EXPLOITING DIRECTIONAL ANTENNAS TO PROVIDE QUALITY OF SERVICE AND MULTIPoint DELIVERY IN MOBILE WIRELESS NETWORKS

Martha E. Steenstrup
Stow Research L.L.C.
Flanders, NJ

ABSTRACT

We present a TDMA-based medium access control (MAC) algorithm specifically designed for link scheduling in a mobile wireless network composed of nodes equipped with directional antennas, each of which is capable of forming a single beam in an arbitrary direction. Using the MAC algorithm, a node negotiates slot reservations with a neighbor based only on information about the two nodes’ available slots and the expected traffic load. We provide simulation results illustrating the performance of this algorithm in terms of goodput, delay, and success of reservations, and we also compare its performance against that of a CSMA/CA MAC algorithm designed for use with directional antennas. Furthermore, we show that if the capabilities of each antenna are expanded to include formation of multiple simultaneous beams, then the MAC algorithm can be extended to efficiently negotiate reservations for multicast transmissions.

INTRODUCTION

The throughput attainable in a multihop wireless network depends upon not only the data rate of the wireless channel but also the number of nodes that participate in forwarding a packet between source and destination and the number of simultaneous noninterfering transmissions that are achievable. Interference can be mitigated by reducing the volume over which the energy of a transmission is dispersed. One approach is to restrict the range at which a transmission can be detected by reducing transmit power, but this increases the number of hops over which a packet must travel to reach its destination. Gupta and Kumar have explored this conflict in depth and have proven in [5] the fundamental asymptotic result that in a multihop wireless network the throughput per flow must decrease as the number of nodes increases. Specifically, if the network consists of $n$ randomly-placed nodes, each of which can transmit $W$ bits per second, the throughput from a node to a randomly-selected destination is $\Theta(\frac{W}{\sqrt{n \log n}})$, and if the network is such that the node positions, transmission ranges, and traffic patterns are all chosen optimally, the throughput from a node to a destination at a bounded distance is $\Theta(\frac{W}{\sqrt{n}})$.

Interference can also be mitigated by using directional antennas to transmit and receive energy within narrow beams. While the asymptotic results of [5] still apply, there are significant throughput gains to be realized with directional antennas, as discussed in [10,15]. By limiting the directions in which energy is dispersed and from which energy is received, nodes equipped with directional antennas should theoretically result in a network capable of supporting more simultaneous noninterfering transmissions and exhibiting a lower probability of detection, than if the nodes were to radiate and acquire energy over a wider volume. Anticipation of these performance benefits has prompted research on networking algorithms for exploiting the capabilities of directional antennas. Much of the research to date is theoretical and based on hypothetical antennas, but with the expected increase in availability of directional antenna technology, the focus is likely to shift toward experimentation in deployed networks.

In this paper, we address a single networking problem – medium access control (MAC) – in the context of networks with directional antennas. Prior work on medium access control in such networks has covered a wide range of approaches including slotted ALOHA [16,13], CSMA/CA [6,9,11], and TDMA [4,2] schemes. We have selected a TDMA-based MAC algorithm because it can provide bounds on delay and delay variance and thus can help to support delay-sensitive applications. This algorithm schedules transmissions on individual links and attempts to reserve time slots on each link according to the current traffic offered to the link. Moreover, it is capable of reserving the same time slot on multiple links to accommodate multipoint transmissions. For each data packet transmitted, the sending node expects to receive an immediate acknowledgement from each intended recipient and may retransmit an unacknowledged packet depending upon the reliability and delay needs of the application. As points for comparison, the TDMA algorithm described in [4] also bases slot reservations on traffic load but does not dynamically adjust reservations as load changes; the TDMA algorithm described in [2] permits simultaneous transmissions to different neighbors but does not attempt to coordinate slot reservations for multipoint transmissions; and neither approach provides acknowledgements for data packets received.

In the context of networks with omnidirectional antennas, there exists a large body of work on TDMA for scheduling node broadcasts, much of it devoted to proving NP-hardness of the problems and discovering good approximation algorithms. More recently, the focus has shifted
to design of distributed scheduling algorithms that can accommodate changes in neighborhoods, caused by node movements, without global recalculation of transmission schedules (see [7,8,3,1]). With these approaches, a node requires information about its one- and two-hop neighbors to construct a conflict-free transmission schedule within this neighborhood. Since our MAC algorithm performs link as opposed to node scheduling (see [12] for an excellent characterization of the general channel assignment problem, including node and link scheduling), it requires less information to determine a transmission schedule; in particular, only the available time slots at sender and receiver are required to create a transmission schedule that is conflict-free with high probability.

Our MAC algorithm enables a node to negotiate a reservation, with one or more neighbors, for a sufficient number of time slots to handle the current traffic load and accommodates multicast as well as unicast traffic. The principal features that distinguish this algorithm from the others referenced above are the ability to negotiate slot assignments on demand according to traffic load and to multiple destinations. Since slot reservations are neighbor-specific, the algorithm readily adapts to the addition or loss of a neighbor, without having to modify reservations associated with other neighbors. When selecting time slots for reservation, a node assesses slot occupancy, accounting for its reserved slots and also using information obtained from neighbors’ transmissions for which it is either an intended or an accidental recipient. The algorithm does not guarantee freedom from collisions among transmissions, but it does attempt to reduce the potential for collisions and to alter reservations when collisions are suspected.

**TDMA MAC Algorithm**

The objectives of this algorithm are to allocate time slots so as to meet the service needs of each session, prevent starvation of any one session, and maximize the number of sessions that can be supported simultaneously in the network. The algorithm attempts to meet these objectives by reserving and scheduling slots for unicast and multicast traffic according to demand, by offering an unbounded number of opportunities for attempting transmission, by resolving contention without bias, and by separating simultaneous transmissions in distance and direction. We now present the algorithm, including a description of the organization of slots and the algorithms for their reservation and usage by both unicast and multicast traffic. First, however, we provide a list of our assumptions about the capabilities of the antennas and the network, which have influenced the design of this algorithm.

**Assumptions**

We assume that communication is half-duplex (i.e., when a node is transmitting it cannot also be receiving, and vice versa), but that a node can rapidly switch between transmitting and receiving modes so that it can quickly respond to requests and receive responses. Moreover, we assume that each node can receive omnidirectionally and can transmit and receive directionally using an antenna that is capable of forming multiple beams in arbitrary directions. All transmissions are directional. Receptions are directional whenever a node expects to receive a packet from a particular neighbor. When a node is neither transmitting nor receiving or expecting to receive a transmission from a particular neighbor, it listens omnidirectionally for transmissions that might be directed toward it from other nodes.

Slot alignment relies on nodes possessing synchronized clocks (which might be provided by GPS). Beam alignment relies on each node possessing accurate knowledge of the direction in which each of its neighbors lies, particularly for narrow beams. We assume that information about the direction in which a neighboring node lies is provided by a neighbor discovery procedure. This procedure might exchange positional information between nodes, compute direction of arrival of transmissions, or use predetermined pointing directions to determine the direction from a node to each of its neighbors. While the neighbor discovery procedure is crucial for the formation and maintenance of the network, and in particular for providing the MAC algorithm with accurate direction information for each neighbor, a detailed description is beyond the scope of this paper.

For multicast reservations and transmissions, we assume that each node can transmit or receive different packets over different beams simultaneously. Endowing an antenna with these capabilities enables a node to negotiate a slot reservation simultaneously with each multicast destination and to receive immediate acknowledgements from neighbors for multicast as well as unicast data packets. If we assume that the antenna is not capable of transmitting or receiving different packets on different beams, then the MAC algorithm must be modified to retain the same functionality and its performance suffers. In this case, a node must have a means of scheduling receptions from each multicast destination, so that it can successfully receive information about slot availability for reservations and acknowledgements for transmitted data packets. Sequential receipt of transmissions from individual multicast destinations not only slows down both slot reservation and data transport, but also increases(2,5),(996,993)
antennas has followed a different course, seeking to exploit a node’s ability to cover multiple destinations with a single transmit beam to reduce energy consumption but sacrificing acknowledged delivery. In particular, Wieselthier, Nguyen, and Ephremides [14] have focussed on the development of techniques for selecting transmit power, beamwidth, and multicast trees to increase the lifetime and throughput of energy-limited networks. While our MAC algorithm (because of the requirement for acknowledgements) cannot easily take advantage of beamwidth adjustment to cover multiple destinations with a single beam, it can be combined with appropriate selection of multicast trees and transmit power for each hop to control the amount of energy dispersed. This is a topic for future exploration and not covered in this paper.

Slotting Structure and Reservations

The MAC algorithm operates within a slotting structure in which time slots are organized into frames and frames into epochs. Each frame contains slots for negotiating reservations, reserved slots, and slots that can be used on a contention basis. The boundaries between these types of slots may be selected according to the amount and nature of the traffic expected in the target network but cannot easily be adjusted once set. Each negotiation slot comprises two minislots for requesting and responding to slot reservations. Reserved slots are used only for transmitting and receiving data packets and acknowledgements related to traffic flows for which there are active reservations. Packets originally transmitted in reserved slots but for which no acknowledgements are received can be retransmitted in contended slots, up to a specified number of times if reliable delivery is desired. Contended slots also carry packets associated with flows that do not desire a reservation, have not yet successfully established a reservation, or submit traffic at a rate higher than the reservation can accommodate.

At the beginning of each frame, a node attempts to negotiate slot reservations for flows. Any node that is not requesting a reservation during a particular negotiation slot listens omnidirectionally for other reservation requests during that slot. Unicast and multicast reservations are intermingled, and negotiation slots are chosen at random for each reservation, subject to the constraints that negotiation of a unicast reservation requires a single slot while negotiation of a multicast reservation requires two consecutive slots. Randomized selection of negotiation slots helps to spread reservation requests throughout the negotiation portion of a frame, provided the number of requests is considerably smaller than the number that can be accommodated within the frame, and thus increases the chances of that the intended recipient receives a reservation request. A node expects a response from the intended recipient of a request, and when a response is not forthcoming, the node retries the request during the following frame.

Reservations may be established during each frame, but are expressed in terms of slots per epoch and may persist over multiple epochs. To negotiate a suitable reservation for a traffic flow, a node requires an estimate of the number of slots desired. Such an estimate might be obtained by receiving a request for an end-to-end reservation initiated by the source of a session or by measuring the traffic offered to a particular link over a given time period, depending upon the granularity of the flow. At the finest granularity, a traffic flow might consist of packets from an individual session between a source and destination, and at the coarsest granularity, it might consist of all packets sent by a node to a neighbor, regardless of session. From the perspective of the MAC algorithm, the granularity of a traffic flow is immaterial, and the means of acquiring an estimate of the desired number of slots is not within its purview. Each successful reservation results in the initiating node establishing for the flow a maximum number of packets that may be transmitted per epoch in reserved slots. Any packets received from the flow in excess of this limit are relegated to contended slots. At the beginning of each frame, a node randomly assigns to contended slots data packets lacking reserved slots. The assignment procedure spreads packets over the contended slots such that at most half of these slots are occupied, ensuring that a node can receive and transmit during this portion of the frame and increasing the chances that it can do so successfully. Packets not assigned a contended slot within the current frame are retained for assignment in the following frame.

Unicast

When negotiating a reservation for a unicast flow, a node sends a request to the corresponding neighbor in the selected negotiation slot. The request contains the number of slots desired and the set of available slots with which the neighbor can make the reservation. From the perspective of the requesting node, a slot in the reserved portion of a frame is available for reservation if the node is not currently scheduled to transmit or receive in that slot. The node transmits its request in the first minislot of the negotiation slot and in the direction $(\theta, \phi)$ toward the neighbor and then listens in the same direction for a response from the neighbor during the second minislot. If the neighbor receives the request, it compares the node’s available slots against its own and attempts to select the requested number of slots in a manner that tends to space them equally over the epoch, starting with the current frame, so as to reduce the time that a packet must wait for a transmission slot. The neighbor formulates a response containing the number and set of slots reserved and reserves these slots for reception. A node and its neighbor, however, might share no common available slots and hence cannot currently establish a reservation. In this case, the neighbor provides a response indicating that no slots have been reserved, and the node does not retry the request unless it continues to receive data packets from the given flow.

When the node receives a response from the neighbor indicating that at least one slot has been reserved for the flow, it reserves all slots for transmission indicated in the response. The reservation persists at the node and its neighbor until there is a hull in traffic of duration exceeding a

3
specified limit, in which case each of the nodes assumes that the flow has concluded and the reservation can be released. While the reservation is in progress, during each slot reserved, the node transmits a data packet from the flow in the direction \((\theta, \phi)\), and the neighbor listens in the direction \((\pi + \theta \mod 2\pi, \pi + \phi \mod 2\pi)\) for such a packet. After transmission of a data packet, the node listens for an acknowledgement from the neighbor in the direction \((\theta, \phi)\), and after reception of a data packet, the neighbor transmits an acknowledgement to the node. Both data and acknowledgement packets fit within a single reserved slot.

In the basic version of the MAC algorithm, a slot is available for reservation at a node if that node is not currently scheduled to transmit or receive during that time. We have experimented with the following additional conditions for deeming a slot available for reservation, which are intended to reduce the probability of interfering transmissions of data packets and acknowledgements. If a node has previously overheard a completed reservation (both a request from a node, \(Y\), and the corresponding response from a node, \(Z\)) and has thus learned of the slots reserved for that flow, it marks each of these as unavailable to be reserved for transmission or reception in the directions from itself to \(Y\) and to \(Z\). Furthermore, if a node overhears a data or acknowledgement packet destined for another node, it marks the slot as unavailable to be reserved for transmission or reception in the direction from itself to the source of the packet. This approach, which we term “eavesdropping” relies on nodes being able to detect and decode the contents of packets not destined for them and to have accurate directional information about the source of overheard packets. Unavailability markings applied to slots by eavesdropping decay with time, provided no traffic is overheard from the specified directions, in order to prevent slots from remaining unavailable for reservation after flows cease.

**Multicast**

When attempting a reservation for a multicast flow, a node sends a beacon to all of the neighbors intended as recipients of a multicast transmission. We assume that each pair of recipients is directionally separated by at least one beamwidth, to simplify the discussion. A beacon contains the number of reserved slots desired and the set of recipients and is used to solicit the list of available slots from each participating neighbor. The node forms multiple beams in the directions \((\theta_i, \phi_i)\) toward each recipient \(i\) and transmits its beacon simultaneously over all beams in the first minislot of the first negotiation slot. It then forms multiple receive beams in those same directions and listens for requests issued by those neighbors during the second minislot of the first negotiation slot. Each of these requests is identical to a unicast request and contains the number of reserved slots desired and the set of available slots, but unlike the case of a unicast request, failure to receive a response does not elicit another request during the following frame.

The node initiating a multicast reservation retries the beacon in the next frame if it fails to receive a request from any neighbor. As long as it receives at least one request, however, the node collects all such requests and attempts to find a set of available slots common to all participating neighbors and itself. In a heavily-loaded network, these nodes might not share any common slot, and multiple slots might be required for multicasting a single data packet to all intended recipients such that each such slot is shared by a proper subset of the recipients. In this case, the objective is to minimize the number of reserved slots devoted to a single multicast packet, and hence the number of transmissions required for multicasting.

The process of finding the minimum set of slots necessary to multicast a single packet is equivalent to finding the minimum set cover given a set of subsets, which is an NP-hard problem. A greedy approximation algorithm exists that at each step chooses the slot available to the most remaining uncovered recipients. We set an upper limit on the number of slots that a single multicast packet can occupy to prevent multicast flows from consuming all slots. If after a single application of the greedy algorithm the number of slots exceeds this limit, the node executes the algorithm again, beginning instead with the slot that covers the second largest set of uncovered recipients. The node is allowed to execute the greedy algorithm a maximum number of times in an attempt to find a set of slots of acceptable cardinality. If no such set is found, the node chooses the set of minimum cardinality among those that it has generated. The node repeats this procedure according to the number of slots originally desired, but may terminate the procedure prematurely if the number of recipients covered at repetition \(i + 1\) is less than that at repetition \(i\), thus ensuring that all recipients are provided an equal number of slots.

Since this procedure requires a nontrivial amount of computation, a node is allotted the duration of one minislot to compute a set of slots for a multicast reservation. The node formulates a separate response for each group of recipients covered by a given set of slots, containing the number and set of slots reserved for that group. During the second minislot of the second negotiation slot, the node forms multiple beams in the directions of the participating neighbors and transmits the separate responses simultaneously over the corresponding beams. Once the slots are reserved, the node transmits multicast data packets by forming multiple transmission beams in the directions of the intended recipients during the designated slots, and each participating neighbor listens for such a packet in the direction associated with the slot and replies with an acknowledgement when a data packet is received successfully.

**Simulations**

We performed simulations to assess the sensitivity of our TDMA-based MAC algorithm to variations in antenna beamwidth, transmission range, and traffic intensity and to capture and eavesdropping. The algorithm is modelled in isolation so that performance is not perturbed by interactions with other network control algorithms. Each network
contains 20 nodes laid out randomly in square of 10 distance units on a side; since neighbor discovery is not modelled, nodes are assumed to be stationary. The beam pattern is modelled as a sector with angle equal to beamwidth and radius equal to transmission range. Beamwidth varies between 5° and 360° and transmission range between 3.5 and 5 distance units. Since route selection is not modelled, each traffic flow has significance only between neighboring nodes. All flows are sporadic lasting from $10^4$ to $6 \times 10^4$ time units with rates varying from 1 to 10 packets per $10^3$ time units, and with up to $10^6$ time units elapsing between successive flows to the same neighbor(s). Moreover, each flow requests a reservation and hence contended slots are used only for excess or retransmitted data packets. Each data packet lacking an acknowledgement can be retransmitted a maximum of 4 times. The number of intended recipients of a multicast packet is limited to 10, and the number of iterations of the multicast slot assignment procedure to 4. Each epoch spans $10^3$ time units and contains 10 frames. Each frame contains 10 negotiation slots, 30 reserved slots, and 10 contended slots. The bit rate is 500 bits per time unit. Packet sizes are 800 bits for data, 324 bits for reservation requests and responses, and 32 bits for beacons and acknowledgements. The duration of each simulation is $15 \times 10^4$ time units. All simulations were performed with a discrete-event simulator implemented in Mac Common Lisp.

We compare the behavior of the TDMA MAC algorithm to that of a directional CSMA/CA MAC algorithm that differs from standard derivatives of IEEE 802.11b in the following respects. All packets are sent directionally, and all packets except RTSs are received directionally, i.e., nodes listen omnidirectionally only for RTSs. A node ignores any packet received but not intended for it; Ramanathan demonstrated in [11] that in the directional context, the benefits of not remaining idle in response to an RTS or CTS meant for another node outweigh the costs of occasional collisions that might result. Also, a node does not sense the channel prior to attempting transmission. If a node fails to receive a CTS in response to an RTS or an ACK in response to a DATA packet, it cycles through its other neighbors for which it has packets to transmit before returning to make another attempt at successful transmission to the given neighbor. Each data packet is allotted a maximum of 5 attempts for successful transmission to a neighbor. When a node has multiple packets to transmit, it enters listening mode with probability 0.5 following each transmission, to increase the chance of receiving data packets from other nodes. Packet sizes are 800 bits for DATA and 128 bits for RTS, CTS, and ACK packets.

RESULTS

A set of numerical results are shown in table 1, each entry of which represents the mean value over 10 different simulations. Performance measures include goodput (ratio of packets successfully received to those queued for transmission, correctly accounting for multicast transmissions), delay (time elapsed between generation and successful reception of a packet), and several measures of reservation completion (percentages of successful reservations for the requested number of slots to the requested number of destinations, partial reservations providing fewer than the requested number of slots or to fewer than the requested number of destinations, and failed reservations). We draw the following conclusions from our simulation experiments.

The goodput provided by the TDMA algorithm is in general high and exceeds that of the CSMA/CA algorithm under all conditions tested. The average delay for successful data transmissions is lower with CSMA/CA than with TDMA, because a data transmission can immediately follow a successful RTS/CTS exchange whereas with TDMA, a data packet may have to wait an entire epoch for its assigned transmission slot. With this CSMA/CA algorithm, the delay a data packet experiences at a node depends upon the number of neighbors and the expected number of transmission attempts required to send data to and from each neighbor, whereas with the TDMA algorithm, the delay depends upon only the duration of an epoch, which is configured. In the presence of multicast traffic (TDMA only), average packet delay is significantly larger than for unicast traffic. We suspect the reason is that the slot assignment procedure tends to evenly separate slots reserved for a unicast flow, whereas no such separation is attempted for multicast, thus leading to longer delays in waiting for slots.

For TDMA, capture offers virtually no discernible performance advantage, except in the extreme case of omnidirectional antennas. For CSMA/CA, however, capture offers significant improvements in performance as beamwidth increases. Presumably this is because each data transmission requires a separate RTS/CTS exchange and nodes listen omnidirectionally for RTSs leaving them vulnerable to interference which increases with beamwidth. With TDMA, once a reservation is established, there is no need to exchange information prior to transmission of a data packet, leaving fewer opportunities for interference. We note that the additional packet exchanges required for successful data delivery imply that CSMA/CA disperses more energy than TDMA, reducing efficiency and creating the potential for more interference and a higher probability of detection. Although the results are not included in table 1, we measured the number of transmissions required for successful packet delivery with each algorithm and not surprisingly found it to be significantly higher for CSMA/CA.

The TDMA algorithm has a high rate of success in establishing reservations under the traffic loads applied. This rate decays with increased beamwidth, in large part because of interference among transmissions in reserved slots, resulting in repeated failure to receive data packets or acknowledgements and ultimately in reserved slots being returned to the available pool, thus reducing the number of slots reserved for the affected flows. Our simulations did not model adjustment of reservations when interference is suspected, and hence once lost, slots are not later regained by a flow. For heavier traffic conditions, resulting from
increased rate, additional neighbors, and introduction of multicast traffic, approximately one-third of the slots are reserved for transmissions or receptions at any given time. Somewhat surprisingly, eavesdropping does not seem to provide a performance advantage to the TDMA algorithm under any conditions tested, although we expect that it will do so in exceptionally dense and roughly linear networks.

Based on the results of our experiments, we believe that a distributed, direction-sensitive slot reservation and scheduling scheme, such as our TDMA MAC algorithm, can effectively exploit many of the characteristics of directional antennas to increase the number of sessions whose desired quality of service can be supported by the network. Furthermore, with multiple-beam antennas capable of carrying different packets on different beams, we can offer a MAC algorithm that provides multicast slot reservations and reliable multicast transmission with minimal additional overhead above that incurred in the unicast case.

REFERENCES


