Design and Evaluation of a Platform for Mobile Packet Telephony

Sandesh Goel†, Partho Mishra†, Huzur Saran*, Cormac Sreenan†

† AT&T Labs - Research, Florham Park, NJ 07932

* Department of Computer Sc. and Eng., Indian Institute of Technology - Delhi

1 Introduction

Advances in packet switching technology have made it feasible to support integrated transport of voice, video and data on a common packet switched network infrastructure. This trend has stimulated research in exploring how integrated transport may be supported over *wireless* networks as well, to provide mobile users access to the same set of services that are available in their desktop environment. Due to spectrum limitations, specially for wide area wireless coverage, it may be practical to initially support only a limited form of integration, for example to enable the transmission of voice and best-effort data on a shared physical channel. However, even this capability makes it possible to design interesting new services. Consider for example, a mobile user who wishes to retrieve personalized directions or traffic updates while driving down a freeway. If the wireless channel supported integrated transport, it becomes feasible to design a service interface in which the user interacts with the appropriate network server using an interactive voice driven interface to specify the information required. This information might then be presented to the user in a variety of styles - as audio, text, images etc. Another example is a repairman dispatched to work at a remote location. Integrated transport might provide him the option of consulting with another repairman at headquarters, while perusing product manuals and transmitting realtime sensor feedback describing the product's current state.

Services that result from voice-data integration may be particularly valuable for cellular/PCS telephony users who constitute the fastest growing segment of the wireless industry. A limited form of wireless integrated access is already possible with so-called *smart phones* like the Nokia communicator [26] and the AT&T PocketNet phone [25] in which the smart phone can function either as a phone or as a (IP) data terminal offering access to e-mail and personalized information on stock quotes, weather reports, etc. These devices use separate wireless access protocols when functioning as a phone or as a data terminal, prohibiting the simultaneous transport of both voice and data. Voice-data integration is underway in the Internet as well with the development of applications centered around Internet telephony, for example click-to-dial and simultaneous voice-data transmission over dial-up (PPP) links.

There are several interesting problems that need to be solved to make it possible to support integrated voice-data access over wireless links [20]. First, it is necessary to design wireless channel scheduling mechanisms that can efficiently satisfy the service quality requirements of both voice and data. Second, network protocols need to be extended to support *handoffs* as mobile terminals change their point of connectivity to the backbone network. Third, forward error correction/packet retransmission mechanisms used over the wireless link need to be be tuned to match the varied loss resilience levels of voice and data. Fourth, integrated voice-data mobile terminals need to be developed that can act as both a phone and information-retrieval device. Finally, there are several interesting issues in providing enhanced voice-data services for mobile-users, for example how to support voice-driven information retrieval and adapt services and data delivery to match the limitations of the terminal and wireless link.

In this paper, we describe an experimental testbed that we have designed to support integrated wireless voice-data access. It consists of off-the-shelf network adapters, PCs as basestations, and handheld mobile terminals. This approach of using commodity hardware allowed us to assemble the testbed relatively quickly, providing us with a platform for prototyping algorithms and applications, and examining performance in a realistic but controlled environment. It also provides a clean upgrade path as new technology becomes available. Our goals were to investigate the network mechanisms required to support integrated voice/data transport to mobile terminals, and explore the design of interesting services/applications for this network environment. The initial focus has been on the design and implementation of a Medium-Access-Control (MAC) and handoff protocols that are suitable for integrated voice/data transport. The MAC protocol provides a simple token-based mechanism to support bandwidth reservation - an essential element for supporting time-sensitive voice traffic. The handoff protocols allow a wireless terminal to move between geographical areas while maintaining connectivity to the backbone network, and transparently rerouting data to/from its communicating peers. We present experimental results that analyze these protocols in terms of throughput and latency performance. We also measure the effect of handoff interruption on packet audio applications, and specifically its impact on playback buffer operation. We compare the trade off between tolerating a small amount of packet loss during a handoff to the additional delay incurred by buffering packets to avoid losses.

The paper is organized as follows. Section 2 describes the testbed components. Section 3 presents details of our wireless MAC protocol. Section 4 describes our algorithms for doing handoff and data rerouting. Section 5 presents our experimental results using the testbed to provide packet audio communications to a mobile handheld computer. Section 6 reviews the related work. We conclude the paper in Section 7.

2 Mobile Packet Telephony testbed

In creating the testbed, we have assumed a network architecture in which wireless links are used for last-hop access only and *basestations* act as gateways to an (integrated services) packet switched network, i.e. all communication to/from a mobile terminal is funneled through a basestation. We assume that the link layer protocol is ATM, i.e. each mobile terminal is an ATM endpoint and basestations are interconnected using ATM switches. Basestations are logically identical to ATM switches, i.e. they execute a NNI signaling protocol with their neighboring ATM switches, and switch cells (or packets) from one network interface to another. However, transmission on the wireless link uses variable size packets that may be larger than a single ATM cell for more efficient utilization of the wireless link capacity. We assume that connectivity to the Internet and to the PSTN is provided through gateways.

Figure 1 illustrates the current structure of the testbed. A mobile terminal is a PDA / subnotebook / laptop equipped with a wireless interface card. Basestations are PCs equipped with one or more wireless interface cards for communicating with mobile terminals, and with ATM network interface cards for communicating with ATM switches. We use two types of ATM switches - ATML switches that are controlled from an external PC that runs custom signaling/control software that we have designed, and FORE switches that run their own signaling/control software. A Powerhub IP router connects the testbed to external (IP) networks. The testbed also includes several *static* terminals, i.e. regular PCs with ATM network interface adapters that connect to one of the ATM switches. All of the functional entities in the testbed have been realized using off the shelf hardware components, while implementing custom algorithms in software. Moreover, we have tried to use open platforms so that we can piggy-back off the existing software base of public-domain applications, signaling protocols and device drivers.

The wireless link used for communication between a mobile terminal and a basestation is realized using Lucent's WaveLAN wireless LAN adapters. WaveLAN is a popular Wireless LAN adapter that operates in the Industrial, Scientific and Medical (ISM) frequency bands (915 MHz and 2.4 GHz) providing a shared bandwidth of up to 1.2 Mb/s in the coverage area[1] The WaveLAN network interface cards and associated operating system device drivers use a single FIFO queue per host, and implement a Ethernet-style random access MAC protocol to arbitrate access to the shared wireless channel among multiple hosts. A common FIFO queue is clearly undesirable when multiplexing traffic from applications with very different QoS requirements. Random-access MAC protocols can allow a single transmitter that seizes access to a channel to lock out other transmitters for several hundred milliseconds [5]. This is clearly unacceptable for packet telephony. To avoid these problems, we have modified the WaveLAN device driver to implement per-connection queueing and to use a basestation controlled token-passing MAC protocol that supports two distinct classes of service: fair-best-effort and guaranteed-bandwidth [13]. The details of this protocol are described in Section 3.

Basestations are PCs equipped with both wireless and ATM network interface cards. The cell/packet switching functionality is implemented in software (in the Linux kernel). The switching tables in the kernel are manipulated by a signaling module running in user space through a pseudo-device driver. The control protocol used across this interface is the General Switch Management Protocol (GSMP) [24]. GSMP was originally designed to allow external switch controllers the ability to modify the switching tables of a hardware ATM switch. It allows a controller to establish and release connections across the switch; add and delete leaves on a point-to-multipoint connection; manage switch ports; request configuration information; and request statistics. The choice of GSMP allows us to run the same signaling code on a PC and use it to control a hardware ATM switch, as we do with the ATML Virata switch in our testbed.

Each basestation handles communication to/from mobile terminals in a particular geographical area. As the mobile terminal changes its geographical location, it may be "handed-off" from one basestation to another. Since mobile terminals use ATM virtual circuits for communication (at least up-to the nearest IP router), a handoff requires rerouting of these virtual circuits. We have augmented the tradi-

[1] The use of WaveLAN as a wireless link restricts us to primarily indoor coverage.

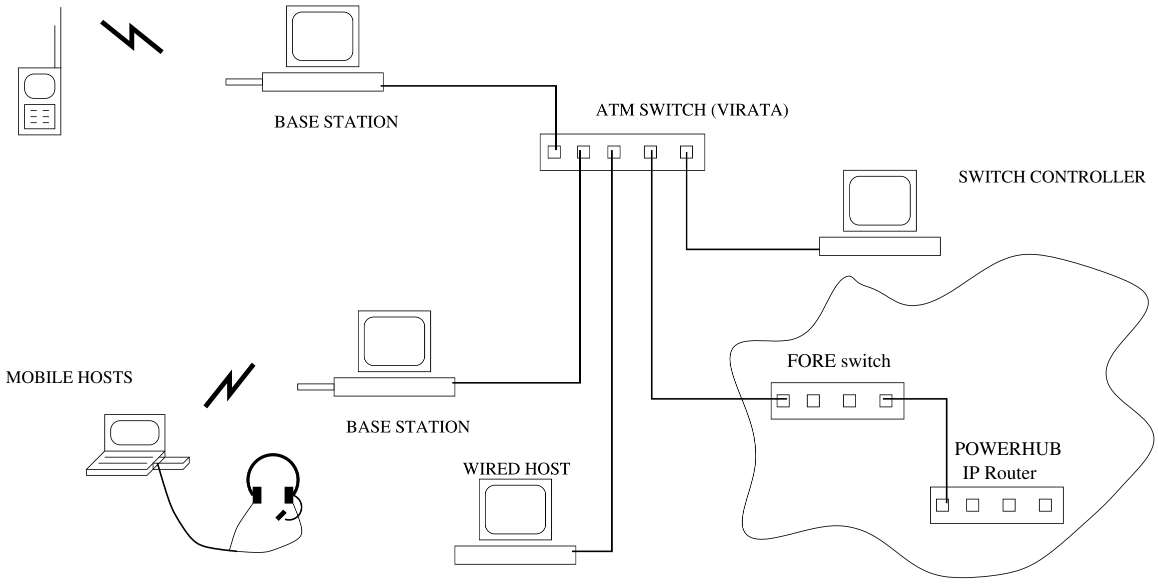

Figure 1. Testbed Configuration

tional ATM UNI/NNI signaling protocols running on mobile terminals and switches to support this rerouting capability - the mechanisms we have implemented allow a very rich set of rerouting policies to be instantiated. The details of this protocol are described in Section 4.

An integrated mobile terminal should ideally be able to function as a phone or a Walkman or a PDA by providing users the ability to access applications such as telephony, networked music play back, email, calendars, address books etc. It should be a small, lightweight device with (wireless) network connectivity. The device would need to support audio processing and a microphone/speakers for audio I/O. It might also need to support one or more user-interfaces such as a keypad, a touch sensitive display or voice-driven I/O. Smart phones such as the Nokia Communicator as well as PDA's such as the 3Com Palm Pilot provide some but not all of these capabilities. We have chosen instead to realize mobile terminals using laptops/palmtops along with the necessary I/O and networking peripherals. One of our mobile terminals is a IBM PC110 palmtop which measures 158x113x33mm and weighs only slightly more than a pound. This device has a 486SX CPU, 20 Mbyte RAM, 52 Mbytes of internal flash memory, a color LCD display (640x480 in a 4.7" screen) and 2 PCMCIA card slots.

All our terminals run the Linux operating system. Linux was chosen because of several reasons. First it allowed us to run, unmodified, applications such as the MBONE audio tool `vat`, and a locally-developed packet telephony application. Second, we already had a working implementation of our custom WaveLAN device driver for Linux. Third, source code availability for both the Linux kernel and various device drivers allows us to perform accurate timing measurements and meet the stringent delay and jitter requirements of packet telephony. Unfortunately, the PC110 has a fairly low quality half-duplex sound card that is unsuitable for packet telephony. Instead, we use a PCMCIA sound card from Communication, Automation & Control Inc for which we have developed a Linux device driver[2]. Figure 2 shows the PC110 equipped with a WaveLAN PC-card and the sound card. Clearly this is somewhat clumsier than would be desirable in a consumer product, but it satisfies our purpose of demonstrating the technology potential and quantifying system performance.

Applications running on a mobile terminal can use either a native ATM or a TCP/IP protocol stack for communicating with applications running on other machines. The native-mode ATM stack is an enhanced version of the IDLINet stack [10, 3] developed by S. Keshav and H. Saran. A client application that wishes to initiate communication with a server application, needs to first send a message to the signaling entity (via UNIX IPC). The signaling entity then initiates the connection setup process by sending out a UNI SETUP message. When the signaling entity at the remote end receives the request it informs the remote server application. If the remote application accepts the call, a virtual circuit is set up between the client and the server. At the time of connection setup, the client describes its service quality requirements to the signaling entity, and this is made available to both local and remote protocol stacks, so

[2]Thanks to Jont Allen of AT&T Research for providing us with an initial version of a device driver for this device and a lot of subsequent help in extending its capabilities.

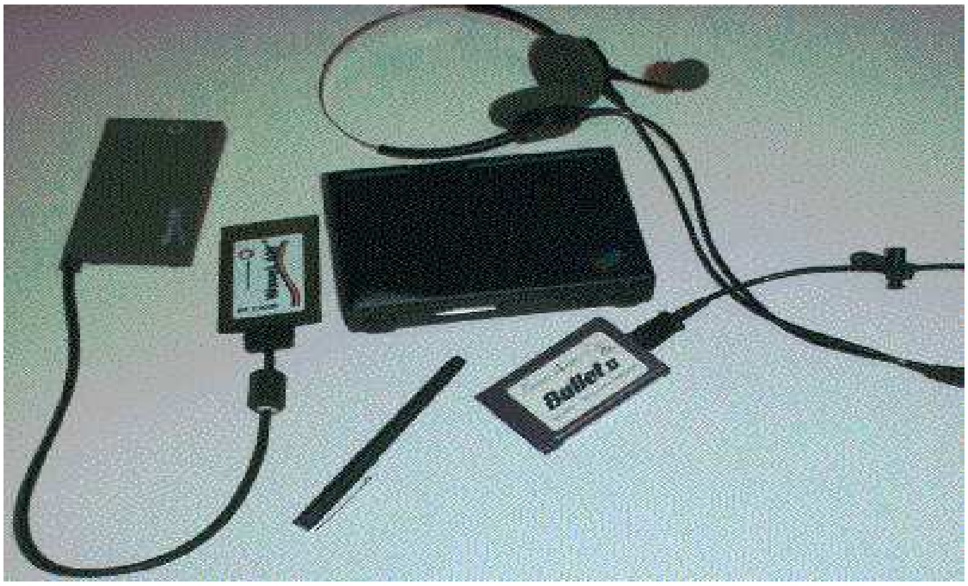

Figure 2. PC-110 with peripherals

that the appropriate control actions can be taken at the end systems as well as by intermediate network elements.

IP-based applications are supported over the ATM network segment, using a Classical IP over ATM implementation [11] in which the entire ATM cloud appears as a single IP subnet to the Powerhub router. We have begun to explore mechanisms to provide QoS-support for IP applications through a mix of pre-configuration and run-time traffic estimation. This allows us to avoid the need to modify existing applications and IP protocol stack implementations.

3 Medium Access Control

The WaveLAN physical layer is implemented using a direct sequence spread spectrum in combination with DQPSK modulation. The raw physical layer data rate is 2 Mb/s. The adapter implements a CSMA/CA medium access scheme. A transmitter that needs to send data first checks to see if the channel is busy. If it not busy, transmission can start immediately. If it is busy, then the transmitter waits until the channel becomes idle and then computes a back-off interval before trying to transmit again. Repeated failures lead to the usual exponential backoff algorithm. Due to the inherent non-determinism, this class of random access protocols are not suitable for telephony-like applications that require guaranteed bandwidth and/or tight control over delays. Many *scheduled access* protocols have been proposed for use in shared-media Local Area Networks that attempt to address some of these problems for example the "point coordination function" of IEEE 802.11. These protocols are typically designed to support multiple classes of traffic with differing service quality requirements.

We chose instead to design a simple scheduled access MAC protocol that supported only two classes: guaranteed bandwidth (voice) and best effort (data). This MAC uses a frame based scheduling policy [9] to control access to the broadcast wireless channel. A frame is a fixed interval of time and is comprised of several slots. Individual connections are provided bandwidth guarantees by allocating them a certain number of slots in each frame. These reservations are setup using the signaling protocols at virtual circuit setup. Any unreserved or unused slots in a frame are allowed to be used for best effort traffic. The MAC protocol operates within a cell, i.e. the geographical area controlled by each basestation. We assume that basestations are spatially separated to minimize interference between independent transmissions in different cells.

We have modified the WaveLAN device driver to implement this scheduled access MAC. In our implementation, a basestation computes the transmission schedule for each frame and based on this transmits a token to a mobile granting it the right to transmit for a certain number of slots. The mobile is expected to hand the token back to the base station after completing the specified transmission. A mobile terminal or a basestation cannot send a packet unless it has a token - this ensures that only one transmitter is active at any time thereby disabling the native CSMA/CA protocol implemented in the WaveLAN adapter. Tokens are piggy-backed onto data transmissions whenever possible for greater efficiency. To facilitate this, the frame computation algorithm attempts to schedule a reception slot from a mobile just after a transmission slot to that mobile. If all of the mobile terminals had tightly synchronized clocks, it is possible to reduce the overhead of explicit token-passing by having the basestation compute slot allocations and broadcast these at the beginning of each frame. This is not feasible with our current hardware.

Ideally, the slot allocation within a frame for best effort traffic should be fair across all active connections. We use the Start Time fair queueing algorithm to implement fair sharing. This requires the basestation to have knowledge of the amount of pending (best effort) data for each connection at each mobile terminal. This is achieved by piggy-backing this information onto regular data/voice packets (in the link layer protocol) whenever a mobile terminal sends packets to the basestation. Signaling traffic is also carried on the best effort slots but is given a higher weight and so normally gets priority over other best effort traffic. If a reserved slot at a mobile terminal is going unused, it is used to transmit best effort data.

When a mobile terminal moves into the coverage area of a new basestation it needs to discover the identity of the new basestation and register with it. A basestation identifies itself by periodically sending out a beacon. In response, a new mobile sends in a registration message. The MAC supports this by allocating two best effort slots for the beacon/beacon-response pair. When more than one mobile unit enters a coverage area simultaneously and try to respond to the beacon, the collision avoidance mechanisms of the WaveLAN hardware kicks-in and ensures that only one mobile terminal transmits during that slots. Those mobile terminals that failed to send a beacon-response on their first attempt defer their transmission until they get the next beacon; thus only one mobile terminal can successfully register per-frame.

The link layer protocol supports variable sized packet transmission.[3] We leave the decision on packet sizes to individual connections but determine the slot allocation based on a normative packet size for each connection which is made available via the signaling protocol. As such our guarantees hold good only if the connection uses packets of this normative size. Although it is potentially possible to police connections to ensure these packet sizes, we have not yet implemented this option. The choice of optimal packet size for a particular connection depends on application-specific requirements. Large packet sizes lead to more efficient use of the channel bandwidth, since the ratio of the data payload to protocol headers is higher. On the other hand, for wireless telephony applications, the use of large packet sizes may increase the end-to-end delay due to the relatively large time interval spent in assembling a packet. The encapsulation protocol used by the link layer protocol is described in [13].

[3]The WaveLAN hardware allows packet sizes to vary from 64 bytes to 1536 bytes.

4 Handoff policies

As a mobile terminal roams geographically, it may move from the vicinity of one basestation to another. Supporting such mobile terminal roaming requires network protocols to track the location of a mobile terminal and reroute existing virtual circuits to/from the mobile terminal. There have been several proposals for how data rerouting might be accomplished [1, 4, 8, 14, 18, 19]. For example, rerouting may be accomplished by extending a connection from one basestation to another, partially rebuilding a connection and so on. Most of the proposed algorithms make particular optimization choices based on specific assumptions about network parameters and topologies. There has also been a lot of activity in exploring how the ATM UNI and NNI signaling interfaces need to be augmented to support the capability to setup a call to mobile terminals, trigger VC rerouting etc. However, there has only been a limited amount of prototyping or experimentation with any of the proposed mechanisms making it difficult to quantify what kind of performance might be achievable with mobility-enhanced ATM signaling and how it might impact the performance seen by applications, specially telephony.

The UNI (and NNI) signaling protocols implemented in our testbed support terminal mobility using mechanisms described in [15] and [6]. The sequence of events leading up to and following a handoff are as follows. The MAC layer at the basestation periodically transmits beacons. At a mobile terminal, the MAC layer measures the signal strength for beacons received from each basestation. When the signal strength from the current basestation is much less than the signal strength from another basestation, the MAC layer decides to trigger a *handoff*. This causes the mobile to register with the new basestation. Subsequently, the new basestation generates a REBUILD request which triggers rerouting of active connection(s) in the following manner - see Figure 3.

The REBUILD request in initially forwarded to the old basestation. From here, a REBUILD_RELEASE message is generated and propagated along the original path of a connection (like a normal RELEASE message) with each switch on the path of the connection making a determination about whether it should act as a *cross-over-switch*. If not, then a switch processes the message like a normal RELEASE message. If yes, then the cross over switch attempts to setup a VC segment towards the new basestation. Modifying how a cross-over switch is identified, allows the above protocol to be used to implement various rerouting policies such as Extend, Total Rebuild, Partial Rebuild to a Fixed COS etc.

Our implementation incorporates several optimizations to improve performance in the wireless environment and ensure correct operation even if a mobile terminal were to

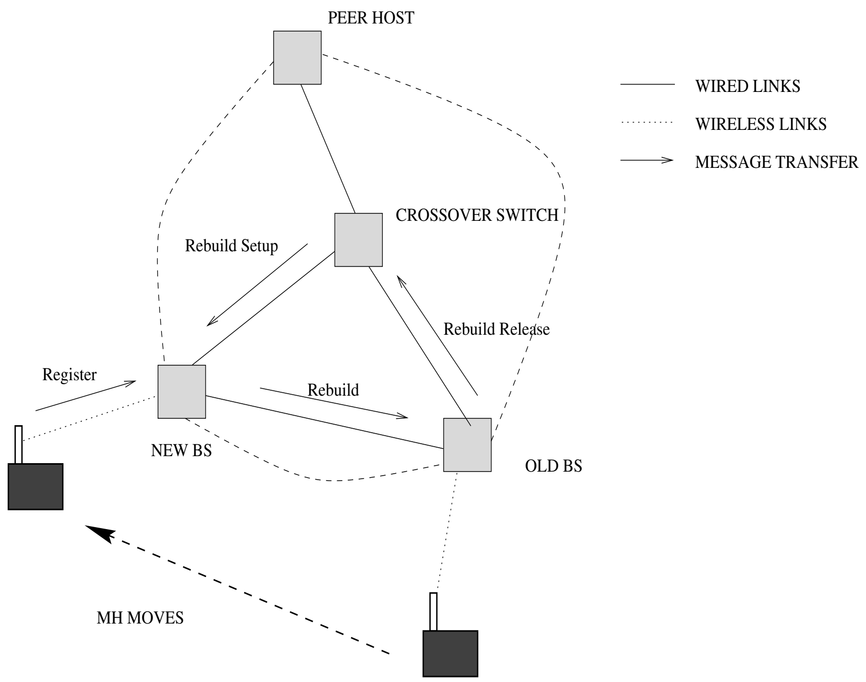

Figure 3. Sequence of steps for VC rerouting

undergo a series of rapid handoffs. For example, the initial registration message carries information about all the virtual circuits terminating at the mobile that need to be rerouted (instead of a separate message for each VC as would be the case with standard ATM signaling). This helps in reducing the number of messages that need to be sent over the wireless link thereby reducing handoff latencies. Also, unlike standard point-to-point ATM signaling, messages over the link between the mobile terminal and the base station do not use SSCOP for ensuring reliable delivery of signaling messages. Instead the signaling protocol is designed to explicitly deal with the possibility of lost messages. The use of SSCOP would require the peer SSCOP entities at the mobile terminal and the new base station to reinitialize their state, following every handoff. This requires an additional *handshake* over the slow wireless link, thereby unnecessarily increasing the handoff latency. Using SSCOP also implies that the signaling layer is unable to immediately detect the successful transmission of a message. This knowledge can be necessary for correct operation in situations where a mobile terminal moves subsequent to transmitting a signaling message [6].

5 Performance Results

We have run a variety of experiments to analyze the performance of the various components of our testbed, including the WaveLAN physical layer, the MAC protocol, the handoff policies etc. In this paper, we present results that attempt to quantify the feasibility of supporting mobile wireless packet telephony. Most of the measurements reported in this paper are with the basestations and terminals located in the same room (approx. 60 feet x 20 feet).

5.1 Performance of MAC protocol

The performance of the scheduled access MAC is determined by both the characteristics of the hardware platform and various implementation artifacts, as well as the parameters chosen for the scheduling slot allocation policy. To isolate these two effects, we have conducted two sets of experiments. First, we quantified the packet transmit/receive latencies and the maximum achievable throughput over a point-to-point WaveLAN link with the scheduling access MAC disabled and with only a single transmitter active at a time[4]. Subsequently, we measure the performance with the scheduled access MAC enabled and with multiple active transmitters. The key performance metrics are the aggregate utilization of the link and the throughput/delay seen by applications.

Table 1 enumerates the results with the scheduled access MAC disabled. The latency measurements are made using an "echo-mode" in the WaveLAN device driver. The mobile terminal initially transmits a packet towards the bases-

[4] i.e. a transmitter follows the CSMA/CA protocol and can transmit a packet instantly, if it senses that the channel is idle.

Packet size (bytes)	Round trip time (RTT) (ms)	Throughput (Mb/s)
98	4.3	0.2
512	11.7	0.6
1482	27.1	1.2

Table 1. WaveLAN link measurements

tation. The basestation immediately reflects the packet back to the terminal, which in turn reflects it, and so on for a large number of iterations. Table 1 enumerates the average *round trip times (rtt)* for various values of the packet size. The minimal round trip latency[5] is around 4.3 ms. On the transmit side, the transmission latency has two principal components: the time taken to transfer a packet from the host to the adapter and the time taken by the adapter to access the channel and transmit the packet over the air. The same is true on the receive side. Measurements indicate that the host to adapter transmit/receive latency is the dominant component of the packet transfer time - for our testbed these values are approximately 2 ms. This arises primarily because the WaveLAN PCMCIA network adapters have a relatively slow host-interface.

The maximum achievable throughput over the link is measured using a block-mode transfer in which a mobile terminal tries to send data as quickly as it can, and without any competing traffic. These values are also enumerated in Table 1. The maximum throughput that is attainable (with large packets) is around 1.2 Mb/s. Since the raw physical data rate that a WaveLAN radio is capable of is 2 Mb/s and the physical layer and MAC headers/trailers add only about 3% overhead for large packets, this throughput limitation also arises primarily from the (slow) speed host interface.

Our next set of experiments attempt to quantify the packet transmit/receive latencies and the maximum achievable throughput over a point-to-point WaveLAN link when using the scheduled access token-passing MAC protocol. Since the token passing MAC uses a frame based scheduling policy, the delays and throughput seen by a connection depend on whether it is a best effort or a guaranteed service connection. For best effort connections, the delay and throughput depend on the volume of traffic contending for the link. For guaranteed service connections, the maximum delay seen by each packet is determined by how often the connection has to wait between successive packet transmissions. This depends on the slot duration, the rate of the connection and may also depend on how multiple slots available to a connection are assigned within a frame. The values of the slot duration and the frame duration also determine the granularity with which bandwidth can be allocated.

As a consequence of implementing the MAC protocol

[5]The WaveLAN hardware enforces a minimum packet size of 64 bytes

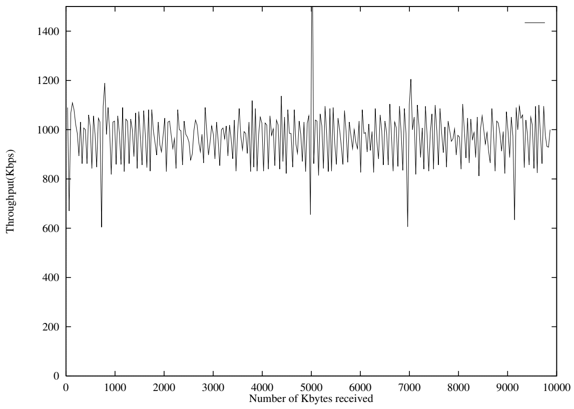

Figure 4. Throughput of single best effort VC

in the WaveLAN device driver running on the host, we are constrained to use relatively large slot durations, e.g. of the order of 5 ms, since this is approximately the empirically measured upper bound on the time taken to grant a token and get it back - see Table 1. For a slot duration of 5 ms and a frame duration of 100 ms, the bandwidth allocation granularity is approximately 64 kb/s but the maximum delay seen by a packet for a 64 kb/s connection may be as much as 100 ms. Increasing the frame duration allows bandwidth to be allocated in smaller quanta but increases the maximum delay, and vice versa. Since, the target end-to-end round trip delay for telephony is about 300 ms, a delay of 100 ms over a wireless link appears to be excessive. However, in practice we get much lower delays on the average as illustrated by the next set of experiments.

These experiments are designed to measure the delay and throughput seen by best-effort[6] and guaranteed service connections using the token-passing MAC for the following three test configurations:

1. single best effort connection from a mobile terminal to basestation

2. three best effort connections from three different mobile terminals to basestation

3. single best effort connection from a mobile terminal to basestation and single 64 kb/s guaranteed bandwidth connection from another mobile terminal to basestation

Figure 4 shows the throughput for a single best effort connection, as measured at the receiver. The average throughput is around 1.1 Mb/s. This is slightly

[6]For a best effort connection, there is an infinite amount of data to send.

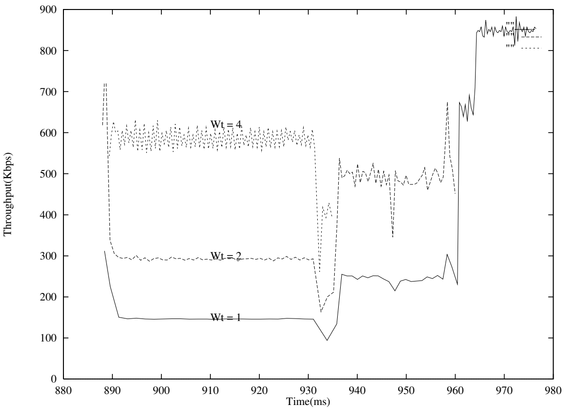

Figure 5. Throughput of three best effort VCs

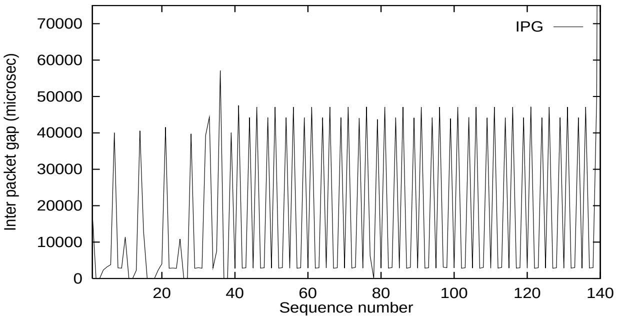

Figure 6. Inter packet delay for a 64 kb/s guaranteed bandwidth connection

lower than the throughput shown in Table 1 with the native WaveLAN MAC because of the overhead due to the token-passing. With three best effort connections (between three different terminals and the basestation) the aggregate throughput stays at about 1.1 Mb/s. In this experiment, the fair-queueing scheduling policy used to determine token-allocation[7] was configured with weights in the ratio 1:2:4 for each of the three VCs. Figure 5 illustrates that the throughput seen by each connection is indeed in proportion to the assigned weights. With a single best effort connection and a single (64 kb/s) guaranteed bandwidth connection, the throughput seen by the best effort connection drops to about 400 kb/s. The delay seen by the voice connection ranges from a few microseconds to several tens of milliseconds - see Figure 6.

5.2 Effect of Handoffs

Our final set of results quantifies the effect of a handoff when a mobile terminal moves from one basestation to another.

[7]We used Start-Time Fair Queueing.

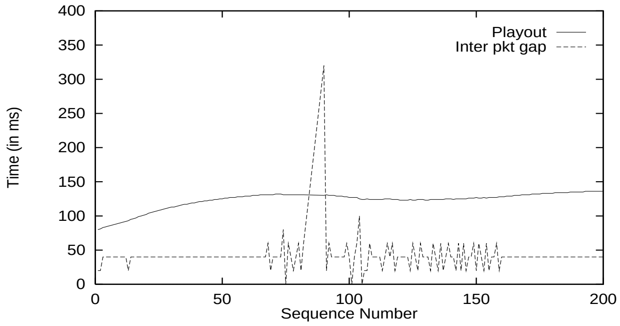

Figure 8. Inter packet delay and playout variation with *vat* during a handoff

Figure 7 illustrates the timeline for a handoff to complete when there is a single active connection. The rerouting policy used here is a FULL REBUILD, i.e. the entire connection is torn down and rebuilt from the peer terminal. For this experiment, the end to end path traverses only 3 links. The total handoff latency is about 32 ms[8]. Out of this about 12 ms is taken up in tearing down the old VC segment and setting up a new VC segment over the wired part of the network. The remaining time is spent in the initial registration handshake and final confirmation of the reroute.

As described earlier, it is possible to try to ensure lossless handoffs, by buffering in-transit data on VC segments being torn down. This reduces the loss rate at the penalty of increasing the delay variance. We attempted to verify whether these options affected audio quality, by running the *vat* telephony application between a mobile terminal and another host, as the mobile terminal underwent handoffs. This experiments was done using PCM encoding and with vat's silence suppression option turned off. vat incorporates an adaptive playback mechanism at a receiver to determines when an incoming packet is ready for presentation. The goal of this playback process is to compensate for delay jitter introduced by the network and restore the inter packet timing pattern as seen at the sender. Packets that arrive after their playout deadline are discarded. The playback time is varied over time using estimates of the mean and variance of the per-packet delay.

Figure 8 plots the inter-packet gap, for packets arriving at the receiver immediately prior to and following the handoff. It also computes the playout time as computed by vat for each arriving packet. Typically, only one or two packets are dropped during a handoff (vat was generating a 355 byte packet every 40 ms) Perceptually, the effect of these losses are barely noticeable with or without lossless hand-

[8]As a point of comparison, handoffs in the cellular network can take several hundred milliseconds.

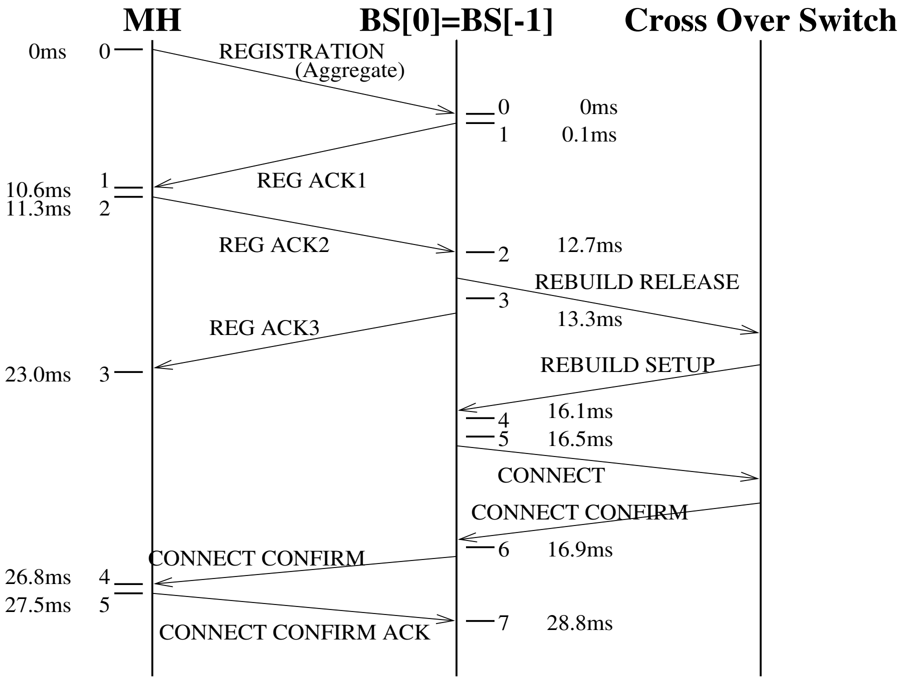

Figure 7. Call Reroute Timings

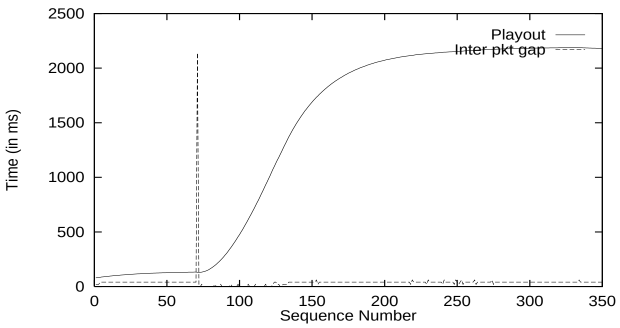

Figure 9. Inter packet delay and playout variation with *vat* forlarge handoff delays

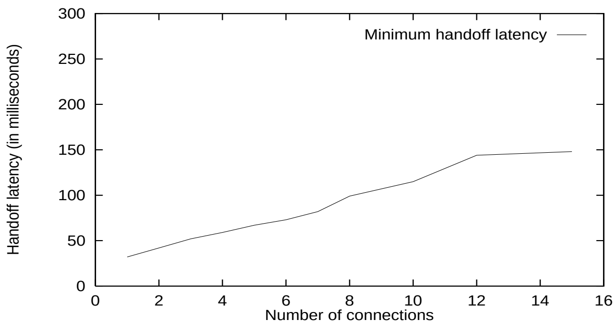

Figure 10. Handoff latency vs. number of connections requiring handoff

off. In the case, where in-transit packets are buffered rather than dropped, the playout value increases slightly due to the increase in the delay variance.

However, if the handoff latency is artificially increased by introducing an additional delay of about 2 seconds, then the use of the lossless handoff option causes the the playout delay computed by vat to increases abruptly following the handoff and to stay elevated, as shown in fig 9.

Another experiment we did was to study how the handoff latency changed as the number of connections terminating at a mobile terminal was increased - see Fig 10. As might be expected, the handoff latency increases with the number of connections requiring rerouting. The primary cause for this increase is because of the *contention* between the signaling messages for each connection, e.g CONNECT_CONFIRM and CONNECT_CONFIRM_ACK for transmission over the slow wireless link. Note that since there is no explicit reservation for signaling messages in the current implementation, contention is possible. In this situation, the CSMA mode of the WaveLAN card becomes active resulting in exponential back-offs after collisions and an increase in the transmission time over the air.

These results suggest that it may be useful to use aggregated signaling for the entire rerouting process, and not just for the initial handshake over the wireless link. Also, this suggests that it may be useful to allow some types of connections, e.g those used for telephony, to complete their handoffs faster. This may be achieved by allowing signaling messages associated with a connection to use the bandwidth reserved for the data transfer over that connection.

6 Related Work

Providing integrated services to mobile terminals has attracted considerable interest in the research community. Much of this work has involved the design of new QoS-oriented MAC algorithms [7, 23, 17]. Typically, these offer some form of scheduled MAC access, with slots assigned centrally by the base station. As an example, the NEC [23] algorithm provides a comprehensive set of QoS-controlled services. Constant bitrate connections are assigned slots periodically, while available bitrate connections are handled on a burst-by-burst basis. The IEEE 802.11 draft standard for wireless LANs also includes as an option a "point coordination function" which provides a centralized scheduler to support time-sensitive traffic. More recent work has explored adapting wireline fair scheduling algorithms to deal with wireless characteristics such as bursty errors and location-dependent capacity [12].

Efficient solutions for data rerouting and handoff have been explored for both ATM and IP networks. Several research groups [2, 22] have explored the design of wireless ATM LANs, with the goal of extending ATM's QoS properties to the wireless link. There has also been work on specifying appropriate ATM signaling extensions to handle mobility, and comparing techniques for rerouting such as VC extension, rebuild, partial rebuild and loop removal. In IP networks, the Mobile-IP standard [21] defines a procedure for rerouting packets to terminals which are mobile. Mobile-IP was not originally designed to support the delay requirements of telephony. There have been mechanisms proposed to extend mobile-IP to support voice traffic using a hierarchical handoff scheme to avoid noticeable disruption to voice traffic [5].

Several research groups have built systems to explore networking and terminal issues of mobility and wireless access. The SWAN [2] (Seamless Wireless ATM Network) system at Bell Labs used custom-built interface adapters as part of an indoor network extending ATM connectivity to laptops and custom-built personal terminals. The focus was the design and implementation of protocol software to handle wireless to wireline interworking and VC rerouting. Infopad [16] at UC Berkeley built a system to explore indoor delivery of multimedia data to mobile terminals. The focus was designing a fairly dumb terminal called a pad, which was low-power and effectively just performed I/O but no general purpose computing. Also explored was the supporting stationary-side infrastructure, including pad managers, filtering proxies, and servers for handling pen input.

In summary, research on wireless integrated services has addressed an array of interesting issues, primarily yielding new proposals for MAC and rerouting algorithms. Our work builds on these results, adopting a pragmatic view in designing a low-cost experimental testbed and focusing our research effort on supporting packet telephony as part of a wireless integrated service model.

7 Conclusions

In this paper, we have described an experimental testbed designed to support integrated wireless voice-data access. It consists of off-the-shelf network adapters, PCs as basestations, and handheld mobile terminals. A key component of this testbed is a token-based MAC protocol designed to support integrated voice-data transport and implemented using off-the-shelf WaveLAN cards. Another important testbed element are the handoff and VC rerouting protocols that have been optimized to provide predictable and fast handoff - essential features for packet telephony services.

Our experimental results have analyzed the MAC protocols in terms of throughput and latency performance. We have also measured the effect of handoff interruption on packet audio applications, and especially its impact on playback buffer operation. We compared the trade-off between tolerating a small amount of packet loss during a handoff to the additional delay incurred by buffering packets to avoid losses. For telephony our results suggest that the low latency of handoff makes it better to drop packets rather than suffer the increased delay by perturbing playback buffer operation.

The essential components and network protocols for our integrated services wireless testbed are fully operational. Future work will focus mainly on issues related to providing telephony service, terminal design and use, and music delivery. In addition to predictable and low-latency transport, providing telephony service to mobile terminals requires support for location management, call signaling, interworking with wireline packet (and PSTN) telephony, and implementing calling features. The nature of handheld terminals prompts work on speech-based user interfaces for data retrieval and handling telephone calls, as well as using proxies to adapt to variations in the quality of the wireless link and terminal capabilities. Finally, for networked music delivery, latency may be more relaxed, but losses are harder to tolerate due to high compression factors. We have begun to investigate these issues using an audio decoder running on the PC110 and accessing CD-quality audio from a web site.

References

[1] A. Acampora and M. Naghshineh. "An Architecture

and Methodology for Mobile-Executed Handoff in Cellular ATM networks." In IEEE Journal of Selected Area in Communications, vol. 12, no. 8, pp. 1365-1375, Oct. 1994.

[2] P. Agrawal, E. Hyden, P. Krzyzanowski, P.P. Mishra, M.B. Srivastava, J. A. Trotter "SWAN: A Mobile Multimedia Wireless Network" IEEE Personal Communications, pp. 18-33, April 1996.

[3] R.Ahuja, S.Keshav and H.Saran. "Design and Implementation of Native Mode ATM Stack." IEEE Transactions on Networking, August 1995.

[4] S. Biswas and A. Hopper. "A Connection Management Scheme for a Mobile Radio LAN." In Proceedings of the IEEE International Conference on Personal Wireless Communications, Bangalore, India, Aug. 1994.

[5] R. Caceres, V.N. Padmanabhan "Fast and Scalable Handoffs for Wireless Internetworks" Proc. of ACM MobiCom, Nov. 1996.

[6] C. Kalmaner, P. P. Mishra and M.B. Srivastava. "ATM Virtual Circuit Rerouting to support Mobile Host Roaming". ATM_FORUM contribution #97-0155, Feb 1997, San Diego, California.

[7] M.J. Karol, Z. Liu, K.Y. Eng "An efficient demand-assignment multiple access protocol for wireless packet (ATM) networks" ACM/Baltzer Wireless Networks, 1(3):267–279, 1995.

[8] K. Keeton, B. Mah, S. Seshan, R. Katz, and D. Ferrari. "Providing connection-oriented services to mobile hosts." In Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing, pages 83-102, Cambridge, Massachusetts, August 1993.

[9] C.R. Kalmanek, H. Kanakia, S. Keshav "Rate Controlled Servers for Very High-Speed Networks." GlobeCom, San Diego California, Dec 1990.

[10] S.Keshav and H.Saran. "Semantics and Implementation of a Native Mode ATM Protocol Stack." AT&T Bell Labs. Technical Report, 1994.

[11] M. Laubach. "Classical IP and ARP over ATM" NIC, Request For Comments 1577, 1994.

[12] S. Lu, V. Bharghavan, R. Srikant "Fair Scheduling in Wireless Packet Networks" Proc. of ACM SIGCOMM 1997

[13] K. Mahajan and P. S. Garg. "Mobile ATM: QoS over the Wireless Hop." Technical Report, Department of Computer Science and Engineering, IITD, 1997.

[14] P. P. Mishra and M. B. Srivastava. "Call Establishment and Rerouting in Mobile Computing Networks." AT&T Bell Laboratories Technical Memorandum 11384-940906-13TM, September 1994.

[15] Partho P. Mishra and Mani B. Srivastava. "Effect of Virtual Circuit Rerouting on Application Performance." Proceedings of IEEE International Conference on Distributed Computing Systems, Baltimore, Maryland, May 1996.

[16] S. Narayanaswamy, S. Seshan, E. Brewer, R. Brodersen, F. Burghardt, A. Burstein, Y-C. Chang, A. Fox, J. M. Gilbert, R. Han, R. H. Katz, A. C. Long, D. G. Messerschmitt, J. Rabaey "Application and network support for InfoPad" IEEE Personal Comms. Mar 1996

[17] K.S. Natarajan "A hybrid medium access control protocol for wireless LANs" Proc IEEE Int Conf on Selected Topics in Wireless Communications, pp 134-137, June 1992.

[18] B. Rajagopalan. "Mobility Management in Integrated Wireless-ATM Networks." In Proceedings of MobiCom `95, Berkeley, California, November 1995.

[19] S. Seshan et. al. "Experiences with Hand-Offs in the Daeduls System." ACM Baltzer Wireless Networking Journal, December 1995.

[20] Mani B. Srivastava and Partho P. Mishra. "On Quality of Service in Mobile Wireless Networks." Proc. of Network and Operating System Support for Digital Audio and Video (NOSSDAV), St Louis, Missouri, May 1996.

[21] C.E. Perkins, P. Bhagwat "A mobile networking system based on the Internet protocol" IEEE Personal Comms, pp 32–41, First quarter 1994.

[22] J. Porter and A. Hopper "An overview of the ORL Wireless ATM System" Proc. of IEEE ATM Workshop, Washington DC, Sept. 1995.

[23] D. Raychaudhuri "Wireless ATM networks: architecture, system design and prototyping" IEEE Personal Communications pp. 42-49, August 1996.

[24] Network Working Group. RFC on GSMP. Version 1.0.

[25] http://www.attws.com/nohost/data/pocketnet/

[26] http://www.nokia.com/com9000/