This paper was published in the proceedings of **IEEE sponsored** ***Conference on Software, Telecommunications and Computer Networks(*** **SoftCOM) 2004**
held during October 10-13, 2004 at Split, Dubrovnik (Croatia) and Venice (Italy)
www.fesb.hr/SoftCOM/2004/final_program.pdf

Convergence in Bluetooth and 802.11 networks

Satyajit Chakrabarti*, Son T. Vuong*, Anirban Sinha**, Rajashree Paul***
*Department of Computer Science, University of British Columbia, Canada
**Department of Computer Science, Institute of Engineering and Management, India
***School of Computing Science, Simon Fraser University, Canada
satyajit@cs.ubc.ca, vuong@cs.ubc.ca, anirban@ieee.org, rpaul2@cs.sfu.ca

Abstract: **Ad Hoc Networks using Bluetooth Technology and 802.11 technology have gained widespread popularity in the networking community. Whereas Bluetooth technology has certain obvious advantages like low power consumption and reliable connection, it has a low area of operation and lower bandwidth in the order of 721 kpbs. 802.11 technology on the other hand, has a wider area of operation and therefore very useful as Access points and higher bandwidth to the order of 11 Mbps in 802.11b. But 802.11 has a higher power consumption than Bluetooth. Due to popularity of devices fitted with both Bluetooth and 802.11 chips, we look into ways of collaboration and convergence between the two technologies. We propose an architecture for convergence of Bluetooth and 802.11 technologies and propose software switching protocols to facilitate the smooth handover of connection with minimal loss of data and without disconnection of service.**

1. INTRODUCTION

Bluetooth [1],[2] in the 2.4 GHz ISM band has emerged the market leader for short range wireless technology. Bluetooth 1.1 specification grew to include the formation of Personal Area Networks (PANs) which is narrower in scope of operation than WLAN. The standardization of PANs is carried is being carried out by the 802.15 working group [3]. The IEEE 802.11 standard [5], [4] for WLANs is the most widely used WLAN standard today. The standard uses the carrier sense multiple access (CSMA), medium access control (MAC) protocol with collision avoidance (CA).

In this paper we present a novel architecture for a hybrid Bluetooth- 802.11 access point and algorithms for interoperability of Bluetooth and 802.11 on a software switching level. In section 2 we discuss some relevant research on interoperability in hybrid networks. In section 3 we propose algorithms for a hybrid network consisting of Bluetooth and 802.11 devices. We analyse the handoff latency in our proposed protocol in section 4. In section 5 we conclude with some directions for future work on this subject.

2. RELATED WORK

Kansal et al. [11] introduce a Handoff scheme for Bluetooth devices to allow mobility of devices in Bluetooth public access (BluePAC) environments. Baatz et al. [8] concentrates on handoff support for mobility with IP over Bluetooth. Perkins et al. [6] present a handoff scheme for mobile IP's.

The paper by Mishra et al. [9] does a thorough analysis of the handoff procedure in the 802.11 MAC layer. Kastell et al. [10] presents security issues involved in hybrid handover procedures. Pack et al. [12] uses a predictive authentication for fast handoff's in 802.1x mode.

3. HANDOFF ALGORITHM

In our hybrid Bluetooth/802.11 network we have access points that have both the Bluetooth and 802.11 antennae and physical interfacing devices/network cards. The devices can access the backbone 802.11 network resources either through the "802.11 AP- 802.11 Client" interface or through "Bluetooth(BT) AP- Bluetooth Client" interface. When the client uses its Bluetooth interface, the Access Point (AP) should be able to forward the Bluetooth packets to the backbone 802.11 network and the incoming packets from the 802.11 to the Bluetooth interface so that they can be passed on to the client. However, when the client uses the 802.11 interface, the packets are directly forwarded to the corresponding 802.11 interface in the AP and from there to the backbone network.

We propose to introduce a handoff algorithm through which the client can switch from one network to the other, either voluntarily (due to his power or bandwidth requirements) or because his mobility takes him out of the Bluetooth radio range. When this switching of interfaces occurs, the application layer must remain oblivious to this change and a lower software layer must be able to activate the appropriate interface that the client will use. For this purpose, we propose to introduce a software engine or daemon which we call *layer of software control* (LSC) in between the TCP/IP and the next lower layer in the hierarchy (LLC for 802.11 and PPP or L2CAP layer for Bluetooth) that would take care of the switching between the Bluetooth device and the 802.11 device both in the client as well as at the hybrid Bluetooth/802.11 AP. Conceptually this software engine is common to both the Bluetooth and 802.11 physical interfaces, as shown in Fig 1. This layer traps all the outgoing & incoming TCP packets, buffers them appropriately and then releases them to the lower layers of the stack (after deciding which interface to activate) so as to facilitate smooth handoff.

The application layer can directly send commands to LSC daemon. For example, when the user wants to voluntarily switch operation from one network to the other, he will simply send a specific command to LSC, which is implemented as API's and this will, in turn, then activate that physical interface and divert all outgoing TCP/IP packets to it & listen for incoming packets from that interface. In other words, after receiving the commands from the application, it becomes the responsibility of this layer to initiate a manual handoff. In order to facilitate this, LSC must send some direct HCI commands to the Bluetooth physical chip. These direct commands are *Get_Address* (for getting PHY address and CLK information), *Create_Connection* (connect devive and set scan mode), *Write_Page_Timeout* (set time spent on paging), *Read_Scan_Enable* (read device configuration regarding page scan), *Write_Scan_Enable* (enable device to enter periodic page scan), *Read_Page_Scan_Activity* (check page scan parameters), *Write_Page_Scan_Activity* (set page scan parameters).

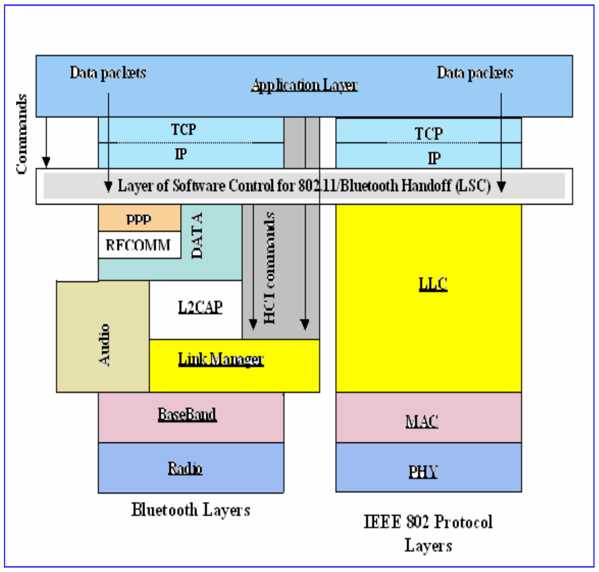

Figure 1. Conceptual representation of LSC daemon with respect to 802.11 & Bluetooth protocol stack.

Let us take two specific cases of voluntary handoff and a single case of natural forced handoff.

A. User is in the radio range of both 802.11 & BT and wants to voluntarily switch from his currently running BT-BT communication to 802.11-802.11, possibly because he needs a higher bandwidth:

1. The LSC layer in client receives request command from the application layer to switch to 802.11.
2. LSC then sends a control packet destined for the LSC layer in the Bluetooth AP that this current client is initiating a manual handoff from BT to 802.11. The control packet also contains the physical address of the client machine.
3. Client LSC creates a buffer for the unsent TCP/IP packets during handoff.
4. The LSC layer at Bluetooth AP, after receiving the control packet creates a buffer for all unsent packets destined for the client address specified in the control packet and acknowledges the handoff request. The acknowledgement packet contains the channel information about the 802.11 AP where the client will listen for beacon frames (passive scanning) or send its probe request (active scanning), once it switches its interface to 802.11. Further, at AP, the address information for all such clients who wishes to switch to 802.11 network is kept in an address queue. For each item in the address queue, there exists a buffer of all unsent data packets from AP for that client.
5. When the acknowledgement packet reaches the LSC layer of the client, the client LSC deactivates the Bluetooth interface and activates the 802.11 interface. The 802.11 interface searches for available 802.11 AP by the scan method. Once the connection is established, all the unsent packets destined for the client physical address is sent by the AP. The unsent packets at the client are also sent. Reauthentication takes less time as the address of the client seeking handoff had already been sent to the AP previously by the LSC control packet.

B. User is in the radio range of both 802.11 and BT and he wants to voluntarily switch from his currently running 802.11-802.11 communication to Bluetooth-Bluetooth communication, possibly because he needs to save power:

Similar steps are followed as in Case A in steps 1 - 4,as during switchover from BT-BT to 802.11-802.11. In step 5, BT interface directly goes to the R0 page scan mode (skips the time-expensive enquiry mode).Since the BT AP knows about the client address and clock information, it pages the client with these information until the client responds. If the client is not in radio range of BT, the BT AP will make four page attempts to connect to the BT interface of the client before giving up. This is explained below in the case when there is a natural handoff from BT to 802.11. After connection is established, all the unsent packets destined for the client physical address is sent by the AP, The unsent packets by the client are also sent.

C. A client running BT connection, moving away from the BT AP so that it no longer lies in its radio range. Our design allows to seamlessly switch to the 802.11 network so the client does not feel any break in connection.

1. BT AP constantly polls the client when the client is accessing the backbone 802.11 network through BT interface. The clients respond to these poll packets.
2. The poll packets reach the LSC layer in both the AP and the client. Polling scheme is round robin, in which the AP (master) polls each connected BT (slave) device. Data requirements may be different for different clients, and the packet duration may be adjusted for this, using single slot packets for slaves with low data rate requirements and multi slot packets of length 3 or 5 for higher data rates.
3. To detect connection loss, the AP keeps a timer and if no reply occurs for the timeout period, $T_{polltimeout}$, connection is assumed to be broken. At the mobile too, a similar procedure is followed, to detect loss of connection. At both the access point and the mobile clients, the timeout value, $T_{polltimeout}$, is specified to be equal to the maximum number of slots that may pass between two successive turns. One poll round time takes s x 2 x l where s is the number of clients (slaves) attached to the AP (master). Assuming s = 7 (can vary from 1 to 7) and l=5 (maximum), $T_{polltimeout}$ = 70 slots which is the worst case value. If the AP needs to communicate with more than 7 slaves, it can do so by instructing active slave devices to switch to low power park mode and inviting other parked slaves to become active in the piconet. This can be repeated to allow a master to serve a large number of slaves.
4. Each Bluetooth AP finishes its poll round and checks if the address queue has any pending addresses of clients trying to connect to BT from 802.11. If there is such an element in the queue, the AP sends a HOLD message to all its connected clients to suspend their connection loss detection timers for a period of T_{AP_Page}.
5. BT AP then pages the mobile client using the address and clock information received. The clients resume their connection loss detection poll timers either on the expiry of the T_{AP_Page}, or if the AP sends a regular poll packet before that. T_{AP_Page} is the maximum time an access point spends on paging for a slave who switches over from 802.11 to BT.

All incoming data packets for all the BT clients must also be buffered so as to be delivered once the HOLD stage is over which is equal to T_{AP_Page}. Since we configure the clients to use the R0 page scan mode, each page train needs to be attempted only once by the AP, which means that both the trains can be tried out in 32 slots. Four page attempts are made for robustness, leading to T_{AP_Page} equal to 128 slots, which is 80 milliseconds. Thus, the worst case time required to discover a break in BT-BT connectivity is $T_{polltimeout}$ + T_{AP_Page} = 70 + 128 slots = 198 slot time (about 124 milliseconds). Since it is possible that the mobile clients move out of the BT range during the HOLD period, we do the following: as the clients will discover that they are out of range only when the HOLD period is over, they will start scanning for 802.11 AP's beacon after the HOLD period. At the same time, the Bluetooth AP will also discover that the particular client is out of the BT range only after the HOLD stage, i.e. both client and AP discover the switchover at the same time. The AP keeps track of the addresses of those Bluetooth clients which are detected as lost & then looks forward for a connection request from the client.

When a client moves away from the BT radio range, the loss of connection is detected at the first unsuccessful poll attempt because only one poll attempt can occur within $T_{polltimeout}$ period after a successful poll. Since the round robin poll always occurs within this time, live BT-BT connections will be able to refresh the timeout counter before timeout.

Once a poll timeout occurs, the LSC at AP immediately forwards the address of this client to the 802.11 interface AP so that when this client connects to 802.11 network, the authentication will take less time. Meanwhile, loss of connection will also be detected by the client due to lack of arrival of the poll packets from the Bluetooth AP after an interval of $T_{polltimeout}$. The client will then activate the 802.11 interface and go to the scan mode to discover the closest available 802.11 AP. When the 802.11-802.11 connection gets established, the unsent data packets from either end are sent.

One important consideration is the length of the address queue that we need to keep at both the 802.11 and Bluetooth side. The number of elements in this queue depends on the number of clients that switch between the network in a specified time interval. In order to accommodate the fact that the queue can be very large in a region where the clients move very fast or they voluntarily switch network frequently, we follow the following procedure. The AP completes one round of polling all the attached clients and then checks to see if the address queue has any elements. If an element exists, it pages that device. The AP processes a single element of the queue at a time between each round of polling. This ensures that the existing clients are not kept waiting for indefinite time while the AP pages for new devices. On the 802.11 side, the AP's are discovered by the mobile clients themselves from the beacons and so there does not arise any problem if the address queue is large.

Since we cannot modify the existing accepted TCP/IP headers or introduce new header for LSC because that would lead to serious backward incompatibility, we need to device some technique so as to distinguish LSC control packet from ordinary data packets. For this reason, we use the 6 unused bits in the TCP header between "TCP header length" field and "URG" flag. When all six of these bits are set to 1 (i.e., 111111), this signifies to te remote LSC engine that the packet is a dummy TCP packet sent by the transmitter and contains control information in the data area of the packet for handoff. A dummy TCP packet looks like an ordinary TCP/IP packet, but it is not sent by any client for data transfer. It is generated by the LSC alone & actually contains the following information bits in the data area that facilitates handoff :-

1. A single bit **RQ/RES** denoting whether it's a client request packet or server response packet. When set, this signifies client request. Otherwise its server response.
2. Two bits **HREQ** signify whether a switchover from 802.11 to Bluetooth or vice versa is requested.
   00 => Client switching off both its interfaces
   01 => 802.11 to Bluetooth switchover is requested.
   10 => Bluetooth to 802.11 switchover is requested.
   11 => Not used.
3. Bits (variable) denoting address and clock information of the client. Relevant only when RQ/RES bit is set. It is used when the client switches over from 802.11 to Bluetooth, i.e., HREQ bit is 01.
4. Bits (variable length, depending on the physical medium used) **CH** denoting the channel information of the 802.11 AP that will be used by the clients when switching over from Bluetooth to 802.11. It is relevant only when RQ/RES bit is reset.

When the six unused bits of the TCP header is not all set, it indicates that the packet is a normal TCP datagram sent by any mobile host for data communication. Table 1 shows the API's that are available with LSC which any application programmer may use to interface his program with LSC for seamless handoff.

Table 1: API's available with the LSC for use.

API	Description
Activate_Bluetooth()	Activates Bluetooth interface
Activate_Wlan()	Activates 802 interface
Read_current_interface()	Returns the current interface
Send_control()	Sends control signal to interface
Read_control()	Reads control signal to interface
Send_control()	Sends control signal to interface
Send_Hold()	Sends Hold signal to interface
Create_buffer()	Creates local buffer
Acknowledge_request()	Acks request to interface

4. ANALYSIS OF THE PROTOCOL

We take up all the three cases discussed above one after the other. In each of these cases, we split the total handoff process into three phases, **detection, search and execution.**

Case 1: ***When the client is operating using the Bluetooth-Bluetooth interface and voluntarily switches to 802.11.***

a. *Detection of Handoff:*

Here, the AP detects that a handoff is requested by using the LSC control frame. Once it receives the LSC control frame, it sends its response in the next slot. Assuming the length of the packet to be of 5 slots, the total delay is 10-slot time, which is about 6 milliseconds. Once the Bluetooth AP receives the response, it switches its interface from Bluetooth to 802.11 and goes to the scanning mode.

b. *Search:*

Searching by 802.11 clients for possible nearest AP takes place ONLY in a single channel as was sent by the acknowledgement packet when the client requested a handoff. Thus, even in the active scanning mode, the scanning time is reduced.

According to the analysis done in *[7],* we note that
MinChannelTime= DIFS + (aCWmin + ~~s~~aSlotTime).
According to 802.11b standard, aCWmin= 31 slots, aSlotTime= 20μsec and DIFS = 50μsec.
Thus MinChannelTime=670 μsec.
Using the analysis in the paper MaxChannelTime=10.24 ms.
Now, in our case, the client scans only one single channel. Assuming that there is equal probability for this channel to be unused as well as to be free,
Total Search Time, s = (T_u + T_e) / 2 where T_u= Time needed to scan a used channel and T_e = Time needed to scan an empty channel.
Now, T_u= $2T_d$ + MaxChannelTime & T_e = $2T_d$ + MinChannelTime
Using Td = 65 ms (for 20 stations), Tu = 140.24 ms & Te = 130.67 ms; So, s = 135.5 ms

c. *Execution of Handoff:*

Now worst-case handoff execution time is 3 ms using a Spectrum24 card.

Thus, total handoff latency : 6 + 135.5 + 3 ms = 144.5 ms

Case 2: ***When the client is operating using the 802 interface and voluntarily switches to Bluetooth.***

a. *Detection of Handoff:*

Here, a control packet is sent by the LSC layer of the client to inform the 802.11 AP that it needs to initiate a handoff to Bluetooth. When the 802.11 AP acknowledges, the client switches over. If we ignore the time taken for the packets to travel, the overall delay in this case would be very

low and hence we neglect the delay in this packet transmission.

b. Time taken to resume Bluetooth-Bluetooth connection:

The time taken to resume connection depends on the number of elements in the address queue (i.e., clients who are willing to switchover from 802.11 to Bluetooth and have sent their requests). Since maximum number of active slaves in a piconet is 7, we assume that already 6 addresses are present in the queue when the 7th one arrives. Time taken to process each of the preceding 6 addresses as well one round polling through all the attached slaves is:

$T_{AP\_Page} + (1\times2\times5) + T_{AP\_Page} + (2\times2\times5) + T_{AP\_Page} + (3\times2\times5) + \ldots + T_{AP\_Page} + (6\times2\times5)$

since one poll round = s × 2 × l slots where s = number of slaves, l = slot length of the packet and s increases as each slaves get attached to the AP (master).

The above expression equals 6 * T_{AP_Page} +210 slots = 128+210 = 338 slot-time = 212 milliseconds.

But, for all practical purposes, this value would be much less as probability for simultaneous 7 handoff requests coming from the clients for 802.11 to Bluetooth switchover is very very less.

Now, for the current address (7th), paging takes approximately 16 slots in R0 scan mode. This is approximately 10 milliseconds.

Thus, total delay in the worst case is 222 milliseconds.

Case 3: When the client is operating using the Bluetooth-Bluetooth interface and moving away from the Bluetooth radio range.

a. Detection of Handoff:

The time taken to detect loss of connection is $T_{polltimeout}$ when no paging attempt takes place. If however, paging attempt is taking place then worst case duration within which a break in connection will be detected is $T_{polltimeout} + T_{AP\_Page}$ = 70 + 128 slots = 198 slot time (about 124 milliseconds) as was discussed before.

Thus, the time to detect break in connection is much less than that when a normal handoff between two 802.11 AP takes place using any 802.11b physical cards [7].

b. Search & Execution of Handoff:

Search for possible nearest 802.11 AP and then finally execution of the handoff procedure takes place by normal 802.11-802.11 handoff mechanism (active scanning by 802.11 client). Worst case time required for search and execution using D-Link 520 802.11 interface is 290 millisecond as is analyzed in [7].

So total handoff latency is 124+290 ms = 414 ms

5. CONCLUSIONS AND FUTURE WORK

In this paper we introduced Handoff algorithms' for different scenario's of switching from a Bluetooth AP-Bluetooth Client connection to 802.11 AP – 802.11 Client and vice versa. We have modified the current protocol stack of Bluetooth and 802.11 to introduce a new layer LSC for interoperability. We have designed the header for messages in the LSC and shown the delay analysis on various handoff's.

We propose to implement the LSC daemon as an external kernel module that can be attached using 'insmod' command in unix. The exact implementation and evaluation of this daemon as kernel module is left as a future extention of this work in another paper.

REFERENCES

[1] Jennifer Bray and Charles F Sturman, Bluetooth : Connect Without Cables, Prentice Hall, 2001.

[2] "The Bluetooth SIG," http://www.bluetooth.com.

[3] IEEE 802.15 Working Group for WPAN standards. http://grouper.ieee.org/groups/802/15.

[4] IEEE Std 802.11 - Wireless LAN MAC and Physical Layer (PHY) specifications. The IEEE, Inc., 1999.

[5] IEEE 802.11 Working Group Task Group. http://grouper.ieee.org/groups/802/11/main.html

[6] C. E. Perkins, K.-Y. Wang, "Optimized Smooth Handoffs in Mobile IP", Proceedings of the Fourth IEEE Symposium on Computers and Communications, July, 1999

[7] H. Velayos, G. Karlsson, "Techniques to Reduce IEEE 802.11b Handoff Time", in the proceeding of IEEE ICC 2004, Paris, France, June 2004.

[8] S. Baatz, "Handoff Support for mobility with IP over Bluetooth", at the 25th Annual Conference on Local Computer Networks (LCN '00), Tampa, November 2000.

[9] A. Mishra, M. Shin, W. Arbaugh, "An Empirical Analysis of IEEE 802.11 MAC layer Handoff process", ACM SIGCOMM Computer Communication Review, Volume 33 Issue 2, April 2003

[10] K. Kastell, U. Meyer, R. Jakoby, "Secure Handover Procedures", in the proceedings of CIC 2003.

[11] Aman Kansal, "A Handoff Protocol for Mobility in Bluetooth Public Access",Proceedings of the 15th ICC, 2002.

[12] S. Pack, Y. Choi, "Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Model," IFIP TC6 Personal Wireless Communications 2002, Singapore, pp.175-182, October 2002.

[13] Pravin Bhagwat, "Bluetooth: Technology for Short-Range Wireless Apps" , IEEE Internet Computing, June 2001

[14] Mustafa Ergen, IEEE 802.11 Tutorial, UC Berkeley, June 2002.