Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

Asynchronous Multihop Wireless Networks for Real-time Support

Chunhung Richard Lin
Dept. of Computer Science and Information Engineering
National Chung Cheng University
Chia Yi 621, TAIWAN
chlin@cs.ccu.edu.tw

Abstract

Personal communications and mobile computing will require a wireless network infrastructure which is fast deployable, possibly multihop, and capable of multimedia service support. The first infrastructure of this type was the Packet Radio Network (PRNET), developed in the 70's to address the battlefield and disaster recovery communication requirements. PRNET was totally asynchronous and was based on a completely distributed architecture. It handled datagram traffic reasonably well, but did not offer efficient multimedia support. Recently, under the WAMIS and Glomo ARPA programs several mobile, multimedia, multihop (M^3) wireless network architectures have been developed, which assume some form of synchronous, time division infrastructure. The synchronous time frame leads to efficient multimedia support implementations. However, it introduces more complexity and is less robust in the face of mobility and channel fading. In this paper, we examine the impact of synchronization on wireless M^3 network performance. First, we introduce MACA/PR, an asynchronous network based on the collision avoidance MAC scheme employed in the IEEE 802.11 standard. Then, we evaluate and compare several wireless packet networks ranging from the total asynchronous PRNET to the synchronized cluster TDMA network. We examine the tradeoffs between time synchronization and performance in various traffic and mobility environments.

1. INTRODUCTION AND BACKGROUND

The advancement in wireless communications and portable computing technologies and the emergence of nomadic applications have recently generated a lot of interest in wireless network infrastructures which support multimedia services. Most of the nomadic computing applications today require single hop type connectivity to the wired network (Internet or ATM). Figure 1, for example, shows the cellular network which is commonly used as the wireless network architecture. A, B, C, and D are fixed base stations which are connected by a wired backbone. Nodes 1 through 8 are mobile nodes. Communications between two mobile nodes completely rely on the wired backbone and fixed base stations. A mobile node is only one hop away from a base station.

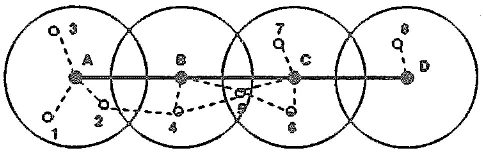

Figure 1: Conventional cellular networks (single-hop)

In parallel with (and separately from) the single hop cellular model, another type of model, based on radio to radio multihopping, has been evolving to serve a growing number of applications which rely on a fast deployable, multihop, wireless infrastructure. The classic examples are battlefield communications and (in the civilian sector) disaster recovery (fire, earthquake) and search and rescue. A recent addition to this set is the "ad hoc" personal communications network, which could be rapidly deployed on a campus, for example, to support collaborative computing and access to the Internet during special events (concerts, festivals etc). Multihopping through wireless repeaters strategically located on campus permits to reduce battery power and to increase network capacity (via spatial reuse). Interestingly, the multihop requirement may also arise in cellular networks. If a base station fails, a mobile node may not be able to access the wired network in a single hop. For example, in Figure 2, if base station B fails, node 4 must access base stations A or C through node 2 or node 5 which act as wireless multihop repeaters.

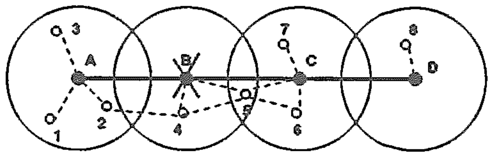

Figure 2: A multihop situation occurs when base station B fails.

In this paper, we are addressing the multihop type model and applications. More precisely, we are concerned with the design of efficient Multihop, Mobile, Multimedia (M^3) wireless networks. The M^3 problem has been recognized as a very difficult problem [1]. Over a decade ago, the ARPA sponsored Packet Radio Network [12] did provide an efficient solution to the Multihop, Mobile requirements of battlefield and disaster relief communications. It fell short, however, of suppoiting Multimedia services. An attempt was actually made to support voice using an ingenious routing technique called "duct routing", which forwards multiple copies of the same packet on multiple paths to overcome packet loss due to mobility. No QoS (Quality of Service) could be guaranteed, however. Consequently, voice quality rapidly degrades as the network load increases, as our results will later show. In spite of its limited multimedia service support, the PRNET was a very important contribution in that it provided a conceptually simple, distributed, reliable, totally asynchronous solution to the problem.

Recently, the M^3 problem was revisited under the ARPA sponsored WAMIS and GLOMO projects. In particular, at UCLA the Cluster TDMA scheme was developed [5]. In this scheme, the network is dynamically partitioned into clusters where each cluster uses a different spreading code (DS-Spread Spectrum channel encoding is used). Clusters (with code separation) improve spatial reuse. They also make it easier to manage real-time connections, since each cluster can manage its own channel bandwidth. Among clusters, a common, globally synchronous slotted TDM frame is defined. Slots can be reserved (in a Fast Reservation, "soft state" mode) by real time traffic. Free slots are accessed by datagrams in a random access mode. QoS routing makes sure that calls are routed on paths with sufficient bandwidth. It also enforces call acceptance control. Besides Cluster TDM, two other network schemes were developed at UCLA, the Virtual Network and SWAN [1]. These latter schemes also assume time frame synchronization, but make a more aggressive use of the DS-SS encoding for Code Division Multiple Access (CDMA). Namely, by means of efficient, distributed power control techniques various connections (with different codes) share the same time slot. The real-time traffic handling performance of all the above schemes was shown to be very good. However, the practical implementation is non trivial because of the use of multiple codes (and the associated power control requirement) and, most importantly the need for global synchronization. In an attempt to alleviate these problems, another scheme was developed under the WAMIS program at UCLA, namely, the Cluster Token [10]. In Cluster Token the TDM access scheme is replaced by an implicit token scheme within each cluster. Furthermore, no synchronization is required across clusters. Yet, the use of different codes in different clusters is retained.

Synchronization and code separation facilitate the support of multimedia traffic and greatly improve throughput (because of spatial reuse). However, they increase the implementation complexity and cost. The question is whether multimedia service can be supported without synchronization and code separation. A partial answer can be found by studying the evolution of MAC protocols and network architectures in the wireless LAN domain. Wireless LANs differ from multihop packet radio networks in that they cover a smaller area (a room or floor), have higher bandwidth, and typically have single hop connectivity to a base station connected to the wired network. However, some of the WLAN protocols can be extended to the campus/metro multihop environment addressed in this paper.

Traditionally, WLANs have used asynchronous random access protocols. The most popular version of asynchronous access protocol is CSMA (Carrier Sense Multiple Access). Carrier sense attempts to avoid collisions by testing the signal strength at the transmitter side. However, collisions occur at the receiver, not the transmitter. Thus, carrier sense does not provide all the information necessary for collision avoidance. This leads to one of the major causes of inefficiency in multihop CSMA networks (including PRNET), namely, the "hidden terminal" problem. Consider the configuration shown in Figure 3, station A and C can not hear each other. Thus, A is "hidden" from C. When A is transmitting a packet to B, C can not sense the transmission from A. Thus, it may also transmit a packet to B and cause a collision at B.

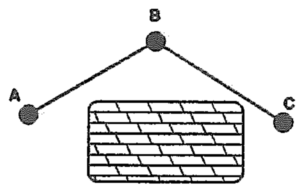

Figure 3: A is hidden from C

To overcome this problem, MACA (Multihop Access Collision Avoidance) [8] attempts to detect collisions caused by hidden terminals by establishing an RTS-CTS dialogue. The RTS-CTS dialogue can be used as the building block to eliminate the hidden terminal problem. However, it still does not resolve all of the hidden terminal problems. If one of the stations within range of CTS does not hear the CTS, it may not honor the CTS and later may cause a collision at the receiver. In the hidden terminal scenario in Figure 4, first, assume S is sending RTS to D at time t_1. Upon receiving RTS, D sends CTS to S at time t_2. Typically, CTS prevents collisions from hidden terminals like D_1. However, assume that S_1 also transmits RTS to D_1 exactly at time t_2. CTS and RTS collide at D_1. D_1 will not receive either RTS from S_1 or CTS from D. Thus, D_1 has not learned of the CTS from D and may later transmit an RTS, which collides at D with the impending transmission from S.

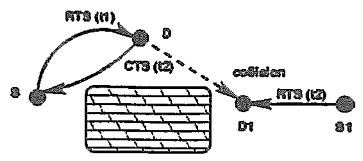

Figure 4: The collision due to the hidden terminal problem

MACAW [3], a variant of MACA, alleviates the residual hidden terminal problem and improves the performance of MACA. In MACA, a hidden terminal collision at link level must be recovered by the transport layer with severe performance degradation due to transport level flow control dynamics. MACAW does not prevent, but provides faster recovery from hidden terminal collisions. MACAW uses rapid link level ACKs for datagram retransmission and thus reduces the hidden terminal degradation. Namely, the receiver immediately ACKs the successful packet. Failure to receive the ACK (because, for example, the packet was damaged by a collision as in Figure 4 scenario) will prompt a retransmission after short timeout within the link level. FAMA [4] is a further refinement of MACAW. It includes non-persistent CSMA at the beginning of each free slot to prevent repeated collisions.

The IEEE 802.11 standard for the wireless LAN physical and MAC layers [7] includes the collision avoidance features of MACA and MACAW. A simplified version of the IEEE 802.11 standard is currently implemented by most wireless LAN vendors (WaveLan, Proxim, etc.). The fundamental access method in IEEE 802.11 is CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). In addition, all directed traffic uses positive acknowledgments (ACK) where retransmission is immediately scheduled by the sender if no ACK is received.

The CSMA/CA protocol as implemented in current WLANs has many desirable properties. It is asynchronous; it is fully distributed; it eliminates the hidden terminal problem; it provides good channel efficiency. It does not, however, provide real-time traffic support. To overcome this limitation, we are proposing MACA/PR (Multiple Access Collision Avoidance with Piggyback Reservations), a network architecture which is based on CSMA/CA and combines the asynchronous operation of WLANs and the QoS support of traditional TDM based network architecture. MACA/PR uses CSMA/CA as a MAC layer. At the network layer, it is equipped with a bandwidth reservation mechanism and a QoS routing protocol which are inspired to our earlier work on Cluster TDMA [5] and Cluster Token [10].

In the following sections we introduce and describe the MACA/PR architecture and its key components, namely, the reservation scheme and the QoS routing algorithm. Then, we evaluate via simulation MACA/PR and compare it with other packet radio architectures (synchronous and asynchronous) in order to assess the impact of time synchronization on performance.

2. MACA/PR

MACA/PR is an extension of IEEE 802.11 [7] and FAMA [4], which provides guaranteed bandwidth support (via reservation) to real-time traffic. Strictly speaking, MACA/PR is a medium access protocol and, as such, it permits us to establish real time connections over a single hop only (i.e. one link). However, by complementing MACA/PR with a QoS routing algorithm and a fast connection setup mechanism, we obtain a wireless protocol suite which supports routing and end-to-end multimedia connectivity in a mobile, multihop network.

The key components of the MACA/PR architecture are:

(a) the MAC protocol for the transmission of data packets.

(b) the reservation protocol for setting up real-time connections, and

(c) the QoS routing algorithm.

For datagrams, the MAC layer protocol is basically the same as MACAW [3] and IEEE 802.11 [7]. It also includes the provision of CSMA with non-persistent transmit request, proposed in FAMA [4], in order to prevent repeated collisions at the beginning of a free "slot" and thus improve performance at heavy load. Namely, a station with a datagram packet to send must first wait for a free "window" in the reservation table (as later definition). It then waits for an additional random time (on the order of a single hop round trip delay), after which it senses the channel. If the channel is free, it proceeds with the sequence <RTS, CTS, PKT, ACK>, as described in section 1. If the channel is busy, it waits until the channel becomes idle and repeats the procedure. This protocol prevents most (but not all) hidden terminal collisions. The ACK mechanism, however, provides rapid and reliable recovery in case of such collisions.

The MAC layer used for the transmission of real time packets is slightly different from the datagram MAC version. This is because real time packets are not retransmitted (after collision) by MACA/PR. Furthermore, packets and ACKs carry the real time scheduling information in the headers. As described in the next section in more detail, real time packets are protected from hidden terminal collision not by the <RTS, CTS> dialog, but by the propagation and maintenance of Reservation Tables (RTs) among neighbors. Transient collisions may occur when stations move and the RTs are not updated fast enough. When the RTs have stabilized, however, each station will transmit only in the permissible windows.

In the following sections, we describe the other two innovative components of MACA/PR, the reservation protocol and the QoS routing algorithm.

3. PIGGYBACK RESERVATION PROTOCOL

A real-time connection is set up using a fast reservation approach. Namely, we assume that real-time packets arrive at constant time intervals. The first data packet in the multimedia stream makes the reservations along the path. Once the first data packet is accepted on a link, a transmission window is reserved (on that link) at appropriate time intervals for all the subsequent packets in the connection. The window is released when idle for a prespecified number of cycle. Conceptually, this scheme is an extension of PRMA (Packet Reservation Multiple Access) to a multihop, unslotted environment.

In order to transmit the first packet successfully on each link, the source node initiates an RTS-CTS (Request To Send - Clear To Send) dialogue. On receiving CTS, the sender transmits the data packet. Both RTS and CTS specify how long the data packet will be. On receiving the data packet, the intended receiver returns an ACK if it received the data packet correctly. If the ACK is not received correctly or not received at all, then RTS is retransmitted, thus, starting the cycle all over again.

Let $CYCLE$ be the maximum interval allowed between two real-time packets. The sender schedules its next transmission after a time $CYCLE$ following the current data transmission and piggybacks the reservation in the current data packet. The intended receiver enters the reservation in its reservation table and confirms it in the ACK

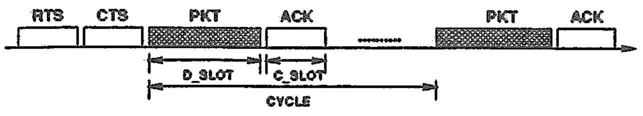

Figure 5: The MACA/PR protocol (RTS-CTS-PKT-ACK-----PKT-ACK)

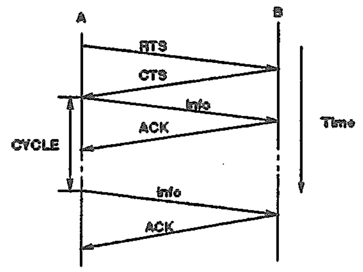

Figure 6: The connection setup on link (A, B) by MACA/PR

returned to the sender. The neighbors which hear the data packet are blocked and avoid colliding with the returning ACK. Furthermore, they learn about the next packet transmission time. Likewise, the neighbors of the receiver which hear the ACK will avoid transmitting at the time when the receiver is scheduled to receive the next data packet. Figure 5 illustrates the MACA/PR dialog. Figure 6 shows the timing of the connection setup on link (A, B). Notice that the RTS-CTS exchange is used only in the first packet transmission to set up the reservations. The subsequent data packets do not require the RTS-CTS exchange. Figure 7 shows how data and ACK packets announce the next transmission schedule to the neighbors of the sender and the receiver respectively. Note that for an ongoing real-time session the ACK serves the purpose of renewing the reservation rather than recovering from packet loss (via retransmission). In fact, if the ACK is not received, the packet is not retransmitted except for the first packet. Moreover, failure to receive N consecutive ACKs (where N is an adjustable parameter ≥ 2) is interpreted by the sender as a failure of the reserved "slot" (i.e. either because of unexpected interference on the next hop, or because of path failure). Thus, the sender, after N consecutive ACK failures, restarts the connection with the RTS-CTS dialog on another slot on the same link, or, in case of path failure, on another path, as later described.

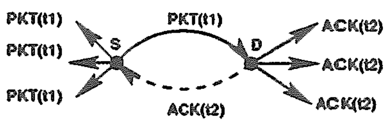

Figure 7: The data packet and ACK packet are used for real-time transmission

Any station near the sender which hears an RTS will defer long enough so that the sender can receive the returning CTS. Any station near the receiver which hears the CTS will avoid collision with the following data packet. The ACK following the packet contains the next transmission time. Any station which hears the ACK will avoid collision with the next scheduled transmission. The "reservation" ACK, albeit a source of overhead itself, is necessary in order to "protect" the receive window of the receiver during the next transmission cycle, and to alter the transmitter when something has gone wrong along the path (i.e. the next node has moved out of range). The above functions are also supported in the background by the reservation table exchange, as later discussed. However, the ACK provides a much more rapid notification.

To implement the reservation scheme, each node has a reservation table (RT) which keeps track of transmit and receive reserved windows (of any station within range). Upon hearing a real-time packet or an ACK, a station reads the next transmit time from the header and records it in its RT. The RT is used to avoid conflicts with ongoing reservations. Figure 6 shows an example topology, with ongoing real time connections shown by arrows. Figure 7 shows the corresponding RTs and schedules for node 1 and 6. For simplicity of explanation, we consider here the slotted case. That is, packets are of fixed size and windows are time synchronized (i.e. slots). A slot is large enough to contain the sequence <RTS-CTS-PKT-ACK>. We also assume uniform cycle time (*CYCLE*). The general algorithm applies to the unslotted case with multiple cycle times (i.e. sub-rate stream). Figure 9 reports for each slot the reservation activity recorded by the node. The format is (node, tx/rcv). For example, for the connection 5 → 1 node 6 hears both data packet from 5 to 1 and ACK from 1 to 5 (in the 2nd slot). Thus, it records {5 tx} and {1 rcv} in that slot. For connection 7 → 2, node 6 hears only the ACK issued by 2. Thus, it records {2 rcv} in the 5th slot.

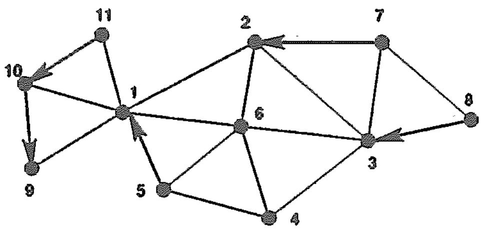

Figure 8: Real time connections in the current topology

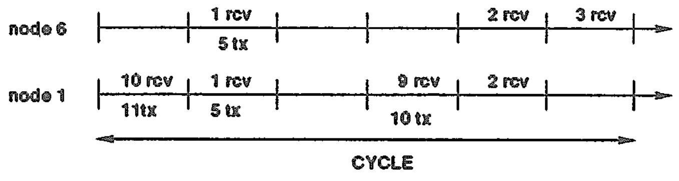

Figure 9: Reservation schedules at node 6 and 1

Suppose node 6 wants to transmit a datagram or set up a new connection to node 1. Node 6 must inspect the RTs (at both 6 and 1) in order to find a feasible (i.e. conflict free) transmission slot. A slot is feasible if no other station in the RT 6 is already receiving on it (in our example, slots 1, 3, 4), and no other station in RT 1 is transmitting on it (in our example, slots 3, 5, 6). In our case, the only feasible slot is slot #3. The proper bookkeeping of the feasible slots (or, in the unslotted case, of the feasible windows) requires that the neighbors periodically exchange RTs. For example, node 6 copies the (2 rcv) and (3 rcv) entries from the RTs received from node 2 and 3 respectively. Note that the RT dissemination among neighbors automatically overcomes the hidden terminal problem. For example, node 6 does not hear the transmission from 7 (hidden terminal) to 2. However, it learns about such reservation (on slot 5) from node 2 and node 3 RTs. The RT exchange is done in conjunction with routing table exchange. Reservations are refreshed every cycle. If reservation is not refreshed for N consecutive cycle, it is dropped.

MACA/PR relies on symmetry. In fact, the role of RTS is to elicit CTS from the receiver. The neighbors upon receiving CTS, learn that they are in range and thus could collide with the impending transmission. If station A can not hear station B's CTS then it is assumed that it can not interfere in any way with B (i.e. the link is bidirectional). Likewise, when the ACK is sent from the receiver, symmetry assures that every station capable of colliding with the next data transmission will honor the reservation (this is what alleviates the hidden terminal problem).

Datagram and real-time packet transmissions can be interleaved at each node, with priority given to real-time traffic. For real-time traffic, the system operates as a TDM system with a cycle time *CYCLE*. Datagram sources fill in the empty windows in the cycle. Datagram packets become backlogged when real-time traffic starts building up. To avoid datagram traffic lockout, a limit must be imposed on the maximum number of reservation in a cycle.

To transmit a datagram packet or the first packet in a real-time session, a station first checks its reservation table to select a free time interval in which to transmit the RTS packet. Then, it listens to the channel to make sure no nearby station is transmitting. After transmitting RTS, if the sender does not receive a CTS in response from the receiver, it will eventually time out, assume a collision occurred, and schedule the packet for retransmission. MACA/PR uses the binary exponential backoff algorithm to select the retransmission timeout.

When a new station is added to the system, it remains in listening mode for a period of time long enough to learn about all current reservation (i.e., in the order of a cycle time) and to receive the reservation tables from all the neighbors.

4. QoS ROUTING

Multimedia applications such as digital audio and video have much more stringent QoS requirements than traditional datagram applications. For a network to deliver QoS guarantees, it must make resource reservation and carry out resource control.

Routing is the first step in resource reservation. The routing protocol must first find a path that has sufficient resources to meet the application requirements. The resource setup protocol can then start hop-by-hop negotiation and setup along the path. Thus, the success or failure of a reservation request, depends on whether the routing protocol can find a suitable path.

In our applications the resource to be reserved is bandwidth, or, more precisely, a slot within the *CYCLE* frame. Thus, the goal of the bandwidth routing algorithm is to find the shortest possible path with adequate bandwidth to support the connection at hand. Since we have assumed that all real-time connections are uniform, i.e. require the same bandwidth, then the problem reduces to computing the shortest path such that the free bandwidth is above the minimum requirement.

To compute the "bandwidth" constrained shortest path, we start by a "loop-free" distance vector routing algorithm. In our simulation we have used the DSDV (Destination Sequenced Distance Vector) routing algorithm [11] which was proven to be loop-free. Loop freedom follows from the fact that the updates generated by a destination are sequentially numbered. Based on sequence numbers, a node uses consistent updates to computer a loop free path to a destination. Other loop-free algorithms could also be used. In our shortest path computation, the weight of each link is equal to 1 (i.e. minimal hop distance routing).

Our main contribution to QoS routing is the introduction of the bandwidth constraint [13]. In its most complete formulation, the bandwidth routing algorithm keeps track of shortest paths for all bandwidth values [13]. To compute these paths, each node periodically broadcasts to its neighbors the {bandwidth, hop distance} pairs for the preferred paths (one per bandwidth value) to each destination. In our case the number of preferred path is the maximum number of slots (or packets) in a cycle. In our experiments, we have simplified the implementation by keeping track of only two bandwidth (i.e. slot) values: the bandwidth on the shortest path, and the maximum bandwidth (over all possible paths). If a node receives a real time packet with a bandwidth request (determined from its cycle time) which can not be satisfied by the currently available paths to the intended destination, it drops the packet without ACK. Eventually, the sender will reroute the call on another path.

The link bandwidth information (for the bandwidth constraint computation) can be easily obtained from the reservation table. Consider for example the network in Figure 8. We have already discovered that if node 6 wants to transmit to node 1, it can do so only in slot 3. Thus, the bandwidth is 1 slot over the entire cycle. In the unslotted case, the bandwidth is the number of packets which can fit in the available windows. If a link has no bandwidth or, more generally, the bandwidth is below a predefined threshold, its weight is set to ∞. So no new connection is allowed through this node until an old connection releases some bandwidth. Due to the bandwidth constraint, the real-time traffic load is evenly distributed among the network avoiding hot spots.

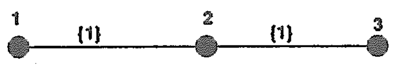

Figure 10: Available bandwidth example

One may expect that the bandwidth available on a path is the minimum of the link bandwidth values on the path as it is the case in wired networks. This, however, is generally not the case in wireless networks. Consider the example in Figure 10. Assume that slot #1 is available for transmission 1 → 2, and again slot #1 is available for transmission 2 → 3. Since in a wireless network node 2 can not receive and transmit simultaneously in slot #1, the available bandwidth is zero, i.e., different from the minimum over (1, 2) and (2, 3), which is 1 slot. Conceptually, this feature renders the problem of finding the shortest path with bandwidth constraints an NP complete problem. A heuristic approach has been developed, which is very efficient and provides guaranteed feasible suboptimal solutions.

While real-time packets are routed using "bandwidth constrained" paths, datagram packets are routed using shortest paths (without constraints). Note that these two sets of paths may differ. Thus, the routing algorithm consists of two components: the shortest path algorithm, and the "bandwidth constraint" algorithm.

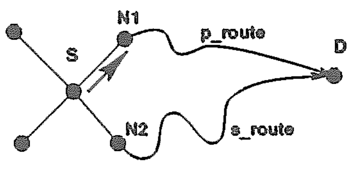

Figure 11: Standby routing

In a mobile environment, it is essential to quickly reconstruct the connection for an existing real-time session when a topological change destroys the current route. We propose to maintain "standby" paths which permit easy and quick rerouting after failure/change. In Figure 11, each node uses the primary path to route its packets. When the link fails, the standby path becomes the primary path, and another new standby path will be reconstructed. This is shown in Figure 12.

The primary path is the shortest path with or without bandwidth constraint depending on whether we are routing real-time packets or datagrams. The standby route is the shortest path with or without bandwidth constraint given that first link in the path differs from the primary path. In Figure 11 example, the shortest path goes through node N_1. The second shortest path goes through node N_2. If the link (S, N_1) fails, node N_2 takes over as next hop to destination D.

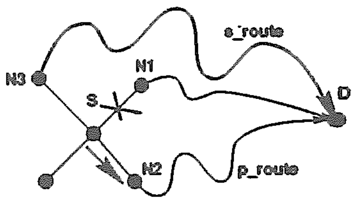

Figure 12: The primary route fails and the standby route becomes the primary route. Another standby route is constructed.

Note that the standby route computation requires no extra routing and control information exchange. In fact, the standby route is obtained directly from the same routing tables used for the shortest path computation.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0		x	x	x				x	x		x								x	x
1	x			x	x		x	x	x	x	x	x						x	x	
2	x			x					x		x								x	x
3	x	x	x						x	x	x								x	x
4		x				x		x		x				x		x	x	x		
5					x												x	x		
6		x								x		x			x					
7	x	x			x				x	x	x			x		x		x	x	
8	x	x	x	x				x		x	x								x	x
9		x		x	x		x	x	x		x	x						x	x	
10	x	x	x	x				x	x	x									x	x
11		x					x			x					x			x		
12														x		x				
13					x			x					x			x	x			
14							x					x						x		
15					x			x					x	x			x	x		
16						x	x							x		x		x		
17		x			x	x		x		x		x			x	x	x			
18	x	x	x	x				x	x	x	x									x
19	x		x	x					x		x								x	

Table 1: The system topology (N=20)

5. SIMULATION RESULTS

The MACA/PR architecture was simulated using the MAISIE simulation platform [9]. The simulator is based on an asynchronous unslotted channel access model. Several sets of experiments were carried out in order to evaluate the performance of MACA/PR as a function of traffic and system parameters, and to compare it with that of other schemes. Most of the simulation experiments share the same network layout and traffic pattern, which are described below.

The network under study consists of 20 nodes, randomly placed in an 100 × 100 feet square. Transmission range is 45 feet. The initial topology connectivity is shown in Table 1. The actual topology varies when nodes move.

The channel rate is 800 kbps (the nominal rate of the radio under development with the ARPA sponsored WAMIS project [1]). All data packets (datagram and real-time) are 4 kbits. Control packets (RTS, CTS, ACK) are 1 kbits. The preamble for DS-SS acquisition is 500 bits. Thus, data and control packet transmission times (at 800 kbps) are 6 ms and 2 ms respectively. A real-time packet transmission of the type <data, ACK> requires 8 ms. A datagram packet transaction of the type <RTS, CTS, data, ACK> requires 12 ms. The default *CYCLE* time is 100 ms. This implies that up to 8 datagram transmissions and up to 12 real-time connections for already established connections can be carried on a link.

The offered traffic consists of two components: real-time sessions and datagrams. In the real-time category, a new real-time session is generated on average every second (Poisson arrival model) between a random pair of nodes. Session duration is exponential with 3 minutes average. The rate on a real-time session can be varied by changing the cycle time. The default value is $CYCLE = 100$ ms. We assume only one packet per session per cycle. The default rate value is one packet every 100 ms. At 4 kbits per packet, this corresponds to 40 kbps. The default rate turns out to be much higher than actual network capacity. Thus, most of VCs are rejected at call set up time because of lack of bandwidth (as advertised by the QoS routing algorithm).

Datagrams are also generated between random node pairs, with average default interarrival time = 100 ms. This default value corresponds to very light traffic. Datagrams have lower priority than real-time packets on the transmission queue. In order to prevent lock out of datagrams, a fraction α of the $CYCLE$ time is reserved for datagrams. The default value is $\alpha = 0$.

5.1. MACA/PR: Real-time and Datagram Traffic Mix

In the first experiment, we consider the effect of variable real-time load on datagram throughput in MACA/PR. Namely, the real-time rate is allowed to vary from ∞ (i.e. 0 pkt/sec) to 100 ms (i.e. 10 pkts/sec). Default values were used for datagram traffic (i.e. $\alpha = 0$ and $CYCLE = 100$ ms). Figure 13 reports datagram and real-time throughput for varying traffic load. The datagram throughput is the aggregate throughput (sum over all source/destination pairs). The real-time throughput is the average throughput over a session. We note that, because of the priority given to real-time traffic, and because of the $\alpha = 0$ threshold assumption. the datagram throughput decreases very rapidly as the real-time traffic increases. This points out the importance of a careful selection of the threshold α for fair bandwidth sharing.

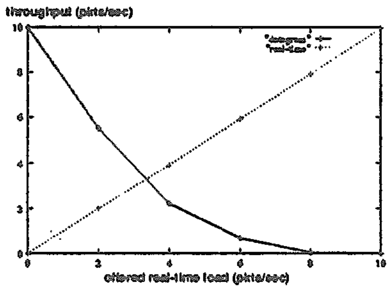

Figure 13: The throughput of mix traffic

5.2. MACA/PR: Standby Routing

A second set of experiments evaluates MACA/PR is a mobile environment with heavy real-time load ($CYCLE = 100$). Due to mobility, real-time packets may be dropped on already established connections during path changes. The goal of this experiment is to access the improvement introduced by the "standby" routing feature, i.e. the availability of an alternate route in case the preferred route fails. This feature is of critical importance when stations are mobile. In particular, it is expected to reduce loss rate. Two experiments were run, with speed 2 feet/sec and 8 feet/sec respectively. Table 2 and Table 3 show the results. Standby routing reduces packet loss by more 50% in bath cases. Note that no extra control traffic overhead is introduced, since the standby route is computed from the same tables used for the primary route computation. Furthermore, if the primary route is loop-free, the standby route is loop-free as well.

5.3. MACA/PR vs Cluster TDMA and Cluster Token

We proceed now to compare MACA/PR with other schemes which also support multimedia, mobile applications. The first two candidates are Cluster TDMA [5] and Cluster Token [10]. For Cluster TDMA, the TDM frame is 100 ms, with 20 control minislots (2 ms for each) and 10 datagram/real-time slots (6 ms for each). For Cluster Token, cycle time is again 100 ms, with datagram/real-time packet of 6 ms. As before, in this experiment we only evaluate the real-time traffic performance ($CYCLE = 100$). Furthermore, in addition to aggregate performance (averaged over all sessions) we also monitor the performance on two specific connections (namely, $16 \rightarrow 3$ and $19 \rightarrow 12$). The initial layout of both connections is shown in Figure 14, and is consistent with Table 1. Both connections are set up at the start of the simulation, and remain active throughout the simulation run (typically, 180 seconds simulation time). The rationale is to study the interference between two connections which run in proximity of each other (without sharing the same links, however). In this case, there are plenty of hidden terminal collision opportunities, as shown in Figure 14.

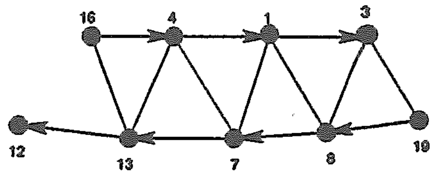

Figure 14: Two time-bounded connections in the current topology

The results, reported in Table 4, 5 and 6, show the performance of various algorithms in a static situation. We note that the performance for individual connections is comparable. No packet are lost in any scheme. The full throughput (about 1 packet per cycle, i.e., 10 pkts/sec) is achieved for the connections in each scheme. End-to-end delay is higher for Cluster TDMA because of the 100 ms TDM frame latency at each hop. In MACA/PR the end-to-end delay is lower since there is no waiting for a new frame at each intermediate node. Rather, the next free slot is immediately acquired. In the Cluster Token, the delay is lower than in Cluster TDMA because the latency of token cycle is lower than that of the TDMA cycle. The aggregate throughput (sum of all VCs) is considerably higher for Cluster TDMA and Cluster Token than MACA/PR, as expected. This is due to various reasons: (a) random position of MACA/PR real-time packets in the cycle (as opposed to the strict slot organization in Cluster TDMA and Cluster Token) which causes bandwidth fragmentation, and (b) less spatial reuse efficiency since the same code is used everywhere (while Cluster TDMA and Cluster Token apply code separation among clusters). Basically, the reduced overall throughput is the price we pay for giving up global synchronization and multi-code operation.

5.4. MACA/PR vs PRNET

Next, we compare MACA/PR with a pure asynchronous scheme, namely PRNET [12]. In this case, we only monitor the performance on the two real-time connections shown in Figure 14. The PRNET simulator is a very faithful model of the actual PRNET implementation [12], featuring CSMA sensing, pacing, echo ACKs and alternate routing. For real-time traffic support in PRNET, duct routing [12] is used. For consistency, packet transmission time in PRNET is assumed to be 6 ms. Static results are reported in Table 7 to 11 and mobile results (speed = 2 and 8 feet/sec; cycle = 100 ms) are reported in Table 12 and Table 13 respectively. In the static experiments, the rate on a real-time connection is progressively increased from 1 pkt/100 ms to 1 pkt/64 ms. The rate increase is accomplished by reducing *CYCLE* time correspondingly.

We note that MACA/PR performs very well, with full rate on both connections for all cycle values. In comparison, PRNET performance is rather poor. Packet loss rate is very high, and it increases as the per connection data rate increases (as expected). Furthermore, the throughput performance on the two connections is asymmetric, indicating that some form of "capture" effect takes place. Interestingly, the sum of the throughputs on the two connections remains about constant while the individual throughputs fluctuate.

In the mobile environment, MACA/PR performs quite well (with packet loss < 3%), while PRNET degrades very rapidly as speed increases (the sum of the two throughputs drops from 9 pkts/sec to 6 pkts/sec). In summary, poor performance of PRNET can be attributed to the following reasons: (a) hidden terminal effect; (b) lack of call acceptance control (no QoS routing); (c) lack of bandwidth reservation; (d) traffic overhead caused by duct routing.

5.5. Scheme Comparison Synopsis

Table 14 and Table 15 summary the comparison of various schemes examined so far. In this experiment, *CYCLE* = 100. The schemes are ordered by increasing implementation complexity (from the total asynchronous PRNET to the highly organized Cluster TDMA). Both throughput per connection and aggregate throughput are compared. Cluster Token performance is very similar to that of Cluster TDMA. This is quite of interest, since Cluster Token uses code separation but does not require global synchronization (it only requires intra cluster synchronization). Performance is only moderately degraded by speed increase. MACA/PR appears to provide an attractive compromise between throughput efficiency and simplicity of implementation. On the positive side, MACA/PR exhibits the lowest end-to-end delay (which is critical for some real-time applications such as voice). On the negative side, MACA/PR can not achieve the aggregate throughput efficiency of the Cluster type solutions (which benefit from code separation).

6. CONCLUSIONS

MACA/PR is a novel wireless architecture for multimedia support which requires minimal synchronization. It extends the "collision avoidance" access scheme earlier introduced by MACA [8] and later refined in MACAW [3] and FAMA [4] to multihop, multimedia environments without giving up the simplicity of the asynchronous operation. This is achieved with two innovative features: "flexible" reservations within a cycle (as oppose slotted reservation as in TDMA schemes), and QoS loop-free routing. A third innovative feature, "standby" routing, enhances performance in the mobile environment.

Performance results show that MACA/PR strikes a good compromise between the totally asynchronous, unstructured PRNET and the highly organized Cluster TDMA. Major positive features are: low latency, low packet loss, good acceptance control, fair bandwidth sharing among connections, good mobility handling, good scaling properties. From the implementation stand point, the simplicity of the protocol (unburdened by synchronization requirements) and the affinity to the IEEE 802.11 standard (with availability of off-the-shelf radio components) promise low implementation costs.

In summary, MACA/PR is a cost effective compromise between PRNET and Cluster type solutions. Further enhancements are currently explored including: embedded coding to permit selective rate reduction; datagram congestion control; fair sharing between datagram and real-time traffic; multi-rate real-time connection support.

REFERENCES

[1] A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Kleinrock, J. Short and J. Villasenor, "Adaptive Mobile Multimedia Networks," *IEEE Personal Communications*, pp. 34-51, April 1996.

[2] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1992.

[3] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LAN's," *Proceedings of ACM SIGCOMM '94*, pp. 212-225, 1994.

[4] C.L. Fullmer and J.J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks," *Proceedings of ACM SIGCOMM '95*, 1995.

[5] M. Gerla and J.T.-C. Tsai, "Multicluster, Mobile, Multimedia Radio Network," *ACM-Baltzer Journal of Wireless Networks*, Vol. 1, No. 3, pp. 255-265 (1995).

[6] D.J. Goodman, R.A. Valenzuela, K.T. Gayliard, and B. Ramamurthi, "Packet Reservation Multiple Access for Local Wireless Communications," *IEEE Transactions on Communications*, pp. 885-890, August 1989.

[7] The editors of IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*

[8] P. Karn, "MACA - a New Channel Access Method for Packet Radio," in *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134-140, ARRL, 1990.

[9] R. Bagrodia and W. Liao, "Maisie: A Language for the Design of Efficient Discrete-Event Simulations", *IEEE Transaction on Software Engineering*, pp.225-38, 1994.

[10] C.-H. Lin, *A Multihop Adaptive Mobile Multimedia Network: Architecture and Protocols*, Ph.D. dissertation, Computer Science Department, University of California, Los Angeles, 1996.

[11] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *Proceedings of ACM SIGCOMM '94*, pp. 234-244, 1994.

[12] N. Shacham, E.J. Craighill and A.A. Poggio, "Speech Transport in Packet-Radio Networks with Mobile Nodes," *IEEE Journal on Selected Areas in Communications*, pp. 1084-1097, Dec. 1983.

[13] M. Gerla, "Routing and Flow Control in ISDN's," *ICCC'86*, pp. 643-647, 1986.

	with standby routing	without standby routing
Pkts lost	19 (1.06%)	42 (2.34%)
End-to-End Delay (msec)	74.94	320.45
Throughput (pkts/sec)	9.89	9.72

Table 2: The performance of standby routing (maximum speed = 2 feet/sec)

	with standby routing	without standby routing
Pkts lost	49 (2.73%)	135 (7.52%)
End-to-End Delay (msec)	78.47	654.45
Throughput (pkts/sec)	9.73	9.24

Table 3: The performance of standby routing (maximum speed = 8 feet/sec)

Connection	19 → 12	16 → 3	total of all connections
pkts received	1796	1797	21199
pkts lost	0	0	0
throughput (pkts/sec)	9.98	9.98	117.77
end-to-end delay (msec)	400.00	300.00	97/hop

Table 4: Performance of Cluster TDMA (N=20, 10 slots in data phase, frame size = 100 ms)

Connection	19 → 12	16 → 3	total of all connections
pkts received	1794	1792	20954
pkts lost	0	0	0
throughput (pkts/sec)	9.96	9.95	116.41
end-to-end delay (msec)	254.63	232.46	85/hop

Table 5: Performance of Cluster Token (N=20, CYCLE=100 ms)

Connection	19 → 12	16 → 3	total of all connections
pkts received	1799	1800	6322
pkts lost	0	0	0
throughput (pkts/sec)	9.99	10.00	35.12
end-to-end delay (ms)	85.00	48.00	22/hop

Table 6: The performance of MACA/PR (CYCLE=100 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	1799	34	1800	1566
pkts lost	0	1765 (98.11%)	0	233 (12.95%)
throughput (pkts/sec)	9.99	0.19	10.00	8.7
end-to-end delay (ms)	84.56	80.45	45.34	65.71

Table 7: The performance of MACA/PR and PRNET (CYCLE=100ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	2045	24	2045	1632
pkts lost	0	2016 (98.82%)	0	408 (20%)
throughput (pkts/sec)	11.36	0.13	11.36	9.07
end-to-end delay (ms)	85.00	79.08	48.00	67.01

Table 8: The performance of MACA/PR and PRNET (CYCLE=88 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	2249	712	2249	1096
pkts lost	0	1532 (68.27%)	0	1148 (51.16%)
throughput (pkts/sec)	12.49	3.96	12.49	6.09
end-to-end delay (ms)	103.00	383.07	48.00	225.92

Table 9: The performance of MACA/PR and PRNET (CYCLE=80 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	2499	1243	2499	711
pkts lost	0	1251 (50.16%)	0	1783 (71.49%)
throughput (pkts/sec)	13.88	6.91	13.88	3.95
end-to-end delay (ms)	95.00	283.68	49.00	663.26

Table 10: The performance of MACA/PR and PRNET (CYCLE=72 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	2810	624	2811	1129
pkts lost	1	2181 (77.75%)	1	1676 (59.75%)
throughput (pkts/sec)	15.61	3.47	15.62	6.27
end-to-end delay (ms)	113.00	560.96	49.00	333.25

Table 11: The performance of MACA/PR and PRNET (CYCLE=64 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	1783	616	1778	1019
pkts lost	16 (0.89%)	1179 (65.68%)	22 (1.22%)	776 (43.23%)
throughput (pkts/sec)	9.91	3.42	9.88	5.66
end-to-end delay (ms)	89.34	160.96	60.54	231.65

Table 12: The performance of MACA/PR and PRNET (maximum speed=2 feet/sec, CYCLE=100 ms)

	Connection 1 (19 → 12)		Connection 2 (16 → 3)	
	MACA/PR	PRNET	MACA/PR	PRNET
pkts received	1745	325	1757	829
pkts lost	55 (3.06%)	1470 (77.75%)	43 (2.39%)	966 (59.75%)
throughput (pkts/sec)	9.69	1.81	9.76	4.61
end-to-end delay (ms)	93.37	576.95	63.56	352.34

Table 13: The performance of MACA/PR and PRNET (maximum speed=8 feet/sec, CYCLE=100 ms)

	PRNET	MACA/PR	Cluster Token	Cluster TDMA
avg VC throughput (pkts/sec)	4.2	9.68	9.89	9.98
total VC throughput (pkts/sec)	15.54	34.65	97.34	107.89
end-to-end delay (ms)	72.02	48.20	254.23	298.15
avg pkt loss per VC	67.70%	3.15%	1.07%	0.01%

Table 14: Overall performance comparison (2 feet/sec)

	PRNET	MACA/PR	Cluster Token	Cluster TDMA
avg VC throughput (pkts/sec)	4.6	9.69	9.88	9.97
total VC throughput (pkts/sec)	14.54	34.45	95.34	107.70
end-to-end delay (ms)	73.05	66.50	298.80	302.13
avg pkt loss per VC	67.30%	3.07%	1.18%	0.02%

Table 15: Overall performance comparison (8 feet/sec)