A Dynamic Range Resource Reservation Protocol for QoS Support in Wireless Networks

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

***Abstract*— Mobile ad hoc networks (MANETs) provide a powerful and dynamic platform to enable mobile computers to establish communications without an existing infrastructure. In order to provide support for multimedia applications, Quality of Service (QoS) support becomes an important component in their design. Research is being conducted to design routing protocols in MANETs that contain QoS support features and capabilities. However, the research in this area is still in its early stages. In this paper, we present a dynamic range bandwidth reservation protocol for TDMA-based MANETs. In this protocol, a source node S, that needs to send data, sends a request message (QREQ) to reserve a QoS path to the desired destination node D. In the reservation message, the source node specifies a dynamic range [b_{min}, b_{max}] of the number of slots needed to transmit the data. The intermediate nodes along the path try to reserve a number of slots, b_{cur}, that is equal to the maximum number of slots that are "available" within this range ($b_{min} \leq b_{cur} \leq b_{max}$). The protocol also permits intermediate nodes to dynamically "downgrade" existing paths that are functioning above their minimum requirements in order to allow the successful reservation for the maximum number of requested paths. When the network traffic load is later decreased, the existing paths are able to be "upgraded" to function with higher bandwidth requirements that are close or equal to the maximum desired level (b_{max}). This allows the network to admit new QoS paths instead of denying such requests by allowing for "graceful degradation" of other paths. The paper also presents several optimization techniques which are designed to increase the efficiency and throughput of the network. This protocol allows a higher level of flexibility in providing and maintaining QoS support in MANETs.**

Keywords: Mobile ad hoc networks (MANETs), QoS, routing, TDMA, wireless networks.

I. Introduction

Networking and communication systems are rapidly growing in use, sophistication, power, flexibility and mobility. Mobile ad hoc networks (MANETs) are also gaining increased attention in the research community because of the great possibilities they provide in many applications such as conferences, disaster recovery, military systems, as well as other environments that require the establishment of dynamic networks between mobile devices without existing infrastructure [13]. In a MANET, 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. 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 communication protocols to continue to provide multi-hop communication between nodes.

Existing MANET routing protocols provide the capability for establishing multi-hop paths between nodes on a best effort basis [11]. 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 [5].

There are several papers that address the subject of QoS routing in MANETs in different environments and with different models and approaches [12][14]. 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 (CDMA: Code Division Multiple Access) environment. QoS routing protocols for CDMA-over-TDMA based ad hoc networks are considered in other papers [2][4][10]. 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, 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, which must be taken into account. The reader is referred to [5] for a more thorough discussion of these issues.

Jawhar and Wu [7][6] provided a race-free bandwidth reservation protocol for QoS routing in TDMA-based ad hoc networks. It extends the work done by Liao and Tseng [8] which presents a bandwidth reservation protocol for MANETs which does not solve the race condition that can arise when multiple paths sharing intermediate nodes are being reserved simultaneously. This problem can reduce the throughput and efficiency of communications in MANETs as the traffic load and mobility of the network increase. The solution provided by Jawhar and Wu uses three states to control slot release and reservation as opposed to the two-state mechanism presented

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

by Liao and Tseng. This process is discussed later in this paper.

In this paper, we propose a protocol that builds on our previous work in [7] by adding the flexibility of dynamic range reservation. Instead of specifying a fixed number (b) of TDMA-frame data slots, the application layer specifies a range $[b_{min}, b_{max}]$ for the number of slots that are needed for a particular session. The advantages of this added flexibility are that it:

- Provides the network with an ability to adapt the resource reservation process to its traffic load conditions.
- Provides a higher probability of successful allocation of the requested QoS path. More sessions can be admitted by the network and allowed to take place simultaneously thereby increasing the number of users and providing better resource sharing, balancing and utilization.
- Provides the ability of $graceful\ degradation$ to multimedia or real-time applications. The quality of the communication (and corresponding bandwidth usage) can be increased during periods of low network traffic and decreased without disruption (“bend but don’t break policy”) during periods of increased network activity.

The protocol is on-demand, source based and similar to DSR [11]. Its on-demand nature makes it generally more efficient, since control overhead traffic is only needed when data communication between nodes is desired. The protocol works in the following manner. A routing protocol of a source node, S, receives a request from its application layer to establish a communication session (QoS path) with a particular node D. The application layer specifies a dynamic range $[b_{min}, b_{max}]$ for the number of data slots (bandwidth) required for this session. Node S issues a QoS path request (QREQ) message to the destination. As the QREQ message propagates to the destination, it will allocate as many slots as possible within the specified range. An intermediate node will only forward the QREQ message if it is able to allocate at least b_{min} slots. If that is not the case, then it will try to $downgrade$ existing QoS paths that are operating above their minimum in order to free enough slots for the required path. When the destination node D receives the QREQ message, this means that the required path was able to be $allocated$. At this point, the allocation is not yet confirmed and the corresponding slots are not $reserved$. The destination will then unicast a QoS reply (QREP) message back to the source along the intermediate nodes. Each intermediate node will then confirm the reservation of the corresponding slots by changing their status from $allocated$ to $reserved$. When the source node S receives the QREP message, this will then confirm the reservation of the path and start data transmission to the destination. When the network traffic load subsides and different QoS paths are terminated, the source nodes that are still transmitting data will then try to upgrade their existing sessions to operate with more bandwidth. These attempts by the source nodes to upgrade their existing sessions can be done periodically or when they $sense$ that the traffic load in the network is below a certain threshold. More research can be done in this area to enable the nodes to monitor the traffic load status of the network.

The remainder of the paper is organized as follows. Section 2 discusses related work in this field. Section 3 provides background and current research. It also discusses the limitations of existing protocols. In section 4, we present our protocol, and the corresponding algorithms at the source and intermediate nodes. Section 5 contains the algorithms for downgrading and upgrading paths. 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 [2] 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 [10] a calculation algorithm for bandwidth is presented. However, it assumes that neighboring nodes broadcast with different codes, which is the case in the CDMA-over-TDMA model. In that case a code assignment algorithm must be used. Such an algorithm was presented in [1].

The protocols in [3][10] 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 [4], this combined approach is also used. The authors modified two existing on-demand routing protocols, the Ad hoc On-demand Distance Vector Protocol, or AODV [11], and Temporally Ordered Routing Algorithm, or TORA [11], 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 [4] use some of the scheduling mechanisms presented in [10]. However, their approach differs 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 the existing protocols to support QoS path reservation and release.

Liao and Tseng present a ticket-based protocol for CDMA-over-TDMA for ad hoc networks [9]. 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 [2], their protocol is based on an on-demand process to search for a QoS route, so no global link state information must be collected in advance. The protocol in [9] 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.

III. Background and Current Research

In the TDMA environment, a single channel is used to communicate between nodes. The TDMA frame is composed of a control phase and a data phase [2][10]. 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 use of the data time slots in the data phase of the frame.

The paper in [8] shows 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. In the model used in this paper, 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.

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 [8]:

1) Slot t is not scheduled for receiving or transmitting in x or 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.

B. The data structures

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. ST_x and RT_x contain slot status information for the 1-hop and 2-hop neighbors. A slot can have one of the following values representing three different states: 0 - for free, 1 - for allocated to send (receive), 2 - for reserved to send (receive). The H_x table contains information about which nodes are 1-hop and 2-hop neighbors of x. More information about these structures can be found in [7].

C. The previous race-free protocol

As mentioned earlier, the protocol in this paper builds upon the protocol in [7]. The following is an overview of that protocol. When a source node S wants to reserve a QoS path to send data to a destination node D, it sends the QREQ message that contains the source, destination and session ids, with the number of required slots b, and other reservation related information which is described in more detail later in this paper. If and when the QREQ message reaches node D, then this indicates 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 message to node S. This reply 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 ($reserve$) of the corresponding allocated slots. The timing and propagation of the QREQ and QREP messages are controlled by timers, a queueing process, and synchronous and asynchronous slot status broadcasts, discussed in more detail in [7].

IV. The Dynamic Bandwidth Reservation Protocol

A. The algorithm at the source

When a node S wants to send data to a node D with a bandwidth requirement of minimum, b_{min}, and maximum, b_{max}, number of slots, it initiates the QoS path discovery process. S determines if enough slots in the specified range are available to send from itself to each one of its 1-hop neighbors. If this is the case, it then broadcasts a $QREQ$(S, D, id, b_{min}, b_{max}, x, $PATH$, NH) to all of its neighbors. The message contains the following fields:

- S, D and id: IDs of the source, destination and the session. The (S, D, id) triple is therefore unique for every QREQ message and is used to prevent looping.
- b_{min} and b_{max}: The minimum and maximum number of slots required in the QoS path from S to D.
- x: The node ID of the host that is forwarding this QREQ message.
- $PATH$: 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}.
- NH: A list of the form (($h_1^{'}$, $l_1^{'}$, b_{1_cur}), ($h_2^{'}$, $l_2^{'}$, b_{2_cur}), ..., ($h_k^{'}$, $l_k^{'}$, b_{k_cur})). 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 triple ($h_i^{'}$, $l_i^{'}$, b_{i_cur}) is the ID of the host, a list of the slots which can be used to send data from x to $h_i^{'}$, and the current number of allocated slots in the QoS path (so far) from S to D (passing through h_i) as the path is being allocated. Note that there is a separate b_{i_cur} associated with each neighbor, i, in the NH list. This is because the number of available slots to send from node y to each of its neighbors can be different. As the QREQ message propagates through the nodes, each intermediate node will try to reserve b_{cur} slots if it can. If it cannot do so, and is only able to reserve a smaller number of slots $ps \geq b_{min}$ then it sets $b_{cur} = ps$ in the QREQ message before forwarding it to its neighbors.

B. The algorithm at an intermediate node

The QREQ message will propagate to the destination through intermediate nodes. When an intermediate node receives the QREQ message, it checks the tuple S/D/SessionID to see if it previously processed this message. If so, then y will drop this QREQ message. This is done to prevent looping.

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 $PATH$
  **if** $y$ $is\ not\ in\ NH\ list$ **then**
    drop QREQ message and exit this procedure
  **end if**
  let $b_{cur} = b_{y\_cur}$ in the $NH$ list in the 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**
    $L = \phi$
    $ANUyz = calcA(z, ST, RT)$
    $Fyz = calcF(z, ST, RT)$
    **if** $Fyz \geq b_{cur}$ **then**
      $b_{z\_cur} = b_{cur}$
    **else**
      **if** $Fyz \geq b_{min}$ **then**
        $b_{z\_cur} = Fyz$
        $L = select\_slot(y, z, b_{z\_cur}, ST, RT)$
      **else**
        $L_{dg} = TryDowngrade(S, D, y, z, b_{min}, b_{max})$
        **if** $size[L_{dg}] \geq (b_{min} - Fyz)$ **then**
          $b_{z\_cur} = b_{min}$
          $L = select\_slot(y, z, b_{z\_cur}, ST, RT)$
          $L = L \mid L_{dg}$
        **end if**
      **end if**
    **end if**
    **if** $L \neq \phi$ **then**
      $NH\_temp = NH\_temp(z, L) \mid (z, L)$
    **end if**
  **end for**
  **if** $NH\_temp \neq \phi$ **then**
    Let ($h_i^{'}$, $l_i^{'}$, $b_{cur}^{'}$) be the entry in $NH$ such that $h_i^{'}$=$y$
    let $PATH\_temp = PATH \mid (x, l_i^{'})$
    broadcast $QREQ$($S$, $D$, $id$, $b_{min}$, $b_{max}$, $x$, $PATH\_temp$,
    $NH\_temp$) message
  **else**
    **for** each 1-hop neighbor node $z$ of $y$ **do**
      **if** $(Fyz + ANUyz) \geq b_{min}$ **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**
```

Otherwise, if y received this QREQ message for the first time, it performs the algorithm given in Algorithm 1 which works in the following manner:

Retrieving b_{cur} and updating the ST and RT tables:

Node y checks if its id is in the NH list that is included in the QREQ message. If so, y will then retrieve the value for b_{y_cur} that is associated with its id in the NH list and set $b_{cur} = b_{y\_cur}$. Node y will also update its ST and RT tables with the information in $PATH$.

Begin determining to which neighbors the QREQ message must be propagated (to include in the NH list):

Node y determines if it has the b_{cur} number of slots available with each of its neighbors. As it considers each neighbor, z, separately, y will calculate the number of free slots, Fyz, and the number of slots that are $allocated\ not\ usable$, $ANUyz$ with this neighbor. If $Fyz \geq b_{cur}$ then y has b_{cur} available to be allocated for this path. So it sets $b_{z\_cur} = b_{cur}$. Otherwise (i.e. $Fyz < b_{cur}$), y does not have b_{cur} slots available to allocate for sending to z, so it checks if ($Fyz \geq b_{min}$). If that is the case, then it sets $b_{z\_cur} = Fyz$ and calls the $select\_slot(y, z, b_{z\_cur}, ST, RT)$ function to return a list of slots available to send to z.

If y does not have enough slots with any neighbor then try to downgrade another path:

Otherwise, if y does not have the minimum number of slots required, b_{min}, for this path it does the following. It calls the function $TryDowngrade(S, D, y, z, b_{min}, b_{max})$ to see if it is possible to downgrade an existing path in order to free up slots to be used for the path requested by the QREQ message that is being processed. The $TryDowngrade$ function will return a non-empty list of slots L_{dg} that are able to be freed by downgrading a path. This is done only if the number of those slots is enough to cover the "slot deficit" to satisfy the b_{min} requirement (i.e. $size[L_{dg}] = b_{min}$). In this case, y will set $b_{z\_cur} = b_{min}$ and will create a list of slots, L, that contains the available slots (returned by the function $select_slot$) and the slots that are in the L_{dg} list. The neighbor z along with the associated list of slots L are then added to the NH list that is being formed. If, however, the returned L_{dg} list is empty, this means that y does not have slots that can be freed in order to satisfy the b_{min} requirement of the QREQ message. In that case, the neighbor z is not added to the NH list, and z will not be a candidate to extend the path that is being allocated.

If the constructed NH list is not empty then forward the QREQ message:

After considering all of its neighbors, y will check if the resulting NH list is not empty. If so, that means that the path that is being allocated could be extended by at least one neighbor. So, $PATH$ is extended with an entry containing x along with the associated slots. Then y will broadcast the resulting QREQ message which will then be processed by only the neighbor nodes specified in the NH list. The QREQ message will then be propagated further to the destination in the same indicated fashion.

If the constructed NH list is empty, decide if it is possible to place the QREQ message in the $QREQ\_pending\_queue$ or drop it:

On the other hand, if the resulting NH list is empty, then that means that y was not able to find any neighbor, z, with whom it is able to allocate b_{min} slots. In that case, y will determine if $Fyz + ANUyz) \geq b_{min}$. Note that a slot is called ANU (allocated-not-usable) if it is not able to be used to send from y to z because of allocations (not reservations) at y or its neighbors. If that is the case, then y will place the QREQ message in the $QREQ\_pending\_queue$ in order to wait for the allocated slots to be either reserved (then the QREQ message is dropped from the queue) or freed (then the QREQ message is forwarded further). This is specified in more detail in [7]. Otherwise, if $Fyz+ANUyz) < b_{min}$, then y drops the QREQ message which cannot be propagated any

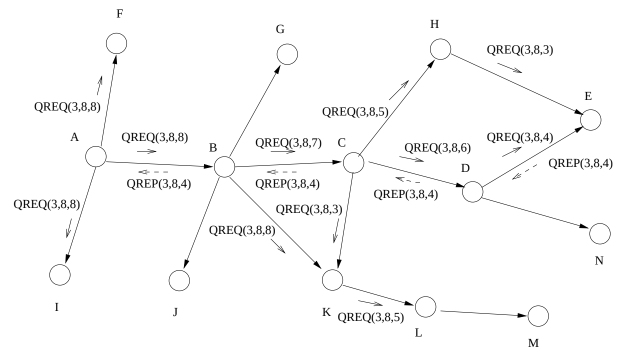

Fig. 1. Propagation of QREQ and QREP messages as a path from source node A to destination node E is being reserved. The parameters indicated are $(b_{min}, b_{max}, b_{cur})$.

further.

C. Sending the reply message from the destination

When the two QREQ messages arrive at the destination node D, this indicates that the QoS path was able to be successfully allocated. Node D can respond to the first arriving QREQ message and drop all others with the same source/destination/sessionID stamp, or it can follow a different policy which can include responding to all received QREQ messages and letting the source node choose among the different allocated paths. 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_{cur} slots where $b_{min} \leq b_{cur} \leq b_{max}$ in each hop was discovered. In this case, the destination D replies by unicasting a $QREP(S$, D, id, b_{min}, b_{max}, b_{cur}, $PATH$, $NH)$ 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 will only reserve b_{cur} slots and will free any additional slots that were allocated for this path. When the QREP message arrives at the source node S, it will then initiate data transmission along the reserved path using b_{cur} data slots per frame where $b_{min} \leq b_{cur} \leq b_{max}$.

D. A detailed example of the path reservation process

An example of the path allocation and reservation process is presented in Figure 1. The application layer of node A requests to establish a session with destination node E. The desired session has a bandwidth dynamic range of $b_{min} = 3$, and $b_{max} = 8$. Consequently, the routing protocol at node A sets $b_{cur} = b_{max}$ and initiates a QREQ message with the parameters $(b_{min}, b_{max}, b_{cur}) = (3, 8, 8)$ and broadcasts it to its neighbors B, F and I. Nodes F and I receive the QREQ message and realize that they do not have any neighbors to send the message to so they drop the QREQ message. Node B receives the message and determines that it does not have enough slots available with nodes G and J with the minimum bandwidth requirements $b_{min} = 3$. However, node B has 8 slots available to send to node K which is equal to b_{max}. Node B also has 7 slots available to send to node C. Although this number is below the required number of slots in the path so far, ,b_{cur} which is 8, it is still above the minimum number of slots b_{min}. Consequently, node B will include node C in the NH list along with a list of corresponding allocated slots and an adjusted $b_{cur} = 7$. Node B will also include node K in the NH list along with the list of allocated slots and a corresponding $b_{cur} = 8$. From this point on, the subsequent nodes along the allocation route going through node C will only try to allocate up to 7 slots (not 8), because all nodes along the path ultimately must transmit with the same number of slots. Consequently, they have to use the number of slots available in the $smallest - bandwidth - link$ along the path. In a similar manner, node K receives the QREQ message with the parameter triple $(b_{min}, b_{max}, b_{cur} = (3, 8, 8)$, and does not propagate it to node C because it does not have 3 slots available to send to C. However, it has 5 slots available to send to node L so it propagates the message to it with the triple (3,8,5). Node L determines that the number of available slots it has with node M is less than 3 so it drops the message.

On the other hand, node C receives the message with the triple (3,8,7) and forwards it to nodes H, D and K with the triples (3,8,5), (3,8,6), and (3,8,3) respectively. When node K receives the message, having processed that message identified by the source/destination/sessionID parameters before, it drops it. This condition is added in the protocol to prevent looping. On the other hand, node H propagates the message to node E, the destination, with the triple (3,8,3) and node D propagates the message to E with the triple (3,8,4). However, node D does not propagate the message to node N since the number of slots it has that are available to send to N is less than the minimum required number of slots for this path which is 3.

When the two QREQ messages arrive at the destination node E, the latter can respond to the first one and drop all other arriving QREQ messages with the same source/destination/sessionID stamp, or another strategy can be deployed to choose among them. In this example, node E receives both messages from nodes H and D and chooses to send a QREP message to confirm and reserve the path allocated along node D, because that path has the larger arriving b_{cur} value which is 4. This means that sending data from the source to the destination using this path can be done with 4 slots instead of 3 which is the number of slots possible along the path passing through node H. The QREP message is then propagated along the nodes in the allocated path (A-B-C-D-E). Each node along that path will then reserve only 4 slots and will free any additional slots that have been allocated for this path. For example, node B, having initially allocated 7 slots to send to node C along this path, will only reserve 4 slots and will free the other 3 slots to be used by other arriving QREQ messages.

V. The Downgrade and Upgrade Algorithms

In this section, the downgrade and upgrade algorithms along with the corresponding control message exchanges are discussed.

A. The downgrade algorithm

The function executing this algorithm is invoked at an intermediate node, y, under the following conditions. The node y receives a $QREQ(S, D, id, b_{min}, b_{max}, x, PATH, NH)$ to set up a path p. Node y will then retrieve b_{y_cur} that is associated with its id in the NH list. It then determines the number of free slots available with each of its neighbors in order to construct the NH list. However, y determines that it does not have enough free slots to satisfy the b_{y_cur} or even the b_{min} requirements with any of its neighbors (i.e. $Fyz < b_{min}$ for every neighbor z). Under these circumstances, y cannot forward the QREQ message any further, unless it is able to free enough slots by downgrading an existing path. In that case, y executes this algorithm to determine if any of the existing paths is $eligible$ to be downgraded in order to free up slots that can be used to extend the QREQ message. When considering the downgrading of existing paths, y proceeds as follows. y determines which neighbor node, z, has the minimum number of additional slots that are needed to extend the QREQ message. Then y proceeds to calculate if there are existing paths which are eligible to be downgraded.

Eligibility of a path to be downgraded:

A candidate path, Pc, is eligible to be downgraded if it satisfies the following conditions.

1) The current number of slots used by Pc, CS, is greater than its minimum number of slots for Pc (i.e. path Pc is functioning above its minimum requirement).
2) If there are enough slots used by Pc which can be freed to be used by p to completely satisfy its additional slot requirements in order to send to at least one neighbor z. In other words, it is highly desirable to be able to downgrade only one path in order to satisfy another. We can however consider downgrading more than one path as an optional feature that is controlled by the network administrator and the particular network and applications involved. However, downgrading more than one path would cost too much in control message overhead, so our protocol does not allow that option.
3) The time elapsed since Pc's status has been changed (downgraded or upgraded) is more than or equal to a predetermined tunable parameter $MinTch$. This reduces the frequency of fluctuations in downgrading and upgrading paths in order to minimize or reduce the possibility of $thrashing$. Thrashing would occur if the benefit reaped by changing path status is lower than the price paid by the increased exchange of the corresponding control messages.

If more than one path qualifies as eligible to be downgraded, then y can choose among them in a way that promotes fairness or by using other criteria. One such way is to keep track of time Tch since the last time a path was changed (downgraded or upgraded) and downgrade the path with the largest value for Tch.

Downgrade only after receiving the QREP message from the destination:

In order to avoid changing the bandwidth usage of a path, Pc, only to find out later that the QREQ message for the desired path, p, was not able to $allocate$ a complete path to the destination, an intermediate node, y, keeps track of the slots that could be freed up for p and adds that list of slots (L_{dg} in algorithm 1 that was presented earlier) to the slot list for that neighbor in the QREQ message's NH list without actually performing the downgrading process for Pc. Consequently, only when the QREP message arrives at y after a successful $allocation$ of the path p, does y actually perform the downgrading process of path Pc, free up the corresponding slots, and reserve them for p.

The downgrade path (DP) message:

There are two types of nodes involved in the downgrading process. The first type are nodes $directly\ involved$ in freeing up particular slots for node y to use. These nodes are 1-hop and 2-hop neighbors of node y. This is because, according to the slot allocation rules, the freeing of slots for use by any node directly involves its 1-hop and 2-hop neighbors' use of these slots. The other type includes all other nodes along the downgraded path that are $indirectly\ involved$. These nodes must free up the number of slots specified in the DP message

The $DP(S, D, id, DNL, NS, DN\_PATH)$ message that is issued by the intermediate node y contains the following fields in addition to the (S, D, id) triple that identifies the message. DNL: This a list of the 1-hop and 2-hop neighbor nodes that are $directly\ involved$ in the downgrading process along with a list of the slots to be downgraded for each node. The node issuing the DP message must create this list and include it in the message. NS: This is the number of slots to be freed. This number is used by the $indirectly\ involved$ nodes along the path which will simply free up this many slots. DN_PATH: This is a list of the nodes along the path that is being downgraded.

Issuing and propagation of the DP message:

The issuing of the DP message from an intermediate node y is done in the following manner:

1) Node y changes the ST and RT tables to free the desired slots and broadcasts a status update message to y's 1-hop and 2-hop neighbors.
2) Node y sends a DP message to all of the nodes that are $directly$ or $indirectly\ involved$ as described earlier.
3) When the DP message is received by the corresponding nodes, each of these nodes frees the corresponding slots and updates its ST and RT tables accordingly. It also sends a status update message to its 1-hop and 2-hop neighbors to update their ST and RT tables.

Unlike the UPA (upgrade path attempt) message, the DP message that is sent to downgrade an existing path does not constitute an attempt. It is actually a command to the nodes to do so; all of the nodes along the path will always be able to free up the required number of slots. This is not the case however, when a path needs to be upgraded, since it is not guaranteed that each of the participating nodes will be able to find the required number of free slots to upgrade the path. So, all of the nodes will have to agree not only on upgrading the path but by how much before the actual upgrading takes place. This process is discussed in the subsequent sections of the paper.

B. The upgrade algorithm

This algorithm is executed by the source in order to upgrade an existing QoS path with a certain destination. An UPA (Upgrade Path Attempt) message is sent from the source to the nodes along the existing path to the destination. This message will attempt to upgrade the existing path to the b_{max} number of slots initially specified by the application layer. These attempts to upgrade the path can be done periodically or can follow a more sophisticated and measured policy which will initiate such attempts only when the traffic load in the network is below a certain threshold. In such a case, the nodes in the network would have to keep track of and communicate this information with each other.

Initiation of upgrade attempts by the source:

Here we follow a policy by which the source will periodically $attempt$ to upgrade any existing paths that are functioning below the b_{max} level. The frequency is a tunable parameter which can be determined by simulation. A high frequency would cost too much overhead in the continuous attempt to upgrade paths while the network has a high traffic load. On the other hand, a frequency that is too low would cause the communication to continue below the desired maximum level even when the network load is low. A proper frequency of upgrade attempts would balance the two factors involved.

The upgrade attempt (UPA) message:

In order to attempt to upgrade an existing path, the source node sets $b_{n\_cur} = b_{max}$ and initiates a $UPA(S, D, id, b_{min}, b_{e\_cur}, b_{n_cur}, b_{max}, x, PATH\_temp)$ message which is unicast along the nodes that are on the existing path to the destination. In addition to the fields S, D, id, b_{min}, b_{max}, x and $PATH$ which were described earlier, the UPA message contains the following fields. b_{e_cur}: is the current number of slots used in the existing path. b_{n_cur}: b_{n_cur} is the new b_{cur} value that will be checked and updated by the nodes as the UPA message propagates to the destination along the nodes in the existing path.

Propagation of the UPA message:

It is important to note here that in the UPA message there is only one b_{cur} value, namely b_{n_cur}, and there is no NH list. This is because only the nodes along the path, which are included in $PATH$, are involved in this process, and there is no need to include an NH list or a different b_{cur} for each neighbor in that list. As each node receives the message, it will follow the same process of allocation and reservation of additional slots that is used in processing the QREQ message to meet the b_{n_cur} requirement that is in the message. In other words, each node will try to $allocate$ more slots to the existing $reserved$ ones for this path to make the number of slots equal to b_{n_cur}. Otherwise, if that additional number of slots is not available, it will try to add as many as possible and update the value of b_{n_cur} before including it in the UPA message which is then forwarded to the next node indicated in $PATH$. If however, an intermediate node is not able to increase the number of slots from the existing value, b_{e_cur}, then it simply drops the UPA message which is not propagated any further. There is no point in checking if the subsequent nodes along the path are able to increase the number of slots used because, ultimately, all of the nodes along the path have to be able to transmit using the same number of slots.

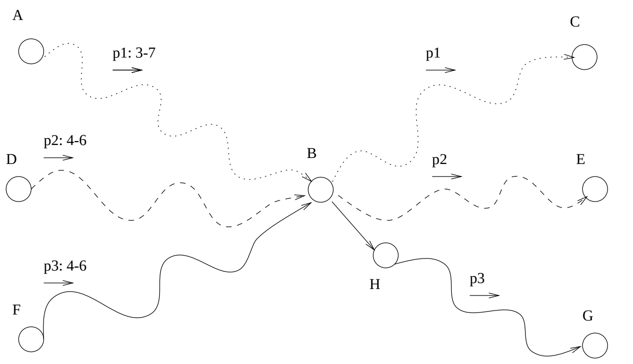

Fig. 2. Dynamic range path reservation. p1 - existing path: A..B..C, p2 - existing path: D..B..E, p3 - path being allocated: F..B..G.

Sending and propagation of the reply to upgrade message (RUPA) from the destination:

When the destination receives the UPA message, it checks if $b_{n\_cur} > b_{e\_cur}$. If that is not the case then the attempt to upgrade the path failed and no additional slots were able to be allocated by the nodes along the path. Otherwise, the attempt to upgrade the path succeeded. The destination will then unicast a Reply to Upgrade Path Attempt message, $RUPA(S, D, id, b_{min}, b_{n\_cur}, b_{max}, x, PATH)$ which is similar to the QREP message described earlier, but contains the new number of slots that must be used for sending data b_{n_cur}. The RUPA message will propagate back to the source along the intermediate nodes. Each intermediate node will then $reserve$ the additional $allocated$ slots. When the source receives the RUPA message it realizes that the path was upgraded successfully. Therefore, the source will update its routing table and start transmitting data using the new increased number of slots.

C. An example illustrating the dynamic downgrading and upgrading of paths

To illustrate the process of dynamic allocation for different paths in an ad hoc network, let's consider the example illustrated in Figure 2. The figure shows an existing path $p1$ which is being used to send data from node A to node C and passes through node B. Path $p1$ has a dynamic range of 3-7 (i.e. $b1_{min} = 3$, and $b1_{max} = 7$). Another path $p2$ is being used to send data from node D to node E and passes through node B as well. Path $p2$ has a dynamic range of 4-6. At time $t0$, node F initiates a QREQ message in order to reserve a path, $p3$, to send data to node G with a dynamic range of 4-6.

Let us assume that the QREQ message arrives at node B at time $t1$, and at that time, $p1$ has a current slot reservation of $b1_{cur} = b1_{max} = 7$ ($b1_{cur}$ is the final number of slots $reserved$ by all nodes in the path $p1$ after the reservation process is completed for that path) to send data from node B, and $p2$ has a a current slot reservation of $b2_{cur} = b2_{max} = 6$ to send data from node B as well. Let us also assume that out of the total number of data slots in the frame, node B determines according to the slot allocation rules that it only has 1 slot available to forward the QREQ message through node H (i.e. $F_{BH} = 1$). Obviously $F_{BH} < b3_{min}$. Before dropping the corresponding QREQ message or putting it in the QREQ_pending_queue, node B must determine if there are existing paths passing through itself that can be downgraded in order to free the additional slots needed for $p3$. The number of additional slots that are needed in this case is $b3_{as} = b3_{min} - F_{BH} = 4 - 1 = 3$. When running the algorithm to find such eligible paths, it determines that downgrading path $p2$ would only provide 2 additional slots with neighbor H which is not enough to propagate $p3$'s QREQ message. However, path $p1$ can be downgraded from 7 slots to 4 slots and stay within its allowable range (it would have $b1_{cur} = 4 > b1_{min} = 3$). Node B $allocates$ these slots for path $p1$, includes them in the NH list in QREQ, and forwards the QREQ message to node H. Note that at this time, node B continues to use these slots to send data for path $p1$ even though the same slots have also been allocated for path $p3$. This is the case because node B will wait for the reception of path $p3$'s QREP message before issuing a DP message for the nodes in $p1$ to downgrade the latter to 4 slots and $reserving$ the 3 previously $allocated$ slots for $p3$. If, however, the allocation time out timer for these slots expires at node B before the reception of the corresponding QREP message (i.e. path $p3$ was not able to be $allocated$ by the subsequent nodes), node B will de-allocate these slots and continue to use them for sending data along $p1$ without having downgraded it unnecessarily.

At time $t2$, node B receives a QREP message for path $p3$ confirming a successful allocation of the path. Node B will then send a $DP(S, D, id, NS, DN_PATH)$ message to the nodes along path $p1$ instructing the corresponding nodes to free $NS = 3$ slots, and updates its own ST and RT table to free the 3 allocated slots and reserve them for sending data from itself to node H. Node B will also send a status update message containing the new slot status information to its 1-hop and 2-hop neighbors which will update their ST and RT tables as well.

It is noteworthy here to indicate the following observation. Recall that node B decided to allocate the 3 additional slots used by path $p1$ to later be used by path $p3$ if the latter was able to be allocated successfully. Node B had to wait for the corresponding QREP message for path $p3$ before sending the DP message to downgrade $p1$ by 3 slots. It would be possible to have a racing condition in this process under the following circumstances. After the forwarding of the QREQ message for $p3$ from node B and before the reception of the corresponding QREP message or the timing out of the allocation of the 3 slots, another node, say node H in this example, residing between B and the destination for $p3$, is faced with the same problem, and decides to also downgrade another path that uses nodes in common with $p1$. In that case, both B and H are anticipating benefit from downgrading the same nodes by possibly the same slots. That would cause the racing condition. In order to avoid this problem, the forwarded QREQ message must also include information about the nodes which would be downgraded along with the corresponding path (session id) and associated slots. All subsequent intermediate nodes will then take that information into account during the slot allocation and path downgrading process.

VI. Conclusions and Future Work

We presented a protocol for dynamic range bandwidth reservation in MANETs. Although the presented algorithm was designed for the TDMA environment, it can be extended to work in other types of networks. In this protocol, the application layer of the source node specifies a dynamic range $[b_{min}, b_{max}]$, which is the number of data slots (or bandwidth) that is required. The network will then try to reserve a path within the desired range. Downgrading of existing paths, within their corresponding ranges, can be performed by the reservation protocol in order to accommodate new requests. Active paths can later be upgraded when the traffic load of the network subsides. This flexibility in the resource reservation is very desirable for QoS support in MANETs in general. It allows the network to dynamically adjust its resource allocation as its traffic load changes in order to satisfy the greatest number of sessions. We are considering more optimization techniques that can be applied to enhance the performance, robustness, and efficiency of the protocol as well as its extension to include other QoS constraints such as delay, and error rates.

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. 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.

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

[4] 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.

[5] I. Jawhar and J. Wu. Quality of sevice routing in mobile ad hoc networks. *accepted to appear in Resource Management in Wireless Networking, M. Cardei, and D. -Z. Du (eds.), Kluwer*, 2003.

[6] I. Jawhar and J. Wu. Quality of service routing in tdma-based ad hoc networks. *The 7th INFORMS Telecommunications Conference*, March 2004.

[7] I. Jawhar and J. Wu. A race-free bandwidth reservation protocol for QoS routing in mobile ad hoc networks. *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04), IEEE Computer Society*, 9, January 2004.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

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