Real-Time Performance Guarantees over Wired/Wireless LANs

Prashant Pradhan Tzi-Cker Chiueh

Computer Science Department
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

(prashant,chiueh)@cs.sunysb.edu

1 Introduction

Distributed audio and video applications require end-to-end performance guarantees from the underlying networks for smooth delivery of real-time media streams. The end-to-end nature dictates that real-time support be built into both wide-area and local-area networks. In wide-area networks emerging reservation protocols such as RSVP [10] together with real-time packet scheduling mechanisms in the routers, promise performance guarantees on IP-based networks. The circuit-switching nature of public telephone networks also offer a viable alternative for performance guarantees over a large geographical area. However, real-time performance guarantees for popular LAN technologies such as Ethernet are conspicuously missing. Being a shared access medium, the bounds provided by Ethernet are probabilistic rather than deterministic, which is not sufficient for providing performance guarantees to such real-time applications as multimedia conferencing systems. The absence of such technology prohibits the realization of the vision of end-to-end desktop-to-desktop audio/video communication, because the network performance on the "last mile" to the desktops is not guaranteed.

REther [1] is a software-based token passing protocol that bridges this gap by providing performance guarantees for real-time applications on existing Ethernet hardware infrastructure. REther has been implemented and tested on both 10-Mbits/sec and 100-Mbits/sec Ethernets. A REther switch has also been built [8] to extend the performance guarantee service over a switched Ethernet environment consisting of both Ethernet and Fast Ethernet segments. Distributed multimedia applications that have been and are being built on top of REther include the Stony Brook Video Server [2], which provides digital video playback over Ethernet to a network of clients, a desktop video-conferencing system that stripes digital video over multiple analog modem links for higher aggregate throughputs, and a satellite video gateway that distributes digital satellite TV programs over a local area network in real time.

With the enormous growth of portable computers equipped with networking capabilities, it is important to develop real-time protocols for wireless LAN technologies, such that real-time audio/video communication services can be extended to these mobile terminals as well. This paper describes the implementation effort of supporting end-to-end performance guarantees over a *heterogeneous* local-area network comprising of both wired and wireless LAN segments. Most of the earlier works [3] [4] [5] [6] [9] on real-time performance guarantees over wireless LANs are based on simulation or analytical studies. One of the exceptions is the ATM wireless LAN testbed effort in NEC &C Lab [7]. However, none of these works focused on the design and implementation of real-time protocols for off-the-shelf wireless LAN technology without special hardware support. In addition, none of the previous works discussed how performance guarantees mechanisms for wired and wireless networks should be *integrated* to form an end-to-end solution. Fundamentally the goal of this work is to provide real-time performance guarantees to interactive multimedia applications over a LAN environment, which could consist of wired as well as wireless LAN segments. This work also addresses the performance guarantee issues associated with mobility or roaming.

The rest of the paper is organized as follows. In Section 2 we provide a brief overview of the original REther protocol for wired Ethernet. Then we describe in detail the wireless REther protocol in Section 3 with the design choices and the internal workings of the protocol. The measurement results for the wireless REther prototype are also reported to justify these design decisions. Section 4 discusses the support provided by wireless REther for ensuring uninterrupted flow of real-time streams in presence of host mobility across the wireless network, and the associated design choices therein. In Section 5 we present a distributed multimedia application, viz the Stony Brook Video Server (SBVS), which uses the wired and wireless REther as the enabling network technology. Section 7 concludes this paper with a summary of the main results and an outline of on-going work.

2 REther: A Real-Time Ethernet Protocol

An Ethernet segment running REther [1] can be in one of two modes. In the absence of real-time flows, the network is in the CSMA mode and the nodes contend for channel access using the CSMA/CD protocol. When the first real-time flow is initiated, the network transparently switches to the Token Passing mode. The node on which the first real-time flow is started initiates this change by requesting other nodes to switch mode. This node also creates and starts passing a *token*,which is circulated around the nodes with a fixed *cycle time*. The timeline is divided into cycles of length equal to this cycle time. Each real-time flow may reserve a fraction of this cycle time, which is the *token holding time* for that flow. When the token arrives at a network node, the node transmits as much data as dictated by the *token holding time* specified in the token, and passes the token to the next node.

A fixed fraction of each token cycle time is reserved for non-real-time packets to ensure smooth operation of existing TCP/IP-based network operations. After all the real-time flows have been serviced in a cycle the token is passed among *all* nodes on the network segment in a round-robin fashion for non-real-time data transfer, till the cycle ends. In the absence of non-real time data to send, a non-real-time node simply forwards the token to the next node in round robin order; thus “spending” spare cycle time. In this sense, the service discipline in REther is non-work conserving. At the end of each cycle, the token is again passed to the first real-time node. In the next cycle, the first node that gets to send non-real-time traffic is the one next to the last non-real-time node in the previous cycle.

An important design decision in REther is that the token contains the current state of the channel, i.e., the set of real-time and non-real-time nodes and their associated *token holding times*. As a result, admission control can be performed in a completely distributed manner. A node makes the admission control decision when it receives the token to send non-real time data. If the node has a pending admission request at this time, it checks whether enough time is left in the token cycle to admit this request and adds itself to the list of real-time nodes in the token.

3 Wireless REther

3.1 Overview

In the design of wireless REther, we have made the following assumptions:

- Each wireless LAN has a base station, which is connected to a wired Ethernet environment.
- All network nodes on the wireless LAN can communicate directly with the base station. They may or may not be able to communicate with each other.
- As in REther, the performance guarantee of interest is network bandwidth guarantee.

Providing bandwidth guarantees over a shared network medium is ensuring *deterministic channel access* delays. Fundamentally there are two ways to provide deterministic channel access to network connections on a shared medium. One is time division multiplexing (TDM) and the other is token passing. However, it is difficult to provide TDM in the absence of hardware support since the software timer resolution is typically too coarse to be useful. Moreover, increasing the kernel timer resolution is not acceptable because of the excessive interrupt processing overhead that it would introduce.

Our original thought was that wireless REther should be a simple extension of wired REther because they are both based on a broadcasting shared-medium architecture. Careful examination reveals that there are subtle differences that the different media access protocols in the two cases introduce. The wireless LAN that we use, AT&T's 915-MHz WaveLAN, uses CSMA/CA as the MAC protocol, where nodes in different *collision domains* can not sense each other's presence. Hence, token passing directly among nodes may not always be possible. One possibility is to involve the base station in token forwarding since all nodes are assumed to be reachable from the base station. The problem with this approach is that the token passing overhead is fairly expensive, i.e., one-hop token passing takes about 1 msec. Moreover, our initial implementation experience showed that the ISA-based WaveLAN cards can not handle fast back-to-back packet transmissions reliably. This problem is serious because the token follows closely normal network packets while being forwarded.

Due to these limitations, wireless REther chooses to centralize channel arbitration and admission control at the base station for each wireless subnet. That is, the base station decides whether and when a wireless network node is eligible to send data and notifies it with a token to indicate "go-ahead." Centralized channel control eliminates the need of fast token forwarding among nodes on the wireless network. Several advantages that result from this centralized design would be alluded to in the rest of the paper, including performance considerations, more efficient allocation of non-real time bandwidth and mobility support.

3.2 Channel Access Control

As in wired REther, the channel switches its state between CSMA/CA and Token Passing when the first real-time flow is admitted or when the last real-time flow leaves. Time is divided into token cycles and each node may reserve a fraction of this time to send its real-time data. Bandwidth reservation requests are sent to the base station, which makes the admission control decision, and adds the request into the real-time list if admitted. The base station automatically switches the network to different modes, depending on whether there are real-time connections. As in wired REther, a fraction of the cycle time is reserved for non-real-time data transfer.

The base station grants a node network access by sending it a notification token. Instead of sending a separate token back to the base station to indicate the completion of packet transmission, a wireless node prepends a special marker to the last frame it puts on the channel. Because the base station is always in the promiscuous mode to monitor the network activity, it can detect the end of packet transmission by the receipt of such marked frames. Once a marked frame is detected, the base station can send the notification token to the next wireless node in the token cycle. In this scheme, only one *physical* token passing is needed for each *logical* token passing among the wireless nodes. As in wired REther, the token is first circulated among real-time nodes and then among all nodes for non-real-time data transfer. However, in contrast to wired REther, the number of packets that a non-real-time node can send at each token arrival is dynamically determined by the base station, based on the recent history of the node's bandwidth demands. This is possible because of the centralized design of wireless REther which enables the base station to keep track of the history of access patterns from different nodes. Thus, the base station may choose several policies to decide the order in which nodes get to send non-real time data as well as the amount of non real time data they get to send per cycle.

When the network is in the CSMA/CA mode and a real-time flow is started, the bandwidth reservation request is sent to the base station. If the network is in the Token Passing mode, the request is deferred till the node gets permission to send non-real-time data. Once admitted, the node passively waits for the notification token from the base station to send real-time and non-real-time data.

The end of the transmission of real-time data from a real-time node is inferred by the base station by detecting the transmission of a marked packet from that node. The end of transmission of non-real-time data is determined by detecting a number of transmissions equal to the number granted by the base station to teh non-real time node or upon receiving an explicit NACK. If a non-real-time node doesn't have any data to send, it sends an explicit NACK

Real-Time Packet Length (bytes)	Interrupt Processing		NIC to kernel Space		Kernel Space to User Space	
	Send-side	Recv-side	Send-side	Recv-side	Send-side	Recv-side
100	38	40	73	86	25	21
192	38	40	109	126	27	24
500	38	40	232	270	38	33
1000	38	40	423	525	45	48

Table 1: *Breakdown of end-host processing time for different packet sizes. The row with packet size 192 corresponds to the token processing breakdown. All times are reported in* μ*sec*

to the base station so that it can give a go-ahead to another node. Since the base station itself can also be a real-time node on the wireless LAN, it can optimize away the token passing and packet transmission termination detection if it itself is a real-time node.

To handle loss of notification tokens, NACKs, or transmitted data, the base station sets a timer before sending every notification token. Upon timeout, the next notification would be sent. A wireless node that constantly causes time out on the base station on the notification token sent to it, will be assumed to be dead and removed from the list of active nodes on the network.

3.3 Performance Measurement and Analysis

3.3.1 Experimental Setup

The prototype implementation of wireless REther is built on Pentium-100s running FreeBSD 2.1.0 and using AT&T's 915MHz ISA WaveLAN cards. The Pentium-100 machines serve as static hosts on the wireless network and as the base station. The mobile hosts are 486DX-based Toshiba Satellite portables, also running FreeBSD 2.1.0 and using PCMCIA WaveLAN cards. The peak link bandwidth is 2Mbps. Both ISA and PCMCIA WaveLAN cards used programmed I/O to transfer data between the network interface and main memory.

3.3.2 Performance Measurements

The following measurements measure the breakup of end-host processing at the sender and receiver end. The measurements are performed on real-time flows that are set up between static hosts across the WaveLAN with differing amounts of bandwidth reservation. The token passing overhead is broken down into interrupt processing overhead, the copy overhead from the NIC to kernel mbufs and from kernel mbufs to user space buffers. Table 1 shows these measurements.

To put these delay measurements in perspective, we compare the numbers for token processing overhead with those for wired REther on a 100Mbps Ethernet. While comparing these numbers, it should be kept in mind that the 100Mbps Fast Ethernet driver uses DMA rather than PIO and the network interface card resides on a PCI bus instead of the ISA bus, as in WaveLAN. Likewise, while comparing transmission times, it is to be kept in mind that the peak link bandwidth for WaveLAN is 2Mbps. Table 2 shows this comparison.

For the sake of comparison, these numbers assume the same token size in both cases (192 bytes). Note, however, that because of the distributed design of wired REther, the token needs to carry all the information about the current reservation status of the network. In wireless REther, this information is not necessary since the base station keeps all such information locally, and hence the size of the token can be kept at the minimum size of an Ethernet frame, which is 64 bytes.

The connection setup latency over 1 hop (2 segments) was measured to be 14.48 msec for a static host as the destination and 23.72 msec for a portable. This time measures the latency of the first packet arrival at the receiver

Operation	Wired REther over 100 Mbps Ethernet	Wireless REther over 2Mbps WaveLAN
Receiver Interrupt Processing	7.42	40
Receiver memory copying time	3.75	126
Sender copy time + transmit time	52.02	1349 (1202+38+109)

Table 2: *Comparison of wired and wireless REther token processing overhead at the end hosts. Note that in case of wired REther, the sender DMA time is clubbed with the transmission and interrupt processing times since it is not possible to know when the DMA completed. In case of wireless REther, the number can be split into interrupt processing, bcopying time and transmission time. All times are reported in μsec*

% of Real-Time Reservation	Sender Interrupt Overhead (μsec)	Receiver Interrupt Overhead (μsec)	Interrupt Overhead as % of Cycle Time	Copy Overhead as % of Cycle Time
1.2%	900	860	2.64%	8.19%
6%	870	830	2.55%	7.83%
12%	830	790	2.43%	7.47%
24%	830	755	2.37%	7.12%
48%	712	600	1.968%	5.69%
96%	600	377	1.465%	3.56%

Table 3: *The performance overhead on the base station due to network packet interrupts in the promiscuous mode. The first column indicates the percentage of link bandwidth reserved for real-time connections. The second and third columns show the sending and receiving interrupt overheads in a token cycle. The fourth column shows the percentage of the overhead with respect to the token cycle time, in this case, 66 msec.*

after initiating a connection establishment request. We use these numbers in section 4 when we discuss roaming.

A few comments about these numbers are in order. First of all, the relatively large end-to-end token processing delay justifies the design choice of using centralized channel arbitration for the wireless network instead of using the base station as a forwarder of token passing among wireless nodes. Also, in applications such as media (voice/video) playback on mobile terminals over a guaranteed performance network, mobiles assume the role of passive receivers whereas the base station gateways the media streams from the media server (typically on a wired segment) to the mobiles. In this case, the per-token interrupt processing and token transmission overhead are completely eliminated from the host processing time since the base station doesn't need to pass the token *to itself*. This increases the effective transmission bandwidth available for real-time flows in a token cycle.

The final important piece needed to make the evaluation of the design complete is the CPU overhead at the base station due to the base station being in the promiscuous mode and performing channel arbitration in a centralized manner. Because the base station is in the promiscuous mode for the wireless subnet, it is possible that it is constantly being bombarded by network interrupts, and thus may suffer serious performance overhead. This is an important consideration given that the base station acts as the gateway between the wireless and wired segments and hence needs to perform interrupt and packet processing for tokens and packets arriving on both the segments. The following measurement shows that fortunately this is NOT the case. We make a bandwidth reservation of different amounts from the base station to one of the wireless node, and assume that all non-real-time nodes that share the remaining bandwidth have nothing to send. Therefore, during the non-real-time part of the token cycle, the base station is busy receiving NACKs and sending out the notification token until the cycle ends. Table 3 shows, within a token cycle time of 66 msec, how the sender and receiver interrupt and copy overheads on the base station vary with the percentage of link bandwidth reserved for the real-time connections. As the real-time reserved bandwidth percentage increases, fewer non-real-time nodes are able to participate in the NRT part of the token cycle. Accordingly, the number of NACKs that the base station has to process within a token cycle time is becoming smaller. Therefore, the total interrupt processing overhead is decreasing with the amount of reserved bandwidth percentage. Even with only 1.2% of the link bandwidth reserved for real-time traffic, the interrupt processing overhead is quite reasonable, viz 2.64%. This demonstrates that putting the base station in the snooping or promiscous mode to save one physical message transfer during every token passing is a good design decision. The reason why network interrupts don't seriously degrade the performance of the base station is that the relative low link bandwidth (2 Mbps) makes it impossible to send even small packets fast enough to saturate the base station.

3.3.3 Evaluation

With these measurements at hand, one can get an estimate of the channel utilization for the protocol when all non-real time nodes have data to send (there are no NACKs exchanged). This means that bandwidth demand is not the limiting factor in determining channel utilization efficiency. The utilization would then depend upon how *fragmented* the real time bandwidth demand is. Let x be the pure overhead of token passing per cycle that is not available for use as bandwidth. If there are N nodes in the network with M real-time reservations reserving a combined amount T_R worth of cycle time, then the overhead of passing real-time notification tokens becomes $1400 * M$ μsec (Note that we save an extra token passing from the real-time node back to the base station by keeping the base station in promiscuous mode; otherwise this time would be $2800 * M$). Assuming that every non-real time node sends an MTU worth of data whenever it gets access, the number of non-real time nodes that get to send data becomes K where

$$K = \lfloor \frac{(T - T_R - x) * B}{MTU} \rfloor$$

Thus, the non-real time notification token overhead becomes $1400 * K$ μsec (again using the fact that the base station can independently figure out completion of transmission by being in promiscuous mode). Given that all the other variables are known, the value of x is given by the solution to the equation :

$$x = 1400 * (K + M)$$

Hence, the expression for the channel utilization efficiency is given by :

$$1 - x/T$$

Another metric of importance for our protocol is the non-real time access latency for the network, defined as the delay suffered by a non-real time packet before it gets access to the network in non-real time mode. Recall that REther chooses to reserve a fixed fraction of the cycle time for non-real time data transfer in order that nodes intending to send non-real time data over the network do not starve for bandwidth. It is important, for instance, to avoid NFS timeouts for nodes on a LAN using a common file system. The fraction of bandwidth set aside for non-real time traffic would then become a design parameter affected by such considerations.

Normally, the *non-real-time access latency* for a network running REther would be dependent upon the cycle time, the amount of bandwidth reserved by real-time flows, the non-real-time access policy implemented by REther and by the number of nodes on the network. Assume the following: The link bandwidth is B, the token cycle time is T, there are N nodes on the network, kT (where $k < 1$) is the amount of cycle time reserved by real-time flows and the non-real-time acccess policy is to allow every node to send at most one non-real-time packet of size equal to the link's MTU in a round robin fashion. Also, for the moment, ignore the overhead due to token passing. Then the worst-case non-real-time access latency for a node is bounded by :

$$\frac{T * (N - 1)}{\lfloor \frac{(1-k)T * B}{MTU} \rfloor}$$

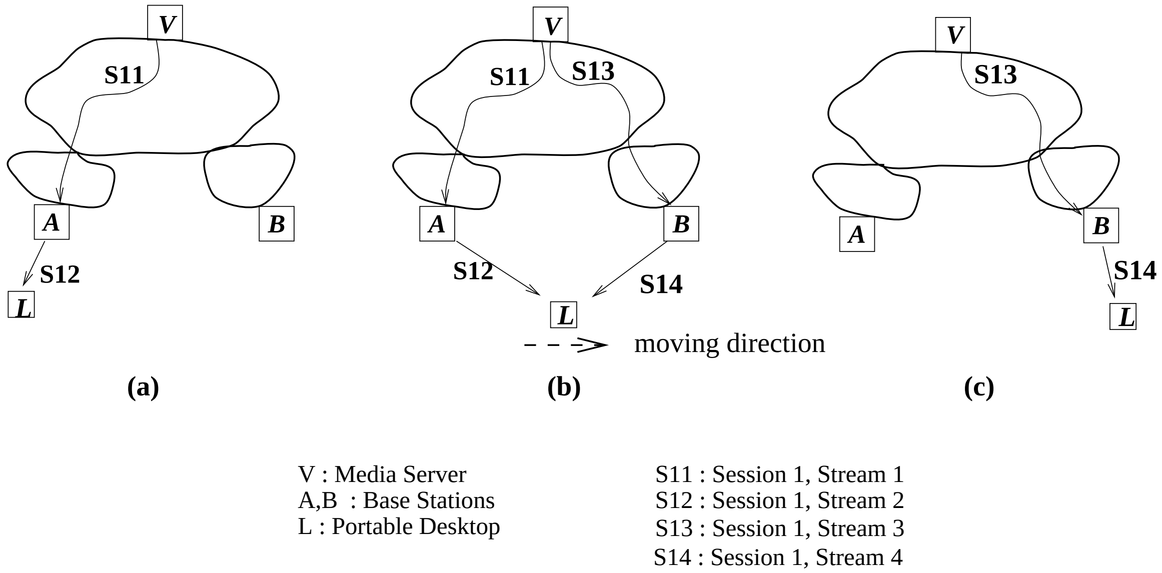

Figure 1: *Video playback to roaming terminals using Wireless REther. The number of real-time flows supporting a mobile real-time session changes from (a) 2 (S11 and S12) to (b) 4 (S11, S12, S13, and S14) to (c) 2 (S13 and S14).*

This number can be compared, for instance, to the NFS timeout to figure out an acceptable value for the design parameter k.

However, the worst-case non-real-time access latency for the base station is different because the order of visiting the nodes within a token cycle is completely in the control of the base station. This means that unlike the static node ordering in wired REther because each node fundamentally does not know other node's demands, in wireless REther, the base station can choose to give high priority to sensitive non-real-time control information like beacons (refer to section 4). Hence, if the base station chooses to prioritize a non-real time packet, its worst case network-access latency becomes

$$MAX(kT\ ,\ MTU/B)$$

4 Roaming

Because portable computers are inherently mobile, performance guarantee support during roaming is crucial. The real-time session abstraction for mobile wireless networks is different from that for wired networks. To support continuity of real-time packet delivery to a moving terminal, a *mobile real-time session* may consist of a variable number of real-time flows at different points of its lifetime. Consider the scenario in Figure 1. In the beginning, the mobile real-time session consists of two real-time flows, $S11$ using wired REther and $S12$ using wireless REther. Then as the mobile terminal, L, moves from the subnet of base station A to the subnet of base station B, L detects the movement of the terminal through a change in the perceived signal strength from the two base stations and requests B to establish a real-time flow, $S13$, to the video server, *using the same session ID*. After $S13$ is set up, B is going to receive the same real-time stream as A. When L finally moves to B's subnet by establishing a real-time flow, $S14$, the flows, $S11$ and $S12$ to A will be terminated. The mechanism of allocating and deallocating real-time flows to support a mobile real-time session is implemented completely in the network layer and is transparent to the applications. For example, in the example shown in Figure 1, the video server simply retrieves video data from the disk, and transfers them to the roaming terminal using a mobile real-time session. The network subsystem sends these packets through one or multiple real-time flows, depending on the location of the mobile terminal. In the most general case, real-time flows to more than two base stations may need to be set up speculatively during the transition period if the direction of movement of the mobile terminal can not be determined definitely.

In order for mobile terminals to detect new subnets while roaming, each base station periodically broadcasts a *beacon* on its wireless subnet. A beacon from a given base station will be heard by the nodes in its own wireless subnet (cell) as well as by nodes lying in the overlap region of its cell and cells of other base stations. Mobile terminals in the overlap region would pick up the beacons from multiple base stations. Upon receipt of a beacon, each mobile node reads off the signal level, silence level and signal quality parameters from its interface registers. These are used to derive an *SNR-index* that indicates the signal quality associated with a given received packet. Each mobile node maintains a table of the beacon signal strengths and the MAC addresses of the senders of the beacons. When the signal strength from a new base station is consistently higher than that from the current base station according to a history of measurements, the node initiates a switch to the new base station's cell.

For wireless REther, cell switching entails the following two additional requirements. First, all real time flows which the node was participating in should be established in the new cell such that the state in which the flow starts in the new cell be exactly identical to the state in which it left off in the previous cell. Moreover, this should be done transparently to the application on the node as well as to the application at the source of the flow. In other words, the abstraction provided to the application by the network subsystem should be a real-time performance guarantee session that could have different physical realizations at different points in time due to roaming. Secondly, the mobile should now participate in the token-passing of the new cell and should be excluded from the token-passing schedule of the earlier cell. The current implementation of wireless REther encapsulates this functionality purely in the network subsystem and none of the cell switching mechanism is visible above the network driver level.

4.1 Implementation

As referred to above, each mobile tracks the SNR-index of the beacons received from the base stations that it can hear from, and based upon a history parameter, decides to switch to the cell whose base station gives the highest signal quality consistently. The first phase in cell switching involves switching all the real-time flows being received by the current base station. The mobile node sends the new base station a request to make bandwidth reservations from the new base station to all the sources of the currently active real-time flows. The new base station does the appropriate kernel buffer allocations and reservations on its cell for this new connection and forwards the duplicate connection establishment request upstream. Every gateway along the way repeats this process until the duplicate connection request reaches the source of the flow. The source identifies the existing state corresponding to this request based upon the session identifier (port number in standard terminology) in the request. The source then goes through a QoS re-negotiation phase on its own subnet by requesting for an increase in the reservation corresponding to this flow. Once this is successfully done, the flow is ready to be replicated at the network layer.

The duplicate network bandwidth reservation is transparent to the I/O or application component at the real-time flow's source. For instance, in the case of a video server, the I/O subsystem which fills the appropriate buffers for the network layer to empty out, is not affected at all. Since the network subsystem has made the appropriate reservations to accomodate the duplicated flow, it forwards real-time data from the flow to appropriate destinations from the same buffer area. When the first packets of the duplicate flow reach the mobile node, the data arriving on the new flow rather than the original flow is copied to the user buffers. Ideally the mobile should receive from the new flow exactly where it left off in the previous cell if the end-to-end delays along the two flows are the same. If that is not the case, e.g., the new flow passes through a larger or smaller number of hops, then the arrival times of packets with the same sequence numbers will not be identical. In practice, the fact that the mobile terminal can receive one flow at a time means that it is never possible that the two flows are received at the same time. Therefore, a playback buffer is necessary to bridge the discrepancy of arrival times and ensure continuity of real-time flows.

4.2 Performance Considerations

In order to ensure continuous data delivery of real-time flows during the mobility period, it is important to identify the individual components of the cell switching time, and design the network topology and software parameters accordingly. Before a mobile terminal can switch to a new cell, it has to detect the existence of the new cell. The time taken to detect a change in the beacon signal strength is determined by the beaconing frequency (F) of the base stations. There is a tradeoff between the amount of (already scarce) wireless link bandwidth wasted in beaconing and the possibility of real-time service interruption if wireless nodes don't have enough time to decide and perform cell switching.

Because beaconing messages are non-real-time messages, the base stations send them out when they get access to the network in non-real-time mode. As discussed in section 3.3.3, the base station may choose to prioritize sensitive non-real time control information. Hence, if the base station chooses to prioritize beacons, the worst case network-access latency for a beacon message from a base station becomes $MAX(kT\ ,\ MTU/B)$.

The next factor that affects the cell switching time is the history parameter H used by the node which measures how many consistent 'wins' of signal strength it needs to accumulate before it decides to switch. The last factor is the possibly multi-hop connection setup time (C) for establishing a duplicate copy of the existing real-time connection to the new base station on the wired infrastructure. Therefore, the total cell switching time is given by (assuming the network access latency is smaller than the beacon period)

$$H * F + MAX(kT, MTU/B) + \frac{T * (N-1)}{\lfloor \frac{(1-k)T*B}{MTU} \rfloor} + C$$

Note that the connection establishment request message itself has to incur some network access latency since it is a non-real time message ; this contributes to the third term in the above expression, which we use from section 3.3.3. The parameter H only scales F and not the second term since the beaconing timer is set independent of the time when the node got non-real time access to actually send out the beacon. So, the beacon period overlaps with the time during which the base station is trying to get non-real time access. This bound should be smaller than the physical transition period for a mobile terminal to move from one cell to another, which in turn depends on the cell layout, i.e., the size of the overlap region, and the moving speed.

As an instance, consider a 2Mbps wireless network with an MTU of 1500 bytes and a 66msec cycle time with upto 85% of the link bandwidth reserved by real-time flows. Assume that wireless REther chooses to use an H of 3, a beaconing period of 500 msec and that there are 5 nodes on the wireless subnet. Plugging in our number for a two-segment (1 hop) connection establishment time for a mobile terminal (23.72 msec), the worst-case bound comes out to be 1.82 sec. Hence for the given parameter values, as long as a complete cell transition takes less than 1.82 sec, the network will be able to switch to a new flow without the user noticing any interruption in service.

5 Application : REther and the Stony Brook Video Server

The Stony Brook Video Server (SBVS) [2] provides digital video playback to clients on a local area network. Possible applications of SBVS include a digital library of videotaped lectures for use in distance and asynchronous learning and a library of training videos for use in the Stony Brook University hospital. SBVS consists of two cleanly seperated subsystems. The I/O subsystem of SBVS is responsible for efficient real-time retrieval of MPEG-1 video streams striped over a software-controlled disk array to the main memory. For real-time video delivery to clients over Ethernet, SBVS uses REther as its network subsystem. The original wired REther provides the abstraction of a *real-time network session* for every video stream that SBVS serves.

Wireless REther extends this abstraction in a number of important ways. Foremost, it abstracts out heterogeneity in the network by seamlessly integrating wired and wireless network segments. Thus, it transforms REther into an end-to-end solution for network performance guarantee on any LAN environment consisting of Ethernet and

WaveLAN segments. The second important extension that wireless REther contributes is the *mobile* real-time session. Just like the application need not care about heterogeneity in the network, it also need not care about terminal mobility across the network. A mobile real-time session provided to an application may consist of one or more physical real-time flows with changing lifetimes, as the node changes its position in the network. SBVS exploits these extensions provided by wireless REther to avoid redundant disk I/O and memory buffering while duplicate flows are established during the transition period.

6 Related Work

The IEEE 802.11 standard [12] is an emerging MAC standard that aims to provide real-time network access by switching channel access modes for CSMA/CA LANs. 802.11 switches to a polled mode of operation to support performance guarantees, which is reported not to perform very well [14] [11]. An additional constraint is that neighboring cells can not be in the polling mode simultaneously. A related work at Bell Labs [11] aims at providing real-time guarantees on a wireless network by equiping individual stations with the ability to *jam* the channel with pulses of energy of specified duration to gain prioritized access to the channel. Both these mechanism operate at the medium access level and require different a MAC protocol and thus hardware support. Besides being free from any of the limitations of 802.11, the protocol described in this paper does not need any special hardware mechanisms, and is shown to be relatively efficient even without hardware assists. The fact that it is an all-software implementation also provides additional flexibility to integrate with other functionalities. For example, by integrating with wired REther, it provides a complete end-to-end solution to the network performance guarantee problem in the LAN environment.

The Infopad project [13] at the University of California, Berkeley aims at providing a network terminal to a mobile user through a small hand-held wireless device. The wired backbone is a high speed network with wireless *picocells*. Each picocell runs several entities that cooperate to give the applications (called *type servers*) a view of the mobile terminal as if it were a fixed network terminal. These entities include servers for maintaining cell related parameters (radio signal strength, current frequency allocations etc.), network gateways (for routing) and servers for device resource management. The base stations use frequency hopping for the downlinks and direct sequence spread spectrum for the uplinks of the pads and it is thus possible to give performance guarantees to flows destined to each pad with appropriate frequency band allocation. Handoff is implemented through IP multicasting since every pad is assigned a unique multicast IP address. The Infopad work differs from our work in that it provides a highly specialized infrastructure for mobile network terminal access, whereas we address the problem of providing end-to-end performance guarantees in a general-purpose mobile computing environment. No specialized servers (like the X, audio and video servers that are maintained per cell in InfoNet) are needed to ensure real-time data delivery to the mobile terminal. Also the performance guarantee problem on the wired network was never addressed in the Infopad project, whereas our work focuses from the very beginning on the provision of a uniform end-to-end real-time link-layer solution across a switched local area network consisting of both wired and wireless segments. Finally, our system is a pure software implementation and can easily port ro off-the-shelf network interface hardware.

7 Conclusion

Distributed multimedia applications requiring network performance guarantees from the network need an *end-to-end* solution from the network subsystem. While most previous works focused on point solutions based on simulation studies, this paper presents the design and implementation of a working prototype that provides true end-to-end bandwidth guarantees to multimedia applications on the most popular wired and wireless LAN technologies, i.e., Ethernet and WaveLAN. Although obvious from hind sight, the choice of centralized channel arbitration is only reached after several failed attempts to port wired REther to WaveLAN. The prototype performance measurements support this design decision by showing its channel utilization efficiency and the CPU overhead on the base station.

We also addressed the real-time performance guarantee issue in the context of mobile terminals, and presented our solution in ensuring the un-interrupted real-time data delivery during the cell switching period. Finally we described how wireless REther is incorporated into a distributed multimedia application, the Stony Brook Video Server, to provide seamless digital video delivery to mobile portables.

There are several directions that we plan to explore. First, the current work only supports horizontal handoff among WaveLAN subnets. Therefore, the system's use is limited to within a building. For seamless outdoor wireless real-time communication, it is necessary to implement vertical handoff that interfaces WaveLAN to analog cellular networks or CDPD. Second, the issue of minimizing forced cut-offs because the new subnet to which the mobile terminal moves does not have sufficient bandwidth to accommodate transferred connections, is not addressed in this work and needs to be studied more extensively. Finally, the role of the base station can be extended to perform higher-level functions by taking into account application data semantics such as video transcoding to reduce bandwidth usage. How these "intelligent base stations" interact with the real-time network performance guarantee mechanisms deserves further investigations.

References

[1] Chitra Venkatramani and Tzi-Cker Chiueh, "Design, Implementation and Evaluation of a Software-Based Real Time Ethernet Protocol," In Proc. of ACM SIGCOMM' 1995

[2] Michael Vernick, Chitra Venkatramani and Tzi-Cker Chiueh, "Adventures in building the Stony Brook video Server," In Proc. of ACM SIGMM' 1996

[3] A. Muir and J. Garcia-Luna-Aceves, "Supporting real-time multimedia traffic in a wireless LAN," In Proceedings of the SPIE Multimedia Computing and Networking Conference, vol.3020, p. 41-54.

[4] F. Ayo, B. Jafarian and A. Aghvami, "A medium access control protocol for wireless ATM LAN based on polling," IEE Colloquium on ATM Traffic in the Personal Mobile Communications Environment (Ref. No.1997/032), p. 2/1-6 London, UK IEE 1997.

[5] S. Tasaka, K. Hayashi, and Y. Ishihashi, "Integrated video and data transmission in the TDD ALOHA-reservation wireless LAN," Proceedings IEEE International Conference on Communications ICC, p. 1387-93 vol.3, 1995.

[6] S. Yao, B. Awada, I. Labbe, and G. Nourry, "Performance of a UHF multimedia wireless LAN," Proceedings of 2nd IEEE International Conference on Universal Personal Communications, p. 771-5, 1993.

[7] D. Raychaudhuri, et al., "WATMnet: a prototype wireless ATM system for multimedia personal communication," IEEE Journal on Selected Areas in Communications, vol.15, no.1, p. 83-95, Jan. 1997.

[8] Chitra Venkatramani and Tzi-Cker Chiueh, "Design and Implementation of a Real-Time Switch for Segmented Ethernets," Proceedings of 1997 International Conference on Network Protocols, October, 1997.

[9] B. Mah, S. Seshan, K. Keeton, R. Katz, and D. Ferrari, "Providing network video service to mobile clients," Proceedings of IEEE 4th Workshop on Workstation Operating Systems, p. 48-54, 1993.

[10] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala, "RSVP : A New Resource Reservation Protocol," IEEE Network Magazine, Sept. 1993.

[11] J.L. Sobrinho and A.S.Krishnakumar, "Real-Time Traffic over the IEEE 802.11 Medium Access Control Layer", Bell Labs Technical Journal, Autumn 1996.

[12] "Tutorial on draft standard D3.0 of IEEE P802.11", available as http://stdsbbs.ieee.org/groups/802/11/fh.pdf.

[13] Shankar Narayanaswamy, Srinivasan Seshan, Eric Brewer, Robert Brodersen, Frederick, Burghardt, Andrew Burstein, Yuan-Chi Chang, Armando Fox, Jeffrey M. Gilbert, Richard Han, Randy H. Katz, Allan C. Long, David G. Messerschmitt, Jan Rabaey; "Application and Network Support for InfoPad" IEEE Personal Communications Magazine, Mar 1996.

[14] M.A.Visser and M.El. Zarki, "Voice and Data Transmission over an 802.11 Wireless Network", Proceedings of PIMRC'95, Toronto, Canada, September 1995, pp.648-652.