Resource allocation for real-time and multimedia communications in TDMA-based wireless networks

Imad Jawhar*

College of Information Technology,
United Arab Emirates University,
Al Ain, UAE
E-mail: ijawhar@uaeu.ac.ae
*Corresponding author

Jie Wu

Department of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, Florida, USA
E-mail: jie@cse.fau.edu

Abstract: In order to support multimedia traffic, QoS routing protocols for Mobile Ad Hoc Networks (MANETs) allow finding a path between two nodes which satisfies the application layer's minimum bandwidth requirements. Currently, such protocols exist for TDMA, and CDMA-over-TDMA environments. However, these protocols do not account for race conditions which can become more significant with increased node mobility, network density and higher traffic loads. This paper provides a QoS routing protocol which enables the network to cope with this and other related problems such as parallel reservations. Furthermore, additional optimisations, which significantly enhance network throughput and efficiency are provided.

Keywords: MANETs; mobile ad hoc networks; QoS; quality-of-service; routing; TDMA; time division multiple access; wireless networks.

Reference to this paper should be made as follows: Jawhar, I. and Wu, J. (xxxx) 'Resource allocation for real-time and multimedia communications in TDMA-based wireless networks', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Imad Jawhar is an Assistant Professor at the College of Information Technology at UAE University. He has a BS/MS in Electrical Engineering from UNC-Charlotte, an MS in Computer Science, and a PhD in Computer Engineering from FAU, Florida. He published numerous papers in international journals, and conferences. He worked at Motorola in the design and development of cutting-edge communication systems and was the president and owner of Atlantic Computer Training and Consulting, Florida, USA. His research focuses on wireless, ad hoc, and sensor networks, mobile computing, distributed, and multimedia systems. He is a member of IEEE, and ACM.

Jie Wu is a Distinguished Professor at the Department of Computer Science and Engineering, Florida Atlantic University and a Program Director at US National Science Foundation. He has published over 450 papers in various journals and conference proceedings. His research interests are in the areas of wireless networks and mobile computing, routing protocols, fault-tolerant computing, and interconnection networks. He served as the chair IEEE TCDP committee. He also served as an editor, program chair, and co-chair of numerous international committees, conferences, and journals. He is the author of the text *Distributed System Design* published by the CRC press. He is an IEEE fellow.

1 Introduction

Networking is becoming an essential part of society, and Mobile Ad hoc Networks (MANETs) provide flexibility and adaptability in this environment (Stallings, 2002; Stojmenovic, 2002). As mobile electronic devices advance in capabilities, communication between these devices becomes essential. MANETs allow mobile computers and devices to communicate 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.

Existing MANET routing protocols provide the capability for establishing multi-hop paths between nodes on a best effort basis (Perkins, 2001; Perkins and Royer, 1998). 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 (De et al., 2002; Hwang and Varshney, 2003; Nelakuditi et al., 2002; Sobrinho and Krishnakumar, 1999; Xiao et al., 2000; Ye et al., 2003). Jawhar and Wu (2005) discuss the issues and challenges of QoS routing in MANETS. Furthermore, a classification of these QoS routing algorithms is presented. The protocols are classified according to the most closely related best effort algorithm, as well as the model and environment they assume, and the communication layer within which they operate. In this paper, we consider the problem of QoS routing in a Time Division Multiple Access (TDMA) 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 (Chen and Nahrstedt, 1999; Gerasimov and Simon, 2002a, 2002b; Lin and Liu, 1999, 2000). 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 neighbour'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 neighbour's use of this slot; its 2-hop neighbour's current use of this slot must be considered as well. This is due to the well-known hidden and exposed terminal problems (Gerasimov and Simon, 2002a; Liao et al., 2002), 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 et al. (2002) 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 (Gerasimov and Simon, 2002a, 2002b). 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 optimisation techniques, which additionally contribute to improving the efficiency of the QoS routing protocol. These techniques include Time to Live (TTL) soft timers for allocated and reserved slots in order to avoid deadlock, and de-allocation messages propagated from the destination to quickly de-allocate unused slots once the path discovery process is complete. These optimisations will be discussed further in subsequent sections in the paper.

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 optimisation 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 message is not received within a period of time which allows it to comply with the QoS route request delay requirements.

The remainder of the paper is organised 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.

2 Related work

Bandwidth reservation with QoS routing in MANETS is an issue that has been and continues to be investigated by current researchers. In Chen and Nahrstedt (1999) 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 neighbours. This is a strong assumption because such a protocol might require a multi-antenna model. In Lin and Liu (2000) and Lin and Liu (1999) a calculation algorithm for bandwidth is presented. However, it assumes that neighbouring 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 Bertossi and Bonuccelli (1995); Garcia-Luna-Aceves and Raju (1997).

The protocols in Ho and Liu (2000), Lin and Liu (2000) and Lin and Liu (1999) 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 Gerasimov and Simon (2002a, 2002b), 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 (Chakrabarti and Mishra, 2001), and Temporally Ordered Routing Algorithm, or TORA (Park and Corson, 1997; Perkins, 2001), 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 Gerasimov and Simon (2002a, 2002b) use some of the scheduling mechanisms presented in Lin and Liu (1999). However, their approach is different from those in the above protocols in that they incorporate QoS path finding based on bandwidth-scheduling mechanisms 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 (Liao et al., 2001). 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 Chen and Nahrstedt (1999), 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 Liao et al. (2001) 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 Liao et al. (2001); Lin and Liu (2000, 1999), 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 Liao et al. (2002), 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 (Gerasimov and Simon, 2002a, 2002b; Liao et al., 2002).

3 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 (Chen and Nahrstedt, 1999; Lin and Liu, 1999). 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 et al. (2002) 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 neighbour'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 neighbours. 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 Liao et al. (2002).

3.1 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 (Liao et al., 2002):

- slot t is not scheduled for receiving or transmitting in node x or y
- slot t is not scheduled for receiving in any node z that is a 1-hop neighbour of x
- slot t is not scheduled for sending in any node z that is a 1-hop neighbour of y.

The protocol provided is similar to that used in Liao et al. (2002) but with modification which solves the race conditions, which is discussed in detail later in this paper. The protocol is on-demand, source based and similar to DSR (Perkins, 2001). 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 S, which is the source node, determines if enough slots are available to send from itself to at least one of its 1-hop neighbours, and if so, then broadcasts a $QREQ(S, D, id, b, x, PATH, NH)$ to all of its neighbours. The message contains the following fields:

- S: ID of the source node.
- D: ID of the destination node.
- id: Message ID. The (S, D, id) triple is therefore unique for every QREQ message and is used to prevent looping.
- b: 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 $((h_1, l_1), (h_2, l_2), \ldots, (h_k, l_k))$. 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 $((h'_1, l'_1), (h'_2, l'_2), \ldots, (h'_k, l'_k))$. 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 neighbours. For a neighbour i and slot j, $ST_x[i, j]$ can have one of the following values representing two different states: 0 - for free, and 1 - 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 neighbours. For a neighbour i and slot j, $RT_x[i, j]$ can have one of the following values representing two different states: 0 - for free, 1 - for reserved to receive.
- $H_x[1..n, 1..n]$: This table contains information about node x's 1-hop and 2-hop neighbourhood. If an entry $Hx[i, j]$ is 1, this means that node i, which is a 1-hop neighbour of node x, has node j as a neighbour; an entry of infinity indicates that it does not.

Let z_1 and z_2 be two 1-hop neighbours 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 neighbours of z_1 are sending and not receiving in slot t, but not all 1-hop neighbours 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 it 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.

3.2 A detailed example of the slot allocation process

In order to illustrate the slot allocation process, consider the example in Figure 1. Node A wants to reserve a QoS path to node F with $b = 3$ (i.e., 3 slots). Node A sends a QREQ message to reserve the path. The QREQ message travels through the nodes on its way to F and arrives at node C. Node C will now try to allocate slots for this QREQ message to send to each of its 1-hop neighbours, if there are b slots available to send from itself to this neighbour.

Let's consider the process of calculating the number of slots available to send from node C to its 1-hop neighbour, node D. Node C has slot allocation information for itself and for all of its 1-hop and 2-hop neighbours including node D, since each node is required to notify its 1-hop and 2-hop neighbours of the allocation status of its slots. Node C realises that it cannot allocate slots 1, 2, 5, 7 and 8, because they are scheduled by nodes C and D to send or receive (slot allocation rule 1). It cannot use slots 3 and 4 because they are scheduled to receive in its 1-hop neighbours, nodes B and G, respectively (slot allocation rule 2). Furthermore, node C cannot use slot 10, because it is scheduled to send in node E, which is a 1-hop neighbour of the node it intends to send to, node D (slot allocation rule 3). However, node C can use slot 6 to send to node D even though it is scheduled to send in node B. This is the exposed terminal problem. In fact, it would be more desirable for node C to allocate this slot to send to node D; this would increase channel reuse, a desired goal in wireless communications. Node C can also use slot 9 even though it is being used to send from node I to node H, since this does not violate any of the slot allocation rules. Consequently, there are 6 slots that are available to be used to send data from node C to node D (slots 6, 9, and 11–14). Since the QREQ message only needs 3 slots, node C is able to forward the QREQ message to node D.

Assume that, after the calculation above, node C allocates slots 6, 9 and 11 to send from itself to D, and broadcasts the QREQ message. In Liao et al. (2002), node C does not keep track of this allocation, which is only remembered in the forwarded QREQ message. So, until node C receives the corresponding QREP message from the destination F, slots 6, 9 and 11 will remain *free*. They will only change status from *free* to *reserved* when and if the corresponding QREP message arrives from node F on its way to node A to confirm the slot reservations of the QoS path $A \to \cdots \to B \to C \to D \to E \to \cdots \to F$. This poses no problem so long as no other requests arrive at node C during the period between forwarding the QREQ and receiving the corresponding QREP message. However, consider a situation where, during this period, another request arrives at node C from another source node J trying to reserve a QoS path from itself to node K with $b = 5$. Node C in this case will look at its slot status tables and will see no allocations for slots 6, 9 and 11–14. In Liao et al. (2002), node C will proceed to reserve some of these slots for this newly requested path causing *multiple reservations* of the same slots for different paths. This is a race condition which results in data collisions at node C during the data transmission phase, and it is discussed in detail in the next section.

3.3 The problems: race condition and parallel race condition

The race condition

This condition occurs when multiple reservations happen simultaneously at an intermediate node. Consider the situation in Figure 2(a). When a node B receives QREQ1 (with b slots required) from node A to node F, 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

Figure 1 Illustration of the slot allocation procedure being done to determine the slots that are available to send data from node C to node D for a QREQ message arriving at node C. The figure shows the slot reservation status before the arrival of the QREQ message at node C. R: scheduled to receive. S: scheduled to send. Empty: not scheduled to receive or send

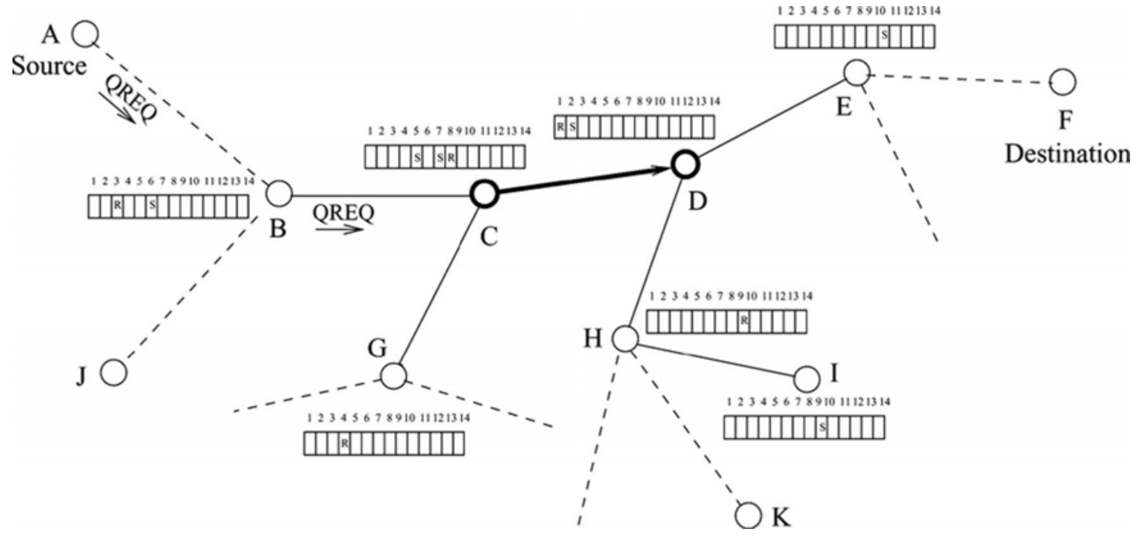
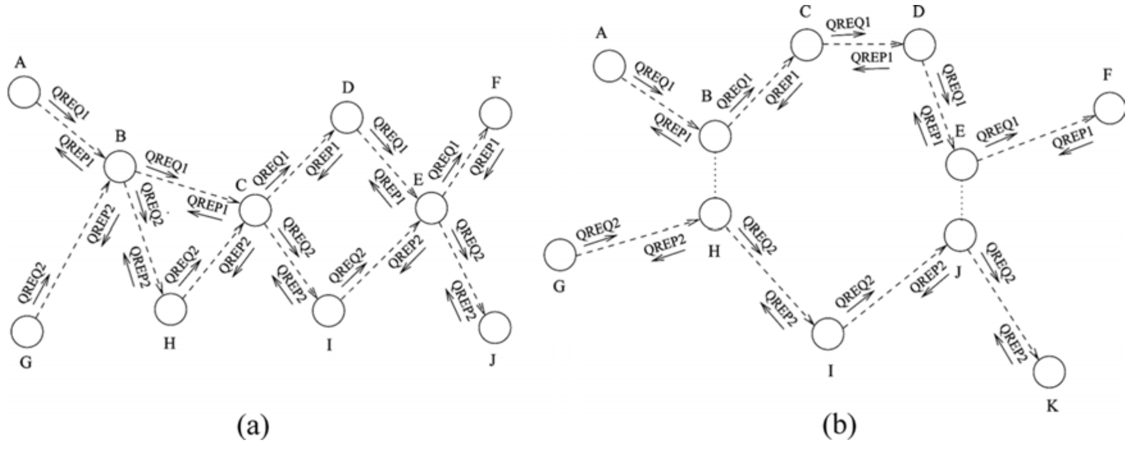


Figure 2 (a) Race condition of two QoS paths passing through common intermediate nodes and (b) race condition of two parallel QoS paths passing through 1-hop neighbours



from node F to A and reserve the allocated slots, it is possible that another request, QREQ2, can arrive at node B. QREQ2 from node G requests to reserve another path from node G to node J passing through node B. In the algorithm in Liao et al. (2002), 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 G to J. When the reply message, QREP1, arrives at B to confirm the QoS path reservation from F 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 G to J, 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. This same race condition can occur at nodes C and E with multiple reservations at those nodes for the same slots for the paths being reserved by QREQ1 and QREQ2.

The conflict arises when the packets are transmitted from A to F and G to J simultaneously, and two data packets from two different paths arrive at nodes B, C and E. In this case, these nodes must decide which data packets they will actually send. The other data packets belonging to the other path will have to be dropped. In this case, node B, C, and E 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.

The parallel reservations problem

Consider the situation in Figure 2(b). In this case, there are two parallel paths, ABCDEF and GHIJK, that are being reserved. Two or more of the intermediate nodes belonging to the two parallel paths are 1-hop neighbours. In this case node B, which belongs to the first path, and node H, which belongs to the other path are 1-hop neighbours. This is indicated in the figure using the dashed lines. The same relationship exists between nodes E and J. When the QREQ1 is propagating from node A to F, 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 no memory of this allocation is kept by the node, as is the case in Liao et al. (2002). This can cause another type of race condition, which is called the parallel reservation problem here. This problem arises if, before QREQ1 propagates and is confirmed, the same process occurs with QREQ2 and node H 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 H can be violating the slot allocation conditions mentioned earlier in this paper. Nodes B and H each did the allocation based on information which did not consider the other 1-hop neighbour node's slot allocation for the corresponding parallel path being reserved. Again, if the two parallel paths are reserved successfully and data transmission is started along these paths, collisions will occur at the 1-hop neighbours belonging to the different parallel paths. In this example, nodes B and H would experience this collision in their transmissions. A similar situation can occur between any 1-hop neighbours belonging to the two parallel paths, for example, between nodes E and J of the same figure. In this paper, we propose an algorithm to fix this problem, which we call the *parallel reservation problem*.

4 The race-free 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, the protocol uses a more conservative strategy. This strategy is implemented using the following features:

- 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.
- As the QREQ message propagates from source to destination, slot status is changed from *free* to *allocated* in the intermediate nodes. Therefore, this information is maintained 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 Liao et al. (2002). As the QREP message propagates from the destination to the source the corresponding slot status in the nodes is changed from *allocated* to *reserved*.
- 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).
- TTL timer for allocated and reserved slots.
- TTL timers for maximum total QREQ propagation delay allowed, and for maximum total QREQ/QREP delay allowed (i.e., maximum QoS path acquisition time).
- Destination-initiated de-allocation messages and other optimisations.

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 is discussed in detail later in the paper. The protocol also has the following optimisation. Along with the QREP message, the destination broadcasts a 'de-allocation message' which includes the source, destination, session id and allocated path information in order to release the slots allocated by nodes that are not a part of the final QoS path during the path discovery process. This is done in order to minimise slot allocation time to further improve slot utilisation and network performance.

4.1 Wait timers

The following timers are defined, which control the allowable delay of the propagation of the QREQ and QREP messages through the system. These timers can be initialised 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 initialisation 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 initialised to a predetermined $TTL\_allocated\_slot\_time$. This time should be at least equal to the Round Trip Time (RTT) 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.

Explicit de-allocation message from the destination

In addition to the above soft allocation timer strategy, further performance improvement can be achieved by having an explicit short deallocation message issued as a flood from the destination to the source. This message is initiated by the destination when it receives as soon as a QoS path is discovered. The reception of the deallocation message by the nodes in the network will cause the immediate deallocation of the slots which were not used in the final path/s. This increases the utilisation and efficiency of the network. Both the soft deallocation timer as well as the explicit deallocation message are incorporated in the protocol.

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 is defined 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 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:

- 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
- 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.

4.2 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 neighbours (i.e., with a 2-hop TTL). This broadcast is done periodically (synchronously) according to a predetermined periodic slot status update frequency. This is defined as *periodic_status_update_time*. These periodic updates enable the nodes to maintain updated neighbourhood information as nodes come within or go out of their range. Furthermore, these updates inform the node of its neighbour'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 neighbour after a time-out period, which is called Status_update_tot, it will assume that this node is no longer one of its 1-hop or 2-hop neighbours, and will delete that neighbour 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 neighbours of the change from allocated to free which results from TTL timer expiration. The neighbours 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 neighbour node will have done that) if no reservation change is indicated from the corresponding neighbour node. Note that the status updates are done with a 2-hop TTL flood to the 1-hop and 2-hop neighbours.

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 neighbour receives a separate and different QREQ, it will now be aware of the *free/allocated/reserved* status of its neighbours' 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.

4.3 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 neighbouring 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 defined 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 neighbours, 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 neighbours, 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 neighbours.
- 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, it is necessary to determine a separate set of $NUyz$, $ANUyz$, and Fyz for each neighbour 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.

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
        $NUyz = calcR(z, ST, RT)$
        $ANUyz = calcA(z, ST, RT)$
        $Fyz = calcF(z, ST, RT)$
        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 $QREQ(S, D, id, b, x, PATH\_temp, NH\_temp)$ 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
```

The algorithm first updates the ST and RT tables with the information in PATH. Then the algorithm initialises the next hop list NH_temp to empty, and then attempts to build it by adding to this list each 1-hop neighbour 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 neighbour z of y has b slots free to send from y to z, this is called *condition* 1, 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 $PATH\_temp = PATH \,|\, (x, li')$). Here, | means concatenation.

Otherwise, if the NH_temp list is empty after checking all of the neighbours, then that means that there are no neighbours 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 neighbour z of y which has the condition $(Fyz + ANUyz) \geq b$. This would be *condition* 2. 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 neighbour 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.

4.4 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.

4.5 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 neighbours of y. It then builds the next hop list accordingly, which will include every neighbour 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
```

4.6 How the new protocol solves the race conditions

The proposed protocol solves the race conditions stated earlier in the following manner.

Solving the race condition

Consider the example of Figure 1, 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 C makes the calculation of the slots available for transmission to node D, it will consider only slots with free status. Before forwarding the QREQ message, node C will designate slots 6, 9 and 11 as *allocated* (not yet fully *reserved*, but not *free* either). If another QREQ message for another path arrives at node C, it will consider only slots of *free* status and will therefore consider allocating slots 13 and 14 for the second path. When QREP for the first path arrives, it will confirm the reservation of slots 6, 9 and 11 and will convert them to *reserved* status. When the reply for the second path arrives, it will also confirm the reservation of slots 13 and 14 and convert them to *reserved* status. When data transmission starts for both paths, there will be no conflict at node C.

Another possibility is that another QREQ message for a different path arrives at node C. Let the number of slots required for that path be $b = 2$. Assume that slot 13 is allocated by another path but slot 14 is still free. Then the QREQ message 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 (one slot in this case) time out (fail to be confirmed before a time out period) or are confirmed. In the first scenario, the QREQ message will proceed, and in the second scenario it will be discarded.

Similarly to the above analysis, the possible race condition illustrated in Figure 2(a) at nodes B, C and E between QREQ1 and QREQ2 is now alleviated for the same reasons discussed in this section.

Solving the parallel reservation problem

The proposed protocol does not have the parallel race condition problem, which was illustrated in the example in Figure 2(b). When nodes B allocates slots for QREQ1 to reserve them for the $A \to B \to C \to D \to E \to F\, path$, it must immediately, due to the asynchronous status updates, broadcast the slot status information to all of its 1-hop and 2-hop neighbours which include node H. So, when nodes H, receives QREQ2 (before QREP1 comes back to node B), it will do the slot allocation for the $G \to H \to I \to J \to K$ 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 H 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. Consequently, there will be no collisions between nodes B and H when the data transfer begins along the two separate and parallel paths $A \to B \to C \to D \to E \to F$ and $G \to H \to I \to J \to K$. The same analysis also applies to slot reservations at nodes E and J of the two parallel paths. In previous protocols, collisions would have taken place between transmissions of nodes B and H on one hand and nodes E and J on the other hand.

4.7 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 (Gerasimov and Simon, 2002a, 2002b; Liao et al., 2002).

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. It is reasonable to believe that the same relationships apply in the case of QoS routing in ad hoc wireless networks. Consequently, those principles are applied in this protocol to improve the performance of the ad hoc networks under more stressed network conditions.

5 Performance analysis

In order to verify, and analyse the performance of the presented protocol, simulation experiments were conducted. The simulator is event driven and was designed using the C++ object oriented language. The simulator incorporates the details of the simulated networking protocols. It includes the following major classes. The area class, which contains all of the nodes in the simulation. The node class which includes the ST, RT, H, and routing, slot, and slot deallocation tables. The event priority queue class, along with various other classes such as simulator, QREQ/QREP messages, statistics, graph, and event classes. The details of the protocols were simulated and collisions during the data transmission process due to violations of the slot allocation rules were detected.

5.1 Simulation

Basically the simulator starts by generating an area with certain dimensions and randomly places a predetermined number of nodes in the area. The nodes have a certain transmission range. From the placement of the nodes and their range a graph is generated. Then the simulator generates a number of data messages with a certain length for each message (different distributions can be used). Each message has a random source and destination pair. The arrival times of the messages is according to a Poisson process with a certain mean inter-arrival time. When the data message is processed by the source, it will generate a QREQ message to discover a QoS path to the corresponding destination. The QREQ message is propagated through the nodes according to the algorithm. As indicated earlier, each node has a routing table as well as all of the tables needed for the algorithm (ST, RT, all of the required slot data structures, etc). When the source receives the QREP message it starts data transmission. The new race-free algorithm as well as the other existing algorithm by Liao et al. (2002) are simulated. The latter algorithm is referred to as the no-allocation algorithm.

A set of simulation experiments were performed. Table 1 shows a sample of the simulation parameters used in the experiments. The results for two sets of experiments are shown. Figures 3, and 4 contain the results for the first set of experiments, and figures 5, 6 contain the results for the second set of experiments. The number of nodes (n) is 30 in an area of $600 \times 600\,\text{m}^2$. The total number of data slots in the frame (dsn) is 60. The number of slots required for each session is a random number with a uniform distribution and a range from 1 to 5 slots (1 to max_b). The range of each node was set to 150 m. The message length is randomly selected according to a uniform distribution with a range from 0 to 1000 Mbytes corresponding to a session length range of 0–4000 s. In the first set of experiments, the traffic rate was varied. The session (or data message) arrival is a Poisson process with a mean which was increased from 0.001 (i.e., 1 session every 1000 s) to 45 messages/sec. In the second set of experiments, the maximum node speed was varied from 2 to 14 m/s. The following section describes the mobility simulation.

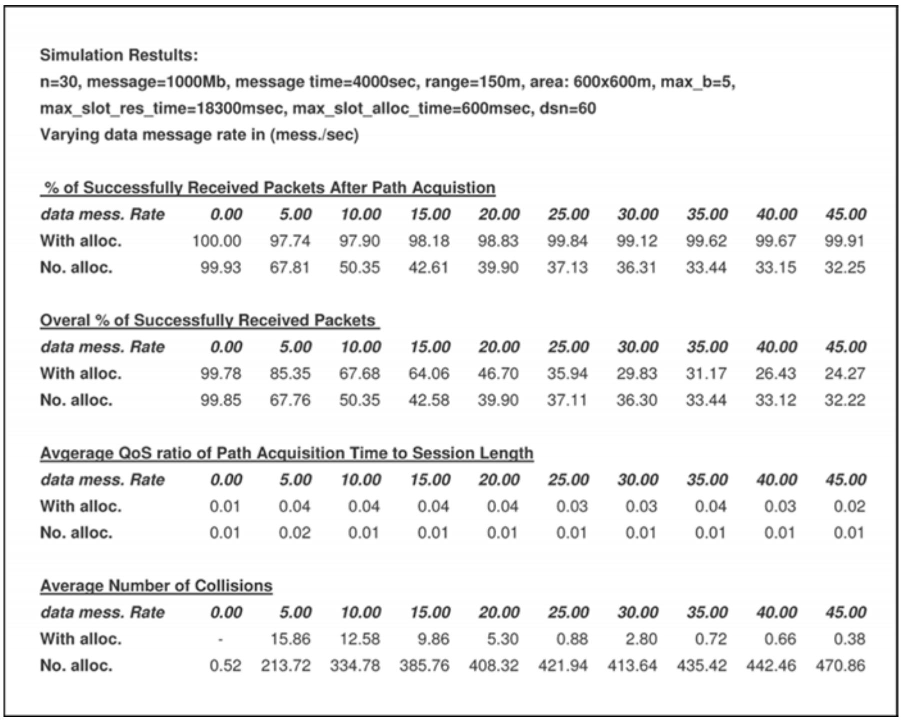
Table 1 Parameters for the race-free protocol simulation

Parameter	*Value*
Network area	$600 \times 600\,\text{m}^2$
Number of nodes	30
Transmission range	150 m
Bandwidth	2 Mb/s
Data packet size	512 bytes
Number of data slots	60
Number of sessions	50
Maximum session time	4000 s
MAX_SLOT_RES_TIME	18300 ms
MAX_SLOT_ALLOC_TIME	600 ms
MAX_B	5 slots

5.2 Mobility simulation

In order to simulation mobility, the Waypoint mobility model was used. In the model, all of the nodes are

Figure 3 Simulation results table. Varying traffic rate



Simulation Restults:

n=30, message=1000Mb, message time=4000sec, range=150m, area: 600x600m, max_b=5,

max_slot_res_time=18300msec, max_slot_alloc_time=600msec, dsn=60

Varying data message rate in (mess./sec)

% of Successfully Received Packets After Path Acquistion

data mess. Rate	***0.00***	***5.00***	***10.00***	***15.00***	***20.00***	***25.00***	***30.00***	***35.00***	***40.00***	***45.00***
With alloc.	100.00	97.74	97.90	98.18	98.83	99.84	99.12	99.62	99.67	99.91
No. alloc.	99.93	67.81	50.35	42.61	39.90	37.13	36.31	33.44	33.15	32.25

Overal % of Successfully Received Packets

data mess. Rate	***0.00***	***5.00***	***10.00***	***15.00***	***20.00***	***25.00***	***30.00***	***35.00***	***40.00***	***45.00***
With alloc.	99.78	85.35	67.68	64.06	46.70	35.94	29.83	31.17	26.43	24.27
No. alloc.	99.85	67.76	50.35	42.58	39.90	37.11	36.30	33.44	33.12	32.22

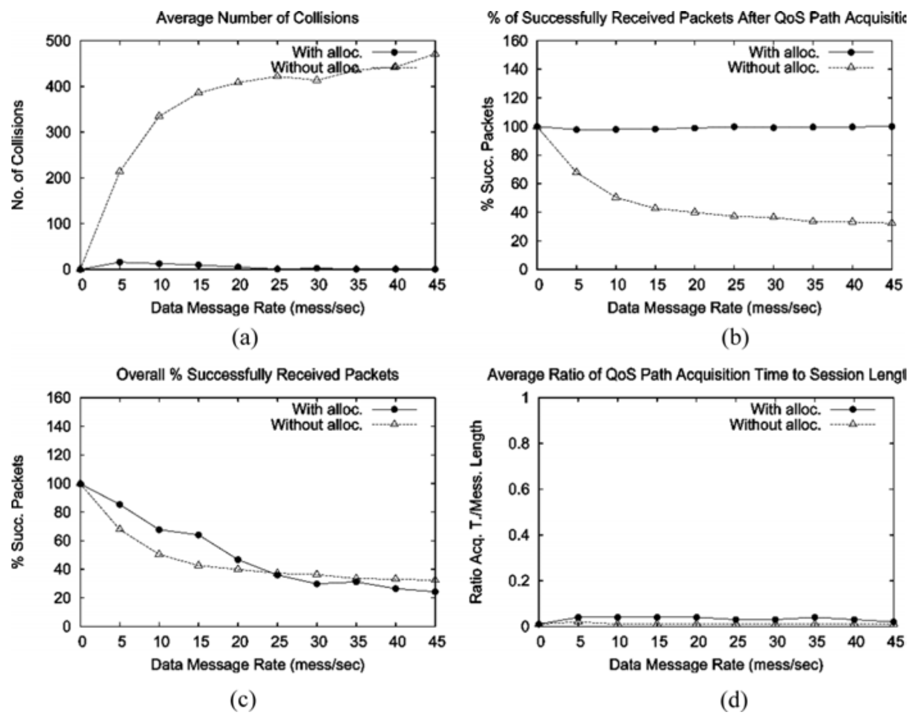
Avgerage QoS ratio of Path Acquisition Time to Session Length

data mess. Rate	***0.00***	***5.00***	***10.00***	***15.00***	***20.00***	***25.00***	***30.00***	***35.00***	***40.00***	***45.00***
With alloc.	0.01	0.04	0.04	0.04	0.04	0.03	0.03	0.04	0.03	0.02
No. alloc.	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Average Number of Collisions

data mess. Rate	***0.00***	***5.00***	***10.00***	***15.00***	***20.00***	***25.00***	***30.00***	***35.00***	***40.00***	***45.00***
With alloc.	-	15.86	12.58	9.86	5.30	0.88	2.80	0.72	0.66	0.38
No. alloc.	0.52	213.72	334.78	385.76	408.32	421.94	413.64	435.42	442.46	470.86

Figure 4 Simulation results. Varying traffic rate



placed in random locations. Each node then selects a random speed between 0 and a maximum speed (in m/sec) according to a uniform distribution. The node also selects a random geographic location within the specified area and starts moving towards that location at the selected speed. When the node arrives at that location, it pauses for a predetermined pause time which is set to 0, in this case, in order to maximise mobility. The node then selects a new speed and geographic destination, and starts the process again, and so on. It is important to note that due to mobility, each node sends messages according to its internal connectivity table (the *H* table which was discussed earlier). However, the simulator only allows nodes to receive messages according to the actual geographic location of the nodes, indicated by the 'true graph' which is generated from the actual locations and ranges of the nodes. Therefore, the actual reception of messages is not done according to the 'nodes perception' of its connectivity, which is in its *H* table. Obviously, mobility leads to a lack of synchronization of the *H* table and the true graph causing messages that are sent by nodes to other nodes, that they 'think' are their neighbours, to be lost. The node's *H* table is only updated after the hello messages are received and is therefore synchronised with the true graph temporarily. This synchronisation is then gradually lost with time, as the nodes move, until the next hello messages are received.

It is worthwhile noting that other variations of simulation parameters such as group mobility, and node density can be used to further study the performance of the protocol under these various conditions. However, this is left as future research for more optimisation of the proposed protocol.

5.3 Simulation results and analysis

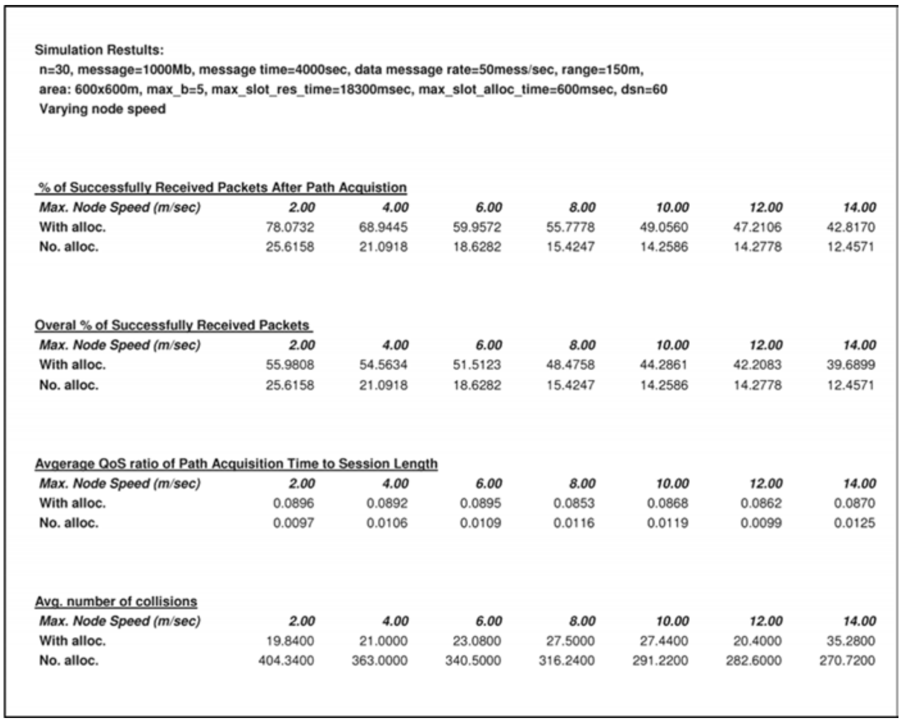
Several performance measures were computed as the traffic rate (messages per second) and maximum speed (node's mobility) were varied. The measured parameters are the percentage of packets received successfully after QoS path acquisition, the overall percentage of packets received successfully, the average ratio of QoS path acquisition time to session length, and average number of collisions.

In both sets of experiments, it can be clearly seen that the percentage of successful packets received drops as the traffic rate and mobility increased. In the first set of experiments, the node locations were randomly selected, the maximum speed was set to 0 (i.e., static network), and the data traffic rate was varied. Figures 4(a) and 4(b) demonstrate the race-free nature of the protocol. Since no, or relatively very few, collisions take place after successful path acquisition, the percentage of successfully received packets stays close to 100% throughout the entire range (from 100% to 99.91). On the other hand, the no-allocation algorithm (i.e., the algorithm by Liao and Tseng) has numerous collisions due to the racing conditions described earlier and this drops the percentage of successfully received packets after QoS path acquisition below 100%. Namely, as the data traffic rate increases in the indicated range, this percentage changes from 99.93 down to 32.25. In Figure 4(c), the overall percentage of successfully received packets (including the sessions that could not acquire a QoS path) decreases as the data traffic rate increases. The average number of collisions that is presented in Figure 4(a) shows the clear advantage of the race-free algorithm and its success in reducing the number of collisions. The race-free algorithm has minimal and relatively negligible collisions with respect to the number of packets sent, while the no-allocation algorithm has an increasing number of collisions which grows up to 470.86 at the end of the range. During data transmission, such collisions can be very detrimental to many sensitive and timing critical applications. The upper layers would have to initiate re-transmissions or other corrective actions, which can be time consuming and can cause a considerable decline in the overall throughput of the network. With regard to the overall percentage of successfully received packets, shown in Figure 4(c), it is important to mention that in the race-free algorithm's case, the reason behind the packets not getting transmitted is mainly due to the session not being able to be established because the network is too overloaded with current traffic. The application layer can then choose to try to establish the connection at a later time. However, once the session is established it is considerably more reliable as demonstrated in the previous analysis. An increase in the number of retries to establish the path would further increase the number of transmitted packets and the overall percentage of successfully received packets, in the race-free algorithm's case. Conversely, the reason for the packets being dropped with the no-allocation algorithm is due to multiple reservations caused by the racing conditions. This means that even after the session is established and a QoS path is reserved, data transmission would be unreliable and vulnerable to numerous collisions which can cause costly data errors, retransmission delays, packet sequencing issues and other complications for the QoS session. These problems can be very harmful and detrimental to varying degrees to the underlying real-time or multimedia application. In the worst case, they can render the data session impractical for especially error and delay sensitive applications.

On the other hand, as expected, the simulation results in Figure 4(d) show that the improved percentage of successful data delivery, and elimination of collisions due to the racing conditions comes at the price of a relatively small increase in the average number of requests needed for establishing a session, and consequently an increase in the average time needed per successful QoS path establishment. This price however, is very tolerable by most QoS applications which would be willing to pay such cost in order for the discovered QoS path and subsequent data session to be considerably more reliable. In addition, it is noted that the message exchange overhead is increased in the race-free protocol due to the asynchronous slot status updates. However, the number of such added messages is still relatively small due to the fact that they are highly localised to the 1-hop and 2-hop neighbours of the node.

The second set of experiments results, shown in Figures 5 and 6 present the performance of the two protocols under varied mobility conditions. The maximum node speed was varied from 2 to 14 m/sec with a constant data message rate of 50 messages/sec. Figure 6(b) shows the percentage of successful data packets received after QoS path acquisition. It is important to note here that when nodes move at a higher speed, collisions are caused by both node mobility, which will affect both protocols, as well as the racing conditions. Therefore, the percentage of successfully received packets after QoS path acquisition would be expected to be below 100% for both protocols. The race-free protocol has 78.07% success rate at maximum speed of 2 m/s, but as the maximum node speed gradually increases to 14 m/s, this percentage drops to 42.82%. This is due to collisions that take place due to mobility leading to the difference between the node's perception of its neighbourhood (in the *H* table) and its

Figure 5 Simulation results table. Varying mobility rate



Simulation Restults:
n=30, message=1000Mb, message time=4000sec, data message rate=50mess/sec, range=150m,
area: 600x600m, max_b=5, max_slot_res_time=18300msec, max_slot_alloc_time=600msec, dsn=60
Varying node speed

% of Successfully Received Packets After Path Acquistion

Max. Node Speed (m/sec)	***2.00***	***4.00***	***6.00***	***8.00***	***10.00***	***12.00***	***14.00***
With alloc.	78.0732	68.9445	59.9572	55.7778	49.0560	47.2106	42.8170
No. alloc.	25.6158	21.0918	18.6282	15.4247	14.2586	14.2778	12.4571

Overal % of Successfully Received Packets

Max. Node Speed (m/sec)	***2.00***	***4.00***	***6.00***	***8.00***	***10.00***	***12.00***	***14.00***
With alloc.	55.9808	54.5634	51.5123	48.4758	44.2861	42.2083	39.6899
No. alloc.	25.6158	21.0918	18.6282	15.4247	14.2586	14.2778	12.4571

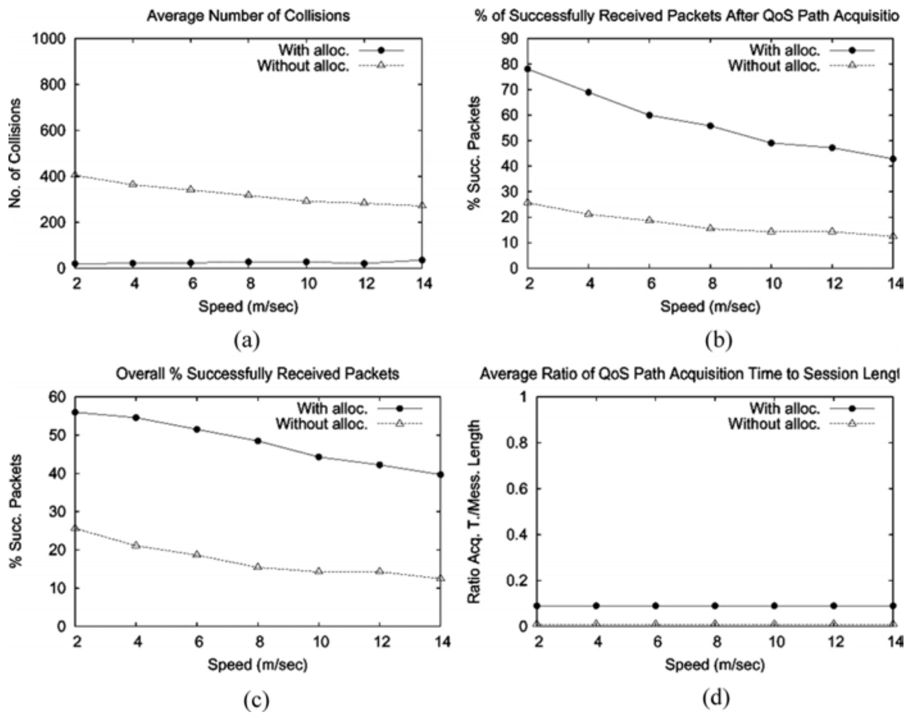
Avgerage QoS ratio of Path Acquisition Time to Session Length

Max. Node Speed (m/sec)	***2.00***	***4.00***	***6.00***	***8.00***	***10.00***	***12.00***	***14.00***
With alloc.	0.0896	0.0892	0.0895	0.0853	0.0868	0.0862	0.0870
No. alloc.	0.0097	0.0106	0.0109	0.0116	0.0119	0.0099	0.0125

Avg. number of collisions

Max. Node Speed (m/sec)	***2.00***	***4.00***	***6.00***	***8.00***	***10.00***	***12.00***	***14.00***
With alloc.	19.8400	21.0000	23.0800	27.5000	27.4400	20.4000	35.2800
No. alloc.	404.3400	363.0000	340.5000	316.2400	291.2200	282.6000	270.7200

Figure 6 Simulation results. Varying mobility rate



actual connectivity (in the 'true graph'). In the case of the no-allocation protocol the success rate varies from 25.62 to 12.46, which is significantly and consistently below that of the race-free protocol. This is because it has collisions due to both the race conditions, as well as mobility. The number of collisions, as shown in Figure 6(a), is also negligibly small for the race-free protocol compared to a relatively large number of collisions (almost 20 times more on average) in the no-allocation protocol. An interesting observation is that the chart shows a slight decrease of the number of collisions for the no-allocation protocol as mobility increases. While at first this might seem unexpected, after further analysis, it was determined that this is due to the following fact. As the maximum node speed increases, and all nodes become more mobile, the data packets that are sent after path acquisition have a lower probability of even going through the early hops (first and second hops, etc.). This indicates that these packets will have no more collisions along the path, since they are dropped by the early nodes in the path (first and second nodes, etc.) and go no further. At lower speeds however, these packets get to traverse more nodes along the reserved path and cause more collisions due to the multiple reservations of the same slots caused by the racing conditions encountered during the path reservation process. Again, this performance gain and increase in quality of service comes at a small price of relatively slightly increased path acquisition time, as shown in Figure 6(d).

6 Conclusions

In this paper, a protocol for bandwidth reservation for QoS support in TDMA-based MANETs is presented. This protocol remedies the race conditions which are 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 to control slot allocation, maximum QREQ node wait time, maximum QREQ/QREP total wait time, and destination-initiated de-allocation messages. The protocol provides a solution to the multiple slot reservations at intermediate nodes and parallel reservation problems in QoS routing, which were not resolved in previous research, and improves the network's performance and its ability to provide QoS support.

Acknowledgement

This work was supported in part by NSF grants CCR 0329741, CCR 9900646, ANI 0073736, EMI 0130806, and UAEU Research grant 08-03-9-11/07.

References

Bertossi, A.A. and Bonuccelli, M.A. (1995). 'Code assignment for hidden terminal interference avoidance in multihop radio networks', *IEEE/ACM Trans. on Networks*, Vol. 3, No. 4, August, pp.441–449.

Chakrabarti, S. and Mishra, A. (2001) 'QoS issues in ad hoc wireless networks', *Communications Magazine, IEEE*, Vol 39, No. 2, February, pp.142–148.

Chen, S. and Nahrstedt, K. (1999) 'Distributed quality-of-service routing in ad hoc networks', *IEEE Journal on Selected Areas in Communications*, August, pp. 1488–1505.

De, S., Das, S.K., Wu, H. and Qiao, C. (2002) 'A resource efficient RT-QoS routing protocol for mobile ad hoc networks', *Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on*, pp.257–261.

Garcia-Luna-Aceves, J.J. and Raju, J. (1997) 'Distributed assignment of codes for multihop packet-radio networks', *Proc. IEEE MILCOM '97*, Vol. 1, November, pp.450–454.

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

Gerasimov, I. and Simon, R. (2002b) 'Performance analysis for ad hoc QoS routing protocols', *Mobility and Wireless Access Workshop, MobiWac 2002. International*, pp.87–94.

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

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

Ilyas, M. (2003) *The Handbook of Ad Hoc Wireless Networks*, CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, Florida, USA.

Jawhar, I. and J. Wu, (2004a) 'Quality of service routing in TDMA-based ad hoc networks', *The 7th INFORMS Telecommunications Conference*, March.

Jawhar, I. and Wu, J. (2004b) '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*, Vol. 9, January.

Jawhar, I. and Wu, J. (2005) 'Quality of sevice routing in mobile ad hoc networks', in Cardei, M., Cardei, I. and Du, D-Z. (Eds.): *Resource Management in Wireless Networking, Springer, Network Theory and Applications*, Vol. 16, pp.365–400.

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

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

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

Lin, C.R. and Liu, J-S. (1999). 'QoS routing in ad hoc wireless networks', *IEEE Journal on selected areas in communications*, Vol 17, No. 8, August, pp.1426–1438.

Nelakuditi, S., Zhang, Z-L., Tsang, R.P. and Du, D.H.C. (2002) 'Adaptive proportional routing: a localized QoS routing approach', *Networking, IEEE/ACM Transactions on*, Vol. 10, No. 6, December, pp.790–804.

Park, V.D. and Corson, M.S. (1997) '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*, Vol. 3, April, pp.1405–1413.

Perkins, C.E. (2001) *Ad Hoc Networking*, Addison-Wesley, Upper Saddle River, NJ, USA.

Perkins, C.E. and Royer, E.M. (1998) 'Ad Hoc On Demand Distance Vector (AODV) routing', *Internet Draft*, August, pp.1–37.

Sobrinho, J.L. and Krishnakumar, A.S. (1999) 'Quality-of-service in ad hoc carrier sense multiple access wireless networks', *Selected Areas in Communications, IEEE Journal on*, Vol. 17, No. 8, August, pp.1353–1368.

Stallings, W. (2002) *Wireless Communications and Networks*, Prentice Hall, Upper Saddle River, NJ, USA.

Stojmenovic, I. (2002) *Hanbook of Wireless Networks and Mobile Computing*, Wiley, New York, NY, USA.

Xiao, H., Lo, K.G. and Chua, K.C. (2000) 'A flexible quality of service model for mobile ad-hoc networks', *Proc. of IEEE VTC2000-Spring*, Tokyo, May, pp.445–449.

Ye, Z., Krishnamurthy, S.V. and Tripathi, S.K. (2003) 'A framework for reliable routing in mobile ad hoc networks', *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, Vol. 1, pp.270–280.