Telecommunication Systems 19:3,4, 329–347, 2002

A Multi-Path QoS Routing Protocol in a Wireless Mobile Ad Hoc Network

WEN-HWA LIAO, SHU-LING WANG and JANG-PING SHEU
whliao@axp1.csie.ncu.edu.tw; sheujp@csie.ncu.edu.tw
Department of Computer Science and Information Engineering, National Central University, Chung-Li, 320, Taiwan

YU-CHEE TSENG* yctseng@csie.nctu.edu.tw
Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu, 300, Taiwan

Abstract. A mobile ad hoc network (MANET) is one composed of a set of mobile hosts capable of communicating with each other without the assistance of base stations. This paper considers the QoS (quality-of-service) routing problem in a MANET, which is important for many real-time multimedia applications. We propose an on-demand protocol for searching for a multi-path QoS route from a source host to a destination host in a MANET, where a *multi-path* is a network with a source and a sink satisfying certain bandwidth requirement. Existing works all try to find a *uni-path* to the destination. The basic idea is to distribute a number of tickets from the source, which can be further partitioned into subtickets to search for a satisfactory multi-path. Through simulations, we justify that the value of our multi-path protocol is in its flexibility: (i) when the network bandwidth is very limited, it can offer a higher success rate to find a satisfactory QoS route than those protocols which try to find a uni-path, and (ii) when the network bandwidth is sufficient, it can perform almost the same as those protocols which try to find a uni-path (in both routing overhead and success rate).

Keywords: mobile ad hoc network (MANET), multi-path, quality-of-service (QoS), routing, wireless communication

1. Introduction

The advancement in wireless communication and economical, portable computing devices have made *mobile computing* possible [Forman and Zahorjan, 7]. One research issue that has attracted a lot of attention recently is the design of a *mobile ad hoc network (MANET)*. A MANET is one composed of a set of mobile hosts which can communicate with one another and roam around at their will. No base stations are supported in such an environment. Due to considerations such as radio power limitation, power consumption, and channel utilization, a mobile host may not be able to communicate directly with all other hosts in a *single-hop* fashion. In this case, a *multi-hop* scenario occurs, where the packets sent by the source host are relayed by several intermediate hosts before reaching

* Corresponding author.

the destination host. Applications of MANETs occur in situations like battlefields or major disaster areas, where networks need to be deployed immediately but base stations or fixed network infrastructures are not available. A working group called "manet" has been formed by the Internet Engineering Task Force (IETF) to study the related issues and stimulate research in MANETs.

Since MANETs are characterized by its fast changing topology, extensive research efforts have been devoted to the design of routing protocols for MANETs [Broch et al., 2, 3; Gerla and Tsai, 9; Haas and Pearlman, 10; Jiang et al., 11; Johnson and Maltz, 12; Liao et al., 14; Perkins and Bhagwat, 17, Perkins and Royer, 18; Royer and Toh, 19; Tseng et al., 21]. These works all address some of the following issues: how to discover a route from a source node to a destination, how to maintain a route while it is being used, and how to deliver data packets to the intended destination host. However, these protocols, when searching for a route to the destination, are only concerned with shortest-path routing and the availability of multitude routes in the MANET's dynamically changing environment. That is, only best-effort data traffic is provided. Connections with quality-of-service (QoS) requirements, such as multimedia with delay and bandwidth constraints, are less frequently addressed.

Some works have started to focus on the QoS issue in a MANET. A ticket-based QoS routing protocol is proposed in [Chen and Nahrstedt, 4] to find a route satisfying certain bandwidth and delay constraints. The basic idea is to use tickets to confine the number of route-searching packets to avoid an unwise blind flooding. In [Lin and Liu, 15], a scheme to calculate the end-to-end bandwidth of a path under a TDMA-over-CDMA mechanism is proposed. The hidden-terminal problem and the bandwidth allocation problem are also addressed. Distributed multiple access protocols to support QoS are considered in [Sobrinho and Krishnakumar, 20], where it is shown how to guarantee real-time packets being transmitted in a MANET with priority and without collision.

The purpose of this paper is to address QoS routing in a MANET environment. QoS routing has been studied extensively in the field of wireline networks [Chen and Nahrstedt, 5]. Certainly, whether in a stand-alone MANET or in a MANET connected to a wireline network, QoS routing is still necessary. In this paper, we propose an on-demand protocol for searching for a multi-path QoS route from a source host to a destination host in a MANET, where a *multi-path* is a network with a source and a sink satisfying certain bandwidth requirements. Our protocol distinguishes from the work of [Chen and Nahrstedt, 4] in that they try to find a *uni-path* to the destination based on a costly *reactive* approach (namely, DSDV [Perkins and Bhagwat, 17]). The basic idea is similar to the work in [Chen and Nahrstedt, 4] by distributing a number of tickets from the source. However, we allow a ticket to be further partitioned/split into subtickets to search for a satisfactory multi-path. Through simulations, we justify that the value of our multi-path protocol is in its flexibility:

(i) when the network bandwidth is very limited, it can offer a higher success rate to find a satisfactory QoS route than those protocols which try to find a uni-path, and

(ii) when the network bandwidth is sufficient, it can perform almost the same as those protocols which try to find a uni-path (in both routing overhead and success rate).

The rest of the paper is organized as follows. Section 2 presents some background and motivation. Our protocol is developed in section 3. Experimental results are in section 4. Section 5 concludes the paper.

2. Background and motivation

The *mobile ad hoc network (MANET)* distinguishes itself from traditional wireless networks by its dynamic changing topology, no base-station support, and the need of multihop communication ability. In a MANET, a mobile host is free to move around and may communicate with others at anytime. When a communication partner is within a host's radio coverage, they can communicate directly with each other in a one-hop manner. Otherwise, a route consisting of several relaying hosts is needed to forward messages from the source to the destination in a multihop fashion.

Existing ad hoc routing protocols may generally be categorized as *table-driven* and *on-demand*. Table-driven protocols attempt to maintain consistent up-to-date routing information from each node to every other node in the network (e.g., DSDV [Perkins and Bhagwat, 17] and CGSR [Chiang, 6]). Contrarily, on-demand protocols create routes only when desired by the source node (e.g., DSR [Johnson and Maltz, 12] and AODV [Perkins and Royer, 18]). A hybrid of these approaches is also possible (e.g., ZRP [Haas and Pearlman, 10]). To assist routing, some protocols even adopt location information in their route discovery and maintenance procedures (e.g., LAR [Ko and Vaidya, 13] and GRID [Liao et al., 14]). However, all of these protocols are only concerned the existence of a route without guaranteeing its quality.

It is difficult to provide QoS in a MANET due to its dynamic nature. First, unlike wireline networks, precise network topology information is unavailable. Second, mobile hosts may join, leave, and rejoin at any time and any location; existing links may disappear and new links may be formed as mobile hosts moves. Hence, established paths can be broken at any time. In the following, we review the QoS routing protocol by [Chen and Nahrstedt, 4]. Then we will motivate our work in this paper.

2.1. QoS routing protocols for MANETs

Real-time applications (such as video and audio transmissions) need QoS (quality of service) support. Two widely used QoS constraints are *bandwidth* and *delay* requirements. The first problem is solvable in polynomial time, given precise state information, while the second is NP-complete [Chen and Nahrstedt, 4].

In [Chen and Nahrstedt, 4], a *ticket-based* protocol is proposed to support QoS routing. This protocol maintains the end-to-end state information at every node for every possible destination. This information is updated periodically by a distance-vector-like protocol (namely, DSDV [Perkins and Bhagwat, 17]). A source node s, on requiring a

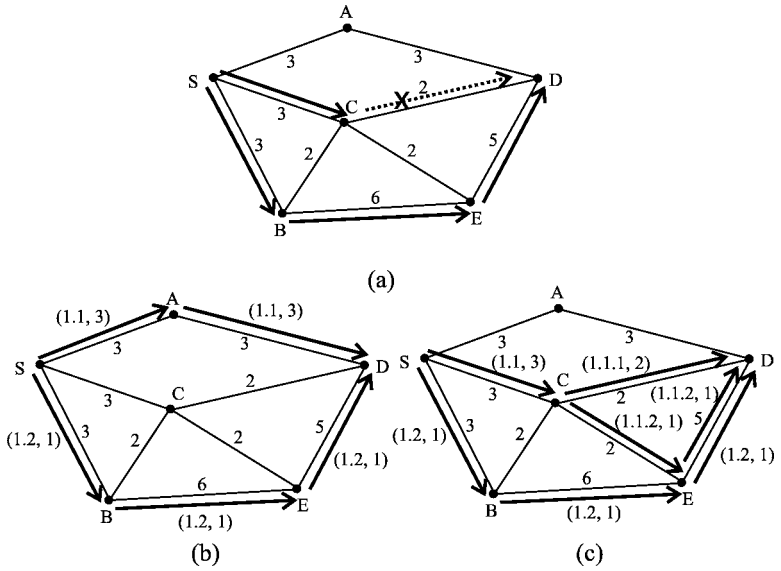

Figure 1. Ticket-based QoS routing example: (a) searching for a route from S to D with bandwidth 3 using two tickets, and (b), (c) two successful multi-path routing examples. In (b) and (c), the 2-tuple on each link means the ticket identity and the reserved bandwidth.

QoS route, can issue a number of probing packets each carrying a *ticket*. Each probe is in charge of searching for one path, if possible. The basic idea of using tickets is to confine the number of route-searching packets to avoid a blind flooding (flooding in a MANET is unwise, according to [Ni et al., 16]). One guideline is: the tighter the QoS requirements are, the more tickets we should issue. Each probe, on reaching any intermediate node, should choose one outgoing path that satisfies the QoS requirements. If a probe enters a node that has no outgoing link satisfying the QoS requirements, the intermediate node sends an *invalidated ticket* to the destination node. To save the number of probing packets, several tickets may be carried by one packet and, if so, the probe can be split in the midway into multiple probes, each carrying some of the tickets and being responsible of searching a different downstream subpath. Thus, the maximum number of probes at any time is bounded by the total number of tickets. For example, figure 1(a) shows a MANET, where the number associated with each link is its corresponding bandwidth. The arrows show the progress of two tickets issued from S to D. It is assumed that a path of bandwidth 3 is required, so the probe going through C fails, while that through B and E succeeds.

2.2. *Observations and motivations*

While the ticket-based protocol in [Chen and Nahrstedt, 4] solves the path-finding problem in an elegant way, the protocol may experience a high failure rate when the bandwidth demand is large. For example, consider the example in figure 1(a) again. If we require a route from S to D with bandwidth 4, then the ticket-based protocol will fail

because apparently no single route satisfies this constraint. This has motivated us to investigate the possibility of "multi-path QoS routing". Specifically, we define a *multi-path route* as one that can contain several subpaths with the destination as the sink. Figure 1(b) shows an example of using two paths to provide the required bandwidth of 4. Paths may even split and merge. For example, figure 1(c) shows a multi-path with bandwidth of 4. As will be shown later, finding a multi-path route turns out to be a quite complicated problem, which interests us from both theoretical and practical points of view.

2.3. Channel model for multi-path routing

In this subsection, we discuss our channel model to support multi-path routing. In a multi-path, each host may have more than one incoming links and more than one outgoing links. Except the source and destination hosts, the total incoming bandwidth of any host should be equal to the total outgoing bandwidth. For each link in a multi-path, we assume that its communication activity is independent of those of its neighbors. To support this assumption, we assume that each host has multiple transmitters and receivers that can work independently and simultaneously. A link refers to a pair of transmitter and receiver located at two neighboring hosts. Each link is assigned a code that is distinct from those codes used by its two-hop neighbors to avoid collision. A code could be a frequency band under the FDMA model or an orthogonal code under the CDMA model. The code assignment problem has been studied in [Bertossi and Bonuccelli, 1; Garcia-Luna-Aceves and Raju, 8], and thus here we regard it as an independent issue.

Every link has an equal bandwidth. However, the bandwidth of a link is further organized based on a TDMA model. The concept of time slot is adopted, and continuous time slots are grouped into fixed-size frames. A multi-path that passes a link can reserve some particular time slot(s) of continuous frames. In this way, bandwidth can be guaranteed and a link can be shared by multiple multi-paths. For example, in figure 2, we demonstrate a host C with two incoming links from A and B with 1 and 3 time slots reserved, respectively, and two outgoing links to D and E with 2 time slots reserved each. Thus a bandwidth of 4 time slots are reserved for the multi-path passing host C.

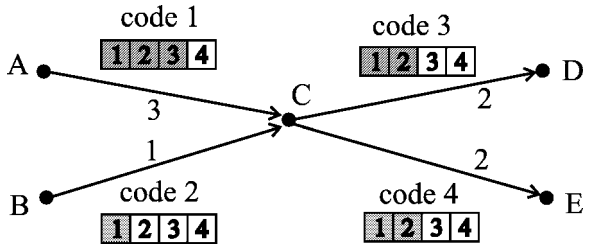

Figure 2. An example of our channel model to support multi-path routing.

3. Our multi-path QoS routing protocol

3.1. Protocol overview

Our protocol will follow an on-demand style to allocate bandwidth. So no global information will be collected in advance before a QoS route is required. A mobile host only knows the available bandwidth to each of its neighbors. When a source node S needs a route to a destination D of bandwidth B, it will send out some probe packets each carrying some tickets. Each ticket is responsible of searching for a multi-path from the source to the destination with an aggregated bandwidth equal to B.

On a ticket/subticket arriving at a node, if the node is not the destination, some bandwidth of a qualified outgoing link will be reserved for this ticket and then the ticket will be sent out through that link. Since we allow a multi-path from S to D, if no link with a sufficient bandwidth exists, the ticket may be split into multiple subtickets, each being responsible of searching for a multi-path with a certain portion of bandwidth B. The destination node will, if possible, receive multiple tickets or subtickets. It will then pick one ticket or a set of subtickets forming a whole ticket and send a reply to the source node. On the reply's way back to the source, the bandwidths reserved by the earlier probes will be confirmed. A reservation that is not confirmed after a time-out period will be released. Below we will discuss our multi-path QoS routing protocol in more details.

3.2. Ticket format

For each bandwidth request, a number of tickets may be sent. In the rest of the discussion, we will call a ticket that has never been split a *whole-ticket*, and one that has been split a *subticket*. However, both whole-ticket and subticket will be called a ticket. As shown below, this can be told from a ticket's content. A ticket will be denoted by $T(S, D, x, y, RID, TID, B, b)$. The meanings of the parameters in the ticket are as follows:

- S: the source host;
- D: the destination host;
- x: sender of the packet carrying the ticket;
- y: receiver of the packet carrying the ticket;
- RID: identity of a bandwidth request. This is unique for each QoS route request;
- TID: identity of a ticket. This is unique for each ticket;
- B: the required bandwidth of the multi-path from S to D;
- b: the required bandwidth of the multi-path from y to D. (So, if this is a subticket, then $b < B$.)

3.3. Ticket splitting and inheritance relation

As mentioned earlier, on a ticket reaching a node from which there is no outgoing link with a sufficient bandwidth, it may be split into several subtickets each responsible for searching for a multi-path with a partial bandwidth. The correctness of our protocol relies on a special representation of ticket identity (*TID*). The format of *TID* is a sequence of numbers separated by periods, i.e., $i_1.i_2.\ldots.i_k$. When a ticket is initiated at the source node, it will be given a unique identity i_1 (unique under the same *RID*). When an intermediate host receives a ticket (whole-ticket or subticket) with identity *TID*, it may decide to split the ticket into subtickets. If so, each subticket will be given an extension number appended after *TID*. Specifically, let the ticket be split into k subtickets. These subtickets will be given identities $TID.1, TID.2, \ldots, TID.k$. This is illustrated in figure 3. The reader may also refer to two real examples in figures 1(b) and (c).

It is critical in our yet-to-be-presented protocol to determine the relationship between tickets. Let $head(T)$ be the first number in a ticket T. Consider two tickets $T_1(\ldots, TID_1, \ldots)$ and $T_2(\ldots, TID_2, \ldots)$. If $head(TID_1) = head(TID_2)$, then they are two subtickets of the same whole-ticket. If TID_1 is a substring of TID_2, then T_1 is a subticket split from T_2. If $head(TID_1) = head(TID_2)$ but none of them is a substring of the other, then they belong to the same whole-ticket, but none of them is an ancestor of the other. These relationships are important in our protocol. We point out some crucial points below:

- In figure 4(a), tickets T_1 and T_2 are two distinct tickets belonging to the same request. When they reach the same intermediate node Y (perhaps at different time), it is not necessary to reserve separate bandwidths for them because they represent the same request. Only a bandwidth of $\max(b_1, b_2)$ has to be reserved.
- In figure 4(b), tickets T_1 and T_2 are two subtickets belonging to the same whole-ticket. In this case, a total bandwidth of $b_1 + b_2$ has to be reserved at the intermediate host B.
- In figure 4(c), tickets T_2 and T_3 are two subtickets belonging to the same whole-ticket T_1, but T_2 is a subticket split from T_1. In this case, a loop is detected and we should discard T_2.

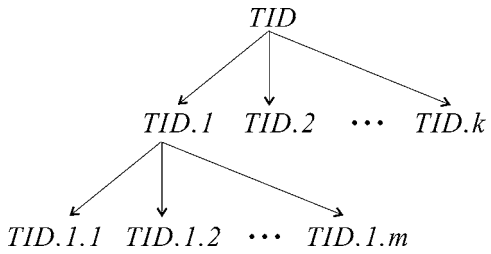

Figure 3. Representation of ticket identities after a ticket is split twice.

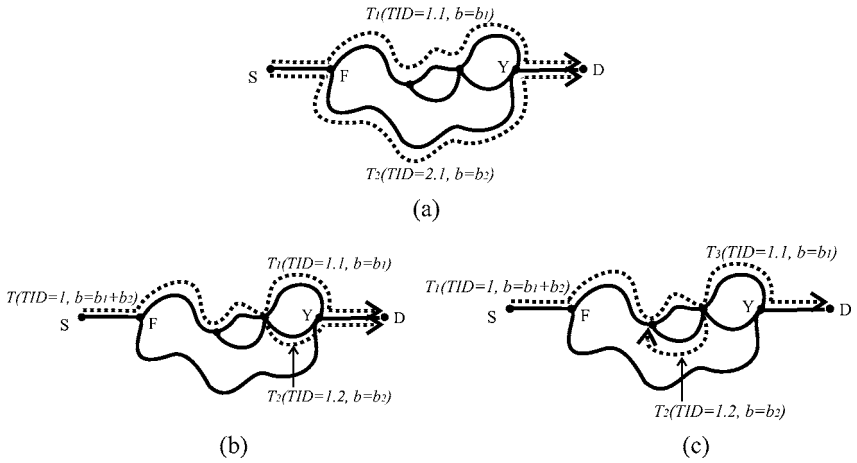

Figure 4. Three possible relationships between tickets: (a) T_1 and T_2 are irrelevant, (b) T_1 and T_2 are irrelevant siblings, and (c) T_2 is T_1's subticket. Note that all these tickets share the same *RID*.

3.4. Loop avoidance

In the route discovery procedure, a probe entering a node that it has visited before typically means a failure of the search. To a protocol, this is an undesirable situation that should be avoided. In traditional approaches such as protocols [Broch et al., 2; Ko and Vaidya, 13; Liao et al., 14; Perkins and Royer, 18], whose goal is to find a path (instead of multi-path) to the destination, loop detection is a simple job of keeping records of the hosts that the probing packet has visited. In the following, we propose a method to avoid the possibility of forming loops to reduce the search failure probability. This can be applied to both traditional approaches and our approach. Since our QoS routing protocol allows multi-paths to appear, this is relatively more important for us.

To prevent loops from happening, we can let a mobile host collect tickets issued by its neighboring hosts, even if the tickets are not intended for itself. This is possible in a MANET due to the radio's broadcasting nature. For example, in Linux, this could mean that mobile hosts turn on their "listening" mode. Specifically, the following rules are used. A host always listens to the medium and collects all tickets issued by its neighbors, no matter if they are intended for itself or not. Now suppose a host receives a ticket T_1 destined to itself. The host will not forward T_1 or subtickets of T_1 to those neighbors who have ever sent a ticket T_2 such that T_1 is a subticket of T_2 (by telling their ticket id's). For example, in figure 5, host A sends B a ticket, B splits the ticket into two subtickets and forwards them to C and E. On C receiving the subticket (with $TID = 1.2$), it will avoid forwarding the subticket to A if it ever heard A's earlier ticket (with $TID = 1$).

Also note that the purpose of the above loop avoidance rules is to increase the success probability of route discovery. It will not affect the correctness of our protocol. This implies that it is alright for a host to miss some tickets issued by its neighbors (perhaps due to collision or mobility).

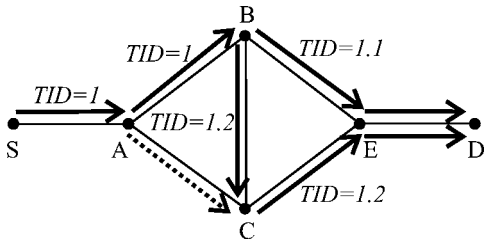

Figure 5. Loop avoidance in forwarding tickets.

3.5. Ticket distribution

Our protocol follows an on-demand fashion to request a route. So when a mobile host S needs a route to a destination host D, it will issue probes carrying tickets to search for a route with a requested bandwidth B. The number of tickets to be issued may depend on many factors, such as how urgent the route is needed, how far the distance from S to D is, how much bandwidth the outgoing links from S have, and the value of B is. Since there is no deterministic value for this number, we will leave it as an experimental parameter to be discovered by simulations.

This part takes care of how tickets are distributed. Consider any mobile host X. Let $Y_1, Y_2, \ldots, Y_k$ be all current neighbors of X known by X. For each Y_i, $1 \leqslant i \leqslant k$, host X should keep the following data structures:

- b_i: the currently available bandwidth from X to Y_i;
- S_i (called the send set): the set of tickets that are sent to Y_i but are not yet confirmed;
- R_i (called the receive set): the set of tickets that are received from Y_i;
- L_i (called the listen set): the set of tickets that are issued by Y_i recently and heard by X.

The following protocol is executed by X when it hears a ticket $T(\ldots, RID, TID, B, b)$ from host Y_i (no matter if it is intended for X or not).

S1. Join T into the listen set L_i.

S2. If the ticket is destined to X, join T into the receive set R_i. Otherwise, exit this procedure.

S3. If X is not the final destination, then T has to be forwarded to some of $Y_1, Y_2, \ldots, Y_k$. Let G be the set containing $Y_1, Y_2, \ldots, Y_k$ excluding those Y_i which contain a ticket T' such that T is a subticket of T' (i.e., sending T to these hosts will form a loop). We sort G according to the following rules:

   (a) Sort $Y_i \in G$ according to the number of tickets $T'(\ldots, RID', TID', \ldots)$ in S_i such that $RID = RID'$ and $head(TID) = head(TID')$ in an ascending order.

   (b) For those $Y_i \in G$ that have a tie in the above sorting, we sort them according to the number of tickets $T'(\ldots, RID', \ldots)$ in S_i such that $RID = RID'$ in an

ascending order. That is, we give priority to those links that have not been tried by the tickets under the same QoS route request.

(c) For those $Y_i \in G$ that have a tie in the above sorting, we sort them according to the remaining bandwidth from X to Y_i in a descending order, where the remaining bandwidth is defined to be

$$b_i - \sum_{\substack{\forall T'(\ldots,RID',TID',B,b')\in S_i :: \\ RID=RID' \wedge head(TID)=head(TID')}} b'.$$

That is, we give priority to those links that have higher remaining bandwidths after subtracting the bandwidths that have been reserved under the same QoS route request.

S4. Then we reserve a bandwidth of b on the link from X to the first host in G. If the remaining bandwidth from X to this host is less than b, then we reserve all of its remaining bandwidth. In this case, we will go to the second host in G and reserve the deficient bandwidth. If the remaining bandwidth from X to this host is not enough, then we reserve all its remaining bandwidth. In this case, we will go to the third host in G. We repeat the above reservation, until a total bandwidth of b is reserved. For each of the above reservations, a ticket will be sent. Also, we will record this by joining the ticket into the appropriate send set S_i.

We illustrate the above steps by an example in figure 6. Suppose that there are two tickets under the same QoS request generated by the source S, which looks for a multi-path to D with bandwidth of 4. For the first ticket (with $TID = 1$), it will go to host A, which has the largest remaining bandwidth. Unfortunately, on reaching host A, the ticket will find that there is not enough bandwidth for it to proceed. So this ticket will be discarded. For the second ticket (with $TID = 2$), the link to C has the highest

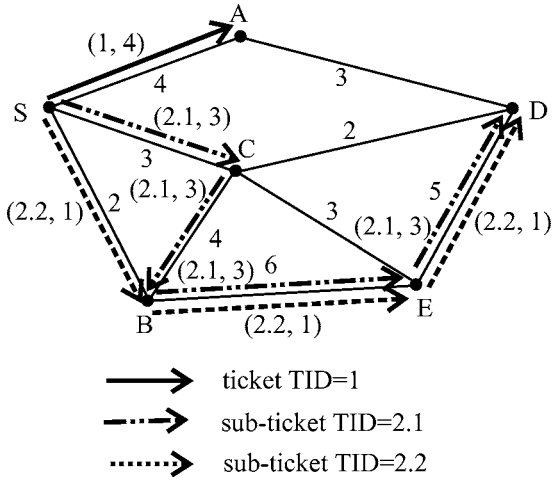

Figure 6. An example of our ticket distribution.

priority. However, its remaining bandwidth is less than 4, so we reserve all its bandwidth. Then we go to the host B, which has the second highest priority. This completes the forwarding of the second ticket. When the subticket with $TID = 2.1$ reaches host C, it will be forwarded to host B according to the above rules. Finally, two subtickets will be received by the destination D.

3.6. Route reply

The purpose of route reply is to confirm the bandwidth reservations that we made in the previous section. Whenever a ticket T reaches the destination D, D can check whether all subtickets under the same *TID* have been received or not. This can be done by summing up their total bandwidths and comparing it to the requested total B. If a satisfactory multi-path has been found, we can send out the reply packets. Each of these reply packets should carry a subticket under the same *TID* to the host where it was sent (this can be tracked back using the receive set R_i). Also, for each subticket being sent, the corresponding entries in the receive set R_i and the listen set L_i should be deleted. These route reply packets should travel backward to confirm the reserved bandwidths, until the source host S is reached. The operations in the intermediate hosts are the same as the above. Our earlier records in the send sets and receive sets will be able to help the reply packets to track back correctly to the source S. We illustrate the route reply in figure 7 based on the example in figure 6.

Note that the destination D may find multiple satisfactory multi-paths. In this case one heuristic is to select the one with the least number of subtickets. Then the bandwidth reservations belonging to the other multi-paths that are not selected will be deleted after a pre-defined timeout period.

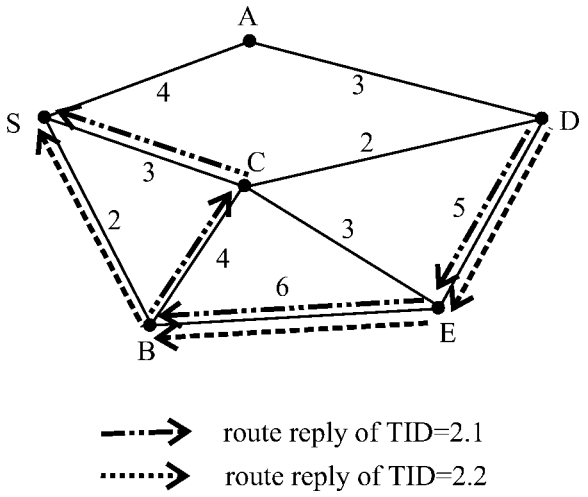

Figure 7. An example of route reply.

4. Experimental results

4.1. The simulation model

We have developed a simulator to evaluate the performance of the proposed multi-path QoS routing scheme (which is abbreviated as MP below). For comparison reason, we also simulated two other protocols: Ticket-based (Tk) and Single-Path (SP). The Tk protocol is that proposed in [Chen and Nahrstedt, 4]. Note that the Tk protocol needs the support of the DSDV protocol [Perkins and Bhagwat, 17] to find out the bandwidth available along each path, and thus is quite different from our on-demand design. For this reason, we also simulated the SP protocol as a referential point to our MP protocol. This protocol is similar to our MP protocol, except that a whole-ticket cannot be split while being forwarded. Thus, we can see how much benefit provided by our protocol.

A MANET in a physical area of size 1000 m $\times$ 1000 m with 50 $\sim$ 100 mobile hosts was simulated. Mobile hosts were randomly distributed in the area. Each mobile host had the same transmission range of 150 meters. Each link owned a bandwidth given by a normal distribution with mean $\mu = 1$–10 and variance $\sigma = 1$–2 following the probability density function $f(x) = (1/(\sqrt{2\pi}\sigma))\mathrm{e}^{-(x-\mu)^2/2\sigma^2}$. For each QoS route request, the source and destination were randomly selected, and the requested bandwidth was a fixed value of 4. For each request, the number of tickets issued could range from 1 to 10. Each simulated result was obtained from an average of 500 runs.

4.2. Observed results

We vary the number of tickets issued and the bandwidth available on each link (the latter is done by adjusting the μ and σ of the link bandwidth probability density function mentioned above). We then compare the success rate to find a satisfactory QoS route and the corresponding overhead for different protocols. We make observations from several aspects.

A. Effect of link bandwidth. Figures 8–11 show the success rate of finding a QoS route for different protocols. Based on the change of bandwidth, we make the following two observations. First, when comparing Tk and SP, we see that when the bandwidth is relatively low, Tk will have a higher success rate than SP. The reason is that under such situations it is difficult to find a satisfactory route. So the Tk protocol, which is based on the costly DSDV protocol, has a higher success rate. However, as there is more bandwidth on links, the SP protocol will gradually outperform the Tk protocol. Our explanation is that we have used a loop avoidance scheme to reduce the possibility of search failure. From the figures, we see that our loop avoidance scheme is quite effective.

Second, when comparing SP and MP, we see that at lower link bandwidths, the MP protocol will have a higher success rate. This is because the bandwidth is scarce, and thus using multi-paths is necessary to resolve the difficulty in finding a satisfactory route. As the link bandwidth gradually increases, the MP and SP protocols will have about the same success rate, because finding a satisfactory route is not so difficult. The trend when

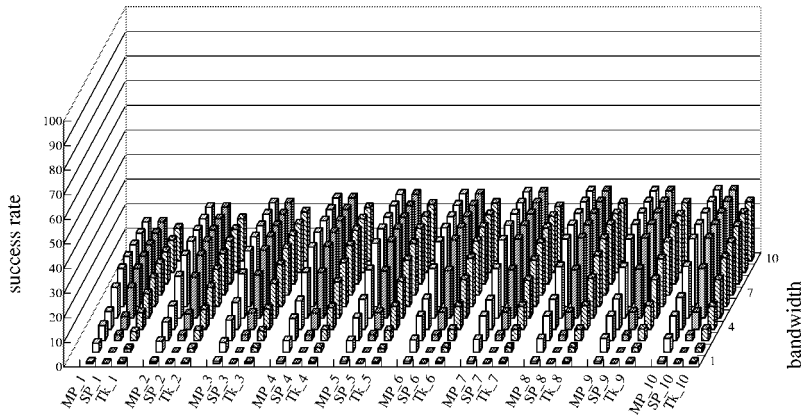

Figure 8. Success rate under different mean link bandwidth and number of tickets issued (network size = 50 and $\sigma = 1$). The number associated with each protocol is the number of tickets issued.

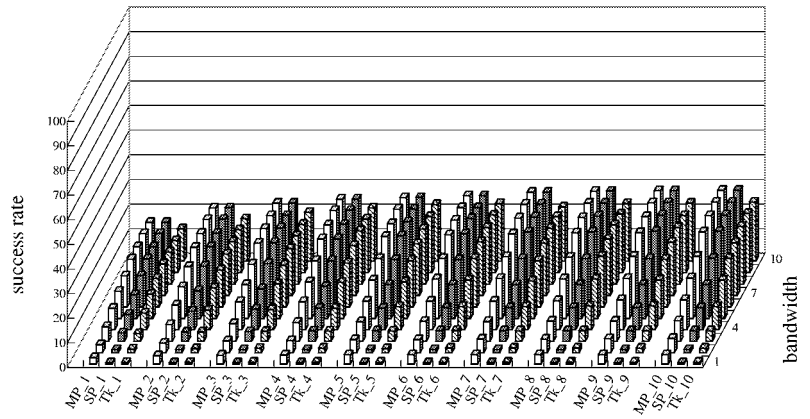

Figure 9. Success rate under different mean link bandwidth and number of tickets issued (network size = 50 and $\sigma = 2$).

comparing MP and Tk is similar. Thus, we conclude that our multi-path protocol is more useful for wireless networks, where bandwidths are typically more limited and precious than wireline networks.

B. Effect of network size. When comparing figures 8–11, we see that the gap between MP/SP and Tk will enlarge as the network scale is enlarged. The reason is two-fold. First, using multi-paths is more effective in larger-scale networks because there may exist more choices to distribute tickets. Second, since the gap between SP and Tk also enlarges, we believe that loop avoidance also gets more important in a larger-scale network.

C. Effect of link bandwidth variance. Figures 12 and 13 show the success rate of finding a satisfactory QoS route using Tk as a reference (i.e., the success rate is normalized to

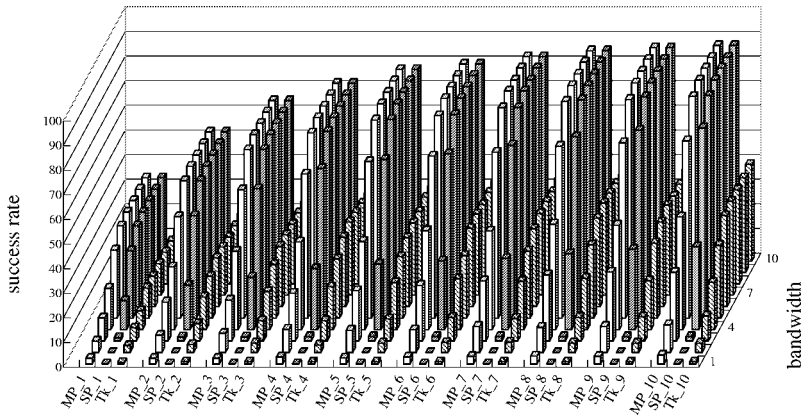

Figure 10. Success rate under different mean link bandwidth and number of tickets issued (network size = 100 and $\sigma = 1$).

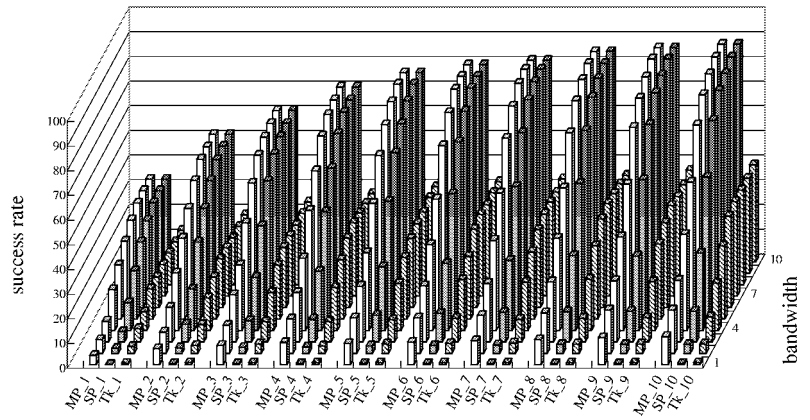

Figure 11. Success rate under different mean link bandwidth and number of tickets issued (network size = 100 and $\sigma = 2$).

that of the Tk protocol). These two figures have used link bandwidth variance of $\sigma = 1$ and 2, respectively. Interestingly, we see that a larger variance will favor our MP protocol over the SP and Tk protocols. This is more evident as the link bandwidth is small. This implies that using multi-paths is more effective when the available bandwidths in the MANET differ from place to place. Further, this effect is in fact magnified as the network scale is enlarged, as we can see from figures 14 and 15.

D. Effect of the number of tickets issued. Figures 12–15 compare the success rates when 1, 4, 7, and 10 tickets are issued. We see that when the mean link bandwidth is less than 4, using more tickets will slightly increase the success rate. However, when the mean link bandwidth is more than 4, the number of tickets used has little effect on the success rate.

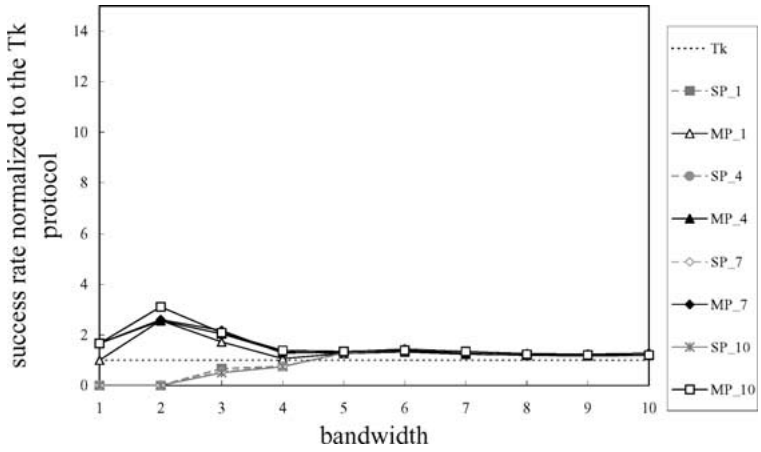

Figure 12. Success rate normalized to the Tk protocol (network size = 50 and $\sigma = 1$).

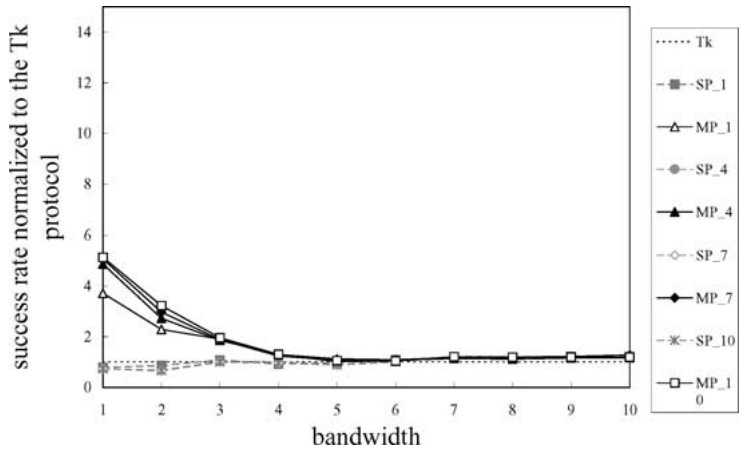

Figure 13. Success rate normalized to the Tk protocol (network size = 50 and $\sigma = 2$).

E. Route discovery overhead. Next, we compare the overhead of each protocol in terms of the number of routing-related packets issued. Each such packet being transmitted for one hop is counted for 1. The simulation results are in figures 16–19. The Tk protocol has a nearly constant overhead in each environment. The reason is that the protocol always forwards every ticket to the destination whether it succeeds or fails. The SP protocol has the lowest overhead when the link bandwidth is low. The reason is that we drop a probe packet once we find that it will fail. As the link bandwidth increases, the overhead will enlarge, up to a level roughly equal to that of the Tk protocol.

The overhead of our MP protocol is the largest when the link bandwidth is low. This makes sense because this represents the situation that a satisfactory QoS route is difficult to find, and thus searching for a multi-path with a higher cost is inevitable. (As we discussed earlier, our MP protocol also offers the highest success rate to discover a

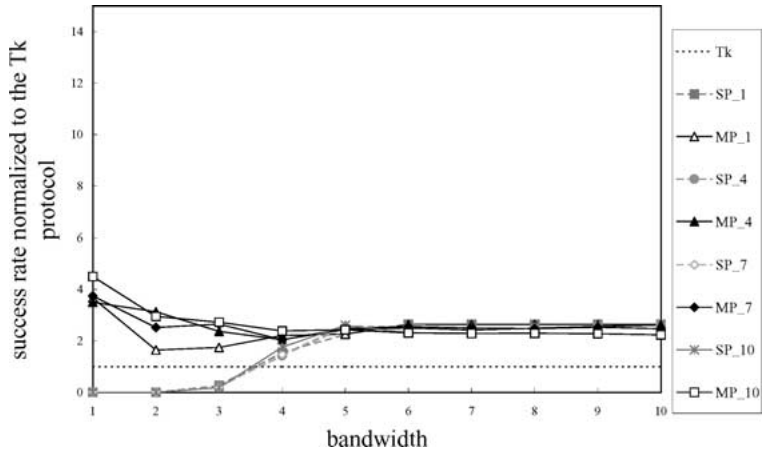

Figure 14. Success rate normalized to the Tk protocol (network size = 100 and $\sigma = 1$).

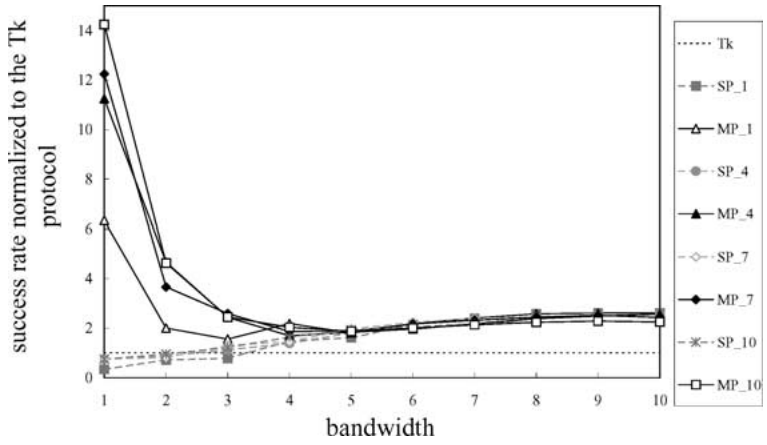

Figure 15. Success rate normalized to the Tk protocol (network size = 100 and $\sigma = 2$).

satisfactory QoS multi-path when the link bandwidth is scarce.) As the link bandwidth increases, the overhead of our MP protocol will converge to a level similar to those of the Tk and SP protocols. This shows the practical value of our MP protocol; it spends cost depending on how difficult it is to find a satisfactory QoS route.

Finally, we remark that the Tk protocol uses a distance-vector method to collect bandwidth information. But a large amount of extra overhead is not counted in figures 16–19. In our simulations, the DSDV protocol will issue about 1,800 and 13,500 packets for the protocol to enter a stable state from an initial state when the network size is 50 and 100 hosts, respectively. Every packet contains a very large table of bandwidth data. So the Tk protocol is only appropriate for a low-mobility MANET.

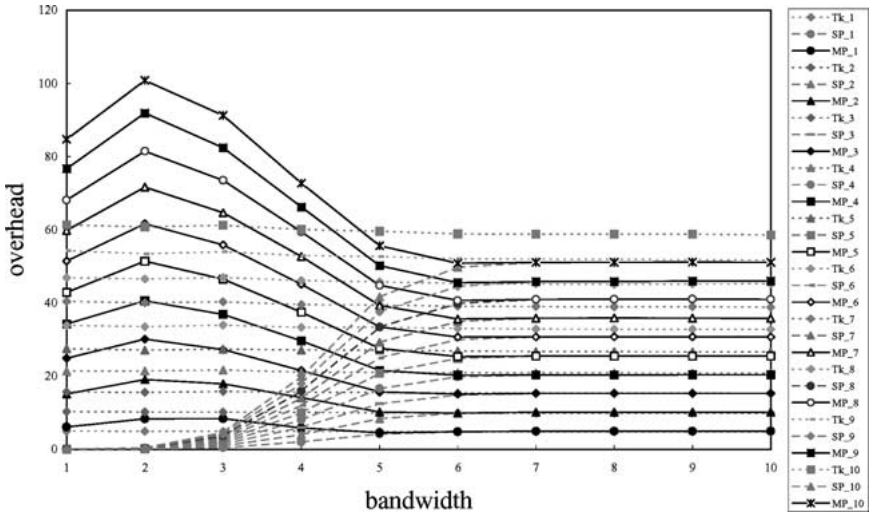

Figure 16. Routing overhead vs. mean link bandwidth (network size = 50 and $\sigma = 1$).

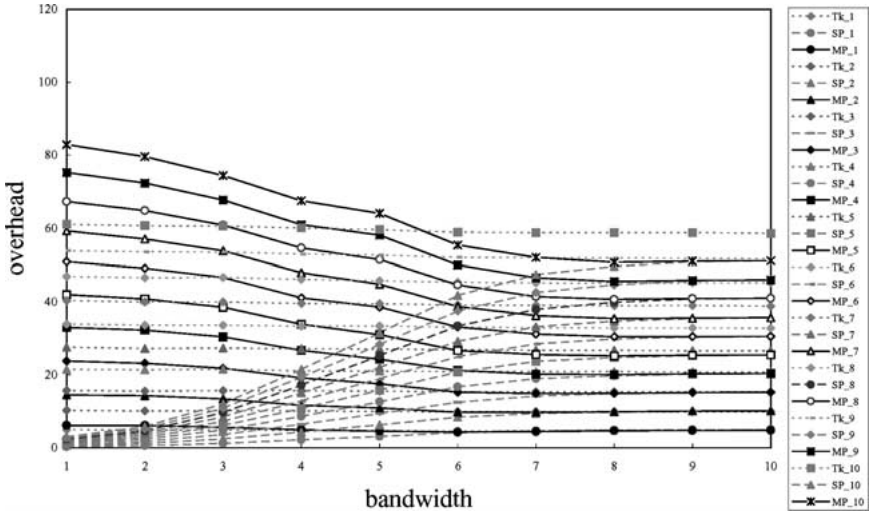

Figure 17. Routing overhead vs. mean link bandwidth (network size = 50 and $\sigma = 2$).

5. Conclusions

We have proposed a multi-path QoS routing protocol for finding a route with a bandwidth constraint in a MANET. As opposed to the proactive routing protocol [Chen and Nahrstedt, 4], our protocol is based on an on-demand manner to search for a QoS route, so no global link state information has to be collected in advance. Our protocol flexibly adapts 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. Simulation results have demonstrated the effectiveness of our protocol.

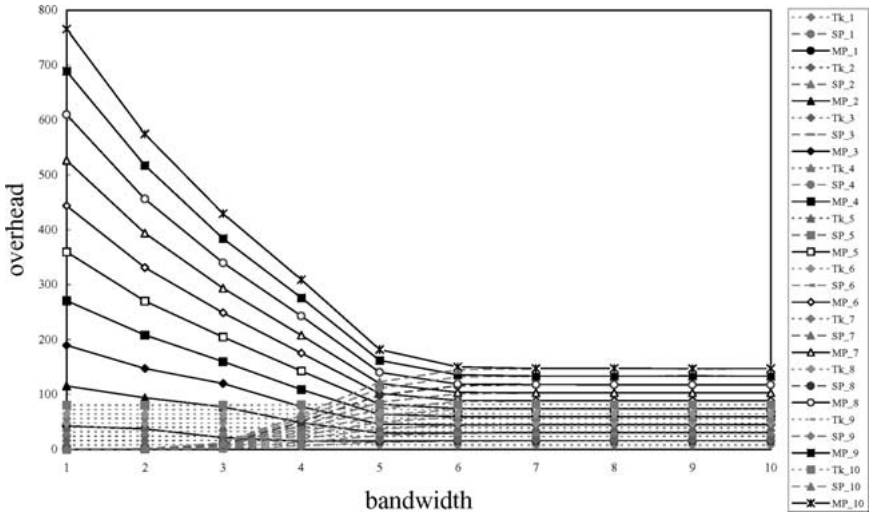

Figure 18. Routing overhead vs. mean link bandwidth (network size = 100 and $\sigma = 1$).

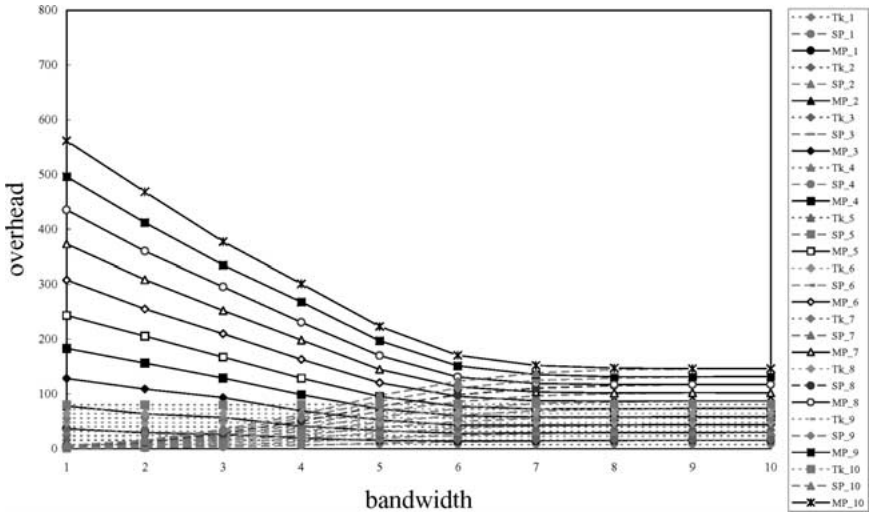

Figure 19. Routing overhead vs. mean link bandwidth (network size = 100 and $\sigma = 2$).

Acknowledgements

This research is supported in part by the Ministry of Education, ROC, under grant 90-H-FA07-1-4 (Learning Technology) and the National Science Council, ROC, under grants NSC89-2218-E-009-093, NSC89-2218-E-009-094, and NSC89-2218-E-009-095.

References

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

[2] J. Broch, D.B. Johnson and D.A. Maltz, The dynamic source routing protocol for mobile ad hoc networks, Internet draft (1998).

[3] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu and J. Jetcheva, A performance comparison of multihop wireless ad hoc network routing protocols, in: *Proc. of ACM/IEEE MOBICOM '98*, 1998, pp. 85–97.

[4] S. Chen and K. Nahrstedt, Distributed quality-of-service routing in ad hoc networks, IEEE Journal on Selected Areas in Communications 17(8) (1999) 1488–1505.

[5] S. Chen and K. Nahrstedt, An overview of quality of service routing for next-generation high-speed networks: Problems and solutions, IEEE Network (November/December 1999) 64–79.

[6] C.-C. Chiang, Routing in clustered multihop, mobile wireless networks with fading channel, in: *Proc. of IEEE SICON '97*, 1997, pp. 197–211.

[7] G.H. Forman and J. Zahorjan, The challenges of mobile computing, IEEE Computer 27(4) (1994) 38–47.

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

[9] M. Gerla and J.T.-C. Tsai, Mulicluster, mobile, multimedia radio network, Journal of Wireless Networks 1(3) (1995) 255–265.

[10] Z.J. Haas and M.R. Pearlman, The zone routing protocol (ZRP) for ad-hoc networks, Internet draft (1998).

[11] M. Jiang, J. Li and Y.C. Tay, Cluster-based routing protocol (CBRT), Internet draft (1998).

[12] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: *Mobile Computing*, eds. T. Imielinski and H. Korth (Kluwer Academic, Dordrecht, 1996) pp. 35–51.

[13] Y.-B. Ko and N.H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: *Proc. of ACM/IEEE MOBICOM '98*, 1998, pp. 66–75.

[14] W.-H. Liao, Y.-C. Tseng and J.-P. Sheu, GRID: A fully location-aware routing protocol for mobile ad hoc networks, Telecommunication Systems 18(1) (2001) 37–60.

[15] C.R. Lin and J.-S. Liu, QoS routing in ad hoc wireless networks, IEEE Journal on Selected Areas in Communications 17(8) (1999) 1426–1438.

[16] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in: *Proc. of ACM/IEEE MOBICOM '99*, 1999, pp. 151–162.

[17] C. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance-vector (DSDV) routing for mobile computers, in: *Proc. of ACM SIGCOMM Symposium on Communications, Architectures and Protocols*, 1994, pp. 234–244.

[18] C. Perkins and E.M. Royer, Ad hoc on demand distance vector (AODV) routing, Internet draft (1998).

[19] E.M. Royer and C.-K. Toh, A review of current routing protocols for ad hoc mobile wireless networks, IEEE Personal Communications (April 1999) 46–55.

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

[21] Y.-C. Tseng, S.-L. Wu, W.-H. Liao and C.-M. Chao, Location awareness in ad hoc wireless mobile networks, IEEE Computer 34(6) (2001) 46–52.