BOWL: Design and Implementation of a (Connectionless) Broadcasting system Over Wireless LAN

Kartik Muralidharan, Karthikeyan Balaji Dhanapal, Atanu Roy Chowdhury
Software Engineering and Technology Labs
Infosys Technologies Ltd
Bangalore- 560100, India
{kartik_muralidharan, karthikeyan_dhanapal, atanu_chowdhury}@infosys.com

Abstract

A truly pervasive computing environment can only be created if ubiquitous computation power can collaborate seamlessly to create superior user experience. With the adoption of mobile computing platforms on the rise, we have explored how the IEEE 802.11 wireless access standards can be used to enable a device to be more aware of the capabilities of the infrastructure in its surroundings. Specifically, we have been able to effectively transfer information to these WiFi enabled devices without explicit connections being established and therefore our solution is proven to be much more scalable. We have also showcased a pilot application using the Broadcast over Wireless LAN (BOWL) technology.

1 Introduction

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. So said Mark Weiser, in his vision of the 21st century computer [1]. Over the past five years or so, pervasive computing has emerged as a new computing paradigm with much appeal. Pervasive computing is all about putting the user, rather than a particular computing device, in the centre of all computing activities. It is about the concept of *any-time, any-where computing* optimized by means of intelligent environments and context-sensitivity. Essentially, all this boils down to a need for an infrastructure enriched with computing, sensing and communication capabilities and with a very high degree of collaboration between all the infrastructural components, all aimed at providing as optimal a user experience as possible.

Today, computation power is no longer limited to computer terminals. There are a plethora of devices like PDAs, smartphones, music players, media centers etc that have huge impact on our lifestyles. The evolution of such computing elements has ushered in a *multi-access-multi-service* environment [4], which is very different from the traditional wired LANs. In this multi-access-multi-service environment, each technology has its own set of limitations. For example, the user needs to be *infrastructure aware* to be able to use the available access channels. Further, each technology has its own association techniques and unique service discovery challenges [5]. All this leads to scalability bottlenecks for large connection oriented transactions. The dynamic environment and security concerns contributes to the lack of seamlessness in the user experience. Hence to realize true pervasive experiences the access technologies must evolve in line with the evolution of devices being used and must also recognize this paradigm of user centric computing.

The WiFi [6] alliance endorses the use of IEEE 802.11 based wireless access to provide users with networked services and this standard enjoys huge proliferation. Thus in this work we have tried to use the existing standard to provide an enriched access mechanism. We have demonstrated how the standard can be used to advertise the pervasive infrastructure without the scalability bottleneck of connection oriented service discovery.

In the rest of this paper, we first describe the association technique in 802.11. The internals of the broadcast protocol is described in Section 3 followed by the performance enhancement techniques. Section 4 presents the performance measures and their interpretations. Section 5 describes a pilot application that uses this technology.

2 Association in IEEE 802.11

Connection establishment in 802.11 networks is essentially a three stage process:

978-1-4244-2100-8/08/$25.00 ©2008 IEEE

1. *Discovery process*: Each station has to discover existing WLAN networks present in the environment either by scanning beacon frames or by sending probing requests.
2. *Authentication process*: The authentication mechanism determines whether a device is allowed on the network or otherwise.
3. *Association process*: Association process allows an access point to map a logical port or association identifier to the wireless station.

Beacon frames are fundamental to the discovery of WiFi networks. Access points or an ad-hoc node broadcast beacon frames periodically to announce their presence. Moreover, these beacons can be overheard by other stations irrespective of them being in a connected or a disconnected state. The beacon frame contains information about transmitter's capability, network id, network name etc which the receiver can use for connection establishment.

For the mobile devices (clients) the standard allows only a single association at any given time, whereas the access point can have multiple client associations . Therefore, a client device would need to disconnect its present association, if it wants to receive data packets from another collocated WLAN. The fresh association would explicitly terminate open connections from the earlier partnership. This is true for both the ad-hoc and infrastructure mode of communication [2]. The fact that 802.11 does not allow multiple simultaneous associations at the client end turns out to be a serious limitation of the standard.

This connection oriented approach under-utilizes the inherent broadcast nature of the media. For a wireless access protocol a connectionless approach is sometimes more beneficial, specially when transmitting small packets or broadcasting. For small packets, a connectionless approach saves on the time required to establish the connection. Similarly for a broadcast packet the connectionless approach does not require all clients to be associated with the same network. Moreover, the connectionless broadcast is not a broadcast by unicast, and hence reduces the number of packets in the air.

3 Broadcast Over Wireless LAN (BOWL)

In this work we introduce a mechanism to transfer data via the beacon frames, which was described earlier. Thus we are able to set up a quick parallel short data transfer stream that can function without explicit point to point connections. This low bandwidth stream can be further multiplexed to carry channel [1] specific information. In our implementation we show how the standard service set identifier (SSID) which is part of the beacon frame can be used to provide this data broadcast. Our protocol does not alter the basic working of the standard and can therefore be implemented on existing WiFi compatible systems by using a software patch. In essence, this protocol can be used for a number of activities like better network management as well as general messaging and alerts within the wireless network.

When a beacon frame is received by a Network Interface Card(NIC), the SSID field is always processed and stored for future reference. We have used the SSID field because it is a user defined namespace and is easily programmable. By embedding information in this field, we are able to create a seamless and connectionless short data stream between the Broadcast System (an adhoc node or an access point) and the client devices. As shown in Fig 1, upto 32 bytes of data can be transferred in a single beacon frame by using the SSID field. However the data to be transferred would in all probability be greater than 32 bytes. Therefore we need to build a fragmentation and reassembling mechanism whereby we can spread the data over multiple beacon frames.

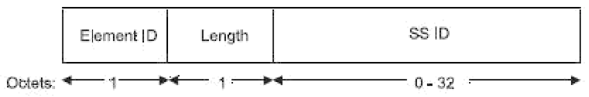


Figure 1. Original SSID field structure

3.1 Fragmentation Scheme

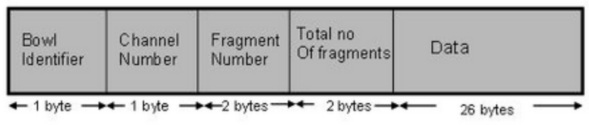


Figure 2. Proposed SSID field structure

The fragmentation scheme for BOWL is shown in Fig 2. Note that although we are using the SSID field for the short data streaming service, the Basic Service Set Identifier (BSSID) of the beacon remains invariant. One octet is always reserved as a **unique identifier** for this scheme, another octet specifies the **channel number**, two octets are assigned for the **fragment number** while another two for the **total number of fragment**. The remaining 26 bytes form

[1]Note: Channels imply a logical structure used by the server to indicate the type of data streamed. e.g. one data stream related to network load, another related to general messages and alerts and yet another for sending security/authentication information. It can also be used to indicate that the data is meant for a particular client device or group of devices.

the effective payload. In our implementation, each channel broadcasts data for a different purpose.

In this scheme, the data is broadcast in periodic cycles. A *broadcast cycle* is considered complete when all data fragments of each channel has been broadcast once. The periodic repetition of fragments over successive cycles allows asynchronous operation between the server and the client. It also makes the transmission reliable, even though the beacon frames are not acknowledged explicitly. In our implementation, channels can be added or deleted only after the completion of at least one broadcast cycle.

3.2 Enhancements

3.2.1 Multiple BSSID

The 802.11 standards specify the use of multiple frequency bands for data transmission. Thus the beacons from a particular device are broadcast in a fixed frequency and can be received only on the same frequency. The client device initiates periodic scans to discover new networks and during this process it spends a predetermined time (called dwell time) in each frequency band. Thus if 4 frequency channels are in use, a client device will not be able to capture more than 1/4 of the beacons that were transmitted during the scan time. There are two more factors that further reduce the number of beacons seen by the client. Firstly, assuming that a client receives some beacons with the same BSSID during its scan of a particular band, we observe that the NIC implementation pushes only the last received beacon to the MAC layer. This is in spite the [BSSID,SSID] tuple having unique SSIDs. Thus the number of beacons seen by the client's MAC in one scan cycle is the number of beacons with different BSSIDs. Moreover, in order to reduce processing requirement at the receiver, the client updates network information only from beacon frames received during scanning while it discards beacon frames received when it is not in scanning state. Thus we see that if we are to use the beacon frame for our short data streaming service, we need to plan for sufficient redundancy owing to a small fraction of the beacons being actually received by the client.

The first and third factors are immutable parts of the standard. However, we can increase the throughput of the broadcast system if we can exploit the BSSID field. As an enhancement to the BOWL scheme, we ensure that successive beacons have different BSSIDs, which are chosen from a predefined set. This ensures that multiple beacons are reported to the MAC during a single scan. The use of multiple BSSIDs is illustrated in Fig 3. The duration after which the set of data fragments, transmitted in the SSID field, are refreshed is referred to as the *fragment refresh interval*. Since the number of different BSSIDs used controls the number of beacons reported, the throughput of the system scales proportionately. In the next section we show the effect of increasing the number of BSSIDs used on the throughput. From a client's perspective the different BSSIDs are interpreted as multiple networks. Fig 4 shows how these virtual networks appear to the user. During our implementation, we observed that the use of more than 20 different BSSIDs led to buffer overflows on the NIC.

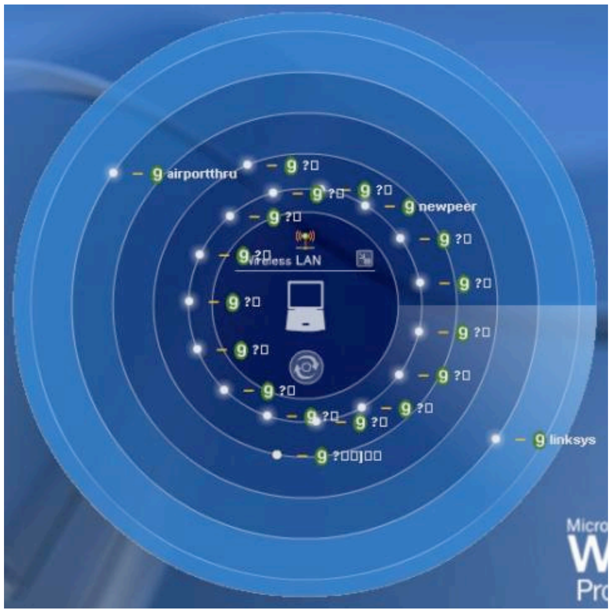


Figure 4. Detection of multiple BSSID by the client device

3.2.2 Frame Recovery

In the BOWL scheme a frame loss can occur for two reasons: (1) frames are received but discarded because of data corruption or (2) frames are not received because the sender and receiver are not in sync. As a result of this a client might require more than one cycle to receive all the fragmented frames, thereby decreasing the overall throughput of the system. We describe a set of techniques below which when used in combination will provide a robust frame loss recovery mechanism.

To recover from burst errors causing loss of frames we use the following scheme. Assuming that the data for a particular channel can be fragmented into n parts denoted by $F_1, F_2, ...F_n$, we group the fragments into G groups of s fragments each. Thus $G = \lceil n/s \rceil$. For each group G_i, where $1 \leq i \leq \lceil n/s \rceil$, we generate a parity frame PB_i according to Eq 1, where $F_i(k)$ and $PB_i(k)$ represents the k^{th} byte of the fragment and parity frame respectively.

$$PB_i(1) = F_{s*(i-1)+1}(1) \bigoplus F_{s*(i-1)+2}(1) \bigoplus ...F_{s*i}(1) \quad (1)$$

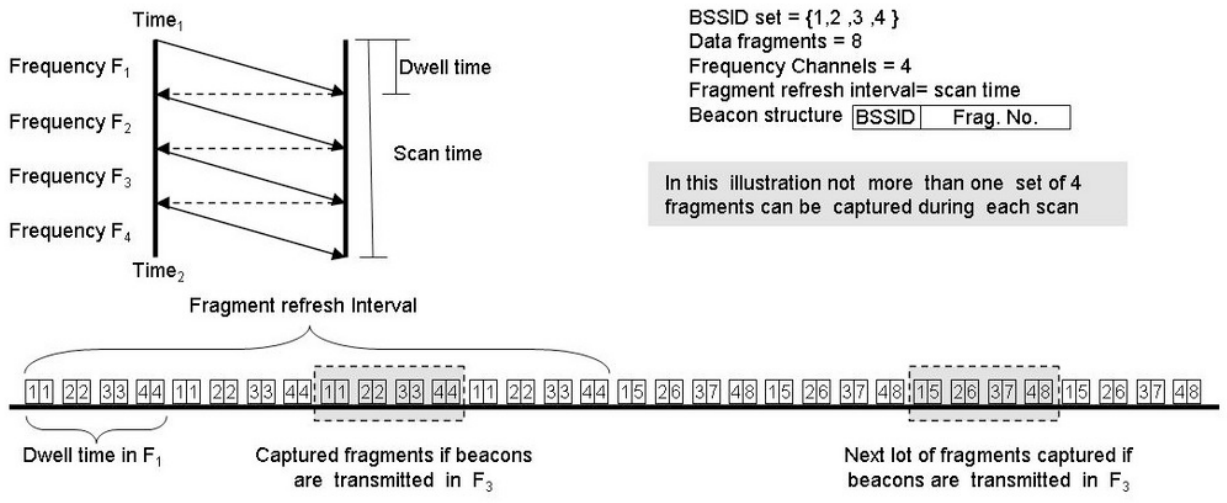

Time$_1$
Frequency F$_1$
Frequency F$_2$
Frequency F$_3$
Frequency F$_4$
Time$_2$
Dwell time
Scan time
BSSID set = {1,2 ,3 ,4 }
Data fragments = 8
Frequency Channels = 4
Fragment refresh interval= scan time
Beacon structure BSSID Frag. No.
In this illustration not more than one set of 4 fragments can be captured during each scan
Fragment refresh Interval
Dwell time in F$_1$
Captured fragments if beacons are transmitted in F$_3$
Next lot of fragments captured if beacons are transmitted in F$_3$

Figure 3. Beacon Sequencing with multiple BSSID

Further the transmission sequence is reordered so that a burst loss of upto $\lceil n/s \rceil$ fragments can be recovered, without waiting for another cycle. The transmission sequence therefore transmits one element from each successive group with the parity fragments being transmitted after every s fragments.

To illustrate the algorithm with 12 fragments, 3 groups with 4 fragments each, $G_1 = F_1, F_2, F_3, F_4$; $G_2 = F_5, F_6, F_7, F_8$; $G_3 = F_9, F_{10}, F_{11}, F_{12}$, three parity fragments are generated viz PB_1, PB_2 and PB_3. In this case the transmission sequence is F_1, F_5, F_9, F_2, PB_1, F_6, F_{10}, F_3, F_7, PB_2, F_{11}, F_4, F_8, F_{12}, PB_3. The algorithm ensures that if multiple frames from the transmission sequence are lost, they can still be recovered from the remaining frames. For example, if F_6 and F_{10} are lost, they can be recovered by XOR ing F_5, F_7, F_8, PB_2 and $F_9, F_{11}, F_{12}, PB_3$ respectively.

Using this technique, we are able to recover one element per group. Note that a higher number of groups ensures better resilience against burst errors, but also reduces the throughput because of additional parity frames.

4 Performance Analysis

4.1 Experimental Setup

The experimental setup consists of DLink 802.11b/g USB transmitters [3] operating on Redhat Linux 7.2 based desktops as the host systems. The Broadcast Server operates in ad-hoc mode and transmits HTML files of different sizes (4KB & 8KB). In case of multiple BSSIDs the server transmits all of them in a burst. For each file size transmitted by the server, receiver calculates the total time (in seconds) taken to receive the entire file. The server uses different BSSID sizes ranging from 1 to 16 to transmit each file as shown in Table 2 and 3. For each file size and BSSID number combination the experiment is repeated a number of times to obtain average time taken to receive the transmitted file. However file transmission introduces a dependency on the sequence in which frames are received. Therefore to measure the raw throughput, we transmitted data chunks rather than a file. The raw throughput measurement is shown in Fig 5.

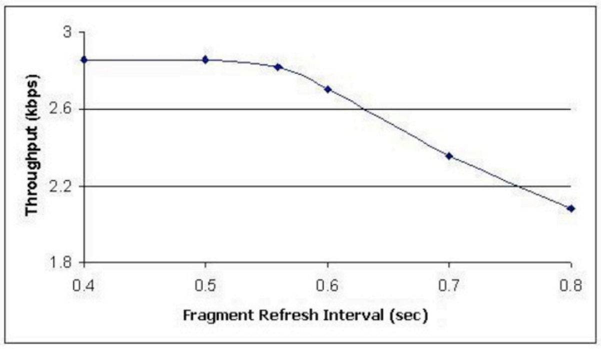


Figure 5. Throughput against Fragment Refresh Interval

4.2 Throughput Measurement

Fig 5 shows the variation in throughput with respect to the Fragment Refresh Interval. To explain the characteristic behavior, we consider the parameters from Table 1.

Parameter	Notation	Exp.Value
Number of frequency bands	N	14
Dwell time per frequency band	T_{dwell}	$40ms$
Scan time	$T_{scan} = N * T_{dwell}$	$560ms$
No of BSSID used	B	8
Frag. Refresh Rate	T_{frag}	0.4 to $0.8s$
Beacon Interval	T_{beacon}	$20ms$
Payload per Beacon Frame	p	26 bytes
Data to be transferred	D	$8KB$
No. of Frag.	$\lceil D/p \rceil$	316
Group Size	s	20
Number of groups	$\lceil n/s \rceil$	16

Table 1. Experimental Parameters

As we mentioned earlier, the maximum number of beacons pushed to the MAC layer per scan time equals the number of beacons with different BSSIDs i.e. B frames per T_{scan} time interval. Thus the maximum effective throughput that can be achieved is $\frac{B*p*8}{1000*T_{scan}}$ kbps. Using the actual values from Table 1, we obtain the theoretical maximum of *2.9 kbps*. From the graph it is clear that as long as beacons with B different BSSIDs can be transmitted within a scan time, the maximal throughput can be achieved. However a higher Fragment Refresh Interval also affects the effective throughput, which can be expressed in Eq 2

$$Throughput = \frac{B * p * 8}{\max(T_{scan}, T_{frag})} bps \qquad (2)$$

4.3 File Transfer

Table 2 corresponds to measurements made without incorporating the error correcting scheme. We observe that the average time taken to receive the transmitted file is inversely proportional to the number of different BSSIDs used. Table 3 captures the results with the frame recovery scheme, with a single BSSID in use and a file of 8Kb being transmitted. Errors were introduced into random frames by the server and were therefore discarded by the client as invalid frames.

5 BOWL Application

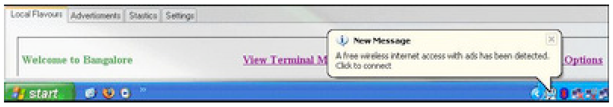
To showcase the utility of this technology, we developed a pilot application. The Broadcast Server was developed on a system running Red Hat Linux. We used DLink 2100g chipsets, with suitable driver modifications to allow programmability of the SSID field. The client side was implemented in C#.NET with suitable libraries to retrieve the beacon frames and was run on a standard windows XP machine.

Size of BSSID set	Filesize	
	4K	8K
1	109.68	226.143
2	55.045	115.179
4	27.109	53.485
8	11.995	27.825
16	7.321	12.297

Table 2. Average file transmission time (in sec) while increasing the BSSID set

Error Rate	Without Frame Recovery	With Frame Recovery
1%	350.939	231.962
2%	359.229	258.925
3%	369.772	296.095
4%	414.569	327.191

Table 3. Average file transmission time (in sec) in presence of induced error



New Message

A free wireless internet access with ads has been detected.
Click to connect

Figure 6. Screenshot of BOWL being used to connect to a wireless LAN

Consider the following scenario which is the basis for the pilot application: *Helen is at Gate 18 in the Bangalore airport, waiting for her connecting flight and would like to use her wireless connection to e-mail. Unfortunately, she is not too tech savvy and does not know how to connect onto the available wireless network. Further, she would like to know where she could find the coffee shop nearest to her terminal. In a world empowered with BOWL technology all she has to do is switch on her device and BOWL would instantly notify her of all available services in the airport, in this case a free internet connection . With a click of a button she is automatically connected to the wireless internet service [Fig 6]. While she is surfing she also has access to a directory* *service that allows her to view the facilities available at the airport [Fig 7] as well as promotional offers [Fig 8].*

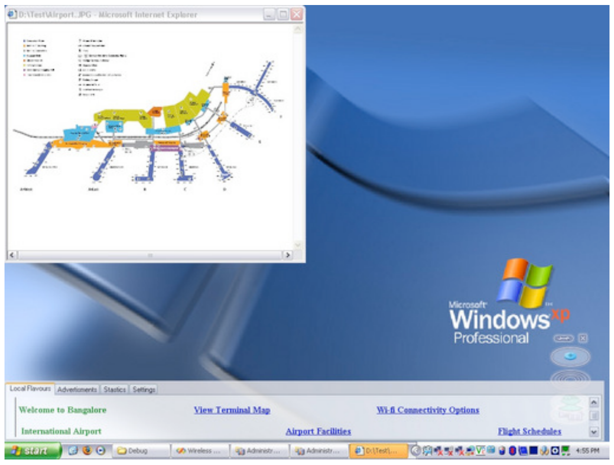


Figure 7. Screenshot of BOWL being used to transmit maps

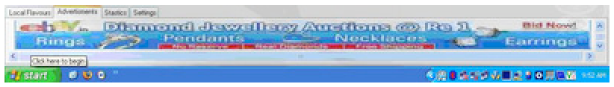


Figure 8. Screenshot of BOWL being used to advertise

6 Conclusion

Ubiquitous access technologies are essential to realize true mobility. In this work we have presented a mechanism to overload beacon frames to achieve a low bandwidth connectionless data transfer stream. Using this innovation we have improved upon the pervasive quotient of the existing Wi-Fi standard. We have further derived the optimum rate at which these beacons need to transmitted as well as described enhancement techniques aimed at maximizing the throughput. We have also demonstrated a sample application that highlights the enriched user experience of this scheme.

References

[1] Mark Weiser, ”The Computer for the 21st Century” - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991

[2] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11, 1999.

[3] Dlink Wireless G USB Adapter DWL-G122 driver www.ralinktech.com/ralink/Home/Support/Linux.html

[4] D. Di Sorte, M. Femminella, G. Reali, ”Service publishing to support an efficient 802.11 network selection,” Proc. of 6th Intl. Workshop on Applications and Services in Wireless Networks (ASWN), Berlin, Germany, May 2006.

[5] Y.W. Lee, S.C. Miller, Network Selection and Discovery of Service Information in Public WLAN Hotspots, 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots, Philadelphia, USA, 2004.

[6] The WiFi Alliance http://www.wi-fi.org/