A Practical Approach for Providing QoS in Bluetooth Piconet

Tsung-Hsun Wu, Ce-Kuen Shieh, *Wen-Shyang Hwang
Department of Electrical Engineering,
National Cheng Kung University, Taiwan, R.O.C.
**Department of Electrical Engineering,*
National Kaohsiung University of Applied Sciences, Taiwan, R.O.C.
thwu@hpds.ee.ncku.edu.tw, shieh@ee.ncku.edu.tw, wshwang@mail.ee.kuas.edu.tw

Abstract

Bluetooth is a promising wireless technology to form personal area network and is being applied in versatile areas including both IP and non-IP protocol services. Current existing medium access control (MAC) scheduling scheme only provides best-effort service for all master-slave connections. It is very challenging to provide quality of service (QoS) support due to the feature of master driven Time Division Duplex (TDD). The Bluetooth standard doesn't address how to meet QoS requirements. Several works [3-9] have been dedicated to address this issue. But all of these approaches require modification of exiting Bluetooth specification and devices, and address only IP protocol services.

To solve this problem, an ideal mechanism must meet following requirements: (1) practical for existing Bluetooth specification and devices, (2) QoS support for both IP and non-IP protocol service, (3) different QoS support in accordance with protocol service, (4) scalable without any changes to slaves.

In this paper, a traffic shaper is introduced in the Logical Link Control and Adaptation Protocol (L2CAP) on master to provide QoS supports for both IP and non-IP packets on existing devices without modification of Bluetooth specification. The approach regulates traffic of different protocol services to comply with constrained rate in Bluetooth piconet. The approach has been implemented and tested in Linux operating system. Experimental results demonstrate that our scheme provides QoS support and is practicable in Bluetooth piconet.

1 Introduction

Bluetooth is a system for providing short-range, low-power and low-cost connectivity operating in the industrial scientific medicine (ISM) band at 2.4GHz [1]. Bluetooth was originally developed as cable replacement solution for consumer electronic products such as cell-phones, serial port and dial-up network. But it has been adapted for printers, keyboards, audio headset and virtually any other digital consumer devices. To date Bluetooth has been a wireless personal area network (PAN) technology for ad-hoc and infrastructure networking as shown in Figure 1(a).

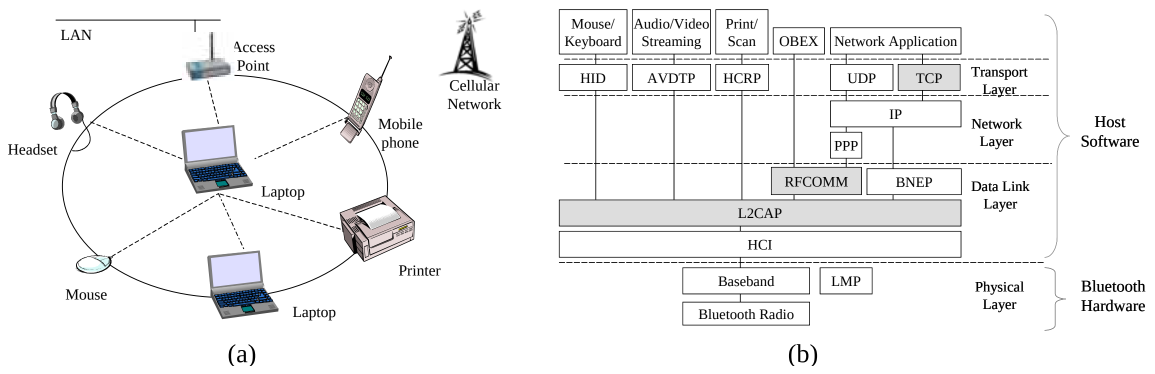


Figure 1. (a) The typical network scenario (b) The Bluetooth protocol stack

Bluetooth supports both voice and data traffic which are treated differently. Voice is circuit-switched connection providing guaranteed service over synchronous connection-oriented (SCO) link on fixed slots. Data are packet-switched connection providing best-effort service over asynchronous connection-less (ACL) link. With ACL link, Bluetooth can support various IP and non-IP protocol services, for example, object exchange, file transfer, synchronization, telnet/ftp, audio/video streaming, print/scan, mouse/keyboard, and audio headset, etc. Because these protocol services have various QoS requirements, it is very important to provide different QoS support for them.

However, current Bluetooth specification doesn't address how to meet these different QoS requirements, and current implementations only provide best-effort service to all applications. The Bluetooth Audio Video Distribution Transport Protocol (AVDTP) specification [2] also state that "When other profiles with stringent requirements are used in conjunction with a profile relying on AVDTP protocol, the performance may be degraded due to insufficient support of QoS in the current Bluetooth specification (v1.1), which all profiles use".

Some literatures of [3-11] propose two kind approaches to alleviate this problem: In [3-9], several polling algorithms are proposed to decrease delay of packets. But, these polling algorithms consider only IP protocol service, and require modification of existing devices and extra interface of Bluetooth baseband specification to interact with different protocol service of application. In [10, 11], several segmentation and reassembly (SAR) policies are proposed to enhance throughput and total bandwidth utilization. Although these polices utilize bandwidth efficiently, they requires ultra information of each queue on baseband, which is not available in current Bluetooth specification. Moreover, neither these policies enhance throughput in accordance with of application nor address the QoS issue. From above related works, we found that their solutions would be impracticable since they require modification of existing Bluetooth specification and devices, and only consider IP protocol service.

To address this problem, in this paper, we introduce a traffic shaper in L2CAP layer on master to provide QoS support without modification of Bluetooth specification, and in accordance with both IP and non-IP protocol services. L2CAP layer is located on host software but not on Bluetooth hardware, therefore it would be practical for existing Bluetooth devices to adapt our approach. One of two major functionality of the L2CAP is protocol multiplexing, therefore L2CAP will be the best place for traffic shaper to regulate both IP and non-IP protocol services. We classify different protocol services into different queues, and regulate traffic of each queue to comply with constrained rate. Compared with existing scheme, our approach is practicable for existing Bluetooth devices without modification of Bluetooth specification, shapes both IP and non-IP packets in Bluetooth piconet, and is scalable without any change to slaves.

The rest of paper is organized as follows. Section 2 gives a background introduction to the Bluetooth technology. In Section 3 we propose our traffic shaper scheme. Section 4 explains the implementation. Section 5 shows experiments and results. Section 6 concludes this paper.

2 Bluetooth technology

As seen in Figure 1(b), the complete Bluetooth protocol stack comprises of both Bluetooth specific protocols like Bluetooth Network Encapsulation Protocol (BNEP), Hardcopy Cable Replacement Protocol (HCRP), Human Interface Device (HDI), AVDTP, L2CAP, Host Controller Interface (HCI), and non-Bluetooth specific existing protocols like RFCOMM, Object Exchange (OBEX), PPP, TCP, and UDP, etc. The re-use of existing protocols helps to adapt existing applications to work with Bluetooth technology. These protocols stacking over L2CAP is referred as upper layer. Here we describe some related background information.

2.1 Upper layer

The BNEP protocol is defined over L2CAP to encapsulate Ethernet protocol. This makes TCP and UDP network applications reside on top of BNEP as network access point and ad-hoc network. TCP is used for best-effort traffic like ftp/telnet, and UDP is mostly used for audio/video real-time applications. The RFCOMM protocol provides emulation of serial ports over L2CAP protocol. The OBEX related applications such as object exchange, object push, synchronization and file transfer are residing on top of RFCOMM protocol, and dial-up network IP applications are residing on top of PPP and RFCOMM protocol. The RFCOMM protocol is a subset of ETSI GSM TS 07.10 standard. The HCRP protocol resides on top of L2CAP for printer and scanner applications. The AVDTP protocol resides on top of L2CAP for audio video streaming. AVDTP defines audio/video stream negotiation, establishment, transmission procedures and message format. The transport mechanism and message formats are based on two major protocols: RTP (Real-time Transport Protocol) and RTCP (Real-time Transport Control Protocol). The HID protocol is defined to use human interface devices, such as mouse, keyboard, joystick, remote sensor, and bar code scanner over Bluetooth protocol stack using the L2CAP layer.

2.2 HCI

Bluetooth devices will have various physical bus interfaces, such as USB, PC card, RS232, and UART, could be used to connect to the Bluetooth hardware. The HCI interface provides a uniform interface method of accessing the Bluetooth baseband capabilities.

HCI data packets are used to exchange data between host software and Bluetooth hardware. The *connection handle* field of HCI packet identifies the ACL connection for the data. The host must use different connection handle for ACL link on different remote device. Since only single ACL link can exist between a master and a slave, only a unique connection handle will be used for each remote device. Therefore connection handle can be used to identify which remote device connected to.

2.3 L2CAP

As shown in Figure 1(b), the Bluetooth protocol stack relies on L2CAP layer to provide services to upper-layer protocols. Two major functionalities of L2CAP are higher layer protocol multiplexing capability and packet SAR operation. The L2CAP protocol multiplexing uses *protocol service multiplexor* (PSM) of L2CAP connection request packet to identify upper layer protocol services (BNEP, RFCOMM, HCRP, AVDTP, etc). The SAR function segments a L2CAP packet into several HCI packets for transmission over baseband/radio air, and reassembles those at the receiver before forwarding them to the upper layer.

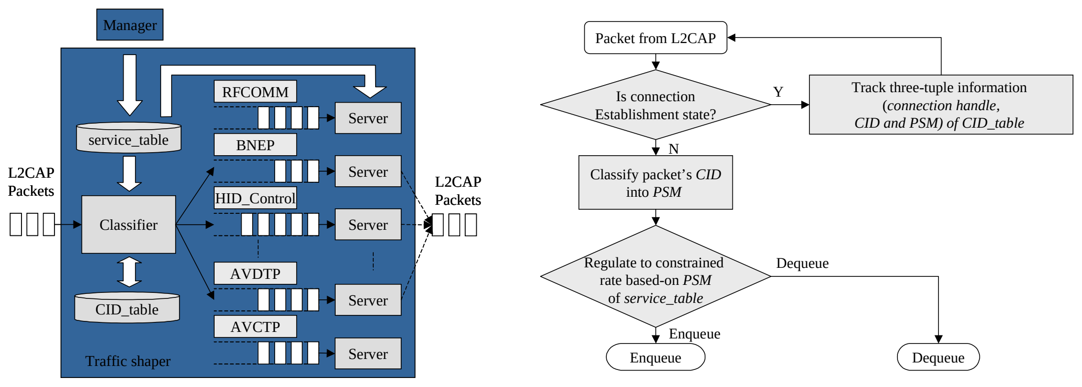
The L2CAP is packet-based and follows a communication model based-on *channel*. A channel represents a data flow between L2CAP entities in remote devices. Each one of the end-points of an L2CAP channel is referred by a *channel identifier* (CID). CID is logical name representing logical channel end-point on device. Same CID is not reused as a local L2CAP channel endpoint for multiple simultaneous L2CAP channels between a local device and same remote device.

CID assignment is relative to a particular device and a device can assign CIDs independently from other devices. Thus, even if the same CID value has been assigned to channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device by different HCI connection handle. Therefore, the *three-tuple* information: connection handle, CID, and PSM can be used to identify L2CAP protocol service in remote devices.

In summary, Bluetooth applications consist of various IP and non-IP protocol services in upper layer using L2CAP protocol. Since the upper layer and L2CAP locate on host software, which would be practical to adapt with existing Bluetooth system. By means of the three-tuple information, it would be ideal to classify and shape protocol services in L2CAP.

3 Traffic Shaper Scheme

In this section, we introduce a traffic shaper scheme in L2CAP layer to regulate traffic over ACL links to comply with constrained rate.



Manager
service_table
Classifier
CID_table
Traffic shaper
L2CAP Packets
RFCOMM
BNEP
HID_Control
AVDTP
AVCTP
Server
L2CAP Packets
Packet from L2CAP
Is connection Establishment state?
Y
N
Track three-tuple information (*connection handle, CID and PSM) of CID_table*
Classify packet's *CID* into *PSM*
Regulate to constrained rate based-on *PSM* of *service_table*
Dequeue
Enqueue
Enqueue
Dequeue

(a) (b)

Figure 2. (a) Traffic shaper scheme (b) Traffic shaper algorithm

Figure 2(a) shows the structure of our scheme in L2CAP layer on a master node of Bluetooth piconet. The traffic shaper of our approach locates on master of Bluetooth piconet as existing IP traffic shaper on network gateway. This achieves practicability and scalability, since master is embedded our scheme in host software and slaves leave no change. The traffic shaper can regulate both incoming and outgoing packet on master. The traffic shaper consists of several major components:

Service_table: The service_table stores settings of protocol service (*PSM*) and corresponding *constrained rate* specified by *Manager* graphic user interface.

CID_table: The CID_table is maintained by classifier to track three-tuple information: connection handle, *CID*, and corresponding protocol service (*PSM*) of L2CAP connection request packet, so as to identify protocol service after channel configured.

Classifier: The classifier allocates single queue and server for each protocol service (*PSM*) of service_table. In connection establishment state, the classifier tracks L2CAP connection request packet to construct CID_Table. After channel configured, the classifier identifies each L2CAP data packet by the tree-tuple information of CID_Table, then dispatches it to corresponding protocol service queue.

Queue: Each protocol service will be allocated a separate buffer for queuing.

Server: Each queue comes with a separate server as leaky bucket traffic shaper to regulate L2CAP data packet to constrained rate defined in service_table. The shaping mechanism is multiple single-server queues.

The algorithm of our scheme is shown in Figure 2(b). During connection establishment state, the connection initiator will send L2CAP connection request packet including *PSM* and source *CID* field to create L2CAP channel (on HCI connection handle) between two devices. The source *CID* represents a channel endpoint on the device sending the request. Once the channel has been configured, L2CAP data packets on transmission must be sent to this *CID*, but without *PSM* field. Therefore, *classifier* constructs *CID_table* composed of three-tuple information of L2CAP connection request packet in connection establishment state, to identify L2CAP data packet’s protocol service after channel configured.

After channel configured, the classifier will examine *CID* field of L2CAP data packet to identify its protocol service *(PSM)*. If *PSM* listed in service_table, the classifier puts packet into a separate queue, otherwise immediately reinjects packet to original path. Once packet has been enqueued, the server shapes traffic to constrained rate of service_table to dequeue packet into original path. If queue is full, L2CAP packet will be dropped.

Consequently, with the help of the CID_table and service_table, the classifier can identify IP and non-IP protocol service, and the server can shapes traffic to constrained rate.

4 Implementation

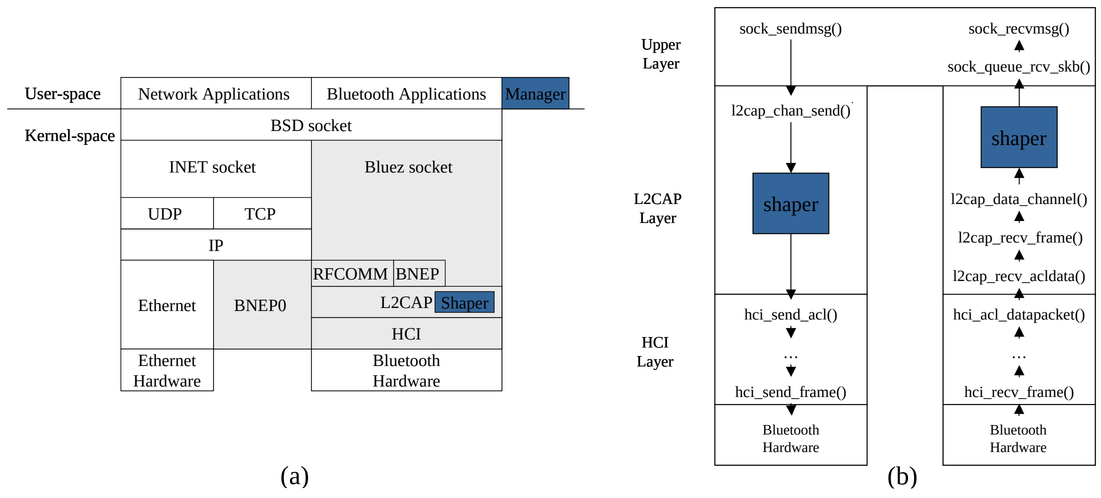


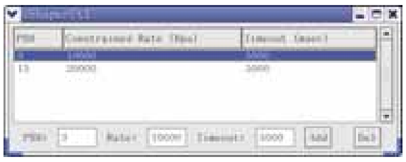
Figure 3. (a) Bluetooth architecture in Linux (b) Embedded Traffic shaper (after channel configured) in BlueZ protocol stack

This section describes the implementation of our scheme in Linux. Linux is an open source operating system with Bluetooth protocol stack. We implement our shaper in L2CAP layer (l2cap.c) of BlueZ protocol stack [12] within Linux kernel 2.4.19. Figure 3(a) illustrates the Bluetooth architecture in Linux.

In connection establishment state, several routines are called to construct three-tuple information of CID_table. The connection initiator use *l2cap_connect* routine to send connection request L2CAP packet, then acceptor is invoked by *l2cap_connect_req* routine with HCI connection handle and L2CAP protocol service *(PSM)* to choose proper source *CID.* If acceptor listens on same protocol service, the acceptor will invoke *l2cap_send_rsp* routine from same *l2cap_connect_req* routine to response initiator. The initiator will be invoked by *l2cap_connect_rsp*, *l2cap_connect_cfm* and *l2cap_conn_ready* routines as connection confirmed. Therefore, we implement classifier of shaper in *l2cap_connect_req*, *l2cap_conn_ready* and *l2cap_connect_rsp* routines to track three-tuple information of CID_table.

After channel configured, when the HCI layer of receiver accepts incoming HCI packet from Bluetooth hardware, it forwards it to *l2cap_recv_acldata* routine in Figure 3(b) to assemble segmented HCI packets to form completed L2CAP packet. Then L2CAP layer passes L2CAP data packet to *l2cap_recv_frame* and *l2cap_data_channel* routine, then invoke *sock_queue_rcv_skb* routine based-on *PSM* to transfer L2CAP packet to *sock_recvmsg* routine of upper layer like RFCOMM and BNEP. If sender's upper layer protocols call *sock_sendmsg* routine to transmit outgoing packets, the kernel will invoke *l2cap_chan_send* to segment L2CAP packet into several HCI packets in HCI layer, then HCI packet will be sent into Bluetooth hardware. Therefore, we implement classifier and server in *l2cap_data_channel* and *l2cap_chan_send* routines to classify, enqueue and dequeue protocol service's queue for incoming and outgoing traffic. The highest rate at the server can shape, is limited by the system timer (trigged by hardware and counted by the timer interrupt in Linux kernel), which normally ticks at 100Hz.

To provide visual friendliness for users, the manager is implemented with GTK+ shown in Figure 4 to interact with shaper to manipulate (add, del, or change, etc) constrained rate for specific protocol service (PSM).



PSM	Constrained Rate (Bps)	Timeout (msec)
3	10000	5000
13	20000	3000

PSM: 3 Rate: 10000 Timeout: 1000 Add Del

Figure 4. Manager

5 Experiments

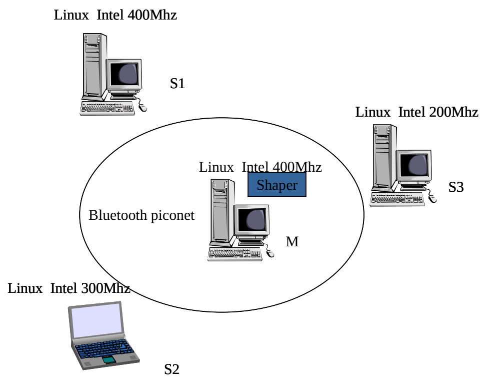


Figure 5. Experimental platform

The experimental platform contains four Bluetooth version 1.1b devices, using one master *M* and three slaves *S1*, *S2* and *S3*. The network is shown in Figure 5. Our traffic shaper scheme is running on host software of master *M.*

In order to verify the effectiveness of our scheme, we measured the throughput and delay of piconet composed of slaves running different protocol services. We use *hcidump* [12], *MGEN* [13], and *iperf* [14] to measure the throughput and delay of OBEX (RFCOMM), UDP and TCP (BNEP), respectively. We experiment different protocol services with three scenarios in Table 1(a), (b) and (c).

Table 1. Experiment parameters of (a) OBEX and TCP, (b) OBEX and UDP, (c) OBEX, TCP and UDP

Slave	**Application Type**	**PSM**	**Constrained Rate (Kbps)**
S1	TCP	BNEP	N/A
S2	OBEX	RFCOMM	70

(a)

Slave	**Application Type**	**PSM**	**Constrained Rate (Kbps)**
S1	UDP	BNEP	N/A
S2	OBEX	RFCOMM	70

(b)

Slave	**Application Type**	**PSM**	**Constrained Rate (Kbps)**
S1	UDP	BNEP	N/A
S2	TCP		
S3	OBEX	RFCOMM	70

(c)

Table 2. (a) Goodput of OBEX and TCP, (b) Goodput of OBEX and UDP, (c) Delay of UDP when OBEX is present (d) Goodput of OBEX, TCP and UDP (e) Delay of UDP when OBEX and TCP present. "Without" means *without* our scheme and "With" means *with* our scheme

Goodput (kbps)

Slave	S1	S2	Total
Without	378.3	164.9	543.2
With	485.4	69.0	554.4

(a)

Goodput (kbps)

Slave	S1	S2	Total
Without	335.7	176.2	511.9
With	453.5	69.0	522.5

(b)

S1	Average delay (s)	Max Delay (s)	Delay Variance (s^2)
Without	1.53	1.64	0.18
With	1.10	1.20	0.17

(c)

Goodput (kbps)

Slave	S1	S2	S3	Total
Without	207.2	219.3	130.3	556.8
With	217.3	243.0	69.0	529.3

(d)

S1	Average delays (s)	Max Delays (s)	Delay Variances (s^2)
Without	0.94	1.73	0.53
With	0.02	0.02	0.002

(e)

The experimental results of first scenario are shown in Table 2(a). With our scheme, the goodput of OBEX is 69.0 kbps, which is approximately close to the pre-defined constrained rate, i.e. 70 kbps. The goodput of TCP protocol is 485.4 kbps, better than 378.3kbps, i.e. without our scheme. Meanwhile, total bandwidth utilization is 554.4 kbps, close to 543.2 kbps.

The results of second scenario are shown in Table 2(b) and (c). With our scheme, the goodput of UDP protocol is 453.5kbps, better than 335.7kpbs. Total bandwidth utilization is 522.5 kbps, close to 511.9 kbps. The average/max/variances of delay are 1.10/1.20/0.17, better than 1.53/1.64/0.18.

The results of third scenario are shown in Table 2(d) and (e). With our scheme, the goodput of UDP protocol is 217.3 kbps, better than 207.2 kbps. The goodput of TCP protocol is 243.0 kbps, better than 219.3 kbps. Total bandwidth utilization is 529.3 kbps, which is 5% less than 556.3 kbps. The average/max/variances of delay are 0.02/0.02/0.002, better than 0.94/1.73/0.53.

6 Conclusions and future work

This paper proposed a traffic shaper scheme in L2CAP layer on master to provide QoS support in Bluetooth piconet. It is implemented in Linux to regulate predefined protocol service to constrained rate.

The experimental results have demonstrated that our approach (1) is practical for existing Bluetooth specification and devices, (2) regulates both IP and non-IP protocol services, (3) provides different QoS support in accordance with protocol service, (4) is scalable without any changes to slaves.

We plan to investigate the performance in scatternet, and present of AVDTP in the near future. We also intend to provide QoS support directly based-on TOS (Type of Service) field in the IP header for differential service.

7 References

[1] Bluetooth SIG, “Specification of the Bluetooth System”, Version 1.1, Volume 1, http://www.bluetooth.org, Feb 2001.

[2] Bluetooth Audio Video Working Group, “Audio/Video Distribution Transport Protocol Specification”, Version 1.0, http://www.bluetooth.org, May 2002.

[3] Lapeyrie, J.-B.; Turletti, T.; “FPQ: a fair and efficient polling algorithm with QoS support for bluetooth piconet”, INFOCOM 2003

[4] Yaiz, R.A.; Heijenk, G.; “Providing delay guarantees in bluetooth”, ICDCS 2003

[5] Yunxin Liu; Qian Zhang; Wenwu Zhu; “A Priority-Based MAC Scheduling Algorithm for Enhancing QoS Support in Bluetooth Piconet”, ICCCAS 2002

[6] Wei-Peng Chen; Hou, J.C, “Provisioning of temporal QoS in Bluetooth networks”, WMWC2002

[7] Yang-Ick Joo; Jong-Soo Oh; Oh-Seok Kwon; Yongsuk Kim; Tae-Jin Lee; Kyun Hyon Tchah; “An efficient and QoS-aware scheduling policy for Bluetooth”, IEEE VTC 2002

[8] Jong Soo Oh; Yang-Ick Joo; Oh-Seok Kwon; Yongsuk Kim; Tae-Jin Lee; Kyun Hyon Tchah; “Differentiated fairness guaranteeing scheduling policies for Bluetooth”, IEEE VTC 2002

[9] Martin van der Zee, Geert Heijenk, “Quality of Service in Bluetooth Networking Part I”, 2001

[10] Kalia, M.; Bansal, D.; Shorey, R.; “Data scheduling and SAR for Bluetooth MAC”, IEEE VTC 2000

[11] Das, A.; Ghose, A.; Razdan, A.; Saran, H.; Shorey, R.; “Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network”, IEEE INFOCOM 2001

[12] BlueZ Linux Bluetooth protocol stack. http://www.bluez.org

[13] MGEN- The Multi-Generator Toolset, http://manimac.itd.nrl.navy.mil/MGEN/

[14] Iperf - The TCP/UDP Bandwidth Measurement Tool, http://dast.nlanr.net/Projects/Iperf