1-1-2007

Bluehoc-based simulation study of user data throughput in Bluetooth-enabled devices

Syed Rahat
Ryerson University

Bluehoc-based Simulation Study of User Data Throughput in Bluetooth-enabled Devices

by

Syed Rahat

A project
presented to Ryerson University
in partial fulfillment of the
requirements for the degree of

Master of Engineering

in the department of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2007

Author's Declaration

I hereby declare that I am the sole author of this project report.

I authorize Ryerson University to lend this project report to other institutions or individuals for the purpose of scholarly research.

_____ _____

Syed Rahat

I further authorize Ryerson University to reproduce this project report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research

_____ _____

Syed Rahat

Borrow List

Ryerson University requires the signatures of all persons using or photocopying this project report.

Please sign below, and give address and date.

Acknowledgement

I would like to thank my advisor Dr. Anpalagan for his continued guidance and support throughout my Masters program. He has been a great mentor, and an excellent role model in research.

I am also thankful to my parents and wife for their understanding, support and encouragement during my Masters program.

Abstract

Bluetooth technology aims at allowing short-range communication between portable and/or fixed devices. It uses short-range radio links to replace cables between Bluetooth-enabled devices. In this way, it is similar in purpose to the Infrared Data Association (IrDA), however, Bluetooth is a radio frequency (RF) technology utilizing the unlicensed 2.5 GHz industrial, scientific, and medical (ISM) band. The key features of Bluetooth technology are robustness, low power and low cost with its primary market for data and voice transfer between communication devices and PCs.

In this project, a simulation study is done with three major goals in mind: (i) to gather expertise on and evaluate a Bluetooth simulation tool called *Bluehoc* for further use, (ii) to gather measurements of some Bluetooth characteristics such as throughput in post connection state and (iii) to describe a model that can be used to get maximum throughput for voice and data applications. We also review some of the key aspects in Bluetooth simulation and present models of the Bluetooth devices to get maximum throughput. We show that user data transfer rate (throughput) between Bluetooth master and slaves is effected by distance, number of slaves and slave's start time.

TABLE OF CONTENTS

List of Figures

List of Tables

Glossary

ACL	Asynchronous Connectionless Link. One of the two types of data links defined for the Bluetooth Systems.
ACO	Authenticated Ciphering Offset.
Active Mode	In the active mode, the Bluetooth unit actively participates on the channel.
AM_ADDR	Active Member Address. It is a 3 bit number. It is only valid as long as the slave is active on the channel.
ARQN	Automatic Repeat reQuest Number is used as a 1-bit acknowledge indication to inform the source of a successful transfer of payload data with CRC.
Baseband	The baseband describes the specifications of the digital signal processing part of the hardware.
BD_ADDR	Bluetooth Device Address. Each Bluetooth transceiver is allocated a unique 48-bit device address.
Bluetooth	An open specification for wireless communication of data and voice.
Bluetooth clock	Every Bluetooth unit has an internal system clock which determines the timing and hopping of the transceiver.
Channel	A logical connection on the L2CAP level between two devices serving a single application or higher layer protocol.
CLK	Clock, typically the master device clock which defines the timing used in the piconet.
CLKE	Clock Estimate, a slave's estimate of the master's clock, used to synchronize the slave device to the master.
CLKN	Clock Native, the clock of the current Bluetooth Device.
CO	Connection-oriented.
DCID	Destination Channel Identifier,
DH	Data-High Rate. An ACL link data packet type for high rate data.
DLCI	Data Link Connection Identifier. This is a 6-bit value representing an ongoing connection between a client and a server application.
DM	Data - Medium Rate. An ACL link data packet type for medium rate data.
Page Scan State	A mode where a device listens for page trains containing its own device access code (DAC).

Payload format	Each packet payload can have one of 2 possible fields, the data field (ACL) or the voice field (SCO).
Piconet	A collection of devices connected via Bluetooth technology in an ad hoc fashion. A piconet starts with two connected devices, such as a portable PC and cellular phone, and may grow to eight connected devices.
Profile	A description of the operation of a device or application.
QoS	Quality of Service.
RFCOMM	Serial Cable Emulation Protocol based on ETSI TS 07.10.
Scatternet	Multiple independent and non-synchronized piconets form a scatternet.
SCO	Synchronous Connection Oriented link. One of the 2 Bluetooth data link types defined.
SIG	Special Interest Group. The Bluetooth SIG is located at www.bluetooth.com.
Slave device	A device in a piconet that is not the master.There can be many slaves per piconet.
Time slot	A single time slot in the Bluetooth system lasts 625us. It can be though of as the time it takes to send one packet from one Bluetooth device to another

1 Introduction

Over the years Bluetooth technology has grown as a solution for a short-range wireless ad-hoc networking. The key benefits of Bluetooth are low cost, robustness and low power consumption. Bluetooth operates in the free spectrum available from Industrial, Scientific and Medical (ISM) [1] band at 2.4 GHz. It uses a frequency hop spread spectrum, which changes the transmission frequency 1600 times per second in a pseudo random way. The range of the Bluetooth radio is approximately 10 meters with a standard Bluetooth device. The range can be extended to 100 meters by increasing the output power.

The Bluetooth specification defines how Bluetooth devices will group themselves for the purposes of communication. A Bluetooth Wireless Personal Area Network (BT-WPAN) consists of piconet. Each piconet is a cluster of up to eight Bluetooth devices. One device is designated as the master, and the others are the slaves. Two piconets can be connected through a common Bluetooth device (a gateway or bridge) to form a scatternet [2].

This project was inspired and started to study the post connection characteristics of Bluetooth piconet. Bluetooth communication is not based on distributed contention resolution, as in traditional wireless LANs, but on a time division duplex (TDD) master-slave mechanism. A Bluetooth piconet consists of one master and up to seven slaves. The master allocates transmission time slots, each of which takes 625 microseconds, to the slaves in the piconet. The master and slaves use alternate transmission slots, with each odd slot being used only by the slave to which the master sent a frame in the previous even transmission slot [3].

Some researchers think that the use of Bluetooth devices is to create some kind of sensor networks or a small network that will collect small amount of data from slave nodes and transmit it to master node. In fact, Bluetooth has many more applications, for example, Bluetooth headsets, Bluetooth printers and even Bluetooth phones with SIM for Extensible Authentication Protocol Method for GSM Subscriber Identity (EAP-SIM) authentication. Therefore, keeping in mind the above argument, this project is an effort to

study effects of different parameters on user data throughput once the connection has been established.

In this project report, simulation for Bluetooth technology along with its issues and challenges is presented. During this simulation study, extensive simulation is done to find parameters for best possible achievable user data throughput. Chapters 2 and 3 are for those readers of this report, who have no or very limited knowledge of Bluetooth technology, simulation using NS-2 [11] and *Bluehoc* [13]. In chapter 2, emphasis is on Bluetooth protocol stack, because, this will help readers of this report to have better understanding about protocols involved in Bluetooth communication before diving deeper into simulation models and results. In chapter 3, simulation in general, NS-2 and *Bluehoc* in particular are discussed. In chapter 4, not only simulation models but also a detail explanation of each result obtained from the simulation is presented. Followed by is the conclusion and future work in chapter 5.

2 Bluetooth Protocol Stack in-depth Review

2.1 The Bluetooth Core System

Bluetooth Protocol Stack is the core portion of the Bluetooth specification. It allows devices to locate, connect and exchange data, execute interoperable and interactive applications with each other. In this chapter, we present the major components of the Bluetooth protocol stack, highlighting the relationships among the various layers. It is important to understand Bluetooth protocol stack in some detail to understand simulation that has been done in this project. Most of the material in this chapter is taken from [4] and [5]. Author of this report strongly believes that these authors have explained Bluetooth technology in detail, therefore, there is no need to consult any more material.

2.1.1 The Protocol Stack Components

As can be seen from Figure 2-1, the high-level components of the Bluetooth protocol stack can be logically partitioned into three sets:

- The transport protocol set,
- The middleware protocol set, and
- The application set.

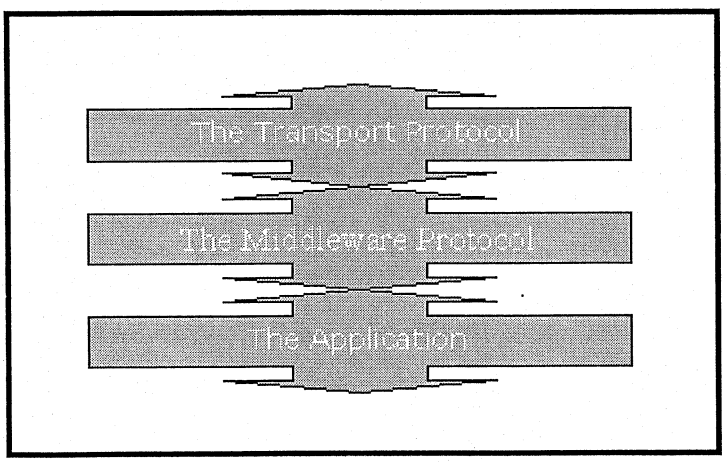


Figure: 2-1 A High Level View of the Bluetooth Stack

2.1.2 The Transport Protocol Set

This set is composed of the protocols designed to allow Bluetooth devices to locate each other and to create, configure and manage both physical and logical links that allow higher layer protocols and applications to pass data through these transport protocols. The protocols in this group include: radio, baseband, link manager, logical link and adaptation and the host controller interface.

Figure 2-2 shows the arrangement of the protocols in the transport set. These protocols are proposed by the Special Interest Group (SIG) to carry audio and data traffic between Bluetooth devices. We will take "top-down" order approach in this chapter to discus these protocols, because, it makes more sense to start from top, that is, from where traffic enters in this protocol set and go down step by step to lower level protocols.

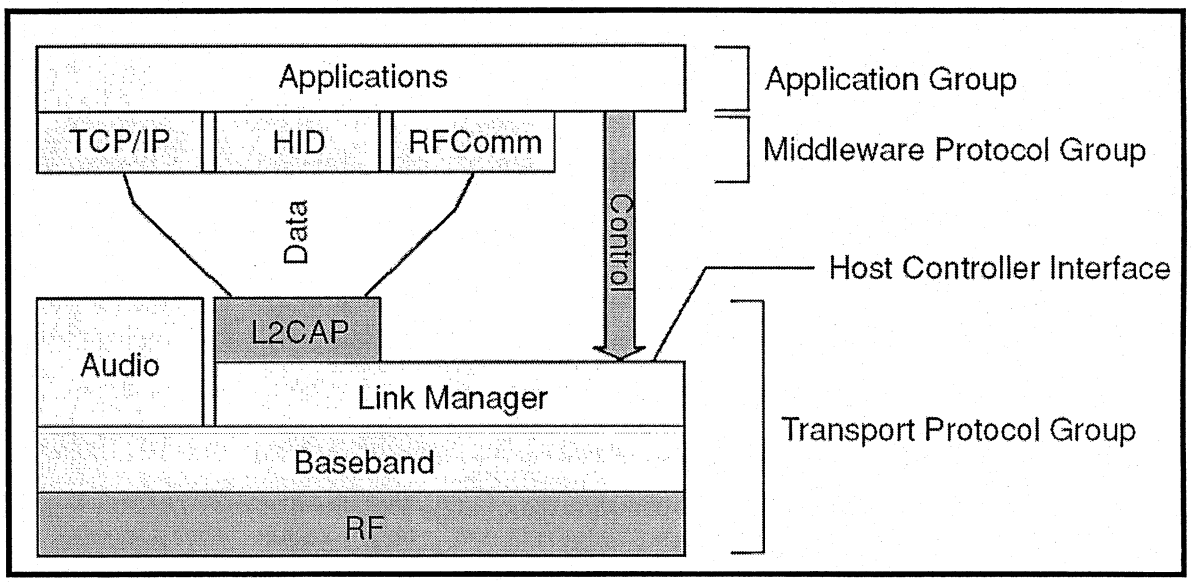


Figure: 2-2 The Transport Protocol Set (Adapted from [3])

Transport protocols support both asynchronous transmissions, for data communications, and synchronous (or periodic) transmissions, for voice communications. Quality of Service (QoS) is defined as, the traffic-flow specification for the local device's traffic (outgoing direction) [5]. In Bluetooth, factors which define application layer's QoS are: bandwidth, latency and error rate, but they are not contained in one location in protocol stack [6].

Voice traffic has higher priority over data traffic to maintain the high QoS expected for voice applications. In fact, voice traffic bypasses all of the intermediary protocol layers and is tunneled directly from the voice application to the baseband layer, which after converting it into small packets transmits it directly over the Bluetooth air-interface.

2.1.2.1 The L2CAP Layer

Logical link control and adaptation protocol (L2CAP) layer acts as buffer between higher-layer protocols and applications. Thus, higher layers neither need to be aware of the things like frequency hopping at the radio and baseband level nor the specific packet formats used for transmission over the Bluetooth air-interface.

L2CAP is responsible for:

- Protocol multiplexing, which allows multiple protocols and applications to share the same air-interface.
- Segmentation of large packets used by higher layers into smaller packets for baseband transmission and the corresponding reassembly of those packets by the receiving device.
- Maintenance of the desired grade of service by negotiating an acceptable level of service. Based on the requested level of service, an L2CAP layer implementation may then exercise admission control for new incoming traffic and coordinate with lower layers to maintain the desired level of service.

2.1.2.2 The Link Manager Layer (LMP)

Link managers, in Bluetooth devices, negotiate bandwidth allocation for desired grade of service for data and bandwidth reservation to support voice traffic. They supervise device pairing and encryption of the data flowing over the air-interface between the devices, whenever needed. Bluetooth link managers use a challenge-response method to authenticate devices. If authentication fails, the link managers prohibit any communication between the devices. Another, function performed by LMP is power control by negotiating low activity baseband modes of operation. Link managers may request adjustments to the transmission power level for further power conservation.

2.1.2.3 The Baseband and Radio Layers

This layer is responsible for instantiating Bluetooth air-interface and also facilitating device discovery and connection. The baseband layer, defines, master and slave roles for the devices. The device that initiates connection process becomes the master of the link, while the other device becomes a slave. The baseband layer also defines how the frequency hopping sequences used by communicating devices are formed. It also defines the rules for sharing the air-interface among several devices based upon a time division duplex, packet-based polling scheme. For example, in synchronous transmissions, the master transmits to and/or polls a slave device periodically. Other functions performed by the baseband layer are to define the various packet types supported for synchronous and asynchronous traffic, as well as error detection and correction, signal whitening encryption, packet transmission and retransmissions. It is also important to note that the concept of master and slave devices ends at this layer, at the layers above it there is no special provisions for different actions in a master or a slave device.

2.1.2.4 HCI Layer

Any Bluetooth device that contains L2CAP layer and any appropriate portions of the higher layers of the Bluetooth stack can attach to another Bluetooth device called host, via some physical interface, called the host transport. To achieve interoperability of the Bluetooth devices made by different vendors, the specification defines a common interface for accessing the lower layers of the stack that reside in the device. The host controller interface (HCI) allows higher layers of the stack to access the baseband, link manager and other hardware registers through a single standard interface.

The main purpose of HCI layer is interoperability among host devices and Bluetooth modules, either of which could come from a variety of vendors. It is important to note that product implementations not necessarily need to comply with the HCI specification to support a fully compliant Bluetooth air-interface. HCI also controls, whether the device may enter in certain modes of operations in which an authentication operation or a device paging state may be performed. The control path shown in Figure 2-2 is not explicitly described in the specification, but it is interwoven among the various

protocols in the stack. Nevertheless, the HCI specification includes the bulk of the information that control path carries.

2.1.3 The Middleware Protocol Set

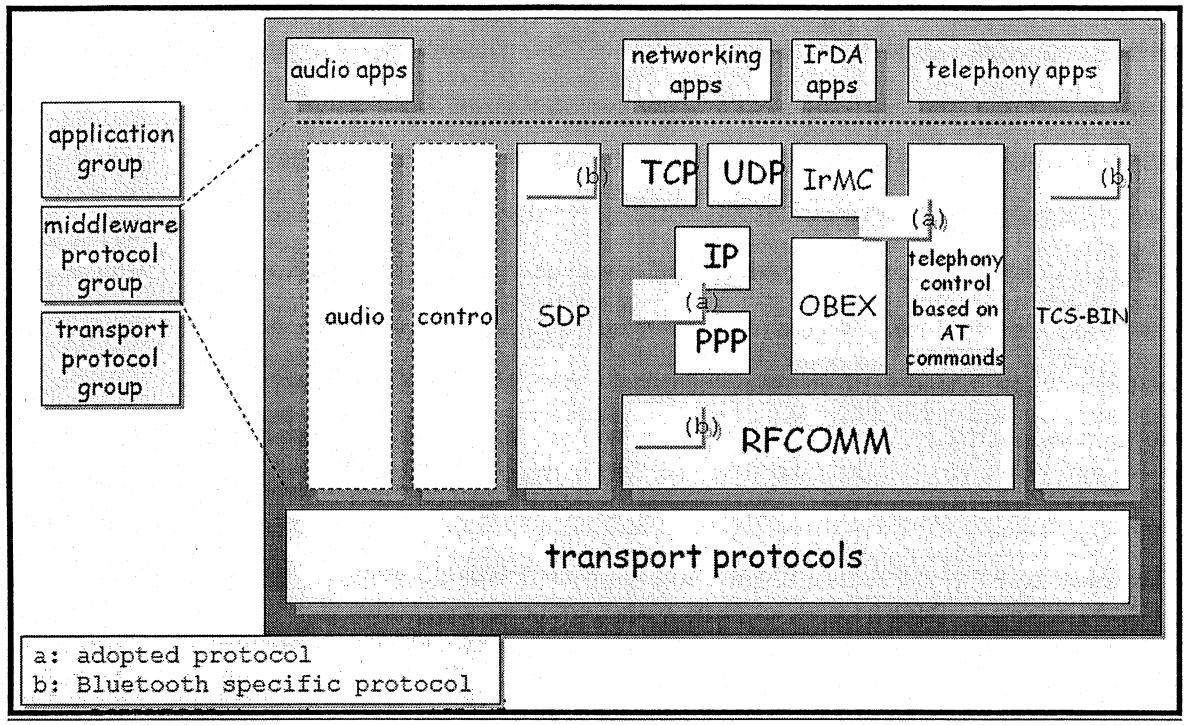


Figure: 2-3 The Middleware Protocol Set (Adapted from [5])

The middleware protocols consist of:

- RFCOMM, a serial port abstraction,
- Service discovery protocol (SDP), used to describe available services and to locate needed services,
- A set of IrDA (abbreviation of Infrared Data Association) interoperability protocols adopted from the IrDA standard that enables interoperable use of IrDA-enabled applications, and
- A telephony control protocol (TCS), used for controlling telephone calls that might be used either for audio or for data.

2.1.3.1 RFCOMM Layer

To provide serial communications over Bluetooth wireless links, the protocol stack defines a serial port abstraction called RFCOMM (abbreviation of Radio Frequency

Communication). RFCOMM presents a virtual serial port to applications. An application can use RFCOMM virtual serial port in same manner as a standard serial port to accomplish: synchronization, dial-up networking and others functions without significant changes to the application. Thus the intent of the RFCOMM protocol is to enable legacy, serial port-based applications to use Bluetooth transports.

2.1.3.2 SDP Layer

In traditional networks, such as Ethernet LANs, services such as file serving, print serving are provided by servers and used by clients. In many cases, the clients locate these network services through some static configuration. In dynamic ad-hoc networks that can be built by Bluetooth wireless communications, this sort of static configuration is not enough. Any two or more devices might start communicating over Bluetooth links at any time and need to know what services are offered by each device, thus, a more dynamic protocol is required for Bluetooth. Service discovery protocol (SDP) addresses above-mentioned requirement. SDP defines a standard method for Bluetooth devices to discover and find out more about the services offered by other devices.

2.1.3.3 IrDA Interoperability Protocols

It is important to state that OBEX (abbreviation of OBject EXchange) is the fundamental building block upon which the usage models of file transfer (object exchange) and object push are built. Among the applications in which OBEX can be used is the exchange of well-defined objects, such as, electronic business cards (vCard format), e-mail or other messages (vMessage format), calendar entries (vCal format) and others can all be exchanged using the OBEX protocol.

2.1.3.4 Networking Layers

Bluetooth networks are different from traditional networks in the sense that Bluetooth uses a peer-to-peer network topology rather than a LAN style topology. However, Bluetooth allows devices to connect to larger networks through a dial-up connection or via a network access point.

Dial-up networking uses middleware protocol stack to establish a connection to a network. If the network being accessed is one that uses the Internet Protocol (IP), then once a dial-up connection to an IP network is established, standard Internet protocols

(such as TCP, UDP, HTTP) can be used by the device. A Bluetooth device might also connect to an IP network via a network access point.

2.1.3.5 TCS Layer and Audio

Bluetooth telephony control specification (TCS) layer is designed to support telephony functions, including call control and group management. TCS is also used to set up the call parameters. Once a call is established, a Bluetooth audio channel can carry the call's voice content.

The telephony control specification binary (TCS-BIN) protocol includes call control functions, group management functions and a method for devices to exchange call signaling information without actually placing a call or having a call connection established. Bluetooth audio communication takes place at a rate of 64 kbps using one of two data encoding schemes: 8-bit logarithmic pulse code modulation (PCM) or continuous variable slope delta (CVSD) modulation. Compression techniques called *A-law* and *μ-law* are applied for PCM audio.

2.1.4 The Application Set

The application set in this report refers to software that resides above the protocol stack defined by the SIG. This set consists of the applications that make use of Bluetooth wireless links. The applications which were not developed for Bluetooth in mind, such as a modem dialer application are called legacy applications, but the applications which were developed with Bluetooth in mind such as, applications that use the telephony control protocol for controlling telephony equipment are called new applications.

We think that the most useful applications are those applications which instantiate the Bluetooth profiles, because, profile defines the interoperability. The SIG defines only the application programming interfaces (APIs). However, look and feel of the Bluetooth applications is not defined in any specifications, therefore, software developers have enough freedom to add features and user interfaces, without violating the interoperability guidelines spelled out in the profiles.

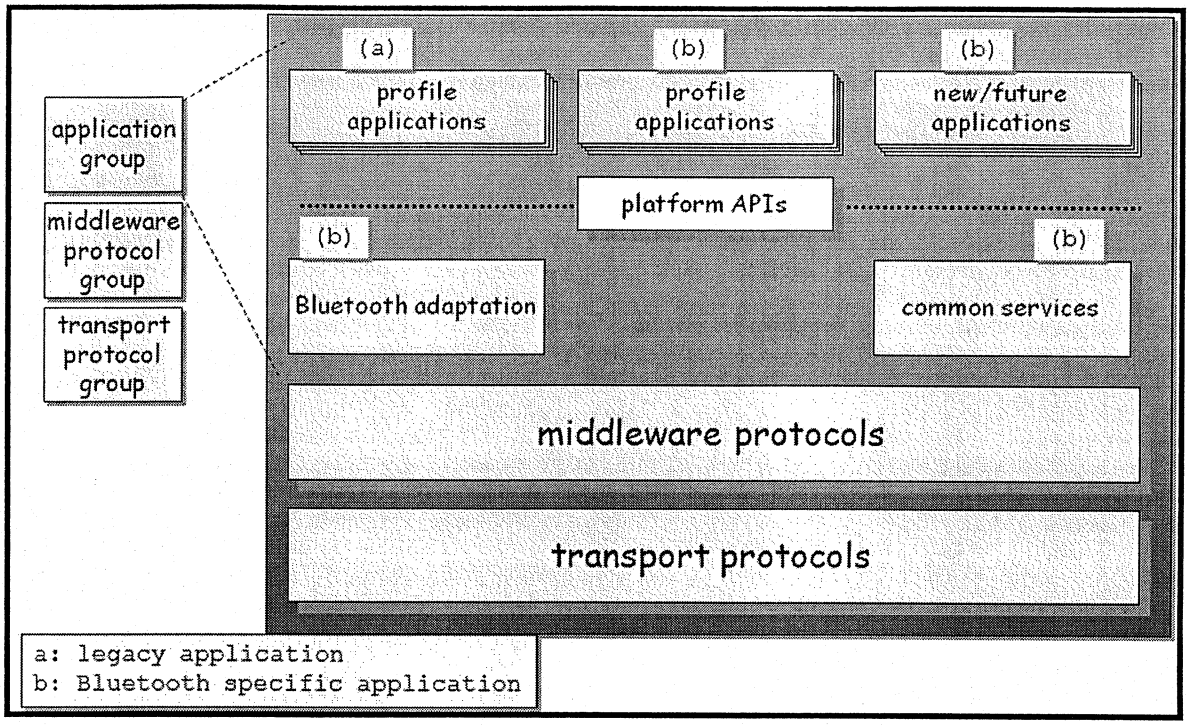


Figure: 2-4 General View of the Application Group (Adapted from [5])

2.1.4.1 Legacy Applications

If an application was designed and built with transport other than Bluetooth wireless communication in mind then by using SIG defined layers of the protocol stack, such as the IrDA interoperability and RFCOMM layers, we can mold these applications to function properly on Bluetooth links. Examples include, existing IrDA applications for object exchange or synchronization and dial-up networking applications.

2.1.4.2 New Applications

If there is no existing application, then it is always a good choice to develop Bluetooth application that includes application features that exploit unique capabilities in the protocol stack. As a good practice it is advantageous to develop common services such as security services, connection management services, service discovery protocol (SDP) services, and so on. Refer to [5] for details on this topic.

SIG has chosen not to specify Linux® APIs, Windows® APIs, Symbian™ APIs or any other APIs, instead the profiles define the necessary function to enable platform

experts to develop appropriate APIs for use with Bluetooth applications on relevant platforms.

In this chapter we have studied that Bluetooth has layer based core system architecture. The lowest layer in the Bluetooth protocol stack is the RF (physical layer), which is responsible for transporting information between two Bluetooth-enabled devices connected by a single physical link. The Baseband layer defines the process by which devices discover each other and establish connections for data transfer. It also defines the master and slave roles for devices. The Link Manager Protocol, manages communication security and bandwidth negotiation, it also reserves a certain Quality of Service parameters for data traffic. Data traffic is routed through the Logical Link Control and Adaptation Protocol, which provides segmentation and reassembly of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. Having studied this much of core Bluetooth protocol stack, a reader with moderate previous knowledge of Bluetooth core specifications should be ready to study our next chapter about *Bluehoc* simulator.

3 Simulation using NS-2 Bluehoc

We are excited by the opportunity that presented itself due to little investigation on the parameters that impact user data throughput in post connection state in Bluetooth piconet. Researchers still have no firm opinion on, whether or not, other radio devices operating in same industrial, scientific and medical (ISM) frequency band as Bluetooth, such as microwave oven, can impact Bluetooth data throughput. Some argue they can [9], while others think they have no significant impact [10]. There are countless opportunities that can be availed to do further research on similar topics in Bluetooth networks. To do further research, one can either use expensive hardware or can utilize simulation tools available. Obviously, simulation is the most cost effective way to do research in the early stage of any scientific study. But, most of the simulation tools need to be evaluated by someone other than their designers to make sure, they can be reliably used to gather data that can answer questions like above. Therefore, a simulation study is done with following major goals in mind:

1. To gather expertise on and evaluate a Bluetooth simulation tool called *Bluehoc*, for further use,
2. To gather measurements of some Bluetooth characteristics such as throughput in post connection state, and
3. To describe a model that can be used to get maximum throughput for voice and data applications.

No matter how big or small a task is, one cannot finish it without proper knowledge and tools of trade, same is true for simulation. To describe a model that can be used to get maximum throughput for voice and data applications, we selected, *Bluehoc* as our simulator, for this project. This tool is an extension of NS-2, therefore, it is imperative to gather knowledge about NS-2 first and then study the specifics of *Bluehoc* simulator. Therefore, this chapter can be divided into two parts, in the first part, we will give brief introduction of NS-2, afterwards, in the second part, we will explain in detail what *Bluehoc* can do.

3.1 NS-2 Network Simulator

NS-2 network simulator is an event driven network simulator developed at UC Berkeley. It simulates variety of IP networks. It can perform simulation both for wired and wireless world.

3.1.1 Components of NS-2

In a simplified form, as shown in Figure 3-1, NS-2 network simulator is an object-oriented Tcl (OTcl) script interpreter. It has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries.

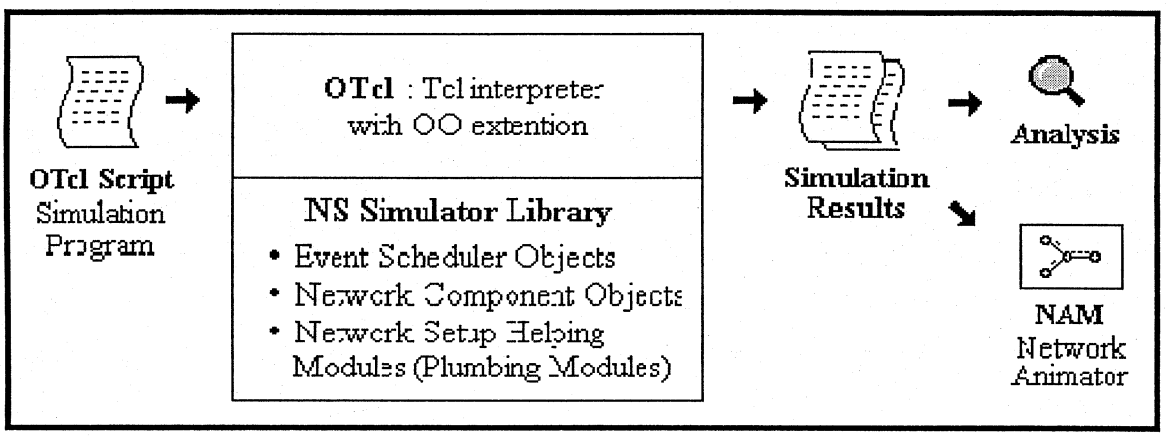


Figure: 3-1 Simplified User's View of NS-2 (Adapted from [11])

3.1.2 How does NS-2 Network Simulator work?

In nutshell, to use NS-2 network simulator, a program in Tcl script language is required. Tcl script initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and also tells traffic sources when to start and stop transmitting packets through the event scheduler.

3.1.3 Summary of the Features Supported by NS-2 for Wireless Networks

Following is the list of some of the features support by NS-2 for wireless simulation.

- Complete implementation of the IEEE 802.15.4, IEEE 802.11e, IEEE 802.16, IEEE 802.11 DCF MAC protocol.
- Complete implementation of the Address Resolution Protocol (ARP).
- Implementations of the following multi-hop ad hoc network routing protocols:
    - Dynamic Source Routing(DSR).

- Destination Sequenced Distance Vector (DSDV).
- Temporally Ordered Routing Algorithm (TORA).
- Ad hoc On-demand Distance Vector (AODV).

- Wireless network interface modeling the Lucent Wave LAN DSSS radio.
- Two Ray Ground Reflection radio propagation model.

3.2 Bluehoc Simulator

Bluehoc simulator was developed and distributed by IBM under Public License (IPL). It provides a Bluetooth extension for NS-2 (ver. 2.1b6). It was developed to simulate Bluetooth features, such as, Bluetooth baseband, Logical Link Control and Adaptation Protocol (L2CAP) and Link Manager Protocol (LMP).

Let us start with the description of a Bluetooth node (Tcl class, BTNode) which is one of the basic aggregate objects used to simulate a Bluetooth device in *Bluehoc*. As stated before, Bluetooth is an extension of NS-2, therefore, *ns-btnode.tcl* class, is derived from the Node class in NS-2 and inherits its members. The Bluetooth specific members are implemented in C++ and can be configured from the Tcl/Tk interface. Table 1 has list of all the added classes for *Bluehoc* in NS-2 [13].

3.2.1.1 C++ Classes Added to NS-2

Module	Function	Source Code
Baseband	Implements basic features of Bluetooth baseband	baseband.cc
BT_DRRScheduler	Deficit Round Robin (DRR) based scheduler	bt-drr.cc
LinkController	Stop and wait ARQ	bt-lc.cc
LBF	Leaky bucket filter	lbf.cc
LMP	Implements some LMP commands	lmp.cc
L2CAP	Basic L2CAP functionality	l2cap.cc
BTHost	Bluetooth host which is layered over L2CAP	bt-host.cc
BTClassifier	Classifies packets based on active mode address	bt-classify.cc

Table 1: C++ Classes added to NS-2 for Bluehoc

3.2.1.2 Command Syntax

The syntax of the command to create a BTNode is:

set btnode [new BTNode <ip_addr> <bd_addr> <0 or 1> <x co-odinate> <y-coordinate>]

Where, *bd_addr* is the Bluetooth device address, which is somewhat similar to Ethernet addresses of 48 bits, except that it is restricted to 32 bits. The second argument is 0 for creating a slave and 1 for creating a master. The *x* and *y* co-ordinates are specified in meters.

3.3 Device Discovery and Paging

When two Bluetooth devices are in range, they can discover and connect to each other by obtaining Bluetooth addresses and clocks of each other by "inquiry procedure" followed by "paging procedure". Device discovery is time consuming, because, the device initiating the inquiry does not know the frequency and time window in which other devices might be scanning. Paging delays, on the other hand are much lesser, because, during paging an approximate idea of the clock of the paged device (through inquiry) is available.

In the indoor wireless environment, where usually we have a lot of other wireless devices that use same ISM band as Bluetooth devices, chances of inquiry and paging exchanges could be lost over the air are high. In this case, both device discovery and paging delays can be very high and unpredictable. Moreover, multiple devices doing inquiry at the same time can make device discovery delays even more unpredictable. These delays are important for the applications, which depend on making fast ad-hoc connections. In this project these delays are also studied.

In the *Bluehoc*, inquiry and paging procedures are simulated almost exactly as in the Bluetooth specifications. In the start of simulation, a master node and many slave nodes are created. The master node starts inquiry procedure.

The BThost at the master node sends an *HCI_Inquiry* command to the baseband. In real world HCI commands are packets, which are transported over a physical bus, in *Bluehoc*, they have been simulated as function calls. *Bluehoc* uses two parameters in the *HCI_inquiry* command, number of responses required and inquiry timeout. In response to

the *HCI_Inquiry* command, the baseband starts sending inquiry packets with the general access code. The transmission frequency depends upon the clock of the master and the general inquiry access code. The slaves respond to inquiry with frequency hopping synchronization (FHS) packets that contain the device address and the clock of the slave.

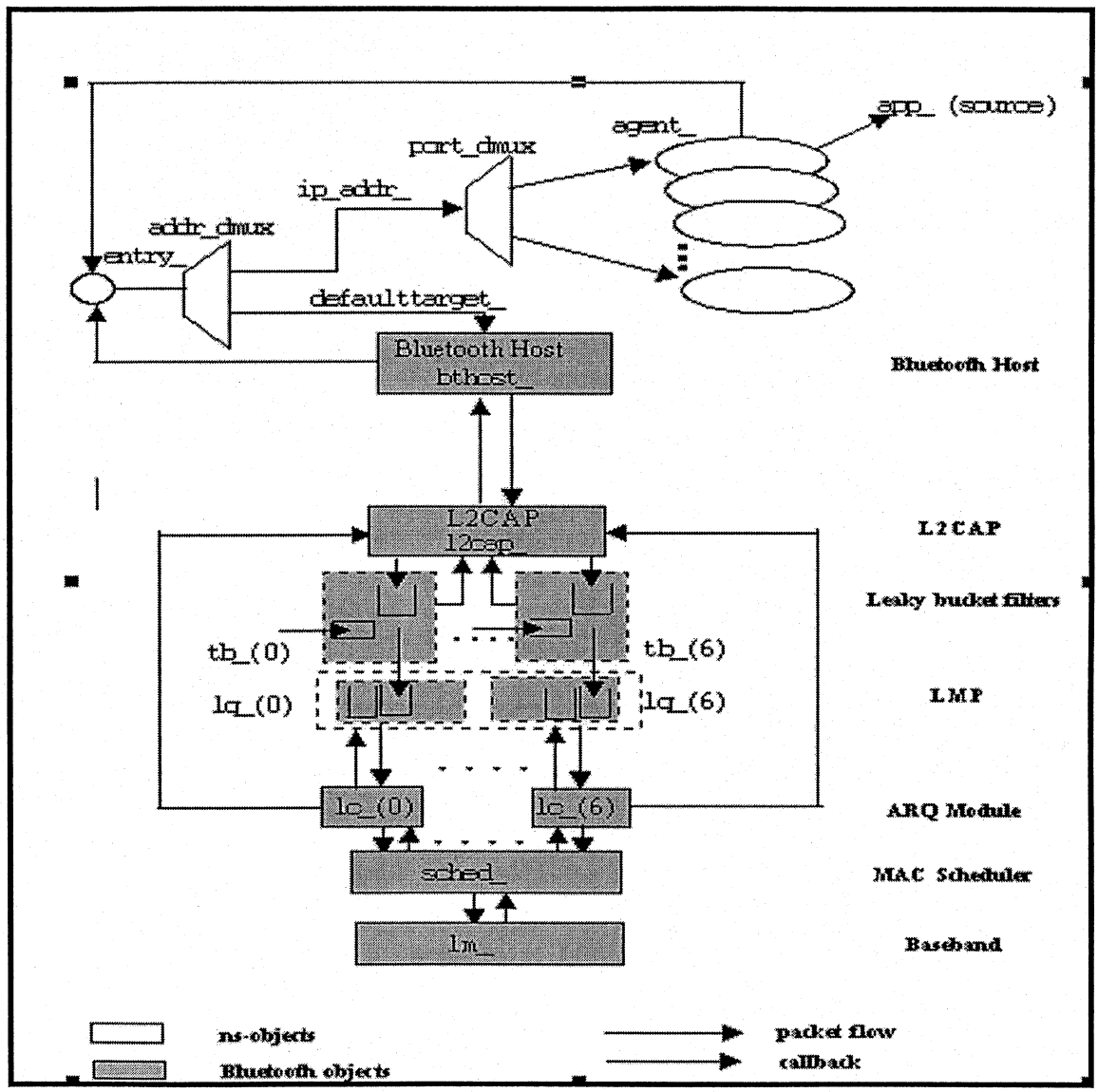


Figure: 3-2 Structure of a Bluetooth node (Adapted from [14])

3.4 Connection Establishment in Bluehoc

Bluetooth core system protocols control signaling between layers, which are responsible for connection establishment. Also the Bluetooth specification enables interoperability between independent Bluetooth enabled devices by defining the protocol messages

exchanged between equivalent layers. Standard interactions are defined for all inter-device operation, where Bluetooth devices exchange protocol signaling according to the Bluetooth specification. Connection establishment related signaling has been simulated in *Bluehoc*, as shown in Figure 3.2and which is discussed below:

3.4.1 LMP Signaling

Link Manager Protocol (LMP) establishes an asynchronous connectionless (ACL) link between the master and the slave after paging is over and the slave is tuned to the master hopping sequence. To accomplish above, master's LMP sends a *LMP_host_connection_req* to the slave LMP. When the master's LMP receives a response from the slave LMP, it sends an HCI event to BTHost reporting status, that is, indicating that the connection with the required device has (or has not) been established. The master then sends a QoS setup command with the QoS parameters. QoS parameters depend on the application for which the connection is required. To keep things simple, it is assumed that there is only one flow per ACL connection.

3.4.2 L2CAP Connection Establishment

After the QoS negotiation is over, LMP sends *HCI_QoS_setup_complete_event* to BThost as shown in Figure 3.3. The host then sends *L2CA_Connect_Req* to L2CAP. Each L2CAP channel has a connection identifier (CID) associated with it. CID 1 is reserved for signaling. *L2CAP_connection_request* and *L2CAP_connection_response* are exchanged between initiating and responding devices respectively. Several L2CAP connections are possible over a single ACL connection but in *Bluehoc* this has not been implemented yet.

3.4.3 QoS Mapping

As stated before, prior to the connection establishment, *Bluehoc* negotiates QoS parameters. This is done by *mapQoS* in bt-drr.cc. The QoS is setup based on the appFlowSpec parameters in bt-host.cc. QoS mapping and negotiations are done only once per ACL connection. The QoS requirements for standard applications (like Telnet, FTP, packetized voice) are conveyed to the MAC (scheduler), and if impossible parameters are requested, the negotiation fails – meaning that these parameters cannot be used to over saturate a link [13]. If loss sensitivity is set, *Bluehoc* asks for, forward error correction

(FEC) encoding, if it is *yes*, it selects, Data-Medium Rate (DM) packets, otherwise, Data-High Rate (DH) packets. The packet type is selected on the basis of the packet size and whether DM or DH has been selected. Table 2 shows above mentioned relationship.

Packet Types	FEC	Packet	
		Size (bytes)	Length (slots)
DM1	Yes	18	1
DM3	Yes	123	3
DM5	Yes	226	5
DH1	No	28	1
DH3	No	185	3
DH5	No	341	5

Table 2: DH and DM Slot Length

3.4.4 Logical Link Control and Adaptation Protocol Simulation

It is important to know about simulated, Logical Link Control and Adaptation Protocol (L2CAP) layer, because, data (for ACL) will pass through this layer and will be used to calculate user data throughput. In *Bluehoc*, L2CAP has two main functionalities:

- Performs segmentation and reassembly (SAR) for higher layer packets.
- Implements *DATA_WRITE* primitive for transfer of higher layer packets over a logical link identified by a channel ID (CID).

3.4.5 Using TCP/IP Simulation of NS-2

NS-2 has very good support for TCP/IP simulations, which is used by *Bluehoc*. After installing *Bluehoc*, the proc.tcl file in ~/ns/run directory has a function called config-app which is responsible for configuring applications and transport layers for *Bluehoc*.

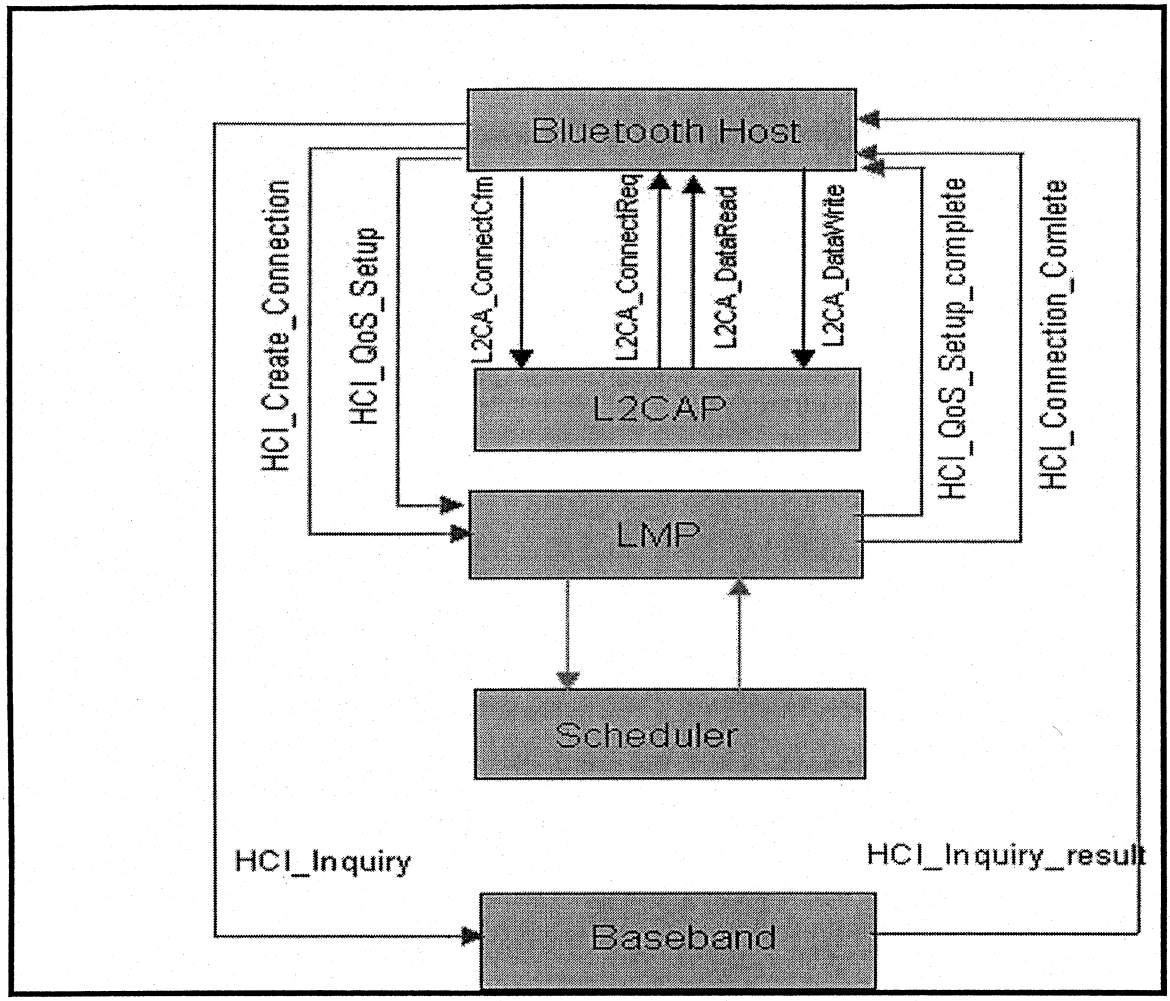


Figure: 3-3 Connections Establishment and QoS Related Message Exchange (Adapted from [13])

3.4.6 Tunable Parameters

Bluehoc has more than a few configuration settings for each of the maximum eight devices in a piconet. User has to be careful about supplying all the required parameters, otherwise, an error message will appear with not much explanation.

The minimum required values for master and slaves nodes are:

Master Node:

Parameter	Typical Value	Explanation
StartTime (sec)	0	Start time of inquiry for the master
NumResponses (sec)	Should be <= the number of slaves	Number of responses required from inquiry
InqTimeout (sec)	10.24	Inquiry timeout

Table 3: The Minimum Required Values for Master

Slave Node:

Parameter	Typical Value	Explanation
StartTime (sec)	< StartTime (sec) of master	Start time of inquiry for that particular slave
InqScanOffset (sec)	Between 0 and 2.56	Inquiry Scan Offset

Table 4: The Minimum Required Values for Slaves

3.4.7 Other Parameters:

Simulation time:

There are 5 options: 10, 20, 30, 40, 50 sec.

Applications:

One of the following applications needs to be selected for each link:

FTP, Telnet, Voice (uses Application/Traffic/Exponential traffic generator of NS-2)

3.5 Configuring Bluehoc Simulator

To run GUI for *Bluehoc* simulator, Tk windowing shell "wish" need to be installed. As soon as one runs "wish Bluehoc.tcl" from ~ns/run directory a window titled "Bluehoc configuration" will come up. One can use this GUI to configure *Bluehoc* simulator.

3.6 Trace / Log Support

This section is important, because, the information in this section will be used to explain results in the next chapter. There are two type traces in BTTrace file. Annex (d) has sample trace files.

3.6.1 Pre Connection Traces

Bluehoc generates these traces for 'hop' and 'node' events. These are responsible for showing packets traveling across links, changing node labels (which represent states) and changing node colors [13]. Bluetooth traces are dumps for each packet received during inquiry and paging procedures. The dumps also include successful reception of LMP messages and L2CAP signaling.

3.6.2 Post Connection Traces

After connection is established and Bluetooth device is in connection state the dumps are for the *L2CA_dataWrite* and *L2CA_dataRead.* Below line corresponds to the

L2CA_dataRead primitive and is printed each time L2CAP has a complete packet to send to the upper layer.

The lines with 'DELAY' are very useful for obtaining user data throughput in post connection state. Below is trace format for these lines, also Table 5 has explanation of all the parameters:

BD_ADDR <bd_addr> DELAY <end_to_end_delay> SIZE <size_in_bytes> clock <simulation_time>

where:

Parameter	Explanation
bd_addr	Bluetooth address of the receiving device
end_to_end_delay	Delay since *L2CA_dataWrite* primitive was called for the packet on a remote device
size_in_bytes	Higher layer packet size (after stripping L2CAP header)
Clock	Clock time since simulation started

Table 5: Trace Parameters

This provides all the data necessary for calculating user data throughput. In ~ns/run/Bluehoc.tcl, this line is processed for plotting throughput and delay graphs.

One can easily co-relate chapter 2 and the material presented in this chapter. In chapter 2 Bluetooth protocol stack is presented, while in this chapter, its implementation in *Bluehoc* is presented. Portion of protocol stack, which is important from simulation and analysis point of view for our simulation study of user data throughput is also covered in this chapter. Extract from log file is also discussed to set the foundation for simulation result analysis, which will be covered in more detail in next chapter.

4 Analysis of Simulation Models and Results

The purpose of this simulation study is to find best possible parameters that can be used to get maximum user data throughput in Bluetooth piconet in post connection state. It is a well known fact that some of the parameters that can effect wireless communication are: distance between transmitter and receiver (master – slave distance), number of devices competing for bandwidth (number of slaves), synchronization between transmitter and receiver (slave's start time and master's inquiry timeout) and finally, how many devices are present in piconet which have fixed pre-allocated channels (synchronous connection-oriented (SCO)) and how many do not have fixed pre-allocated channels (asynchronous connectionless (ACL)).

Above mentioned parameters also impact Bluetooth communication which can be divided into two distinctive portions:

1. Connection Establishment
2. Connection Transmission

A lot of work has already been done on connection establishment part [16]. Researchers have looked at Bluetooth from sensor network perspective [17]. Also impact of the interference generated by IEEE 802.11 enabled devices on Bluetooth communication [18] and so on has been studied in detail. However, very little research has been done on the post connection transmission part, that is, parameters that effect user data throughput once the connection has been established. Post connection state is also highlighted with rectangle in Figure 4-1.

In this project, simulation study is carried out by creating 201Tcl scripts. Each of these scripts can generate traffic that is used to calculate user data throughput. The data transmission is always from master to one of its slaves and is unicast. Three models are used during the simulation study to calculate the effect of change of one parameter on overall system's user data throughput. In model # 1, distance is varied from 3m to 6m and effect on user data throughput is studied for both ACL and SCO links. In model # 2, inquiry timeout of master and start time of slaves are studied by repeating simulation for

SCO and ACL links at start time of 0.1, 0.5 and 1 second along with inquiry timeout of 20.48 S and 10.24 S. In model # 3, SCO and ACL link slaves are mixed to see gradual change in user data throughput as more and more SCO links are introduced in system.

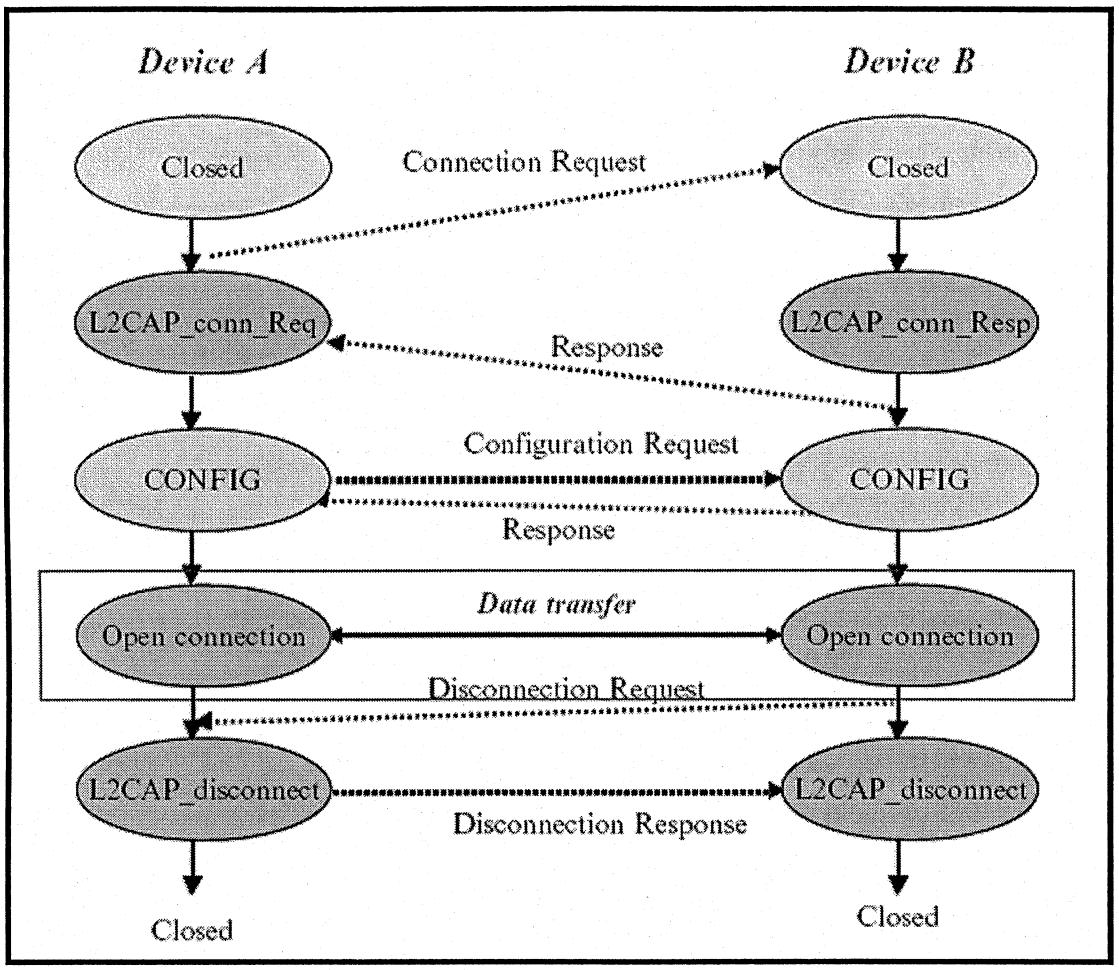


Figure: 4-1 State Diagram for L2CAP Connection (Adapted from [4])

In this project, we applied the best possible values calculated from the earlier model in the next model to calculate again best possible values and then apply again those best possible values in the next model, that is, from the first simulation model we calculated the best possible distance between master and slaves to get maximum throughput and used that set of parameters to calculate the best possible inquiry timeout for master and start time for slaves in the second simulation model. In the third simulation model we used three sets of parameters that we calculated in model # 1 and 2, to see their impact on mixed SCO and ACL links in the piconet.

4.1 Simulation Model #1.0: ACL Link Slaves with Start Time 0.1 S

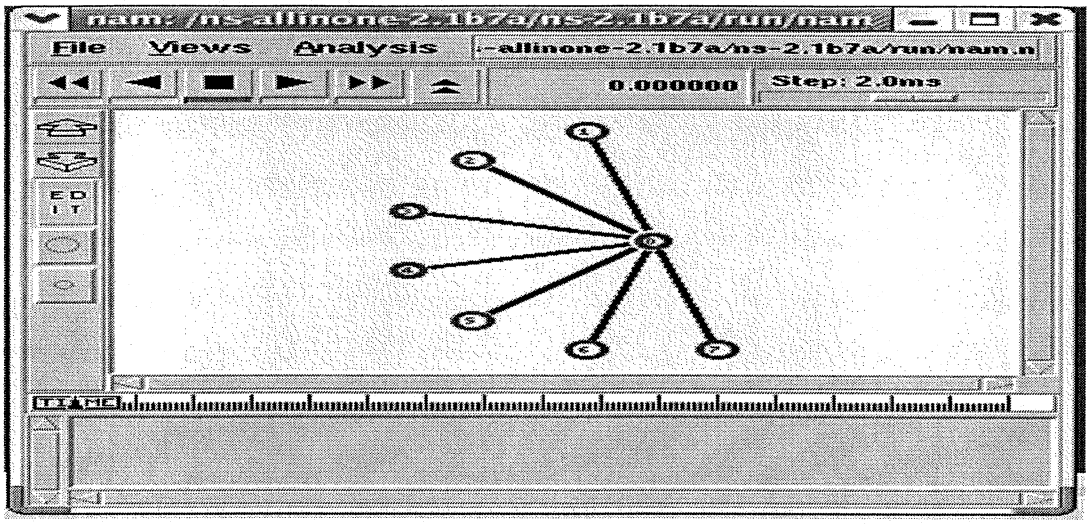


Figure: 4-2 NAM Output Screenshot

The purpose of this simulation model is to find a distance between master and slaves where user data throughput of the system is the maximum. During the simulation, piconet model is used, because, *Bluehoc* implements the Bluetooth piconet connection functionalities [8][9][18]. In fact, it is the only supported model in *Bluehoc* [13][14]. Based on the piconet model, the simulation is repeated 7 times, starting with one and ending at seven slaves as shown Figure 4-2, each time number of slaves is increased by one. During each repeat cycle, a set of Tcl scripts for 3m, 4m, 5m and 6m distances between master and slaves is executed. Simulation script files for each distance are only different in the distance between master and slaves, while all other parameters remain same.

4.1.1 Simulation Parameters

Following simulation parameters are kept the same in all the scripts.

---

set Sim(Transport) [list TCP/Reno]
set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .1]
set InqScanOffset [list 3200]

4.1.2 Results and Analysis

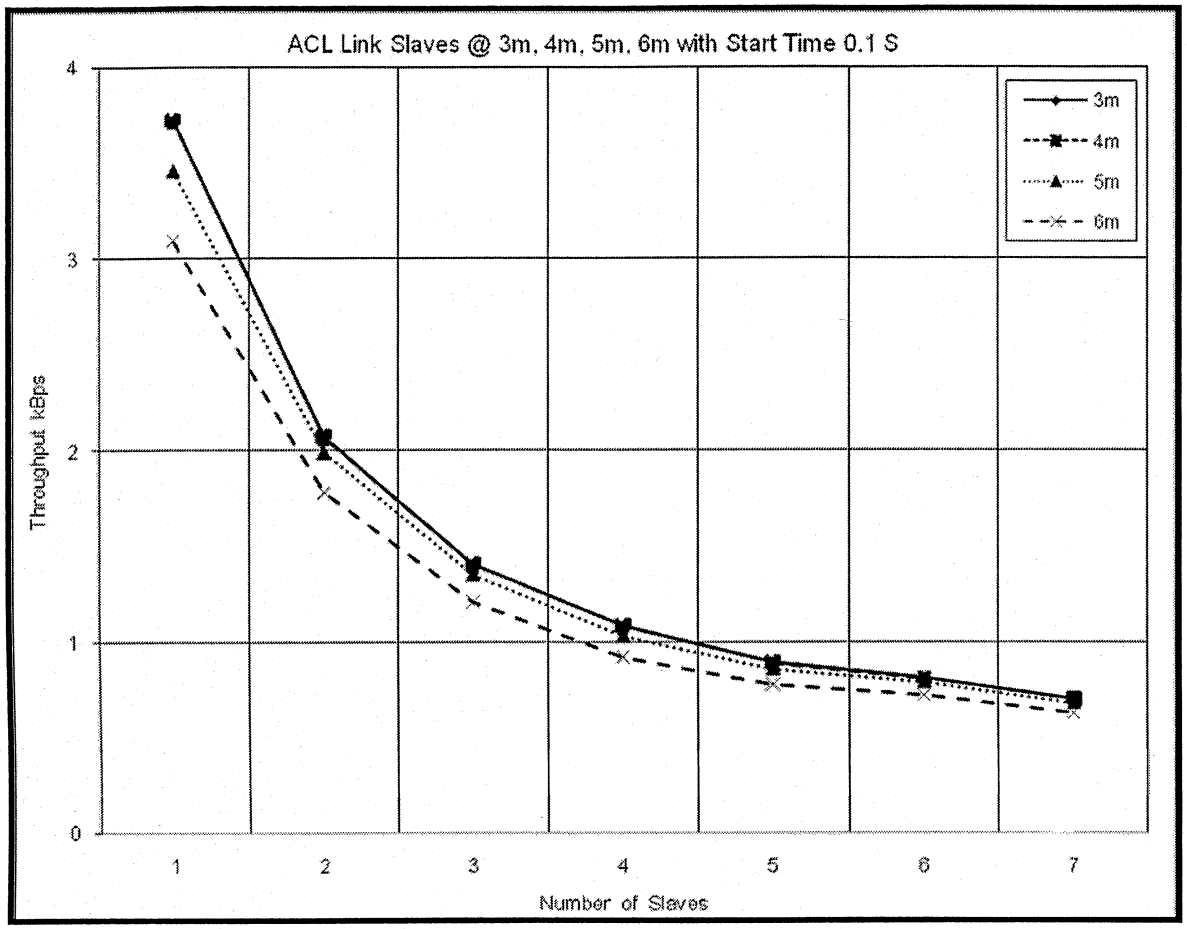


Figure: 4-3 ACL Link Slaves @ 3m, 4m, 5m, 6m with Start Time 0.1 S

There are four curves in the graph shown in Figure 4-3 each representing one fixed distance between master and slave nodes. All slaves are data slaves (ACL). Average throughput for each slave is calculated over 3m, 4m, 5m and 6m and then added and averaged to get system's average throughput (for example, at 2 slaves throughput of slave #1 and 2 is added and then divided by 2 to get system throughput). As can be seen, at all the distances, when there is just one slave, throughput is significantly high, but as soon as one more slave is added, it drops significantly, that is, drop in throughput from one slave to two slaves is exponential while from two to seven it is linear.

Transmission on ACL links, which are used for data transmission, is established on a per-slot basis (if slots not reserved for SCO links). After an ACL transmission from the master, only the addressed slave device may respond during the next time slot, or if

no device is addressed, the packet is considered a broadcast message. Most ACL links include packet re-transmission [1].

Based on the above behavior of ACL, let us analyze our results. When there is just one slave, master node just polled one slave - it means that whenever, there is data, it is sent in the next time slot. Therefore, let us consider this throughput (3.72kBps) as our base line, because, system can not perform better than this. By introducing another slave in the system, master node is forced to poll two slaves and send data one by one to each slave in turns, that is, alternate time slots are allocated to each slave. This sharing of time slots is responsible for dragging system's average throughput to 55% per slave, that is, approximately, half of base line throughput. This can be verified from Table 6. Similarly, by introducing another slave system's average throughput slipped to approximately, 37% per slave of base line throughput, which is approximately, one-third of base line throughput.

If we keep increasing slaves we will see that system's average throughput per slave would slip approximately to *1/n* of base line throughput, where *n* is number of slaves in the system.

No. of Slaves	Average Throughput kBps	% Change in Throughput
1	3.72	100
2	2.07	55
3	1.4	37
4	1.08	29
5	0.89	23
6	0.81	21
7	0.7	18

Table 6: Number of Slaves vs. Percentage Change in Throughput

4.1.3 Variable Distance

When we look at the results in Figure 4-3, there is no difference in throughput at a distance of 3m and 4m (not visible in chart, because, lines are overlapping each other) but there is a small difference in throughput at 5m and 6m, this might suggest that free space model gives no penalty at distance less than 4m and gives little penalty at distance above 4m. The author in [14] also confirms this behavior in his thesis.

No. of Slaves	Average Throughput kBps @3m	Average Throughput kBps @4m	Average Throughput kBps @5m	Average Throughput kBps @6m
1	3.72	3.72	3.46	3.09
2	2.07	2.07	1.99	1.78
3	1.4	1.4	1.35	1.21
4	1.08	1.08	1.03	0.92
5	0.89	0.89	0.86	0.78
6	0.81	0.81	0.79	0.72
7	0.7	0.7	0.68	0.63

Table 7: Throughput at 3m, 4m, 5m, 6m Distances Between Slave(s) and Master

4.2 Simulation Model # 1.1: SCO Link Slaves with Start Time 0.1 S

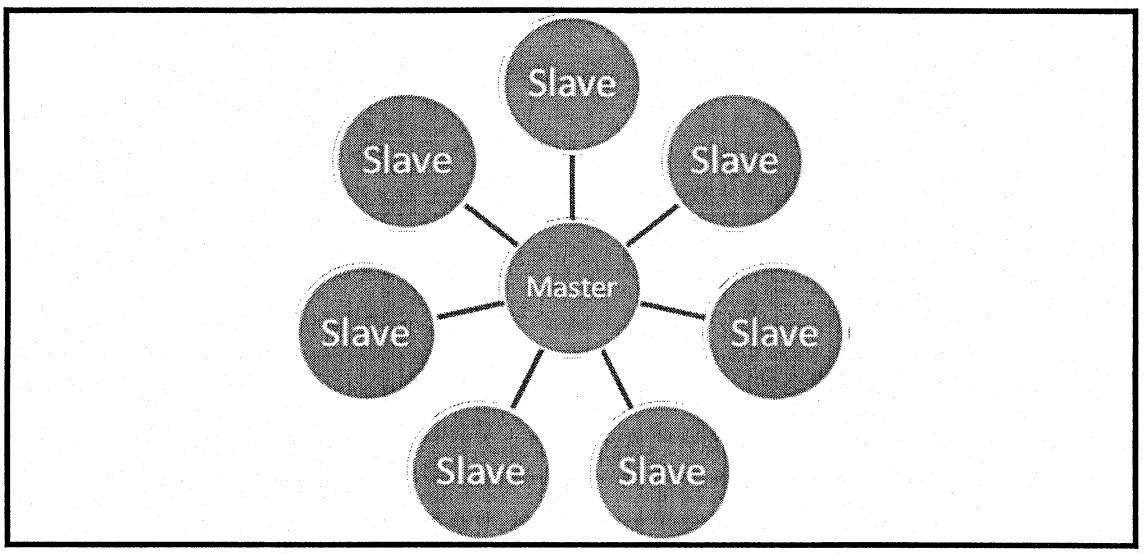


Figure: 4-4 Model Used for Variable Distance Modeling

SCO links between master and slaves are used for this simulation model to find a distance between them, where throughput of the system is the maximum as shown in Figure 4-4. Based on piconet model, the simulation is repeated 7 times starting with one SCO slave and adding one more at each cycle until we reach at seven slaves. During each repeat cycle, a set of Tcl scripts for 3m, 4m, 5m and 6m distances between master and slaves node is executed. Simulation script files for each distance are only different in one parameter only, that is, distance between master and slave nodes. Every possible effort is used to make sure this SCO simulation model and previous ACL data model are identical except for link properties between master and slaves.

4.2.1 Simulation Parameters

Following simulation parameters are kept the same in all the scripts.

---

set Sim(Transport) [list UDP]
set Sim(Application) [list Traffic/Exponential]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .1]
set InqScanOffset [list 3200]

4.2.2 Results and Analysis

There are four curves in Figure 4-5 each representing a constant distance between master and slaves node. As can be seen there are just two curves visible, reason being that the average system throughput is same at 3m, 4m and 5m. At all the distances, when more and more slaves are added throughput drops but not significantly.

This behavior is different from earlier simulation model using ACL links, but can easily be explained by the fact that ACL links are established on a per-slot. After an ACL transmission from the master, only the addressed slave device may respond during the next time slot. Most ACL links include packet retransmission [1]. SCO links are typically used for voice transmission. These are point-to-point symmetric connections that reserve time slots in order to guarantee timely transmission. The slave device is always allowed to respond during the time slot immediately following an SCO transmission from the master. SCO packets are never retransmitted [1], also they use UDP as their transport protocol.

One more difference that is observed between results of the simulation done using ACL and SCO links is that, SCO link has higher throughput ranging between (45kBps-57kBps) while, ACL link has range of (0.63kBps and 3.72kBps). This behavior can be explained by the fact that, ACL traffic experiences delay that is caused by L2CAP in packets transmission, re-transmissions (in case of error conditions) and segmentation and reassembly (SAR). On the other hand SCO traffic (because the timeslots reserved for SCO traffic) experiences no such delays.

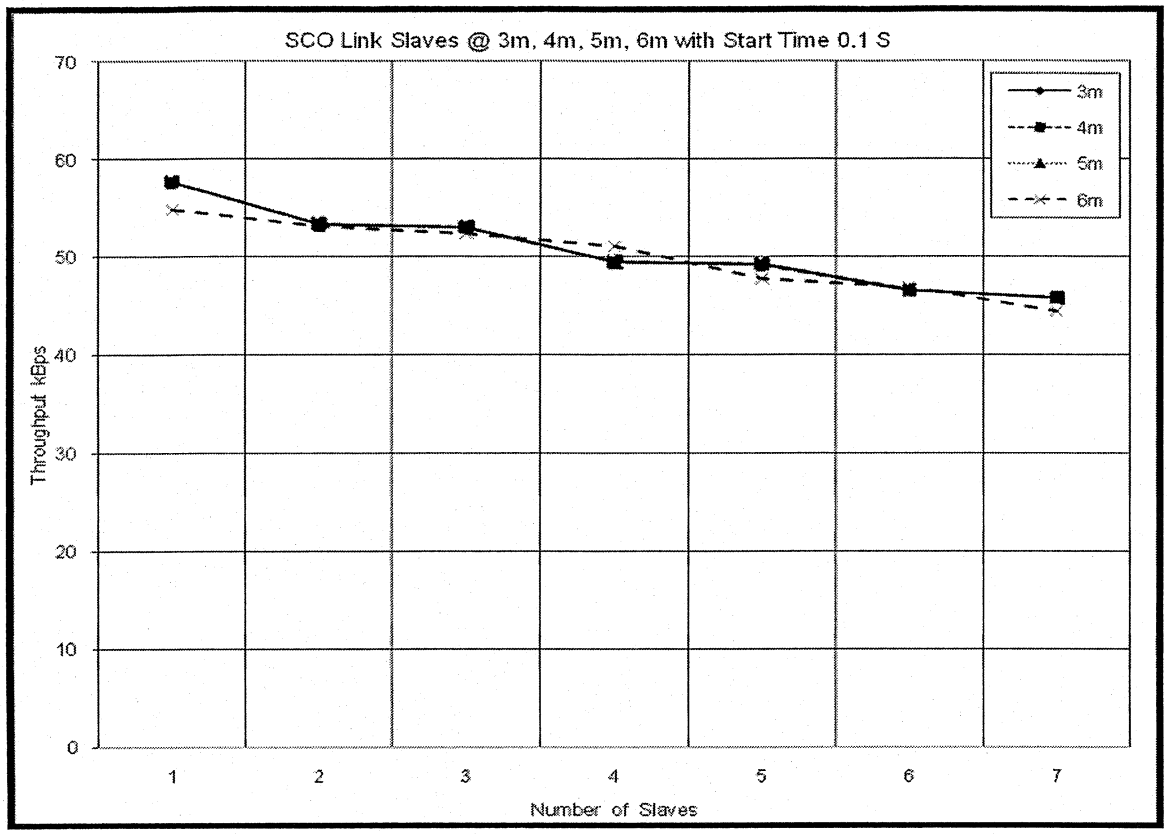


Figure: 4-5 SCO Link Slaves @ 3m, 4m, 5m, 6m with Start Time 0.1 S

No. of Slaves	Average Throughput kBps	% Change in Throughput
1	57.69	100
2	53.36	92
3	53.09	92
4	49.49	85
5	49.24	85
6	46.6	80
7	45.81	79

Table 8: Throughput at a Distance of 3m

4.2.3 Variable Distance

There is no difference in throughput at a distance of 3m, 4m and 5m but there is little difference in throughput at 6m. This might suggests that free space model gives no penalty at distance less than 5m and gives little penalty at 6m. The author also confirms this in his thesis [14].

No. of Slaves	Average Throughput kBps @3m	Average Throughput kBps @4m	Average Throughput kBps @5m	Average Throughput kBps @6m
1	57.69	57.69	57.69	54.8
2	53.36	53.36	53.36	53.09
3	53.09	53.09	53.09	52.37
4	49.49	49.49	49.49	51.05
5	49.24	49.24	49.24	47.78
6	46.6	46.6	46.6	46.88
7	0.7	0.7	0.68	0.63

Table 9: Throughput at 3m, 4m, 5m, 6m Distances Between Slave(s) and Master

4.3 Simulation Model # 2.0: ACL Link Slaves with InqTimeout 20.48 S and Start Time 0.1 S

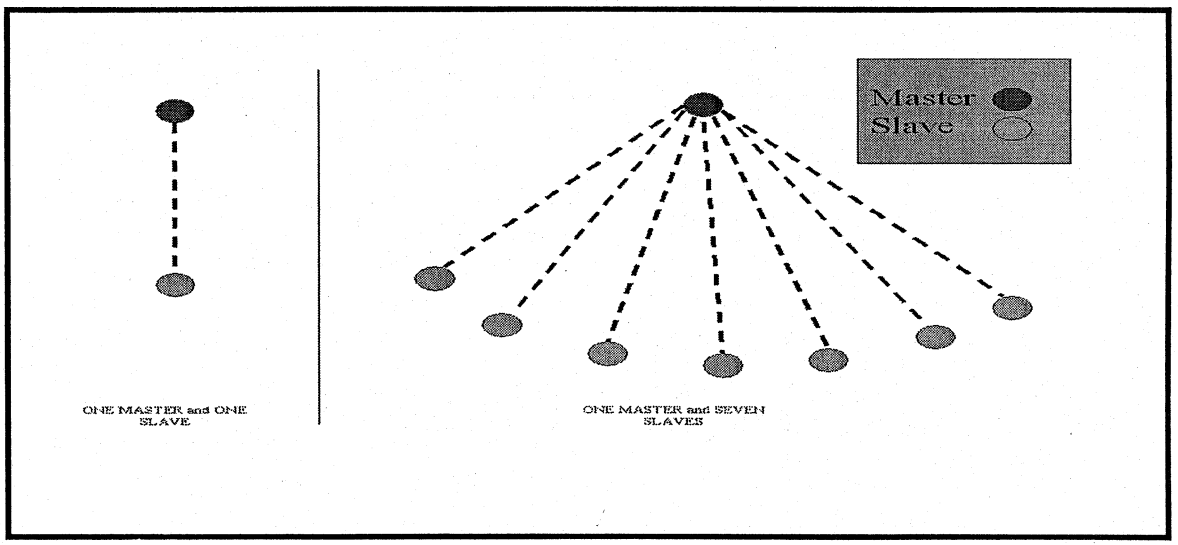


Figure: 4-6 ACL Link Slaves with InqTimeout 20.48 S and Start Time 0.1 S

This simulation model is same as the simulation model #1.0 (ACL) as shown in Figure 4-6, except that this time, target is to find best possible value for master's inquiry timeout instead of distance between master and slaves when load (number of slaves) increases, while keeping all other parameters constant. Simulation is repeated 7 times starting with one and ending at seven slaves, in each cycle, number of slave(s) is increased by one. During each repeat cycle, a set of Tcl scripts for 3m, 4m, 5m and 6m distance between

master and slave node is executed. During each simulation cycle data is collected for further analysis.

4.3.1 Simulation Parameters

Following simulation parameters are kept same in all the scripts.

set Sim(Transport) [list TCP/Reno]
set Sim(Application) [list FTP]
set InqTimeout 64768
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .1]
set InqScanOffset [list 3200]

4.3.2 Results and Analysis

No. of Slaves	Average Throughput in kBps @ 10.24 S Inquiry Timeout and @ 3m	Average Throughput in kBps @ 20.48 S Inquiry Timeout and @ 3m
1	3.72	3.72
2	2.07	2.08
3	1.4	1.42
4	1.08	1.19
5	0.89	0.95
6	0.81	1.27
7	0.7	0.95

Table 10: Average Throughput @ 10.24 S and 20.48 S Inquiry Timeout

As can be seen in Figure 4-7 and Table 10, slaves at a distance of 3m and 4m have consistent higher throughputs as compared to the ones at 5m and 6m, as expected, at all the distances. When there is just one slave, throughput is significantly high, but as soon as one more slave is added, it drops significantly, that is, drop in throughput from one slave to two slaves is exponential while from two to seven it is linear. Also as can be seen from Table 10, there is no significant change in throughout at 10.24 S and 20.48 S inquiry timeout.

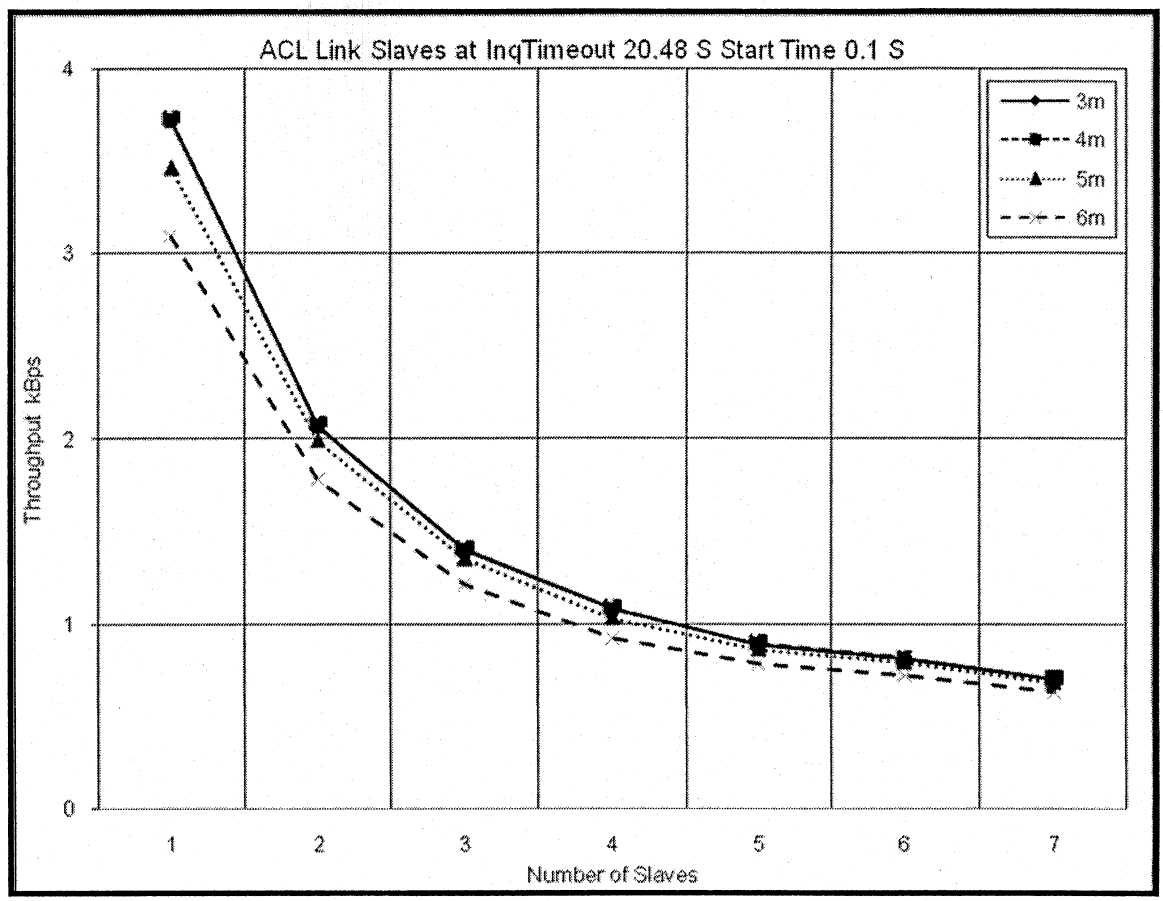


Figure: 4-7 ACL Link Slaves @ 3m, 4m, 5m, 6m with Inquiry Timeout 20.48 S

4.3.3 Impact on Throughput of Inquiry Timeout

As shown in Table 10, average throughput @ Inquiry Timeout 20.48 S is better than average throughput @ Inquiry Timeout 10.24 S.

4.4 Simulation Model # 2.1: SCO Link Slaves with InqTimeout 20.48 S Start Time 0.1S

SCO links between slaves and master are used for this simulation model, as shown in Figure 4-8, to find best possible value of inquiry timeout for master when load (number of slaves) increases, while keeping all other parameters constant.

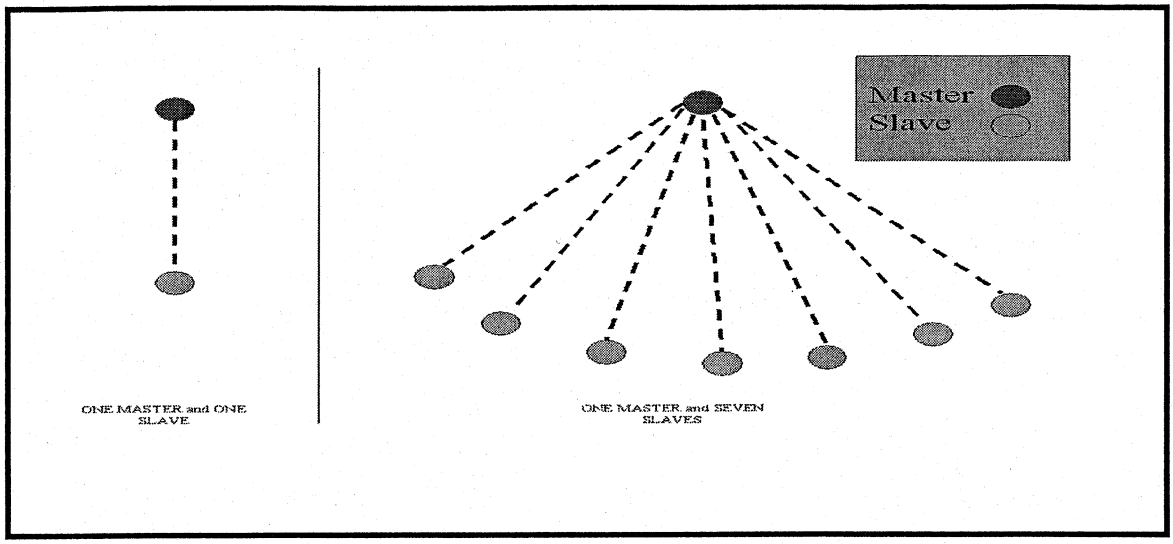


Figure: 4-8 SCO Link Slaves @ 3m, 4m, 5m, 6m with Inquiry Timeout 20.48 S

4.4.1 Simulation Parameters

Following simulation parameters are kept the same in all the scripts.

---

set Sim(Transport) [list UDP]
set Sim(Application) [list Traffic/Exponential]
set InqTimeout 64768
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .5]
set InqScanOffset [list 3200]

4.4.2 Results and Analysis

Figure 4-9 has four graph lines but 3m, 4m, 5m lines are overlapping each other and are visible as one line. As can be seen from Figure 4-9 that the throughput when there are 3 slaves in the system is little higher than when there are 2 slaves in the system. Same is applicable when there are 5 slaves in the system, that is, throughput is higher as compare to when 4 slaves are in the system. These unexpected changes in throughput can also be explain by the fact that the system throughput depends on the number of active connections between slaves and master.

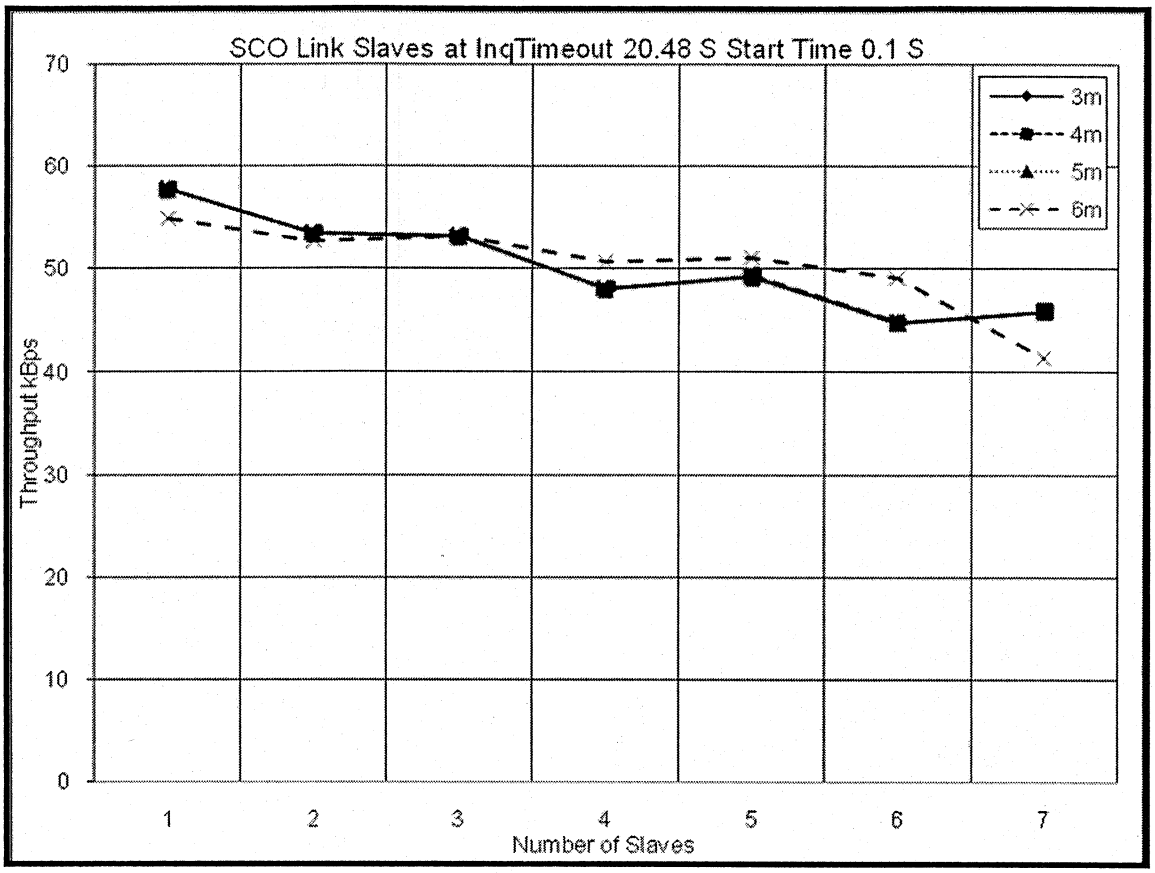


Figure: 4-9 SCO Link Slaves @ 3m, 4m, 5m, 6m with Inquiry Timeout 20.48 S

No. of Slaves	Average Throughput in kBps @ 10.24 S Inquiry Timeout and @ 3m	Average Throughput in kBps @ 20.48 S Inquiry Timeout and @ 3m
1	57.69	57.69
2	53.36	53.36
3	53.09	53.09
4	49.49	47.98
5	49.24	49.24
6	46.6	44.76
7	45.81	45.81

Table 11: SCO Link Slaves @ 3m, 4m, 5m, 6m with Inquiry Timeout 20.48 S

4.4.3 Impact on Throughput of Inquiry Timeout

As shown in Table 11, average throughput @ inquiry timeout 20.48 S is almost same as average throughput @ inquiry timeout 10.24 S.

4.5 Simulation Model # 2.2: ACL Link Slaves with Start Time 0.5 S

This simulation model is the same as simulation model # 1.0 except that this time target is to find best possible value for slaves' start time instead of distance between master and slaves when load (number of slaves) increases, while keeping all other parameters constant. Simulation is repeated 7 times starting with one and ending at seven slaves, in each cycle, number of slave(s) is increased by one. During each repeat cycle, a set of Tcl scripts for 3m, 4m, 5m and 6m distance between master node and slave(s) is executed. During each simulation cycle data is collected for further analysis.

4.5.1 Simulation Parameters

Following simulation parameters are kept same in all the three scripts.

---

set Sim(Transport) [list TCP/Reno]
set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .5]
set InqScanOffset [list 3200]

4.5.2 Results and Analysis

As can be seen in Figure 4-10, there is no change in behavior as far as throughput vs. distance is concerned. Slaves at a distance of 3m and 4m have consistent higher throughputs as compared to ones at 5m and 6m, as expected. However, after 5 slaves, system becomes unstable. As can be seen in Figure 4-10, there is sudden increase in throughput when there are 6 slaves in the system. There is also improvement in throughput when there are 7 slaves in the system as compared to when there are 5 slaves in the system.

This anomaly in throughput is because of the fact that when there are 5 slaves in the system only 4 are able to connect after successful inquiry within time limit (InqTimeout = 10.24 S). When there are 7 slaves in the system only 5 are able to connect (that is why, throughput, when there are 7 slaves in system is better than when there are 5 slaves). However, when there are 6 slaves in the system all of them are able to connect

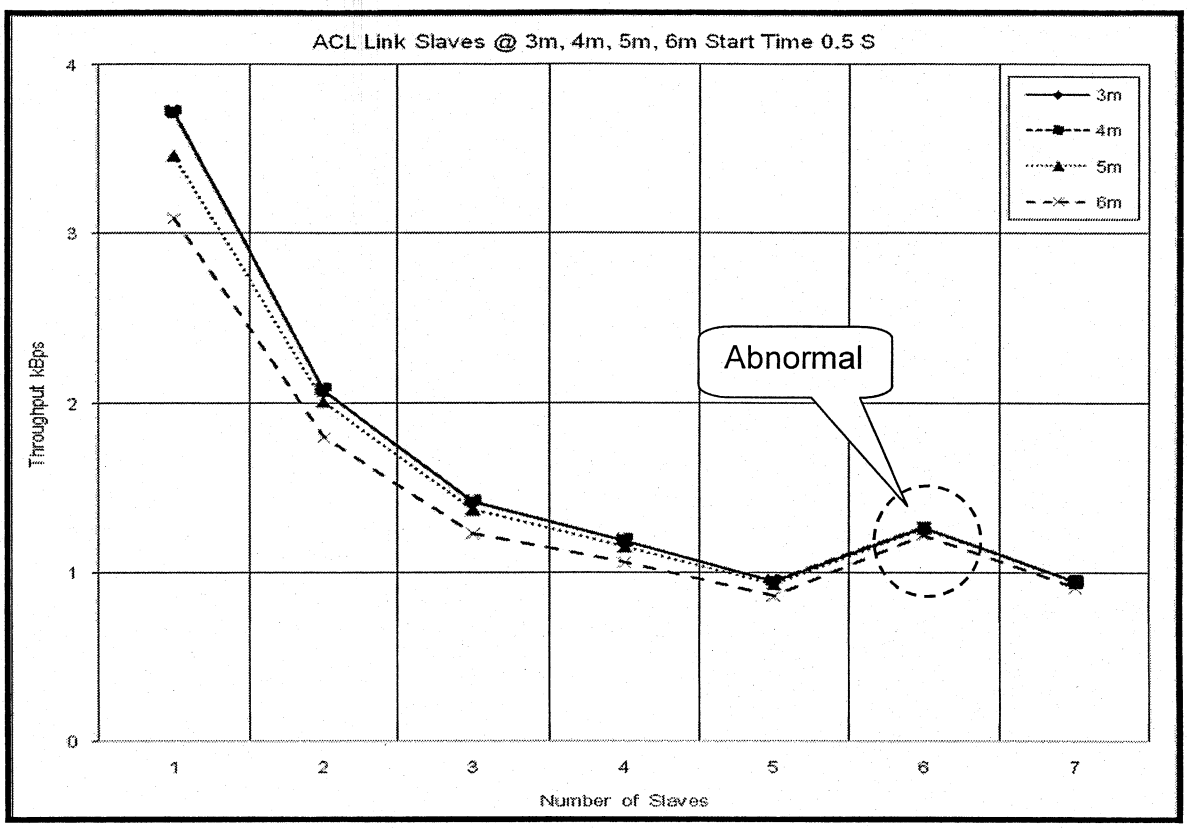


Figure: 4-10 ACL Link Slaves @ 3m, 4m, 5m, 6m with Start Time 0.5 S

within service discovery inquiry time limit. All above stated discovery inquiry calculations are done based on traces, which are also presented in annex (b) for reference.

4.5.3 Impact on Throughput of Start Time

As shown in Table 12, average throughput @ 0.5 S, start time is better than average throughput @ 0.1 S, start time, but the only problem is that, there is a glitch in throughout when there are 6 slaves in the system.

No. of Slaves	Average Throughput in kBps @ 0.1 S (start time)	Average Throughput in kBps @ 0.5 S (start time)
1	3.72	3.72
2	2.07	2.08
3	1.4	1.42
4	1.08	1.19
5	0.89	0.95
6	0.81	1.27
7	0.7	0.95

Table 12: Average Throughput @ 0.1 and 0.5 S Start Time

4.6 Simulation Model # 2.3: SCO Link Slaves with Start Time 0.5 S

SCO links between slaves and master are used for this simulation model to find best possible value of start time for slaves when load (number of slaves) increases, while keeping all other parameters constant.

4.6.1 Simulation Parameters

Following simulation parameters are kept the same in all the scripts.

---

set Sim(Transport) [list UDP]
set Sim(Application) [list Traffic/Exponential]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 .5]
set InqScanOffset [list 3200]

4.6.2 Results and Analysis

Figure 4-11 has four graph lines but 3m, 4m, 5m lines are overlapping each other and are visible as one line. As can be seen from Figure 4-11 that the throughput when there are 3 slaves in the system is little higher than when there are 2 slaves in the system. Same is applicable when there are 5 slaves in the system, that is, throughput is higher as compare to when 4 slaves are in the system. These unexpected changes in throughput can also be explain by the fact that the system throughput depends on the number of active connections between slaves and master.

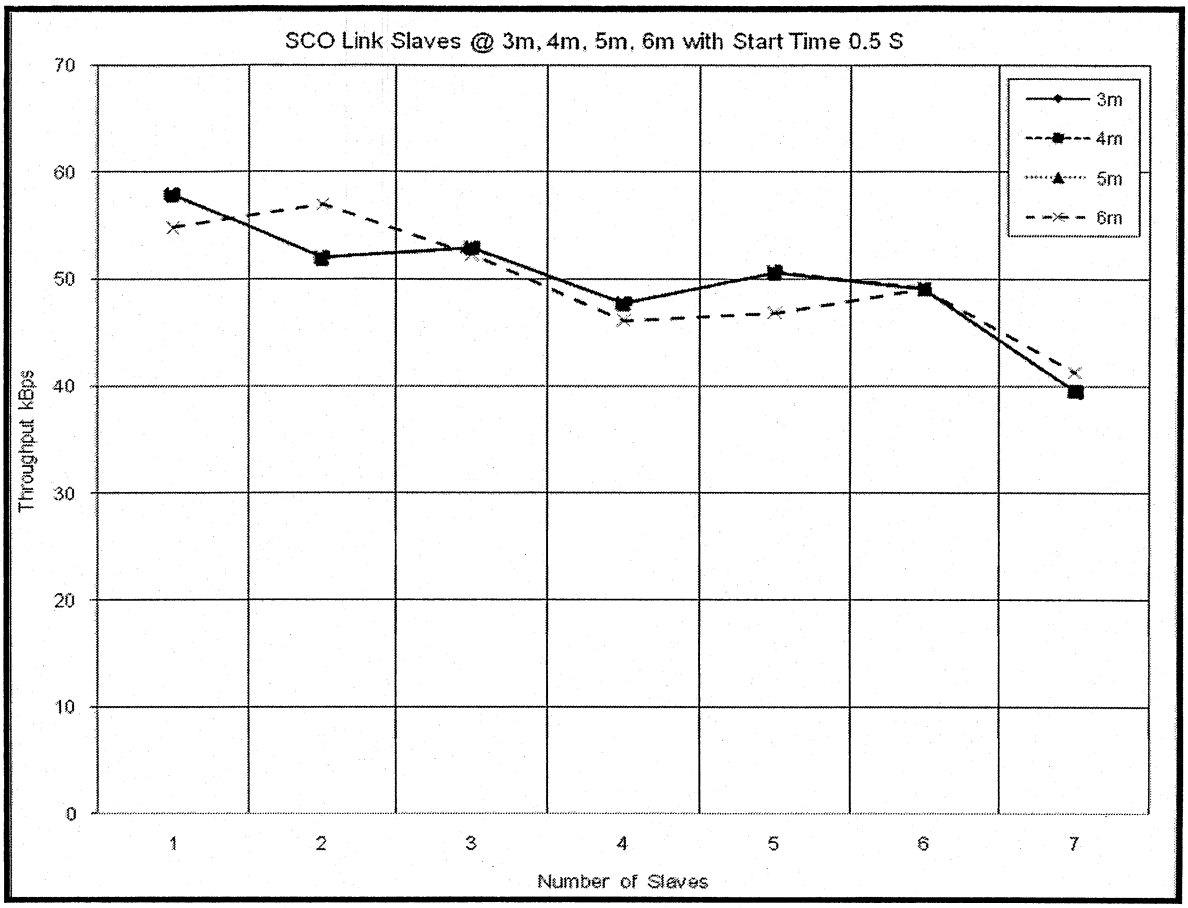


Figure: 4-11 SCO Link Slaves @ 3m, 4m, 5m, 6m with Start Time 0.5 S

4.7 Simulation Model # 2.4: ACL Link Slaves @ Inquiry Timeout (20.48 S and 10.24 S)

We have seen in simulation model # 2.2 that system's throughput depends on the number of active connections between master and slaves. It is clear from the logs in annex (b) that slaves are unable to connect, because, they fail to pass inquiry (also called discovery) within inquiry timeout limit. To increase the chances of slaves' discovery we can tune two parameters. First parameter is, *InqTimeout* of the master node, by increasing its value, there is more time available to slave nodes to pass discovery state. The other parameter is *StartTime* of slaves nodes, by increasing its value, there are fewer slave nodes waiting to be discovered at any given instant of time. To investigate the impact of above mentioned two parameters, we create two sets of scripts, one for each parameter. Results from model # 2.2 are used as base line set. The first set of scripts has slaves' *StartTime* of 1.0 S, which is almost double what we use in model # 2.2, while,

InqTimeout of master node is kept fixed at 10.24 S. In the second set of scripts, slaves' *StartTime* is kept fixed at 0.5 S, while, *InqTimeout* of master node is increased to 20.48 S. Simulation run time is also increased to 40 S to compensate increase in inquiry timeout limit for second set of scripts. During simulation load is increased (number of slaves), while keeping all other parameters constant in each run for each set of scripts. Simulation parameters are noted in simulation parameter section below. The distance between master and slaves is 3m.

Simulation Parameters

Following simulation parameters are used in simulation scripts.

Set 1: Start Time 1 S, Inquiry Timeout 10.24 S and Simulation Time 20 S

set Sim(Transport) [list TCP/Reno]
set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>
set SimulationTime 20.0
set StartTime [list 0 1]
set InqScanOffset [list 3200]

Set 2: Start Time 0.5 S, Inquiry Timeout 20.48 S and Simulation Time 40 S

set Sim(Transport) [list TCP/Reno]
set Sim(Application) [list FTP]
set InqTimeout 64768
set NumResponses <Same as number of Slaves>
set SimulationTime 40.0
set StartTime [list 0 .5]
set InqScanOffset [list 3200]

4.7.1 Results and Analysis

For analysis, Figure 4-12 can be divided into three sections. In section - A, number of slaves connected to master mode are same for all three sets of results. In section – B, number of connected slaves are identical for set 2 and base line set, while, the set 1 has lesser number of slaves connected. In section – C, set 2 has 6 slaves connected, which is better than both base set as well as set 1.

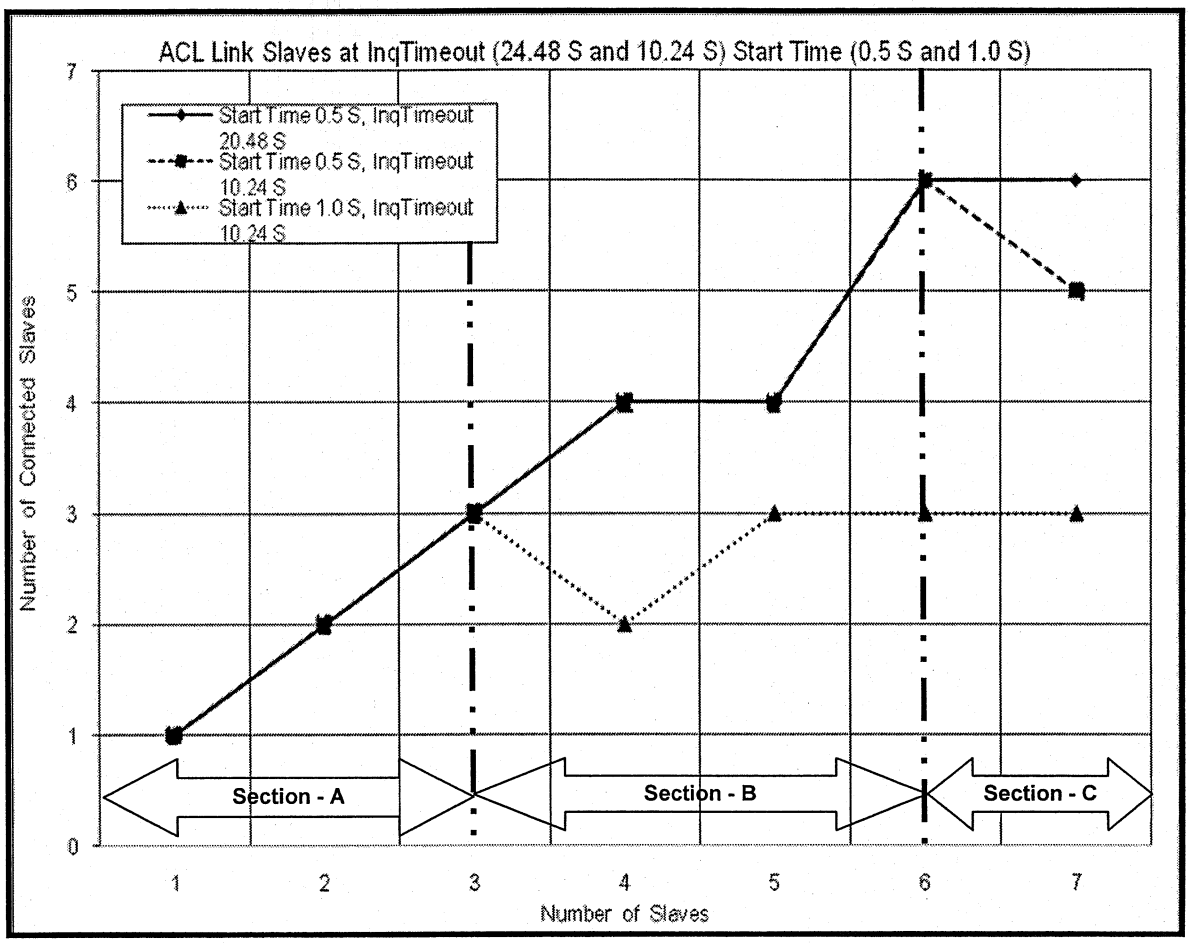


Figure: 4-12 Number of Slaves Connected

As can be seen in Figure 4-13, system's throughput is lower for set 1 than base line set. Also, system's throughput is lower for set 2 than base line set, but curve is smooth and more predicable. Therefore, it is concluded, by increasing *InqTimeout* parameter, system becomes stable and throughput decreased in expected manner. Also, number of slaves connected increases by one at 7 slaves in the system. Increase in *StartTime* to 1.0 S has no significant impact on throughput, while, number of connected slaves decrease significantly. To understand number of connected slaves vs. throughput behavior in more detail, that is, at IP packets level refer to annex (a) and annex (b).

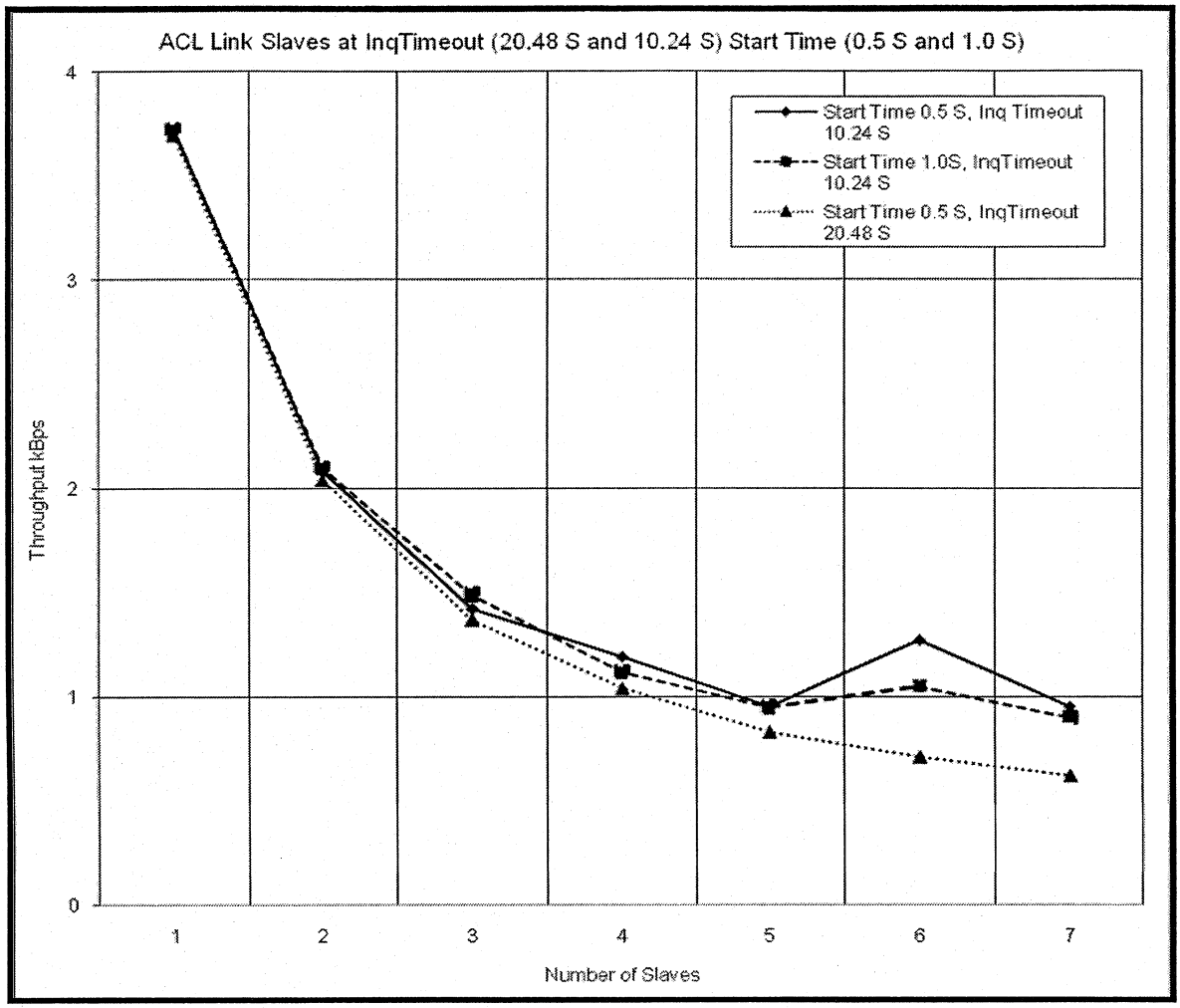


Figure: 4-13 Throughput vs. Inquiry Timeout and Start Time

4.8 Simulation Model # 2.5: SCO Link Slaves @ Inquiry Timeout (20.48 S and 10.24 S)

In this simulation model, we repeat simulation model # 2.4 with one exception, that is, we use SCO instead of ACL link slaves. Simulation parameters are noted in simulation parameter section. The distance between master and slaves is 3m.

4.8.1 Simulation Parameters

Following simulation parameters are used in simulation scripts.

Set 1: Start Time 1 S, Inquiry Timeout 10.24 S and Simulation Time 20 S

et Sim(Transport) [list UDP]
set Sim(Application) [list Traffic/Exponential]
set InqTimeout 32384
set NumResponses <Same as number of Slaves>

set SimulationTime 20.0
set StartTime [list 0 1]
set InqScanOffset [list 3200]

Set 2: Start Time 0.5 S, Inquiry Timeout 20.48 S and Simulation Time 40 S

set Sim(Transport) [list UDP]
set Sim(Application) [list Traffic/Exponential]
set InqTimeout 64768
set NumResponses <Same as number of Slaves>
set SimulationTime 40.0
set StartTime [list 0 .5]
set InqScanOffset [list 3200]

4.8.2 Results and Analysis

Figure 4-14 has three curves but only two are visible, curves for set 2 and base line set (results from model # 2.2) are overlapping each other. The curve (dotted line) is for set 1. As can be seen in Figure 4-14 set 1 has least number of slaves connected, while there is no difference in number of slaves connected in remaining two sets.

As can be seen in Figure 4-15, system's throughput is very unstable for set 1. Also, system's throughput for set 2 is lower than base line set. Therefore, it is concluded, increase in *InqTimeout* parameter has no positive or noticeable impact on system's throughput, while, increase in *StartTime* has negative impact on system in terms of both stability and throughput.

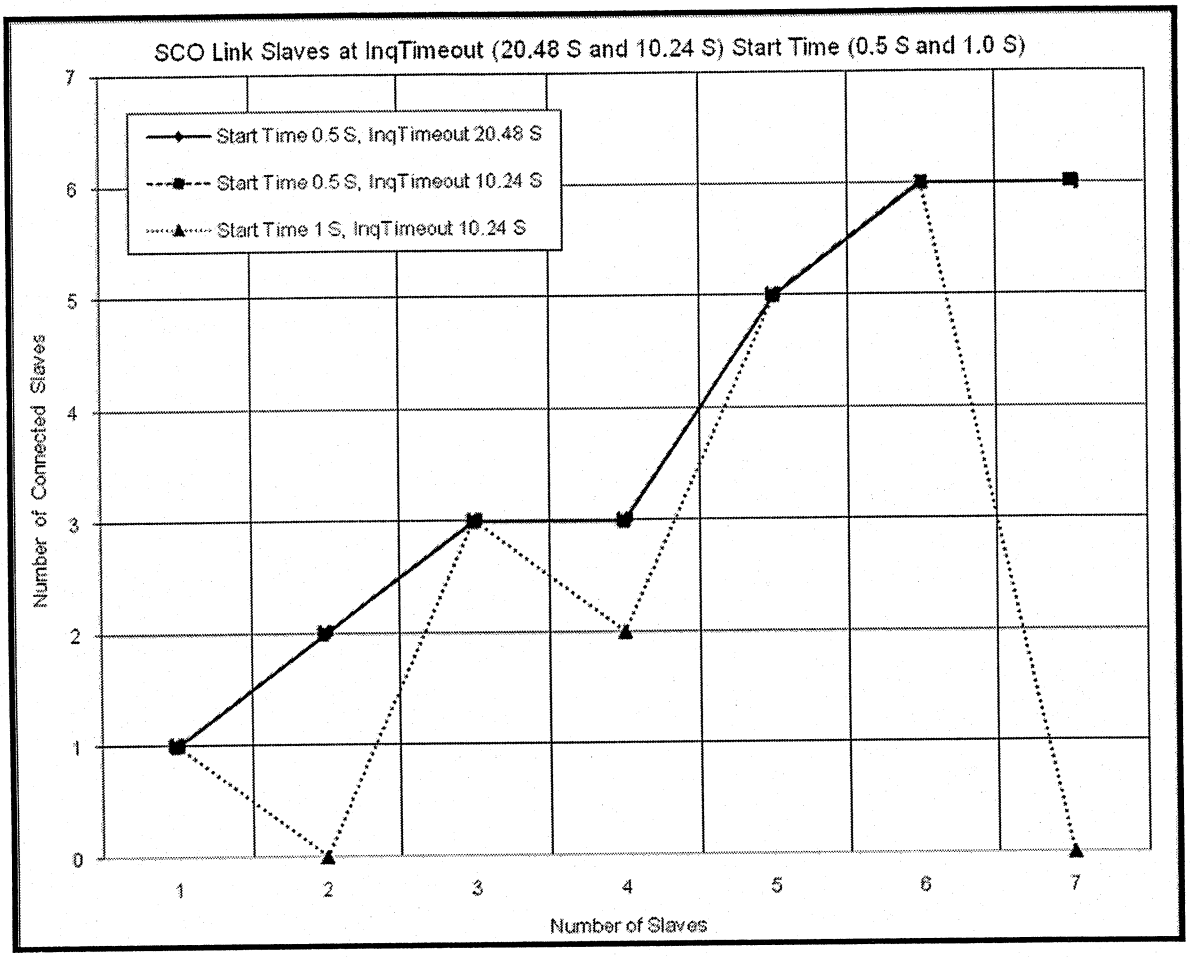


Figure: 4-14 Number of Slaves Connected

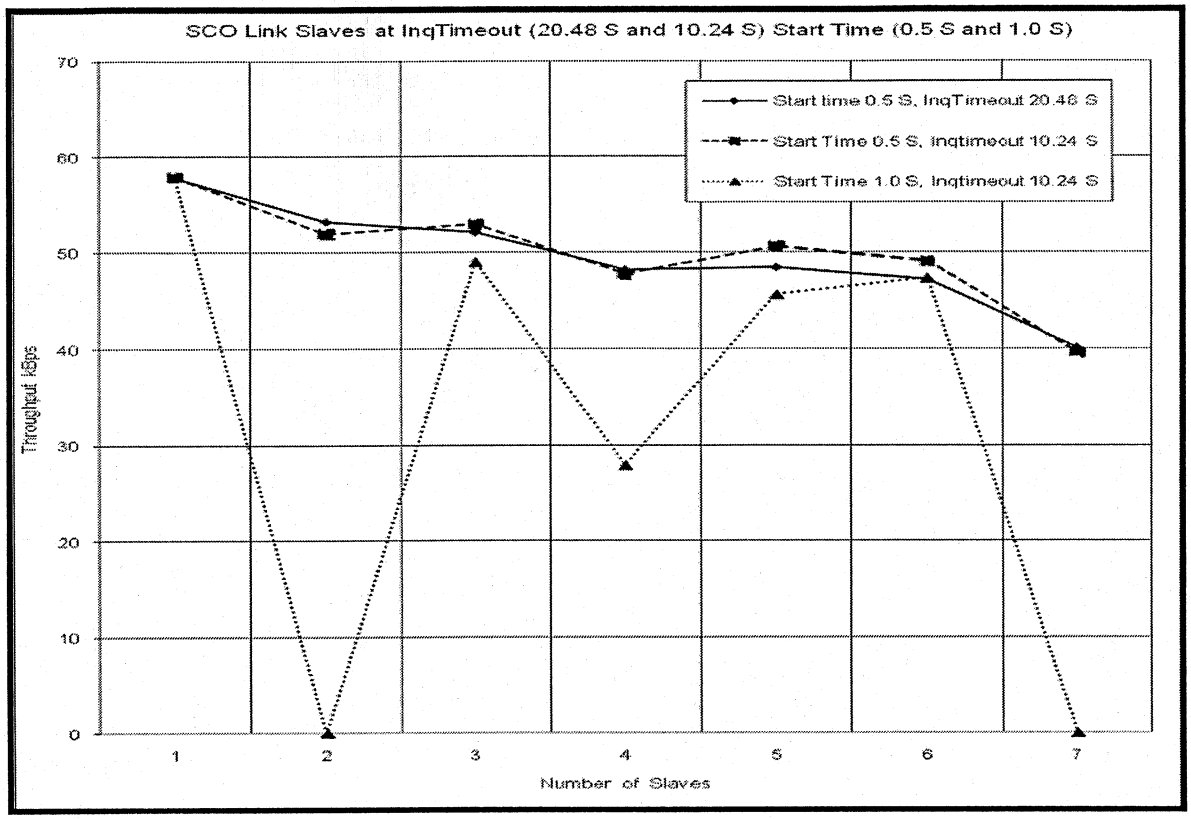


Figure: 4-15 Throughput vs. Inquiry Timeout and Start Time

4.9 Simulation Model # 3.0: SCO and ACL Link Slaves @ 3m with Start Time 0.1 S

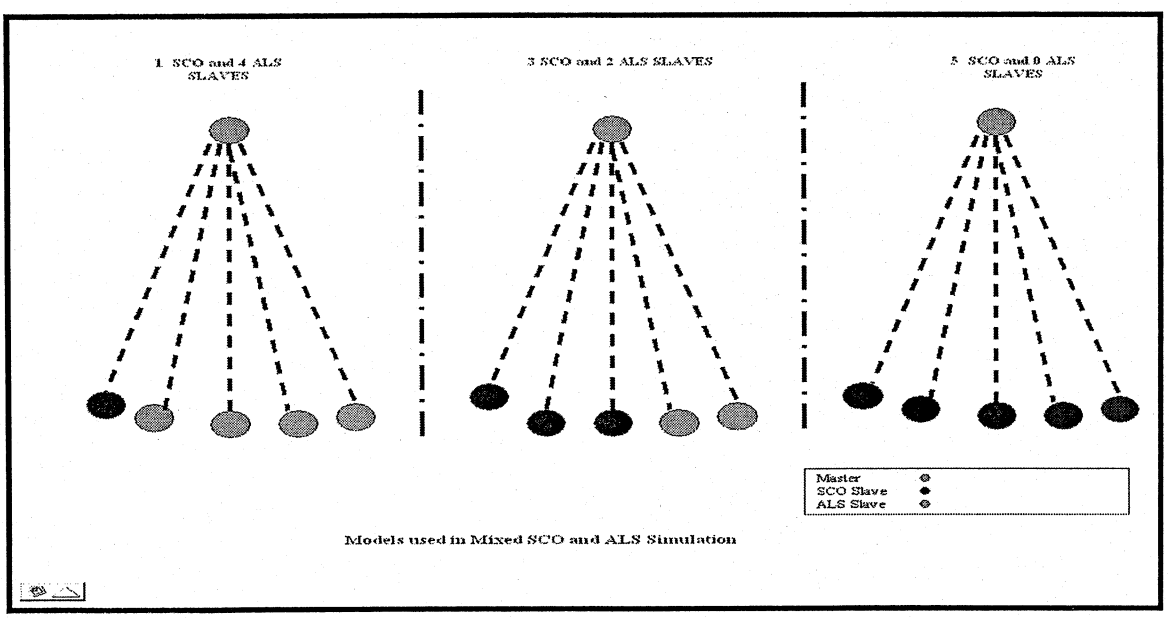


Figure: 4-16 SCO and ACL Link Slaves @ 3m with Start Time 0.1 S

This simulation is the combination of both SCO and ACL link slaves, as shown in Figure 4-16. The distance between the master and slaves is 3m with Start Time of 0.1 second. So far, these are the best possible values calculated from previous simulations. One reason is that this model is used in the simulation to study the impact of having both voice (SCO) and data (ACL) link slaves in a single piconet. Another, reason is to apply the best possible parameters calculated in earlier simulation, that is, to verify that values calculated while keeping data and voice slaves separate in earlier simulations are still valid when applied on the combination of both SCO and ACL link slaves.

Simulation is repeated 3 times, in each cycle number of SCO slaves are increased, that is, we start with one and stop at five salves, each time number of SCO slaves is increased by two. Three sets of scripts are created, one for each cycle.

4.9.1 Simulation Parameters

Following simulation parameters are used in three sets of simulation scripts:

Set 1
1(SCO)-4(ACL) (One voice and four data slaves)

set Sim(Transport) [list UDP TCP/Reno TCP/Reno TCP/Reno TCP/Reno]
set Sim(Application) [list Traffic/Exponential FTP FTP FTP FTP]
set InqTimeout 32768
set NumResponses 5
set SimulationTime 20.0
set StartTime [list 0 .1 .2 .3 .4 .5]
set InqScanOffset [list 3200 3200 3200 3200 3200]

Set 2
3(SCO)-2(ACL) (Three voice and two data slaves)

set Sim(Transport) [list UDP UDP UDP TCP/Reno TCP/Reno]
set Sim(Application) [list Traffic/Exponential Traffic/Exponential Traffic/Exponential FTP FTP]
set InqTimeout 32768
set NumResponses 5
set SimulationTime 20.0
set StartTime [list 0 .1 .2 .3 .4 .5]
set InqScanOffset [list 3200 3200 3200 3200 3200]

Set 3
5(SCO) (Five voice slaves)

set Sim(Transport) [list UDP UDP UDP UDP UDP]
set Sim(Application) [list Traffic/Exponential Traffic/Exponential Traffic/Exponential Traffic/Exponential Traffic/Exponential]
set InqTimeout 32768
set NumResponses 5
set SimulationTime 20.0
set StartTime [list 0 .1 .2 .3 .4 .5]
set InqScanOffset [list 3200 3200 3200 3200 3200]

4.9.2 Results and Analysis

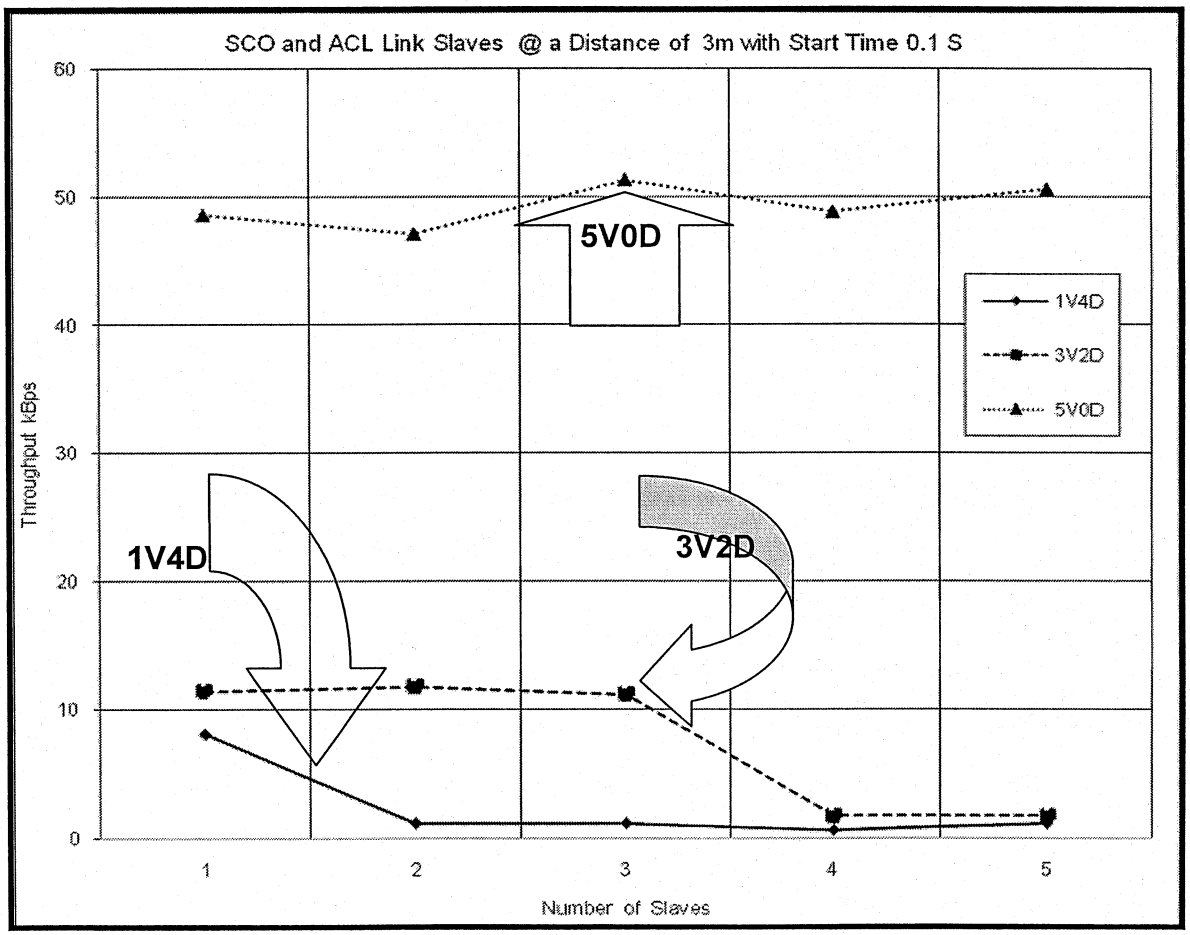


Figure: 4-17 Combined SCO and ACL Link Slaves

Our results from simulation as shown in Figure 4-17 show that in all three cycles, SCO link slaves have higher throughputs as compared to ACL link slaves. This may be

explained by the fact that ACL data uses services from the L2CAP layer before using services from baseband. User application submits data to this layer in variable-sized frames for delivery in the same form to the corresponding application on the remote device, for example, a 1000 byte ACL packet (used in our simulation) is subdivided into 4 DM5 packets of 226 bytes each with the remainder being transmitted in a single 123-byte DM3 packet. Also, L2CAP channels have QoS setting that defines restrictions on the delivery of the frames of data, for example, that the data is isochronous and, therefore, has a limited lifetime after which it becomes invalid, that the data should be delivered within a given time frame, or that the data is reliable and should be delivered without error, however long this takes. The L2CAP channel manager is also responsible for multiplexing ACL data onto the baseband logical link with other L2CAP channels with similar characteristics. All above functionality provided by L2CAP creates overheads which are only applicable to ACL type traffic [19].

SCO links are typically used for un-framed transmission of in-stream, or a pure stream data. This kind of data does not use L2CAP channels and go directly to baseband logical link. The Bluetooth core system supports the direct transport of application data that is isochronous and of a constant rate, either bit-rate or frame-rate, using a SCO logical link. These logical links reserve physical channel bandwidth and provide a constant rate (64kbps) transport locked to the piconet clock. Data is transported in fixed size packets (in our case SCO packet size is 210 bytes which is send in one DH5 packet) at fixed intervals with both of these parameters negotiated during channel establishment. Also there is no QoS commitment and re-transmission in SCO. Therefore, it concluded that ACL transport has more overheads as compare to SCO transport [19].

In this chapter, we analyze the results from our three simulation models, and find that the best possible distance between a master and slave node(s) for ACL link is between 3 or 4 meters, while, it is up to 5m in case of SCO link. The best possible inquiry timeout is 10.24 S, because, at 20.48 S there is no gain in throughout, while there is wastage of time in extra long inquiry. The best possible start time for both SCO and ACL link slaves is 0.1 second. SCO link slaves give higher throughput as compared to ACL

link slaves, it does not matter whether, all slaves are SCO or they are in mixed environment of ACL and SCO link slaves.

5 Conclusion and Future Work

In this project efforts are made to find the best possible parameters to get maximum throughput after connection is established between Bluetooth the master and slaves. Studied parameters are: variable distances between master and slaves, inquiry timeout, start time, number of slaves and the combination of SCO and ACL link nodes in piconet.

It has been observed based on simulation results that:

- Best possible distance between a master and slave node for ACL link is 3 or 4 meters, while, in case of SCO link it could go up to 5 meters with no change in user data throughput.
- Best possible inquiry timeout is 10.24 S because at 20.48 S there is no gain in throughout, while unnecessary time is wasted in extra long inquiry.
- Best possible start time for both SCO and ACL link slaves is 0.1 second. Even at 0.5 second, throughput becomes unstable, because, some of the nodes start behaving abnormally.
- SCO link slaves give higher throughput as compared to ACL link slaves, it does not matter whether all slaves are SCO links or they are in mixed environment of ACL and SCO link slaves.

Below limitations / issues have been observed during the simulation:

- Free space model used in *Bluehoc* gives no penalty at distance less than 4m in ACL and up to 5m in SCO links. It gives little penalty at 5m and 6m in ACL and 6m in SCO links. The Author also confirms this observation in his thesis [14].
- No support for scatternets.
- Latest version of *Bluehoc* only supports simplex transmissions from master to slave.
- There is no support for setting up multi-slot packet.
- The position of the nodes in Network AniMator (NAM) trace output has no one to one correspondence with the position in Tcl execution scripts.

The following list summarizes the key issues to be considered by researchers in the field of Bluetooth simulation:

- Most of the non-commercially available simulators have no or very little support from their developers.
- It is important to make sure that researcher has access to operating system, with proper privileges, which is required to run simulator and all related tools. For example, if one wants to use *Bluehoc* and wants output in NAM format then he should make sure that he has access to Linux console not just access to terminal.
- Most of the simulators produce data that was useful for their developers. Make sure one understand the language that is used to developed simulator, because, modification of the source files may be required. In case of *Bluehoc* one need to understand C/C++, Tcl and Shell scripting languages.
- Limit data processing during the simulation, otherwise, it may impact simulation results. Instead of processing output data in simulation script, write a shell script that run once the simulation is over.
- Limit simulation run time to an optimal value, otherwise, useless data will be produce.

The contributions can be summarized as follows:

- Analyzed and found best possible parameters to maximized throughput of a Bluetooth piconet in post connection state.
- Found limitations in *Bluehoc* simulator, this will help researchers in future to make a wise decision, because, after reading this report they will be aware of the issues, limitations and challenges in Bluetooth simulation in general and *Bluehoc* in particular.

This simulation study can be further extended by using other simulators also scatternet is not studied during this project which can be studied as well.

References

[1] K. Sairam, N. Gunasekaran, et al, "*Bluetooth in wireless communication*", IEEE Communications Magazine, Volume 40, Issue 6, Jun 2002, pp. 90 – 96.

[2] P. McDermott-Wells, "*What is Bluetooth?*", IEEE Potentials, Volume 23, Issue 5, Dec. 2004-Jan. 2005, pp. 33 – 35.

[3] T. Godfrey, "*Blueware: Bluetooth Simulator for NS*", MIT Technical Report, http://nms.lcs.mit.edu/projects/blueware/, last accessed 18-December-2007.

[4] H. Labiod, H. Afifi, et al, "*Wi-Fi, Bluetooth, Zigbee and WiMax*", Publisher: Springer; 1 edition (Jun 25 2007), ISBN-10: 1402053967.

[5] B. A. Miller, C. Bisdikian, "*Bluetooth Revealed: An Insider's Guide to the Open Specification for Global Wireless Communications*", Publisher: Prentice Hall (3 Oct 2000) ISBN-10: 0130902942.

[6] Submitted proposal, "*Bluetooth Quality of Service.doc*", IEEE 1451.5 project has been approved by IEEE as a full-use standard - March 26, 2007. http://grouper.ieee.org/groups/1451/5/, last accessed 18-December-2007.

[7] Webopedia, http://www.webopedia.com/, last accessed 08-December-2007.

[8] B. Miller, "*Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer*", Bluetooth Consortium 1.C.118/1.0, July 1, 1999.

[9] P.S. Neelakanta and J. Sivaraks, "*A Novel Method to Mitigate Microwave Oven Dictaded EMI on Bluetooth Communications*", Microwave Journal, July 2001, pp. 70–88.

[10] T.W. Rondeau, M.F. D'Souza, "*Residential microwave oven interference on Bluetooth data performance*", IEEE Consumer Electronics, Volume 50, Issue 3, Aug. 2004, pp. 856 – 863.

[11] F. Kevin, "*The ns Manual (formerly ns Notes and Documentation)*", The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf, last accessed 18-December-2007.

[12] "*BlueHoc: Bluetooth Performance Evaluation Tool*" http://Bluehoc.sourceforge.net/ last accessed on last accessed 08-December-2007.

[13] K. Apurva, "*Bluehoc Manual*", IBM Corporation, http://bluehoc.sourceforge.net/tutorial/docpage.html#Section2, last accessed 14-December-2007.

[14] L. Martin, "*Evaluation of Bluetooth Communication: Simulation and Experiments*", DIKU tech report 02-03 (2002), University of Copenhagen, ISSN: 0107-8283.

[15] "*Nam: Network Animator Manual*", http://www.isi.edu/nsnam/nam/, last accessed 08-December-2007.

[16] G.V. Zaruba, I. Chlamtac, "*Accelerating Bluetooth inquiry for personal area network*", IEEE Global Telecommunications Conference, Volume 2, 1-5 Dec. 2003 pp. 702 – 706.

[17] Z. Xin, G.F. Riley, "*Bluetooth simulations for wireless sensor networks using GTNetS*", IEEE Computer Society 12th Annual International Symposium, Issue , 4-8 Oct. 2004, pp. 375 – 382.

[18] C. de M Cordeiro, D.P. Agrawal, "*Employing dynamic segmentation for effective co-located coexistence between Bluetooth and IEEE 802.11 WLANs*", IEEE Global Telecommunications Conference, Volume 1, 17-21 Nov. 2002, pp.195 - 200.

[19] "*Core Architecture*", http://www.bluetooth.com, last accessed 08-December-2007.

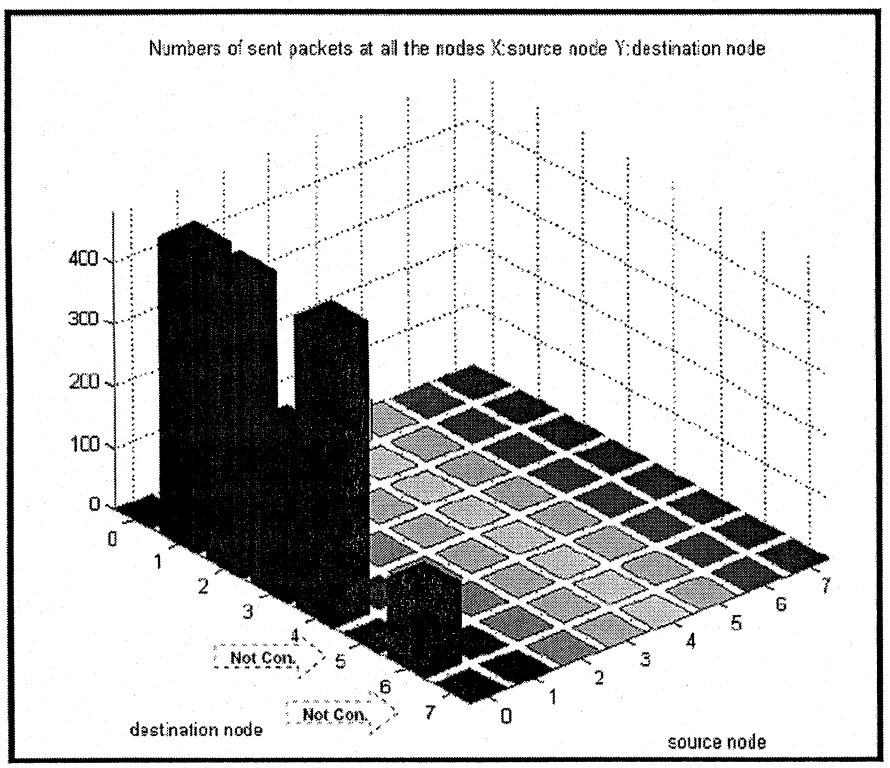
Annex (a)

Below three Figures are drawn from the IP traffic data collected by turning on following options in the simulation scripts.

---

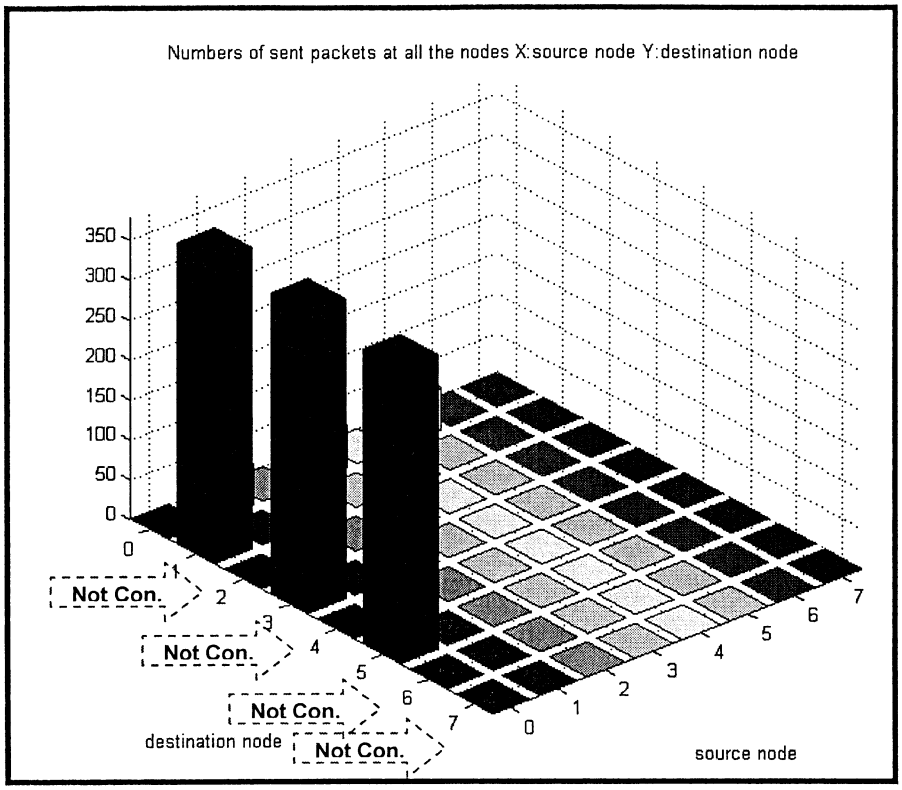
set Sim(Trace) ON
set Sim(AgentTrace) ON

These Figures show that IP level packets delivery and consequently connectivity (number of "NOT" connected slaves) are impacted by changing values of Bluetooth bearer application level parameters, such as, *StartTime, InqTimeout* and *SimulationTime.*



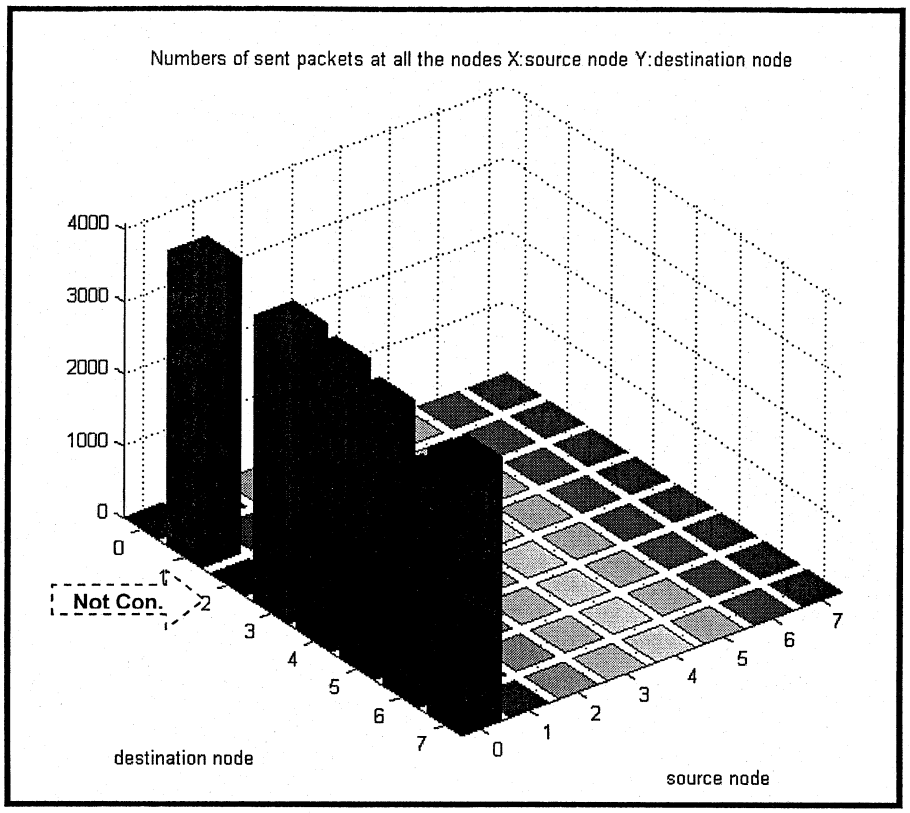
Above Figure is for simulation parameters:
StartTime 0.5 S
InqTimeout 32384
SimulationTime 20 S

Result: There are two "NOT" connected slaves.



Above Figure is for simulation parameters:
StartTime 1 S
InqTimeout 32384
SimulationTime 20 S

Result: There are four "NOT" connected slaves.



Above Figure is for simulation parameters:
StartTime 0.5 S
InqTimeout 64768
SimulationTime 40 S

Result: There is one "NOT" connected slave.

Annex (b)

Log files for abnormal behavior for simulation model # 2.1

Log from 5 slaves in system.

```
PAGE_MSG ****-->2          CLKN: 27388    clock: 9.559017e+00
PAGE_ACK 2-->0             CLKN: 30590    clock: 9.559647e+00
MASTER FROZEN              CLKE: 27390
FHS_PKT 0-->2              CLKN: 27393    clock: 9.560580e+00
FHS_ACK 2-->0              CLKN: 30594    clock: 9.560892e+00
POLL_ACK 2-->0             CLK:  30599    clock: 9.562455e+00
LMP_HOST_CONNECT_REQ       AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4          CLKN: 31517    clock: 1.184933e+01
PAGE_ACK 4-->0             CLKN: 37919    clock: 1.184996e+01
MASTER FROZEN              CLKE: 31519
FHS_PKT 0-->4              CLKN: 31521    clock: 1.185058e+01
FHS_ACK 4-->0              CLKN: 37922    clock: 1.185089e+01
POLL_ACK 4-->0             CLK:  37927    clock: 1.185245e+01
LMP_HOST_CONNECT_REQ       AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->5          CLKN: 31517    clock: 1.234933e+01
PAGE_ACK 5-->0             CLKN: 39519    clock: 1.234996e+01
MASTER FROZEN              CLKE: 31519
FHS_PKT 0-->5              CLKN: 31521    clock: 1.235058e+01
FHS_ACK 5-->0              CLKN: 39522    clock: 1.235089e+01
POLL_ACK 5-->0             CLK:  39527    clock: 1.235245e+01
LMP_HOST_CONNECT_REQ       AM_ADDR: 3
LMP_ACCEPTED               AM_ADDR: 3     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 3
LMP_ACCEPTED               AM_ADDR: 3     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3          CLKN: 39781    clock: 1.393183e+01
PAGE_ACK 3-->0             CLKN: 44583    clock: 1.393246e+01
MASTER FROZEN              CLKE: 39779
FHS_PKT 0-->3              CLKN: 39785    clock: 1.393308e+01
FHS_ACK 3-->0              CLKN: 44586    clock: 1.393339e+01
POLL_ACK 3-->0             CLK:  44591    clock: 1.393495e+01
```

Log from 6 slaves in system.

```
PAGE_MSG ****-->1          CLKN: 23257    clock: 7.768080e+00
PAGE_ACK 1-->0             CLKN: 24859    clock: 7.768710e+00
MASTER FROZEN              CLKE: 23259
FHS_PKT 0-->1              CLKN: 23261    clock: 7.769330e+00
FHS_ACK 1-->0              CLKN: 24862    clock: 7.769642e+00
POLL_ACK 1-->0             CLK:  24867    clock: 7.771205e+00
LMP_HOST_CONNECT_REQ       AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->2          CLKN: 27388    clock: 9.559017e+00
PAGE_ACK 2-->0             CLKN: 30590    clock: 9.559647e+00
MASTER FROZEN              CLKE: 27390
FHS_PKT 0-->2              CLKN: 27393    clock: 9.560580e+00
FHS_ACK 2-->0              CLKN: 30594    clock: 9.560892e+00
POLL_ACK 2-->0             CLK:  30599    clock: 9.562455e+00
LMP_HOST_CONNECT_REQ       AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4          CLKN: 31517    clock: 1.184933e+01
```

```
PAGE_ACK 4-->0            CLKN: 37919    clock: 1.184996e+01
MASTER FROZEN             CLKE: 31519
FHS_PKT 0-->4             CLKN: 31521    clock: 1.185058e+01
FHS_ACK 4-->0             CLKN: 37922    clock: 1.185089e+01
POLL_ACK 4-->0            CLK:  37927    clock: 1.185245e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 3
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 3
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3         CLKN: 39781    clock: 1.393183e+01
PAGE_ACK 3-->0            CLKN: 44583    clock: 1.393246e+01
MASTER FROZEN             CLKE: 39779
FHS_PKT 0-->3             CLKN: 39785    clock: 1.393308e+01
FHS_ACK 3-->0             CLKN: 44586    clock: 1.393339e+01
POLL_ACK 3-->0            CLK:  44591    clock: 1.393495e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 4
LMP_ACCEPTED              AM_ADDR: 4     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 4
LMP_ACCEPTED              AM_ADDR: 4     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->5         CLKN: 43908    clock: 1.622152e+01
PAGE_ACK 5-->0            CLKN: 51910    clock: 1.622215e+01
MASTER FROZEN             CLKE: 43910
FHS_PKT 0-->5             CLKN: 43913    clock: 1.622308e+01
FHS_ACK 5-->0             CLKN: 51914    clock: 1.622339e+01
POLL_ACK 5-->0            CLK:  51919    clock: 1.622495e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 5
LMP_ACCEPTED              AM_ADDR: 5     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 5
LMP_ACCEPTED              AM_ADDR: 5     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->6         CLKN: 43908    clock: 1.672152e+01
PAGE_ACK 6-->0            CLKN: 53510    clock: 1.672215e+01
MASTER FROZEN             CLKE: 43910
FHS_PKT 0-->6             CLKN: 43913    clock: 1.672308e+01
FHS_ACK 6-->0             CLKN: 53514    clock: 1.672339e+01
POLL_ACK 6-->0            CLK:  53519    clock: 1.672495e+01
```

Log from 7 slaves in system

```
PAGE_MSG ****-->1         CLKN: 31517    clock: 1.034933e+01
PAGE_ACK 1-->0            CLKN: 33119    clock: 1.034996e+01
MASTER FROZEN             CLKE: 31519
FHS_PKT 0-->1             CLKN: 31521    clock: 1.035058e+01
FHS_ACK 1-->0             CLKN: 33122    clock: 1.035089e+01
POLL_ACK 1-->0            CLK:  33127    clock: 1.035245e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 1
LMP_ACCEPTED              AM_ADDR: 1     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 1
LMP_ACCEPTED              AM_ADDR: 1     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4         CLKN: 27388    clock: 1.055902e+01
PAGE_ACK 4-->0            CLKN: 33790    clock: 1.055965e+01
MASTER FROZEN             CLKE: 27390
FHS_PKT 0-->4             CLKN: 27393    clock: 1.056058e+01
FHS_ACK 4-->0             CLKN: 33794    clock: 1.056089e+01
POLL_ACK 4-->0            CLK:  33799    clock: 1.056245e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 2
LMP_ACCEPTED              AM_ADDR: 2     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 2
LMP_ACCEPTED              AM_ADDR: 2     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->2         CLKN: 34692    clock: 1.184152e+01
PAGE_ACK 2-->0            CLKN: 37894    clock: 1.184215e+01
MASTER FROZEN             CLKE: 34690
FHS_PKT 0-->2             CLKN: 34697    clock: 1.184308e+01
FHS_ACK 2-->0             CLKN: 37898    clock: 1.184339e+01
POLL_ACK 2-->0            CLK:  37903    clock: 1.184495e+01
LMP_HOST_CONNECT_REQ      AM_ADDR: 3
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 3
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3         CLKN: 48041    clock: 1.651308e+01
```

PAGE_ACK 3-->0	CLKN: 52843	clock: 1.651371e+01
MASTER FROZEN	CLKE: 48039	
FHS_PKT 0-->3	CLKN: 48045	clock: 1.651433e+01
FHS_ACK 3-->0	CLKN: 52846	clock: 1.651464e+01
POLL_ACK 3-->0	CLK: 52851	clock: 1.651620e+01
LMP_HOST_CONNECT_REQ	AM_ADDR: 4	
LMP_ACCEPTED	AM_ADDR: 4	ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ	AM_ADDR: 4	
LMP_ACCEPTED	AM_ADDR: 4	ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->6	CLKN: 43908	clock: 1.672152e+01
PAGE_ACK 6-->0	CLKN: 53510	clock: 1.672215e+01
MASTER FROZEN	CLKE: 43910	
FHS_PKT 0-->6	CLKN: 43913	clock: 1.672308e+01
FHS_ACK 6-->0	CLKN: 53514	clock: 1.672339e+01
POLL_ACK 6-->0	CLK: 53519	clock: 1.672495e+01

Annex (c)

In total 201 scripts are used during this simulation study. Simulation script for 1 ACL link slave at a distance of 3m, 4m, 5 and 6m are included below as samples. Moreover, proc.tcl and run.tcl sub-scripts that are called by all scripts are also included.

Script file for 3m @ 0.1 StartTime

```
set Sim(Trace) OFF
set Sim(AgentTrace) OFF
set Sim(NumDevices) 2
set Sim(Transport) [list TCP/Reno]

set Sim(xpos_) [list 5.42 8.42]
set Sim(ypos_) [list 3.82 3.74]

set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses 1

set SimulationTime 20.0
set StartTime [list 0 .1]

set InqScanOffset [list 3200]

source ./proc.Tcl
source ./run.Tcl
```

Script file for 4m @ 0.1 StartTime

```
set Sim(Trace) OFF
set Sim(AgentTrace) OFF
set Sim(NumDevices) 2
set Sim(Transport) [list TCP/Reno]

set Sim(xpos_) [list 5.34 9.38]
set Sim(ypos_) [list 3.84 3.68]

set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses 1

set SimulationTime 20.0
set StartTime [list 0 .1]

set InqScanOffset [list 3200]

source ./proc.Tcl
source ./run.Tcl
```

Script file for 5m @ 0.1 StartTime

```
set Sim(Trace) OFF
set Sim(AgentTrace) OFF
set Sim(NumDevices) 2
set Sim(Transport) [list TCP/Reno]

set Sim(xpos_) [list 2.66 7.7]
set Sim(ypos_) [list 3.78 3.9]

set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses 1

set SimulationTime 20.0
set StartTime [list 0 .1]
```

```
set InqScanOffset [list 3200]

source ./proc.Tcl
source ./run.Tcl
```

Script file for 6m @ 0.1 StartTime

```
set Sim(Trace) OFF
set Sim(AgentTrace) OFF
set Sim(NumDevices) 2
set Sim(Transport) [list TCP/Reno]

set Sim(xpos_) [list 1.54 7.58]
set Sim(ypos_) [list 4.02 4.06]

set Sim(Application) [list FTP]
set InqTimeout 32384
set NumResponses 1

set SimulationTime 20.0
set StartTime [list 0 .1]

set InqScanOffset [list 3200]

source ./proc.Tcl
source ./run.Tcl
```

Script file for proc.tcl

```
proc create-master {num_resp inqTO} {
        global Sim
        set Sim(dev:0) [new BTNode 0 0 1 [lindex $Sim(xpos_) 0] [lindex $Sim(ypos_) 0]]
        $Sim(dev:0) set-target $Sim(repl)
        $Sim(repl) install 0 [$Sim(dev:0) set lm_]
        set bthost [$Sim(dev:0) set bthost_]
        $bthost set numResponses_ $num_resp
        $bthost set inqTimeout_ $inqTO
        if {$Sim(Trace) == "ON"} {
                $Sim(dev:0) color "red"
                $Sim(dev:0) shape "circle"
                $Sim(dev:0) dump-namconfig
        }
}

proc create-slave {index inqScanOffset} {
        global Sim
        set Sim(dev:$index) [new BTNode $index $index 0 [lindex $Sim(xpos_) $index] [lindex $Sim(ypos_) $index]]
        $Sim(dev:$index) setISoffset $inqScanOffset
        $Sim(dev:$index) set-target $Sim(repl)
        $Sim(repl) install $index [$Sim(dev:$index) set lm_]
        if {$Sim(Trace) == "ON"} {
                $Sim(dev:$index) color "red"
                $Sim(dev:$index) shape "circle"
                $Sim(dev:$index) dump-namconfig
        }
}

proc config-trace {} {
        global ns Sim
        set nam_fp $Sim(namFP)
        set tr $Sim(trFP)
        $ns trace-all $tr
        for {set i 0} {$i < $Sim(NumDevices)} {incr i} {
                if { $i!=0 && $Sim(AgentTrace) == "ON"} {
                        set transport [lindex $Sim(Transport) [expr $i-1]]
                        set temp [string range $transport 0 2]
                        if {$temp == "TCP"} {
```

```
                                        $ns add-agent-trace $Sim(trans:[expr $i-1]) tcp$i
                                        $ns monitor-agent-trace $Sim(trans:[expr $i-1])
                                        $Sim(trans:[expr $i-1]) tracevar cwnd_
                                }
                        }

                        set Sim(trace:$i) [$ns create-trace Deque $tr $Sim(dev:0) $Sim(dev:$i)]
                        $Sim(trace:$i) namattach $nam_fp
                        $Sim(repl) install $i $Sim(trace:$i)
                        $Sim(trace:$i) target [$Sim(dev:$i) set lm_]
                        set dropT($i) [$ns create-trace Drop $tr $Sim(dev:0) $Sim(dev:$i)]
                        $dropT($i) namattach $nam_fp
                        $Sim(dev:$i) set-droptarget $dropT($i)
                        set nodetrace($i) [new Trace/BTNodeColor 0]
                        $nodetrace($i) namattach $nam_fp
                        $Sim(dev:$i) set-target $nodetrace($i)
                        $nodetrace($i) target $Sim(repl)
                        $nodetrace($i) set src_ $i
                }
}

proc config-app {i} {
        global Sim ns
        set transport [lindex $Sim(Transport) $i]
        set Sim(trans:$i) [new Agent/$transport]
        set app [lindex $Sim(Application) $i]
        set Sim(appl:$i) [new Application/$app]
        set temp [string range $transport 0 2]
        if {$temp == "TCP"} {
                set Sim(sink:$i) [new Agent/TCPSink]
        } else {
                set Sim(sink:$i) [new Agent/Null]
        }
        $ns attach-agent $Sim(dev:0) $Sim(trans:$i)
        $ns attach-agent $Sim(dev:[expr $i+1]) $Sim(sink:$i)
        $ns connect $Sim(trans:$i) $Sim(sink:$i)
        $Sim(dev:0) attach-app $i $Sim(appl:$i)
        $Sim(appl:$i) attach-agent $Sim(trans:$i)
}
```

Script file for run.tcl

```
set NumSlaves [expr $Sim(NumDevices)-1]
set ns [new Simulator]

if {$Sim(Trace) == "ON"} {
        Baseband set namTrace_ 1
        Trace set namBTTrace_ 1
        set Sim(namFP) [open $Sim(NamFile) w]
        set Sim(trFP)  [open out.tr w]
        $ns namtrace-all $Sim(namFP)
}

set Sim(repl) [new Classifier/Replicator]

create-master $NumResponses $InqTimeout

for {set i 0} {$i < $NumSlaves} {incr i} {
        create-slave [expr $i+1] [lindex $InqScanOffset $i]
}

for {set i 0} {$i < $NumSlaves} {incr i} {
        config-app $i
}

if {$Sim(Trace) == "ON"} {
        config-trace
        for {set i 0} {$i < $NumSlaves} {incr i} {
```

```
                puts $Sim(namFP) "l -t * -s 0 -d [expr $i+1] -S UP -r 5000000 -D .0005 -c black -o [expr 105+$i*30]deg"
        }
}

for {set j 0} {$j < $Sim(NumDevices)} {incr j} {
        set bthost [$Sim(dev:$j) set bthost_]
        $ns at [lindex $StartTime $j] "$bthost start"
}

$ns at $SimulationTime "finish"

proc finish {} {
        global Sim
        puts "Simulation over"
        if {$Sim(Trace) == "ON"} {
                puts "Starting nam..."
                exec nam $Sim(NamFile) &
        }
        exit 0
}

$ns run
```

Annex (d)

Following three log files are: file (a) all ACL links, file (b) all SCO links and file (c) 3SCO and 2ACL links.

File (a): All ACL links

```
appname[0] = Application/FTP
appname[1] = Application/FTP
appname[2] = Application/FTP
appname[3] = Application/FTP
appname[4] = Application/FTP
INQ_MSG ****-->1   CLKN: 3216     clock: 1.105268e+00
INQ_MSG AFTER BO *-->1     CLKN: 3344     clock: 1.145268e+00
FHS_PKT 1-->0              CLKN: 3667     clock: 1.146210e+00
INQ_MSG ****-->1   CLKN: 3377     clock: 1.155580e+00
INQ_MSG ****-->2   CLKN: 3216     clock: 1.205268e+00
INQ_MSG ****-->3   CLKN: 3216     clock: 1.305268e+00
INQ_MSG AFTER BO *-->3     CLKN: 3344     clock: 1.345268e+00
FHS_PKT 3-->0              CLKN: 4307     clock: 1.346210e+00
INQ_MSG ****-->3   CLKN: 3377     clock: 1.355580e+00
INQ_MSG ****-->4   CLKN: 3216     clock: 1.405267e+00
INQ_MSG AFTER BO *-->1     CLKN: 4177     clock: 1.405580e+00
FHS_PKT 1-->0              CLKN: 4500     clock: 1.406522e+00
INQ_MSG ****-->1   CLKN: 4212     clock: 1.416517e+00
INQ_MSG ****-->5   CLKN: 3216     clock: 1.505267e+00
INQ_MSG AFTER BO *-->5     CLKN: 3408     clock: 1.565267e+00
FHS_PKT 5-->0              CLKN: 5011     clock: 1.566210e+00
INQ_MSG ****-->5   CLKN: 3441     clock: 1.575580e+00
INQ_MSG AFTER BO *-->3     CLKN: 4177     clock: 1.605580e+00
FHS_PKT 3-->0              CLKN: 5140     clock: 1.606522e+00
INQ_MSG ****-->3   CLKN: 4212     clock: 1.616517e+00
INQ_MSG AFTER BO *-->4     CLKN: 4080     clock: 1.675267e+00
FHS_PKT 4-->0              CLKN: 5363     clock: 1.676210e+00
INQ_MSG ****-->4   CLKN: 4113     clock: 1.685580e+00
INQ_MSG AFTER BO *-->2     CLKN: 4944     clock: 1.745267e+00
FHS_PKT 2-->0              CLKN: 5587     clock: 1.746210e+00
PAGE_MSG ****-->1          CLKN: 6705     clock: 2.195580e+00
PAGE_ACK 1-->0             CLKN: 7027     clock: 2.196210e+00
MASTER FROZEN              CLKE: 6707
FHS_PKT 0-->1              CLKN: 6709     clock: 2.196830e+00
FHS_ACK 1-->0              CLKN: 7030     clock: 2.197142e+00
POLL_ACK 1-->0             CLK:  7035     clock: 2.198705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 1
LMP_ACCEPTED               AM_ADDR: 1     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3          CLKN: 6705     clock: 2.395580e+00
PAGE_ACK 3-->0             CLKN: 7667     clock: 2.396210e+00
MASTER FROZEN              CLKE: 6707
FHS_PKT 0-->3              CLKN: 6709     clock: 2.396830e+00
FHS_ACK 3-->0              CLKN: 7670     clock: 2.397143e+00
POLL_ACK 3-->0             CLK:  7675     clock: 2.398705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                AM_ADDR: 2
LMP_ACCEPTED               AM_ADDR: 2     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->5          CLKN: 6705     clock: 2.595580e+00
PAGE_ACK 5-->0             CLKN: 8307     clock: 2.596210e+00
MASTER FROZEN              CLKE: 6707
FHS_PKT 0-->5              CLKN: 6709     clock: 2.596830e+00
FHS_ACK 5-->0              CLKN: 8310     clock: 2.597143e+00
POLL_ACK 5-->0             CLK:  8315     clock: 2.598705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 3
```

```
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 3
LMP_ACCEPTED              AM_ADDR: 3     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4         CLKN: 14965    clock: 5.076830e+00
PAGE_ACK 4-->0            CLKN: 16247    clock: 5.077460e+00
MASTER FROZEN             CLKE: 14967
FHS_PKT 0-->4             CLKN: 14969    clock: 5.078080e+00
FHS_ACK 4-->0             CLKN: 16250    clock: 5.078392e+00
POLL_ACK 4-->0            CLK:  16255    clock: 5.079955e+00
LMP_HOST_CONNECT_REQ      AM_ADDR: 4
LMP_ACCEPTED              AM_ADDR: 4     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 4
LMP_ACCEPTED              AM_ADDR: 4     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->2         CLKN: 23257    clock: 7.468080e+00
PAGE_ACK 2-->0            CLKN: 23899    clock: 7.468710e+00
MASTER FROZEN             CLKE: 23259
FHS_PKT 0-->2             CLKN: 23261    clock: 7.469330e+00
FHS_ACK 2-->0             CLKN: 23902    clock: 7.469642e+00
POLL_ACK 2-->0            CLK:  23907    clock: 7.471205e+00
LMP_HOST_CONNECT_REQ      AM_ADDR: 5
LMP_ACCEPTED              AM_ADDR: 5     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ               AM_ADDR: 5
LMP_ACCEPTED              AM_ADDR: 5     ACCEPTED PDU: LMP_QOS_REQ
RECV L2CAP_CONNECT_REQ          CH: 4
RECV L2CAP_CONNECT_RSP CH: 4
L2CA_DATA_WRITE                 SIZE: 1000     CID: 6          DEST_IP 2 clock 7.476205e+00
RECV L2CAP_CONNECT_REQ          CH: 0
RECV L2CAP_CONNECT_RSP CH: 0
L2CA_DATA_WRITE                 SIZE: 1000     CID: 2          DEST_IP 1 clock 7.481205e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
BD_ADDR 1 DELAY 1.062500e-02 SIZE 1000 clock 7.491830e+00
L2CA_DATA_READ          SIZE: 40       CID: 2
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.493705e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 2          DEST_IP 1 clock 7.493705e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 2          DEST_IP 1 clock 7.493705e+00
RECV L2CAP_CONNECT_REQ          CH: 1
RECV L2CAP_CONNECT_RSP CH: 1
L2CA_DATA_WRITE                 SIZE: 1000     CID: 3          DEST_IP 3 clock 7.494955e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
BD_ADDR 3 DELAY 1.062500e-02 SIZE 1000 clock 7.505580e+00
L2CA_DATA_READ          SIZE: 40       CID: 3
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.507455e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 3          DEST_IP 3 clock 7.507455e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 3          DEST_IP 3 clock 7.507455e+00
RECV L2CAP_CONNECT_REQ          CH: 2
RECV L2CAP_CONNECT_RSP CH: 2
L2CA_DATA_WRITE                 SIZE: 1000     CID: 4          DEST_IP 5 clock 7.508705e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
BD_ADDR 5 DELAY 1.062500e-02 SIZE 1000 clock 7.519330e+00
L2CA_DATA_READ          SIZE: 40       CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.521205e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 4          DEST_IP 5 clock 7.521205e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 4          DEST_IP 5 clock 7.521205e+00
RECV L2CAP_CONNECT_REQ          CH: 3
RECV L2CAP_CONNECT_RSP CH: 3
L2CA_DATA_WRITE                 SIZE: 1000     CID: 5          DEST_IP 4 clock 7.522455e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
BD_ADDR 1 DELAY 4.687500e-02 SIZE 1000 clock 7.540580e+00
L2CA_DATA_READ          SIZE: 40       CID: 2
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.542455e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 2          DEST_IP 1 clock 7.542455e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 2          DEST_IP 1 clock 7.542455e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
BD_ADDR 3 DELAY 4.562500e-02 SIZE 1000 clock 7.553080e+00
L2CA_DATA_READ          SIZE: 40       CID: 3
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.554955e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 3          DEST_IP 3 clock 7.554955e+00
L2CA_DATA_WRITE                 SIZE: 1000     CID: 3          DEST_IP 3 clock 7.554955e+00
L2CA_DATA_READ          SIZE: 1000     CID: 2
```

```
BD_ADDR 5 DELAY 4.437500e-02 SIZE 1000 clock 7.565580e+00
L2CA_DATA_READ          SIZE: 40      CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.567455e+00
L2CA_DATA_WRITE                 SIZE: 1000    CID: 4      DEST_IP 5 clock 7.567455e+00
L2CA_DATA_WRITE                 SIZE: 1000    CID: 4      DEST_IP 5 clock 7.567455e+00
L2CA_DATA_READ          SIZE: 1000    CID: 2
BD_ADDR 4 DELAY 5.187500e-02 SIZE 1000 clock 7.574330e+00
L2CA_DATA_READ          SIZE: 40      CID: 5
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.576205e+00
L2CA_DATA_WRITE                 SIZE: 1000    CID: 5      DEST_IP 4 clock 7.576205e+00
L2CA_DATA_WRITE                 SIZE: 1000    CID: 5      DEST_IP 4 clock 7.576205e+00
L2CA_DATA_READ          SIZE: 1000    CID: 2
BD_ADDR 1 DELAY 9.312500e-02 SIZE 1000 clock 7.586830e+00
.
.
. ***Actual file has 5191 lies of data*****************
.
.
L2CA_DATA_READ          SIZE: 1000    CID: 2
BD_ADDR 1 DELAY 9.518750e-01 SIZE 1000 clock 1.999433e+01
L2CA_DATA_READ          SIZE: 40      CID: 2
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 1.999620e+01
L2CA_DATA_WRITE                 SIZE: 1000    CID: 2      DEST_IP 1 clock 1.999620e+01
```

File (b): All SCO links

```
appname[0] = Application/Traffic/Exponential
appname[1] = Application/Traffic/Exponential
appname[2] = Application/Traffic/Exponential
appname[3] = Application/Traffic/Exponential
appname[4] = Application/Traffic/Exponential
INQ_MSG ****-->1   CLKN: 3216     clock: 1.105268e+00
INQ_MSG AFTER BO *-->1     CLKN: 3504     clock: 1.195268e+00
FHS_PKT 1-->0              CLKN: 3827     clock: 1.196210e+00
INQ_MSG ****-->2   CLKN: 3216     clock: 1.205268e+00
INQ_MSG ****-->1   CLKN: 3537     clock: 1.205580e+00
INQ_MSG ****-->3   CLKN: 3216     clock: 1.305268e+00
INQ_MSG AFTER BO *-->2     CLKN: 3664     clock: 1.345268e+00
FHS_PKT 2-->0              CLKN: 4307     clock: 1.346210e+00
INQ_MSG ****-->2   CLKN: 3697     clock: 1.355580e+00
INQ_MSG AFTER BO *-->2     CLKN: 3793     clock: 1.385580e+00
FHS_PKT 2-->0              CLKN: 4436     clock: 1.386522e+00
INQ_MSG ****-->2   CLKN: 3828     clock: 1.396517e+00
INQ_MSG ****-->4   CLKN: 3216     clock: 1.405267e+00
INQ_MSG AFTER BO *-->2     CLKN: 3860     clock: 1.406517e+00
FHS_PKT 2-->0              CLKN: 4503     clock: 1.407460e+00
INQ_MSG ****-->2   CLKN: 3893     clock: 1.416830e+00
INQ_MSG AFTER BO *-->4     CLKN: 3376     clock: 1.455267e+00
FHS_PKT 4-->0              CLKN: 4659     clock: 1.456210e+00
INQ_MSG ****-->4   CLKN: 3409     clock: 1.465580e+00
INQ_MSG ****-->5   CLKN: 3216     clock: 1.505267e+00
INQ_MSG AFTER BO *-->1     CLKN: 4945     clock: 1.645580e+00
FHS_PKT 1-->0              CLKN: 5268     clock: 1.646522e+00
INQ_MSG ****-->1   CLKN: 4980     clock: 1.656517e+00
INQ_MSG AFTER BO *-->1     CLKN: 5108     clock: 1.696517e+00
FHS_PKT 1-->0              CLKN: 5431     clock: 1.697460e+00
INQ_MSG ****-->1   CLKN: 5141     clock: 1.706830e+00
INQ_MSG AFTER BO *-->4     CLKN: 4753     clock: 1.885580e+00
FHS_PKT 4-->0              CLKN: 6036     clock: 1.886522e+00
INQ_MSG ****-->4   CLKN: 4788     clock: 1.896517e+00
INQ_MSG AFTER BO *-->3     CLKN: 5136     clock: 1.905267e+00
FHS_PKT 3-->0              CLKN: 6099     clock: 1.906210e+00
INQ_MSG ****-->3   CLKN: 5169     clock: 1.915580e+00
INQ_MSG AFTER BO *-->5     CLKN: 4656     clock: 1.955267e+00
FHS_PKT 5-->0              CLKN: 6259     clock: 1.956210e+00
PAGE_MSG ****-->1          CLKN: 6705     clock: 2.195580e+00
PAGE_ACK 1-->0             CLKN: 7027     clock: 2.196210e+00
MASTER FROZEN              CLKE: 6707
```

```
FHS_PKT 0-->1                CLKN: 6709     clock: 2.196830e+00
FHS_ACK 1-->0                CLKN: 7030     clock: 2.197142e+00
POLL_ACK 1-->0               CLK:  7035     clock: 2.198705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 1
LMP_ACCEPTED                 AM_ADDR: 1     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 1
LMP_ACCEPTED                 AM_ADDR: 1     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->2            CLKN: 6705     clock: 2.295580e+00
PAGE_ACK 2-->0               CLKN: 7347     clock: 2.296210e+00
MASTER FROZEN                CLKE: 6707
FHS_PKT 0-->2                CLKN: 6709     clock: 2.296830e+00
FHS_ACK 2-->0                CLKN: 7350     clock: 2.297143e+00
POLL_ACK 2-->0               CLK:  7355     clock: 2.298705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 2
LMP_ACCEPTED                 AM_ADDR: 2     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 2
LMP_ACCEPTED                 AM_ADDR: 2     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4            CLKN: 6705     clock: 2.495580e+00
PAGE_ACK 4-->0               CLKN: 7987     clock: 2.496210e+00
MASTER FROZEN                CLKE: 6707
FHS_PKT 0-->4                CLKN: 6709     clock: 2.496830e+00
FHS_ACK 4-->0                CLKN: 7990     clock: 2.497143e+00
POLL_ACK 4-->0               CLK:  7995     clock: 2.498705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 3
LMP_ACCEPTED                 AM_ADDR: 3     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 3
LMP_ACCEPTED                 AM_ADDR: 3     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3            CLKN: 14965    clock: 4.976830e+00
PAGE_ACK 3-->0               CLKN: 15927    clock: 4.977460e+00
MASTER FROZEN                CLKE: 14967
FHS_PKT 0-->3                CLKN: 14969    clock: 4.978080e+00
FHS_ACK 3-->0                CLKN: 15930    clock: 4.978393e+00
POLL_ACK 3-->0               CLK:  15935    clock: 4.979955e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 4
LMP_ACCEPTED                 AM_ADDR: 4     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 4
LMP_ACCEPTED                 AM_ADDR: 4     ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->5            CLKN: 14965    clock: 5.176830e+00
PAGE_ACK 5-->0               CLKN: 16567    clock: 5.177460e+00
MASTER FROZEN                CLKE: 14967
FHS_PKT 0-->5                CLKN: 14969    clock: 5.178080e+00
FHS_ACK 5-->0                CLKN: 16570    clock: 5.178392e+00
POLL_ACK 5-->0               CLK:  16575    clock: 5.179955e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 5
LMP_ACCEPTED                 AM_ADDR: 5     ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 5
LMP_ACCEPTED                 AM_ADDR: 5     ACCEPTED PDU: LMP_QOS_REQ
RECV L2CAP_CONNECT_REQ       CH: 4
RECV L2CAP_CONNECT_RSP CH: 4
RECV L2CAP_CONNECT_REQ       CH: 0
RECV L2CAP_CONNECT_RSP CH: 0
RECV L2CAP_CONNECT_REQ       CH: 1
RECV L2CAP_CONNECT_RSP CH: 1
RECV L2CAP_CONNECT_REQ       CH: 2
RECV L2CAP_CONNECT_RSP CH: 2
RECV L2CAP_CONNECT_REQ       CH: 3
RECV L2CAP_CONNECT_RSP CH: 3
L2CA_DATA_WRITE              SIZE: 210      CID: 2          DEST_IP 1 clock 5.299669e+00
L2CA_DATA_READ        SIZE: 210      CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.303080e+00
L2CA_DATA_WRITE              SIZE: 210      CID: 2          DEST_IP 1 clock 5.325919e+00
L2CA_DATA_READ        SIZE: 210      CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.329330e+00
L2CA_DATA_WRITE              SIZE: 210      CID: 2          DEST_IP 1 clock 5.352169e+00
L2CA_DATA_READ        SIZE: 210      CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.355580e+00
L2CA_DATA_WRITE              SIZE: 210      CID: 2          DEST_IP 1 clock 5.378419e+00
L2CA_DATA_READ        SIZE: 210      CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.381830e+00
```

```
L2CA_DATA_WRITE                 SIZE: 210       CID: 2          DEST_IP 1 clock 5.404669e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.408080e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 5          DEST_IP 3 clock 5.413047e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 3 DELAY 3.782644e-03 SIZE 210 clock 5.416830e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 2          DEST_IP 1 clock 5.430919e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.434330e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 5          DEST_IP 3 clock 5.439297e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 3 DELAY 3.782644e-03 SIZE 210 clock 5.443080e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 2          DEST_IP 1 clock 5.457169e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.460580e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 5          DEST_IP 3 clock 5.465547e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 3 DELAY 3.782644e-03 SIZE 210 clock 5.469330e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 2          DEST_IP 1 clock 5.483419e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 1 DELAY 3.410890e-03 SIZE 210 clock 5.486830e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 5          DEST_IP 3 clock 5.491797e+00
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 3 DELAY 3.782644e-03 SIZE 210 clock 5.495580e+00
L2CA_DATA_WRITE                 SIZE: 210       CID: 2          DEST_IP 1 clock 5.509669e+00


.
.
.*** Actual file has 4290 lines of data***************
.
.
.
BD_ADDR 5 DELAY 3.167506e-03 SIZE 210 clock 1.993933e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 4 DELAY 5.084603e-03 SIZE 210 clock 1.994308e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 3          DEST_IP 2 clock 1.995596e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 2 DELAY 3.373446e-03 SIZE 210 clock 1.995933e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 6          DEST_IP 5 clock 1.996241e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 4          DEST_IP 4 clock 1.996425e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 5 DELAY 3.167506e-03 SIZE 210 clock 1.996558e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 4 DELAY 5.084603e-03 SIZE 210 clock 1.996933e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 3          DEST_IP 2 clock 1.998221e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 2 DELAY 3.373446e-03 SIZE 210 clock 1.998558e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 6          DEST_IP 5 clock 1.998866e+01
L2CA_DATA_WRITE                 SIZE: 210       CID: 4          DEST_IP 4 clock 1.999050e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 5 DELAY 3.167506e-03 SIZE 210 clock 1.999183e+01
L2CA_DATA_READ          SIZE: 210       CID: 2
BD_ADDR 4 DELAY 5.084603e-03 SIZE 210 clock 1.999558e+01
```

File (c): Three SCO and two ACL links

```
appname[0] = Application/Traffic/Exponential
appname[1] = Application/Traffic/Exponential
appname[2] = Application/Traffic/Exponential
appname[3] = Application/FTP
appname[4] = Application/FTP
INQ_MSG ****-->1   CLKN: 3216      clock: 1.105268e+00
INQ_MSG ****-->2   CLKN: 3216      clock: 1.205268e+00
INQ_MSG ****-->3   CLKN: 3216      clock: 1.305268e+00
INQ_MSG AFTER BO *-->2      CLKN: 3664      clock: 1.345268e+00
FHS_PKT 2-->0               CLKN: 4307      clock: 1.346210e+00
INQ_MSG ****-->2   CLKN: 3697      clock: 1.355580e+00
```

```
INQ_MSG AFTER BO *-->1      CLKN: 4176      clock: 1.405267e+00
INQ_MSG ****-->4   CLKN: 3216      clock: 1.405267e+00
FHS_PKT 1-->0               CLKN: 4499      clock: 1.406210e+00
INQ_MSG ****-->1   CLKN: 4209      clock: 1.415580e+00
INQ_MSG AFTER BO *-->1      CLKN: 4241      clock: 1.425580e+00
FHS_PKT 1-->0               CLKN: 4564      clock: 1.426522e+00
INQ_MSG ****-->1   CLKN: 4276      clock: 1.436517e+00
INQ_MSG AFTER BO *-->4      CLKN: 3344      clock: 1.445267e+00
FHS_PKT 4-->0               CLKN: 4627      clock: 1.446210e+00
INQ_MSG ****-->4   CLKN: 3377      clock: 1.455580e+00
INQ_MSG ****-->5   CLKN: 3216      clock: 1.505267e+00
INQ_MSG AFTER BO *-->2      CLKN: 4785      clock: 1.695580e+00
FHS_PKT 2-->0               CLKN: 5428      clock: 1.696522e+00
INQ_MSG ****-->2   CLKN: 4820      clock: 1.706517e+00
INQ_MSG AFTER BO *-->3      CLKN: 4624      clock: 1.745267e+00
FHS_PKT 3-->0               CLKN: 5587      clock: 1.746210e+00
INQ_MSG ****-->3   CLKN: 4657      clock: 1.755580e+00
INQ_MSG AFTER BO *-->4      CLKN: 4465      clock: 1.795580e+00
FHS_PKT 4-->0               CLKN: 5748      clock: 1.796522e+00
INQ_MSG ****-->4   CLKN: 4500      clock: 1.806517e+00
INQ_MSG AFTER BO *-->1      CLKN: 5684      clock: 1.876517e+00
FHS_PKT 1-->0               CLKN: 6007      clock: 1.877460e+00
INQ_MSG ****-->1   CLKN: 5717      clock: 1.886830e+00
INQ_MSG AFTER BO *-->5      CLKN: 4560      clock: 1.925267e+00
FHS_PKT 5-->0               CLKN: 6163      clock: 1.926210e+00
PAGE_MSG ****-->2           CLKN: 6705      clock: 2.295580e+00
PAGE_ACK 2-->0              CLKN: 7347      clock: 2.296210e+00
MASTER FROZEN               CLKE: 6707
FHS_PKT 0-->2               CLKN: 6709      clock: 2.296830e+00
FHS_ACK 2-->0               CLKN: 7350      clock: 2.297143e+00
POLL_ACK 2-->0              CLK:  7355      clock: 2.298705e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 1
LMP_ACCEPTED                AM_ADDR: 1      ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                 AM_ADDR: 1
LMP_ACCEPTED                AM_ADDR: 1      ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->1           CLKN: 14965     clock: 4.776830e+00
PAGE_ACK 1-->0              CLKN: 15287     clock: 4.777460e+00
MASTER FROZEN               CLKE: 14967
FHS_PKT 0-->1               CLKN: 14969     clock: 4.778080e+00
FHS_ACK 1-->0               CLKN: 15290     clock: 4.778393e+00
POLL_ACK 1-->0              CLK:  15295     clock: 4.779955e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 2
LMP_ACCEPTED                AM_ADDR: 2      ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                 AM_ADDR: 2
LMP_ACCEPTED                AM_ADDR: 2      ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->4           CLKN: 14965     clock: 5.076830e+00
PAGE_ACK 4-->0              CLKN: 16247     clock: 5.077460e+00
MASTER FROZEN               CLKE: 14967
FHS_PKT 0-->4               CLKN: 14969     clock: 5.078080e+00
FHS_ACK 4-->0               CLKN: 16250     clock: 5.078392e+00
POLL_ACK 4-->0              CLK:  16255     clock: 5.079955e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 3
LMP_ACCEPTED                AM_ADDR: 3      ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                 AM_ADDR: 3
LMP_ACCEPTED                AM_ADDR: 3      ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->3           CLKN: 23257     clock: 7.568080e+00
PAGE_ACK 3-->0              CLKN: 24219     clock: 7.568710e+00
MASTER FROZEN               CLKE: 23259
FHS_PKT 0-->3               CLKN: 23261     clock: 7.569330e+00
FHS_ACK 3-->0               CLKN: 24222     clock: 7.569642e+00
POLL_ACK 3-->0              CLK:  24227     clock: 7.571205e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 4
LMP_ACCEPTED                AM_ADDR: 4      ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                 AM_ADDR: 4
LMP_ACCEPTED                AM_ADDR: 4      ACCEPTED PDU: LMP_QOS_REQ
PAGE_MSG ****-->5           CLKN: 23257     clock: 7.768080e+00
PAGE_ACK 5-->0              CLKN: 24859     clock: 7.768710e+00
MASTER FROZEN               CLKE: 23259
FHS_PKT 0-->5               CLKN: 23261     clock: 7.769330e+00
```

```
FHS_ACK 5-->0                CLKN: 24862     clock: 7.769642e+00
POLL_ACK 5-->0               CLK:  24867     clock: 7.771205e+00
LMP_HOST_CONNECT_REQ  AM_ADDR: 5
LMP_ACCEPTED                 AM_ADDR: 5      ACCEPTED PDU: LMP_HOST_CONNECTION_REQ
LMP_QOS_REQ                  AM_ADDR: 5
LMP_ACCEPTED                 AM_ADDR: 5      ACCEPTED PDU: LMP_QOS_REQ
RECV L2CAP_CONNECT_REQ              CH: 4
RECV L2CAP_CONNECT_RSP CH: 4
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 7.776205e+00
RECV L2CAP_CONNECT_REQ              CH: 0
RECV L2CAP_CONNECT_RSP CH: 0
RECV L2CAP_CONNECT_REQ              CH: 1
RECV L2CAP_CONNECT_RSP CH: 1
RECV L2CAP_CONNECT_REQ              CH: 2
RECV L2CAP_CONNECT_RSP CH: 2
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.786205e+00
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 4 DELAY 1.062500e-02 SIZE 1000 clock 7.796830e+00
L2CA_DATA_READ              SIZE: 40        CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.798705e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.798705e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.798705e+00
RECV L2CAP_CONNECT_REQ              CH: 3
RECV L2CAP_CONNECT_RSP CH: 3
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 5 DELAY 3.187500e-02 SIZE 1000 clock 7.808080e+00
L2CA_DATA_READ              SIZE: 40        CID: 6
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.809955e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 7.809955e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 7.809955e+00
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 4 DELAY 2.562500e-02 SIZE 1000 clock 7.824330e+00
L2CA_DATA_READ              SIZE: 40        CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.826205e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.826205e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.826205e+00
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 5 DELAY 2.312500e-02 SIZE 1000 clock 7.833080e+00
L2CA_DATA_READ              SIZE: 40        CID: 6
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.834955e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 7.834955e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 7.834955e+00
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 4 DELAY 5.062500e-02 SIZE 1000 clock 7.849330e+00
L2CA_DATA_READ              SIZE: 40        CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 7.851205e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.851205e+00
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 7.851205e+00
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 5 DELAY 4.812500e-02 SIZE 1000 clock 7.858080e+00
L2CA_DATA_READ              SIZE: 40        CID: 6

.
.****Actual file has 5994 lies  of data*****************8
.
.
.

BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 1.996120e+01
L2CA_DATA_WRITE                     SIZE: 1000      CID: 6          DEST_IP 5 clock 1.996120e+01
L2CA_DATA_READ              SIZE: 1000      CID: 2
BD_ADDR 4 DELAY 6.893750e-01 SIZE 1000 clock 1.997558e+01
L2CA_DATA_READ              SIZE: 40        CID: 4
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 1.997745e+01
L2CA_DATA_WRITE                     SIZE: 1000      CID: 4          DEST_IP 4 clock 1.997745e+01
L2CA_DATA_WRITE                     SIZE: 210       CID: 5          DEST_IP 3 clock 1.997874e+01
L2CA_DATA_READ              SIZE: 210       CID: 2
BD_ADDR 3 DELAY 2.809158e-02 SIZE 210 clock 1.998058e+01
L2CA_DATA_READ              SIZE: 210       CID: 2
```

BD_ADDR 3 DELAY 5.591576e-03 SIZE 210 clock 1.998433e+01
L2CA_DATA_READ SIZE: 1000 CID: 2
BD_ADDR 5 DELAY 6.931250e-01 SIZE 1000 clock 1.999183e+01
L2CA_DATA_READ SIZE: 40 CID: 6
BD_ADDR 0 DELAY 1.875000e-03 SIZE 40 clock 1.999370e+01
L2CA_DATA_WRITE SIZE: 1000 CID: 6 DEST_IP 5 clock 1.999370e+01