Title
Sync-scan: A fast hand-off procedure for 802.11 link layer roaming

Authors
Ramani, Ishwar
Savage, Stefan

Publication Date
2004-05-03

Peer reviewed

SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks

May 3, 2004

Abstract

Wireless networks scale by stitching together services from multiple overlapping base stations. To provide location transparency, individual clients must seamlessly "hand-off" between base-stations as they move across the network. However, in many such systems, the overhead of locating the candidate base stations limits the continuity of hand-off - making it inappropriate for highly interactive applications such as voice. In this paper, we describe a technique for minimizing this cost by synchronizing short listening periods at the client with periodic transmissions from each base station. We have implemented this SyncScan algorithm for 802.11 infrastructure networks and show that we can practically support low overhead handoff in this environment. Moreover, our approach only requires trivial implementation changes, is incrementally deployable and is completely backward compatible with the existing 802.11 standard.

1 Introduction

Wireless broadcast networks, by their very nature, provide the opportunity for user mobility. Within the range of a given wireless base station, a client may roam freely and with complete transparency to the network medium. It is this exactly this capability that has driven the success of the $1.5 billion 802.11-based wireless network market [Gro03]. Inexpensive 802.11-based access points provide transparent connectivity to the wired Internet at low cost and with minimal configuration overhead. However, each individual 802.11 access point (AP) has a limited range - frequently under 100 meters indoors - and therefore large-scale deployments of access points are required to provide comprehensive coverage of a building or campus (as seen in Figure 1).

Preserving the same network transparency across collections of such access points requires a far greater

Figure 1: A map of wireless coverage over a large University campus. The entire coverage area is serviced by over 250 distinct access points

degree of coordination and management. As a client moves outside the range of one access point it must "hand-off" to another to preserve the illusion of seamless connectivity. In 802.11-based systems, this handoff process is managed autonomously and independently by each client [sC99]. Unlike traditional cellular data and voice systems, 802.11 networks do not provide information about the local network topology. Instead, when a client attempts a handoff, it must temporarily abandon its current access point, actively probe the network to discover alternatives, and only then reconnect to the current best AP. This stateless mode of discovery minimizes management overhead and is highly robust to failure, but can cause temporarily "gaps" in connectivity of up to a second in duration [MSA02, VK03, FKABAD03]. While such disruptions may be acceptable for nomadic applications with limited mobility and flexible response time requirements, emerging applications like wireless voice-over-IP are far more demanding [Hen01, CIS]. Such applications require highly interactive response *during* mobility and are extremely sensitive to network outages and delays. Moreover, the

limited range of 802.11 radios makes hand-off actions quite common for continuously mobile clients (such as a user walking with an 802.11-based phone).

In this paper, we propose a new mechanism for dramatically reducing the overhead of hand-off for such applications. Our solution, called SyncScan, replaces the large transient overhead of active access point discovery with a continuous process that passively monitors other channels for the presence of nearby access points. The potential disruption of channel switching is minimized by synchronizing short listening periods at the client with regular periodic transmissions from each access point. We have implemented the SyncScan algorithm using commodity 802.11 network interfaces and show that we can reduce the cost of handoff by up to an order of magnitude compared to the existing approach – enough to enable mobile voice applications. Finally, our approach does not require any modification to the 802.11 protocol itself, is incrementally deployable and requires only minor modifications to existing implementations.

The rest of the paper is organized as follows. In Section 2 we review how existing 802.11 hand-off mechanisms function and previous work in improving their overheads. Section 3 describes the SyncScan algorithm and its optimizations, followed by an experimental analysis of our prototype implementation in Section 4. Finally, we summarize our results and open questions in Section 5.

2 Background and Related work

The hand-off process has several phases in 802.11 networks – each with its own costs. Figure 2 depicts this time-line graphically.

First, a client must *determine* that it is nearing the periphery of its coverage and must find an alternative access point to continue. Minimally, this involves detecting that packets are no longer being successfully received. However, typical implementations also monitor the current signal-to-noise ratio (SNR) and will also initiate the scanning phase when this value passes a predefined minimum threshold. Setting this threshold is something of a black art: if the client waits to long to look for new access points then it may incur additional disruption, yet if the client is too eager then it may ping-pong between access points needlessly.

Once the scanning phase is initiated, the client

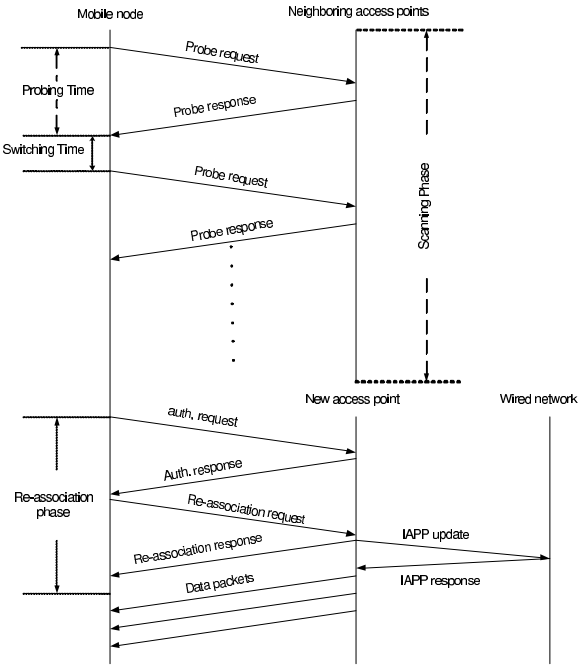

Figure 2: A timing chart of the 802.11 hand-off procedure. Three phases: scanning, authentication and association are illustrated.

searches each channel (11 in 802.11b, 3 in 802.11g, and 8 for 802.11a indoors) for potential access points. This scan can be either active or passive. In the former the case, the client switches channels and listens for special *beacon* packets periodically generated by access points to announce their presence (typically every 100ms). However, to reduce the response time, most implementations actively broadcast a probe packet on each channel to force an access point to respond immediately. The delay incurred in completing an active scan is regulated by five parameters:

- *Channel switching time*. This is a fixed physical characteristic of the wireless hardware and reflects the time for the network interface card (NIC) to switch to a new center frequency, resynchronize and start demodulating packets. For widely deployed hardware, we have measured this value to be between 5-10ms.

- *ProbeDelay*. Upon settling on a new channel, the NIC passively monitors for channel for a specified number of microseconds before sending a probe packet. Typically this value is negligibly small.

- *Media contention*. The 802.11 MAC protocol provides a distributed contention mechanism that can

cause a probe packet to be delayed seeking access to the medium. On lightly loaded networks, this value is negligibly small.

- *MinChannelTime*. This represents the amount of time to wait for the first response before declaring the channel empty (i.e., no access point in range). This value is an adjustable parameter and is set differently by different systems.

- *MaxChanneltime*. This represents the amount of time to wait to collect potential *additional* probe responses from other access points. This value is meant to be configured based on an estimate of the number of overlapping access points and the load on the channel.

While the exact overhead of scanning can vary significantly based on the environment and the choice of these parameters, for popular Intersil-based hardware average values are typically between 350-400ms. In our environment, passive measurements of Intersil-based cards from LinkSys and DemarcTech revealed average scanning times of approximately 400ms.

Based on the measured SNR for the probe response packet received from each access point, the client will then select the best new access point for *authentication* and *association*. Authentication is required to validate the client's right to use a particular access point and minimally requires a two way handshake (the use of shared keys expends this to a four-way handshake). Following authentication, Association is the process of binding the client with a particular access point. Both authentication and association typically involve the reliable exchange of two messages and therefore require two round-trip times between the client and the chosen access point. These values are typically under 1ms each in our measurements, although others have measured a slightly wider range of values. Association may sometimes be in the form of a re-association, when the node switches access point in the same service set (access points operating with the same SSID).

Finally, association triggers an Inter-Access Point Protocol (IAPP) session to notify the wired networks that future packets directed to the client should be routed through the associated access point and to transfer any state from the old access point [sC02]. In many implementations the IAPP is implicitly implemented by the wired data-link layer (typically Ethernet), although several vendors do provide an explicit IAPP mechanism for transferring buffered packets from the old access point.

PHASE	TIME
Scanning	350-400ms
Authentication	<10ms
Association	<10ms
IAPP	<20ms

Table 1: Rough overheads for current hardware engaged in each stage of the hand-off process. These values represent a combination of our direct measurements and previous studies; particularly [MSA02, VK03, FKABAD03].

Table 1 summarizes these costs. While these values can vary (and we refer the reader to [MSA02] for a more in-depth measurement analysis of these overheads), the overall conclusion does not. The scanning phase completely dominates the cost of hand-off – usually contributing more than 90 percent of the overhead.

There have been several previous attempts to minimize 802.11 hand-off latency. Velayos et al. and Pack et al. both propose a scanning mechanism using topographical knowledge of access point placement [VK03, SP02]. If each client is made aware of its neighboring access points and their channels, the the client can probe a reduced set of channels during the scanning phase. While this approach is attractive, the maintenance and dissemination of this knowledge represents a significant management burden for those deploying large-scale 802.11 networks. Moreover, this class of solution also requires changes to the 802.11 protocol itself and would therefore create significant backward compatibility issues.

Closer to our work, Misra et al. postulate several potential methods for improving hand-off overheads [MSA02]. Among these is the idea to interleave passive scanning with normal connectivity. It is exactly this idea that we have independently pursued.

3 SyncScan

802.11-based access points periodically broadcast special beacon packets to identify themselves to potential clients and to synchronize state information with currently associate clients. While it is common for these beacons to be sent every 100ms, the 802.11 standard does not specify particular controls over the absolute time at which such packets are generated. We pro-

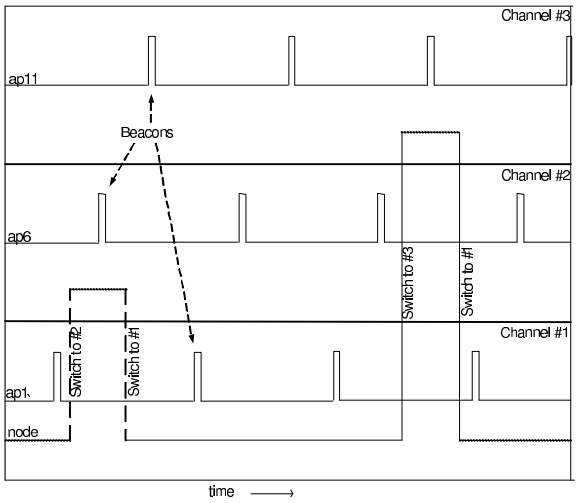

Figure 3: Client associated with AP1 uses SyncScan to receive beacons from AP6 on channel 2 and AP11 on channel 3.

pose to exploit this degree of freedom to synchronize clients with the timing of beacon broadcasts on each channel. At the heart of our proposal is the creation of a staggered periodic schedule of beacon periods spread across channels. For example, all access points operating on channel 1 will broadcast beacons at time t (or close to it), while access points on channel 2 do the same at time t+d and channel 3 access points send at time t+2d. Thus, a mobile client associated with an access point operating in channel c can detect APs operating in channel c+1 by switching to that channel d ms after receiving a beacon from its own access point.

A mobile client can utilize this property to efficiently locate all the access points in its neighborhood. By regularly switching to each channel (as shown in Figure 3), a complete picture of all nearby access points can be observed and yet the client minimizes the time it is out-of-contact with its own access point. Consequently, when a hand-off must be attempted, the cost is reduced to that of authentication, association and IAPP. Moreover, since APs are being continuously monitored there is the opportunity use continuous changes in observed SNR (Rather than a single sampled value) to make better AP choices.

Clearly, this procedure is backward compatible since it will co-exist with clients using the current scanning procedure. It is also incrementally deployable, since a client employing SyncScan can always fall back to the traditional means of hand-off if it cannot synchronize with the infrastructure. Moreover, since it only changes the *timing phase* of beacon generations it does not require any change to the 802.11 standard itself.

However, SyncScan does add several complexities. First, the accuracy of clocks in access points is critical to the global synchronization of beacon timings. Over short time-scales our experiments reveal the drift is negligible for existing access points. Over longer time-scales access points require a means of time synchronization. By far the simplest approach is to leverage the wide availability of Network Time Protocol (NTP) service over the Internet [Mil92]. This would provide a simple out-of-band means for synchronizing APs to absolute time reference (in fact, many commercial access point products already support NTP) although it would also be possible to develop more complex methods of self-synchronization in environments where NTP was not appropriate.

Another problem is that multiple access points operating in the same channel might interfere with one another's beacons. To address this issue, the beacon generation time may be randomly varied over a small window (e.g. 5ms). A client lingering on a channel for the entire window should expect to receive most of the beacons on that channel.

Finally, the SyncScan procedure has a hidden cost. While it removes the transient overhead of the scanning phase, it replaces it with a regular overhead. While a client is listening to other channels, it cannot be sending or listening to its own access point. Moreover, the client may miss packets sent to it while it is exploring other channels. How significant these factors are depends, in part, on the overhead of implementing the SyncScan procedure. For each SyncScan session, the client must switch channels, wait for any beacon and then switch back. Thus, the SyncScan time is $2\times$ channel switch time + wait time. How often this overhead in incurred is a function of how many channels are scanned and how regularly a scan is initiated. In the next section we will evaluate the factors in the context of our prototype implementation.

4 Implementation and evaluation

We implemented a SyncScan prototype using the popular hostap driver [Mal] for Intersil-based wireless NICs. The hostap driver configures the NIC firmware to allow the host to explicitly implement 802.11 management functions, such as scanning, beacon monitoring

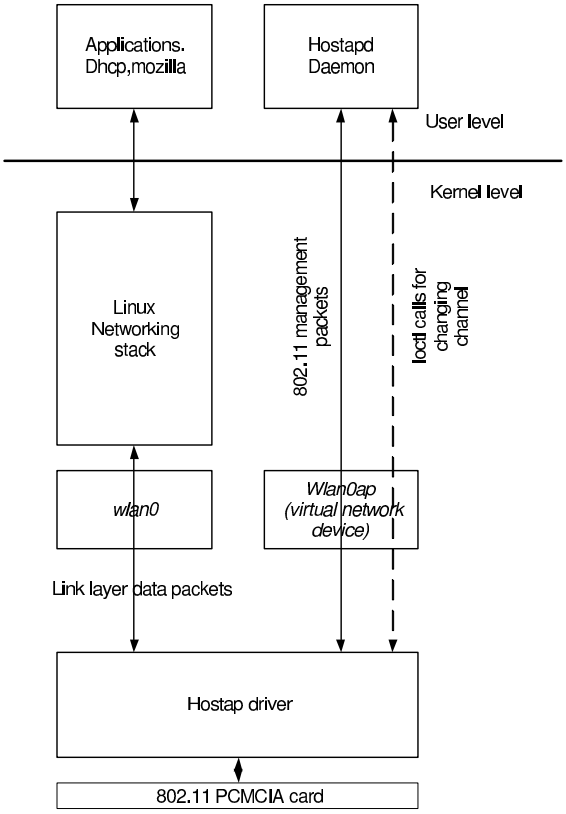

Figure 4: Prototype SyncScan architecture in Linux. The hostap driver was modified to handle packet flow through both wlan and wlan0ap.

and association. The architecture of our SyncScan prototype is illustrated in figure 4. The standard hostap implementation installs a hostap driver at the kernel level and a hostapd daemon at the user level. The driver handles packet transmission and reception while hostapd handles the 802.11 access point functions. The driver communicates with the daemon using a virtual network device for management control. As Figure 4 shows, we modified the driver to multiplex packets between the actual network device and the virtual network device. This implementation, with the card set in promiscuous mode, allowed the card to send and receive packets from both the networking stack (wlan0) and the hostapd program (wlan0ap). This was necessary so the host could send and receive its own packets as well as emulate 802.11 client management functions.

While our implementation is sufficient to demonstrate the SyncScan idea, it is far from ideal. In particular, there are limitations in the host interface to the

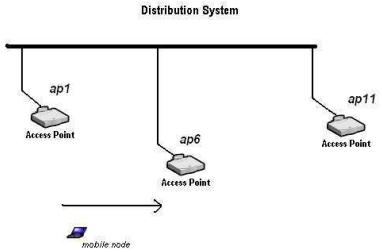

Figure 5: Experiment setting of three access points in the corridor of our department.

802.11 NIC that require a complete card reset after every channel switch. On our Linksys WPC11 PCMCIA card, using the latest firmware, we measured a channel switch time of 12ms. In addition, the card requires another 7ms to stabilize before it will receive packets on that channel. Using a channel waiting time of 10ms, this produced a total SyncScan overhead of 48ms. We believe that a firmware implementation would be dramatically faster. By comparison, using a recent card from Atheros, we were able to switch channels in under 6ms and stabilize in under 3ms (however, we do not yet have a complete SyncScan implementation for this card). Clearly, these overheads do not reflect any physical limitations of 802.11 radio technology but rather the current demands of the commercial market. In our implementation, we actively scanned the three orthogonal channels of 802.11b. These are channel 1,6 and 11. Since most of the existing configurations of access points work in these three channels, we exploited this feature to save on SyncScan performed in other channels.

Since the SyncScan overhead in our prototype is greater than the maximum retransmission time for 802.11 frames, it is possible for a client to lose a packet sent while the client was listening on another channel. To overcome this drawback, we modified the client to announce that it was entering Power Saving Mode (PSM) just before switching channels [sC99]. This causes the access point to buffer any packets destined for the client until it returns to the channel. While the client is scanning, it buffers any outbound packets to ensure that they are not lost also. This technique is sim-

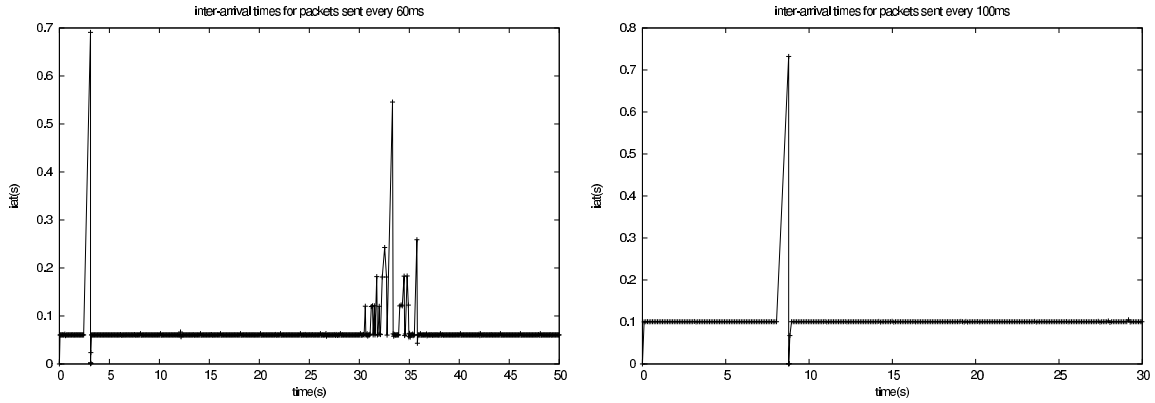

Figure 6: plots of inter-arrival time(iat) for 60 and 100ms when the mobile node is operating with the standard driver

ilar to that of Bahl et al. used to create virtual wireless NICs [PBC03].

For our experiments we did not implement synchronized access points. Instead, we simulated their effect at the client. When the SyncScan client first starts, it explicitly synchronizes with all available access points by waiting on each channel for a beacon and recording the time of its arrival. Once this synchronization is accomplished it associates with strongest access point and allows the host to transmit and receive. Based on the schedule calculated from the initial beacons hostapd periodically commands the NIC to switch to alternate channels and intercept any beacons broadcast there. As the client modes it keeps track of the signal strengths of each of its access points.

Our experiments were performed in a corridor in our department. The corridor was long enough to warrant three access points which operated at channel 1, 6 and 11 as shown in figure 5. The mobile client was a Dell Inspiron 6000 running Linux, using a Linksys WPC11 PCMCIA card for 802.11 connectivity. Another laptop, a Sony VAIO Z505 with the same configuration acted as a packet generator destined to the mobile client.

In the remainder of this section we perform two experiments. First, we simulate a real-time application by streaming packets to the mobile client at regular intervals. We examine the packet loss and jitter that occurs while the client is idling and handing-off using the conventional hand-off approach, SyncScan and SyncScan + PSM. this experiment is meant to demonstrate the potential for providing continuous connectivity using SyncScan. Second, we look at the impact of SyncScan on traditional bulk data applications. We transfer a file using FTP and compare the bandwidth delivered to a normal client and one using SyncScan + PSM. This is a worst-case scenario for SyncScan – packets are being transmitted during the scan – and we investigate it to understand if SyncScan has a significant detrimental effect.

4.1 Real-Time Traffic

To analyze the performance of SyncScan with real-time interactive traffic we used ICMP Echo Requests generated at periodic intervals. The packet generator fed these packets into AP1 and traces were gathered at the mobile client. The experiment was conducted by moving the mobile client from the neighborhood of AP1 to AP11. The mobile client switched access points at most twice: from AP1 to AP6 and maybe from AP6 to AP11. The handoff from AP6 to AP11 depended on the trajectory of movement. Hence it did not happen in some cases.The parameters of SyncScan were set as follows: The $waittime$ for the beacon capture was set to 10ms and SyncScan performed very aggressively – every one second (i.e., in two seconds the mobile node has knowledge of all the access-points). We performed two instances of this experiment: one with a 60 ms inter-arrival period between packets, the second with 100 ms.

As a basis for comparing the performance of SyncScan with the existing scanning procedures, we first measured the arrival times of the ping packets when the mobile node is using the Linksys WPC11 card operating in Windows XP. Figure 6 presents the inter-arrival times for packet streams generated at 60 and 100 ms. In the first case with the 60ms ping packets, the card switched access points twice. In the second

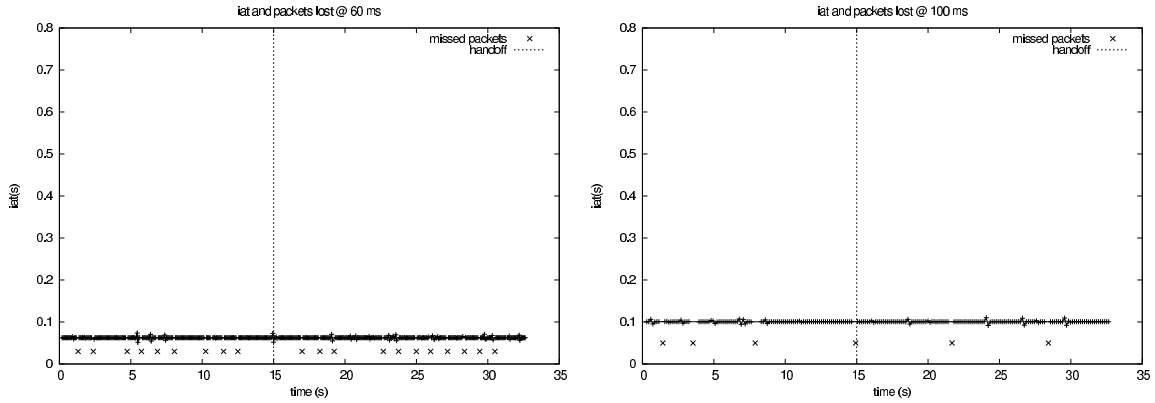

Figure 7: inter-arrival time(iat) when the mobile node is running on SyncScan. The crosses indicate packet loss

case with 100ms ping packets, the card switched access point once. We also observed a substantial loss of up to 13 packets at the 35 second handoff in the 60ms graph. Notice that when handoff takes place at 5 and 35 seconds for the 60ms graph and at 9 seconds for the 100ms graph, there is considerable variability in the inter-arrival time. The delay is caused due to the scanning and re-association performed by the card driver at this stage. As observed in this particular case, the latency of scanning is around 700ms.

An interesting observation in the graph for 60ms pings is the high variability around the handoff at 35 seconds. We believe this happens because the handoff algorithm in the card keeps switching back and forth between AP6 and AP1. This could happen when the signal strength registered from both the access points are roughly the same which makes it difficult for the card to choose between the two. Another interesting feature is the buffering of packets once the card deauthenticates with the old access point. This happens at the 5 second handoff in the 60ms graph and 9 second handoff for the 100ms graph. These packets are delivered to the new access point with which the card re-associates. This feature explains the sudden drop in the arrival times after the long delay during handoff.

Figure 7 shows the same experiment using SyncScan. The plotted values for the two streams show that the inter-arrival time is around the expected values. The crosses marked above the x-axis denote the packet loss. The line at 15 seconds indicate the place where the handoff was performed. Comparing these graphs with Figure 6 clearly shows that the variability is an order of magnitude smaller during the handoff. The SyncScan operations are performed periodically, every second in these experiments. When a ping request coincides with the SyncScan operation it is not received by the mobile node. These lost packets are indicated by an X at the time when they occur. As expected this happens more often in the 60ms packet stream since the probability of missing a packet is inversely proportional to the inter-arrival time and directly proportional to SyncScan time.

Another factor that affects the packet loss is the frequency of SyncScans. As the graph shows the packet loss drops from the 60ms to the 100ms stream . By varying the rate of the packet stream we noticed that packet loss becomes negligible around 150ms. We would like to point out that it is not possible to avoid packet loss since there is always a probability of packets getting sent during the SyncScan operation. But this loss, as pointed out, would become minimal when the SyncScan time is further reduced.

Figure 8 repeats the experiments using SyncScan+PSM. The graphs are very similar to the previous graphs using SyncScan. But the difference is in the absence of packet loss which is due to packets getting buffered at the access point during the SyncScan operations. As observed in the previous case, the variation in arrival time is minimal compared to the normal handoff.

Figure 8 also shows the cumulative distribution of the inter-arrival times for SyncScan+PSM. As seen 97% of the packets arrive at the right time and the remaining vary between the expected time and expected time plus the time for SyncScan. This happens because of the delivery of buffered packets after the completion of the SyncScan operation. This delivery can delay the arrival of the packets by SyncScan time in the worst case. Since the current implementation of Sync-

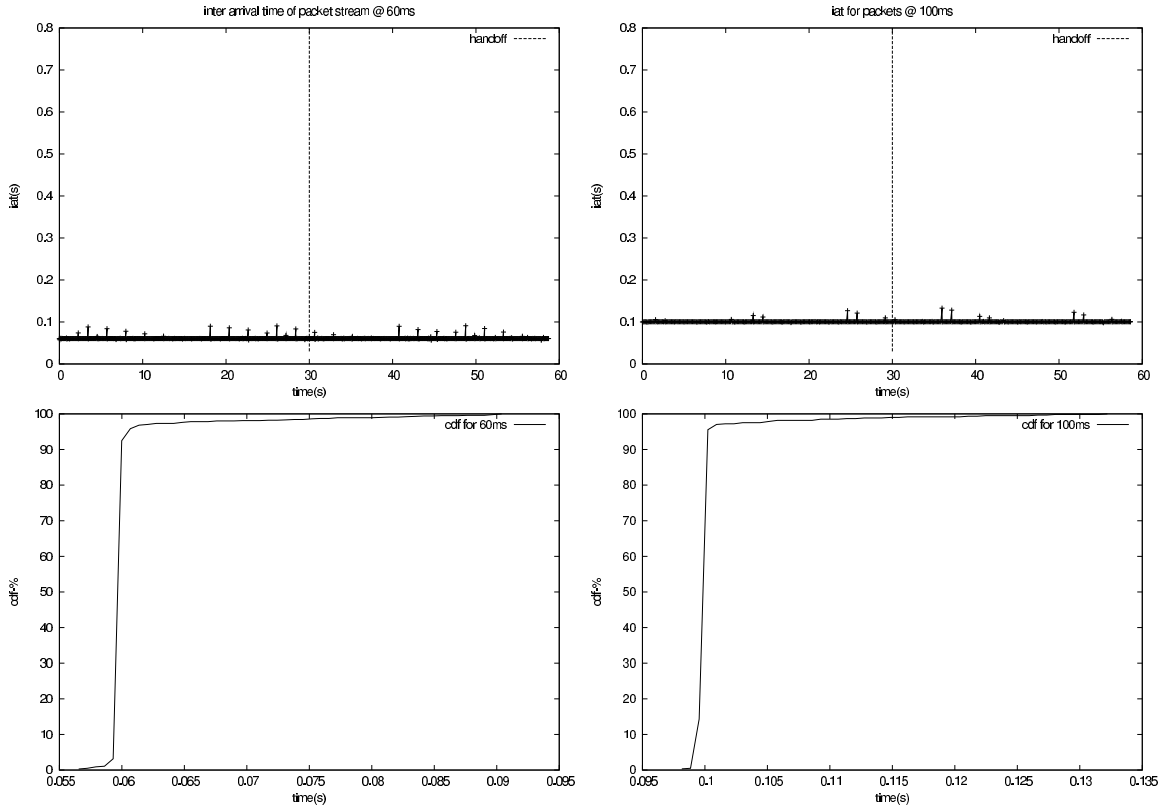

Figure 8: inter-arrival time(iat) and CDF of packets in SyncScan+PSM mode for 60 and 100ms packet streams and

Scan takes 50ms, the worst case delay is around 50ms. In real-time traffic this translates into additional jitter. Thus the performance benefits of SyncScan in PSM mode are zero packet loss with a trade off of additional jitter which is within limits for real-time traffic.

4.2 Bulk data traffic

We also analyzed the performance of TCP/IP over SyncScan using FTP. We compared the performance of FTP of the same file over SyncScan with PSM and the normal 802.11 infrastructure mode. In both cases we used ethereal to capture the traffic at the receiver. The file size was 1.5MB and it was fetched from a remote site. The results are plotted in Figure 9. The first graph shows instantaneous bandwidth in 50ms buckets. During the SyncScan operations the bandwidth drops to a smaller value at 0.7 seconds and 1.9 seconds. But due to PSM buffering the traffic is not affected. The second graph is a CDF of packets received over running time. Both the CDFs run parallel to each other indicating the negligible effect of SyncScan on the traffic.

5 Conclusion

Mobile voice applications are the next challenge for 802.11-based wireless networks. One of the major impediments is the high cost of hand-off as clients roam between access points in an infrastructure network.

In this paper we have proposed using implicit time synchronization to reduce the key cost of discovering new wireless access points. By synchronizing the announcement of beacon packets a client can arrange to listen to other channels with very low overhead. As a result, hand-off using this SyncScan approach is an order of magnitude quicker than using the conventional approach.

We have designed and implemented the SyncScan hand-off procedure using existing commodity 802.11 hardware. In spite of the limitations of existing hardware, we have been able to demonstrate the dramatic benefits of SyncScan for interactive real-time traffic and that it has minimal impact on existing bulk-data applications.

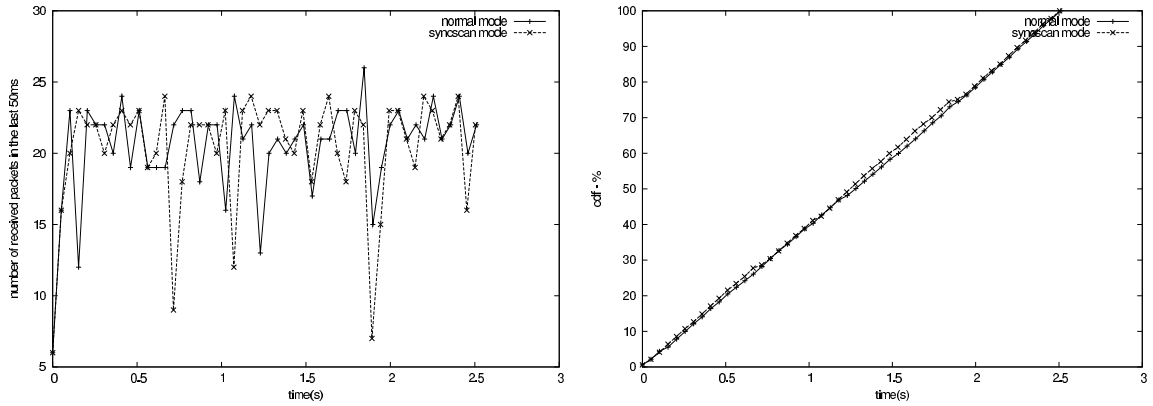

Figure 9: instantaneous bandwidth and CDF of packets for FTP over SyncScan+PSM and normal mode. The instantaneous bandwidth is measured in number of packets every 50ms

References

[CIS] CISCO. Cisco 802.11b based ip phone. http://www.cisco.com/warp/public/cc/pd/tlhw/.

[FKABAD03] Prasad Boddupalli Fahd Khalaf Al-Bin-Ali and Nigel Davies. An inter-access point handoff mechanism for wireless network management: The sabino systsem. In *Proceedings of The 2003 International Conference on Wireless Networks*, Las Vegas, Nevada, June 2003.

[Gro03] Dell'Oro Group. Dell'oro group's wireless lan five year forecast report, jul 2003.

[Hen01] T. Henriksson. Hardware architecture for 802.11b based h.323 voice and image ip telephony terminal. In *Swedish system-on-chip conference2001, Proceedings of the SSoCC*, Arild, Sweden, March 2001.

[Mal] Jouni Malinen. Host ap driver for intersil prism2/2.5/3. http://hostap.epitest.fi/.

[Mil92] D.L Mills. Network time protocol (version 3) specification, implementation and analysis. In *Network Working Group Report RFC-1305*, March 1992.

[MSA02] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the ieee 802.11 mac layer handoff process. Technical Report UMIACS-TR-2002-75, University of Maryland, College Park, Nov 2002.

[PBC03] P. Bahl P. Bahl and R. Chandra. Enabling simultaneous connections to multiple wireless networks using a single radio. Technical Report MSR-TR-2003-46, Microsoft Research, June 2003.

[sC99] IEEE Computer Society LAN MAN standards Committee. Ieee standard for information technology: Part 11: Wireless lan medium access control (mac) and physical layer(phy) specications, 1999.

[sC02] IEEE Computer Society LAN MAN standards Committee. Ieee. recommended practice for multi-vendor access point interoperability via an inter-access point protocol across distribution systems supporting ieee 802.11 operation, Jan 2002. IEEE Draft 802.1f/D3.

[SP02] Yanghee Choi Sangheon Pack. Fast inter-ap handoff using predictive authentication scheme in a public wireless lan. In *IEEE Networks 2002 (Joint ICN 2002 and ICWLHN 2002)*, Atlanta, Georgia, August 2002.

[VK03] Hctor Velayos and Gunnar Karlsson. Techniques to reduce ieee 802.11b mac layer handover time. Technical Report TRITA-IMIT-LCN R 03:02, ISSN 1651-7717, ISRN KTH/IMIT/LCN/R-03/02–SE, KunglTekniska Hogskolen, Stockholm, Sweden, April 2003.