Implementation Experiences of Bandwidth Guarantee on a Wireless LAN

Srikant Sharma Kartik Gopalan Ningning Zhu

Gang Peng Pradipta De Tzi-cker Chiueh

Rether Networks Inc.

99 Mark Tree Road, Suite 301, Centereach, NY 11720

rni@rether.com 631-467-4381

Abstract

The Rether[1] protocol was originally developed to support guaranteed Quality of Service (QoS) for shared Ethernet LANs. In view of the growing popularity of IEEE 802.11-based wireless LANs, we have modified the Rether protocol to provide QoS guarantee on these networks. In this paper, we present the design and software-only implementation of the Wireless Rether protocol for 802.11 networks, as well as our experiences with wireless LAN hardware. Wireless Rether supports QoS for TCP and UDP traffic in both upstream and downstream directions. The protocol can seamlessly inter-operate with any priority-based QoS mechanisms (such as Diffserv [6]) on the wired networks that connect the wireless access network to the rest of the Internet. The QoS requirements of real-time applications are specified as a simple configurable policy. Legacy networking applications can benefit from QoS guarantees provided by Wireless Rether without requiring any modifications. In addition to QoS guarantee, Wireless Rether provides a novel low-latency handoff mechanism for Mobile IP connections over infrastructure-mode wireless LANs.

1 Introduction

With the growing popularity of 802.11 wireless LANs, it is advisable to look beyond connectivity and security issues and start focusing on Quality of Service (QoS) support for advanced multimedia applications, such as one-way video playback or two-way audio/video communication. Typically wireless LANs serve

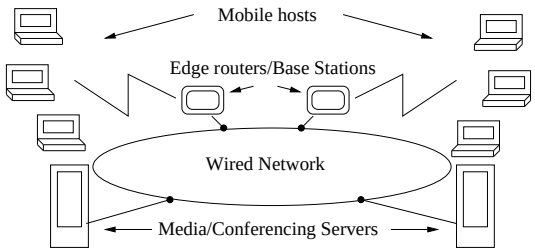

Figure 1: *Wired/Wireless network architecture. The wireless LANs lie at the edge of the wired infrastructure. Typically media playback servers reside on the wired network and stream multimedia data to mobile hosts. On the other hand, audio/video conferencing applications require special conferencing servers to relay traffic among mobile hosts.*

as the access networks to the wired infrastructure for mobile terminals, as shown in Figure 1. Mobile hosts are connected to the rest of the network through Layer-2 access points and Layer-3 edge routers. Mobile IP [3] takes these access networks one step further and allows mobile terminals to roam seamlessly between co-operating 802.11 wireless network segments (having different IP prefixes) without loss of connectivity.

The wireless connectivity and the ability to roam across multiple networks make wireless 802.11 LANs an ideal choice for transient and dynamic networks. A network is dynamic if its constituent nodes are dynamically changing, as in the case where mobile terminals roam from one network to another. Examples of locations that require wireless networking infrastructure include public gathering places like conference venues, airports, university campus etc. where people are always on the move.

As the popularity of wireless LANs increases, multimedia network applications such as audio/video conferencing and media streaming are expected to be deployed on these networks, both of which are sensitive to packet latency and effective bandwidth characteristics of the underlying network. On the wired network, IETF's Integrated Services [11] and Differentiated Services [6, 10] architectures are available to support guaranteed QoS and traffic prioritization respectively above the link layer. IEEE's 802.1p is a layer 2 traffic prioritization standard for switched Ethernet environments. However, no commercially available QoS solution exists for IEEE 802.11 networks. The goal of this work is to develop a bandwidth guarantee mechanism for wireless LANs based on IEEE 802.11 technology.

Since a wireless LAN is a shared medium, collisions arise and subsequently some form of random back-off mechanism is triggered when multiple nodes attempt to send data simultaneously. In order to guarantee deterministic network access and eventually support QoS for different network nodes on a wireless LAN, a special media access control protocol is required that can avoid collisions altogether. The basic idea of the original Rether [1] protocol is to circulate a software token among the Ethernet nodes within a fixed-length cycle, and to permit only the network node that currently holds the token to transmit

data on the wired link. The amount of data a node can send when it is the token holder depends on its bandwidth reservation. Since at most one node is transmitting at a time, there is no collision by construction and every wired LAN node can reserve a different amount of the shared link bandwidth. A unique feature of Rether is that it is designed to be a software-based protocol that can run on existing Ethernet hardware, thus allowing users to leverage their existing network hardware investment.

On the surface Rether appears to be immediately applicable to *wireless* LANs, since a wireless LAN is also just a shared medium. In practice, however, there are major technological differences between Ethernet and 802.11 networks that require significant re-thinking of the details of Rether. This paper describes the design and implementation of a variant of Rether that is specifically tailored to 802.11 wireless networks.

This paper is organized as follows. Section 2 briefly describes the current research related to QoS in wireless networks. Section 3 describes the architecture of the Wireless Rether protocol and the design issues that we encountered. Section 4 describes our prototype implementation of Wireless Rether and its novel features such as the low-latency handoff mechanism. In Section 5, the performance of Wireless Rether implementation is described. Finally in section 6 we summarize our work and outline future research possibilities.

2 Related Work

Quality of Service research on wireless LANs is mainly driven by attempts to make wireless LANs interoperate with telephony networks. To carry voice traffic, 802.11 networks support distributed and point co-ordination functions (DCF and PCF). PCF is a connection oriented capability. It involves a central co-ordination by a *point coordinator* (PC) which initiates and controls *contention free periods*. PCs poll terminals with time bound traffic by sending poll tokens. The distributed coordination function is similar to CSMA/CA during the contention period. The actual use of these protocols are not common and these have been mostly analyzed for performance in simulated environments [4, 5].

IEEE is proposing new 802.11e standard aimed at improving QoS on 802.11a and 802.11b networks. 802.11e aims at proposing a better way to handle time-sensitive traffic for multimedia applications. This is done by accommodating time-scheduled and polled communications during contention free periods similar to PCF functionality in 802.11b but in an improved manner.

HomeRF aims at providing wireless networking for home appliances. The latest standard HomeRF 2.0 targets quality of service by providing different types of services using separate time-slots. HomeRF supports reservation slots of 100Kbps, which are useful only for voice data but not for video. Although

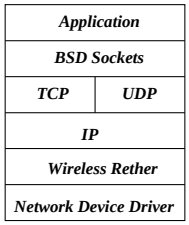

Figure 2: *Wireless Rether protocol layer. Wireless Rether is implemented as a layer between the link-layer hardware's device driver and the IP protocol layer. With this layering structure, Wireless Rether is able to exercise QoS-related control over all outgoing network connections.*

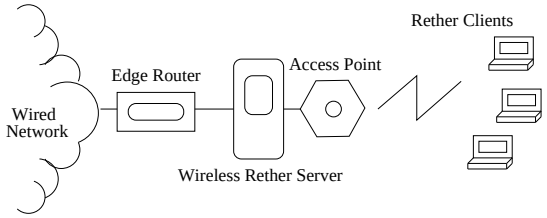

Figure 3: *Wireless Rether Architecture. Each edge network is augmented with a Wireless Rether Server (WRS), which acts as a bridge between the Layer-2 access point and the Layer-3 edge router. All wireless LAN hosts are equipped with Wireless Rether client software.*

HomeRF networks are considered to be slower than 802.11, support for QoS enables IP telephony in home networking.

Lu et.al present an analytic and simulation based study of a fair scheduling algorithm for wireless network [8]. The scheduler is work-conserving and approximates GPS. All the hosts need to set their wireless network interfaces in promiscuous mode in order to learn channel state. This increases power consumption due to unnecessary packet reception. In contrast, we propose a non-work-conserving approach that reduces delay-jitter while at the same time prevents the best-effort flows from getting starved. Furthermore, use of tokens to explicitly arbitrate channel access eliminates the need to set client's wireless interfaces in promiscuous mode. Other works in QoS for wireless networks include mostly algorithmic approaches like Packet Fair Queuing [7] and simulation based approaches like the one presented in [4]. An extensive treatment of programmable mobile networks is done as a part of Mobiware [9] which requires programmable Mac protocols.

3 Design Issues and System Architecture of Wireless Rether

In this section we discuss important design issues in the development of the Wireless Rether protocol, the solutions we chose to address them and the rationale behind these solutions.

3.1 Hardware vs Software Implementation

Wireless Rether protocol can either be integrated with the firmware on the wireless LAN card or as a software layer above the network hardware's device driver. The advantage of the hardware approach is that it can significantly reduce the overhead involved in passing the token between hosts. The disadvantages of the hardware approach are that the solution becomes tied to individual vendor's hardware, the cost of hardware implementation is significantly higher, and there is less flexibility in configuring QoS policies.

On the other hand, a software-based approach has the advantages of being able to work with any vendor hardware, being cost-effective, providing much more flexibility in configuring QoS policies and in refining implementation details of QoS mechanism. Our previous experiences with Rether showed that with careful protocol and software design, the token passing overhead can be kept to a minimum. Hence we decided to adopt an all-software implementation approach for the wireless Rether protocol, whose software architecture is shown in Figure 2.

3.2 Peer-to-peer vs. Centralized Token Passing

A central design decision in Wireless Rether is how the token is passed from one host to the next. One option is to maintain a logical ring among the wireless hosts and to implement a *peer-to-peer token passing* protocol. The host holding the token transmits the token to the next logical neighbor in the ring and transmits an ACK to the previous logical neighbor in the ring. With such a distributed token maintenance protocol there is no single point of failure in the network and hosts can join and leave the the ring at any time. This was the scheme used in *wired Rether* [1] and it worked well in the context of shared Ethernet segments, where all hosts can communicate with each other directly. In the context of wireless LAN, since mobile hosts can move out of each other's transmission range, direct token passing between wireless hosts is impractical and infeasible. However, since all hosts are assumed to be reachable from the access point, the access point is in a better position to relay the token.

To get around the hidden node problem, we chose to implement a *centralized token-passing protocol* in which a central server, called *Wireless Rether Server* (WRS), is placed right between the access point and the wired network and is responsible for granting the token to wireless LAN hosts, called *Wireless Rether Clients* (WRC). The WRS grants tokens to all the WRCs in a weighted round robin fashion. The weight corresponds to the duration for which a WRC can hold the token and thus transmit data over the wireless channel, i.e., the approved bandwidth reservation for the WRC. The sum total of all the weights is smaller than the *cycle time*, i.e. the average token revisit time for each WRC. A portion of the token cycle time is dedicated to best-effort or non-real-time traffic.

This centralized architecture, shown in Figure 3, has several advantages. First, it is the WRS and not the token that maintains the QoS-related state. Therefore token loss is not fatal to the proper functioning of the Wireless Rether protocol. Secondly, since the WRS can intercept all traffic entering and leaving the wireless LAN, it can snoop on the wireless channel to determine the end of a packet that a WRC sends. This can be used to reduce the token passing overhead by eliminating the need for explicit ACKs from WRCs if ACKs are piggybacked with the last packet transmitted. A potential problem with Wireless

Rether is that it now introduces a single point of failure in the WRS. This is not a major concern to us because eventually we expect the WRS to be merged into the access point. Overall, the centralized token passing protocol is much simpler, lightweight, and more efficient than its distributed counterpart.

Combining the WRS and the access point would have lead to a cleaner hardware set-up. However, this integration is not possible currently since commercial off-the-shelf access points do not expose any programming interface to add third-party code. In retrospect, this limitation may be a blessing in disguise, because the separate-WRS architecture requires the resulting Wireless Rether implementation to be independent of and thus able to inter-operate with 802.11 access points from multiple vendors.

3.3 Work-Conserving vs. Non-Work-Conserving

Within a Wireless Rether cycle, the token first visits those network nodes that have made bandwidth reservation (called real-time or RT nodes), and after all the RT nodes have been visited the token visits the network nodes that potentially have best-effort traffic to send (called non-real-time or NRT nodes). Note that *every* wireless LAN node is an NRT node. If the token cannot visit all NRT nodes within a cycle, it continues to visit the NRT nodes in the next cycle starting from where it left off in the previous cycle. Wireless Rether supports a non-work-conserving service model because even when there is no NRT traffic, the token is still passed among NRT nodes until the current cycle ends, i.e. RT nodes can never send data at a higher rate than their reservation even when other RT or NRT nodes have less data to send. A non-work-conserving model reduces the extent of data burst and thus decreases the delay jitters that applications experience at the cost of some wastage in network bandwidth. In addition, this model fits well with the goal of minimizing performance impacts of RT traffic on NRT or best-effort traffic.

3.4 To ACK or Not to ACK

The Wireless Rether protocol requires *mandatory ACK*, i.e. a WRC holding the token would send an acknowledgment message to the WRS at the end of its data transmission. The WRS would transmit the token to the next WRC only upon receipt of the acknowledgment from the previous WRC. The acknowledgment serves two purposes. First, it informs the WRS when the token-holding WRC's data transmission is complete and the channel is free for the next WRC. Secondly, it informs the WRS that the WRC is still alive and participating in the protocol. Unfortunately, a token-ACK message pair incurs considerable transmission overhead over the wireless media - around 2-ms round-trip delay - which is clearly inefficient. Specifically there is a large gap between the air time for the token-ACK message and the token-ACK round-trip delay which suggests that the wireless channel may be idle during most of the round-trip delay

period. [1] Given a 33-msec cycle time, this means that Wireless Rether can only support up to 16 WRCs without even transporting any payload.

To use the wireless channel more efficiently, we considered an *optional-ACK* design, in which the WRS sends a token to the next WRC at the end of the time slot allocated to the current WRC, with or without receiving an ACK from the current WRC. Here we assumed that the WRS is able to predict the end of a WRC's data transmission transaction. The WRCs still send back ACKs, but the WRS does not rely on them for distributing tokens, with the exception that when a WRC is done with its data transmission and there is plenty of time left in its allocated time slot, the WRS will send out a token immediately to the next WRC upon receipt of such ACK. An advantage of the optional-ACK design is that the token-ACK round-trip delay would no longer be on the critical path, and therefore the token cycle can be more efficiently utilized. However we found that the data transmission time over the wireless network cannot be predicted reliably and as a result collisions were induced. In addition to collision, access point buffering delays accumulated over the time. This eventually led to multiple tokens circulating simultaneously in the system, which in turn introduced more collision. The token-ACK protocol could not work in a reliable manner because of this unpredictability of wireless channel usage. Due to these problems we chose to stick with the mandatory ACK design whose advantage of deterministic nature far outweighs the inefficiency due to token-ACK exchanges.

3.5 Intralan Traffic between Mobile Nodes

It is possible that two communicating nodes may belong to the same wireless LAN. In *ad hoc* mode, the packet transmission is peer-to-peer with no additional overhead. But in *infrastructure* mode the source first hands all packets over to the access point which then relays the packets to the destination node. Since the access point has to retransmit packets over the same wireless medium the transmission cannot be cut-through but has to be store-and-forward. Thus in infrastructure mode each Intralan packet effectively occupies two wireless packet transmission slots. This requires special consideration when making bandwidth reservation: The reservation mechanism needs to check if the destination host is an Intralan host and if so the bandwidth reservation should be doubled.

Since the number of hosts in a wireless LAN is limited and the WRS has the complete knowledge of participating hosts, periodically the list of mobile hosts can be transmitted to all the WRCs to update their local copies which can be used by WRCs to detect the intra-lan communication. WRCs in this scenario

[1] The header of an 802.11b packet is physically transmitted at 1 Mbit/sec to maintain backward compatibility with 802.11, even though the packet body can be transmitted at 11 Mbits/sec. Therefore small messages are extremely inefficient on 802.11b networks.

can make a bandwidth reservation for the required additional transmission slots. The problem with this approach is that, when total bandwidth reservation is close to link capacity, earlier packets forwarded through the access point may collide with the subsequent packets from the same sender. This can result in degradation of performance and affect the QoS guarantees. The alternative approach which we chose to implement is to route all packets through the WRS and shift the responsibility of reserving additional downstream bandwidth over to the WRS. This solves both the problems mentioned above.

3.6 Bandwidth Reservation Mechanism

The QoS mechanism of Wireless Rether works seamlessly with the QoS mechanism in the wired network in order to achieve end-to-end QoS for applications. A major design issue here is whether or not to support end-to-end application level signaling such as RSVP [11]. Such a signaling mechanism would be ideal from the perspective of new QoS-aware applications that can be written from scratch to make use of explicit signaling. However, we conclude this is impractical for two reasons. First, end-to-end per-connection signaling requires that intermediate routers maintain state about each connection. This tends to hurt scalability and is one of the main reasons that RSVP has failed to take off. The second reason is that many legacy applications exist which require QoS guarantees, but cannot be rewritten to make use of end-to-end signaling.

One of the primary goals of developing the Wireless Rether protocol was to support QoS for legacy and third party applications without modifying them. Since such applications cannot perform explicit reservation, there is no direct way of determining their bandwidth requirements. The solution is to use an indirect way of determining the bandwidth requirements. The end-points for such network applications are network hosts and ports. A commonly used convention is port based reservation mapping. In such a scheme the network port number used determines the bandwidth requirements. The mapping between the port numbers and the bandwidth is specified via a system-wide QoS policy. A further enhancement is to use the network addresses in conjunction with wild cards. The reservation mechanism in Wireless Rether uses a *quintuple specification* of

{`SrcAddress/Mask, DestAddress/Mask, SrcPortRange, DestPortRange, Bandwidth`}

For example, a policy specification like {`192.168.1.0/24, 192.168.2.6/32, *, 80-80, 1000`} would specify a reservation of 1Kbits/sec for Web traffic between the subnet 192.168.1.x and HTTP server 192.168.2.6. If a matching specification from policy is found for any new packet stream, the reservation request for the stream is sent to the WRS. The WRS performs admission control check to admit or reject the reservation request for the new stream. If the request is admitted, at run-time the network

packets of the corresponding stream are queued in a special real-time (RT) packet queue and dispatched by the network scheduler according to the reservation. If the WRS rejects a request, the WRC treats the stream as a best effort stream.

3.7 Support for TCP

Wireless Rether protocol is designed to reserve bandwidth for UDP, TCP and ICMP traffic types. TCP traffic is inherently bidirectional in nature due to presence of TCP ACKs and hence requires special consideration. In order to guarantee bandwidth reservation for TCP data traffic in one direction, it is necessary to reserve bandwidth for TCP ACKs in the reverse direction since absence of ACKs would trigger TCP's congestion control mechanism and cause the sender to reduce transmission rate. As a result, the perceived bandwidth for TCP connection is less than the available bandwidth, leading to a wastage of reserved bandwidth.

Wireless Rether can transparently detect TCP streams and perform reverse bandwidth reservation for TCP ACKs. The WRC module snoops on each outgoing packet. If the packet happens to be a TCP ACK, it consults the policy table to determine if the data traffic for the same TCP stream in the reverse direction has an associated reservation in the policy table. If this is the case, the WRC module makes reservation for TCP ACKs with a bandwidth reservations which is a certain percentage of that in the reverse direction. This percentage is a configurable parameter and is currently set to 10%. This value is empirically determined after observing the bandwidth required for non-piggy-backed TCP acknowledgments and is believed to be a conservative estimate.

3.8 Buffer Overrun

It is quite possible that the data transmission rate for an application may exceed its reserved bandwidth. It is also possible that the available link bandwidth for a particular node may be less than the expected link bandwidth because of poor radio characteristics. In both cases the packet dispatching rate will be less than the packet generation rate by the applications. Such packet overrun will cause an accumulation of packets in the WRC buffers. There are two options to deal with this *run over* effect. The WRC module can choose to discard excess packets after a certain limit or can choose to queue these packets in best-effort queues. Both approaches have advantages and disadvantages. If the radio characteristics of a channel are poor then utilizing the best-effort bandwidth to compensate for perceived bandwidth loss of real time streams seems to be a justifiable approach. If the packets are discarded then the degradation in the quality of application because of data loss may not be acceptable. On the other hand if the application is aggressive

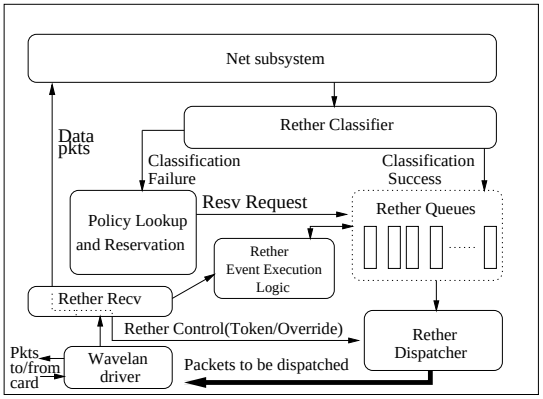

Figure 4: *WRC Software Architecture: Rether Classifier intercepts all the outgoing packets and queues them in corresponding Rether Queues. Policy Lookup mechanism is responsible for processing of reservation requests. The event execution logic takes actions based on control messages from WRS. Rether Dispatcher is responsible for dispatching the packets as per the QoS requirements.*

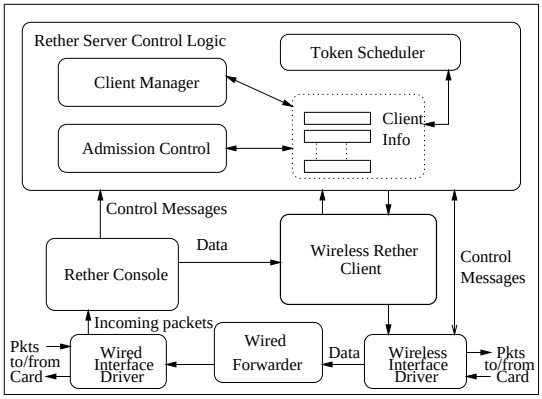

Figure 5: *WRS Software Architecture: Each WRS has a built in WRC module to deal with the downstream traffic. The upstream traffic is forwarded to the wired network in a prioritized manner. Rether Console deals with the management messages from remote machines. Server control logic is responsible for maintaining client related data structures, admission control and channel access scheduling.*

and starts generating packets in excess of reservation then redirection of excess packets to the best-effort queue may have an undesirable effect on the legitimate best-effort traffic. While dealing with aggressive applications, discarding of packets seems to be the appropriate approach. On close observation it can be deduced that an optimum approach would be to somehow avoid the data loss without penalizing the existing best-effort traffic. If we assume that the degradation in the radio characteristics are transient and the mobile host will move to a better radio region within a reasonable period time we can solve this problem by providing longer per-connection buffers to queue the packets without discarding them. In Wireless Rether we decided to adopt the approach of providing larger buffers which could absorb transient radio characteristics problems.

4 Prototype Implementation

The Wireless Rether prototype assumes a client-server architecture where the *Wireless Rether Server* (WRS) has a global view of the bandwidth usage and the *Wireless Rether Clients* (WRCs) act according to the directives from the WRS. The WRS and WRCs communicate with each other by exchanging *Rether Control* messages. These include messages like *beacons* for liveness and presence detection, *registration* requests and replies, *reservation and dereservation* requests and replies, and *token-ACK* messages, etc. The overall setup of Wireless Rether is depicted in Figure 3.

4.1 Wireless Rether Client

In a Wireless Rether network, all the nodes which transmit data on the wireless channel act as the *clients* for the wireless channel. In order to provide proper QoS guarantees it is imperative that *all* the clients should be enabled with Rether software layer. Every WRC is granted an exclusive channel access in a weighted round robin fashion by means of token messages on a regular basis. The weight given to each WRC corresponds to the allocated reservation for the WRC. If the WRC has no reservation then it receives a *non-real-time* (NRT) token once in a while to send out best-effort (NRT) traffic. The internal architecture of WRC is shown in Figure 4.

The WRC maintains different Queues for all the data that has to be transmitted over the network. These queues are *Control Queue* for Rether messages, *NRT Queue* for best effort traffic, *Special NRT Queue* for prioritized traffic (which needs no bandwidth guarantees e.g. urgent messages or advisory ICMP messages etc.) and separate *RT Queues* for every real-time connection. As described earlier, Rether follows quintuple specification scheme for the reservation specifications. These quintuples are placed in kernel Policy Table for fast lookup while admitting new streams.

After initialization the WRC listens on the network for WRS BEACONS without transmitting any data. All the data packets generated by user applications are held back to avoid interference with the functioning of an existing Rether network. If a beacon is not received within a specified timeout period the WRC switches to normal mode and starts participating in the network as a CSMA/CA host. On receipt of beacon a registration request is sent to the WRS. On successful receipt of registration reply the WRC starts participating in the Rether Network.

Many operations in WRC are driven by timeouts and asynchronous control messages. The WRC *Event Handling* logic takes care of managing events like token timeout, reservation replies, override messages during transmission, end of RT stream etc. The Event handler responds to these events by taking appropriate cleanup action, enabling of real time queues, ceasing transmission, sending dereservation requests respectively.

Each outgoing packet in a client is classified by WRC as an RT or best-effort packet. Each RT packet is queued in a corresponding RT queue that is created by Wireless Rether when the first packet belonging to the RT stream is encountered by the Rether Classifier. Upon creation of every new RT queue a lookup on the Policy Table is performed to generate and send an appropriate reservation request for this queue.

The Best-effort packets are classified into three categories, namely, *control*, NRT and *special* NRT.

Control messages are Wireless Rether's protocol messages. Special NRT packets are the urgent messages but do not require any bandwidth reservation (such as ICMP, IGMP and application-level control messages). Any other best-effort traffic is classified as NRT.

Whenever a token visits a client, Wireless Rether's packet scheduler can dispatch packets from appropriate queues of the client using *deficit round robin* scheduling where the deficit can be either *time-based* and/or *byte-based*. Time-based policy dispatches packets based on share of token cycle time of the client whereas byte-based policy dispatches packets based on number of bytes reserved by a client during each cycle. Dispatching packets based on time share gives a fairly good control over the duration of transmission but results in a bursty dispatch of packets when the wireless bandwidth is better than expected. Dispatching packets based on number of bytes to be transmitted during each cycle avoids burstiness but results in long token holding time if the link bandwidth drops below the expected level. To overcome these problems we implemented a *hybrid* packet scheduler in which the limiting parameters are both the reserved time share and the data share per cycle. In this approach the packet are dispatched till either the time share expires or the required amount of data is transmitted. This scheme properly utilizes bandwidth in case of conservative reservations and also deals gracefully with the variable link bandwidth conditions.

4.2 Wireless Rether Server

The Wireless Rether Server (WRS) is the central node of a Wireless Rether network that has the overall control of the entire network. The WRS is responsible for distributing the wireless link bandwidth among the wireless nodes by providing them channel access tokens. Since all the downstream traffic is transmitted by the access point, which cannot run the Rether protocol, the WRS also takes the role of a proxy WRC for the access point. The internal architecture of WRS is shown in Figure 5.

All the client related information is consolidated in a central data structure called the *Client Info Table*. This table holds an entry for each WRC which is created at the time of registration. Each entry carries information about a WRC such as identification, reservation status, behavior history, the best-effort load reported etc.

Many activities in WRS are time bound activities, e.g. sending beacons soliciting registration, distributing tokens amongst clients, waiting for response from a client within a timeout period etc. Since these activities need to be performed in an overlapped manner, WRS employs a timer driven notification scheme. All the WRS modules set and reset various timers that are managed by a central timer thread. The timer thread notifies the modules whenever appropriate timers expire and the modules in turn take the required

actions.

The primary responsibility of WRS is to provide access tokens to the WRCs in a timely manner for their real-time traffic and in a fair manner for their best-effort traffic. For this purpose it consults the Client Info Table data structure which holds the information like the reservation for each client, the best-effort load index for each client, time of previously issued tokens, etc. Based on this information the most eligible WRC is identified and a token is sent to it. It is quite possible that the transmission from some client may exceed its allocated time share because of poor radio characteristics. Usually such time share extensions are absorbed by the time share reserved for the best-effort traffic. But when the channel is heavily loaded this may result in an extension of the cycle length. In order to provide QoS guarantees the WRS needs to maintain the average cycle length close to the specified cycle length. The token scheduler employs a cycle length compensation technique wherein it keeps track of recent cycle lengths and dynamically adjusts the cycle length by growing and shrinking best-effort time slots in future cycles. The WRS also acts as a proxy WRC for the access point and hence includes the entire functionality of a WRC.

Whenever a WRC detects a new stream it performs a policy lookup to check the reservation requirements of that stream. If a matching quintuple is found, it sends a reservation request to the WRS. The WRS verifies whether the request can be admitted or not and accordingly sends back a positive or negative reply to WRC. The WRS adopts a simple static admission control policy where it consults only the reserved bandwidth and checks whether the new admission would exceed the allowed bandwidth share for real-time streams.

The *Client Manager* in WRS is responsible for managing all the local resources allocated to clients in the form of data structures. The Client Manager handles the registration, deregistration requests and client timeouts. The *Rether Console* exposes a management interface to remote hosts for remote management. Current functionality of Rether Console is limited to changing configuration and exchanging statistics. This can be extended to support a full fledged management using SNMP. The WRS also acts as a simple link level *Forwarder* of the upstream traffic from wireless network to the wired network.

4.3 Low-Latency Hand-off

One of the principal advantages of wireless networking is the flexibility provided by host mobility. Mobile IP protocol [3] is an extension to the IP protocol to support mobility of hosts without having to reconfigure the IP address when a host moves from its home subnet to a foreign subnet. *Foreign agents* located in the new subnet periodically broadcast *advertisements* so that any new mobile IP host moving into the subnet

can register with the foreign agent and keep its connections with rest of the world alive. The latency of network level (Mobile IP) handoff is governed by the periodicity of the *agent advertisements*. Since these advertisements are broadcast packets, Mobile IP specifications limit the periodicity to a maximum of one every second for performance reasons. This scenario is not encouraging when the handoff latencies are required in sub-second ranges.

One of the design goals in Wireless Rether was to support mobility between subnets so that protocols such as Mobile IP can operate smoothly. In the current design, the Wireless Rether server periodically broadcasts *beacon* messages on the local wireless subnet such that any new host can register with the Wireless Rether server by responding with a *register* message. When a mobile host moves from one subnet to another, its network card (or the card's device driver) detects the new access point and performs a link-level handoff to the new access point.[2]

When an 802.11 network is operating in the infrastructure mode, mobile hosts cannot receive the advertisement messages from the foreign/home agents from the new subnet until the link level handoff is complete. Thus the mobile nodes do not have a prior knowledge of the agent to be contacted for network level handoff. The network level handoff is initiated only after the advertisement from the new agent is received. Hence the handoff latency is dependent on the agent advertisement frequency and the link-level handoff delay.

Following the link layer handoff, the mobile host can receive *beacon* messages from the Wireless Rether server in the new wireless subnet and register with it by sending a *registration* message. These beacons are piggybacked with *cached foreign agent advertisements*. Wireless Rether module on clients transparently retrieves these piggybacked agent advertisements and provides them to the Mobile IP software on clients. Further, Mobile IP registration requests are expedited by providing extra transmission slots for newly registered clients. All the above optimizations together reduce the hand-off latency to less than 100 msec, rather than a few seconds in the generic Mobile IP implementation. A more detailed description of Mobile IP support in Wireless Rether is available in a separate paper[12].

4.4 Prototyping Platform

The current Rether prototype is implemented on the Linux platform (kernel version 2.2.16, user environment Redhat 7.0 and Redhat 6.2) The WRS is a Pentium II - 400MHz machine with 128MB of RAM, and EEPRO100 Ethernet cards. The WRCs are a mix of IBM compatible desktops (Pentium III 650 MHz) and portables (Pentium III 600 MHz) with 64MB RAM and PCMCIA Lucent Orinoco cards as wireless

[2]We used Lucent Orinoco wireless network cards and access point in our testbed and found this to be the behavior. It is quite probable that cards and access point from some other vendor might behave differently.

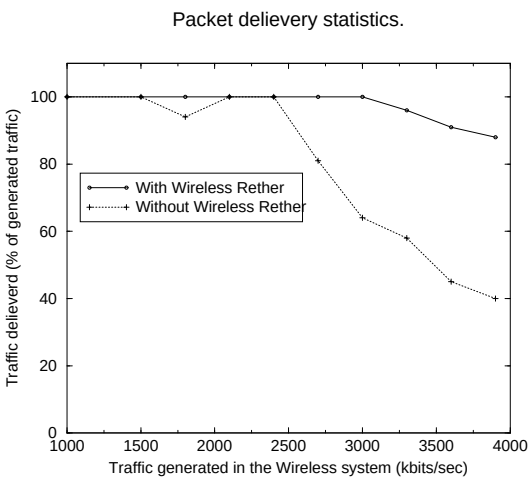

Figure 6: *Comparison of packet loss with and without Wireless Rether. Even with just three senders transmitting, packet loss upto 60% is observed without Wireless Rether. This can be limited to a mere 10% by enabling Wireless Rether while providing bandwidth guarantees.*

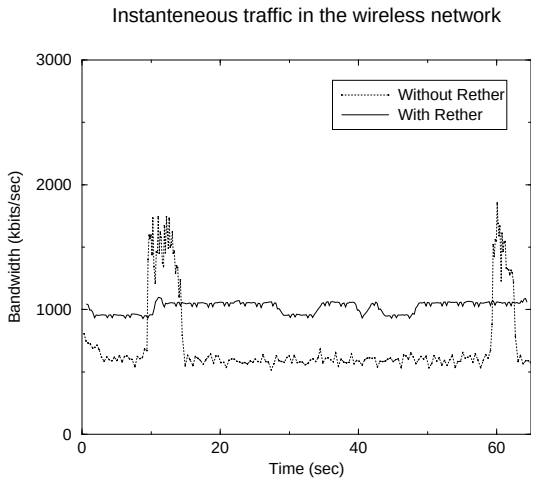

Figure 7: *When there is no QoS in the network the transmissions tend to be bursty in nature. This bursty nature can be reduced by enabling QoS.*

interfaces. The desktops use PCI-PCMCIA adaptors to interface with the Orinoco cards. The access point used is Lucent AP-1000 with Orinoco PCMCIA cards. The wired interface of access point operates at 100Mbps and the wireless interface is configured to run at 11Mbps.

5 Performance Evaluation

5.1 Bandwidth Guarantee

Although the peak link bandwidth of the 802.11b hardware is 11 Mbps, the sustained bandwidth that is attainable on the wireless network is around 6.5 Mbps when a single sender transmits 1500-byte packets without collision. When there are multiple senders, collisions result in packet loss and hence reduction in the overall throughput. Our experimental setup consisted of three hosts transmitting simultaneously - two of them upstream and one downstream. Figure 6 shows a comparison of packet loss observed in the presence and absence of Wireless Rether. Without Wireless Rether, packet loss starts to appear even at a total load of 2.5 Mbps, beyond which the loss increases drastically to 60% when the load is around 4 Mbps. In contrast, with Wireless Rether, the packet drop rate stays within 10% of the total system load. The primary reason of packet loss is the buffer overflow at the WRC that occurs due to reduced throughput at higher bandwidths.

Another advantage of Wireless Rether is that it reduces the bursty nature of the traffic. Figure 7 compares the throughput variation of a network connection over time with and without Wireless Rether.

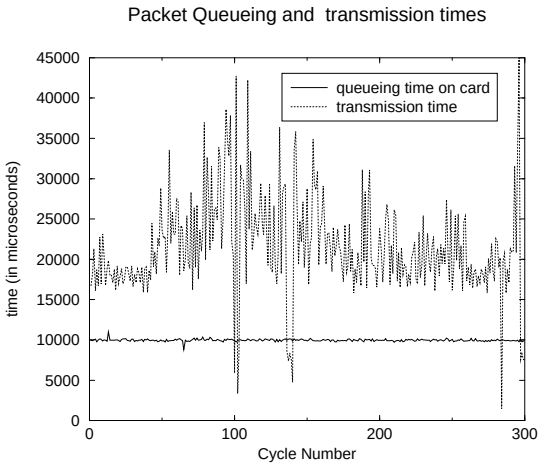

Figure 8: *Observed packet transmission times per cycle and the queuing time for the packets. The packet transmission time is not constant for the same amount of data across successive cycles. The queuing time is very stable and considerably less than the transmission time. In this case the sender node is transmitting data at rate 3 Mbps and the packet size is 1400 bytes.*

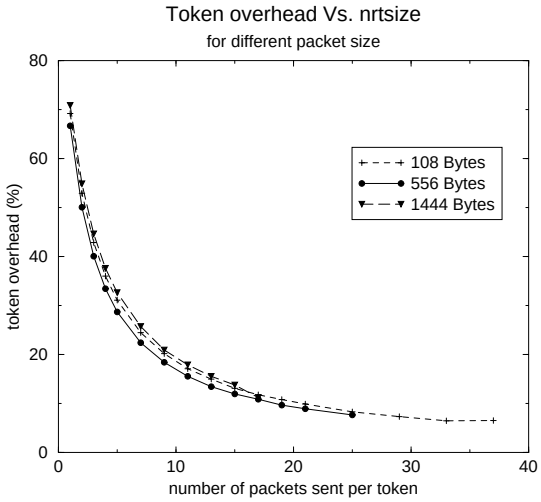

Figure 9: *Rether overhead in terms of % lost time. Maximum overhead is around 70% when only one data packet is transmitted per token-ACK exchange. The overhead reduces to 7% to 11% when there is only one token-ack exchange in the entire cycle.*

In both cases three senders attempt to transmit data at a rate of 1.1 Mbps. The figure shows bandwidth received by one of the three streams. It can be seen that the deviation in the bandwidth guarantee for a reserved stream is minimal and is within 5% of the reservation. Without Wireless Rether the received bandwidth tends to be bursty and actual throughput is much lower than the required rate of 1.1 Mbps. When Wireless Rether is in place, the same sender receives the requested bandwidth and the burstiness is greatly reduced. The deviation in the bandwidth guarantee for a reserved stream is minimal and is within 5% of the reservation.

5.2 Extended Cycle Time

Even when the channel is guaranteed to be collision free the actual time for transmission over the wireless channel is not deterministic. It is observed that the transmission time for a constant amount of data varies from instance to instance. The direct effect of this irregular transmission time is an extended cycle length. Rether deals with this cycle length extension in a graceful manner by carrying out cycle length compensation. Since the average wireless channel transmission time over several cycles is predictable and is dependent on the available link bandwidth, the long term transmission behavior over the wireless channel can be controlled. The cycle length compensation technique works by dynamically growing and shrinking the cycle length so that the deviation from the requested cycle length stays minimal without affecting the QoS guarantees of real-time streams.

Compared to the transmission time, the time to queue data on wireless LAN card is surprisingly stable. Also the packet queuing time with the card is small compared to even the average channel usage time. This observation can be attributed to the fact that usually data queuing is carried out using DMA and if the card has multiple buffers then consecutive packets can be queued independent of the transmission status of the previous packet. The channel usage characteristics and the queuing time behaviors are shown in Figure 8. The channel transmission time is measured by snooping on the wireless network using a machine that is set in promiscuous mode. The queuing time is measured by taking timestamps in Rether kernel before and after the queuing operation.

This mismatch in queuing time and transmission time has an immediate impact on the Rether scheduler. Since the Rether scheduler is a hybrid scheduler where the transmission is stopped when either the required amount of data is transmitted or the allocated channel time is completely utilized, the knowledge of the exact duration of time spent during transmission becomes pertinent. Since the average transmission time is more important than the instantaneous rate, the knowledge of channel usage can be accurately obtained from WRS itself. The WRS can provide the WRC with this information while sending the next token. The WRC can correlate the channel usage time and the amount of data transmitted over the past few cycles and estimate the current available link bandwidth. Using this information the amount of data that can be transmitted over the next few cycles can be determined.

5.3 Token Passing Overhead

Every Rether transmission is limited by a token and an acknowledgment for the token. For small bandwidth reservations the data to be transmitted per cycle can be as low as a single packet. In this case Rether needs to pay a heavy performance penalty of two small packets in each token visit. Figure 9 shows that maximum overhead for all packet sizes, when highest possible number of clients are active, is around 70% when only one data packet is transmitted per token-ACK exchange and is around 7% to 11% when there is only one token-ACK exchange in the entire cycle. Figure 11 shows that the maximum number o off token-ACK exchanges in a cycle, and hence number of WRCs, is around 12 or 13 for small payload packets and around 5 or 6 for large payload packets with a cycle time of 33 ms.

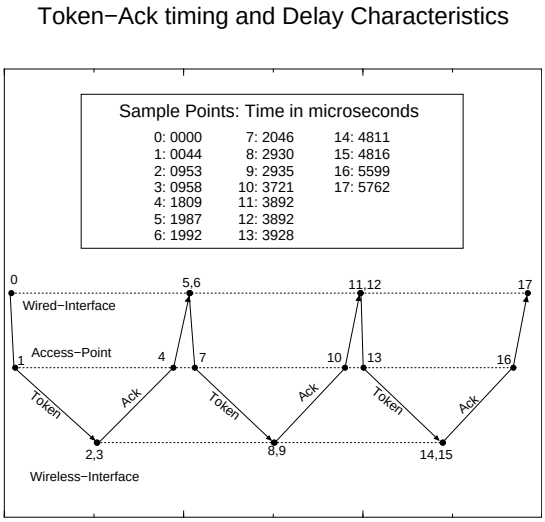

Figure 10: *Detailed timing observation of a Token-Ack transactions and delay between wired and wireless network forwarding.*

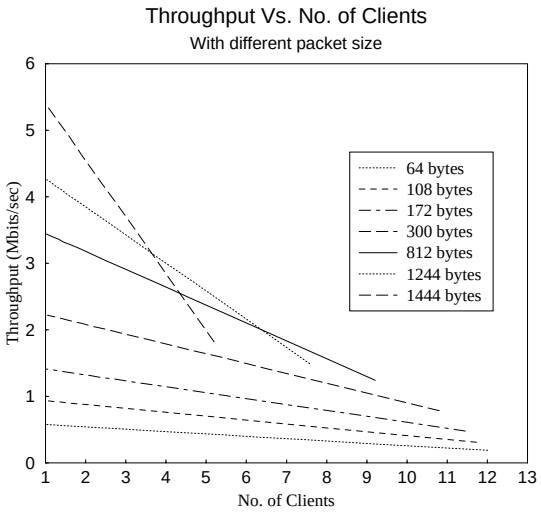

Figure 11: *Maximum throughput possible in Wireless Rether for different packet sizes with different number of clients.*

5.4 Peculiarities of Wireless LAN

5.4.1 Access Point delays

The access point is a link layer bridge between wireless and wired networks. Packet forwarding delay on the access point between the wired and wireless networks is not symmetric.[3] For transmissions on 100 Mbps wired segment, the delay introduced for each packet is of the order of 50 to 200 microseconds. On the other hand for wireless segments this delay is of the order of milliseconds. These delay characteristics are shown in Figure 10. Because of this asymmetry between the speeds, the access point needs to buffer the data received from the wired interface before transmitting it over the wireless interface. This buffering introduces a time skew between the transmission of packets by the wired host (WRS) and the actual reception by the wireless clients. This time skew may lead to timeouts on WRS, token retransmissions and subsequently degraded performance. We address this asymmetry problem by carefully pacing packet transmissions on the WRS to match the wireless network speed.

5.4.2 Intralan traffic

Intralan traffic in infrastructure-mode wireless networks requires special care. Since the access point acts as a repeater for all the Intralan packets, self-collisions among received and relayed packets is possible, which leads to packet loss and degradation in throughput. The throughput for Intralan traffic when self-collision occurs is shown in Figure 12. With wireless Rether all Intralan traffic is diverted through the

[3]This is an observed behavior with Lucent AP1000 access point. Other access points may behave differently.

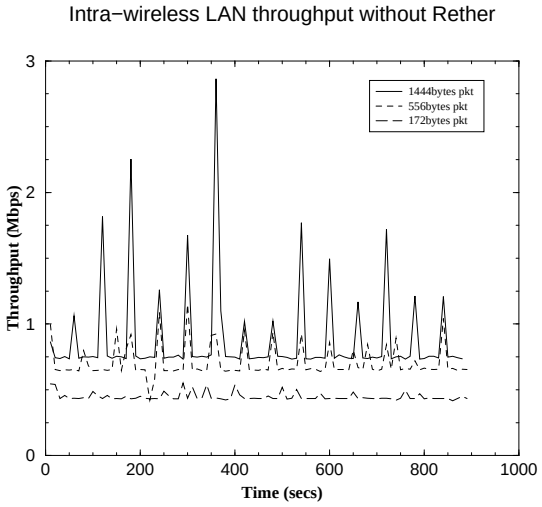

Figure 12: *Channel usage in a wireless LAN without Rether for Intralan traffic. The traffic from a sender collides with the traffic forwarded by the access point resulting in performance degradation.*

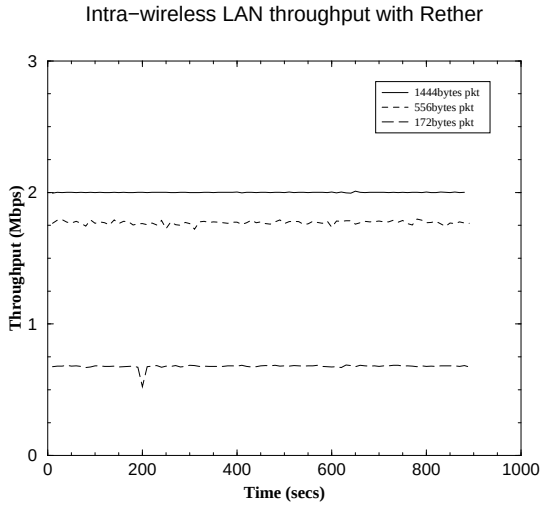

Figure 13: *Channel usage in a wireless LAN with Rether for Intralan traffic. All traffic is diverted through WRS and hence there is no self-collision resulting in better performance. The small kinks in the graph occur due to occasional token loss.*

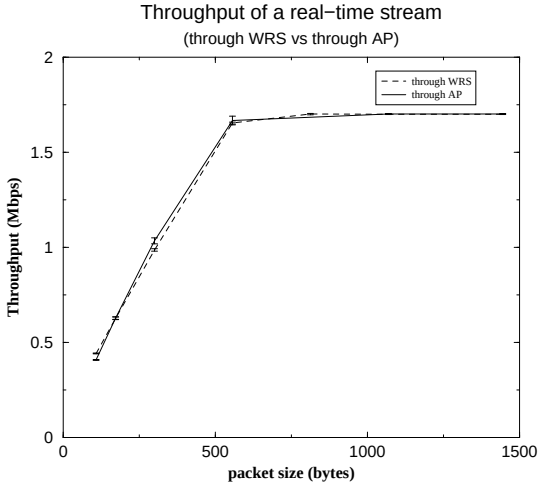

Figure 14: *A throughput comparison of Intralan traffic through access point and WRS at 33ms. There is no performance penalty for diverting the traffic through WRS. The vertical lines in the graph indicate the deviation from the normal throughput.*

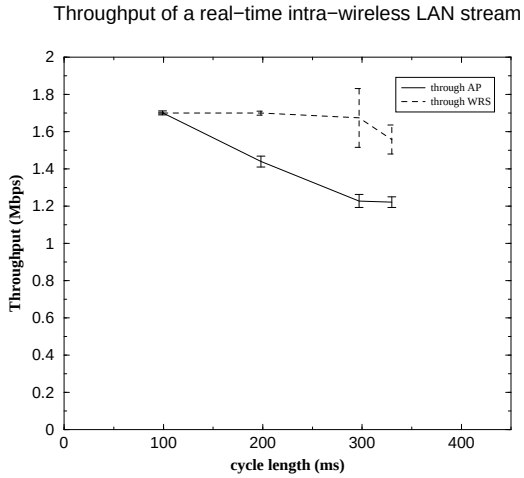

Figure 15: *Comparison of Intralan traffic through access point and through WRS for different cycle lengths. At higher cycle lengths the performance for traffic through access point drops whereas the performance for the traffic through WRS is unaffected to a large extent. The vertical bars in the graph indicate the deviation from the normal throughput.*

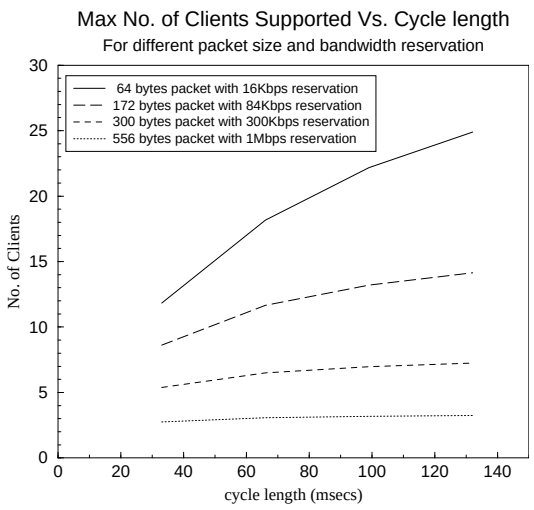

Figure 16: *Scalability of Rether. The maximum supported clients increase with increasing cycle length. The base cycle length is 33 ms.*

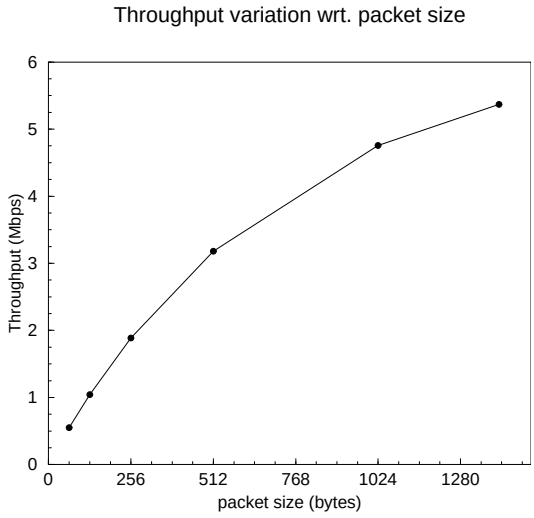

Figure 17: *Bandwidth throughput variation with packet size. As the packet size increases the maximum possible bandwidth also increases.*

WRS, then there is no self-collision and the QoS guarantees are maintained. The resulting throughput in this case is shown in Figure 13. It is also possible to limit self-collision by transmitting the data from the WRC in a staggered fashion. Packet staggering is automatic because the Wireless Rether scheduler limits the amount of data transmitted during a cycle. As a result, when the cycle time is small, it is not necessary to divert Intralan traffic through the WRS. A performance comparison of throughput for cycle length of 33 msec when diverting data through the WRS and relaying through the access point is shown in Figure 14. But at cycle lengths greater than 100ms, the stagger size required increases and relaying Intralan packets only through the access point is less efficient that through the WRS and the access point. A comparison of throughput with varying cycle times when relaying packets through the access point and diverting through the WRS is shown in Figure 15.

5.5 Scalability Study

The number of connections or mobile clients that Rether can support increases with the cycle length due to the reduction in the token-ACK overhead. The assumed base cycle length for Rether is 33 milliseconds. Rether can provide QoS guarantees for at most 3 clients with bandwidth reservation of 1 Mbps each when the cycle length is 33 ms. The number of clients that can be supported can be increased to 4 by increasing the cycle length to 132 ms. Similarly for a reservation of 300 Kbps each the maximum number of clients is 6 at 33 ms and can be increased to 8 at 132 ms. For 16 Kbps reservations the maximum number of clients is 11 at 33 ms and it increases upto 25 for 132 ms. Figure 16 shows the relationship between cycle time

and the maximum number of clients that can be accommodated. Although cycle time increase increases the number of clients that can be supported, it also results in a burstier traffic.

Figure 17 shows how the total bandwidth of a wireless LAN available for reservation varies with the packet size. For smaller packet sizes the token-ACK overhead dominates. As the packet size increases the token-ACK overhead decreases since the total amount of traffic increases with the packet size. The maximum bandwidth available for reservation with 64-byte packets is around 1 Mbps. It increases to 2 Mbps with a packet size of 256 bytes and reaches 5 Mbps when the packet size is 1400 bytes.

6 Conclusion

In this paper we presented the design and implementation of Wireless Rether protocol that provides QoS guarantees on 802.11 networks. Wireless Rether grew from attempts to adapt the original Rether [1] protocol to wireless LANs. It adopts a centralized token passing architecture and supports a non-work-conserving service model that reduces the extent of data bursts and decreases delay jitters. Sequential token-ACK message exchanges between Wireless Rether Server (WRS) and Wireless Rether Clients (WRCs) ensure that the wireless channel is kept collision free, thus increasing the effective throughput of the channel. Wireless Rether supports host-address and port-based reservation for real-time traffic streams. For real-time TCP streams, it transparently performs reverse bandwidth reservation for TCP ACKs. It uses a hybrid packet scheduler at WRCs that limits the transmission rate of real-time streams based on time-share as well as byte-share per cycle. Wireless Rether inter-operates with higher layer protocols such as Mobile IP to support roaming of mobile terminals. Our extensive performance measurements validate the ability of Wireless Rether to provide bandwidth guarantees over wireless LANs. Wireless Rether also supports a novel low-latency hand-off mechanism that reduces the Mobile IP hand-off latency to under 100 msec. We believe wireless Rether is the most comprehensive QoS system that has ever been built for 802.11 networks.

In terms of future work, we plan to extend the bandwidth guarantee mechanism to wireless multicast traffic. Also, we are currently investigating reducing the control message overhead in wireless Rether by exploiting the MAC-layer header, which is available through a RF-mode monitoring facility. An immediate application of the Wireless Rether technology lies in its utility value as a wireless LAN traffic manager, which regulates only traffic in downstream direction. Finally, the traffic shaping ability of Wireless Rether can be leveraged to reduce power consumption on the wireless NICs at the clients.

References

[1] Tzi-cker Chiueh, Chitra Venkatramani, “Supporting Real-Time Traffic on Ethernet,” Proceedings of IEEE Real-Time Symposium, 1994.

[2] Prashant Pradhan, Tzi-cker Chiueh, “Real-Time Performance Guarantees over Wired/Wireless LANs,” IEEE Conference on Real-Time Applications and Systems, Jun 1998.

[3] C. Perkins, ed., “IP Mobility Support,” RFC 2002, Oct 1996.

[4] Andreas Kopsel, Jean-Pierre Ebert, Adam Wolisz,“A Performance Comparison of Point and Distributed Coordination Function of an IEEE 802.11 WLAN in the Presence of Real-Time Requirements,” Proceedings of Seventh Int’l Workshop on Mobile Multimedia Communications, Oct 2000.

[5] S. Khurana, A. Kahol, S. K. S. Gupta, P. K. Srimani, “Performance evaluation of distributed coordination function for IEEE 802.11 LAN protocol in the presence of mobile and hidden terminals,” Seventh Int’l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Oct 1999.

[6] W. Feng, K. Kandlur, D. Saha, K. Shin, “Adaptive packet marking for providing differentiated services on the Internet,” Proc. Intl. Conf. on Network Protocols, Oct 1998.

[7] T.S. Eugene Ng, I. Stoica and H. Zhang, “Packet fair queuing algorithms for wireless networks with location-dependent errors,” IEEE Infocom’98, Mar 1998.

[8] Songwu Lu, Vaduvur Bhargavan and Rayadurgam Srikant, “ Fair Scheduling in Wireless Packet Networks, ” IEEE/ACM Transaction on Networking, 1999.

[9] O. Angin, A. T. Campbell, M. E. Kounavis and R. R.F. Liao, “The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking,” IEEE Personal Commun. Mag., Aug 1998.

[10] P. Almquist, “Type of Service in the Internet Protocol Suite,” RFC 1349, Jul 1992.

[11] Lixia Zhang, et. al., “RSVP: A New Resource Reservation Protocol,” IEEE Network Magazine, Sep 1993.

[12] Srikant Sharma, Ningning Zhu and Tzi-cker Chiueh,“Low-Latency Mobile IP Handoff for Infrastructure-Mode Wireless LANs,” Technical Report-119, Experimental Computer Systems Laboratory (ECSL), Dept. of Computer Science, SUNY at Stony Brook.