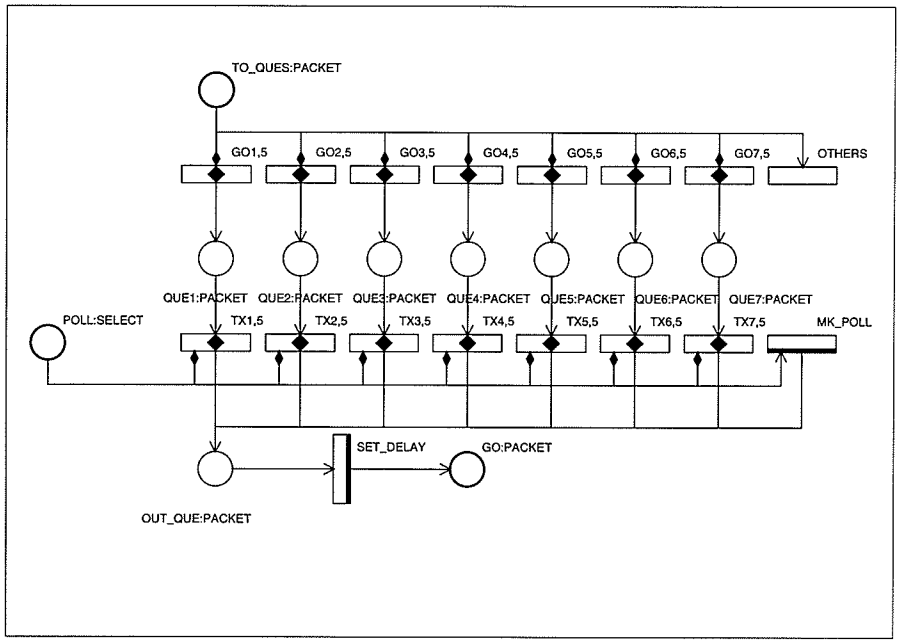


Figure 4.5: MASTER: Downlink Queues

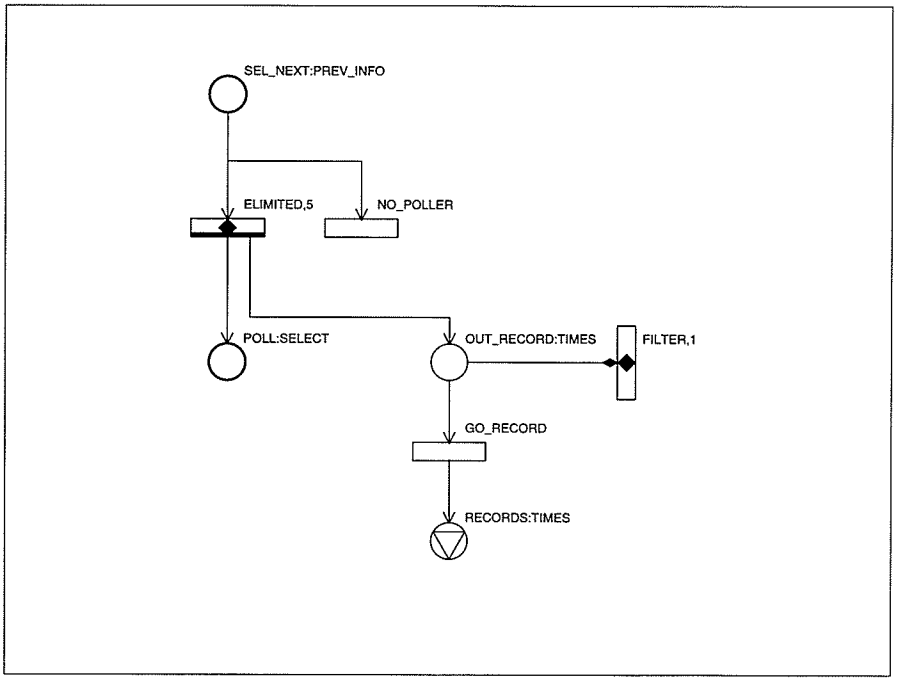


Figure 4.6: MASTER: Polling

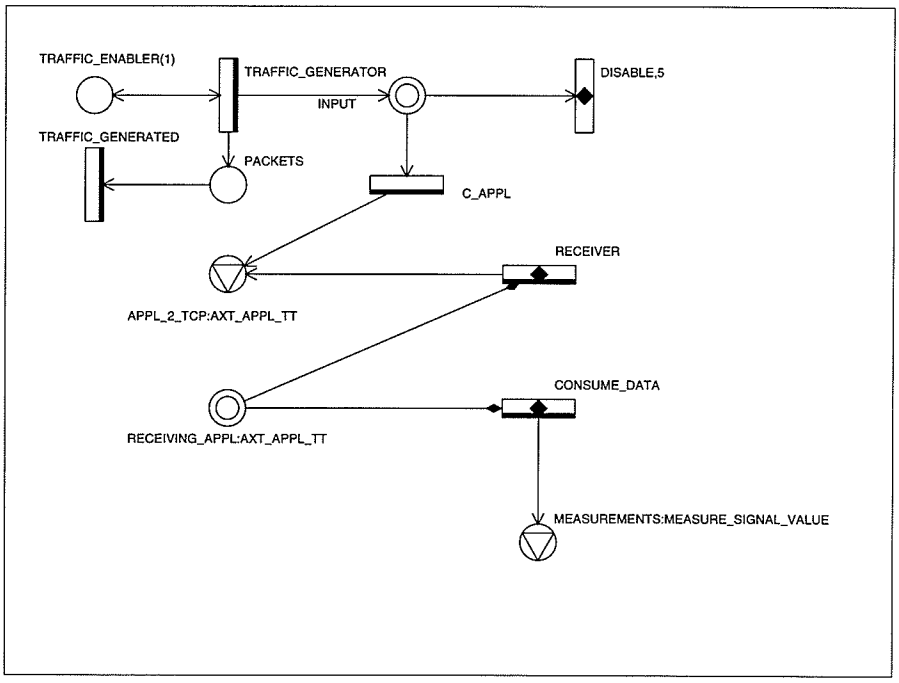


Figure 4.7: Application Layer

TCP/IP layer

Artifex version of TCP based on [32; 33] was used in my simulation. Artifex TCP model implements Jacabson slow start, Congestion Avoidance, and Retransmission Timeout (RTO) estimation algorithm which was described in [34]. Timer implemented in Artifex TCP model is multiple timers and the only means of packet loss detection is through a timeout mechanism, which is a very costly operation. Moreover, multiple timers are great in theory, but in reality require considerable overhead [14].

There were also some bugs noticed in Artifex TCP model. The major error was during the slow start and congestion avoidance phase. In the Artifex version of TCP, in the slow start phase for each acknowledgement received, the sender doubles the congestion window and in Congestion avoidance it adds one MSS (Maximum Segment

Size) to the congestion window. Therefore, the slow start and congestion avoidance phase of Artifex TCP model is equal double the slow start and slow start respectively. This completely overrules slow start and congestion avoidance phase in TCP.

The existing version of Artifex TCP model was modified to new Reno. A single timer was implemented based on [15]. The single timer works as follows:

1. When a data segment is transmitted, start the timer with the current RTO (Retransmission Timeout) if the timer is not already running.

2. On receiving a new data ACK, check whether it is the ACK for the last segment sent. If so stop the timer. Else restart the timer with the current RTO.

3. On a timeout, re-transmit the data segment and start the retransmission timer with twice the current RTO (maximum is 64 sec).

Figure 4.8 shows the acknowledgment management page and Figure 4.9 shows the retransmission page in which new Reno code was incorporated.

Segmentation and Reassembly

The packets that pass the IP layer are sent to the L2CAP layer where the TCP segment is segmented into 4 DH5 and 1 DH3 packets and passed to the token buffer for flow control. The packets that pass the flow control are placed in the baseband buffer after which it is transmitted to the wireless medium when master polls it. Figure 4.10 shows the segmentation unit along with the application layer and TCP layer embedded in it while Figure 4.11 shows the uplink channel showing the token buffer scheme and data transmission in slave.

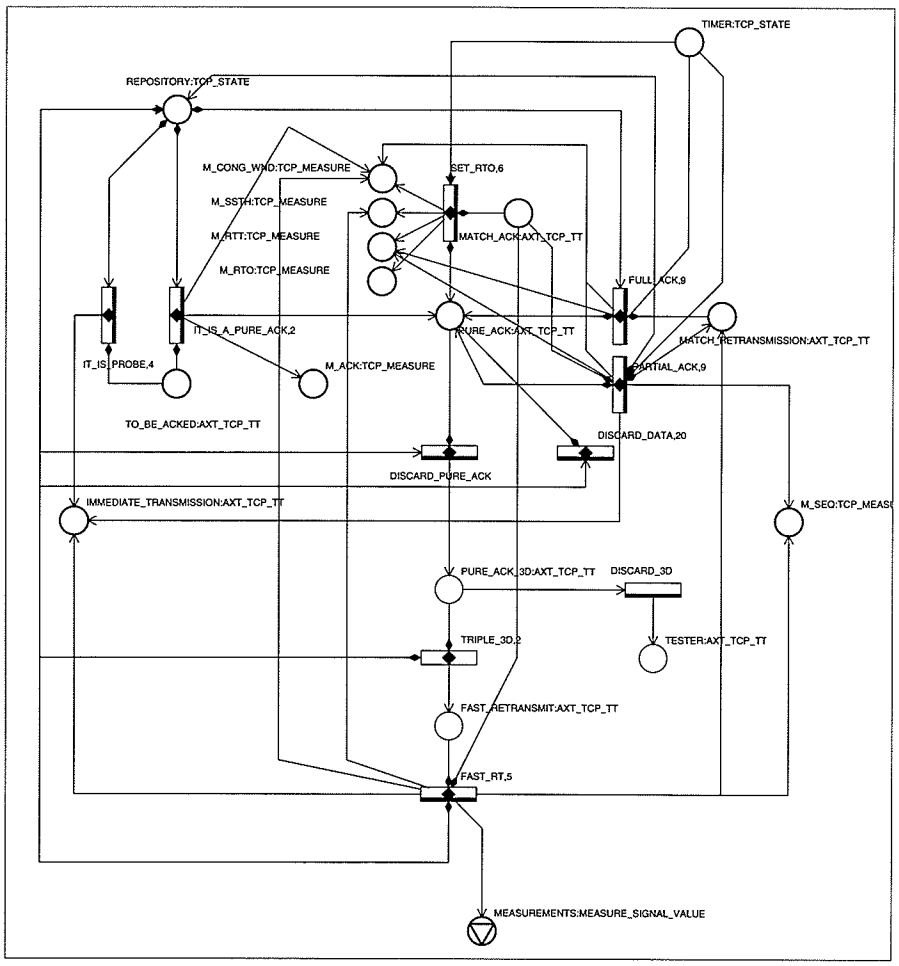


Figure 4.8: TCP: ACK management

On reception, the packets are investigated for error detection (described in 4.2). Figure 4.12 shows the error detection mechanism in the slave. The packets that pass the error recovery mechanism are reassembled. Figure 4.13 shows the reassembly unit. Once the packets are reassembled, it is send to the TCP layer.

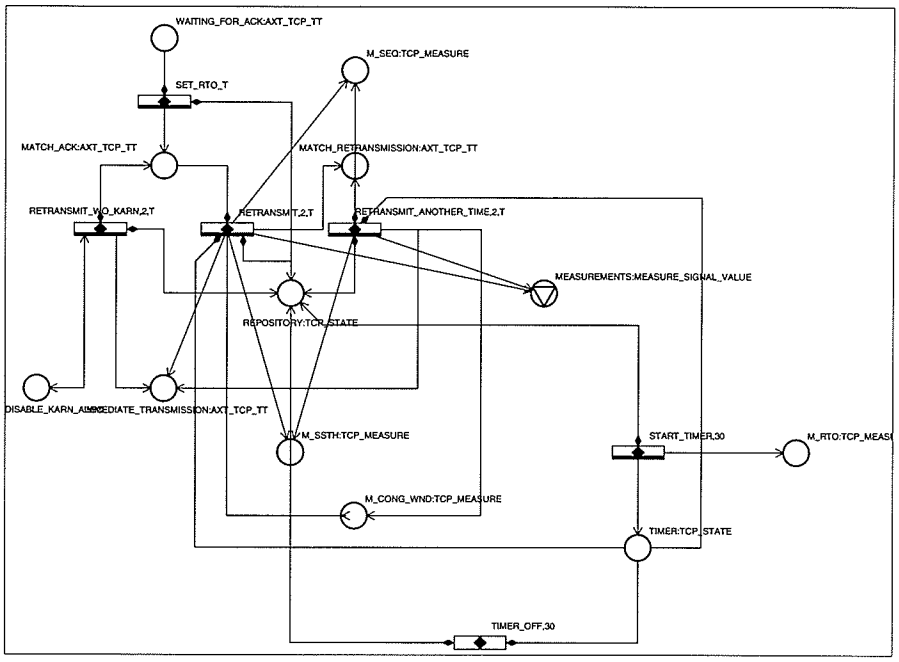


Figure 4.9: TCP: Retransmission

4.1.2 Bridge

The bridge implemented in this research is the SS Bridge. The SS Bridge is designed according to walk-in bridge scheduling where the bridge stays in one piconet for one piconet cycle before switching to the other. Figure 4.14 shows the SS bridge implementation. The master polls the bridge according to E-limited scheduling policy with a POLL packet and if the bridge is present then it sends an NULL packet after which master and bridge exchange data. If the bridge is not present it sends NOT-LISTENING signal to the master so that master keeps polling the next slave.

The packets received in bridge are investigated for errors (described in 4.2) as performed in master and slave device. Figure 4.15 shows the error detection implemented in bridge.

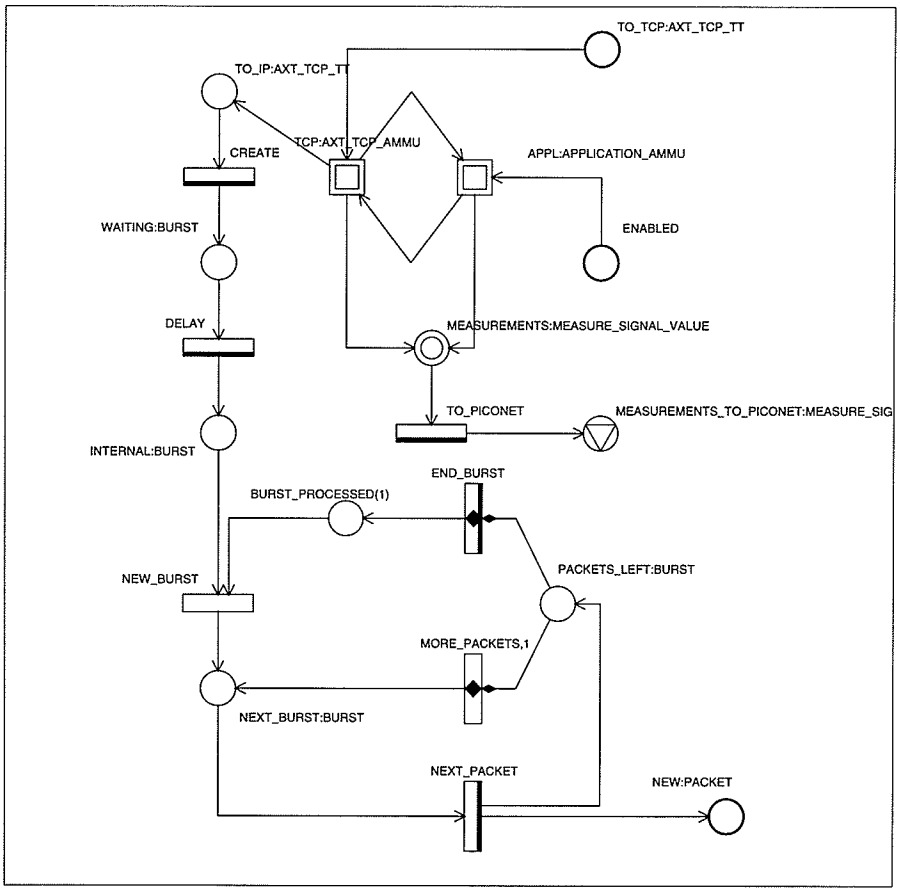


Figure 4.10: SLAVE: TCP/IP packets segmented

4.2 Method adopted in simulation to find whether a packet is successfully transmitted in the wireless medium

When the packets are transferred from slave-1 to slave-2 via master in a piconet, wireless packet losses can take place in two places, in the master and in slave-2 as each slave master channel is independently modeled. When the baseband packet reaches

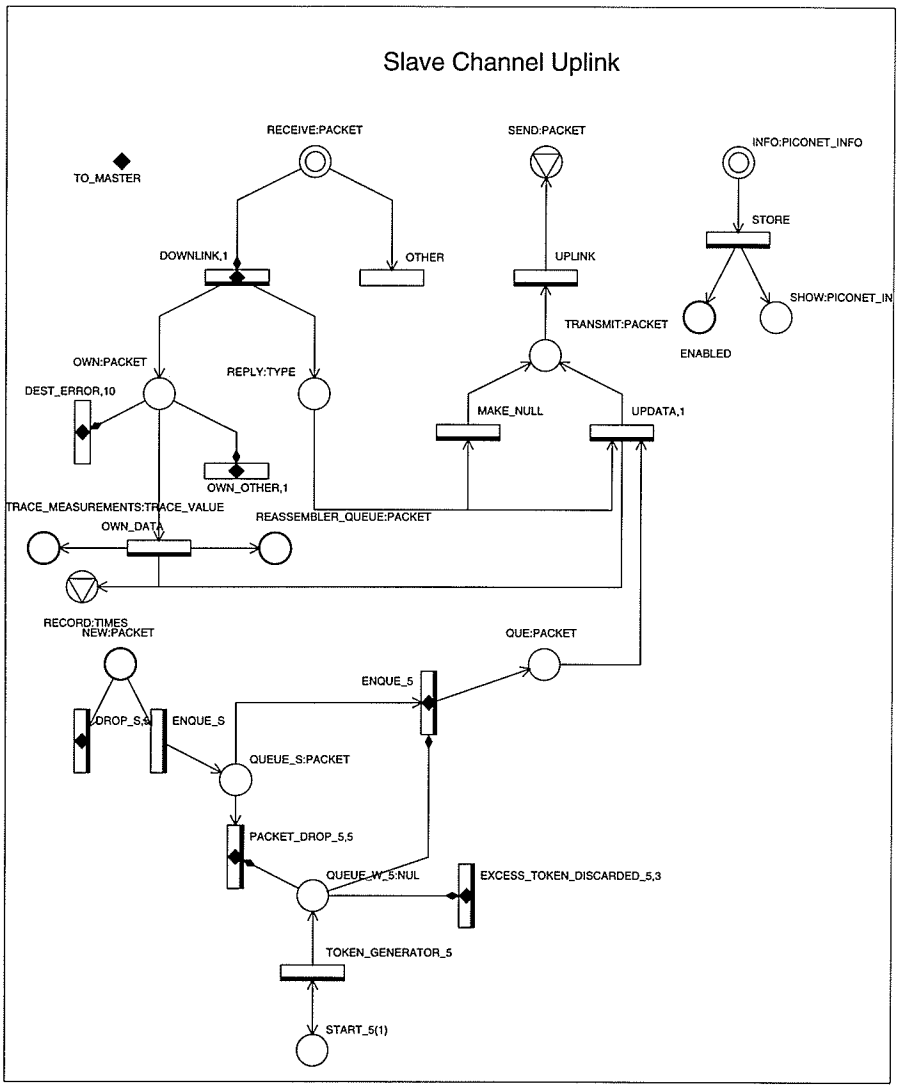


Figure 4.11: SLAVE: L2CAP flow control and packet reception/transmission

the master, the master checks the state (good/bad) of slave-1 and master. Based on the state, it calculates the packet error rate (DH5/DH3) from the BER of the channel (calculation of the PER from BER is explained in 4.2.1) and checks whether the packet is damaged by comparing it with the random double number generated. If the random double number is less than the PER then the packet is considered to be

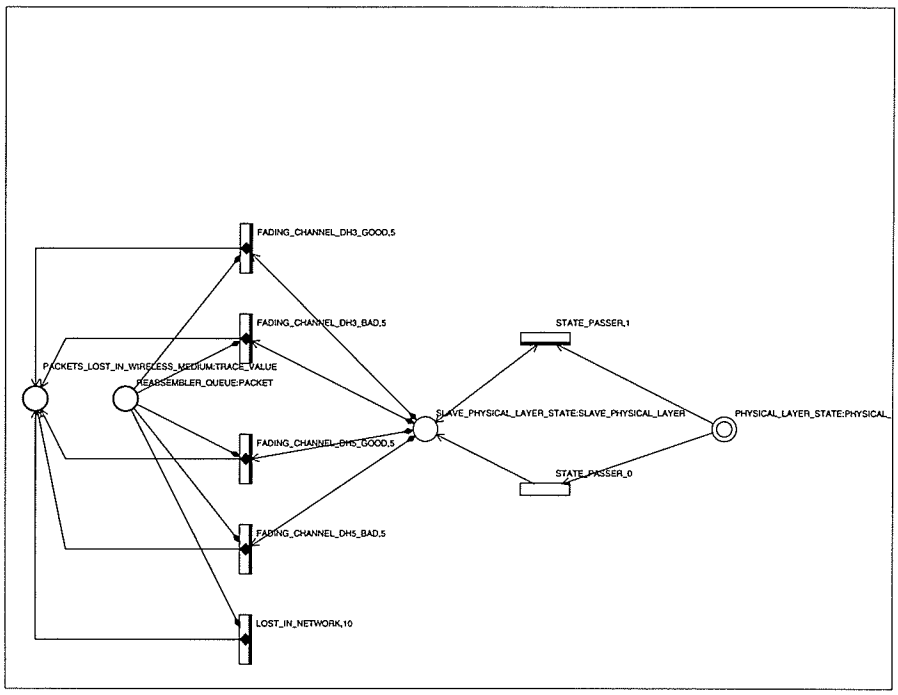


Figure 4.12: SLAVE: Physical layer

lost. The master continues to transmit the segment to the slave-2 with signal lost in it. The packet is sent to slave-2 irrespective of whether the packet is lost or not due to interference. If the packet is not lost in the master, then slave-2 checks the state (good/bad) of slave-2 and master and calculates the PER accordingly and decides whether the packet is lost or not as performed by the master

4.2.1 Calculation of PER from BER

Assuming BER as "b" and packet error rate as "p" and "L" be the packet size in bits, PER p is given as "$p = 1 - (1 - b)^L$" and Table 4.1 shows the calculated PER of various packets under consideration.

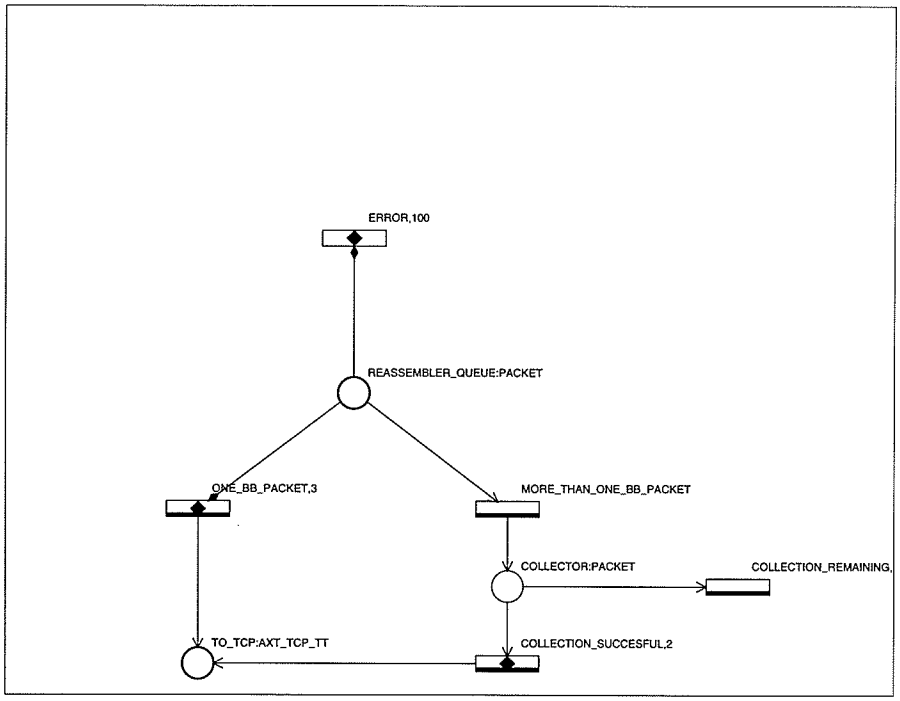


Figure 4.13: SLAVE: Reassemble unit

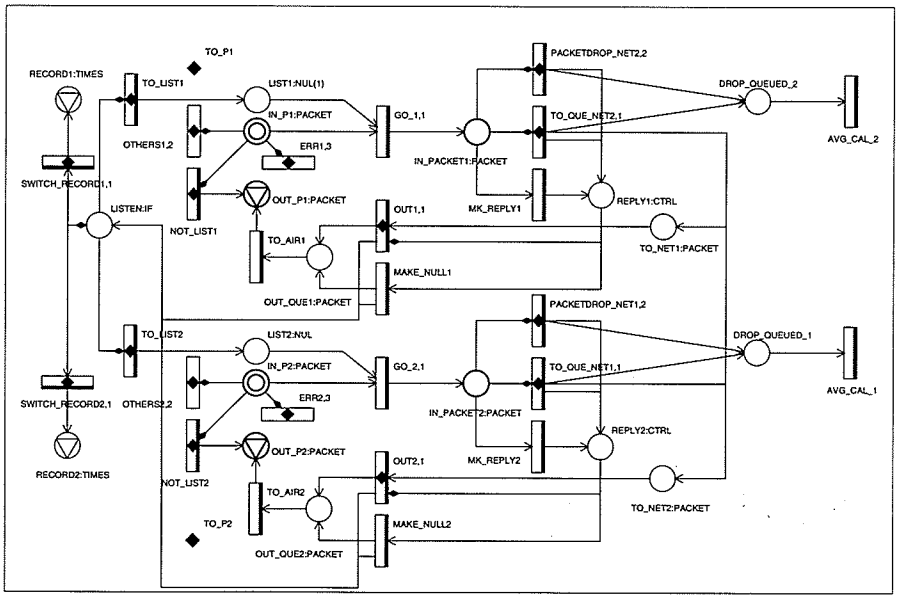


Figure 4.14: Bridge

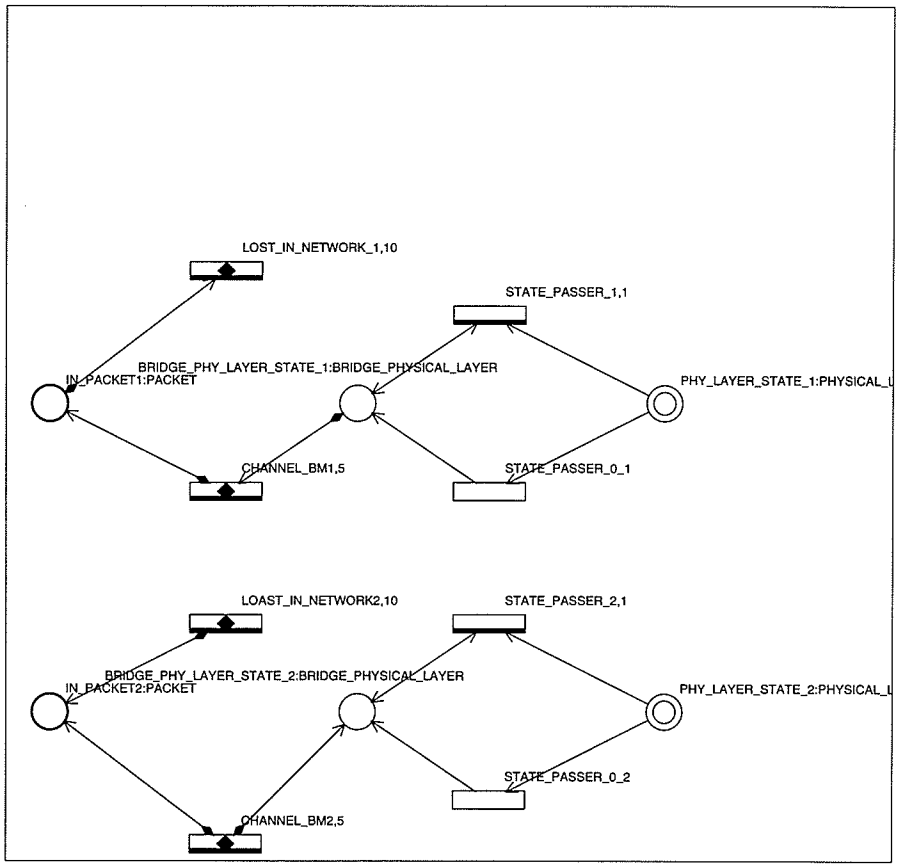


Figure 4.15: BRIDGE: Physical layer

Table 4.1: PER of various ACL packets under consideration with $P_G = 6.8979 * 10^{-5}$,$P_B = 1.263 * 10^{-3}$

Packet Type	Packet Size in bits	Packet Length (Bluetooth slots)	State	PER
DH5	2712 (339 bytes)	5	Good	0.1702
			Bad	0.96753
DH3	1464 (183 bytes)	3	Good	0.09581
			Bad	0.84279

Chapter 5

Code Validation

5.1 Validation of good and bad period in simulations

Since the times spent in good and bad state are exponentially distributed it was important to show that average good and bad duration obtained from simulation matches with the input ρ_G, and ρ_B and their durations were exponentially distributed through trace analysis. The good and bad durations were generated using $xu\_RndNExp()$ function provided in Artifex. Shown in Table 5.1, are the results of average good and bad state duration obtained from simulation. The average good and bad durations obtained for slave1-master and slave2-master channel was almost identical to the input ρ_G and ρ_B.

Table 5.1: Average good and bad state duration (in sec) taken from simulation

Time spent in each state (Exponentially distributed)		From simulation			
		Slave1-Master Channel		Slave2-Master Channel	
Average Good duration (ρ_G)	Average Bad duration (ρ_B)	Average good duration	Average bad duration	Average good duration	Average bad duration
0.4375	0.0558	0.43867	0.056	0.43912	0.05566
0.4475	0.0458	0.44779	0.04577	0.44966	0.04579
0.4575	0.0358	0.45752	0.03583	0.4606	0.0357
0.4675	0.0258	0.46422	0.02581	0.46608	0.02581
0.4775	0.0158	0.47745	0.01578	0.47752	0.01591
0.4875	0.0058	0.48869	0.00584	0.48898	0.00576
0.4923	0.001	0.49351	0.001	0.4921	0.001

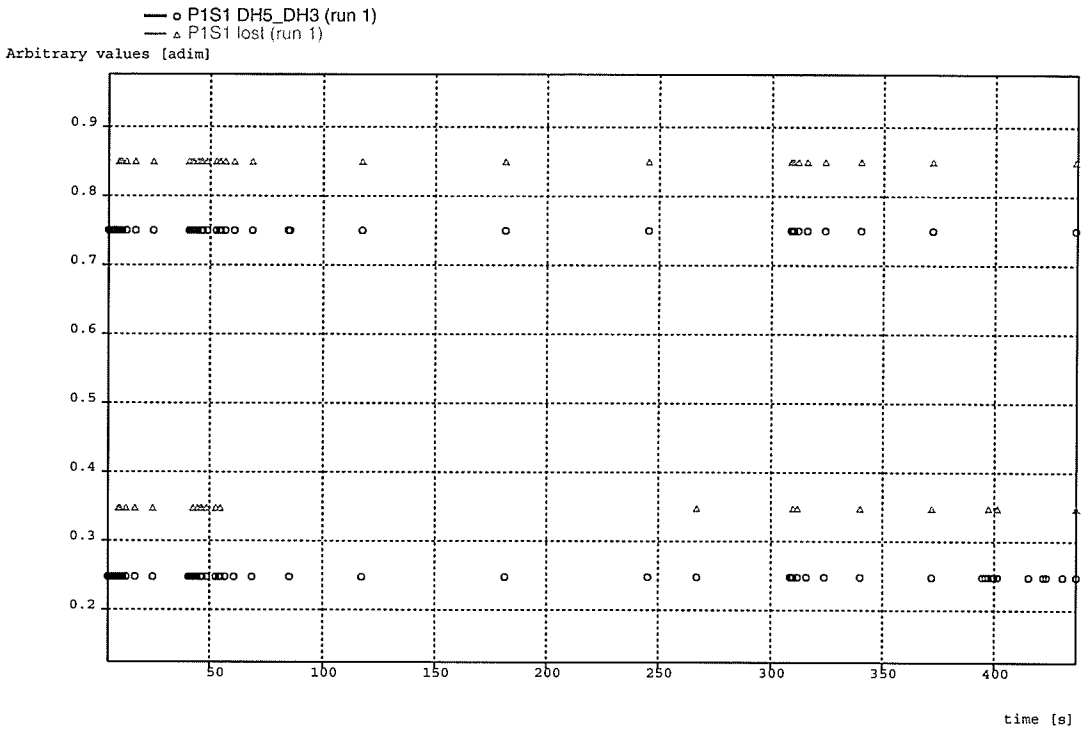
5.1.1 Trace analysis

The goal of trace analysis is to show that packets do get lost in error-prone environment and also to show that the duration of good and bad states are exponentially distributed. Figure 5.1 and figure 5.2 shows the traces obtained when two slaves in a piconet having independent TCP connection with each other using the following parameters.

Offered load = 80 kbps, token rate tb = 250 kbps, Scheduling parameter M = 5, Queue S = 25, buffer L = 20, good duration 437.5 ms and bad duration 55.8 ms

Figure 5.1 trace shows the baseband packets that are received in the Slave-1 along with the packets that are lost after analysis. It can be inferred that some packets (of both DH5 & DH3 type) get lost in error-prone environment after traveling 2 hops.

Figure 5.2 trace shows the baseband packets that are received in Slave-1 along with the states (good/bad). The Figure 5.2 also shows that good and bad state durations are exponentially distributed.



.85 -> received DH5 is lost (after analyzing).

.75 -> received DH5

.35 -> received DH3 is lost (after analyzing).

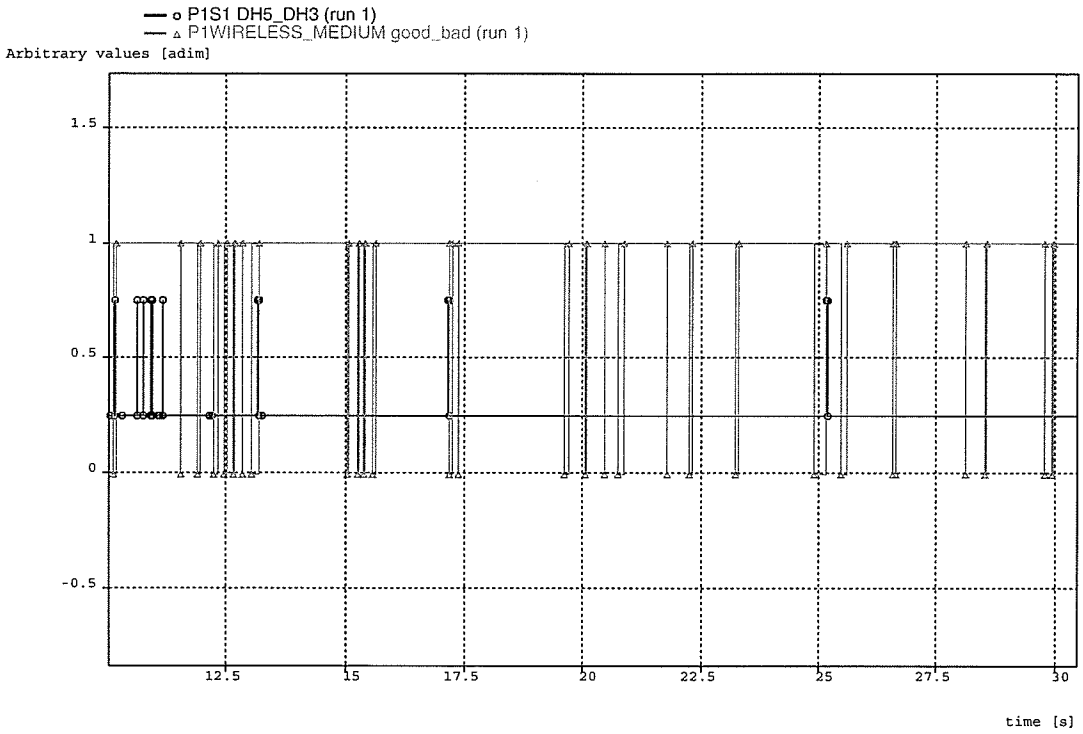
.25 ->received

Offered load = 80kbps, tb = 250kbps, M = 5, S = 25, L = 20, good duration 437.5ms and bad duration 55.8ms

Figure 5.1: [Piconet with two slaves] Trace of DH5 and DH3 packets received in Slave-1 along with the lost ones

5.2 Sensitivity Analysis

Having validated the good and bad durations, the next job was to analyze how the good and bad states duration affects the TCP performance in a piconet with two



.75 -> received DH5 .

.25 -> received DH3

1 -> good state

0 ->bad state

Offered load = 80kbps, tb = 250kbps, M = 5, S = 25, L = 20, good duration 437.5ms and bad duration 55.8ms

Figure 5.2: [Piconet with two slaves] Trace of DH5 and DH3 packets received in Slave-1 along with the state of the channel (good or bad)

slaves having independent TCP connection with each other.

The offered load was varied from 80-200 kbps with token rate tb = 250 kbps, scheduling parameter M = 5, and Queue S = 25 fixed along with buffer L = 20 and Receiver buffer size = 8 MSS. The good and bad durations were varied by increasing

the good duration state and decreasing the bad duration state as shown in Table 5.1. The resulting graphs, namely, average TCP goodput, Average congestion window size, average number of fast retransmits and timeouts are shown below in Figure 5.3 & Figure 5.4.

The TCP goodput performance increases with the increase in good period state. This is obvious as wireless channel is in good state for more time, the packet losses decreases. Congestion window size also increases with increase in good state duration state.

However, the number of timeouts was increasing with increase in good state duration (explained in detail in theoretical analysis which is to follow shortly). The number of fast retransmits was zero irrespective of increase in good state period because the average congestion window is never more than 2 MSS. This indicates that triple duplicate acknowledgements were certainly not possible with the given parameters.

5.3 Theoretical Analysis

Assuming the simulation is running for a long time the success rate formula for 1 hop becomes

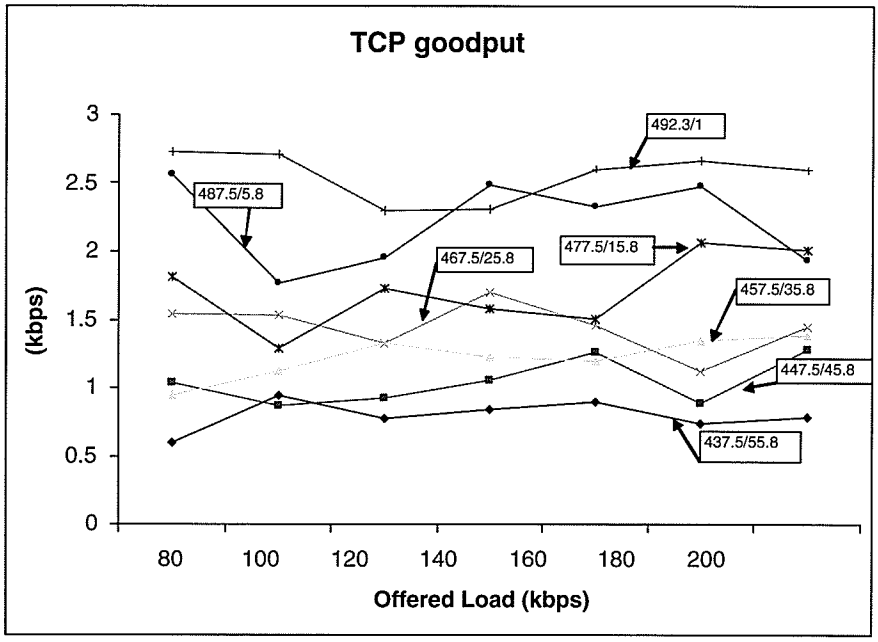
Success rate of a packet for 1 hop = $[1 - PER_G]P_g + [1 - PER_B]P_b$

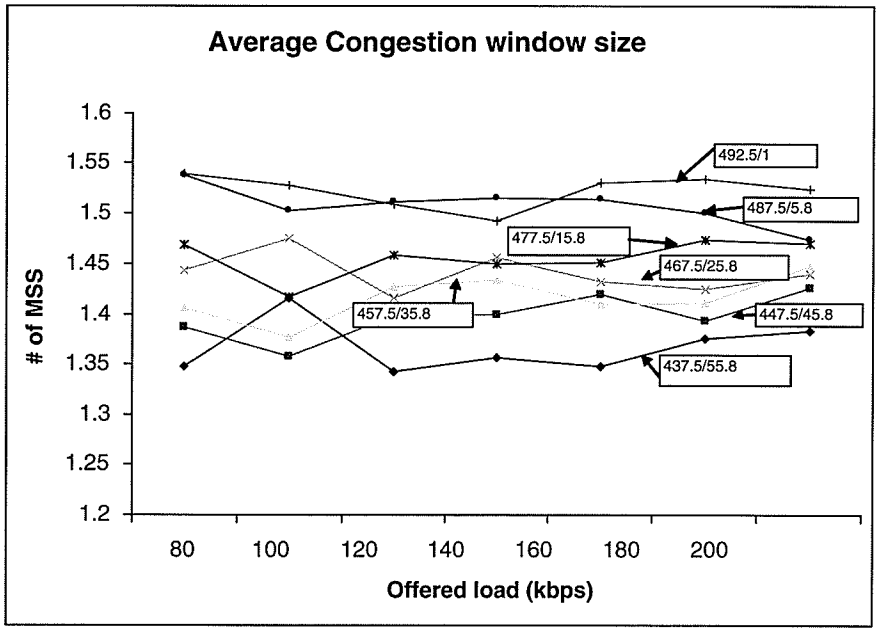
Where PER_G is packet error rate in good state,

PER_B is packet error rate in bad state,

P_g (Probability that the channel is in good state) = $\rho_G/(\rho_G + \rho_B)$

i.e. P_g = average duration of good state/(average duration of good +bad state)



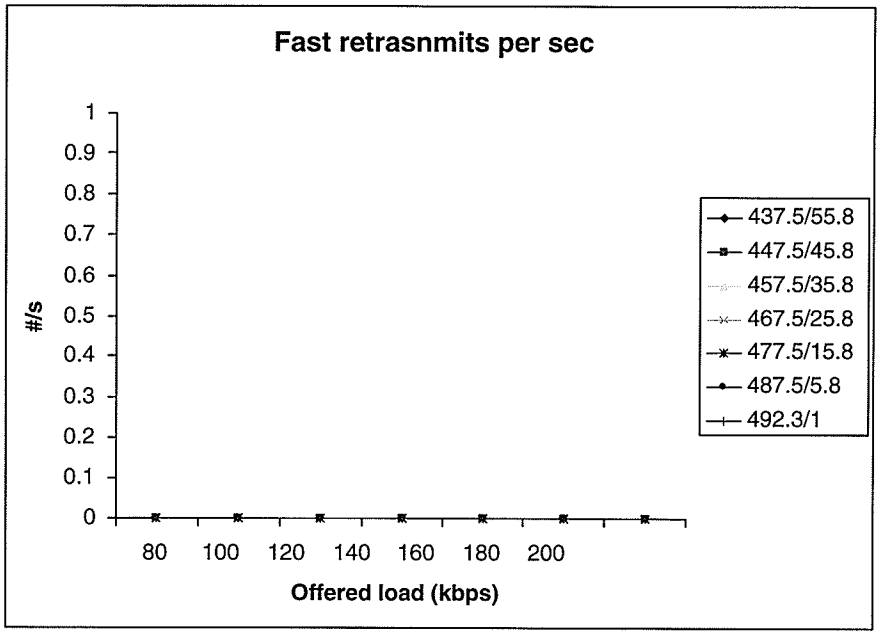


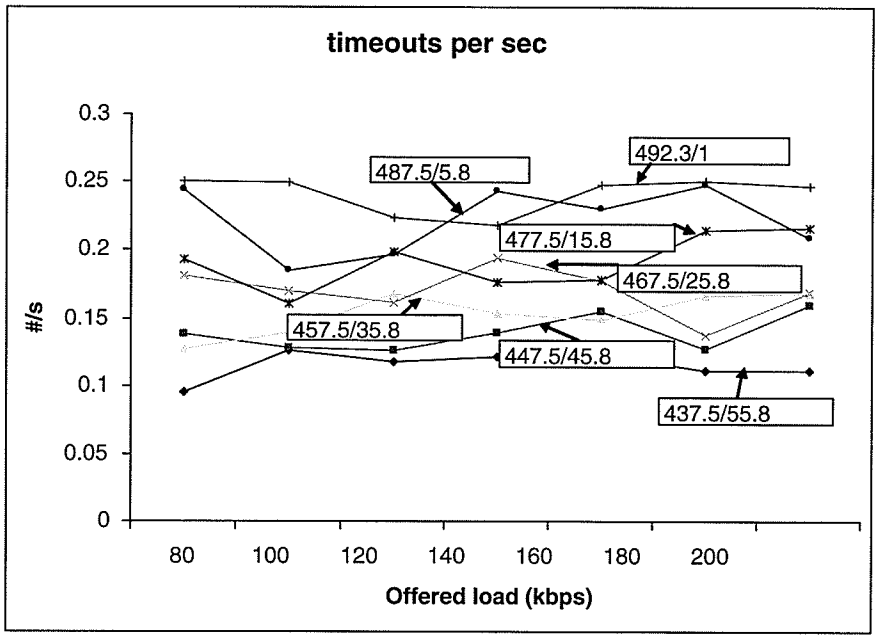
Legend Notation: 437.5/55.8 good duration 437.5 ms and bad duration 55.8 ms

---

tb = 250kbps, M = 5, S = 25, L = 20

Figure 5.3: [Sensitivity Analysis] TCP performance under a piconet with 1 master and 2 slaves (2 TCP connections)





Legend Notation: 437.5/55.8 good duration 437.5 ms and bad duration 55.8 ms

---

tb = 250kbps, M = 5, S = 25, L = 20

Figure 5.4: [Sensitivity Analysis] TCP performance under a piconet with 1 master and 2 slaves (2 TCP connections) (Cont'd)

Table 5.2: Success rate of a TCP segment including ACK for 2 hops with $P_G = 6.8979 * 10^{-5}$,$P_B = 1.263 * 10^{-3}$

ρ_G (good state duration)	ρ_B (bad state duration)	Success rate of a TCP segment including ACK
0.4375	0.0558	0.04043
0.4475	0.0458	0.04922
0.4575	0.0358	0.06335
0.4675	0.0258	0.08112
0.4775	0.0158	0.10339
0.4875	0.0058	0.13116
0.4933	0	0.15026

P_b (Probability that the channel is in bad state) $= 1 - P_g$

Therefore success rate for n-hops becomes $[[1 - PER_G]P_g + [1 - PER_B]P_b]^n$. The success rate of a TCP segment (segmented into 4 DH5 and 1 DH3 packets) including the acknowledgement (1 DH3) for 2 hops are summarized in Table 5.2 as follows.

From Table 5.2, we see that the success rate of a TCP segment including ACK increases from 4% to 15% when good state duration was increased from 437.5 ms to 493.3 ms and correspondingly reducing the bad duration state from 55.8 ms to 0 ms. Although the success rate increases with good state duration, the maximum it can go was only 15%.This means that even when the channel state was good during the entire period, the success rate was only 15%. With this success rate it is very difficult for packet to be successfully transmitted. This ultimately leads to timeout. Further, there is no guarantee that the retransmitted packed will be successfully transmitted with this 15% success rate which might lead to timeout once again. Each time, when a timeout occurs for the same packet, the retransmission period is doubled. For e.g. if the retransmission period was initially 1 sec, after a timeout it is doubled to 2 sec. Now if a timeout happens once again for the same packet, the retransmission period

is increased to 4 sec which goes to a maximum of 64 sec.

In Figure 5.4, timeouts increases with increase in good state. This was because the number of retransmits for a successful transmission of a packet was decreased with increase in good state (which increases the success rate of packet) which leads to increase in TCP goodput and congestion window. However, this increase in congestion window was not sufficient for triple duplicate acknowledgements which is evident from the congestion window (< 2 MSS) and fast retransmits (zero) graph. The successful transmission of a packet depends on the good duration state. Thus the time taken for a packet to get successfully transmitted reduces, the number of timeouts increase in my case to improve the TCP goodput.

Technically, if the success rate is 15% for a TCP segment, then the number of retransmission $N = 1/0.15 = 6.66$ times, as there were no FEC and ARQ schemes in the data link layer. During the timeouts, the system becomes totally idle because TCP is not able to distinguish packet losses due to buffer overflows or due to interference and assumes packet losses due to interference as indication of congestion and causes timeouts and doesn't allow any new packets to be transmitted during this period. With the number of retransmissions equal to 6.66 for a 15% TCP success rate, the worst idle period of the system will be somewhere between 64 and 128 seconds during the transmission of 1 TCP packet.

5.4 Trace files of varying load

Code validation ends by showing the trace files of TCP segments transmitted and acknowledged, congestion window and threshold, and DH5/DH3 packets received

along with the lost ones under a varying load condition.

Scenario considered: 2 slaves establishing independent TCP connection with each other in a piconet.

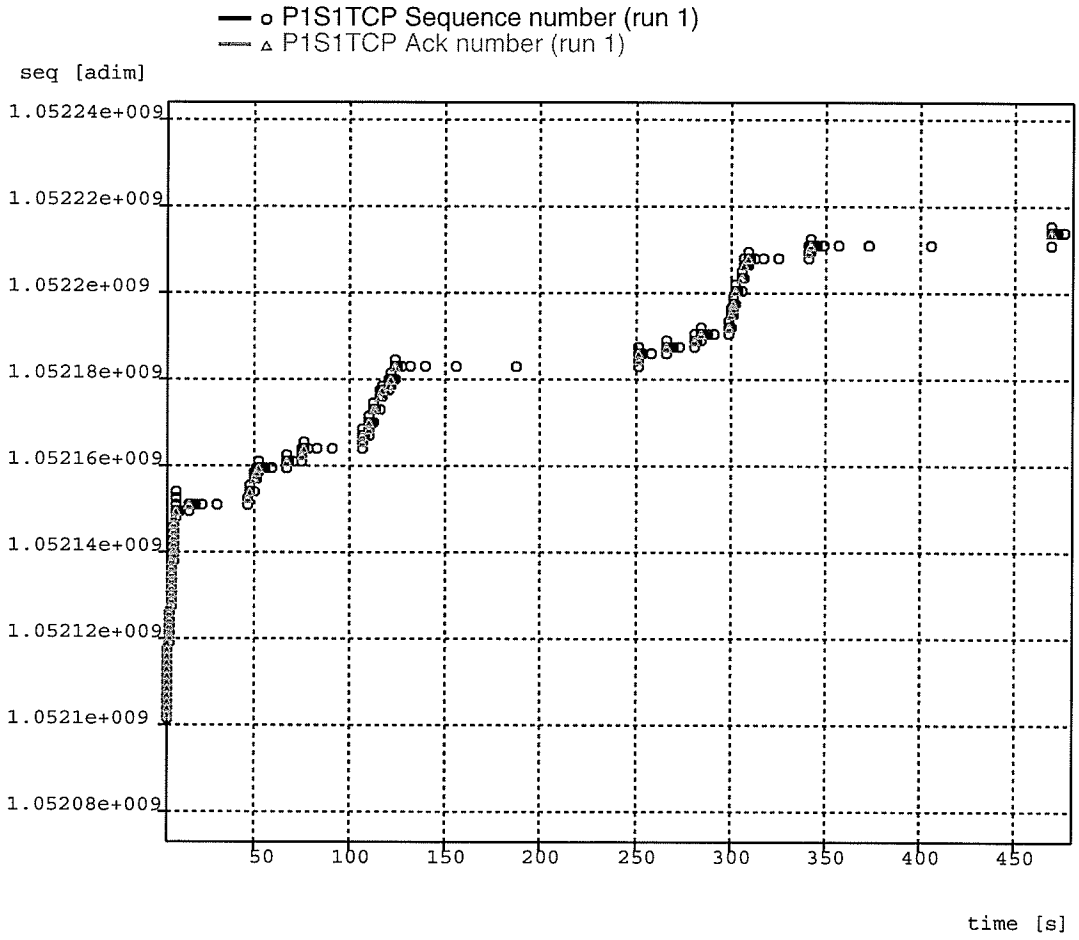
Offered load is varied as follows

Offered load	Time
80 kbps	<= 60 sec
120 kbps	> 60 and <= 120 sec
160 kbps	> 120 and <= 180 sec
200 kbps	> 180 sec

Parameter values: Token rate = 250 kbps, Scheduling parameter M = 5, Queue S = 25, buffer L = 20 and Receiver buffer size = 8MSS, good duration 437.5 ms and bad duration 55.8 ms Wireless medium enabled only after 10 sec.

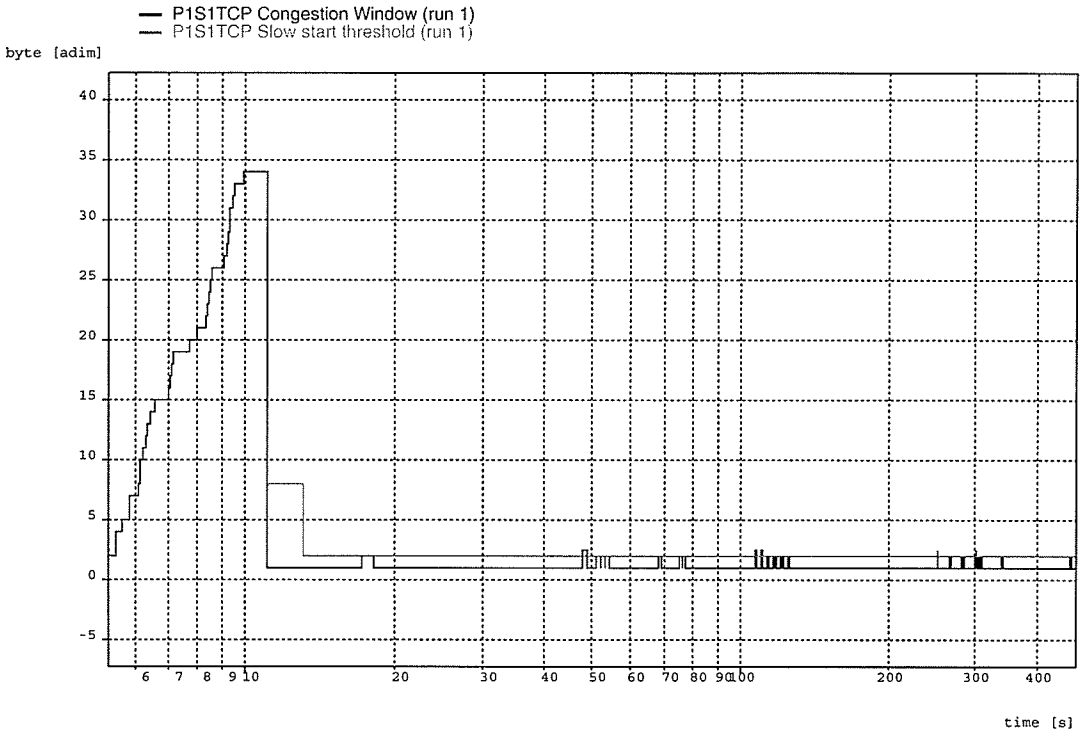
Figure 5.5 shows the TCP segments that are transmitted with their corresponding acknowledgements, while Figure 5.6 shows the congestion window and the threshold. The DH5/DH3 packets received in slave-1 are seen in Figure 5.7 along with the packet losses. We can see from Figure 5.6 that the congestion window increases very fast during the first 10 seconds of the simulation because during this period the wireless medium is not enabled. Therefore, no packets are lost due to interference in the wireless medium during this period. This feature is also seen in Figure 5.7 where many packets were received. After the 10th sec wireless medium was enabled and packet losses do take place due to interference. We can see from Figure 5.6 that congestion window after the 10th sec falls to 1MSS and thereafter hardly the congestion window grows which is evident from the packet loss (due to wireless medium) seen in Figure 5.5. Packet loss deteriorates the TCP performance to a large extent (similar results are also obtained in [35]). For e.g. in Figure 5.5 the packet that was transmitted roughly at 125-th sec was successfully transmitted only at around 252th

sec with the same packet being retransmitted 6 times making the system idle for approximately 127 seconds.



Offered load = varying load, tb = 250kbps, M = 5, S = 25, L = 20, good duration 437.5ms and bad duration 55.8ms

Figure 5.5: [Piconet with two slaves] Trace of TCP segments transmitted and their corresponding ACK's in Slave-1

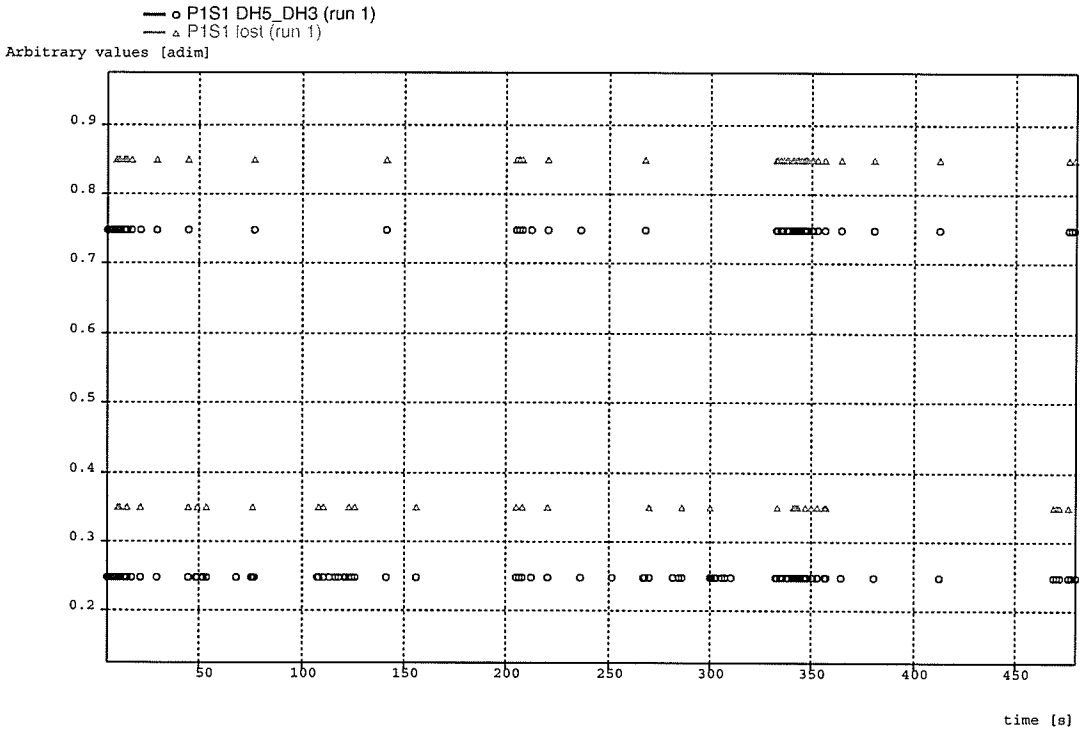


Offered load = varying load, tb = 250kbps, M = 5, S = 25, L = 20, good duration 437.5ms and bad duration 55.8ms

Figure 5.6: [Piconet with two slaves] Trace of Congestion window size with the threshold in Slave-1

5.5 Summary

Validation of good and bad periods in the wireless channels was done. Results pertaining to sensitivity analysis agree with the theoretical analysis indicating that goal of this chapter "code validation" was achieved.



.85 -> received DH5 is lost (after analyzing).

.75 -> received DH5

.35 -> received DH3 is lost (after analyzing).

.25 ->received

Offered load = varying load, tb = 250kbps, M = 5, S = 25, L = 20, good duration 437.5ms and bad duration 55.8ms

Figure 5.7: [Piconet with two slaves] Trace of DH5 and DH3 packets received in Slave-1 along with the lost ones

Chapter 6

Results and Discussion

6.1 Phase 1: Simulation results for TCP performance in a Bluetooth Piconet over a Perfect Physical Layer [PPL]

6.1.1 2 Slaves

To start with, only 2 active slaves were considered in a piconet establishing independent TCP connection with each other.

Offered Load Vs Token buffer S

Offered load was varied from 80 to 240 kbps and the Token buffer S from 5 to 25 baseband packets keeping the other parameters fixed [M = 5, Token rate *tb* = 250 kbps, Uplink Queue L = 20]. The TCP goodput, RTT, average Congestion window size, timeouts and fast retransmits obtained through simulations using Artifex

simulation software are shown in Figure 6.1. Increase in S from 5 to 25 improves the goodput by decreasing the timeouts and increasing the fast retransmits [larger buffer S means lower buffer loss].

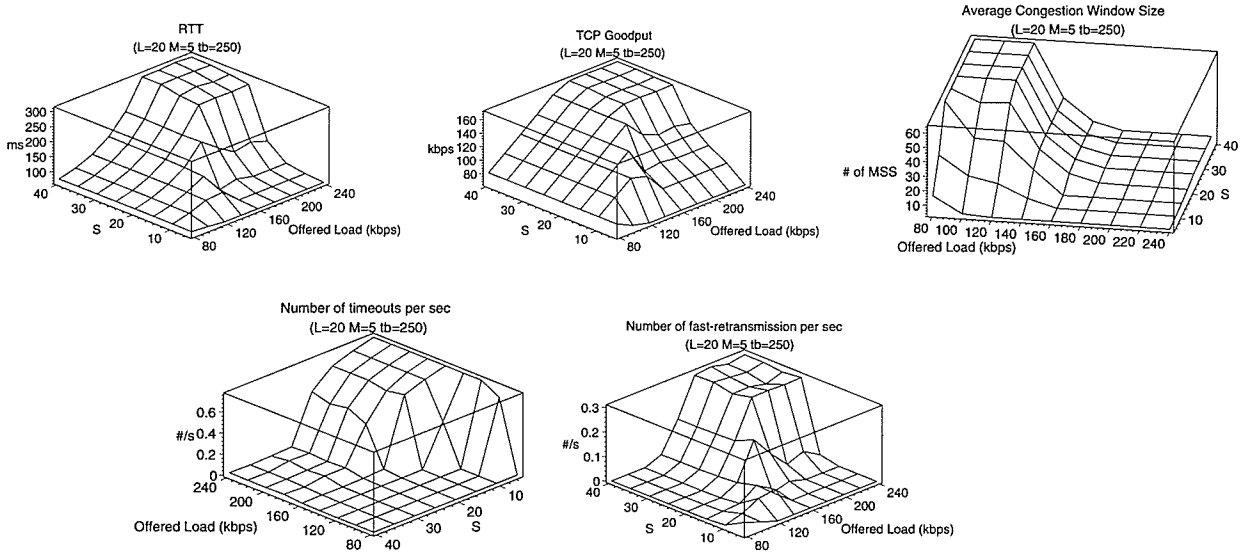


Figure 6.1: TCP NEW RENO performance as the function of buffer size S, in the piconet with two slaves [PPL]

The average congestion window size grows with S, which was an expected behavior [lower buffer loss builds congestion window]. At moderate and high load the congestion windows falls drastically because packet loss probability is directly proportional to offered load. At S = 25, TCP reaches its peak, capturing maximum number of fast retransmits and minimum timeouts with round trip delay of approximately 300 ms. Thereafter, TCP remains constant with further increase in S indicating that S = 25 was the optimal Token buffer size that gives maximum throughput with negligible timeout rates.

Offered Load Vs Scheduling Parameter M

Having found the optimal token buffer S, the next set of experiments was carried out to find the optimal scheduling parameter M by varying the offered load from 80 to 240 kbps and M from 5 to 25 keeping S = 25. The other fixed parameters are token rate tb = 250 kbps and Uplink Queue L = 20 baseband packets. The results are shown in Figure 6.2. Increase in M increases the time the piconet master takes to service a slave. Round trip delay increases from 250 ms to around 400 ms when M was increased from 5 to 25. The maximum goodput performance is achieved at M = 5, which equals to the number of baseband packets for a TCP segment, by capturing maximum fast retransmissions with negligible timeout rates.

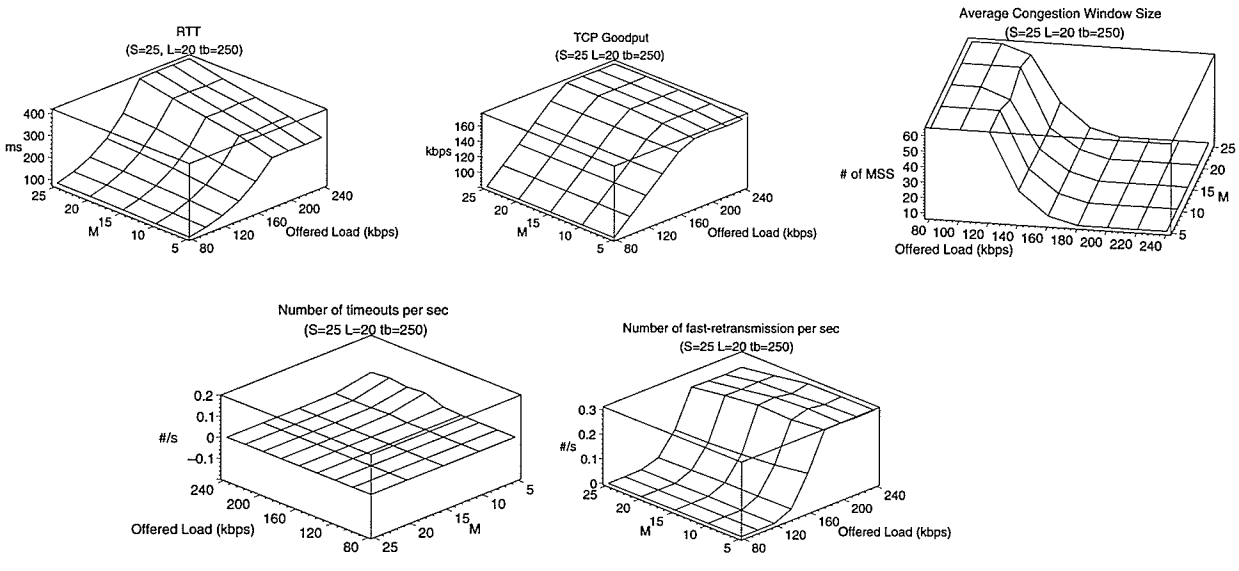


Figure 6.2: TCP NEW RENO performance as the function of Scheduling Parameter M, in the piconet with two slaves [PPL]

Offered Load Vs Token rate *tb*

Finally, the analysis of TCP performance for 2 slaves was investigated by varying the offered load and token rate *tb*. The results are shown in Figure 6.3. Increasing the token rate to a very high value [$tb > 250kbps$] beyond the maximum achievable goodput only decreases the TCP goodput because it increases the buffer blocking probability at the buffer L.

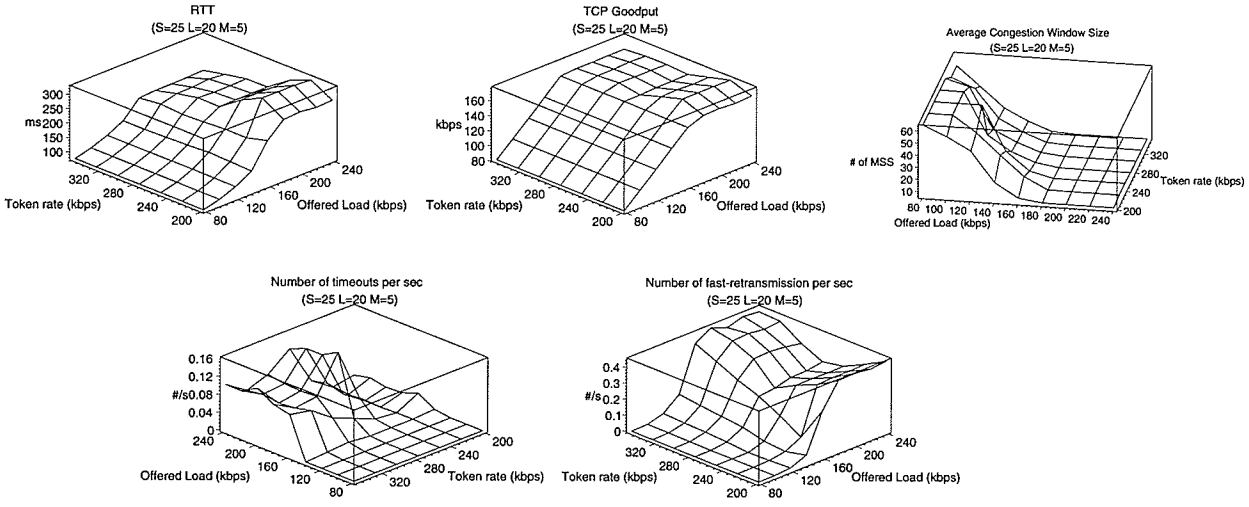


Figure 6.3: TCP NEW RENO performance as the function of token rate *tb*, in the piconet with two slaves [PPL]

6.1.2 4 and 6 Slaves

After the initial investigation of TCP performance for 2 slaves, the number of active slaves was increased to 4 and 6. Independent TCP connections were created in the following manner. For each slave i, for i = 1..n creates a TCP connection with slave j =(i+1) mod n (If j = 0 then slave j will be n) creating n identical TCP

connections where "n" is the number of active slaves. The TCP goodput, RTT, congestion window, fast retransmits and timeouts graphs were plotted by varying offered load with Token buffer S, Scheduling Parameter M, and token rate *tb*.

Offered Load Vs Token buffer S

The offer load and token buffer S for 4 and 6 slaves are varied as shown below

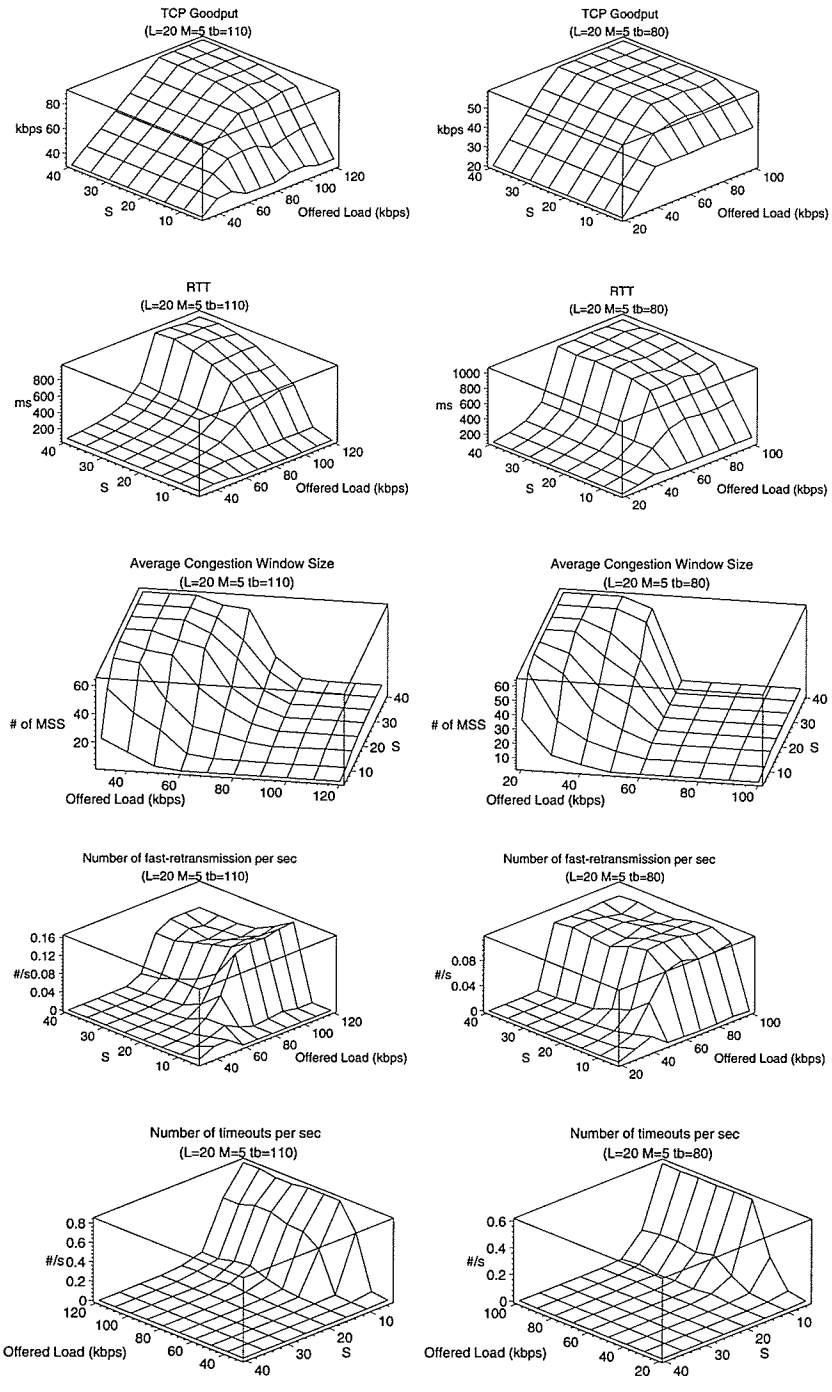
Slaves	Fixed Parameters	Offered Load	Token Buffer S
4	M = 5, *tb* = 110 kbps, L = 20	30 to 120 kbps	5 to 40
6	M = 5, *tb* = 80 kbps, L = 20	20 to 100 kbps	5 to 40

The results are shown in Figure 6.4. Network performance for 4 and 6 slaves are similar to that of 2 slaves. For 4 active slaves, the maximum goodput reachable was around 90 kbps (360/4) and for 6 active slaves, it was reduced to roughly 60 kbps (360/6). The goodput decreases with the increase in number of slaves. This is due to the fact that it increases the RTT, which takes TCP a longer time to react to packet losses. Round trip delay increased from 900 ms for 4 slaves to 1 sec in 6 slaves. Here also, S = 25 gives the maximum TCP goodput with maximum fast retransmits and negligible timeouts.

Offered Load Vs Scheduling Parameter M

Offered load and Scheduling parameter for 4 and 6 slaves were varied as shown below by fixing S = 25, the value that gave the maximum goodput. Figure 6.5 shows the results.

Slaves	Fixed Parameters	Offered Load	Scheduling Parameter M
4	S = 25, *tb* = 110 kbps, L = 20	30 to 120 kbps	5 to 25
6	S = 25, *tb* = 80 kbps, L = 20	20 to 100 kbps	5 to 25



4 Slaves 6 Slaves

Figure 6.4: TCP new Reno performance as a function of buffer size S, in the piconet with 4 and 6 slaves [PPL]

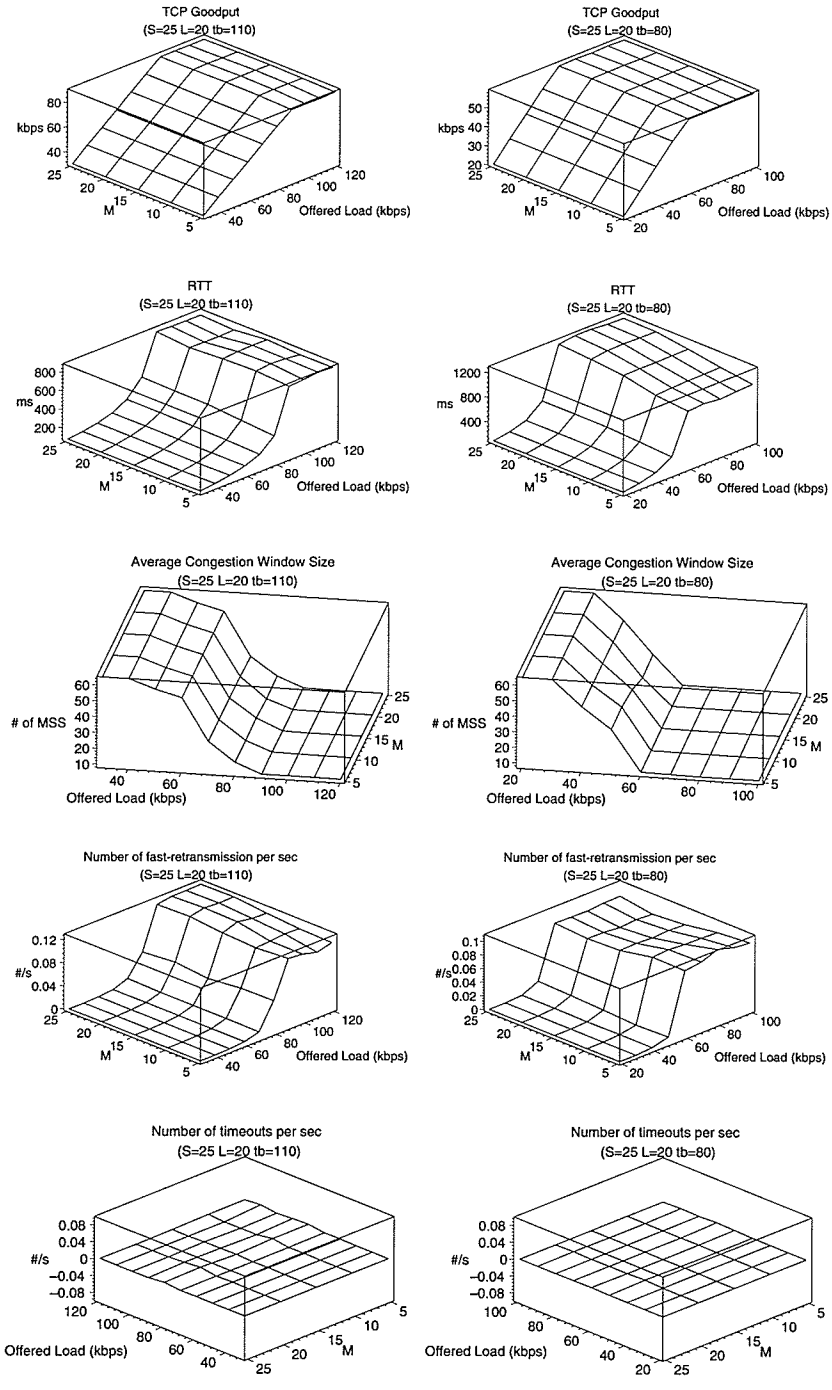
Similar characteristics of 2 slaves were seen for 4 and 6 slaves. Here also, for M = 5 baseband packets that required to transmit one TCP segment, gives the maximum goodput although not seen predominantly. Other characteristics, like increase in RTT and decrease in number of fast-retransmits are seen with the increase in number of slaves.

Offered Load Vs Token rate *tb*

Finally, the TCP performance for 4 and 6 slaves was studied by varying the offered load and token rate as shown below by fixing token buffer S = 25 and scheduling parameter M = 5 and the resulting graphs are shown in Figure 6.6

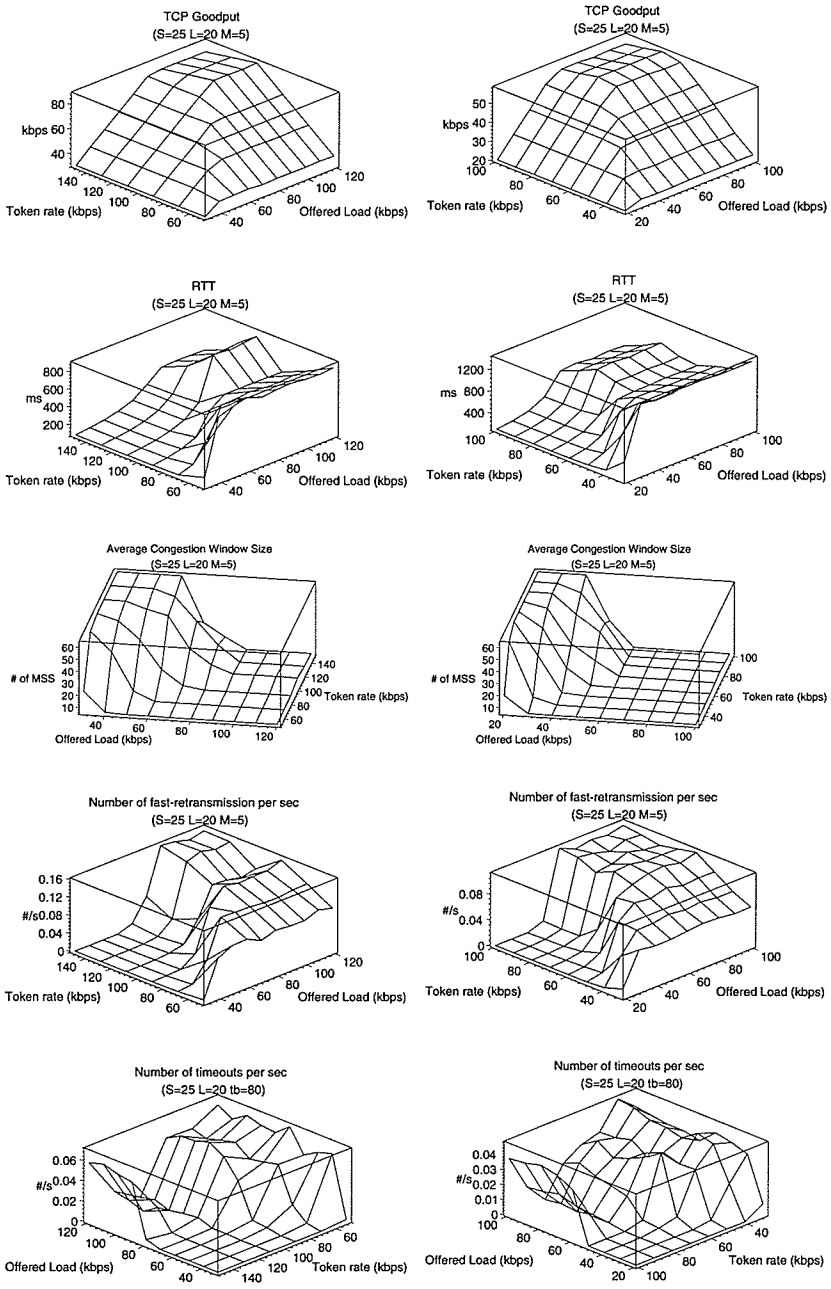
Slaves	Fixed Parameters	Offered Load	Token rate *tb*
4	M = 5, S = 25, L = 20	30 to 120 kbps	50 to 140 kbps
6	M = 5, S = 25, L = 20	20 to 100 kbps	30 to 100 kbps

Network characteristics of 2 slaves are repeated here too for 4 and 6 slaves. Token rates less than the maximum goodput and very high token rates(> 50% of maximum achievable goodput), increases the blocking probability for buffer S and uplink queue L respectively, thereby deteriorating TCP goodput. Figure 6.6 shows that increase in buffer blocking probability (token buffer S and uplink queue L), decreases the round trip delay because of retransmits taking place, which makes the buffer free. The decrease in RTT should not be considered as a better performance indicator. For better performance the buffers should be used efficiently.



4 Slaves 6 Slaves

Figure 6.5: TCP new Reno performance as a function of Scheduling Parameter M, in the piconet with 4 and 6 slaves [PPL]



4 Slaves 6 Slaves

Figure 6.6: TCP new Reno performance as a function of token rate *tb*, in the piconet with 4 and 6 slaves [PPL]

6.2 Phase 2: Simulation results for TCP performance in a Scatternet over a PPL

In phase 2, the performance analysis of TCP traffic was extended to a Bluetooth scatternet that has two piconets connected through a SS bridge. As explained in 3.1.1, the bridge will remain in a piconet for only one piconet cycle before it switches to the other. The poll value for bridge Mb was kept at Mb = n * M where "n" is number of active slaves in a piconet. The bridge buffer size is kept to Mb. Selecting Mb this way helps in transferring the complete load that was given by all the slaves to the bridge in one piconet cycle. Also by adopting this method, the bridge buffer size will be optimal with no overflows in it.

6.2.1 1 Slave / Piconet

To start with, a scatternet with 2 piconets was considered having only one slave each. Each of these slaves establishes independent TCP connection with each other through SS-bridge. Other slaves are not active.

Offered Load Vs Token buffer S

The values of TCP goodput, RTT, congestion window, TCP timeouts and fast retransmits obtained through Artifex simulation software are shown in Figure 6.7 with the following constants M = 5, token rate tb = 250 kbps, Uplink Queue L = 20. Token buffer size was varied from 5 to 25 baseband packets and offered load was varied between 80 and 180 kbps. For S = 15 TCP goodput reaches its peak of 160

kbps. Further increase in S to 25 indicates that there was no packet loss which can be seen in timeouts, fast retransmits, and congestion window graphs. Higher number of timeouts can be noticed for S = 5. Increase in S to 10, reduces the timeouts to zero while increasing the fast retransmits which was an expected behavior.

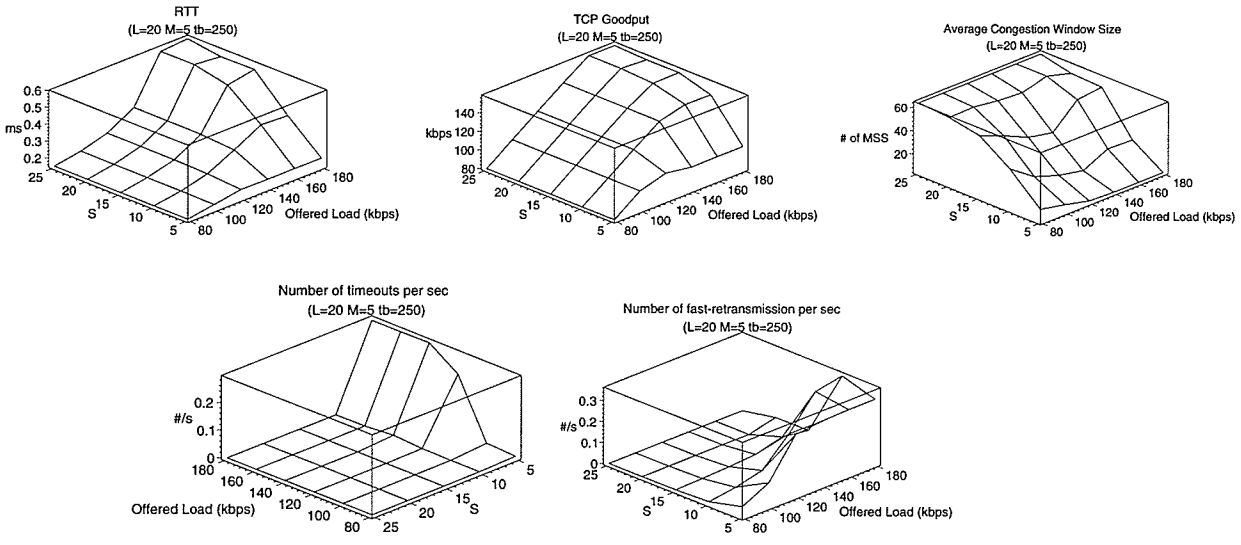


Figure 6.7: TCP NEW RENO performance as the function of buffer size S, in the Scatternet (1 slave/piconet) [PPL]

For S = 15 and S = 20, the congestion window drops especially when the offered load was between 100 and 140 kbps. At these moderate offered load, which is very much less that the capacity that the system can support, leads to situations where the number of unacknowledged packets might be zero. However, the maximum amount of unacknowledged packets in the network [minimum(congestion window, receiver window)] still remains high. On such a scenario, further messages created by slave's application unit are directly sent to token buffer S (TCP has no control over the sender in this situation), which leads to high buffer blocking probability of S or the Uplink

queue L based on the token rate. In this case, it is the buffer blocking probability of S as the token rate of 250 kbps was not very high. However, increase in S to 25 was able to withstand the moderate load scenario.

Increase in S from 5 to 15, increases the RTT. Thereafter it remains constant indicating the system has become stable. Compared to TCP performance for 2 slaves in a piconet, goodput has been reduced due to increase in RTT to 600 ms (double the RTT value for 2 slaves). Increase in RTT was an expected behavior as the path traversed was increased from 2 in a piconet to 4 in a scatternet.

Offered Load Vs Scheduling Parameter M

Having completed the investigation of token buffer S, the next set of experiments were carried out by varying Scheduling parameter M from 5 to 25 and the offered load from 80 to 180 kbps keeping S at 25. The other constants are token rate tb = 250 kbps and Uplink Queue L = 20. The results are shown in Figure 6.8. Increase in M decreases the TCP goodput, reflected with an increase in RTT. There are no packet losses with change in M indicating that for S = 25 the system has become stable irrespective of the scheduling parameter M. For M = 5 gives the optimal output indicating that it is better than the exhaustive service.

Offered Load Vs Token rate tb

From the above discussions these constant values S = 25, M = 5 along with Uplink Queue L = 20 was used in finding the relationship between offered load and token rate. The results are shown in Figure 6.9. For token rate of 175 kbps, there was large number of fast retransmits and some timeouts. This is because too small tb increases

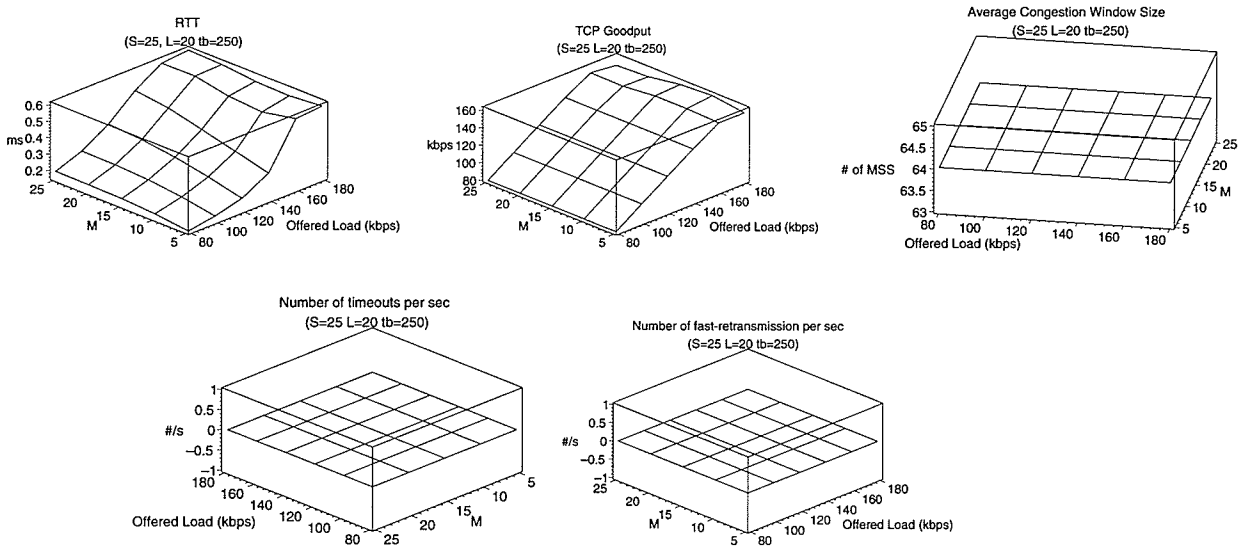


Figure 6.8: TCP NEW RENO performance as the function of Scheduling parameter M, in the Scatternet (1 slave/piconet) [PPL]

the blocking probability of the token buffer S. For token rate of 200 kbps and above (exception 225 and 250 kbps) there were no packet losses except at moderate offered load (100 to 140 kbps) indicating that the system has become stable. For token rates less than $225kbps$ there are packet losses at moderate offered load indicating the blocking probability of the token buffer S and for token rate greater than 275 kbps. The packet losses are due to the blocking probability of the uplink queue L.

6.2.2 2, 4 and 6 Slaves / Piconet

Here the scatternet with two piconets was considered. The number of active slaves per piconet was increased to two, four and six. Independent TCP connections between slaves are created as follows. P-1-S-1 (piconet one slave id one) and P-2-S-1 establishing independent TCP connections with each other. Similarly P-1-S-2 with

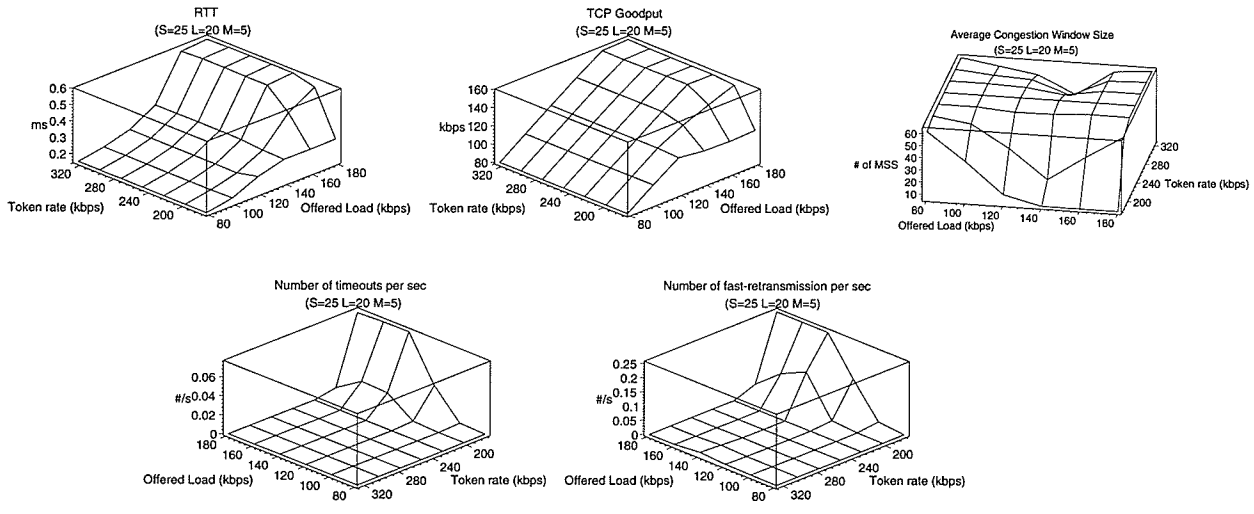


Figure 6.9: TCP NEW RENO performance as the function of token rate, in the Scatternet (1 slave/piconet) [PPL]

P-2-S-2, P-1-S-3 with P-2-S-3 etc., The graphs of TCP goodput, RTT, congestion window, TCP timeouts and fast retransmits are been plotted by varying the offered load vs. S, offered load vs. M and offered load vs. token rate.

Offered Load Vs Token buffer S

The offered load and token buffer S were varied as shown below and Figure 6.10 shows the resulting graphs.

Slaves/Piconet	Fixed Parameters	Offered Load	Token buffer S
2	M = 5, tb = 120 kbps, L = 20	40 to 100 kbps	5 to 25
4	M = 5, tb = 70 kbps, L = 20	25 to 55 kbps	5 to 25
6	M = 5, tb = 50 kbps, L = 20	15 to 35 kbps	5 to 25

Similar network performance to the case when one slave resided per piconet is seen. There are more packet losses at S = 5 and 10 and with values of S greater than 10, small packet losses takes place at moderate offered load. The maximum TCP

goodput decreases with the increase in number of slaves for piconet. For two slave per piconet it was 80 kbps and for 6 slaves it was around 27 kbps. RTT also increased from 1.2 sec for two slaves per piconet to 3.2 sec for 6 slaves per piconet. This was an expected behavior as increase in the number of slaves increases the time taken for the master to revisit the slave.

Offered Load Vs Scheduling Parameter M

Offered load and Scheduling parameter M were varied as shown below

Slaves/Piconet	Fixed parameters	Offered load	Scheduling Parameter M
2	S = 25, *tb* = 120 kbps, L = 20	40 to 100 kbps	5 to 25
4	S = 25, *tb* = 70 kbps, L = 20	25 to 55 kbps	5 to 25
6	S = 25, *tb* = 120 kbps, L = 20	15 to 35 kbps	5 to 25

Figure 6.11 shows the resulting graphs. Network performance similar to the one slave residing per piconet was observed. Maximum TCP goodput decreases with increase in M. There were packet losses at moderate offered load, which was not the case with one slave/piconet. This was due to the increase in number of slaves/piconet. For lower loads the timeouts and fast retransmits are less compared to moderate offered loads. For higher offered loads the packet losses were less compared to lower and moderate offered loads indicating TCP has become stable.

Offered Load Vs Token rate *tb*

Offered load and token rate *tb* were varied as shown below and the resulting graphs are shown in Figure 6.12

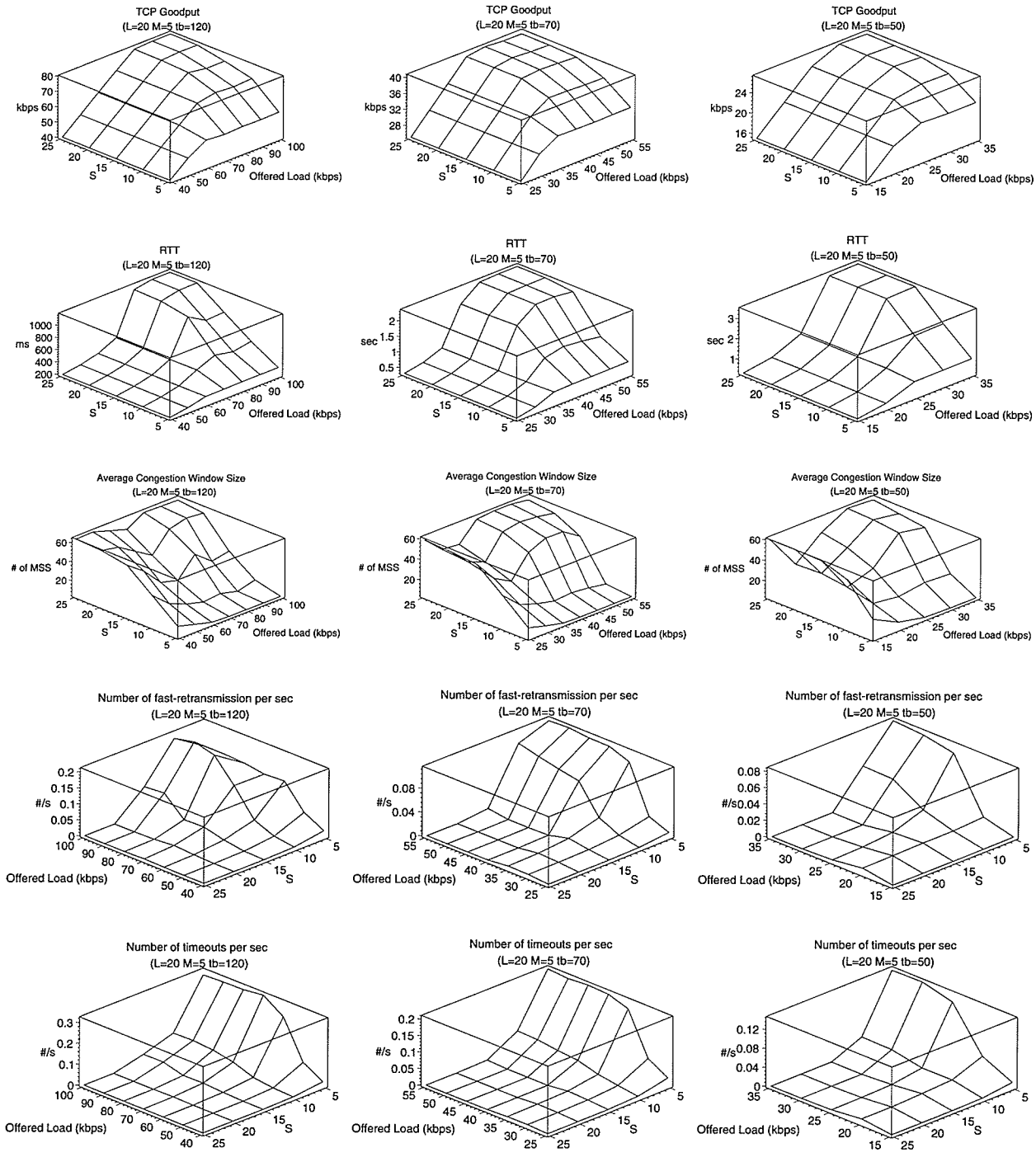
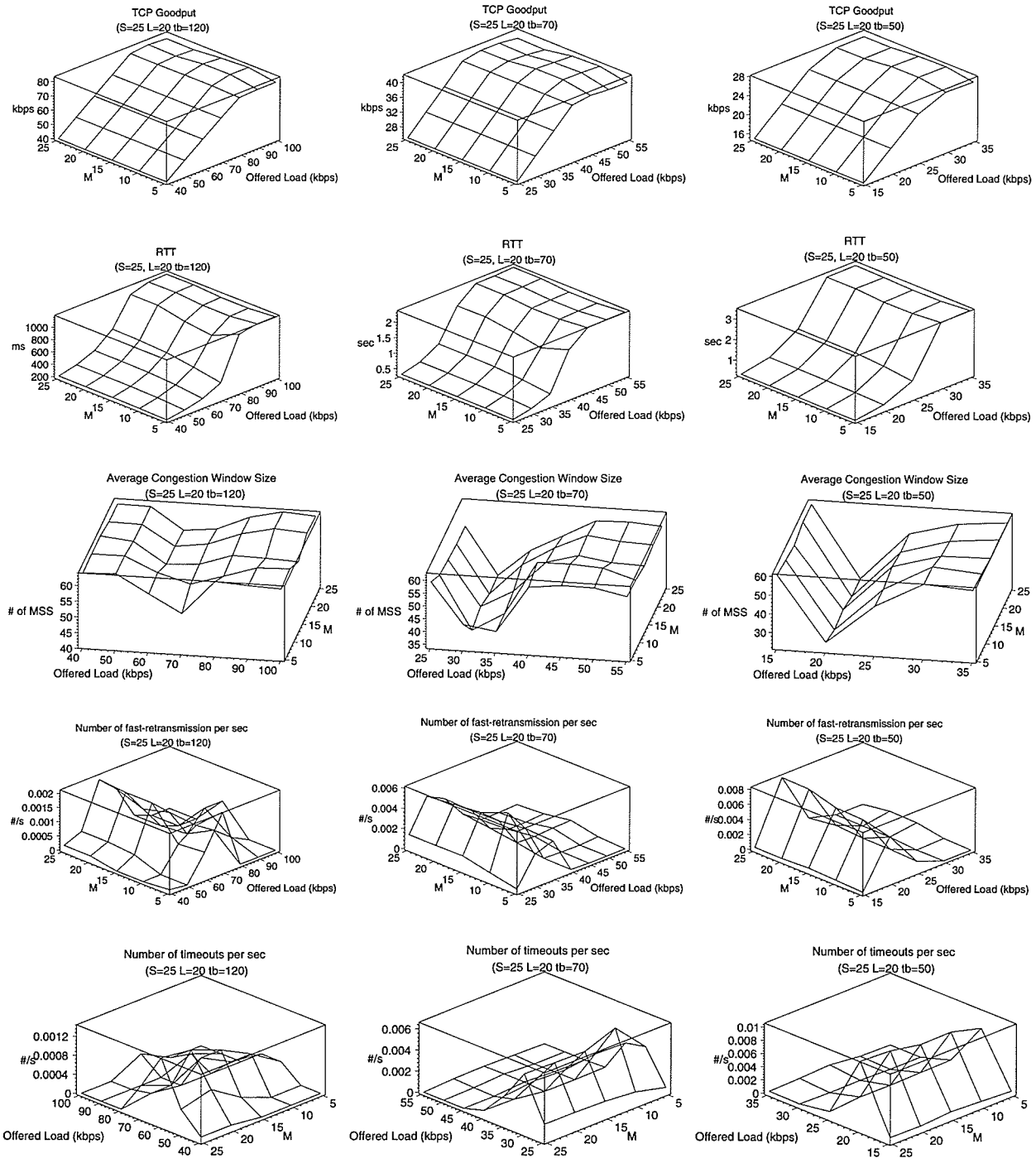


Figure 6.10: TCP new Reno performance as a function of buffer size S, in the Scatternet (2, 4, 6 slaves/piconet)[PPL]

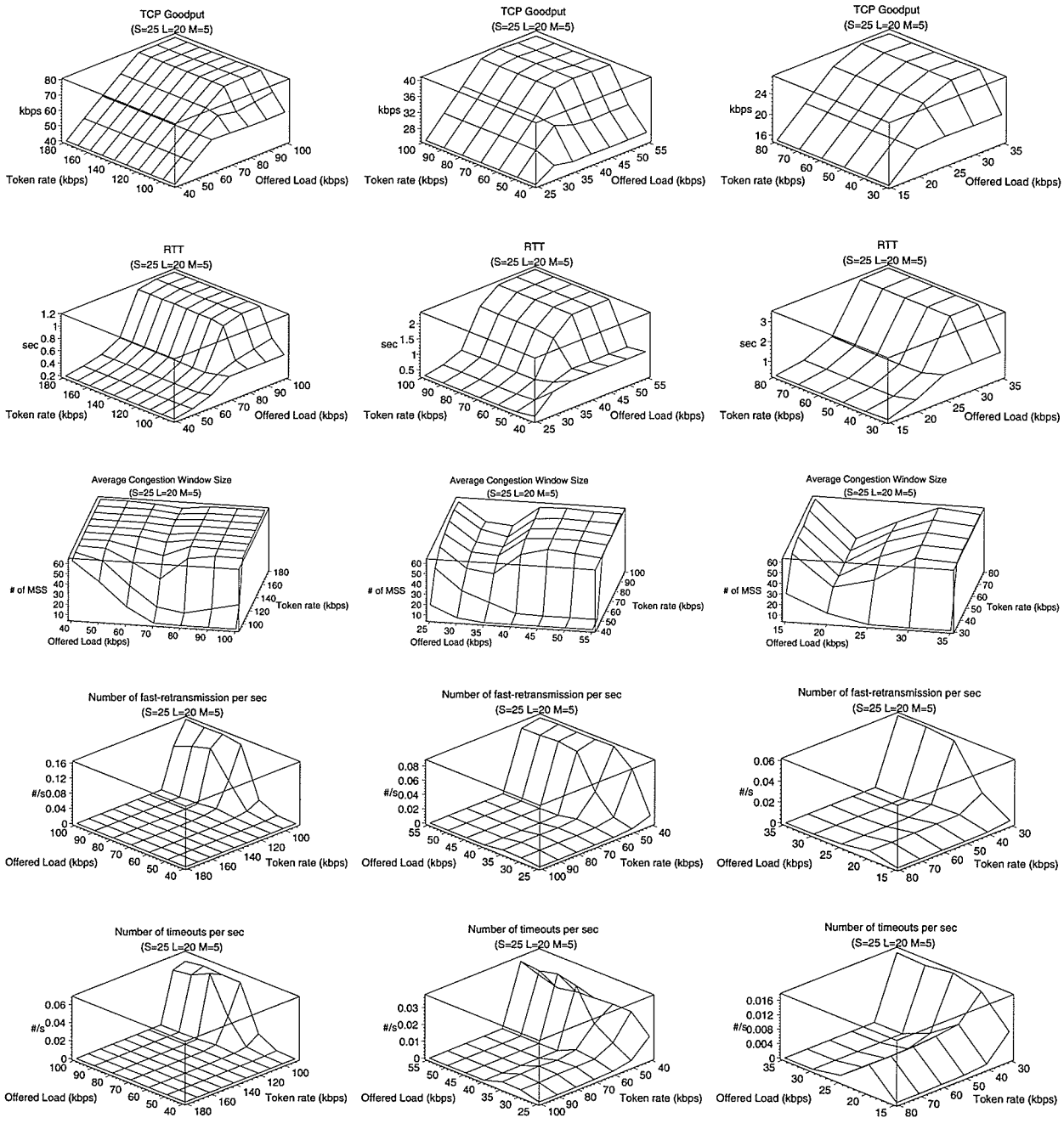


2 Slaves/Piconet 4 Slaves/Piconet 6 Slaves/Piconet

Figure 6.11: TCP new Reno performance as a function of Scheduling Parameter M, in the Scatternet (2, 4, 6 slave/piconet)[PPL]

Slaves/piconet	Fixed parameters	Offered load	Token rate *tb*
2	S = 25, M = 5, L = 20	40 to 100 kbps	90 to 180 kbps
4	S = 25, M = 5, L = 20	25 to 55 kbps	40 to 100 kbps
6	S = 25, M = 5, L = 20	15 to 35 kbps	30 to 80 kbps

The one slave/piconet characteristic is repeated for 2, 4 and 6 slaves resided per piconet. At low token rate, there were packet losses which are seen in timeouts, fast retransmits and congestion window graphs, indicating the increase in buffer blocking probability at token buffer S. Correspondingly fall in the TCP goodput was also seen. At too high token rate, there were some packet losses at moderate offered loads which indicates the increase in blocking probability of uplink queue L. However, this effect was not seen with high offered load indicating that the TCP has become stable.



2 Slaves/Piconet 4 Slaves/Piconet 6 Slaves/Piconet

Figure 6.12: TCP new Reno performance as a function of toke rate *tb*, in the Scatternet (2, 4, 6 slaves/piconet)[PPL]

6.3 Phase 3: Simulation results for TCP performance in a Piconet/Scatternet over a Imperfect Physical Layer [IPL]

TCP was originally designed for wired networks where packet losses take place due to buffer overflows. However, in wireless transmission such as Bluetooth, packet losses can take place not only by buffer overflows, but also due to noise (interference due to other Bluetooth devices or 802.11 devices or other cordless devices working in the same 2.4 GHz frequency) in the wireless medium. In this phase, the performance of TCP new Reno was studied in a piconet and later in a scatternet with the wireless medium.

Having found in Section 5 (code validation) that TCP performance was very poor due to very high error rates the following new error rates were used in the rest of the simulation $P_G = 6.8979 * 10^{-6}$,$P_B = 1.263 * 10^{-4}$, $\rho_G = 437.5ms$, $\rho_B = 55.8ms$. The corresponding error rates and TCP success rates are shown in Table 6.1 and Table 6.2

Table 6.1: New PER for various packets under consideration

Packet Type	Packet Size in bits	Packet Length (Bluetooth slots)	State	PER
DH5	2712 (339 bytes)	5	Good	0.018483
			Bad	0.29004
DH3	1464 (183 bytes)	3	Good	0.01002
			Bad	0.168825

Table 6.2: New TCP segment success rate for 2 hops and 4 hops

TCP segment success rate	2 hops	4 hops
	0.59622	0.355478

6.3.1 Piconet

The performance of TCP traffic in a piconet was analyzed in an error-prone Bluetooth environment.

Two slaves

Initially the system was tested with only two slaves active in a piconet, establishing independent TCP connection with each other. The offered load was varied from 80 to 200 kbps and the token buffer S was varied from 5 to 25. The values of TCP goodput, RTT, congestion window, timeouts, and fast retransmits obtained through Artifex simulation software are shown in Figure 6.13 with the following constants token rate tb = 250 kbps, Uplink Queue L = 20 and scheduling parameter M = 5.

TCP goodput increases with the increase in buffer S. However, there was no change in goodput with increase in load. Moreover, the maximum goodput reached was only around 44 kbps. Also, RTT, congestion window and fast retransmits increases with increase in S which was an expected result. Larger S means less buffer loss. The number of timeouts increases with the increase in S. This is because the system was being tested in an IPL where packet losses also take place due to interference or noise in the medium. As seen from Table 6.2, the success rate of a TCP segment in 2 hops is 59.6%. So this increases the number of timeouts as the average congestion window was only around 2.9 MSS when S = 25, which indicates the number of fast

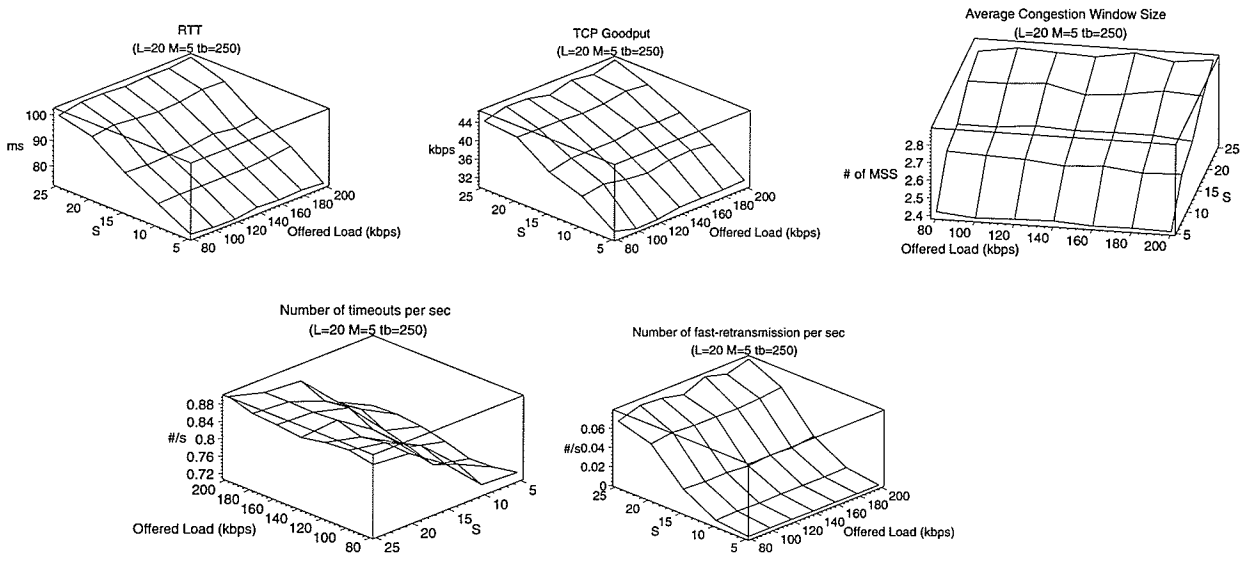


Figure 6.13: TCP NEW RENO performance as the function of buffer size S, in the piconet with two slaves [IPL]

retransmits was low and it's only the timeout mechanism that plays a major role in it. Also, a close look at RTT indicates that for S = 25, RTT was only 100 ms meaning the buffers are mostly free during the simulation period.

4 and 6 Slaves

Here the number of slaves was increased to four and six slaves establishing independent TCP connections with the adjacent slaves. i.e. slave 1 establishing TCP connection with slave2, slave 2 with slave 3 etc. The graphs of TCP goodput, congestion window, RTT, TCP timeouts, fast retransmits are shown in Figure 6.14 by varying the offered load and token buffer S as shown below

No of slaves	Fixed parameters	Offered load	Token buffer S
4	M = 5, tb = 110 kbps, L = 20	40 to 100 kbps	5 to 25
6	M = 5, tb = 80 kbps, L = 20	30 to 90 kbps	5 to 25

Similar network performance of two slaves was observed with four and six slaves. The TCP goodput when S = 25 decreases to 34 kbps in four slaves and in six slaves it was around 28 kbps with the corresponding drop in the congestion window, number of fast retransmits and timeouts. However, RTT was increased to 200 ms in four slaves to 280 ms in six slaves. This is an expected behavior (Increase in number of slaves increases the time taken for the master to visit the slave once again). TCP goodput and timeouts are dropped with the increase in number of slaves per piconet, due to increase in RTT. Here too, the number of timeouts increases with S for 4 slaves and for 6 slaves it remains constant (exception S = 5). For 6 slaves, timeouts was less for 30 kbps offered load, as the maximum goodput reached was matching to that of the offered load.

6.3.2 Scatternet

1 Slave/Piconet

Here a scatternet with 2 piconets was considered with one slave each establishing independent TCP connection with each other in IPL through a SS bridge with no other slaves active.

The offered load was varied from 80 to 180 kbps and the token buffer S from 5 to 25 and the values of TCP goodput, RTT, congestion window, timeouts and fast retransmits drawn through Artifex are shown in Figure 6.15 with the following constants M = 5, token rate tb = 250 kbps, uplink Queue L = 20. It can be noticed that, there was not much change in any of these graphs. TCP goodput was averaging around 14 kbps irrespective of increase in offered load and S. With the success rate

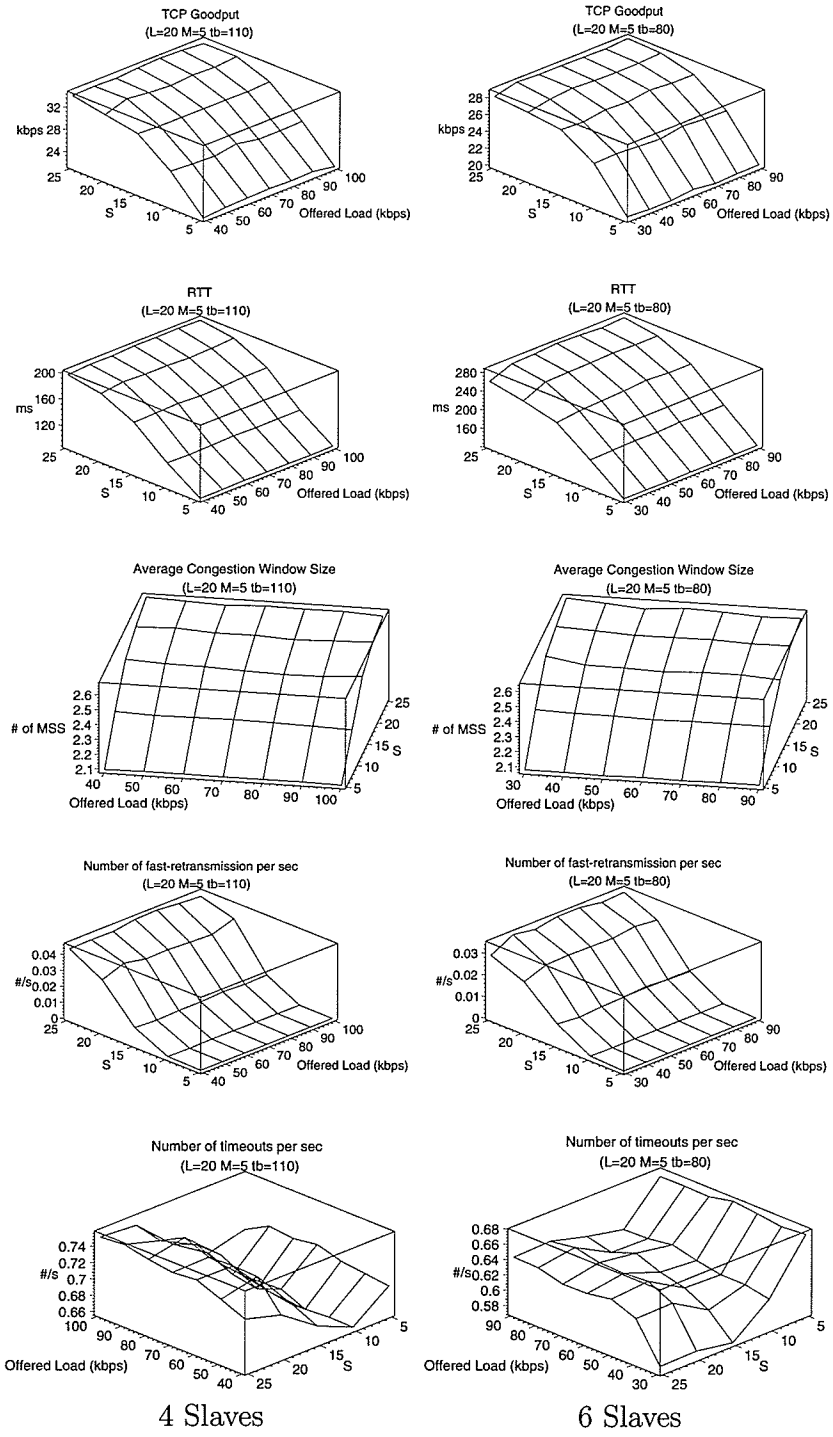


Figure 6.14: TCP new Reno performance as a function of buffer size S, in the piconet with 4 and 6 slaves [IPL]

of a TCP segment in 4 hops being 35.5% it's only through the timeouts that play a major role irrespective of the buffer size S.

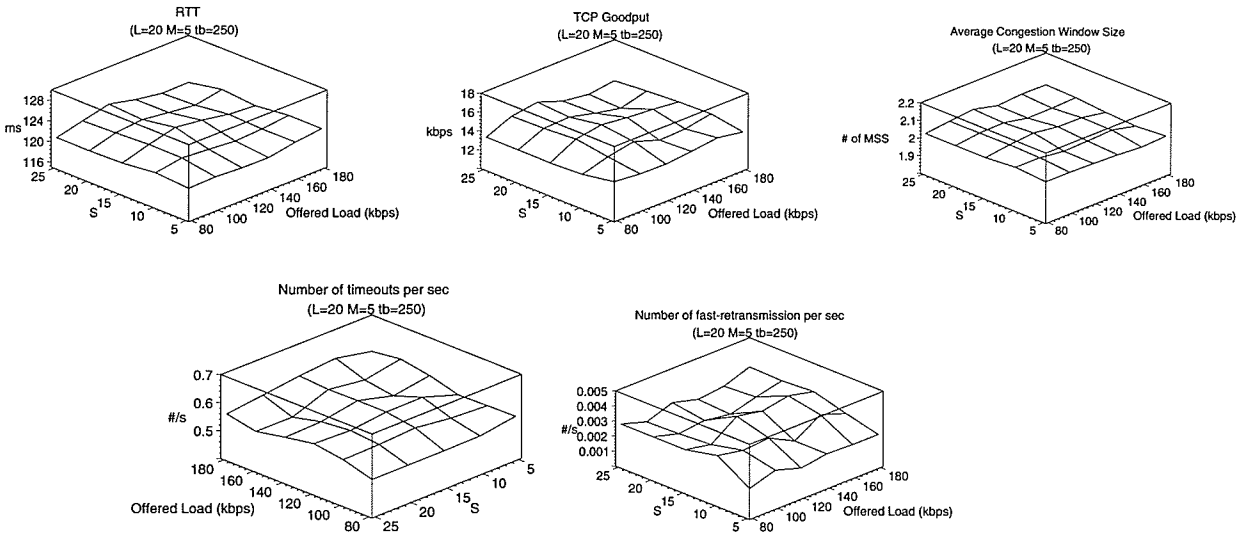


Figure 6.15: TCP NEW RENO performance as the function of buffer size S, in the Scatternet (1 slave/piconet) [IPL]

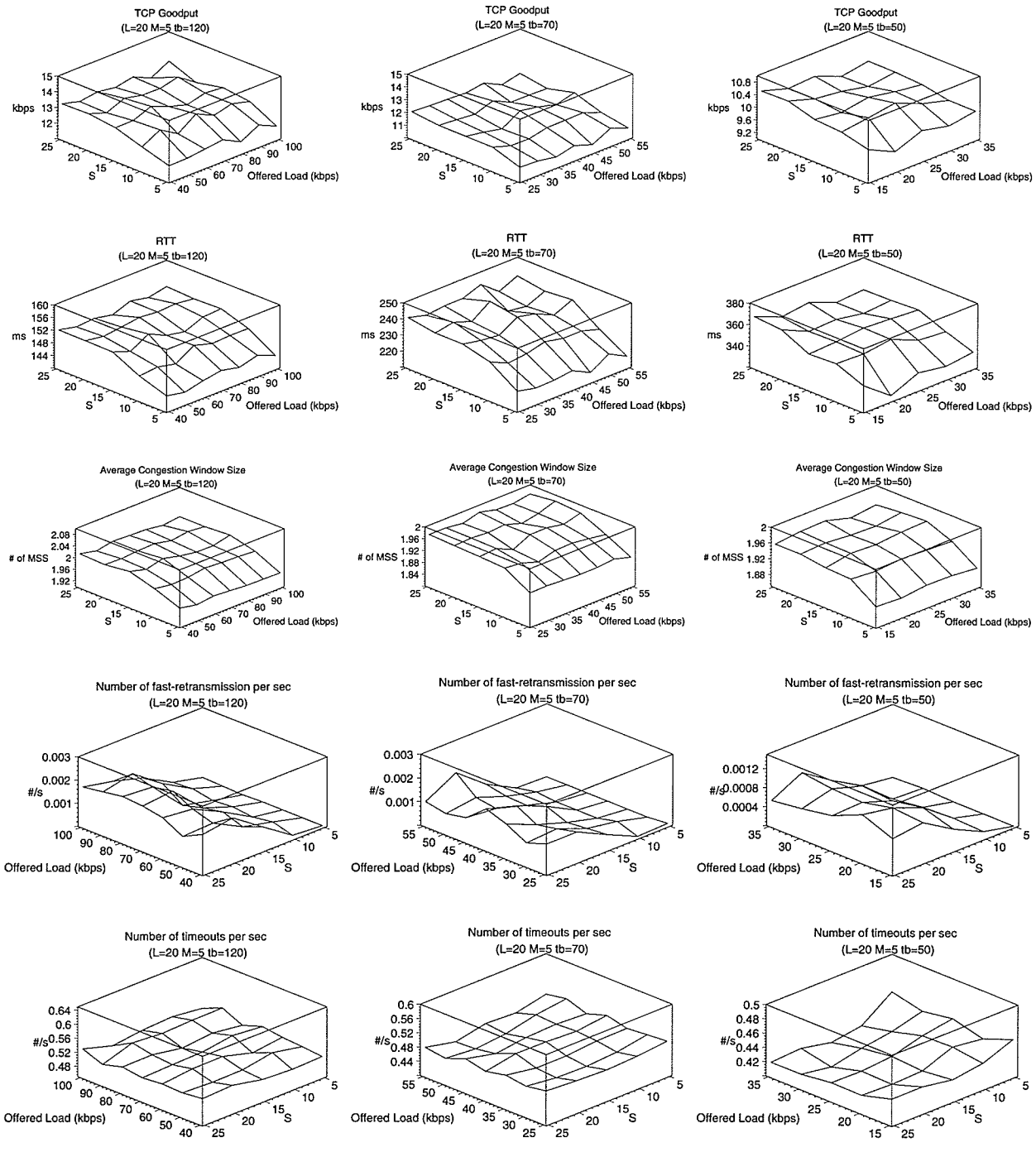
2, 4 and 6 Slaves/piconet

The number of slaves was increased to two, four and six slave in each piconet and the performance of TCP was analyzed in an IPL. The token buffer and offered load were varied as shown below

Slaves/piconet	Fixed parameters	Offered load	Token buffer S
2	M = 5, tb = 120 kbps, and L = 20	40 to 100 kbps	5 to 25
4	M = 5, tb = 70 kbps, and L = 20	25 to 55 kbps	5 to 25
6	M = 5, tb = 50 kbps, and L = 20	15 to 35 kbps	5 to 25

Figure 6.16 shows the resulting graphs. Similar characteristics of one slave/piconet like uniform TCP goodput, RTT, congestion window, timeouts and fast retransmits

was seen for buffers S = 10 to 25. TCP goodput decreases with increase in number of slaves/piconet with a corresponding increase in RTT and decrease in congestion window, timeouts and fast retransmits, which is all an expected behavior. Once again timeouts, play a major role here too with TCP segment success rate being 35.5%. The number of timeouts decreases with increase in number of slaves/piconet due to increase in RTT.



2 Slaves/Piconet 4 Slaves/Piconet 6 Slaves/Piconet

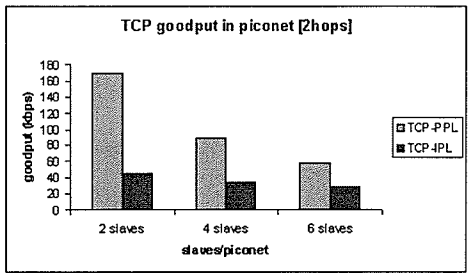
Figure 6.16: TCP new Reno performance as a function of buffer size S, in the Scatternet (2, 4, 6 slaves/piconet)[IPL]

6.4 Comparative analysis of TCP over PPL and IPL

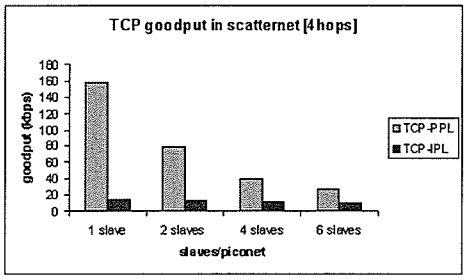
Finally, the comparative analysis of TCP performance over PPL and IPL were performed. Figure 6.17 shows the TCP new Reno performance in two and four hops respectively with the TCP goodput, RTT and timeouts when scheduling parameter M = 5 and token buffer S = 25 with the token rates [piconet: 2 slaves = 250 kbps, 4 slaves = 110 kbps, 6 slaves = 80 kbps and Scatternet: 1 slave/piconet = 250 kbps, 2 slaves/piconet = 120 kbps, 4 slaves/piconet = 70 kbps, 6 slaves/piconet = 50 kbps].

In PPL, the TCP goodput in a piconet decreases with increase in number of slaves (2 slaves to 4 slaves) from 170 kbps to 60 kbps. In a scatternet the TCP goodput drops to around 25 kbps when the scatternet is fully loaded. In IPL, similar characteristic of TCP goodput in PPL was seen in piconet however in a small magnitude.

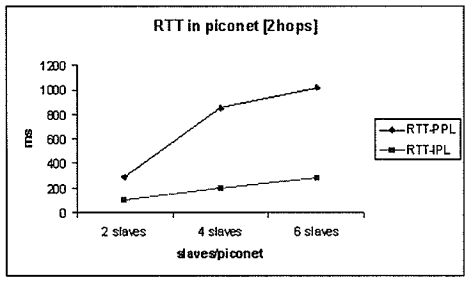
RTT increases with increase in hops for both IPL and PPL. In PPL, RTT in a piconet increases from 200ms to more than 1 sec with the increase in number of slaves from 2 to 6. In scatternet RTT increases from 500 ms when 2 slaves resided per piconet to more than 3.5 sec when 6 slaves resided per piconet. In PPL the increase is RTT is very high whereas in IPL it is not so high. This is because of large number of timeouts taking place in IPL, making the buffers more free comparative to PPL. Also it can be found that timeouts are negligible for PPL due to new RENO mechanism (new RENO prevents timeouts). The timeouts in IPL decreases when the network is fully loaded. Also timeouts decrease with increase in number of hops all due to the fact that RTT has been increased.



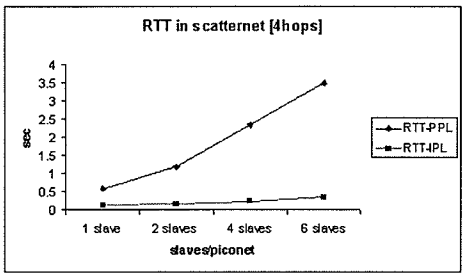
(a) TCP Goodput in piconet (2 hops)



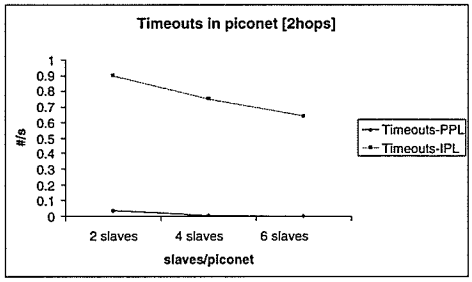
(b) TCP Goodput in Scatternet (4 hops)



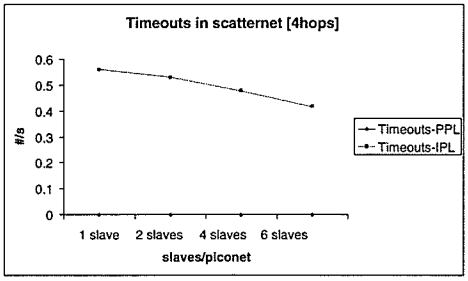
(c) RTT in piconet (2 hops)



(d) RTT in Scatternet (4 hops)



(e) Timeouts in piconet (2 hops)



(f) RTT in Scatternet (4 hops)

Figure 6.17: TCP NEW RENO performance in a piconet and scatternet

Chapter 7

Conclusions

7.1 Summary

The performance of TCP traffic over a Bluetooth network (both piconet and scatternet) was modeled and analyzed in both error free and error-prone environments. The E-limited scheduling scheme was used as the intra-piconet scheduling scheme because of inherent fairness with QoS in spite of its being simple to implement. For inter-piconet scheduling, a walk-in bridge scheme was used because of its low computational overhead. The impact of token buffer, token rate, scheduling parameter, outgoing baseband buffer on goodput, RTT, congestion window size, fast retransmits and timeout rate were investigated.

Both in piconet and scatternet [PPL] buffer size S = 25 gives the maximum goodput. Scheduling parameter in E-limited policy confined to M = 5 baseband packets (number of baseband packets for 1 TCP segment) and the token rate not higher than 50% of the maximum goodput dedicated to a slave yields adequate performance.

In IPL, network performance was found to deteriorate drastically with the increase in number of hops. Also it was found that, the error-affected packets are mostly retransmitted only by means of timeout.

7.2 Future Work

Future investigations on this research can be extended as follows

ARQ factor In this research, ARQ scheme was not used and the transmission of missing packets lost due to buffer overflows or interference in the wireless medium was taken care by the transport layer. Thus, the impact of ARQ scheme when TCP traffic is carried over Bluetooth network can be explored.

Data and voice traffic together Bluetooth has the ability to transfer both voice and data traffic through SCO and ACL links. However, this research focuses only on data traffic. Thus, the performance of data traffic in the presence of voice traffic is another area that can be performed in the future.

Master/Slave Bridge The bridging device can be either a Master/Slave bridge or Slave/Slave bridge. In this research, performance analysis was carried out using a Slave/Slave bridge. Therefore, in future the performance analysis can be carried out using a Master/Slave bridge.

Appendix A

Acronyms

ACK Acknowledge

ACL Asynchronous Connectionless

ARQ Automatic Repeat-reQuest

BER Bit Error Rate

BF Best Fit

CRC Cyclic Redundancy Check

CWND Congestion Window

DH (packet) Data High rate

DM (packet) Data Medium rate

FEC Forward Error Correction

FEP Fair Exhaustive Polling

FHSS Frequency Hopping Spread Spectrum

IPL Imperfect Physical Layer

L2CAP Logical Link Control and Adaptation Protocol

LCS Locally Coordinated Scatternet

LDQ Longest Downlink Queue

LWRR Limited and Weighted Round Robin

MAC Medium Access Control

MP Maximum Priority

MS (bridge) Master/Slave bridge

MSS Maximum Segment Size (TCP/IP)

OSI Open System Interconnection

OSU Optimum Slot Utilization

PCSS Pseudo-random Coordinated Scheduling Scheme

PDU Protocol Data Unit

PER Packet Error Rate

PHY Physical Layer

P-m-S-n Piconet m Slave n

PPL Perfect Physical Layer

QoS Quality of Service

RF Radio Frequency

RTO Retransmission Timeout

RTT Round-Trip Time

RWND Receiver Window

SACK Selective Acknowledgment

SAR Segmentation And Reassembly

SCO Synchronous Connection-Oriented Link

SIG Special Interest Group

SLQ Stochastically Largest Queue

SS (bridge) Slave/Slave bridge

TCP/IP Transmission Control Protocol/Internet Protocol

UDP User Datagram Protocol

WPAN Wireless Personal Area Network

Bibliography

[1] Bluetooth SIG, *Specification of the Bluetooth System - Core System Package [Host volume]*, Version 1.2, Nov. 2003, vol. 3. [Online]. Available: http://bluetooth.org/foundry/adopters/document/Bluetooth_Core_Specification_v1.2

[2] ——, *Specification of the Bluetooth System - Architecture & Terminology Overview*, Version 1.2, Nov. 2003, vol. 1.

[3] "Wireless PAN Medium Access Control MAC and Physical Layer PHY Specification," IEEE, New York, NY, IEEE standard 802.15, 2002.

[4] A. Capone, R. Kapoor, and M. Gerla, "Efficient Polling Schemes for Bluetooth Picocells," in *Proceedings of IEEE International Conference on Communications ICC 2001*, vol. 7, Helsinki, Finland, June 2001, pp. 1990–1994.

[5] A. Das, A. Ghose, A. Razdan, H. Saran, and R. Shorey, "Enhancing Performance of Asynchronous Data Traffic over the Bluetooth Wireless Ad-hoc Network," in *Proceedings Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies IEEE INFOCOM 2001*, vol. 1, Anchorage, AK, Apr. 2001, pp. 591–600.

[6] N. Johansson, U. Körner, and P. Johansson, "Performance Evaluation of Scheduling Algorithms for Bluetooth," in *Proceedings of BC'99 IFIP TC 6 Fifth International Conference on Broadband Communications*, Hong Kong, Nov. 1999, pp. 139–150.

[7] M. Kalia, D. Bansal, and R. Shorey, "MAC Scheduling and SAR Policies for Bluetooth: A Master Driven TDD Pico-Cellular Wireless System," in *Proceedings Sixth IEEE International Workshop on Mobile Multimedia Communications (MOMUC'99)*, San Diego, CA, Nov. 1999, pp. 384–388.

[8] Bluetooth SIG, *Specification of the Bluetooth System*, Version 1.1, Feb. 2001. [Online]. Available: https://www.bluetooth.org/spec/spec1_1_request.php

[9] J. Mišić, K. L. Chan, and V. B. Mišić, "TCP Traffic in Bluetooth 1.2: Performance and Dimensioning of Flow Control," in *Proceedings of IEEE Wireless Communications and Networking Conference WCNC 2005*, New Orleans, LA, Mar. 2005.

[10] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," Internet Engineering Task Force (IETF), draft Internet standard RFC 2581, Apr. 1999.

[11] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," Internet Engineering Task Force (IETF), draft Internet standard RFC 3782, Apr. 2004.

[12] RSoft Design, Inc., *Artifex v.4.4.2*, San Jose, CA, 2003.

[13] H. Zimmermann, "OSI reference model: The ISO model of architecture for open systems interconnection," pp. 2–9, 1988.

[14] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet.* Boston, MA: Addison-Wesley Longman, 2000, ch. 3.

[15] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," Internet Engineering Task Force (IETF), draft Internet standard RFC 2988, Nov. 2000.

[16] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[17] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," Internet Engineering Task Force (IETF), draft Internet standard RFC 2018, Oct. 1996.

[18] C. de Morais Cordeiro, D. Sadok, and D. P. Agrawal, "Piconet Interference Modeling and Performance Evaluation of Bluetooth MAC protocol," in *GLOBECOM 2001 - IEEE Global Telecommunications Conference*, San Antonio, TX, Nov. 2001, pp. 2870–2874.

[19] N. Golmie, R. E. V. Dyck, A. Soltanian, A. Tonnerre, and O. Rébala, "Interference Evaluation of Bluetooth and IEEE 802.11b Systems," *Wireless Networks archive*, vol. 9, no. 3, pp. 201–211, 2003.

[20] J. Mišić, K. L. Chan, and V. B. Mišić, "Performance of Bluetooth piconets under

E-limited scheduling," Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, Tech. report TR 03/03, May 2003.

[21] H. Takagi, *Queueing Analysis.* Amsterdam, The Netherlands: North-Holland, 1991, vol. 1: Vacation and Priority Systems.

[22] Z. Liu, P. Nain, and D. Towsley, "On Optimal Polling Policies," *questa*, vol. 11, no. 1–2, pp. 59–83, 1992.

[23] Y.-Z. Lee, R. Kapoor, and M. Gerla, "An Efficient and Fair Polling Scheme for Bluetooth," in *Proceedings MILCOM 2002*, vol. 2, 2002, pp. 1062–1068.

[24] S. Baatz, M. Frank, C. Kühl, P. Martini, and C. Scholz, "Adaptive Scatternet Support for Bluetooth using Sniff Mode," in *Proceedings of the 26th Annual Conference on Local Computer Networks LCN 2001*, Tampa, FL, Nov. 2001.

[25] ——, "Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communication Societies (IEEE INFOCOM 2002)*, New York, June 2002, pp. 782–790.

[26] A. Rácz, G. Miklós, F. Kubinszky, and A. Valkó, "A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets," in *Proceedings 2001 ACM International Symposium on Mobile ad hoc networking & computing*, Long Beach, CA, Oct. 2001, pp. 193–203.

[27] G. Tan and J. Guttag, "A Locally Coordinated Scatternet Scheduling Algo-

rithm," in *Proceedings of the 26th Annual Conference on Local Computer Networks LCN 2002*, Tampa, FL, Nov. 2002, pp. 293–303.

[28] P. Johansson, R. Kapoor, M. Kazantzidis, and M. Gerla, "Rendezvous Scheduling in Bluetooth Scatternets," in *Proceedings of IEEE Internationl Conference on Communications (ICC 2002)*, New York, Apr. 2002, pp. 318–324.

[29] V. B. Mišić, J. Mišić, and K. L. Chan, "Walk-In Bridge Scheduling in Bluetooth Scatternets," *Cluster Computing*, vol. 8, no. 2–3, pp. 197–210, July 2005.

[30] J. Kim, Y. Lim, Y. Kim, and J. S. Ma, "An adaptive segmentation scheme for the Bluetooth-based wireless channel," in *Proceedings Tenth International Conference on Computer Communications and Networks*, Scottsdale, AZ, Oct. 2001, pp. 440–445.

[31] N. Johansson, M. Kihl, and U. Körner, "TCP/IP over the Bluetooth Wireless Ad-hoc Network," in *NETWORKING 2000 Broadband Communications, High Performance Networking, and Performance of Communication Networks.* Paris, France: Springer-Verlag, May 2000, pp. 799–810.

[32] J. Postel, "Transmission Control Protocol," Internet Engineering Task Force (IETF), draft Internet standard RFC 793, Sept. 1981.

[33] R. Braden, "Requirements for Internet Hosts-Communication layers," Internet Engineering Task Force (IETF), draft Internet standard RFC 1122, Oct. 1989.

[34] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988, pp. 314–329.

[35] L.-J. Chen, R. Kapoor, M. Y. Sanadidi, and M. Gerla, "Enhancing Bluetooth TCP Throughput via Link Layer Packet Adaptation," in *IEEE International Conference on Communications (ICC)*, Paris, France, 2004.