

TECHNICAL RESEARCH REPORT

A New Protocol for Scheduling TDMA Transmissions in Mobile Ad Hoc Networks

by Chenxi Zhu, M. Scott Corson

**CSHCN TR 2001-19
(ISR TR 2001-33)**



The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site <http://www.isr.umd.edu/CSHCN/>

A new protocol for scheduling TDMA transmissions in mobile ad hoc networks

Chenxi Zhu and M. Scott Corson
 Institute for Systems Research, University of Maryland
 College Park, Maryland 20742
 (czhu,corson@isr.umd.edu)

Abstract— We present a new protocol for scheduling TDMA transmissions in a mobile ad hoc network. In this protocol, nodes may reserve time slots for unicast, multicast or broadcast transmissions. The protocol uses contention-based access between nodes to reserve transmission time slots, and its operation is distributed and concurrent. The protocol is independent of network size (measured in the number of nodes), and can be used in large or dynamic networks. The protocol’s performance is studied via simulation and compared with the IEEE 802.11 protocol.

I. INTRODUCTION

An important aspect of mobile ad hoc networks is medium access control (MAC). The IEEE 802.11 protocol [1] is based on CSMA/CA, whereby nodes acquire the channel using a contention-based RTS/CTS message exchange. Although it is widely used, its contention-based nature makes it difficult to reserve bandwidth, which is often desired by real-time multimedia traffic. Schedule-based MAC protocols, such as TDMA, are potentially better suited to meet these Quality of Service (QoS) requirements by reserving bandwidth and following the resultant transmission schedule. Generation of such a schedule with a distributed protocol is difficult, especially in a network whose topology and traffic pattern (bandwidth requirements) change frequently. TDMA scheduling in ad hoc networks (or packet radio networks) has been extensively studied. It is hard to produce the optimal schedule even with accurate, global information (NP-complete) [2], [3], [4], and some previous works have designed distributed protocols requiring only local information [3], [5], [6]. In these protocols, nodes follow a pre-defined order and make their slot reservations one after another. They are constrained by network size and work best in small networks. Some protocols developed recently ([7], [8], [9]) generate transmission schedules using contention as a means of signaling. Their scheduling overhead is insensitive to the network size, hence they are more flexible and can be used in large networks.

In this paper, we develop a new distributed protocol for generating and maintaining TDMA transmission sched-

ules which accommodate both a randomly-changing network topologies and a dynamic bandwidth requirement. The protocol uses contention when generating the schedules, hence its operation is not affected by the network size but only by the node density. Consequently it is scalable and can be used in large networks. In this protocol, termed the Evolutionary-TDMA scheduling protocol (E-TDMA), a node can reserve conflict-free time slots for transmission to one (unicast), or some (multicast) or all (broadcast) of its one-hop neighbors. The resulting schedule is a mixture of unicast, multicast and broadcast transmissions. The protocol deals with the frequent changes in the network topology and in the traffic pattern by frequently updating the current schedules in an incremental, or evolutionary, manner. The schedules can be updated in many parts of the network simultaneously, and a node only interacts with its neighbors when reserving time slots. By transmitting in reserved, conflict-free time slots, a node is guaranteed bandwidth and may enjoy better quality-of-service support than when using a contention-based protocol such as 802.11. Simulation is used to compare E-TDMA with 802.11, and shows that E-TDMA outperforms 802.11 for constant bit rate (CBR) traffic in terms of throughput and delay, especially under heavy load, in networks with low to moderate mobility.

II. EVOLUTIONARY-TDMA SCHEDULING PROTOCOL

The E-TDMA protocol allows nodes to assign TDMA transmission slots among themselves as network composition and bandwidth demands change. The protocol produces two TDMA schedules simultaneously, each used in a different portion of the same channel and for different purpose. The first schedule is a broadcast schedule in which every node is assigned one slot. This broadcast schedule is used for nodes to exchange information in the control frame and is called the control schedule (*ctrl_schedule*). The second schedule carries user generated traffic in the information frame, and is called the information schedule (*info_schedule*). All reservations here are *one-hop* reservations. In the *info_schedule*, a

node can reserve a different number of slots to transmit to one (unicast), or some (multicast), or all (broadcast) of its neighbors, depending on its need. Both the *ctrl_schedule* and the *info_schedule* reflect the topology of the network. Furthermore, as the network topology and the bandwidth requirements change, the schedules adjust accordingly to maintain conflict-free transmissions. The algorithm copes with changes in the network topology and bandwidth requirements in an incremental manner in order to minimize the re-scheduling overhead and to support QoS to the extent possible in these networks.

In the protocol, all nodes participate in the scheduling process on an equal basis. The scheduling process is executed across the entire network at the same time. Nodes do not wait in some particular order to schedule their transmissions. They determine who can reserve transmission slots by contending for a permission in their neighborhoods (called a temporary color), and many nodes can acquire this permission and schedule their transmissions simultaneously. This reduces protocol overhead and enhances robustness. Essentially, every node is responsible for its own transmission schedule. A node can reserve a conflict-free time slot to transmit to a set of its one-hop neighbors. If any of its receivers begin to suffer collisions caused by some topological change, the transmitter learns this from that receiver and stops transmission in the slot. It can reserve another time slot if it needs to. After a transmission is complete, the transmitter releases the slot, which can be reserved for another transmission. A node only needs to exchange information with its one-hop neighbors. Because of the local nature of the protocol, it is not sensitive to the network size. It is not affected by network partition either. It is suitable for a large network of changing size, such as a large, mobile military formation.

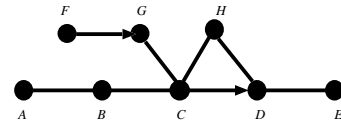
We make the following assumptions about the network:

- Nodes keep perfect timing. Accurate, global time is available to every node;
- Links are symmetric. The topology of the network can be represented by an undirected graph;
- The network topology changes slowly relative to packet transmission time;
- Every node is able to operate the Five Phase Reservation Protocol (FPRP)¹;
- The only reception error is caused by packet collision.

A. Notations used by E-TDMA

The activity of a node n_i in a given slot s is represented as a pair $(state(s), target(s))$, where $state(s)$ is the activity of this node in slot s , and $target$ is a set of one-

¹FPRP is a protocol with which a node can contend for and reserve a broadcast slot. Details of FPRP can be found in [7].



A: ($state(s) = Idle$)
 B: ($state(s) = Block_r$)
 C: ($state(s) = Trans, target(s) = D$)
 D: ($state(s) = Recv, target(s) = C$)
 E: ($state(s) = Block_t$)
 F: ($state(s) = Trans, target(s) = G$)
 G: ($state(s) = Collision$)
 H: ($state(H) = Block_{tr}$)

Fig. 1. Slot states defined by the E-TDMA protocol.

hop neighbors to/from which this node transmits/receives in slot s . An one-hop neighbor is sometimes simply called a neighbor. Due to the constraints required for conflict-free TDMA transmissions, the activity of a node n_i in a slot s can be classified into the following states:

1. *Trans*—node n_i transmits to a set of neighbors R : ($state(s) = Trans, target(s) = R$). If the transmission is a broadcast, $target(s) = Broadcast$;
2. *Recv*—node n_i receives from a neighbor n_j : ($state(s) = Recv, target(s) = n_j$). For this case $|target(s)|=1$;
 If node n_i is not transmitting or receiving in slot s , it is in one of the following states:
3. *Block_t*—node n_i is blocked from transmitting because at least one of its neighbors receives from another node, and none of its neighbors transmits: ($state(s) = Block_t$);
4. *Block_r*—node n_i is blocked from receiving because at least one neighbor is transmitting to another node, and none of its neighbors receives: ($state(s) = Block_r$);
5. *Block_{tr}*—node n_i is both blocked from transmitting because at least one neighbor is receiving, and blocked from receiving because at least another neighbor is transmitting: ($state(s) = Block_{tr}$);
6. *Collision*—node n_i is experiencing a collision when it is supposed to receive from a neighbor: ($state(s) = Collision$);
7. *Idle*—node n_i is idle when none of its neighbors transmits or receives in slot s : ($state(s) = Idle$).

Note that the *target* field is only defined for states *Trans* and *Recv*. For the other states it is not meaningful. These states are mutually exclusive, i.e. a node can be in only one of these states at a given time slot. Any slot when a node does not transmit can be called a passive slot. Figure 1 illustrates these different states. Suppose in a slot s a node C transmits to D and node F transmits to G (their transmissions to the intended receivers are shown with arrows). Note that these transmissions also reach other one-hop neighbors of the transmitters due to the broad-

cast medium (not shown). Node D can successfully receive from C , because C is its only neighbor transmitting in that slot. The transmission of C also reaches G and collides with the transmission from F . Node G suffers a collision in the slot. The states of other nodes (A, B, E, H) not in *Trans* or *Recv* are determined by their positions relative to the transmitters (F, C) and the intended receivers (D, G).

B. Frame structure of E-TDMA

The protocol operates within a single TDMA channel. The channel is partitioned into two epochs: a control epoch where the schedules are updated by the protocol, and an information epoch where user data transmission takes place. The two epochs are interleaved periodically. The frame structure of the protocol is defined in Figure 2. An information epoch has K information frames, and each consists of L information slots. In an information slot, a node transmits or receives a MAC data frame with its neighbors according to the *info_schedule*. How many slots a node needs in the *info_schedule* and to which neighbor(s) the transmission in a slot is addressed depends on the type and the amount of out-going traffic at this node, and this can be time-varying. E-TDMA accommodates these transmission requirements by updating the *info_schedule* periodically. The active *info_schedule* is updated in the preceding control epoch. A control epoch has two phases: a contention (C) phase and an allocation (A) phase. A contention phase is divided into N contention slots, each consisting of a number of Five Phase Reservation Protocol (FPRP) cycles. A contention slot corresponds to a temporary color (to be defined later), and if a node needs to acquire a temporary color, it contends using the FPRP protocol in the corresponding C slots. If successful, it reserves the temporary color for the current control epoch. An allocation phase has N frames. In an A frame, nodes exchange information with their one-hop neighbors by transmitting according to the *ctrl_schedule*. A transmission in the *ctrl_schedule* is a broadcast; hence the *ctrl_schedule* is a broadcast schedule. In the parlance of graph theory, the broadcast schedule corresponds to a distance-2 node coloring. For this reason a slot in the *ctrl_schedule* is also called a color. There are two types of colors in the *ctrl_schedule*: N temporary colors and M permanent colors. A node has at most one permanent color and one temporary color in the *ctrl_schedule*. Possessing a temporary color equates to having a permission to reserve new information slots or permanent colors. If a node needs to make a new reservation in a control epoch, it first needs to acquire permission. A temporary color becomes invalid after the control epoch in which it is granted, and if a node wants to make

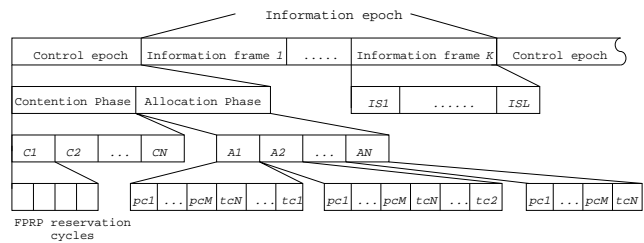


Fig. 2. Frame structure of E-TDMA. There are M permanent colors ($pc1, \dots, pcM$) and N temporary colors ($tc1, \dots, tcN$). There are K information frames in an information epoch and L information slots ($IS1, \dots, ISL$) in an information frame.

another reservation later it has to contend again using the FPRP. The permanent color of a node lasts much longer. A node needs a permanent color in the *ctrl_schedule* for exchanging its scheduling information with its neighbors (but not for making new reservations). Once a node acquires a permanent color, it transmits in every slot designated this color as long as its transmission does not collide with others. If a collision occurs due to some topological change, a node will discard its current permanent color and acquire a new one. How a node acquires its permanent color will be described later. Different A frames have different lengths. The first A frame ($A1$) has M slots corresponding to the M permanent colors and N slots corresponding to the N temporary colors. Slots corresponding to the temporary colors are placed after the permanent colors and are arranged in *reverse* order. The number of slots corresponding to the temporary colors decrements in each subsequent A frame. Temporary color $tc1$ only appears in $A1$, $tc2$ only in $A1$ and $A2$, and so on. The last A frame has only $M + 1$ slots corresponding to the M permanent colors and the temporary color tcN .

C. Details of E-TDMA

Every node generates and maintains its own schedules in collaboration with its neighbors. The schedule of the entire network is simply the collection of the schedules of all the nodes. No single node has global information such as the size, the membership or the schedules of the entire network. A node only directly interacts with its one-hop neighbors. For a given node, another node two hops away could cause interference if they both transmit in the same slot, but such interference or a collision takes place at a one-hop neighbor. By knowing the schedules of this one-hop neighbor, this node indirectly learns the relevant part of the schedules of this two-hop neighbor. This way it is able to avoid a collision.

For example, suppose nodes $n1$ and $n2$ are one-hop and

two-hop neighbors of a node n_0 , respectively. If n_2 reserves a slot to transmit to n_1 , by knowing that n_1 receives from n_2 in that slot, n_0 indirectly learns about the transmission of n_2 . It will avoid transmission in the same slot in order not to interfere. In this way the transmission information of the two-hop neighbors are embedded in the schedules of the one-hop neighbors, and nodes two hops apart do not need to communicate directly. A node does not need or have any information about nodes beyond its two hop range.

Nodes exchange their schedules periodically in the control epochs. A node keeps its neighbor information in a list NB . By tracking its neighbors and their schedules, it knows which slots are in use and hence which slots are available. It uses this information to reserve its new slots. Many nodes can reserve their transmission slots simultaneously. Because the set of nodes that reserve their slots at the same time is determined by the FPRP protocol, they are likely to be three or more hops apart. Thus they can reserve their time slots independently without causing collisions.

When the schedules are updated, they are always updated on the basis of the existing schedules. A reservation is only released when its transmission is complete, or when it suffers a collision, but one is never released to accommodate a new reservation. A new reservation can be made only if it does not conflict with any reservations established earlier. The resulting schedules evolve over time with the changing topology and traffic pattern. This gives E-TDMA protocol its name. In order to illustrate the incremental nature of the protocol, we describe below how it works in a scenario where a set of existing schedules (*ctrl_schedule* and *info_schedule*) have already been running in the network. It will be seen later that network initialization, where the old schedules are non-existent, is a trivial case of this scenario.

C.1 The Contention (*C*) Phase

The purpose of the contention phase is to assign the N temporary colors to the nodes. Remember temporary colors are permissions for reserving new slots in a control epoch. If a node needs to reserve a permanent color for its *ctrl_schedule* (if it does not already have one), or an information slot in the *info_schedule* (if it has new traffic arrival and requires more transmission bandwidth), it first contends in the *C* phase for a temporary color with the FPRP protocol. FPRP is a protocol for a node to reserve a broadcast slot with contention using a five-phase message exchange [7]. It ensures that only nodes three hops

apart or further can acquire the same temporary color². If successful, a node will transmit in slots corresponding to this temporary color (along with any slots corresponding to its permanent color) in the allocation phase of this control epoch. In one of these slots it will reserve the times slots it needs if available.

C.2 The Allocation Phase

In this phase a node transmits its current schedules in a schedule update frame (*su_frame*) in a slot designated to its permanent or temporary color, and listens for schedules transmitted by its neighbors in other slots. It updates its list NB as it receives transmissions from others. If the entry of a neighbor in NB has not been updated for some time, that neighbor is deemed to have moved away (or powered off) and its entry is deleted. As a node listens for the schedules of the others, it also makes adjustments to its own schedules based on what it hears. Among the possible states of a slot described earlier, the most important states are *Trans* and *Recv*. We describe under what conditions a node can start (or stop) transmission to (or reception from) a one-hop neighbor.

A node should explicitly release a slot it no longer needs. A node may release an information slot in *info_schedule* when its transmission to the receiver(s) has completed, or when learning its transmission is colliding at a receiver (by hearing the schedule broadcasted by that receiver). Also, a node may release a slot in the *ctrl_schedule* (a permanent color) after it finds its transmission in this slot is colliding at one of its one-hop neighbors. Unused slots are released in the beginning of the allocation phase. To release a slot s , a node simply changes its state of the slot from *Trans* to one of the passive states, depending on the states of its neighbors. Its neighbors will be informed of this change when this node broadcasts its updated schedules. When a receiver of this transmission receives the broadcast, it learns the slot is released by the transmitter and stops receiving in that slot. A released slot can be reserved later for another transmission.

Reserving a new slot requires more care than releasing a slot, due to the possible conflict caused by a reservation. In the i th allocation frame, only nodes which acquired the i th temporary color in the preceding contention phase can reserve new information slots and permanent colors. There can be many nodes in this set, and the size of this set is likely to grow with the size of the network. In the i th allocation frame, the slot designated to temporary color tci is located at the very end. A node with color tci chooses its

²We neglect the small probability of collision among two adjacent nodes not sharing a common neighbor here for ease of discussion.

new transmission slots just before it announces its schedules to the neighbors in this last slot. (If a node also broadcasts in an earlier slot of this frame designated to its permanent color, it is not allowed to choose its new slots then.) By this time it has received broadcasts from all its one-hop neighbors with valid permanent or temporary colors and has learned their schedules. This node now chooses its new transmission slots based on this latest information. It can reserve a permanent color in the *ctrl_schedule* if it needs one, and reserve information slots in the *info_schedule* depending on its traffic requirement. If a node n_i wants to reserve a new slots transmitting to a neighbor n_j , it picks a slot s when the receiver n_j is either *Idle* or *Block_t*, and itself is either *Idle* or *Block_r*. If there are multiple slots satisfying the criteria, a node chooses one of them randomly. A node incorporates its new reservations into its schedules and broadcasts the updated schedules to its neighbors. The receivers of its new transmissions changes their states in the corresponding slots to *Recv*. A reservation is established in this way. Transmission subsequently takes place in this reserved slot in the corresponding information frames until the transmitter releases the slot.

After the i th allocation frame, every node with temporary color tci has had a chance to reserve new information slots and permanent colors. Whether they are able to reserve the slots they need depends on the current load and the schedules of its neighbors. The temporary color tci is no longer useful, so it disappears from the rest of this allocation phase. The Appendix contains a pseudo-code of E-TDMA that provides more details on the algorithm.

It can be proven that E-TDMA satisfies the following properties. Their proves are omitted here.

Property 1: If nodes in a network do not move, the schedules produced by E-TDMA are conflict-free for every node with a valid permanent color.

Property 2: When nodes move, collisions could take place in the schedules. For a node with a valid permanent color, a collision could last, at most, the duration of an information epoch.

Note that the schedules are conflict-free only for nodes with valid permanent colors. For a node n_i without a permanent color, it cannot make its current schedules known to its neighbors. Therefore there is no guarantee that frames will not collide at this node. The minimum number of permanent colors required to cover every node of a network is its distance-2 chromatic number, and is closely related to the maximal nodal degree ρ (it is lower bounded by $\rho+1$). The number of permanent colors should be large enough that every node can acquire a permanent color with high probability. By providing only a fixed number of permanent colors, E-TDMA is limited by nodal density and

cannot cope with the situation when all the nodes gather in a small area. It works best when the nodal density is uniform across the network.

When a node loses an information slot or its permanent color due to a collision, if it still needs it (a node always needs a permanent color) it will try to reserve another one in the next control epoch. To reduce the duration of the collisions, it is desirable to update the schedules more often and thus to have short information epochs. However, unless the length of the control epoch is reduced accordingly, this increases the overhead of E-TDMA. As a compromise, one can choose the *shortest* control epoch (i.e. only using *one* temporary color) while reducing the length of the information epochs (by reducing the number of information frames) until the overhead of E-TDMA reaches its maximal allowance. (This is the approach we take in our implementations in the simulator.) However, the frame structure described previously, where the number of the temporary colors is a design variable, is still useful if the length of the information epoch is determined by other considerations. As nodes move faster, the topology changes more frequently and collisions take place more often. It takes time for the protocol to resolve the conflict and to make new reservations. For E-TDMA to work well, it is necessary that nodes do not move too fast. As a reservation-based protocol, E-TDMA fails when the network becomes too volatile. How frequently the schedules are updated determines how well E-TDMA handles network mobility.

When a node is turned on, if there are other nearby nodes already operating, it can learn the schedules of these neighbors and build up its NB list by listening for their broadcasts. After it acquires a permanent color, this node becomes fully operational. If every node is turned on at the same time, both the *ctrl_schedule* and the *info_schedule* are null everywhere in the network. All the nodes would contend in the beginning because they all need permanent colors. In each control epoch, some nodes would succeed, first to acquire a temporary color then to acquire a permanent color, and become operational afterwards. There is no difference from the protocol's point of view; therefore E-TDMA does not have an explicit "network initialization phase".

D. Comparison with other protocols

To better understand E-TDMA, we compare it with other protocols that also use contention to make reservations. In FPRP [7], HRMA [8] or ADAPT [9], nodes contend in the slots they want to reserve (or their equivalence) directly. In E-TDMA, a node contends for permission to make a reservation in its neighborhood exclusively. A node only needs to get the permission once, even

if it needs multiple slots. After obtaining the permission, a node picks its new slots after gathering fresh information from its neighbors. It is possible to reserve different number or different types of slots, and to use different slot selection algorithms (a randomized scheme is used here). The way E-TDMA combines contention and reservation resembles the D-TDMA protocol in cellular networks [10], where a user terminal uses contention to make its bandwidth request known to the basestation, which then assigns the new slots based on the current schedules in the cell. To send a request successfully in D-TDMA corresponds to to obtain a temporary color in E-TDMA, for both are permissions in a local region to reserve (potentially many) new time slots. The difference is that while in D-TDMA a basestation is central controller, in E-TDMA there is no controller and a node with a temporary color functions like a (temporary) local coordinator to reserve its own new slots.

E. An example

We now illustrate via an example how E-TDMA updates the schedules (Figure 3). There are 6 nodes (A to F) in the network, and the E-TDMA protocol has temporary colors ($tc1, tc2$) and 4 permanent colors ($pc1$ to $pc4$). There are 4 information slots in an information frame ($is1$ to $is4$). The original topology is shown in Figure 3.a. Suppose the control schedule and the information schedule were both conflict-free when they were generated according to the original topology, and these schedules are shown in Figure 3.c. Suppose node E moved towards node C and a new link appeared between them (Figure 3.b). This causes conflict in the original schedules, and the corrupted schedules are shown in Figure 3.d. Two transmissions, from D to C in $is1$ and from F to E in $is2$, are corrupted, and they need to reserve new time slots. We also assume that at the same time, node A needs to reserve a new slot to transmit to node B . So we will see how the protocol reallocates conflicting transmissions and accommodates a new one. When the next control epoch begins, the three nodes A , D and F , which require new information slots, contend for the temporary colors. Assume they all succeed, and nodes A and D acquire $tc1$ and node F acquires $tc2$. In $A1$, nodes A and D update their schedules after hearing broadcast from all their neighbors. Both of them schedule their transmissions in $is4$. The partially updated schedules after $A1$ are shown in Figure 3.e. In $A2$, node F with $tc2$ updates its schedule. It picks $is3$ for transmission to node E . The updated schedules after $A2$ are shown in Figure 3.f, where the conflicting transmissions are reallocated to new slots and the newly arrived transmission is also assigned a slot. Although only unicast transmissions are shown in

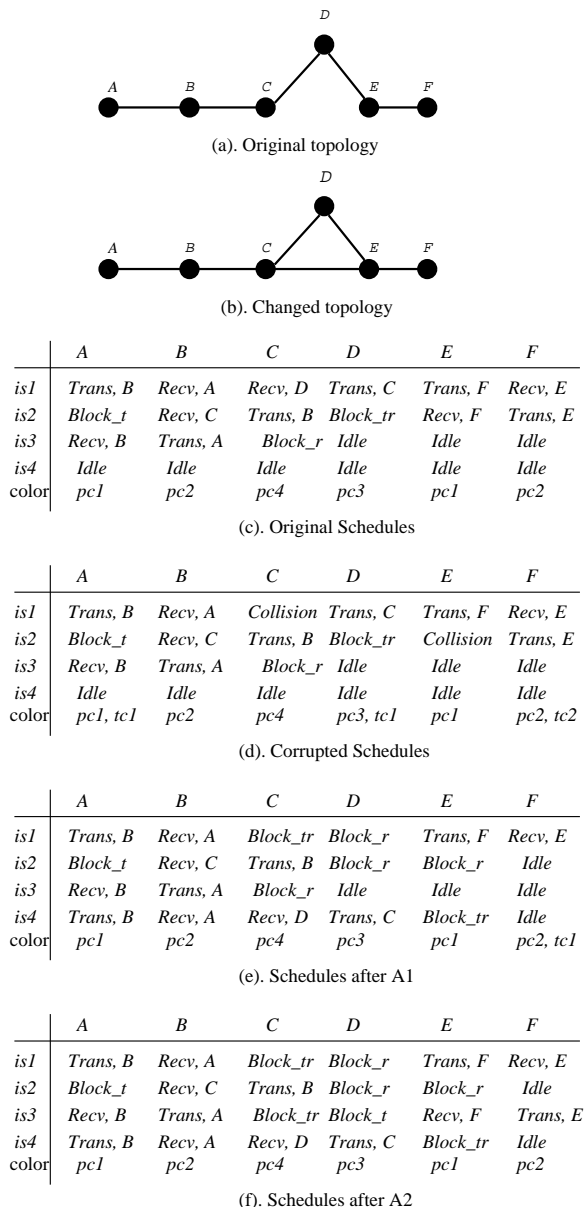


Fig. 3. Schedule update of E-TDMA in a small network. There are 4 permanent colors ($pc1$ to $pc4$), 2 temporary colors ($tc1, tc2$) and 4 information slots ($is1$ to $is4$).

the example, multicasts and broadcasts can be handled in similar ways.

III. SIMULATIONS

A. Implementation of E-TDMA

We have implemented the E-TDMA protocol with $ns2$ [11]. There are 40 slots in an information frame, each slots carries 32 bytes of information and a header of 4 bytes. The guard time between the slots is $20 \mu s$. Four information frames consist an information epoch. In a control epoch, there is only 1 temporary color, and 8

FPRP cycles are used in the contention phase for its assignment. There are 12 permanent colors. Transmission rate is 1 Mbps. E-TDMA itself consumes 14% of the bandwidth and the schedule update frequency (frequency of the control epoch) is 17 Hz.

Because IP packets have variable lengths, a packet often has to be transmitted in multiple time slots. Fragmentation and re-assembly of the packet into MAC data frames is necessary in this case. Besides slots required for user data transmission, every node also reserves a broadcast slot in the *info_schedule*. This broadcast slot is used for transmission of MAC control frames. IP packets generated by routing protocols are often broadcast, and they can be very irregular compared with other user data such as RTP voice flows. Without reserving a broadcast slot in advance, the delay for E-TDMA to reserve a slot upon the arrival of an IP packet may be unbearable. This broadcast slot is also used for sending user data packets when there is control packets to transmit. Transmission of an IP packet will fail if a constituent MAC fragment suffers a collision. A packet may also be dropped at the network layer if there is no route to the destination, or at the link layer if the interface queue is full (maximum length 50 packets used in simulations). The routing protocol used with E-TDMA is a QoS routing protocol based on AODV [12]. The protocol can establish bandwidth-reserved QoS routes for CBR flows. It also maintains best-effort routes like the original AODV protocol and uses them to transmit packets not having their QoS routes established. We defer the reader for details of this routing protocol to [13]. E-TDMA does not work well with the original AODV protocol, which changes routes frequently. Frequent route changes require frequent slot reservations and puts a heavy burden on E-TDMA. Routes are more stable with the QoS routing protocol mentioned here. The QoS routing protocol also reduces congestion by using bandwidth-reserved routes. Because the time frames in E-TDMA are pseudo-periodic (i.e. the interleaved control epoch makes the information frame aperiodic), an information slot cannot synchronize with data packets. We do not require that a source generate one packet per frame or that a packet be transmitted in a single time slot. When there are multiple flows being transmitted to a neighbor and time slots are reserved for these flows, packets from these flows are multiplexed and transmitted in all these slots. There is not one-to-one mapping between a time slot and a flow.

We compare E-TDMA with the IEEE 802.11 protocol via simulation. Because bandwidth cannot be reserved in 802.11, the QoS routing protocol mentioned early does not apply. Instead, the original AODV protocol provided in *ns2* is used. The intent here is to see how the combination

of E-TDMA and AODV-inspired QoS routing compares with the combination of 802.11 and pure AODV routing, the latter operating in a more or less “best effort” mode.

B. Simulation scenarios

A simulated mobile ad hoc network is generated as follows. There are 25 nodes in the network, and they are confined in a square area of 1000 m by 1000 m. The transmission range of a node is 250 m. A modified “way-point” movement model is used to model the random movement of the nodes [14]. In the beginning, the nodes are randomly placed in the area. Each node remains stationary for a pause time, the duration of which follows an exponential distribution with a mean of 10 seconds. The node then chooses a random point in the area as its destination and starts to move towards it. The speed of the movement follows a uniform distribution between 0 and the maximal speed v . Network mobility is varied when we change v . Different network scenarios for $v = 0, 5, 10$ m/s are generated. The scenario $v = 0$ represents a static network with no link change. At $v = 10$ m/s, on average a node experiences a link change every 5 seconds. After reaching a destination, a node pauses again and starts to move towards another destination as previously described. This process is repeated for the duration of the simulation (300 seconds). The only constraint of the movement pattern is that it does not cause network partitions. Without network partition, there is always a route from a source to a destination, so no packet is dropped because the destination is unreachable. All dropped packets are due to network congestion or temporary route failure. When the movement pattern is generated, care is taken to prevent network partition. If a partition occurs, the node causing the partition randomly picks another destination and starts to move towards it. The node does not pause in this case. An possible example of this network is a group of soldiers moving on foot in a loose formation. Changes in their relative positions are modeled by this movement pattern. In order for the leader to issue command to his soldiers, no one is allowed to stray away, therefore no partition occurs in the network. User traffic is generated with constant-bit-rate (CBR) sources, where the source and the destination of a flow are chosen randomly among the nodes. During its lifetime of 30 seconds, a CBR source generates 20 packets per second. A CBR source does not adjust its transmission depending on the network congestion, and all 600 packets are always transmitted irrespective of how many of them get through. There is no admission control for a CBR source. The size of a CBR packet is 64 bytes, and it becomes 84 bytes after an IP header is added. A packet is transmitted in three time slots. The starting time of a

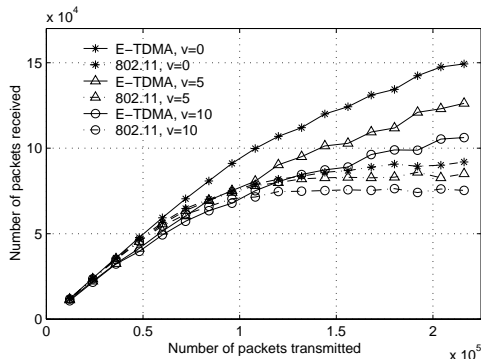


Fig. 4. Packet throughput for E-TDMA and 802.11.

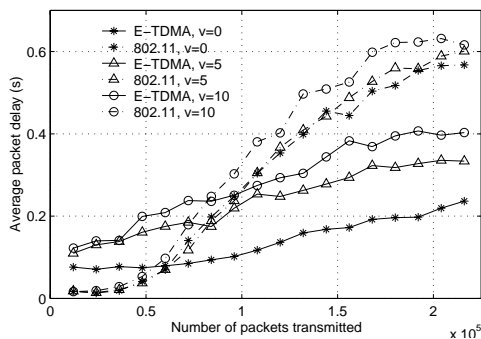


Fig. 5. Average packet delay for E-TDMA and 802.11.

session (which consists of a single CBR packet flow) is randomly chosen between 0 to 270 seconds, so a session always ends naturally by the end of the simulation. The offered traffic load is varied by increasing the number of CBR sessions generated during the simulation from 20 to 360. Ten different traffic patterns are generated and their simulation results are averaged. We measure the number of packets received by the destinations and the average packet delay. We also measure the number of sessions that are serviced and average packet delay for these serviced sessions. A session is called "serviced" if at least 90% of its packets are received by the destination. This is a measurement of QoS provided to the application layer.

C. Simulation results

Figures 4 and 5 show the packet throughput and average packet delay of E-TDMA and 802.11 under different traffic loads and node speeds. We start by looking at the immobile case ($v = 0$). When the network is static, once a slot is reserved it remains conflict-free. So this is the ideal case for E-TDMA. When the network traffic is light, both protocols deliver almost all the packets. The packet delay is much lower with 802.11, because under low traffic there is little collision, and a packet is usually trans-

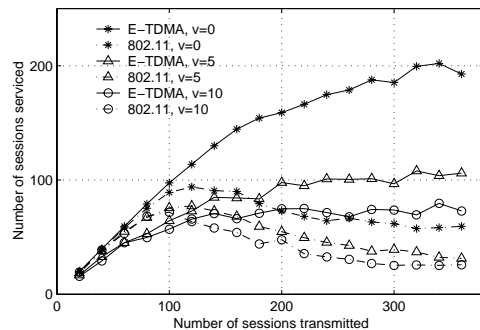


Fig. 6. Session good-put for E-TDMA and 802.11.

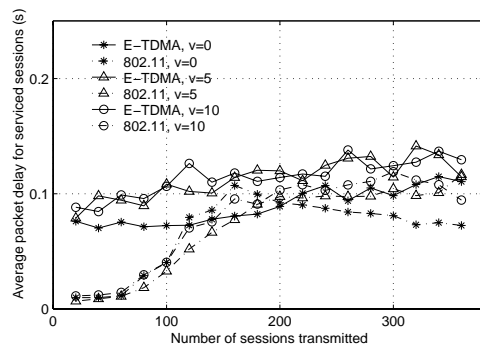


Fig. 7. Average packet delay for serviced sessions for E-TDMA and 802.11.

mitted successfully right away. With E-TDMA, a slot has to be reserved first which causes a non-negligible delay. When traffic gets heavy, more collisions (and backoffs) take place with 802.11, and the throughput saturates. Beyond a threshold, packet delay with 802.11 increases dramatically. With E-TDMA, every transmission is collision-free, which means its packet throughput increases steadily until every slot is reserved. Average packet delay with E-TDMA only increases slowly with offered traffic. Because a CBR source always transmits at the same rate, under heavy traffic E-TDMA cannot reserve enough slots. The network then becomes overloaded and packets are delayed and dropped. Compared with 802.11, E-TDMA is more susceptible to nodal movement. When nodes start to move, a slot reserved by E-TDMA can be corrupted and frame collisions take place. An E-TDMA node needs to contend again if it loses an information slot. It is also possible that the permanent color of a node becomes corrupted and has to be discarded. Before this node reserves another permanent color, it experiences a "black-out" and collisions could happen in its schedule. Every session passing through this node is affected. When this happens, the routing protocol needs to find another route not using this node. In contrast, the 802.11 protocol does not

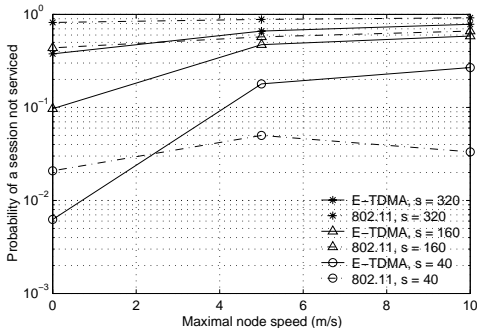


Fig. 8. Probability that a session is not serviced. Load s is the number of sessions transmitted in 300 seconds.

maintain any channel state, and the medium is acquired by RTS/CTS exchange for every packet. Mobility is handled only at the network layer. When network topology changes and a link breaks, the routing protocol reacts quickly by changing to a different route. Such a route change, and the resulting changes in bandwidth, are handled easily by 802.11 and AODV. In comparison, these changes are handled poorly by E-TDMA, especially with the original AODV protocol (simulated but not shown). As a consequence, E-TDMA degrades with node speed more quickly, both in terms of packet throughput and packet delay. It is expected that when node speed increases further E-TDMA will become inferior to 802.11 and breakdown at some point.

When compared at the session level (Figures 6 to 8), behavior of the two protocols becomes different from that at the packet level. Although the packet throughput of 802.11 saturates when traffic gets heavier, the corresponding session good-put decreases. This is because with 802.11, every packet is transmitted in the channel on an equal basis, and is equally likely to be dropped when traffic is heavy. As more packets are dropped from all the sessions, fewer sessions have 90% or more packets delivered, thus the session good-put decreases. With E-TDMA, a session which has its bandwidth reserved is guaranteed of its throughput, and is therefore not affected by network congestion. The session good-put is kept relatively high under heavy traffic. However, the session good-put drops faster with node speed than does the packet throughput. Compared with $v = 0$, at $v = 5$ m/s only half as many sessions are serviced, and at $v = 10$ m/s only one third of the sessions are serviced. This is because once a session is broken by some topological change, it may not restore its time slots, or the delay of doing so is long and many packets are dropped. This is not a problem with 802.11. In fact when nodes move, under light traffic the session good-put is lower with E-TDMA than with 802.11, due to the delay required to

restore the corrupted time slots. A serviced session often suffers little disturbance during its lifetime, and its packet delay is well below the average delay of all packets (Figure 7). It is clear that 802.11 is better suited for light traffic and highly mobile networks, whereas E-TDMA is better suited for heavy traffic and less mobile networks.

Besides CBR traffic, we also experimented with sending TCP traffic and confirmed that E-TDMA and TCP do not work well together. Because a TCP source adjusts its transmission based on its throughput, its transmission rate varies with time. E-TDMA has difficulty calculating and reserving a stable bandwidth for a TCP session. This then couples with the positive feedback nature of TCP and the resulting throughput is much lower than 802.11. More work is needed if E-TDMA is to be used to carry TCP traffic.

IV. CONCLUSIONS

A new protocol for generating and maintaining conflict-free TDMA transmission schedules for mobile ad hoc networks has been developed. It is based on the idea of frequently updating the current TDMA schedules on a local basis by many nodes in many parts of the network simultaneously. It is, in fact, a hybrid scheme which uses contention to determine the set of nodes which can make new slot reservations at a given instance. By using contention, the operation of a node is only affected by those nodes in its two-hop neighborhood, and the algorithm is insensitive to the network size. Consequently the protocol is scalable and can be used for large or dynamic networks. The schedules of the entire network evolve over time to accommodate changes in both the network topology and in the bandwidth requirements. E-TDMA is unique in that it uses a separate, dynamically maintained broadcast schedule (*ctrl_schedule*) to exchange scheduling information between nodes, and uses limited contention for signaling; in the schedule used for user data transmission (*info_schedule*), a node can reserve and mix different kind of transmissions (unicast, multicast and broadcast) freely. It is designed for heavy CBR traffic under low to medium network mobility. Its performance has been studied with simulations and is compared with that of the IEEE 802.11 protocol. Simulation results showed that E-TDMA works better under heavy CBR traffic, producing higher throughput and lower delay.

V. APPENDIX: PSEUDO-CODE OF E-TDMA

Parameters of E-TDMA {
number of permanent colors M ;
number of temporary colors N ;
 $PC = \{\text{all permanent colors}\}$, $TC = \{\text{all temporary colors}\}$;
 $ctrl_frame = PC \cup TC$;

number of information frames K in an information epoch;
number of information slots L in an information frame;
 $info_frame = \{\text{all information slots}\}$;
life time of a neighbor node nb_ttl ;
}

Data structure E-TDMA {

Data maintained at a node {
 my_id ;
 $my_ctrl_schedule$;
 $my_permanent_color = \{c \in PC, state(c) = Trans\}$;
 $my_temporary_color = \{c \in TC, state(c) = Trans\}$;
/* a node has at most 1 permanent and 1 temporary color */
 $my_info_schedule$;
information about a slot (color) s in $my_info(ctrl)_schedule$
is referred to as $my_state(s)$ and $my_target(s)$;
a list NB of 1-hop neighbors and their schedules, where an entry
contains: (id , $ctrl_schedule$, $info_schedule$, exp_time);
information of $n_i \in NB$ in a slot s is referred to as:
 $NB(n_i) \rightarrow state(s)$;
 $NB(n_i) \rightarrow target(s)$;
 $NB(n_i) \rightarrow exp_time$;
}

Information contained in a schedule-update packet (su_packet) {
(id , $ctrl_schedule$, $info_schedule$);
} /* su_packet should be encoded efficiently */
}

/* A node resets its states and NB list when powered on */

```
function node_initialization() {
  NB =  $\emptyset$ ;
  for ( $\forall s \in ctrl\_frame \cup info\_frame$ )
    {  $my\_state(s) = Idle$ ; }
}
```

/* A node contends for a temporary color for permission */
/* to reserve a permanent color or information slots */

```
function contention_phase() {
  if ( $my\_permanent\_color = \emptyset$  || need new info slots) {
    contend for a temporary color with FPRP;
    if (succeed to acquire  $tc \in TC$ ) {
       $my\_state(tc) = Trans$ ;
       $my\_target(tc) = Broadcast$ ;
    }
  }
}
```

/* In the allocation phase a node updates its schedules */

```
function allocation_phase() {
  at the beginning of A phase {
    for ( $\forall n_i \in NB, NB(n_i) \rightarrow exp\_time < current\_time$ ) {
      delete  $n_i$  from  $NB$ ;
      for ( $\forall s \in ctrl\_frame \cup info\_frame$ ,
      ( $my\_state(s) = Recv$  ||  $my\_state(s) = Trans$ )
      &&  $my\_target(s) = n_i$ ) {
        check_passive_slot(s);
      }
    }
  }
  release_unused_information_slot();
} /* delete obsolete neighbors and release unused slots */
```

in a slot c {
if ($c = my_permanent_color$) {

send ($my_id, my_ctrl_schedule, my_info_schedule$)
in a su_packet ;

} /* transmit schedules to the neighbors */

else if ($c = my_temporary_color$) {

if (it is the c^{th} A frame) {

if ($my_permanent_color = \emptyset$)

{ reserve_permanent_color(); }

if ($my_permanent_color \neq \emptyset$ && need new info slot)

{ reserve_information_slot(); }

} /* make new reservations */

send ($my_id, my_ctrl_schedule, my_info_schedule$)
in a su_packet ;

}

else { /* listen for the schedules of others */

listen for any incoming su_packet ;

if (receive an error free su_packet from node n_i) {

if ($n_i \notin NB$) {

add n_i to NB ;

} /* add a new neighbor */

$NB(n_i) \rightarrow ctrl_schedule = su_packet \rightarrow ctrl_schedule$;

$NB(n_i) \rightarrow info_schedule = su_packet \rightarrow info_schedule$;

$NB(n_i) \rightarrow exp_time = current_time + nb_ttl$;

update_my_schedule();

} /* update the schedules based on this packet */

if (receive a packet with error) {

$my_state(c) = Collision$;

} /* this color is now has a collision */

}

}

at the end of A phase {

if ($my_temporary_color \neq \emptyset$)

{ $my_state(my_temporary_color) = Idle$; }

} /* invalidate the temporary color */

}

/* Reserve a permanent color in $ctrl_schedule$ */

```
function reserve_permanent_color() {
```

if (\exists a color $c \in PC, my_state(c) = Idle$) {

$my_state(c) = Trans$;

$my_target(c) = Broadcast$;

} /* when there are more than one c , choose one randomly */

```
}
```

/* Reserve an information slot in $info_schedule$ */

```
function reserve_information_slot() {
```

for every required info slot to transmit to $R \subseteq NB$ {

if (\exists a slot $s \in info_frame$,

(($my_state(s) = Idle$ || $my_state(s) = Block_r$)

&& ($NB(n_i) \rightarrow state(s) = Idle$ ||

$NB(n_i) \rightarrow state(s) = Block_t, \forall n_i \in R$)) {

$my_state(s) = Trans$;

$my_target(s) = R$;

} /* when there are more than one s , choose one randomly */

```
}
```

```
}
```

/* Update my $ctrl_schedule$ and $info_schedule$ */

/* based on the schedules of the 1-hop neighbors */

```
function update_my_schedule() {
```

for ($\forall s \in ctrl_frame \cup info_frame$) {

if ($my_state(s) = Trans$)

{ check_transmission_slot(s); }

else

```

    { check_passive_slot(s); }
  }
}

/* Check a slot when I transmit */
function check_transmission_slot(s) {
  for ( $\forall n_i \in my\_target(s)$ ) {
    statei = NB( $n_i$ ) → state(s);
    targeti = NB( $n_i$ ) → target(s);
    if (statei = Collision || statei = Blockr ||
        statei = Blocktr || (statei = Recv &&
        targeti ≠ myid)) {
      check_passive_slot(s);
    } /* stop transmission when error occurs */
  }
}

/* Check a slot when I do not transmit */
function check_passive_slot(s) {
  num_trans_neighbor = 0;
  num_trans_to_me = 0;
  num_recv_neighbor = 0;
  for ( $\forall n_i \in NB$ ) {
    if (NB( $n_i$ ) → state(s) = Trans) {
      num_trans_neighbor ++;
      if (myid ∈ NB( $n_i$ ) → target(s) ||
          NB( $n_i$ ) → target(s) = Broadcast) {
        my_target(s) =  $n_i$ ;
        num_trans_to_me ++;
      }
    }
    else if (NB( $n_i$ ) → state(s) = Recv &&
            myid ∉ NB( $n_i$ ) → target(s))
      { num_recv_neighbor ++; }
  }
  if (num_trans_to_me > 0) {
    if (num_trans_neighbor > 1)
      { my_state(s) = Collision; }
    else
      { my_state(s) = Recv; }
    return;
  }
  if (num_recv_neighbor ≥ 1 &&
      num_trans_neighbor ≥ 1)
    { my_state(s) = Blocktr; }
  if (num_recv_neighbor ≥ 1 &&
      num_trans_neighbor = 0)
    { my_state(s) = Blockt; }
  if (num_recv_neighbor = 0 &&
      num_trans_neighbor ≥ 1)
    { my_state(s) = Blockr; }
  if (num_recv_neighbor = 0 &&
      num_trans_neighbor = 0)
    { my_state(s) = Idle; }
}

/* Stop transmissions in slots I do not need */
function release_unused_information_slot() {
  for ( $\forall s \in info\_schedule, my\_state(s) = Trans$ ) {
    if (s is no longer in use)
      { check_passive_slot(s); }
  }
}

```

```

/* Transmits or receives in information slot with info_schedule */
function information_slot(s) {
  if (my_state(s) = Trans && my_target(s) = R) {
    transmit an information packet (or a fragment thereof) to R;
  }
  else if (my_state(s) = Recv && my_target(s) =  $n_i$ ) {
    listen for an information packet info_packet from  $n_i$ ;
    if (info_packet is error free) {
      pass info_packet to upper layer;
    }
    else if (info_packet has error) {
      my_state(s) = Collision;
      drop info_packet;
    }
  }
}
}
}

```

REFERENCES

- [1] Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ, 08855, USA. *IEEE P802.11 Draft Standard for Wireless LAN: Medium Access Control (MAC) and Physical Layer (PHY) Specification*, draft p802.11d5.0 edition, July 1996.
- [2] E. Arıkan. Some complexity results about packet radio networks. *IEEE Trans. on Inform. Theory*, IT-30:681–685, 1984.
- [3] A. Ephremides and T. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, COM-38(4):456–460, April 1990.
- [4] A. Sen and M. Huson. A new model for scheduling packet radio networks. In *Proc. of INFOCOM*, 1996.
- [5] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE Trans. on Computers*, 38:1353–1361, 1989.
- [6] L. C. Pond and V. O. K. Li. A distributed time-slot assignment protocol for mobile multi-hop broadcast packet radio networks. In *Proc. IEEE MILCOM*, pages 70–74, 1989.
- [7] C. Zhu and M. S. Corson. A five phase reservation protocol (FPRP) for mobile ad hoc networks. In *Proc. IEEE INFOCOM*, 1998.
- [8] Z. Tang and J. J. Garcia-Luna-Aceves. Hop reservation multiple access (hrma) for multichannel packet radio networks. In *Proc. of IC3N*, 1998.
- [9] I. Chlamtac, A. D. Myers, V. R. Syrotiuk and G. Zaruba. An adaptive medium access control (MAC) protocol for reliable broadcast in wireless networks. In *Proc. of ICC*, 2000.
- [10] K. Joseph N. D. Wilson, R. Ganesh and D. Raychaudhuri. Packet CDMA versus dynamic TDMA for multiple access in an integrated voice/data pen. *IEEE JSAC*, 11:870–884, 1993.
- [11] The VINT Project: A collaboratoin between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PAR. *The NS manual*, May 2001.
- [12] C. Perkins, E. M. Royer and S. R. Das. Ad Hoc On-Demand Distance Vector routing. In *Internet-Draft, draft-ietf-manet-aodv-06.txt*, July 2000.
- [13] C. Zhu and M. S. Corson. Qos routing for mobile ad hoc networks. submitted to INFOCOM 2002.
- [14] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocol. In *Proc. of MOBICOM*, 1998.