INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2004; **17**:233–252 (DOI: 10.1002/dac.640)

QoS routing through alternate paths in wireless *ad hoc* networks

Baoxian Zhang[‡] and Hussein T. Mouftah[*,†]

School of Information Technology and Engineering, University of Ottawa, Ontario, Canada, K1N 6N5

SUMMARY

Quality of service (QoS) routing plays an important role in QoS provisioning for mobile *ad hoc* networks. This work studies the issue of route selection subject to QoS constraint(s). Our method searches for alternate routes with satisfied QoS requirement(s) to accommodate each communication request when the shortest path connecting the source–destination pair of the request is not qualified. In order to effectively reduce protocol overhead, a directed search mechanism is designed to limit the breadth of the searching scope, which aims at achieving a graceful tradeoff between the success probability in QoS route acquisition and communication overhead. Efficient hop-by-hop routing protocols are designed for route selection subject to delay and bandwidth constraint, respectively. Simulation results show that the designed protocols can achieve high performance in acquiring QoS paths and in efficient resource utilization with low control overhead. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: mobile *ad hoc* network (MANET); quality of service (QoS); unicast; hop-by-hop routing

1. INTRODUCTION

Quality of service (QoS) routing plays an important role in providing real-time services with desirable service quality in terms of bandwidth and/or end-to-end delay. This work studies the issue of QoS routing for mobile *ad hoc* networks (MANETs). The goals of QoS routing are two folds: (a) selecting paths that can satisfy given QoS requirements of arriving communication requests, and (b) achieving global efficiency in resource utilization. Recently, the issue of QoS routing has received a lot of attention for providing QoS in MANETs and much work has been carried out to address this critical issue.

QoS is difficult to guarantee in MANETs because the wireless channel is shared among adjacent hosts and network topology can change as hosts move. Here, we list some general considerations in designing a QoS routing protocol. It is very difficult to guarantee an initial QoS contract with a session, which has specific QoS requirement(s), due to network dynamics caused by node mobility and link state impreciseness. There may exist transient time when the required QoS is not guaranteed due to path break or network partition. A designed routing protocol should scale well with network size, and with computation, communication and storage overhead.

*Correspondence to: Hussein Mouftah, School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada, K1N 6N5.
[†] E-mail: bxzhang@site.uottawa.ca
[‡] E-mail: mouftah@site.uottawa.ca

Received 9 September 2003
Revised 2 November 2003
Accepted 1 December 2003

This work designs hop-by-hop QoS routing protocols. The design objective is to discover cost-effective QoS-satisfied routes with controllable protocol overhead. We study two QoS routing issues, namely, delay- and bandwidth-constrained least-cost unicast routing. The main contributions of this work are as follows. First, a simple alternate routing mechanism is designed to support QoS routing for MANETs. Second, a directed search mechanism is designed, which creates an ellipse-searching zone, to effectively reduce the communication overhead associated with a path-searching process. Efficient hop-by-hop routing protocols are designed using these two mechanisms for supporting delay- and bandwidth-sensitive applications, respectively.

The rest of this paper is organized as follows. Section 2 formulates the network to be studied, the issues to be addressed as well as a brief review of related work coping with the issue of QoS routing. Section 3 presents the ideas behind QoS routing through alternate routes and behind directed search. Section 4 gives design details of a protocol supporting delay-constrained routing. Section 5 presents details of a protocol designed for bandwidth-constrained routing. Section 6 provides simulation results to evaluate the performance of the designed protocols. Section 7 concludes this paper.

2. PRELIMINARIES

2.1. Network model

A communication network is modelled as a set V of nodes that are interconnected by a set E of communication links. V and E are changing over time when nodes move. Each node has a unique identifier. We assume that the effective transmission range of each node is equal. Each node is equipped with omni-directional antenna. Two nodes are immediate neighbours and an undirected link connecting them exists if they are in the transmission range of each other. In the rest of this paper, we will use the terms host, router and node interchangeably unless otherwise specified.

The state information associated with a link $(i, j) \in E(G)$ includes (a): $\text{cost}(i, j)$, which can be simply one as hop count or a function of the link utilization, $\text{cost}(i, j) \in R^+$; (b): $\text{delay}(i, j)$, the delay that a packet experiences when it goes through the link, $\text{delay}(i, j) \in R^+$; and (c): $\text{width}(i, j)$, the residual (available) bandwidth on the link.

For a directed path p, the cost, delay, and width of path p are defined as follows, respectively:

$$\text{cost}(p) = \sum_{(i,j)\in p} \text{cost}(i, j) \tag{1}$$

$$\text{delay}(p) = \sum_{(i,j)\in p} \text{delay}(i, j) \tag{2}$$

$$\text{width}(p) = \min_{(i,j)\in p} \{\text{width}(i, j)\} \tag{3}^{\S}$$

In the rest of this paper, we will mainly focus on designing routing protocols to select QoS-satisfied paths for each arriving request with QoS requirement(s). The following assumptions are

[§] We will present more details with respect to path width calculation in MANETs in Section 5. Equation (3) provided here is just to ease the explanation of our routing protocol.

made for protocols designed in this work to execute successfully. First, as for *resource availability*, each network node is assumed to be able to monitor the available resources, such as delay, cost, and available bandwidth, on each of its outgoing links. Second, as for *resource reservation*, a medium access protocol is assumed to be able to resolve media contention and support resource reservation (as necessary) at the MAC layer.

2.2. Problem formulations

This work studies the following two routing problems: delay- and bandwidth-constrained least-cost unicast routing. Each problem can be formulated as follows:

Delay-constrained least cost (*DCLC*) *unicast routing problem*: Given a source node s and a destination node d, and a delay constraint Δ, find a directed simple path p from the source s to the destination d such that

(a) $\text{delay}(p) \leqslant \Delta$, and
(b) $\text{cost}(p)$ is minimal among all those feasible paths connecting the s–d pair.

The DCLC problem is known to be NP-complete [1].

Bandwidth-constrained least cost (*BCLC*) *unicast routing problem*: Given an s–d pair, and a bandwidth constraint B, find a directed simple path p from the source s to the destination d such that

(a) $\text{width}(p) \geqslant B$, and
(b) $\text{cost}(p)$ is minimal among all those feasible paths connecting the s–d pair.

The BCLC problem is *polynomial* if accurate global network state information (i.e. topology and state information on each link in the network) is available. With such state information, the source s can *locally* compute a bandwidth-constrained path (if available) using a path-calculating algorithm such as Dijkstra's shortest path first algorithm [2], after pruning those links without enough bandwidth resources. However, gathering and maintaining global state information suffer from the issues of scalability and information inaccuracy, hence designing simple and efficient hop-by-hop heuristic solutions is highly desirable for providing bandwidth-sensitive services in MANETs.

2.3. Related work

Recently, the QoS routing issue has attracted a lot of attention and much work has been carried out. Here, we will focus exclusively on those algorithms and protocols addressing either the DCLC issue or the BCLC issue as defined in the preceding section and we will not discuss those designed for route selection subject to multi-constraints, which is somewhat orthogonal to the issues studied in this work. For a good survey for such work please refer to References [3, 4] and references cited therein. Next, we will first present a review of related work designed for wired networks and then of that designed for MANETs.

2.3.1. QoS routing in wired networks. To address the DCLC issue, Widyono [5] presented a constrained Bellman–Ford (CBF) algorithm. CBF performs breadth-first search to identify the optimal constrained path and its running time grows exponentially with network size. In

Reference [6], Hassin proposed two ε-optimal approximation algorithms with the complexity of $O(\log\log B(|E|(|V|/\varepsilon) + \log\log B))$ and $O(|E|(|V|^2/\varepsilon)\log(|V|/\varepsilon))$, respectively, where B is an upper bound on the optimal cost. In Reference [7], Lorenz and Raz presented an ε-optimal algorithm with the complexity of $O(|E||V|(\log\log|V| + 1/\varepsilon))$. All these ε-optimal algorithms produce a path with a maximum cost of ε-factor from the optimal solution (if available). However, despite their elegance, these algorithms have scalability issue to be employed in MANETs wherein nodes are with very limited computing capabilities and also gathering fresh global network state information from time to time is very difficult in MANETs.

In References [8, 9], the authors presented hop-by-hop delay-constrained routing heuristics, which have the following properties in common. First, each node must maintain the least-cost (LC) and also the least-delay (LD) information to every other node in the network. Second, the path searching process adds one node at a time and at each time the next node to be added lies on either the LC path or on the LD path from the head of partial path probed thus far to the destination node. In Reference [10], the authors attempted to combine the benefits of probing- and backtracking-based algorithms (better adaptiveness and wider search) with those of distance-vector algorithms (lower setup time). With any one of the above heuristics in References [8–10], at any time, only a single probe message propagates in the network for route selection. The worst case message complexity of algorithms in References [8–10], are $O(|V|^2)$, $O(|V|)$, and exponential, respectively. The implementation of all the above heuristics requires at least two versions of wide-area distance vector routing protocols running in the background, which converge with respect to (w.r.t.) cost and delay metric, respectively. The large amount of control overhead introduced for obtaining and updating such distance vector information, respectively, makes these protocols difficult to be employed in dynamic and resource-scarce MANETs.

In Reference [11], Wang and Crowcroft designed routing protocols for returning the shortest–widest path, which is the path with the maximum width among all paths, and if there are multiple such paths, the shortest one w.r.t. a cost metric such as hop count is selected. In Reference [11], two algorithms based on modified distributed Bellman–Ford and Dijkstra's algorithms, respectively, were designed to return such path.

2.3.2. QoS routing in MANETs. Next, we provide a review of recent work addressing the QoS routing issue in the context of MANETs. This work can be divided into three categories: shortest path routing, flooding, and multiple-path routing. To provide QoS, the shortest path routing strategy simply returns the shortest path if this path meets the QoS requirement(s) of an arriving request, or otherwise rejects the request. Example protocols using this strategy can be found in References [12, 13] and they work by simply checking the feasibility of the mini-hop path connecting the source–destination pair of each request to make a decision on accepting the request or not. Advantages of this strategy are its simplicity, fast route acquisition, and little control overhead. It can work well in an environment where traffic demand is light, in which case a mini-hop path probably has enough resources to accommodate a new request with QoS requirements. However, this strategy can suffer from low success probability in acquiring a feasible route (when available) as traffic demand increases (e.g. as observed in Reference [14]). This is because the non-feasibility of the mini-hop path does not mean the non-existence of feasible paths in the entire network.

In contrast, flooding is another strategy for QoS routing, which works by flooding a route-searching message across the entire network to search for a QoS-route on demand. Intermediate

nodes forward a received route-searching message provided that the given QoS requirement(s) are not violated yet. Once the intended destination receives a route-searching message, a QoS-route is discovered. Example protocols using this strategy can be found in References [14, 15]. Flooding can achieve high success probability in terms of route acquisition due to its wide searching scope. On the other hand, path discovery using blind flooding, an expensive operation in resource-scarce MANETs, can introduce excessive communication overhead.

The strategy of multiple-path routing aims at achieving a good tradeoff between success probability in route discovery and protocol overhead. It works by searching multiple paths in parallel for a feasible path.

Following this strategy, in Reference [16], Chen and Nahrstedt presented a protocol using ticket-based probing (TBP). TBP requires each node to maintain a distance vector to reach each other node with respect to cost, delay, and bandwidth metrics, respectively. To obtain such distance information, each node runs multiple underlying wide-area routing daemons in parallel, one for each of the concerned metrics. Upon receiving a request for a QoS route to an intended destination, the source issues a fixed number of *probe* packets, each carrying a ticket. Each *probe* is in charge of searching for a path, if possible. The maximum number of *probes* at any time is bounded by the number of tickets. The basic idea of using tickets is to confine the number of probe packets to avoid a blind flooding. Although elegant in concept, TBP has the following deficiencies. First, *proactively* running multiple wide-area routing daemons in networks as specified above can generate excessive protocol overhead, an undesirable feature for a protocol to be employed in resource-scarce MANETs. In addition, maintaining all one-hop neighbouring nodes' routing tables at each node, as required by TBP, can introduce a large amount of storage overhead at nodes.

3. QoS ROUTING THROUGH ALTERNATE ROUTES

The protocols designed in this paper follow in part the philosophy in multiple-path routing for route discovery. They search for alternate routes with satisfied QoS properties when the shortest path is not qualified. The design objective is to discover cost-effective QoS routes at high success probability and with low overhead. Next, we present the key idea behind alternate routing subject to QoS constraint(s) and that behind directed search to reduce searching space, key components in our designed protocols as presented later.

3.1. Alternate routing

Our routing mechanism searches for alternate routes with satisfied QoS properties if the shortest path is not qualified. To achieve simple QoS routing, we focus on those alternate paths meeting the form of $P(s,x) + P(x,d)$, $x \in V \setminus \{s,d\}, P(s,x)$ and $P(x,d)$ is the shortest path from the source s to an intermediate node x, and that from node x to the intended destination d, respectively, and the symbol '+' represents the concatenation of two path segments that have a common end point. A node that connects two path segments is referred to as a *relay node*. A concise description of such a concatenated route is as follows: going directly from the source s to an intermediate node x and then going from the node x directly to the destination d (see Figure 1). Among all such concatenated routes connecting the s–d pair, the path that can maximize network resources utilization is selected provided that it meets the given QoS requirement(s).

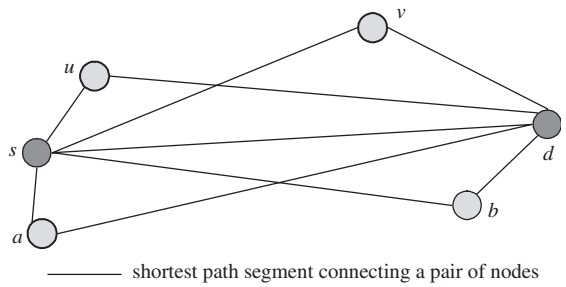

Figure 1. Illustrating the idea behind alternate routing. Source: s, destination: d.

Procedure *Alternate_Routing*

Input: A source node s, a destination node d and specific QoS requirements.
Route Set: Paths meeting the form of $P(s,x)+P(x,d), \forall x \in V \setminus \{s,d\}$.
Output: The path with the minimal cost among all of the above concatenated paths provided that the path satisfies the given QoS requirement(s).

Figure 2. A generic procedure for alternate routing subject to QoS constraint(s).

Note that a concatenated route meeting such a form may contain a loop. However, as shown later, the path returned by our algorithm is guaranteed to be loop free. Figure 2 gives a generic procedure for identifying such an alternate route.

To identify qualified alternate routes, a route-selecting protocol is needed. Due to the scalability concern in implementing source-routing algorithms, in this work, we design hop-by-hop routing protocols to search for alternate routes with satisfied QoS properties. The procedure for this purpose is described as follows. The source s broadcasts a *probe* message, which carries the specified QoS requirements and the identifier of the intended destination d, to its neighbours for an alternate route if the shortest path is not qualified. Upon receiving such a message, an intermediate node, say x, checks the *feasibility* of the concatenated path $P(s,x)+P(x,d)$. If this concatenated path is feasible, node x sends a *reply* message back to the source to notify the discovery of a qualified alternate route; otherwise, it forwards the *probe* message further to its neighbours to continue this path-searching process.

3.2. Directed search

In order to avoid excessive communication overhead while maintaining a reasonable high-success probability in route discovery, we design a method of directed search, with which distance information stored at intermediate nodes is collectively used to guide *probe* messages to proceed in promising directions and to avoid blind flooding.

Before illustrating how our method works, we first introduce a closely related method named TTL-scoping, a frequently used strategy for restricting searching scope in existing work. There are two cases worthy to be specified. One is to discover a best-effort route. For this purpose, expand ring search can be used and it works as follows. The source initially sets the TTL value in the packet header of a route-searching packet as a pre-determined value and starts a timer.

Each router decrements the TTL value by one. If the TTL value reaches zero, the packet is discarded. If timed out without receiving a corresponding route-reply message, the source broadcasts the route-searching message again with the TTL value incremented by a certain amount. This process continues until a path is found or the TTL value reaches a threshold. Another case is to discover a QoS-route as described by Chen in Reference [17]. For this purpose, Chen's method is to limit the searching scope by relaxing the TTL value of a route-searching message as follows:

$$D = h(s, d) + H \tag{4}$$

$h(s, d)$ is the minimum-hop distance between the s–d pair, and H is a small integer. As a result, the upper bound on path length is D and the corresponding searching zone is a *circle* area about the source s of radius D (see Figure 3). Note that the distance in Figure 3 is measured in hop count.

Our method of directed search comes from the following observation. In Figure 3, the ellipse represents the set of nodes such that the sum of the distances from the two fixed points (i.e. the source s and the destination d) is constant D, a curve that can be formally expressed as follows:

$$h(s, x) + h(x, d) = D \tag{5}$$

where x is a node on the ellipse. From ellipses' property, we know that for a searching process with an upper limit of path length D, its *effective* searching zone includes only those nodes inside and on the ellipse determined by (5). Nodes outside this ellipse should not process or forward a route-searching message issued to find a path subject to such a length constraint D. In this way,

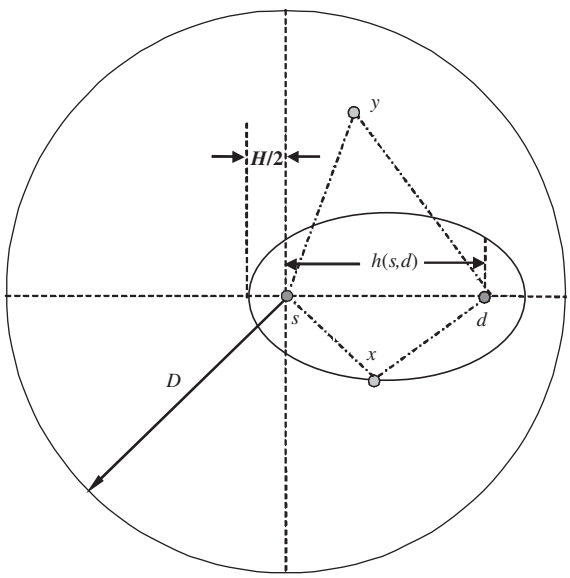

Figure 3. Illustrations of searching scope for a QoS route connecting source s and destination d subject to a path length constraint D, where $D = h(s, d) + H$, $h(s, d)$ is the minimum-hop distance from s to d, and H is a small positive integer.

the searching scope is effectively reduced. This is our method of directed search, which creates an ellipse-forwarding zone for path discovery.

The parameter H, in determining the value of D, can be used to control the tradeoff between communication overhead and the end-to-end length of a returned path, and between communication overhead and success probability of finding a feasible path. Note that if H is set to infinity, the route-searching process degenerates into a pure flooding, and if H is set to zero, the route-searching process degenerates to the shortest path routing. It is worthy pointing out that a longer path (i.e. by setting a larger value of H) can consume more network resources and is break-prone compared with a shorter one in a dynamic network.

With the strategies of alternate routing and directed search presented above, in the next sections, we design efficient protocols for route selection subject to delay and bandwidth constraints, respectively.

4. DELAY-CONSTRAINED ROUTING

This section presents the design details of a delay-constrained routing protocol, referred to as DLR, to cope with the NP-hard DCLC routing issue defined in Section 2.2. Before providing the procedures of the DLR protocol, we first give the information to be maintained at each network node for a successful implementation of DLR.

4.1. Routing information kept at each node

With DLR, each node $u, u \in V$, must maintain a *distance table* consisting of $|V| - 1$ entries (one entry for every other node). The entry for node v at node u contains the following information:

- The destination's identifier, v.
- The hop count distance of the mini-hop path from u to v, $h(P(u, v))$.
- The delay value of the above path, delay$(P(u, v))$.
- The successor of u to reach v on the above path.

In the rest of this paper, we will use the function cost$(\cdot)$ and $h(\cdot)$ interchangeably unless otherwise stated. The above-specified information can be provided by the execution of a distance-vector routing protocol such as DSDV [18], with an additional item to record information regarding the end-to-end delay on the corresponding path. This requirement is similar to that in Reference [13] to establish connections with desired QoS properties in MANETs. It should be pointed out here that nodes implementing the DLR protocol execute just a single wide-area routing daemon, which converges w.r.t. hop count to gather the above-specified distance information. This implementation decision is one of the key features that distinguish DLR from those protocols in References [8–10, 16] as introduced earlier, each of which requires two underlying routing daemons that converge w.r.t. cost and delay metrics, respectively. This decision can greatly reduce the proactive communication overhead introduced for supporting hop-by-hop QoS routing.

4.2. Procedure for DLR

Suppose that an application wishes to establish a connection between a source node s and a destination node d with a delay constraint Δ. Upon receiving such a request, source s first checks the feasibility of the mini-hop path $P(s, d)$ from s to d. If delay$(P(s, d)) \leqslant \Delta$, DLR returns this


```
Procedure rcv_probe
1.  if the probe message was received via the link, over which node v forwards data packets to the source s, and
2.      h(p)+h(P(v, d)) ≤ D, and
3.      delay(p)← delay(p)+d(u,v) ≤ Δ
4.      if delay(p)+delay(P(v,d)) ≤ Δ    //A path with satisfied delay property is found
5.        h(p)← h(p)+h(P(v, d))
6.        A reply message is sent back to the source node, which contains the following information: end-to-end
            path cost h(p) and the identifier of the relay node v;
7.      else
8.        Re-broadcast the probe to its immediate neighbors after including the updated delay(p) and h(p).
9.      end if-else
10. else
11.     The probe message is simply discarded.
12. end if-else
13. end rcv_probe
```

Figure 4. Procedure for DLR protocol at an intermediate node v upon receiving a *probe* message from an immediate neighbour u.

path since it is the optimal solution for the request; otherwise, a *probing* process is executed to search for an appropriate relay node resulting in a cost-effective alternate path with satisfied delay property. According to DLR, the source node first composes a *probe* packet containing the following information:

- Identifiers of the source s, the destination d, a locally assigned session ID s_{id}, and a locally assigned sequence number $s_{\text{sequ\_num}}$.
- Delay bound: Δ.
- $D \leftarrow h(P(s,d)) + H$.
- delay(p) $\leftarrow 0$; //the accumulated delay on the path probed so far is zero initially.
- $h(p) \leftarrow 0$; //the end-to-end length (in hop count) of the path probed so far, which is zero initially.

where p is the path probed thus far, which is initially *null*. The source s then broadcasts this *probe* message to its immediate neighbours and starts a timer. The receipt of a *reply* message at the source s means the discovery of a feasible path. If the source receives multiple *reply* messages, each bringing a feasible path for the request, it selects the one with the minimal length to maximize network resource utilization.[‖] If timed out without receiving any *reply* message, the source can perform a QoS negotiating process to relax the delay constraint or transmit data packets of the session *temporarily* as best effort traffic.

Figure 4 gives procedures of DLR to be executed at an intermediate node $v, v \neq d$, for processing a *probe* message received from one of its immediate neighbours, say u. In Figure 4, Steps 1–3 gives the preconditions (as explained next), under which a received *probe* message can be accepted. First, the message was received over the link on the reverse shortest path to the source (see Step 1). Second, the pre-determined upper bound on path length is not violated (see Step 2). Third, the partial path probed thus far meets the delay requirement (see Step 3). Step 4 gives the successful terminating condition of the probing process as follows: the delay on the partial path probed thus far *plus* the end-to-end delay on the shortest path from the current node to the intended destination satisfies the given delay requirement. If so, node v sends a *reply*

[‖] Optionally, the source can keep other routes (if any) or a subset of them as standby paths for fast restoration when the working path breaks.

message back to the source s to notify the discovery of such a concatenated path $P(s,v)+P(v,d)$ (see Step 6); otherwise, node v broadcasts the *probe* message further to its neighbours to continue the probing process (see Step 8).

Lemma 1
Paths returned by the DLR protocol are loop-free.

Proof
If a returned path is the shortest path from the source s to the destination d, $P(s,d)$, it is obviously loop-free. Now suppose that the protocol returns a concatenated path that can be expressed as $P(s,v)+P(v,d)$, where $v \in V \setminus \{s,d\}$. We need to prove that $P(s,v)+P(v,d)$ is loop-free. To prove this claim, we assume to the contrary that there exists a loop in the returned path. Suppose that there is a node w, $w \neq v$, on $P(s,v)$, which is also on $P(v,d)$ (see Figure 5). Since $P(s,v)+P(v,d)$ is the path returned by the heuristic, we have

$$\text{delay}(P(s,w)+P(w,d)) < \text{delay}(P(s,v)+P(v,d)) \leqslant \Delta \tag{6}$$

According to the procedures in Figure 4, the probing process should have terminated successfully and returned when it arrived at node w before reaching node v, which leads to an obvious contradiction. Thus, the returned path must be loop-free. □

4.3. Data packet forwarding

Once a cost-effective constrained path is identified, the next step is to select an appropriate mechanism to forward data packets of the session along the discovered path. There are two choices for this purpose as described below.

4.3.1. Stateful forwarding mechanism. After identifying a QoS-satisfied route, the source can explicitly send a signalling message along the discovered route to create a forwarding entry for the session and to reserve resources at each intermediate node along the path. A forwarding entry contains information regarding identifiers of the next hop, the last hop, the source, and the destination, and the session ID, and the amount and attribute of the resources reserved for the connection. After receiving an acknowledge message from the destination, the source then can send data packets along the established connection.

4.3.2. Stateless forwarding mechanism. This mechanism is chosen to implement the relay concept in a hop-by-hop manner. Tunnelling and packet encapsulation [19, 20] can be used for such a purpose. Suppose that a relay node v has been identified. Without loss of generality, assume that $v \notin \{s,d\}$. To forward a packet from the source s to the relay node v first and then from node v to the destination d, the original IP packet is encapsulated into another (outer) packet. The destination field of the outer packet is inserted as the address of the relay node v,

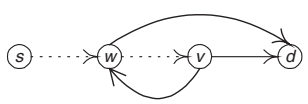

Figure 5. Hypothetical scenario for the occurrence of a loop in a returned path.

while the destination field of the inner packet is inserted as the address of the intended destination d. The data packet then can be forwarded in a standard hop-by-hop manner. The outer packet will be forwarded along $P(s, v)$ to node v, which in turn strips off the outer header and forwards the inner packet to the destination node d along $P(v, d)$. Note that no encapsulation is needed if the returned path is the shortest path connecting the source–destination pair of the session.

Once data packets are encapsulated using the above method, intermediate nodes on a QoS-path are exempted from maintaining any flow-specific state information and they forward data packets of a QoS-sensitive flow in the same way as in best-effort routing. Of course, this comes at the expense of some overhead associated with packet encapsulation and tunnelling.

4.4. Path maintenance

There are two different cases to consider. One case is when a stateful forwarding mechanism is used. A node is assumed to be able to detect a link break by receiving a link layer feedback signal from the MAC protocol, or not receiving hello packets for a certain period of time. When a route is disconnected, the immediate upstream node of the broken link sends a route error packet to the source node of the session to notify the route invalidation. Nodes along the path relay this message to the source node after removing the entry locally maintained for the session. Upon receiving such a message, the source initiates a route re-discovery process to search for a QoS-satisfied route to the intended destination if it still has data to send.

Another case is when stateless forwarding is implemented. In this case, if a returned route is the shortest path connecting the source–destination pair, the source can directly check whether $\text{delay}(P(s, d)) \leqslant \Delta$ is satisfied periodically or upon a change of the path delay value by looking up its routing table for the entry stored for the destination; else if the returned route is a concatenated route, we suggest that a periodic query message is sent from the source to the *relay_node* to query the delay value on the shortest path $P(relay\_node, d)$ from the relay node to the destination. After receiving a response message from the relay node, which contains the requested information, the source can check whether $delay(P(s, relay\_node)) + delay(P(relay\_node, d)) \leqslant \Delta$ still holds. Once a violation of the delay requirement is detected, the source can enforce a QoS-route re-discovery process to search for another route satisfying the delay constraint.

5. BANDWIDTH-CONSTRAINED ROUTING

This section presents a bandwidth-constrained routing protocol, referred to as BWR, to deal with the BCLC issue defined in Section 2.2. The BWR protocol follows the philosophy in implementing QoS routing through alternate paths, as presented earlier. Before presenting the BWR protocol, we first give a discussion related with the issue of path width calculation for MANETs.

5.1. Path width

Here, we discuss some issues associated with determining the width of a path with a length of two or more hops in MANETs. In (3), we show that the width of a path is determined by the bottleneck link on the path. However, this relationship does not always hold in MANETs due to

the issue of exposed and hidden nodes. In details, for a directed path $p = (1, 2, k-1, k, k+1, \ldots, n), n > 2$, where node 1 is the source, node n is the destination, the transmissions on any neighbouring links, e.g. $(k-1, k)$ and $(k, k+1)$, cannot occur simultaneously because each node, say k in this example, has only one antenna for both transmitting and receiving; the transmissions on any next neighbouring links, e.g. $(k-1, k)$ and $(k+1, k+2)$, cannot be active at the same time because $k-1$ and $k+1$ are senders hidden from each other. For these reasons, for a path with a length of exact two hops or a path with a length of three or more hops, its path width is, at best, a half (for the former) or a third (for the latter) of the channel capacity, instead of being determined by the bottleneck link. For a TDMA system, path width can also be affected by the scheduling of free time slots on each link consisting of a path (see References [12–15] for more details).

The BWR protocol designed here, however, can work well with different MAC protocols supporting resource reservation. This is because BWR works in a hop-by-hop manner in determining the feasibility of a route. A path width calculating function can be included in BWR, to determine the width of the partial path probed thus far, in a hop-by-hop manner, by considering the used medium access protocol (e.g. as do in References [14, 15]). Equation (3) shown earlier is just to ease the explanation of our routing protocol and can be modified based on the selected MAC protocol.

Next, we present procedures of the BWR protocol to search for bandwidth-satisfied routes. For the BWR protocol to execute successfully, routing information maintained at each node is the same as that specified in Section 4.1 as required by the DLR protocol except replacing the delay value of the least-cost path with the width information of the corresponding path.

5.2. *Procedures for BWR*

Suppose that an application wishes to establish a connection between a source node s and a destination node d with a bandwidth requirement B. Upon receiving such a request, source s first checks the feasibility of the *least cost* path from s to d. If $\text{width}(P(s,d)) \geqslant B$, the routing process returns this path. Otherwise, a *probing* process is executed to search for an alternate route with enough residual bandwidth. For this purpose, the source node first composes a *probe* packet that contains the following information: s, d, s_{id}, $s_{\text{sequ\_num}}$, B, D, $\text{width}(p) \leftarrow$ INFINITY,

Procedure *rcv_probe*
1. **if** the message was received via the link, over which node v forwards data packets to the source s, and
2. $h(p)+h(v, d) \leq D$, and
3. $width(p) \leftarrow min\{width(p), width(u, v)\} \geq B$
4. **if** $width(P(v,d)) \geq B$ //A feasible concatenated path is found
5. $h(p) \leftarrow h(p)+h(P(v,d))$
6. A *reply* message is sent back to the source node, with the following information: end-to-end path cost $h(p)$, the identifier of relay node v, and, optionally, $width(p)$;
7. **else**
8. Re-broadcast the *probe* message to its immediate neighbors after including the updated $width(p)$ *and* $h(p)$.
9. **end if-else**
10. **else**
11. The *probe* message is simply discarded.
12. **end if-else**
13. **end** *rcv_probe*

Figure 6. Procedure for BWR protocol at an intermediate node v upon receiving a *probe* message from an immediate neighbour u.

$h(p) \leftarrow$ zero. Most of these symbols were defined and explained in the earlier-presented DLR protocol.

The source s then broadcasts the *probe* message to its immediate neighbours and starts a timer for collecting *reply* message(s). The receipt of a *reply* message at the source s means the discovery of a feasible path. If multiple feasible paths are found, it selects the one with the minimal hop distance or the widest one. If timed out without receiving any *reply* message, the source can perform a QoS re-negotiation process by reducing the bandwidth requirement or transmit data packets of the session *temporarily* as best effort traffic.

Figure 6 gives the procedures of BWR to be implemented at an intermediate node v upon receiving a *probe* message from one of its immediate neighbours, say u. The procedures for BWR work in a way similar to that for DLR. After identifying a constrained route, either a stateful or a stateless forwarding mechanism as specified earlier can be used for forwarding data packets belonging to the session.

6. SIMULATION RESULTS

In this section, we conduct extensive simulations to evaluate the performance of the protocols that we designed. The results about the delay-constrained routing protocol DLR were presented. Three performance metrics, success ratio, average message overhead, and average path cost were evaluated in our simulations and each of them is defined as follows, respectively:

$$\text{success ratio} = \frac{\text{total number of routed connection requests}}{\text{total number of connection requests}} \tag{7}$$

$$\text{average message overhead} = \frac{\text{total number of control messages sent}}{\text{total number of connection requests}} \tag{8}$$

$$\text{average path cost} = \frac{\text{total cost of all established paths}}{\text{total number of routed connection requests}} \tag{9}$$

In our simulations, an ideal MAC protocol is assumed, which can resolve the problems of hidden and exposed nodes and guarantee delivery. Nodes can enable a promiscuous receiving mode. Sending a *probe* or a *reply* message over a link is counted as a control message.

The network topologies used in our simulations are randomly generated. Forty nodes are uniformly distributed within a 15×15 m^2 area. All nodes are static. The transmission radius of a node is 3.5 m, which is the same for all nodes in the network. A link is added between two nodes if they are located within the transmission range of each other. The source node and the destination node of each communication request are randomly selected. Each link has unit cost and is associated with a delay value uniformly distributed in the range [1, 50 ms]. Three routing algorithms were evaluated in our simulations: the flooding algorithm (Flood), the shortest path algorithm (SP), and the DLR algorithm presented in this paper.

The flooding algorithm floods route-searching messages across the entire network for a delay-constrained route to the intended destination. Each route-searching message carries the accumulated delay on the path that it has traversed, and the message proceeds only if the accumulated delay does not exceed the delay bound. In Reference [21], Shin and Chou show that, when certain scheduling policies are used at nodes and control messages are set to the appropriate priority, the route-searching messages travel at speeds according to the link delay.

Hence, the message travelling along the least-delay path arrives first. With this assumption, an intermediate node needs only to forward the first received message and discard all successively received ones. As a result, there will be at most one message sent along each link. The algorithm can find a feasible route when such a path exists and hence it is the optimal solution w.r.t. success ratio. The flooding algorithm does not have an efficient mechanism for the termination detection. It selects the path taken by the first routing message that the destination receives. The advantage of this flooding algorithm is that it does not require any global state information or a path-calculating process, but, at the expense of much more communication overhead in the process of searching a constrained path.

The SP algorithm returns the minimal-hop path connecting the source–destination pair of a request if this path is feasible or otherwise rejects the request. SP algorithm requires each node to maintain a distance vector consisting of $|V|-1$ entries, one for each other node in the network, which records the hop count distance and next hop information as well as a piece of additional information recording the end-to-end delay value of the mini-hop path, which is the same as specified in Section 4.1 as required by DLR.

6.1. Success ratio

Figure 7 compares the *success ratios* of the three algorithms. Unless otherwise specified, for each measured value presented in this section, each of the studied algorithms was run repeatedly with new random communication requests until half-width of the resulting confidence interval $<7.5\%$ of the mean value using 95% confidence level was achieved by using the sequential batch means method in Reference [22, p. 92]. In Figure 7, we can see that the *success ratio*, in acquiring a delay-constrained path by using each of the studied algorithms, increases with the relaxation of delay constraint. The flooding algorithm, as expected, has the best success ratio since it can always identify a feasible path when one exists. The *success ratio* of DLR is very close to that of the flooding algorithm. The SP algorithm has a relatively low success ratio in constrained-path discovery. Figure 8 shows the *success ratios* of DLR and SP relative to that of the flooding algorithm. We can see that, in terms of *success ratio*, DLR and SP are up to 5 and 39% worse than the flooding algorithm, respectively, which occurs when the delay constraint is tight. With the relaxation of delay bound, the performance gap w.r.t. success ratio between them decreases.

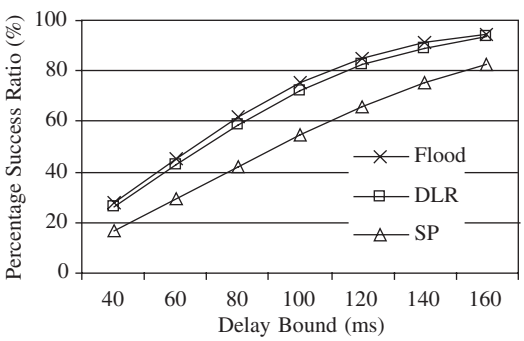

Figure 7. Comparisons of success ratio by different algorithms for acquiring a delay-constrained path versus delay bound.

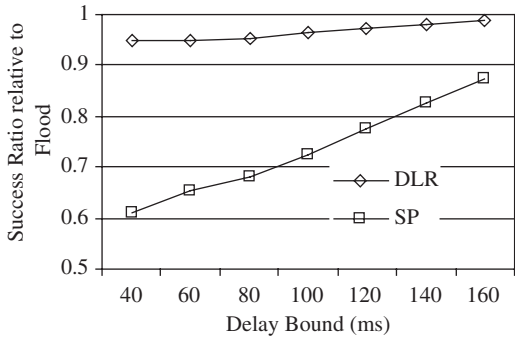

Figure 8. Comparisons of success ratio relative to the flooding algorithm for acquiring a delay-constrained path versus delay bound.

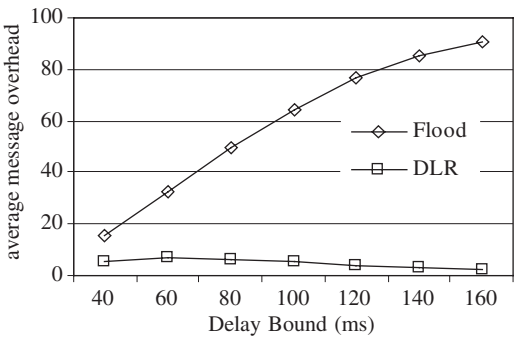

Figure 9. Comparisons of average message overhead by different algorithms for discovering a delay-constrained path versus delay bound.

Additional experiments were conducted as well to determine the appropriate value of parameter H. Through extensive simulations, we set H as two in all other experiments. That is, the length of a returned path in our simulations can be at most two hops longer than that of the minimum-hop path for each communication request. In our simulations, little gain w.r.t. *success ratio* was observed by increasing the value of H further.

6.2. Communication overhead

Figure 9 compares the *average message overhead* introduced in searching for a delay-constrained path by using each of the studied algorithms. The flooding algorithm has a prohibitively high communication overhead. In contrast, DLR has a very low message overhead as observed in Figure 9. Here, the stateless forwarding mechanism presented in Section 4.3 is applied to DLR. With such a forwarding mechanism, the SP algorithm (not shown in Figure 9) introduces no communication overhead for a constrained-route acquisition since a source node can locally determine the feasibility of a mini-hop path that originates from the node itself, without enforcing any reactive probing process. In Figure 9, we can see that the total number of control messages consumed for discovering a delay-constrained path by using DLR, in average, is up to seven per communication request. The reasons of this low communication overhead are as

follows. First, the directed search method employed in DLR can effectively restrict the path-searching scope and avoid blind flooding. Second, no path-searching process is enforced if the direct path connecting the source–destination pair of a request meets the delay requirement since this path is known to be the optimum solution in this case.

6.3. Cost performance

Figure 10 compares the *average path* cost (w.r.t. hop count) of the flooding algorithm and DLR. The cost performance of SP is not shown here because SP always returns the least cost path if it is feasible or otherwise rejects a request. In Figure 10, we can see that DLR has a lower average path cost than the flooding algorithm. The reason is as follows. DLR can always return the mini-hop path if this path is feasible, or else it selects the path with the minimal cost among multiple possible alternate constrained routes. On the other hand, the flooding algorithm always returns the least delay path (if feasible), which does not consider any cost optimization. The reason that average path cost due to either protocol increases with the relaxation of delay bound is as follows. This relaxation can increase the success probability to accommodate those requests, whose sources and destinations are connected with (mini-hop) paths with a length of more hops and which would be likely to get rejected when delay bound is tight. Figure 11 shows

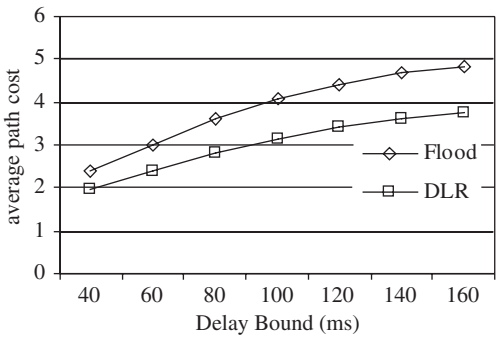

Figure 10. Comparisons of different algorithms w.r.t. average path cost versus delay bound.

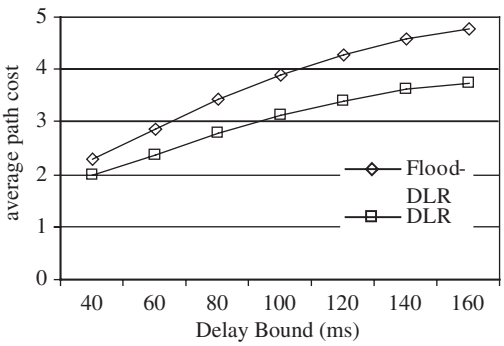

Figure 11. Comparisons of different algorithms w.r.t. average path cost using the same set of established paths versus delay bound.

a more fairer comparison, where only those connections that can be established by both Flood and DLR are considered, and this figure shows that DLR outperforms the flooding algorithm w.r.t. cost performance to almost the same degree as observed in Figure 10.

6.4. *Mobility test*

This test is to evaluate the performance of our designed protocols in the presence of node mobility. The mobility model used here is similar to that in References [23, 24]. Every node stays at its current location for a period of time, which is called the stationary time, and then it moves to another randomly chosen location. Each node repeats this behaviour, alternatively staying and moving to another location. The velocity of node movement is randomly selected between 0.5 and 1 m/s. Note that we place all nodes within an area of 15×15 m^2 and the transmission range of each node is 3.5 m. The time a node takes to reach a new location is called the moving time. The mobility ratio of a node is defined as follows:

$$\text{mobility ratio} = \frac{\text{total moving time}}{\text{total moving time} + \text{total stationary time}} \tag{10}$$

By adjusting the stationary time, we can change the mobility ratio.

When a new request arrives, it is routed along a feasible path by using DLR. The period during which the path meets the delay bound is called the *QoS time*. During this period, the delay requirement is said to be satisfied. Once a violation of the delay requirement is detected, a route recovery process is enforced. The time that it takes to re-discover a new feasible path is called the best effort time, during which data packets belonging to the session are sent as best effort traffic.

We use the performance metric *QoS ratio*, to measure the percentage of a connection's lifetime during which the required QoS is ensured.

$$\text{QoS ratio} = \frac{\text{total QoS time}}{\text{total QoS time} + \text{total best effort time}} \tag{11}$$

We study how the mobility ratio affects the QoS ratio. Before providing our results, we will first present several implementation decisions made in the simulations. Again, the simulation run continuously until half-width of confidence intervals $<7.5\%$ of the mean value using 95% confidence level is achieved. Moreover, only samples in steady state were counted and those in transient phase were eliminated. After a QoS violation is detected on the working route, a route recovery process is enforced immediately to discover a new feasible path (if any). If the violated QoS cannot be satisfied again, packets belonging to the session are transmitted as best-effort traffic until a path with satisfied delay property can be found again. Forty connection requests are simulated per mobility ratio. Each connection lasts through the simulation. The source and destination node of each request are randomly selected. We assume there is an underlying unicast routing protocol, which can always forward data packets along the shortest path whenever a path exists. This implementation decision decouples the QoS routing protocol that we designed from the characteristics of any underlying unicast routing protocol, and allows us to focus on the protocol' own behaviour. Since no specific MAC protocol is implemented, the delay that a control message experiences when going through a link (including the queuing delay at the sending node) is set to be 100 ms. This relatively large link-traversal delay can give us a conservative evaluation of the performance of our protocol.

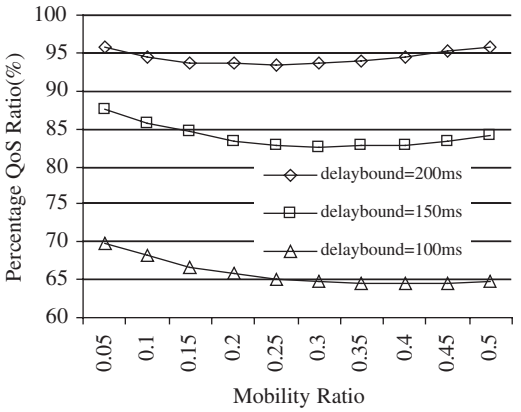

Figure 12. Percentage QoS ratio versus node mobility with varying delay requirements.

In Figure 12, we can see, when mobility ratio is low, QoS ratio decreases slightly as mobility ratio increases. This is because when mobility ratio is low, the movement of any intermediate node on a QoS route is likely to lead to the break of the QoS path and followed by a certain period of latency for acquiring a new QoS path during which time the traffic belonging to the session will be sent temporally as best effort traffic. As mobility ratio increases further, this tendency diminishes. This is because the movement of other (not-on-QoS-path) nodes can largely remedy the deviation of those on-QoS-path intermediate nodes from their on-path positions, in which case no any enforcement of route re-discovery is triggered. Remember that the stateless forwarding mechanism is applied to DLR and this implementation decision makes the routing protocol insensitive to those local changes on a path segment consisting of a QoS path as long as such change does not lead to a violation of the end-to-end delay requirement. The result in Figure 12 indicates that DLR can maintain a relatively steady QoS ratio in the presence of node mobility. On the other hand, since a stateless forwarding mechanism does not require intermediate nodes on a QoS path to locally maintain flow-specific entry or to perform resource reservation, a stateless QoS routing protocol is more suitable for providing *soft QoS*.

7. CONCLUSIONS

In this work, we studied the issue of route selection subject to QoS requirement(s) in MANETs. Our method is to search for alternate routes to accommodate communication requests with QoS requirement(s) when the shortest path is not qualified. To achieve simple QoS-based forwarding, we restrict our scope to those alternate paths consisting of at most two concatenated path segments connected at a *relay node*. Within a path segment, all nodes use the same routing metric for packet forwarding. To identify an appropriate *relay node* in a fully decentralized manner, we presented the design of a directed search method to limit the searching scope, which aims at achieving graceful tradeoff between communication overhead and success probability in

acquiring QoS routes. Efficient hop-by-hop protocols were designed to support delay- and bandwidth-constrained routing, respectively. The concatenated nature of a returned QoS path enables these protocols support stateless forwarding of QoS traffic. Simulation results demonstrate that the designed protocols can achieve high performance in acquiring QoS-satisfied paths and in utilizing network resources efficiently with low protocol overhead.

REFERENCES

1. Garey MR, Johnson DS. *Computer and Intractability*: *A Guide to Theory of NP-Completeness*. CW. H. Freeman and Company: New York, 1979.
2. Bertsekas D, Gallager R. *Data Networks* (2nd edn). Prentice-Hall: Englewood Cliffs, NJ, 1992.
3. Chen S, Nahrstedt K. An overview of quality of service routing for next-generation high-speed networks: problems and solutions. *IEEE Network Magazine*, *Transmission and Distribution of Digital Video* 1998; **12**(6):64–79 (special issue).
4. Kuipers F, Mieghem PV, Korkmaz T, Krunz M. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Communications Magazine* 2002; **40**(12):50–55.
5. Widyono R. The design and evaluation of routing algorithms for real-time channels. *Technical Report ICSI Tr*-94-024, University of California at Berkeley, International Computer Science Institute, June 1994.
6. Hassin R. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research* 1992; **17**(1):36–42.
7. Lorenz DH, Raz D. A simple efficient approximation scheme for the restricted shortest path problem. *Operational Research Letters* 2001; **28**(5):213–219.
8. Reeves DS, Salama HF. A distributed algorithm for delay-constrained unicast routing. *IEEE/ACM Transactions on Networking* 2000; **8**(2):239–250.
9. Sun Q, Langendorfer H. A new distributed routing algorithm for supporting delay-sensitive applications. *Computer Communications* 1998; **21**:572–578.
10. Sriram R, Manimaran G, Siva Ram Murthy C. Preferred link based delay-constrained least cost routing in wide area networks. *Computer Communications* 1998; **21**:1655–1669.
11. Wang Z, Crowcroft J. Quality-of-service routing for supporting multimedia applications. 1996; **14**(6): 1228–1234.
12. Chen TS, Gerla M, Tsai JT. QoS routing performance in a multi-hop, wireless networks. In *Proceedings of ICUPC'97*, San Diego, California, USA, October 1997; 557–561.
13. Lin CR, Liu J-S. QoS routing in ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications* 1999; **17**(8):1426–1438.
14. Lin CR. On-demand QoS routing in multihop mobile networks. In *Proceedings of IEEE Infocom'01*, Anchorage, Alaska, April 2001; 1735–1744.
15. Zhu C, Corson MS. QoS routing for mobile ad hoc networks. In *Proceedings of IEEE Infocom'02*, New York, USA, June 2002; 958–967.
16. Chen S, Nahrstedt K. Distributed quality-of-service routing in ad hoc networks. *IEEE Journal on Selected Areas in Communications* 1999; **17**(8):1488–1505.
17. Chen S. Routing support for providing guaranteed end-to-end quality-of-service. *Ph.D. Thesis*, University of Illinois at Urbana–Champaign, 1999.
18. Perkins C, Bhagwat P. Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers. In *Proceedings of ACM SIGCOMM'94*, London, UK, August 1994; 234–244.
19. Perkins C. IP encapsulation within IP. *IETF RFC* 2003 (Standards Track), October 1996.
20. Perkins C. Minimal encapsulation within IP. *IETF RFC* 2004 (Standards Track), October 1996.
21. Shin KG, Chou C-C. A distributed route-selection scheme for establishing real-time channel. In *Proceedings of the Sixth IFIP International Conference on High Performance Networking* (*HPN*'95), Palma de Mallorca, Spain. September 1995; 319–329.
22. Bratley P, Fox BL, Schrage LE. *A Guide to Simulation*. Springer: Berlin, 1987.
23. Johnson DB, Maltz DA. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Imielinski T, Korth H (eds). Chapter 5. Kluwer Academic Publishers: Dordrecht, 1996; 153–181.
24. Yoon J, Liu M, Noble B. Random waypoint considered harmful. In *Proceedings of IEEE Infocom'03*, San Francisco, California, USA, March 2003; 1312–1321.

AUTHORS' BIOGRAPHIES

Baoxian Zhang (bxzhang@site.uottawa.ca) received his BS MS, and PhD degrees in Electrical Engineering from Northern Jiaotong University (now Beijing Jiaotong University). Beijing, People's Republic of China in 1994, 1997, and 2000, respectively. He is now a Postdoctoral fellow with the School of Information Technology and Engineering, University of Ottawa. His research interests include routing algorithm and protocol design, QoS management, multicast communications, and wireless *ad hoc* networks. He is a member of IEEE.

Hussein Mouftah joined the School of Information Technology and Engineering (SITE) of the University of Ottawa in September 2002 as a Canada Research Chair Professor (Tier I). He has been with the Department of Electrical and Computer Engineering at Queen's University since 1979, where he was prior to his departure in August 2002, a Full Professor and the Department Associate Head. He served as Editor-in-Chief of the IEEE Communications Magazine (1995–1997) and IEEE Communications Society Director of Magazines (1998–1999). He is the author or coauthor of three books and more than 700 technical papers and 8 patents in the area of broadband packet switching networks, mobile wireless networks and quality of service over the optical Internet. He is the recipient of the 1989 Engineering Medal for Research and Development of the Association of Professional Engineers of Ontario (PEO). He is the joint holder of the Best Paper Award for a paper presented at SPECTS'2002, and the Outstanding Paper Award for papers presented at the IEEE HPSR'2002 and the IEEE ISMVL'1985. Also he is the joint holder of a Honorable Mention for the Frederick W. Ellersick Price Paper Award for Best Paper in the IEEE Communications Magazine in 1993. Dr Mouftah is a Fellow of the IEEE (1990).

