A Race-Free Bandwidth Reservation Protocol for QoS Routing in Mobile Ad Hoc Networks

Imad Jawhar and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

***Abstract*— Routing protocols are an essential part of the efficient design of mobile ad hoc networks (MANETs). Existing routing protocols such as DSR, AODV, and TORA are based on a best effort strategy [18][19]. However, in order for MANETs to be practical for more demanding real time applications such as multimedia, providing a certain needed level of quality of service becomes an essential component in the communication protocol design [2][11][21][22]. QoS routing protocols provide the capability of finding a path between two nodes which satisfies the application layer's minimum bandwidth requirements. Previous papers addressed this issue for different communication environments such as TDMA (Time Division Multiple Access) [7][8][12][13] and CDMA (Code Division Multiple Access)-over-TDMA [3][14][15]. While most of these models are generally more practical and less expensive, they impose on the designer the constraint of the hidden terminal and exposed terminal problems. The paper by Liao and Tseng [12] addressed these issues and provided a TDMA-based bandwidth reservation protocol for QoS routing in MANETs. However, this protocol does not account for the race condition which can become more significant with increased node mobility, network density and higher traffic loads. This race condition is also a limitation of other QoS routing protocols [7][8]. This paper addresses this issue and provides a protocol which enables the network to cope with this and other related problems such as parallel reservation. We also provide increased optimizations which significantly enhance the throughput and efficiency of the QoS routing protocol.**

Keywords: mobile ad hoc networks (MANETs), quality-of-service (QoS), routing, Time Division Multiple Access (TDMA), wireless networks.

I. Introduction

Networking is becoming an essential part of society, and mobile ad hoc networks (MANETs) provide flexibility and adaptability in this environment [21][22]. As mobile electronic devices advance in capabilities, communication between these devices becomes essential. MANETs allow mobile computers and devices to communication with each other without an existing fixed topology or wiring. Mobile nodes establish a network on the fly as they come within range of each other. Communication between two nodes is done either directly with 1-hop if they are within range of each other, or indirectly using multiple hops through intermediate nodes in between. Nodes are free to move around, join and leave the network as needed. As this happens, new links form as nodes come within range of each other, and existing links break as two nodes move out of range of each other. These constant changes in topology impose a significant challenge for the communication protocols to continue to provide multi-hop communication between nodes.

This work was supported in part by NSF grants CCR 0329741, CCR 9900646, ANI 0073736, and EMI 0130806. Email: {imadj, jie}@cse.fau.edu

Existing MANET routing protocols provide the capability for establishing multi-hop paths between nodes on a best effort basis [18][19]. However, some applications, such as real-time and multimedia, need not only the capability to establish communications between nodes but also require of the network quality of service (QoS) guarantees on bandwidth, bit error rate, and delay. The bandwidth requirement is usually the most essential and challenging in such a dynamic environment.

There are several papers that address the subject of QoS routing in MANETs in different environments and with different models and approaches [5][10][16][20][23][24]. In this paper, we consider the problem of QoS routing in a TDMA (Time Division Multiple Access) environment. This communication protocol is a simpler and less costly alternative to the CDMA-over-TDMA environment. QoS routing protocols for CDMA-over-TDMA based ad hoc networks are considered in other papers [3][7][8][14][15]. In the latter protocol, a particular node's use of a slot on a link is dependent only upon the status of its 1-hop neighbor's use of this slot. However, in the TDMA model, which we assume in this paper, a node's use of a slot depends not only on the status of its 1-hop neighbor's use of this slot; its 2-hop neighbor's current use of this slot must be considered as well. This is due to the well-known hidden and exposed terminal problems [7][12], which must be taken into account. A $hidden\ terminal$ problem in a wireless environment is created when two nodes, B and C for example, which are out of range of each other transmit to a third node A, which can hear both of them. This creates a collision of the two transmissions at this third node, A. On the other hand an $exposed\ terminal$ is created in the following manner. A node A is within range of two other nodes B and C (between them) which are out of range of each other, and A wants to transmit to one of them, node B for example. The other node, C in this case, is still able to transmit to a fourth node, E which is in C's range (but out of the range of node A). Here A is an exposed terminal to C but can still transmit to B.

Liao and Tseng [12] provided a TDMA-based bandwidth reservation protocol for QoS routing in MANETs. However, their approach does not consider several issues, such as racing conditions and parallel reservation problems. They use only

two states to indicate the status of each slot: $free$ and $reserved$. Since simultaneous QoS route request messages reserve slots independently, multiple reservations can occur at each particular slot. These race conditions can reduce the throughput and efficiency of communications in such an environment as mobility of the nodes increases. In this paper we address these issues and provide a solution to these problems. Namely, we provide a race-free bandwidth reservation protocol for QoS routing in TDMA-based ad hoc networks. Our protocol also improves the performance of the network, especially in conditions of higher network density, higher node mobility and increased traffic. Furthermore, we provide some optimization techniques, which additionally contribute to improving the efficiency of the QoS routing protocol.

In order to solve the race condition and parallel reservation problem, our protocol adopts a more conservative strategy. While previous work in this area uses two states to control slot release and reservation: $free$ and $reserved$, our protocol uses three states: $free$, $allocated$, and $reserved$ to better control this process and provide race-free operation. The addition of the $allocated$ state which is described in detail later in this paper, allows nodes to avoid the multiple allocation of the same slots which are allocated by a forwarded QoS route request message but not yet confirmed (i.e. $reserved$) with a QoS route reply message. Furthermore, our protocol provides more performance optimization through the use of a wait-before-reject strategy which allows a QoS route request a better chance of getting forwarded (i.e. not rejected) by an intermediate node (i.e. enough slots are able to be allocated for the QoS request) in case the allocated slots are freed within a predetermined acceptable delay. This is done using TTL timers which revert slot status from $allocated$ to $free$ in the case where the QoS reply massage is not received whithin a period of time which allows it to comply with the QoS route request delay requirements.

The remainder of the paper is organized as follows. Section 2 discusses related work that has been done in this field. Section 3 provides background and current research. It also discusses the limitations of existing protocols and the racing conditions which are possible with certain situations, and which degrade the performance of the routing protocol. Also, in this section, we provide examples and discuss the occurrence of the racing conditions. In section 4, we present our protocol along with the corresponding algorithms, queue and timer definitions and slot status update rules. We also show how our protocol solves the race conditions and discuss the effect of the strategies used on the network performance. The last section will present conclusions and future research.

II. Related Work

Bandwidth reservation with QoS routing in MANETS is an issue that has been and continues to be investigated by current researchers. In [3] a ticket-based QoS reservation protocol has been proposed. However, it makes the assumption that the bandwidth calculation of a node can be determined independently of its neighbors. This is a strong assumption because such a protocol might require a multi-antenna model. In [14][15] a calculation algorithm for bandwidth is presented. However, it assumes that neighboring nodes broadcast with different codes, which is the case in CDMA-over-TDMA model. In that case a code assignment algorithm must be used. Such an algorithm was presented in [1][6].

The protocols in [4][9][14][15] combine information from both the network and data link layers. One of several paths to the destination are discovered, regardless of the link bandwidth available on the nodes along those paths. The path bandwidth to the destination is calculated only after the path is discovered. Having to discover the paths to the destination before determining whether the required bandwidth is available along those paths provides for less scalability, less adaptability to fast topology changes, added calculation overhead, and increased message traffic. In [7][8], this combined approach is also used. The authors took two existing on-demand routing protocols, the Ad hoc On-demand Distance Vector Protocol, or AODV [19], and Temporally Ordered Routing Algorithm, or TORA [17][18], and modified them to perform scheduling and resource reservation for time-slotted data link control mechanisms, such as TDMA. Although the focus of that work is on bandwidth reservation within a TDMA framework, this technique can be extended to other data link layer types. The protocols in [7][8] use some of the scheduling mechanisms presented in [15]. However, their approach is different from those in the above protocols in that they incorporate QoS path finding based on bandwidth-scheduling mechanism into already existing ad hoc non-QoS routing protocols, AODV and TORA. Their routing algorithms add several messages and procedures to those protocols to support QoS path reservation and release.

Liao and Tseng present a ticket-based protocol for CDMA-over-TDMA for ad hoc networks [13]. It is a multi-path QoS routing protocol for finding a route with bandwidth constraints in a MANET. As opposed to the proactive routing protocol in [3], their protocol is based on an on-demand process to search for a QoS route, so no global link state information has to be collected in advance. The protocol in [13] can flexibly adapt to the status of the network by spending route-searching overhead only when the bandwidth is limited and a satisfactory QoS route is difficult to find.

As opposed to the CDMA-over-TDMA model used in [13][14][15], this paper assumes the simpler model of TDMA environment. This model is less costly for implementation. However, the bandwidth calculations would be further complicated by the hidden and exposed terminal problems. In [12], Liao and Tseng proposed a bandwidth reservation protocol for QoS routing in TDMA-based MANETs, which considers the hidden and exposed terminal problems. However, that paper along with the other papers mentioned above did not address the issue of racing conditions and parallel reservation conflicts. Such problems arise in MANETs and become more significant with higher traffic loads and increased node density, and mobility [7][8][12].

III. Background and Current Research

The networking environment that we assume in this paper is TDMA-based. In this environment, a single channel is used to

communicate between nodes. The TDMA frame is composed of a control phase and a data phase [3][15]. Each node in the network has a designated control time slot, which it uses to transmit its control information. However, the different nodes in the network must compete for the use of the data time slots in the data phase of the frame.

Liao and Tseng [12] show the challenge of transmitting and receiving in a TDMA single channel environment, which is non-trivial. The hidden and exposed terminal problems make each node's allocation of slots dependent on its 1-hop and 2-hop neighbor's current use of that slot. This will be explained in a detailed example given in a following section. The model we use in this paper is similar to that used by Liao and Tseng, but includes modifications to support our protocol. Each node keeps track of the slot status information of its 1-hop and 2-hop neighbors. This is necessary in order to allocate slots in a way that does not violate the slot allocation conditions imposed by the nature of the wireless medium and to take the hidden and exposed terminal problems into consideration. Below are the slot allocation conditions which are discussed in detail in [12].

A. Slot allocation conditions

A time slot t is considered free to be allocated to send data from a node x to a node y if the following conditions are true [12]:

1) Slot t is not scheduled for receiving or transmitting in neither node x nor y.
2) Slot t is not scheduled for receiving in any node z that is a 1-hop neighbor of x.
3) Slot t is not scheduled for sending in any node z that is a 1-hop neighbor of y.

The protocol we use is similar to that used in [12] but with modification which solves the race conditions, which we will discuss in detail later in this paper. The protocol is on-demand, source based and similar to DSR [18]. Its on-demand nature makes it generally more efficient, since control overhead traffic is only needed when data communication between nodes is desired.

When a node S wants to send data to a node D with a bandwidth requirement of b slots, it initiates the QoS path discovery process. Node x, which is the source node, determines if enough slots are available to send from itself to at least one of its 1-hop neighbors, and if so, then broadcasts a $QREQ(S, D, id, b, x, PATH, NH)$ to all of its neighbors. The message contains the following fields:

1) S: ID of the source node.
2) D: ID of the destination node.
3) id: Message ID. The (s, D, id) triple is therefore unique for every QREQ message and is used to prevent looping.
4) b: Number of slots required in the QoS path from S to D.
5) x: The node ID of the host that is forwarding this QREQ message.
6) $PATH$: A list of the form $((h_1, l_1), (h_2, l_2), ..., (h_k, l_k))$. It contains the accumulated list of hosts and time slots, which have been allocated by this QREQ message so far. h_i is the ith host in the path, and l_i is the list of slots used by h_i to send to h_{i+1}.
7) NH: A list of the form $((h_1^{'}, l_1^{'}), (h_2^{'}, l_2^{'}), ..., (h_k^{'}, l_k^{'}))$. It contains the next hop information. If node x is forwarding this QREQ message, then NH contains a list of the next hop host candidates. The couple $(h_i^{'}, l_i^{'})$ is the ID of the host, which can be a next hop in the path, along with a list of the slots, which can be used to send data from x to $h_i^{'}$.

Each node maintains and updates three tables, ST, RT and H. At a node x, the tables are denoted by ST_x , RT_x and H_x. The tables contain the following information:

- $ST_x[1..n, 1..s]$: This is the send table which contains slot status information for the 1-hop and 2-hop neighbors. For a neighbor i and slot j, $ST_x[i, j]$, can have one of the following values representing three different states: 0 - for free, 1 - for allocated to send, 2 - for reserved to send.
- $RT_x[1..n, 1..s]$: This is the receive table which contains slot status information for the 1-hop and 2-hop neighbors. For a neighbor i and slot j, $RT_x[i, j]$, can have one of the following values representing three different states: 0 - for free, 1 - for allocated to receive, 2 - for reserved to receive.
- $H_x[1..n, 1..n]$: This table contains information about node x's 1-hop and 2-hop neighborhood. If an entry $Hx[i, j]$ is 1, this means that node i, which is a 1-hop neighbor of node x, has node j as a neighbor; an entry of infinity indicates that it does not.

Let z_1 and z_2 be two 1-hop neighbors of a node y. Note that, according to the slot selection rules stated earlier, a slot t that is available to send from y to z_1 is not necessarily available to send from y to z_2. This is because the slot could be free to send and receive in y's ST and RT tables, and all 1-hop neighbors of z_1 are sending and not receiving in slot t, but not all 1-hop neighbors of z_2 are sending and not receiving in t.

The QREQ message is forwarded by the intermediate nodes that are able to allocate b slots to send data and can therefore be a part of the QoS path that is being discovered and reserved.

As the QREQ message propagates from the source to the destination, the slot reservation information is not updated in the ST and RT tables. This unconfirmed reservation information is only maintained and updated in the QREQ message as propagates through the nodes. The status of the corresponding slots in the ST and RT tables in the nodes continues to be $free$. This can lead to multiple reservations of the same slots by different QREQ messages due to a race condition, which is explained later in this paper. If and when the QREQ message arrives at the destination node D, then indeed, a QoS path to send data from S to D with b slots in each hop was discovered. In this case, the destination D replies by unicasting a $QREP(S, D, id, b, PATH, NH)$ back to the source, which confirms the path that was allocated by the corresponding QREQ message. The QREP message propagates from D to S through all of the intermediate nodes that are specified in PATH. PATH contains a list of the nodes along the discovered

path along with the slots which were allocated for this path at each node. As the QREP message propagates through the intermediate nodes, each node updates its ST and RT tables with the slot reservation information in the QREP message and changes the status of the corresponding slots to $reserved$. This represents the confirmation of the reservation of the slots for the discovered path.

B. Race condition and parallel race condition

Race condition This condition occurs when multiple reservations happen simultaneously at an intermediate node. Consider the situation in Figure 1. When a node B receives QREQ1 (with b slots required) from node A to node C, it allocates b slots and forwards the request. Let slot t be among these allocated slots. Before B receives the reply message, QREP1, which would confirm the QoS path reservation from node C to A and reserve the allocated slots, it is possible that another request, QREQ2, can arrive at node B. QREQ2 from node D requests to reserve another path from node D to node E passing through node B. In the algorithm in [12], node B would potentially go ahead and allocate one or more of the same slots, including slot t in this example, for the other request, QREQ2, for the path from D to E. When the reply message, QREP1, arrives at B to confirm the QoS path reservation from C to A, node B will go ahead and confirm these allocated slots, including slot t, and mark them as reserved in its ST and RT tables. Later, when the other reply message, QREP2, arrives at node B to confirm the QoS path from D to E, node B will potentially again reserve the same slots, including slot t in this example, for the second QoS path. Therefore, due to this race condition, the same slot t was reserved for two different QoS paths. This would create a conflict when the source nodes start using these reserved QoS paths to send data.

The conflict arises when the packets are transmitted from A to C and D to E simultaneously, and two data packets from two different paths arrive at node B. In this case, node B must decide which data packet it will actually send. The other data packet will be dropped. In this case, node B can, if the protocol requires, inform the other source of this error condition, or the source would simply time out the request. The corresponding source must then start the process of trying to reserve a new QoS path all over again. This leads to a decline in the throughput. In this paper, we propose to fix this problem which we call the *race condition* due to multiple reservations at an intermediate node.

Parallel reservations problem. Consider the situation in Figure 2. In this case, we have two parallel paths, ABCD and EFGH, that are being reserved. Two or more of the intermediate nodes belonging to the two parallel paths are 1-hop neighbors. In this case node B, which belongs to the first path, and node F, which belongs to the other path are 1-hop neighbors. This is indicated in the figure using the dashed lines. The same relationship exists between nodes C and G. When the QREQ1 is propagating from node A to D, the slots are allocated at the intermediate nodes. However, if the slot allocation information is not maintained by the nodes, say node B here, but is only placed in the QREQ1 message, then

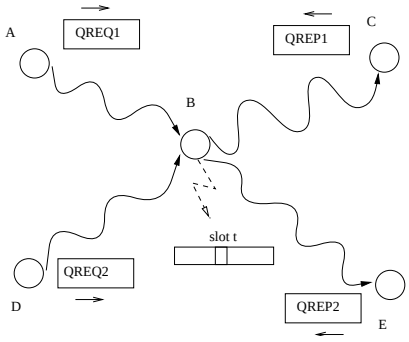

Fig. 1. Multiple QoS path reservation competition.

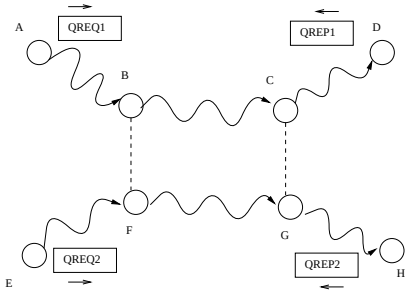

Fig. 2. Parallel reservation problem at nodes B and F (similarly, at nodes C and G).

no memory of this allocation is kept by the node, as is the case in [12]. This can cause another type of race condition, which we call the parallel reservation problem. This problem arises if, before QREQ1 propagates and is confirmed, the same process occurs with QREQ2 and node F allocates slots for the other QoS path and does not take into consideration the allocation of slots for QREQ1 at node B.

If both QREQ messages are successful in reserving their corresponding paths, a potential problem exists because the slot allocations at nodes B and F can be violating the slot allocation conditions mentioned earlier in this paper. Nodes B and F each did the allocation based on information which did not consider the other 1-hop neighbor node's slot allocation for the corresponding parallel path being reserved. Again,

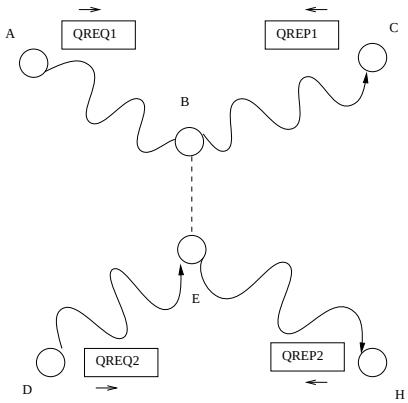

Fig. 3. Parallel reservation problem at nodes B and E.

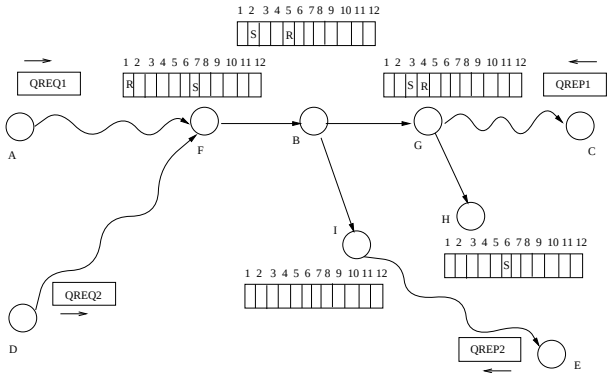

Fig. 4. Multiple QoS path reservation competition. R: scheduled to receive. S: scheduled to send. empty: not scheduled to receive or send.

if the two parallel paths are reserved successfully and data transmission is started along these paths, collisions will occur at the 1-hop neighbors belonging to the different parallel paths. In this example, nodes B and F would experience this collision in their transmissions. A similar situation can occur between any 1-hop neighbors belonging to the two parallel paths, for example, between nodes C and G of the same figure.

It is important to note that this parallel reservation problem can occur in any situation where the two paths have 1-hop neighbors, with each belonging to the other path. This would also be the case in the example presented in Figure 3, where nodes B and E are 1-hop neighbors who belong to two different QoS paths. In this paper, we propose an algorithm to fix this problem, which we call the *parallel reservation problem*.

C. A more detailed example

Consider the nodes in Figure 4. Node A wants to request a QoS path to node C with $b = 3$ (i.e. the required bandwidth is 3 slots). Node A sends a QoS request, QREQ1, to reserve the path. The QREQ message travels through the nodes on its way to C and arrives at node B. We see that node B has nodes F and G as 1-hop neighbors, and node G has node B and H as 1-hop neighbors. Node B will now try to allocate slots for this arriving QREQ1 message to send to each of its 1-hop neighbors, if there are b slots available to send from itself to this neighbor. It will calculate the number of slots available to each of those neighbors and will place those neighbors along with the allocated slots in the next hop list (NH). Node B will then include the next hop list (NH) in the QREQ1 message before it broadcasts (forwards) it.

Let's consider the process of calculating the number of slots available to send from node B to its 1-hop neighbor, node G. Node B has slot allocation information for itself and for all of its 1-hop and 2-hop neighbors including node G. Node B realizes that it cannot allocate slots 2 and 5, because they are scheduled by node B itself, to send and receive (slot selection rule 1). It also realizes that it cannot allocate slots 3 and 4, because they are scheduled to send and receive in node G (slot allocation rule 1). It cannot use slot 1 because it is scheduled to receive in one of its 1-hop neighbors, node F (slot allocation rule 2). Note that this is due to the hidden terminal problem; if node B sends to G using slot 1, this will cause a collision at node F which is using slot 1 to receive as well. Furthermore, node B cannot use slot 6, because it is scheduled to send in node H, which is a 1-hop neighbor of the node it intends to send to, node G (slot allocation rule 3). Note that this is another example of the hidden terminal problem, because if node B sends to node G using slot 6, it will cause a collision at node G. However, node B can use slot 7 to send to node G even though it is scheduled to send in node F. This is the exposed terminal problem. In fact, it would be more desirable for node B to allocate this slot to send to node G; this would increase channel reuse, a desired goal in wireless communications. Therefore, this leaves slots 7 through 12 which are free to send from node B to node G in this example.

After the calculation above, node B allocates slots 7, 8, and 9 to send from itself to G. It includes G in its next hop list NH along with the list of the slots 7, 8, and 9. It then broadcasts the QREQ1 message. In [12], node B does not keep track of this allocation which is only remembered in the forwarded QREQ1 message. So, until node B receives the corresponding QREP1 message which will be propagated from the destination C, slots 7, 8, and 9 in node B will remain $free$. They will only change status from $free$ to $reserved$ when and if the corresponding QREP1 message arrives from node C on its way to node A to confirm the QoS path A..FBG..C slot reservations. This poses no problem so long as no other requests arrive at node B during the period between forwarding QREQ1 and receiving the reply message QREP1. However, consider a situation where another request, QREQ2, arrives at node B from a source node D trying to reserve a QoS path from itself to node E with b=3 (i.e. the required bandwidth is 3 slots). Node B in this case will look at its slot status tables and will see no allocation for slots 7 through 12. It will then proceed to allocate some of these slots for this newly requested path. If the corresponding slot allocation procedure allocates slots 7, 8 and 9 for this new path and includes them in the next hop list, NH, then Node B will broadcast (forward) QREQ2 to node I which is on the path to node E. When QREP1 arrives at node B, it will change the status of slots 7, 8 and 9 to reserved. Afterwards, QREP2 will arrive at node B from node E on its way to node D. Node B will then have the problem of double allocation of slots 7, 8 and 9. In [12] the slots are reserved again (double reservation) for the second path. This will lead to a conflict at node B when data transmission using the two different paths starts. This is a multiple reservation problem due to a race condition at node B.

A similar example can be shown for the parallel reservation problem. This was described in Figure 3 where node B would select the slots to forward QREQ1 by considering only the status of the slots in node E prior to the allocation done by node E for the slots for QREQ2. When QREP1 returns to node B and QREP2 returns to node E, they both reserve the allocated slots. These slot selections can be in violation of the slot allocation rules and result in collisions when data transfer using the two different QoS paths begin.

IV. Our Protocol

In order to solve the race conditions described earlier and enhance network performance, especially in situations of increased node mobility, increased node density and higher traffic loads, our protocol uses a more conservative strategy. This strategy is implemented using the following features:

1) Three states for each slot in the ST and RT tables described earlier: $reserved$, $allocated$, $free$. The three states are defined in the following manner: $Free$: not yet allocated or reserved. $Allocated$: in process of being reserved, but not yet confirmed. This means that the slot is allocated by a QREQ message but the corresponding QREP message has not yet arrived to confirm the reservation. $Reserved$: reservation is confirmed and the slot can be used for data transmission.
2) As the QREQ message propagates from source to destination, slot status is changed from $free$ to $allocated$ in the intermediate nodes. Therefore, we maintain this information in the ST and RT tables of the nodes as opposed to only preserving this information in the QREQ message with no memory of it in the nodes as is the case in [12]. As the QREP message propagates from the destination to the source the corresponding slot status in the nodes is changed from $allocated$ to $reserved$.
3) Wait-before-reject at an intermediate node with three conditions to alleviate the multiple reservation at intermediate node problem. ($conditon$ 1: all required slots are available, $condition$ 2: not-now-but-wait, and $condition$ 3: immediate drop or reject of QREQ).
4) TTL timer for allocated and reserved slots.
5) TTL timers for maximum total QREQ propagation delay allowed, and for maximum total QREQ/QREP delay allowed (i.e. maximum QoS path acquisition time).

The following is an overview of the protocol. When a source node S wants to reserve a QoS path to send data to a destination node D, it sends the $QREQ(S, D, id, b, x, PATH, NH)$ message which was described earlier. If and when the QREQ message reaches node D, then this means that there was a QoS path from S to D which was discovered, and there were at least b free slots to send data from each node to each subsequent node along the discovered path. These slots are now marked as $allocated$ in the corresponding nodes (in the ST and RT tables). In this case, node D unicasts a $QREP(S, D, id, b, PATH, NH)$ message, which was also described earlier, to node S. This message is sent along the nodes indicated in $PATH$. As the QREP message propagates back to the source node, all of the intermediate nodes along the allocated path must confirm the reservation of the corresponding allocated slots (i.e. change their status from $allocated$ to $reserved$). The timing and propagation of the QREQ and QREP messages are controlled by timers, a queueing process, and synchronous and asynchronous slot status broadcasts, which we discuss in detail later in the paper.

A. Wait timers

We define the following timers, which control the allowable delay of the propagation of the QREQ and QREP messages through the system. These timers can be initialized to a tunable value which can vary according to the requirements of the application being used. It is also possible to disable some of these timers, which are specified below, if the application does not have such delay requirements.

TTL_allocated_slot_time. Each slot t in ST and RT tables has a TTL_t (Time to Live) count down timer associated with it. This TTL_t timer is only needed when the slot is set from free to allocated. As soon as a slot is converted from free to allocated, its TTL timer gets set to a certain time to live parameter. This is a tunable parameter, which can be determined according to the application needs. The TTL_t timer is set to 0 upon initialization and when the slot becomes free. When the status of a slot t is changed from free to allocated due to a QREQ, which is processed by the node, the TTL_t timer is initialized to a predetermined $TTL\_allocated\_slot\_time$. This time should be at least equal to the RTT (Round Trip Time) for a QREQ to come back as a QREP. This time is a tunable parameter which can be fixed according to the application requirements and/or the network size and/or density. It can be increased with a larger number of nodes in the network. A reasonable value could be $2*RTT$, but it could be set to a smaller or larger value depending on the size and propagation delay characteristics of the network involved.

A large value for this TTL_t timer corresponds to a conservative strategy. If it is too large, a slot would have to wait too long to automatically convert back to free. That lengthens the path acquisition time for a QREQ, which might not be desirable in certain applications. On the other hand, if the TTL time is too small, then a node is too anxious to return allocated slots to free status before the reservation is confirmed with a QREP message. This creates a risk of converting a slot back to free status too soon. After a short amount of time, the corresponding QREP message of the QREQ message that initially allocated this slot comes back. However, this slot which was changed to free can now be allocated for another path. This way, double allocation of the same slot exists for two different paths, and this leads to a racing condition, the very condition the protocol strives to avoid.

TTL_reserved_slot_time. When a slot is reserved (i.e. its allocation is confirmed and it is in $reserved$ status) for a particular QoS path, it must be used for actual data transmission within a certain time out period which we define as the $TTL\_reserved\_slot\_time$. This time is a parameter which can be set according to the application and network environment involved. If at any time a slot is not used for data transmission for more than this time, it must be returned to free status. This is done in the following manner. The associated timer is refreshed each time the slot is used for data transmission. The timer is constantly counted down. If this timer reaches zero at any time then the slot is returned back to $free$ status. This timing is also useful for a situation where the QREP message used to confirm slot reservation is successful in propagating from the destination through some nodes but then is not forwarded to the source. In this case, the nodes which already confirmed the reservation of their slots

will still be able to return these slots back to free status after this time out period.

Max_QREQ_node_wait_time. The QREQ can wait at an intermediate node for a maximum amount of time $Max\_QREQ\_node\_wait\_time$. This is a parameter that is set to a tunable value according to the application and network requirements and characteristics. A reasonable value can be equal to $2 * RTT$. Its effect is similar to what was described earlier in the $TTL\_allocated\_slot\_time$ section. Namely, it can vary according to a conservative or aggressive strategy. Also it depends on the size and propagation delay characteristics of the network. Furthermore, this time affects the QoS path acquisition latency which might be might be limited depending on the application involved.

Max_QREQ_tot_wait_time. Another related delay type is the QREQ total wait time. This is the maximum allowable cumulative wait delay for the QREQ as it propagates through the network. This delay is controlled by the timer $max\_QREQ\_tot\_wait\_time$. This timer is decremented at each node according to the time the QREQ had to wait at that node, and it is forwarded along with the QREQ to the next node.

Max_QREQ_QREP_tot_wait_time. A third timer can be defined as $Max\_QREQ\_QREP\_tot\_wait\_time$. This is the total time for path acquisition ($QREQ\ propagation$ + $QREP\ propagation$); this time is also decremented by each node accordingly and forwarded along with the corresponding QREQ and QREP as they propagate through the system. Whenever a node is forwarding a QREQ or a QREP message, it checks this time. If it is zero, then this means the QoS path reservation process has taken longer than the maximum allowable time and the corresponding QREQ or QREP message should now be dropped. Furthermore, the protocol can also take one of the following actions: (1) Send a notification message to all of the nodes along the reserved path (the nodes which forwarded the QREP message from the destination to this node) to return the corresponding slots which have been allocated and/or reserved by this path to free status. Or (2) Let those already-reserved-slots time out to free status as described by the $TTL\_reserved\_slot\_time$ defined earlier.

The $Max\_QREQ\_node\_wait\_time$, $Max\_QREQ\_tot\_wait\_time$, and $MAX\_QREQ\_QREP\_tot\_wait\_time$ timers are optional and can be set to different values, according to their importance and/or criticality in the application that is being used.

Similar timing techniques can be employed for the transmission of data packets as well. Timing might be even more significant as a requirement and in its effect over the performance of different applications, such as multimedia, voice, and video. Such applications are known to have strict requirements on the total delay permitted for a data packet. This is due to the fact that the packet can hold voice or video frames that must be delivered within a certain amount of time beyond which they become useless and must simply be discarded.

B. Status broadcasting and updating

There are two types of node status broadcasts: synchronous (periodic) and asynchronous.

Synchronous periodic status updates. Each node broadcasts its slot allocation status (the ST and RT table information updates) to its 1-hop and 2-hop neighbors (i.e. with a 2-hop TTL). This broadcast is done periodically (synchronously) according to a predetermined periodic slot status update frequency. We define this as $periodic\_status\_update\_time$. These periodic updates enable the nodes to maintain updated neighborhood information as nodes come within or go out of their range. Furthermore, these updates inform the node of its neighbor's slot status information on a periodic basis.

When a node does not receive any synchronous (periodic) or asynchronous (due to changes in slot status) updates from a neighbor after a time out period, which we call Status_update_tot, it will assume that this node is no longer one of its 1-hop or 2-hop neighbors, and will delete that neighbor from its ST and RT tables.

Asynchronous status updates. The status update is done asynchronously as the status of slots is changed from free to allocated, or from allocated to reserved. There is no need to inform the neighbors of the change from allocated to free which results from TTL timer expiration. The neighbors will count down the time of the allocated slots as well and will change them to free status (i.e. will assume that the corresponding neighbor node will have done that) if no reservation change is indicated from the corresponding neighbor node. Note that the status updates are done with a 2-hop TTL flood to the 1-hop and 2-hop neighbors.

The asynchronous updates of receive and send slot status with the three state information which includes the $allocated$ status, solves the parallel reservation problem stated earlier in the paper, and eliminates the associated race condition which is caused by it; this was not done in previous research. When the 1-hop neighbor receives a separate and different QREQ, it will now be aware of the $free/allocated/reserved$ status of its neighbors' slots, rather than just their $free/reserved$ status. This way, it will consider only slots which are totally free according to slot selection and will prevent the related race condition from occurring. This consideration is done in the $select\_slot()$ function which is described later.

C. The main algorithm at an intermediate node

When a node y receives a broadcasting message $QREQ(S, D, id, b, x, PATH, NH)$ initiated by a neighboring host x, it checks to determine whether it has received this same source routed request (uniquely identified by (S, D, id)) previously. If not, y performs the following steps. If y is not a host listed in NH then it exits this procedure. Otherwise, it calculates the values of the variables $NUyz$, $ANUyz$, and Fyz, which are define in the following manner:

- $NUyz$: The number of slots that are not-usable for sending from y to z. This means that there exists at least one confirmed reservation at y or its neighbors, which does not allow slot t to be used from y to send to z. This

is due to any violation of any of the three slot allocation conditions.

- $ANUyz$: The number of slots that are allocated-not-usable for sending data from y to z. A slot is called ANU (allocated-not-usable) if there exists totally allocated reservations at y or its neighbors, which do not allow slot t to be used from y to send to z. This could be due to any violation of any of the three slot allocation conditions. However, these violations of any of the lemma conditions are only and totally due to pure allocations (not confirmed reservations) at y and/or its neighbors.
- Fyz: The number of slots that are free at a node y to send to a node z respectively. This means that this slot is currently completely available to be used for sending from node y to node z and therefore satisfies all three of the slot selection conditions.

Therefore, at node y, we have to determine a separate set of $NUyz$, $ANUyz$, and Fyz for each neighbor z of y. When a node y receives a QREQ message from a node x, it uses algorithm 1 which is shown below to forward the message, or to insert it in the $QREQ\_pending\_queue$, or to drop it.

Algorithm 1, which is shown below, is used to fix the race conditions stated earlier. It works in the following manner. When a QREQ message arrives at a node y from a node x, it does the following. First, it uses three routines to calculate $NUyz$, $ANUyz$, and Fyz from ST and RT tables. Note that calculating these values would have taken into account all three of the slot selection conditions.

The algorithm first updates the ST and RT tables with the information in PATH. Then the algorithm initializes the next hop list NH_temp to empty, and then attempts to build it by adding to this list each 1-hop neighbor z of y which has b slots free to send from y to z. The algorithm uses the select_slot function which takes into account the three slot allocation conditions mentioned earlier and the information in the updated ST and RT tables. There are three possible conditions that can take place.

If at least one neighbor z of y has b slots free to send from y to z, we call this $condition1$, then the NH_temp list will not remain empty and the node y will broadcast (i.e. forward) the QREQ message after incorporating the node x and the list li' (i.e. the list of slots used to send from x to y) PATH (using $PATH\_temp = PATH \mid (x, li')$). Here, $\mid$ means concatenation.

Otherwise, if the NH_temp list is empty after checking all of the neighbors, then that means that there are no neighbors z of y which have b slots free to send from y to z according to the slot selection conditions. At this point, the algorithm tries to determine if there is any "hope", i.e., if there is at least one 1-hop neighbor z of y which has the condition $(Fyz+ANUyz) \geq b$. This would be $condition2$. In this case, the algorithm checks if the maximum time left for the required allocated slots to become free (or reserved) does not exceed the maximum total wait time left for this QREQ message ($Max\_QREQ\_tot\_wait\_time$), then this QREQ message is placed in the $QREQ\_pending\_queue$. This queue will be scanned each time a slot becomes free to see if at that point, the QREQ message can be forwarded. This queue will be discussed in more detail later in this paper. If on the other hand, no 1-hop neighbor z of y has a condition of $(Fyz + ANUyz) \geq b$ then there is "no hope" at the current time. Therefore, the QREQ message is dropped.

Algorithm 1 The main algorithm at an intermediate node

When a node y receives a QREQ message
  Update the ST and RT tables with the information in PATH
  $NH\_temp = \phi$
  for each 1-hop neighbor node z of y **do**
    $NHyz = calcR(z, ST, RT)$
    $ANUyz = calcA(z, ST, RT)$
    $Fyz = calcF(z, ST, RT)$
    if $Fyz \geq b$ **then**
      $L = select\_slot(y, z, b, ST, RT)$
      if $L \neq empty$ **then**
        $NH\_temp = NH\_temp(z, L) \mid (z, L)$
      else
        Error: cannot have $Fyz \geq b$ and $L = empty$
      end if
    end if
  end for
  if $NH\_temp \neq \phi$ **then**
    Let ($h_i^{'}$, $l_i^{'}$) be the entry in NH such that $h_i^{'}$=y
    let $PATH\_temp = PATH \mid (x, l_i^{'})$
    broadcast $QREQ(S, D, id, b, x, PATH\_temp, NH)$ message
  else
    for each 1-hop neighbor node z of y **do**
      if $(Fyz + ANUyz) \geq b$ **then**
        let t_{mas} = $maximum\ time\ left\ for\ required\ allocated\ slots\ to\ become\ free$ (*or reserved*)
        if $max\_QREQ\_tot\_wait\_time \geq t_{mas}$ **then**
          insert QREQ message in $QREQ\_pending\_queue$
          exit this procedure
        end if
      end if
    end for
  end if
  Drop QREQ message

D. The select_slot function

The $select\_slot(y, z, b, ST, RT)$ function will return a list of slots that are available to send from node y to z. It will do so according to the slot allocation rules stated previously, and the slot status information which is in the updated ST and RT tables. $select\_slot()$ will return an empty list if b slots are not available to send from node y to z.

E. The QREQ_pending_queue

The QREQ's that are waiting for slots to become free are placed in a $QREQ\_pending\_queue$. While waiting for the status of the different slots in the table to change, some slots will be freed and others will be confirmed. Every time a change in slot status is done (due to timer expiration, or confirming a reservation), the queue is scanned.

Scanning the $QREQ\_pending\_queue$. Every time the queue is scanned, all QREQ messages, which have any of their corresponding wait timers expired, are deleted from the queue. These timers are: Max_QREQ_node

$\_wait\_time$, $Max\_QREQ\_QREP\_tot\_wait\_time$, and $Max\_QREQ\_tot\_wait\_time$. Also, for each QREQ in the queue, the new values for Fyz, $ANUyz$, and $NUyz$ are calculated, and it is determined under which conditions the new QREQ status falls. There are three possibilities:

- Changed to condition 1 (i.e. now $Fyz \geq b$): In this case, forward the pending QREQ and delete the QREQ from the $QREQ\_pending\_queue$.
- Changed to condition 2 (i.e. now $(Fyz + ANUyz) \geq b$): In this case, leave the corresponding QREQ in the $QREQ\_pending\_queue$.
- Changed to condition 3 (i.e. $(Fyz + ANUyz) < b$): In this case, delete the corresponding QREQ from the $QREQ\_pending\_queue$ (i.e. drop this QREQ message). Here another policy can be adopted which would be to send a reject message back to the source of the QREQ to inform it of the rejection if the protocol requires informing the source nodes of the failing QREQ.

If the TTL for an allocated slot expires, this means that the slot has been allocated for $too\ long$ and not confirmed (i.e. reserved) by a QREP message. In this case, the corresponding slot status in ST and and RT tables is set to $free$.

If the status of a QREQ message in the queue changes into condition 1, then the algorithm calls the $select\_slot()$ function for all nodes that are 1-hop neighbors of y. It then builds the next hop list accordingly, which will include every neighbor node z, for which there are b slots available to send from y to z, and the list of these slots. This is done using Algorithm 2.

Algorithm 2 Forwarding the QREQ message from the $QREQ\_pending\_queue$

$NH\_temp = \phi$
for every 1-hop neighbor z of y **do**
  $L = select\_slot(y, z, b, ST, RT)$
  if $L \neq \phi$ **then**
    $NH\_temp = NH\_temp \mid (z, L)$
  end if
end for
if $NH\_temp \neq \phi$ **then**
  let ($h_i^{'}$, $l_i^{'}$) be the entry in NH such that $h_i^{'}$=y
  let $PATH\_temp = PATH \mid (x, l_i^{'})$
  broadcast $QREQ(S, D, id, b$,y$, PATH_temp, NH_temp)$
  delete QREQ message from the $QREQ\_pending\_queue$
end if

F. How our protocol solves the race conditions

Our protocol solves the race conditions stated earlier in the following manner.

Solving the race condition. Consider the example of Figure 4, which was presented earlier. The algorithm does not have the race condition due to multiple reservations at an intermediate node. This is due to the fact that each slot has three states $free$, $allocated$ and $reserved$ as mentioned earlier. Specifically, when node B makes the calculation of the slots available for transmission to node G, it will consider only slots with free status. Before forwarding QREQ1, node B will designate slots 7, 8, and 9 as $allocated$ (not yet fully $reserved$, but not $free$ neither) . When QREQ2 arrives at node B, it will consider only slots of $free$ status and will therefore allocate slots 10, 11, and 12 for the second path. When QREP1 arrives, it will confirm the reservation of slots 7, 8 and 9 and will convert them to $reserved$ status. When QREP2 arrives, it will also confirm the reservation of slots 10, 11, and 12 and convert them to $reserved$ status. When data transmission starts for both paths, there will be no conflict at node B.

Another possibility is that QREQ2 arrives at node B and slot 10 was allocated by another path but slots 11 and 12 are still free. Then QREQ2 will not be discarded because the number of free slots + the number of allocated slots is less than or equal to b. It will wait in the $QREQ\_pending\_queue$ until either the allocated slots time out (fail to be confirmed before a time out period) or are confirmed. In the first scenario, QREQ2 will proceed from B to E, and in the second scenario it will be discarded.

Solving the parallel reservation problem. Our protocol does not have the parallel race condition problem, which was illustrated in the examples in Figure 3 and Figure 2. Consider the example in Figure 3. When node B allocates slots for QREQ1 to reserve them for the A..B..C path, it must immediately, due to the asynchronous status updates, broadcast the slot status information to all of its 1-hop and 2-hop neighbors which include node E. So, when node E receives QREQ2 (before QREP1 comes back to node B), it will do the slot allocation for the D..E..H path with free slots only (and later confirm them with the QREP2 message) based on complete and up-to-date slot status information. Therefore, node E will allocate only slots which are not at risk of being in violation of these conditions even when QREP1 comes back to node B and confirms the slots reserved for the first path (A..FBG..C). Consequently, there will be no collisions between nodes B and E when the data transfer begins along the two separate and parallel paths A..B..C and D..E..F. A similar analysis can be done for the example in Figure 2 which would find no risk of collisions due to parallel and simultaneous reservations for the two separate and parallel paths A..B..C..D and E..F..G..H. In previous protocols, collisions would have taken place between transmissions of nodes B and F on one hand and nodes C and G on the other hand.

G. Network performance improvements

The effectiveness and impact of this conservative strategy of asynchronous status updates and three-state slot status will become increasingly significant as both the density and mobility of the nodes in the network increase. Since the race condition is more prevalent and costly in both dense networks and those with increased node mobility, the increased communication overhead of the asynchronous updates will be considerably offset by the performance gains resulting from the elimination of the race condition.

This conservative strategy of the asynchronous status updates with three-state slot status will be more effective and have more significant impact as the density of the nodes in the network increases, and as the mobility of the nodes increase as well. This is due to the fact that the price paid by the increased

communication overhead of the asynchronous updates will be more significantly offset by the payoff in the elimination of the race condition, since the latter is more prevalent and costly in more dense networks and with increased node mobility [7][8][12].

These results of the stronger payoff of conservative strategies is supported by and in concert with the usual case in research where conservative strategies work better with stressed network conditions, such as increased traffic. On the other hand, the more optimistic strategies work better for light loads and light conditions and worse under heavier traffic loads. An example of this would be the case with token ring networks, which uses a conservative strategy. Nodes can only transmit when they acquire the token. Conversely, Ethernet networks adopt a less conservative or optimistic strategy. A node transmits as needed, and when collisions occur, the node backs off and tries again later. It is common knowledge that token ring networks, with increased overhead, more controlled transmission and conservative strategies have better performance under heavy traffic load conditions as opposed to Ethernet networks, with less overhead, less controlled transmissions, and less conservative strategy, which work better under lighter traffic load conditions. We believe that the same relationships apply in the case of QoS routing in ad hoc wireless networks. Consequently, we are applying those principles in this paper to improve the performance of the ad hoc networks under more stressed network conditions.

V. Conclusions and Future Work

In this paper, we presented a protocol for TDMA-based bandwidth reservation for QoS routing in mobile ad hoc networks. This protocol remedies the race condition which is not addressed in current research. The algorithm relies on the maintenance of three-state slot status information $(free/allocated/reserved)$ at each node, synchronous and asynchronous slot status updates to 1-hop and 2-hop neighbor nodes, wait-before-reject strategy, TTL timers for allocated slots, maximum QREQ node wait time, and max QREQ/QREP total wait time. In addition, this algorithm provides a solution to the parallel reservation problem in QoS routing, which was not addressed in previous research. In the future, we intend to improve this protocol further by applying more techniques in optimizing the selection of the next-hop neighbors, and introducing more delay control. We are also considering the application of a ticket-based approach used in [13] to control the number of next hop neighbors selected by a node and to provide the possibility of a multi-path QoS routing using our protocol. In addition, we intend to study, and analyze the performance of the protocol through simulation.

References

[1] A. A. Bertossi and M. A. Bonuccelli. Code assignment for hidden terminal interference avoidance in multihop radio networks. *IEEE/ACM Trans. on Networks*, 3(4):441–449, August 1995.

[2] S. Chakrabarti and A. Mishra. QoS issues in ad hoc wireless networks. *IEEE Communications Magazine*, 39(2):142–148, February 2001.

[3] S. Chen and K. Nahrstedt. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, pages 1488–1505, August 1999.

[4] T.-W. Chen, J. T. Tsai, and M. Gerla. QoS routing performance in mulithop, multimedia, wireless networks. *IEEE 6th International Conference on Universal Personal Communications Record*, 2:557–561, October 1997.

[5] S. De, S.K. Das, H. Wu, and C. Qiao. A resource efficient rt-qos routing protocol for mobile ad hoc networks. *Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on*, 1:257–261, 2002.

[6] J. J. Garcia-Luna-Aceves and J. Raju. Distributed assignment of codes for multihop packet-radio networks. *Proc. of IEEE MILCOM '97*, 1997.

[7] I. Gerasimov and R. Simon. A bandwidth-reservation mechanism for on-demand ad hoc path finding. *IEEE/SCS 35th Annual Simulation Symposium, San Diego, CA*, pages 27–33, April 2002.

[8] I. Gerasimov and R. Simon. Performance analysis for ad hoc QoS routing protocols. *Mobility and Wireless Access Workshop, MobiWac 2002. International*, pages 87–94, 2002.

[9] Y.-K. Ho and R.-S. Liu. On-demand QoS-based routing protocol for ad hoc mobile wireless networks. *Fifth IEEE Symposium on Computers and Communications, 2000. Proceedings. ISCC 2000*, pages 560–565, July 2000.

[10] Y. Hwang and P. Varshney. An adaptive QoS routing protocol with dispersity for ad-hoc networks. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 302–311, January 2003.

[11] M. Ilyas. *The handbook of ad hoc wireless networks*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, Florida, USA, 2003.

[12] W.-H. Liao, Y.-C. Tseng, and K.-P. Shih. A TDMA-based bandwidth reservation protocol for QoS routing in a wireless mobile ad hoc network. *Communications, ICC 2002. IEEE International Conference on*, 5:3186–3190, 2002.

[13] W.-H. Liao, Y.-C. Tseng, S.-L. Wang, and J.-P. Sheu. A multi-path QoS routing protocol in a wireless mobile ad hoc network. *IEEE International Conference on Networking*, 2:158–167, 2001.

[14] C. R. Lin and C.-C. Liu. An on-demand QoS routing protocol for mobile ad hoc networks. *Conference on IEEE International Networks, (ICON 2000) Proceedings*, pages 160–164, Spetember 2000.

[15] C. R. Lin and J.-S. Liu. QoS routing in ad hoc wireless networks. *IEEE Journal on selected areas in communications*, 17(8):1426–1438, August 1999.

[16] S. Nelakuditi, Z.-L. Zhang, R. P. Tsang, and D.H.C. Du. Adaptive proportional routing: a localized QoS routing approach. *Networking, IEEE/ACM Transactions on*, 10(6):790–804, December 2002.

[17] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 3:1405–1413, April 1997.

[18] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, Upper Saddle River, NJ, USA, 2001.

[19] C. E. Perkins and E. M. Royer. Ad hoc on demand distance vector (aodv) routing. *Internet Draft*, August 1998.

[20] J. L. Sobrinho and A. S. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1353–1368, August 1999.

[21] W. Stallings. *Wireless Communications and Networks*. Prentice Hall, 2002.

[22] I. Stojmenovic. *Handbook of wireless networks and mobile computing*. Wiley, 2002.

[23] H. Xiao, K. G. Lo, and K. C. Chua. A flexible quality of service model for mobile ad-hoc networks. *Proceedings of IEEE VTC2000-Spring, Tokyo*, May 2000.

[24] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. A framework for reliable routing in mobile ad hoc networks. *IEEE INFOCOM 2003*, 2003.