# QoS Routing in Ad Hoc Wireless Networks

Chunhung Richard Lin and Jain-Shing Liu

***Abstract*— The emergence of nomadic applications have recently generated much interest in wireless network infrastructures that support real-time communications. In this paper, we propose a bandwidth routing protocol for quality-of-service (QoS) support in a multihop mobile network. The QoS routing feature is important for a mobile network to interconnect wired networks with QoS support (e.g., ATM, Internet, etc.). The QoS routing protocol can also work in a stand-alone multihop mobile network for real-time applications. Our QoS routing protocol contains end-to-end bandwidth calculation and bandwidth allocation. Under such a routing protocol, the source (or the ATM gateway) is informed of the bandwidth and QoS available to any destination in the mobile network. This knowledge enables the establishment of QoS connections within the mobile network and the efficient support of real-time applications. In addition, it enables more efficient call admission control. In the case of ATM interconnection, the bandwidth information can be used to carry out intelligent handoff between ATM gateways and/or to extend the ATM virtual circuit (VC) service to the mobile network with possible renegotiation of QoS parameters at the gateway. We examine the system performance in various QoS traffic flows and mobility environments via simulation. Simulation results suggest distinct performance advantages of our protocol that calculates the bandwidth information. It is particularly useful in call admission control. Furthermore, "standby" routing enhances the performance in the mobile environment. Simulation experiments show this improvement.**

***Index Terms*—Ad hoc wireless networks, code division multiple access (CDMA), quality-of-service (QoS), personal communication networks, routing, TDMA, vitural circuit.**

## I. Introduction

PERSONAL communications and mobile computing require a wireless network infrastructure that is fast deployable, possibly multihop, and capable of multimedia service support [3]. The wireless network is often connected to a wired network (e.g., ATM or Internet) so that the ATM or Internet multimedia connection can be extended to the mobile users. There are several contributions that have already appeared in the wireless extensions of the wired ATM networks [1], [20]. Most of them focus on the cellular architecture for wireless personal communication networks (PCN's) supported by ATM backbone infrastructures. In this architecture, all mobile hosts in a communication cell can reach a base station in one hop. A time division multiple access (TDMA) scheme is generally used in the wireless extension for bandwidth reservation for the mobile host to base station connections.

In parallel with (and separately from) the single hop cellular model, another type of model, based on radio to radio multihopping, has been evolving to serve a growing number of applications that rely on a fast deployable, multihop wireless infrastructure [3]. The classic examples are battlefield communications (in the civilian sector), disaster recovery (fire, earthquake), and search and rescue. A recent addition to this set is the "ad hoc" personal communications network, that could be rapidly deployed on a campus, for example, to support collaborative computing and access to the Internet during special events (concerts, festivals, etc.). Multihopping through wireless repeaters strategically located on campus permits the reduction of battery power and the increase in network capacity (via spatial reuse).

The problem of interconnecting the multihop wireless network to the wired backbone requires a quality-of-service (QoS) guarantee not only over a single hop, but also over an entire wireless multihop path. The QoS parameters need to be propagated within the network, in order to extend the ATM virtual circuit (VC) into the wireless network and to carry out intelligent handoff between ATM gateways (base stations) by selecting the gateway that offers the best-hop distance/QoS tradeoff. The QoS-driven selection of the next gateway for handoff can be effectively combined with the soft handoff solutions (e.g., preestablishment of a VC to the next ATM gateway) already proposed for single hop wireless ATM networks [1], [20]. The key to the support of QoS reporting is QoS routing, which provides path QoS (bandwidth) information at each source. Prior research efforts in multihop mobile networks have not fully addressed this problem.

In this paper, we address the problem of supporting real-time communications in a multihop mobile network using QoS routing, and we propose a protocol for QoS routing. We consider different QoS traffic flows in the network to evaluate the performance of our protocol. The paper is organized as follows. Section II presents the routing protocol, derived from destination sequenced distance vector (DSDV) [19], which contains the features of bandwidth calculation and reservation. In Section III, there are some simulation experiments to be done to demonstrate the efficiency of the QoS routing. Section IV concludes the paper.

## II. Bandwidth Reservation

Multimedia applications such as digital audio and video have much more stringent QoS requirements than traditional

Manuscript received May 15, 1998; revised January 9, 1999. This work was supported in part by the National Science Council, Taiwan, under Contract NSC-87-2213-E-194-036.

C. R. Lin is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan R.O.C. (e-mail: chlin@acm.org).

J.-S. Liu is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan R.O.C. (e-mail: gis86807@cis.nctu.edu.tw).

Publisher Item Identifier S 0733-8716(99)04798-8.

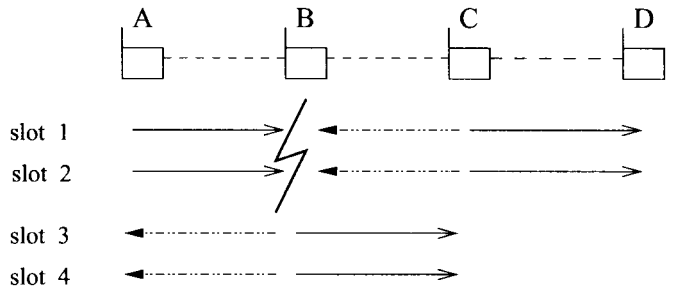



Fig. 1. Hidden terminal problem.

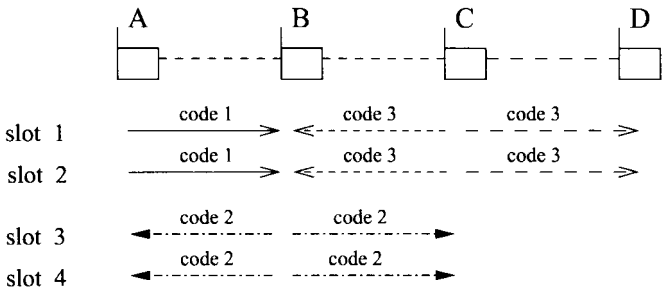



Fig. 2. CDMA over TDMA.

datagram applications. For a network to deliver QoS guarantees, it must reserve and control resources. A major challenge in multihop, multimedia networks is the ability to account for resources so that bandwidth reservations (in a deterministic or statistical sense) can be placed on them. We note that in cellular (single hop) networks, such accountability is made easily by the fact that all stations learn of each other's requirements, either directly or through a control station (e.g., the base station in cellular systems). However, this solution cannot be extended to the multihop environment. To support QoS for real-time applications, we need to know not only the minimal delay path to the destination, but also the available bandwidth on it. A VC should be accepted only if there is enough available bandwidth. Otherwise, it would disrupt the existing VC's.

We only consider "bandwidth" as the QoS (thus omitting signal-to-interference ratio (SIR), packet loss rate, etc.). This is because bandwidth guarantee is one of the most critical requirements for real-time applications. "Bandwidth" in time-slotted network systems is measured in terms of the amount of "free" slots. The goal of the QoS routing algorithm is to find a shortest path such that the available bandwidth on the path is above the minimal requirement. To compute the "bandwidth"-constrained shortest path, we not only have to know the available bandwidth on each link along the path, but we also have to determine the scheduling of free slots. Though some algorithms were proposed to solve this QoS routing problem, they unfortunately may only work in some special environments [2], [13], [18].

### *A. Code Division Multiple Access (CDMA) over TDMA*

Consider the example of the wireless network shown in Fig. 1 that uses TDMA for data transmission. The mobile host $A$ intends to transfer data to the mobile host $D$. All slots in the TDMA frame are assumed to be free. Suppose that $A$ reserves slots 1 and 2 for transmitting data to $B$, and $B$ uses slots 3 and 4 to forward packets to $C$. Because $A$ and $C$ are hidden from each other, $C$ may want to send packets to $D$ by using slots 1 and 2. Thus, there will exist a collision at $B$ (because $B$ may receive packets from $A$ and $C$ simultaneously) [8].

CDMA can be used to solve this problem (all spreading codes are assumed to be orthogonal to each other). For example, consider the same topology in Fig. 1. Fig. 2 shows that when $B$ uses slots 3, 4 to transmit packets to $C$, we can assign a code (say code 2) to node $B$ which is different from the code (say code 1) used by $A$. That is, we use a transmitter-based code assignment [14] to assign a code to each transmitter for data transmission. A spreading code can be reused if two nodes have a hop distance greater than two [8]. In Fig. 2, $C$ can use the same slots (1 and 2) as $A$ to send packets to $D$ encoded by a different code (say code 3) without any collision at $B$. It is notable that this case is assumed to be only one session through $A$, $B$, $C$, and $D$. If $A$ and $C$ (different sessions) intend to send packets to $B$ in the same slot, then only one packet can be received. and another will be lost depending on which code $B$ locks on. In this paper, we assume TDMA within our network. CDMA can also be overlaid on top of the TDMA infrastructure; namely, multiple sessions can share the same TDMA slot via CDMA, as shown in Fig. 2. In this case, the near–far problem and related power control algorithm become critical to the efficiency of the channel access [6]. A code assignment scheme [14] is assumed to be running in the lower layer of our system. Thus, the hidden terminal problem shown in Fig. 1 can be avoided.

In the network, each real-time connection will be assigned a VC. The VC is an end-to-end path along which slots have been reserved. As we shall see later, the path and slots of a VC may change dynamically during the lifetime of a connection due to mobility. Each node schedules each of its slots to transmit either datagram or VC traffic. Since real-time traffic (which is carried on a VC) needs guaranteed bandwidth during its active period, each node has to reserve its own slots to the VC at connection setup time.

We assume that a radio station can only receive a single transmission at a time and cannot transmit and receive simultaneously. The channel in our system is assumed to be time slotted. All nodes keep accurate common time (there exits a global clock or time synchronization mechanism). Each slot includes the number of redundant bits (e.g., for error control coding, retransmission) that must be sent when the channel has a low signal-to-noise ratio in order to get a successful transmission. We consider the assumptions found in most other radio data link protocols [5], [7]–[8], [10], [12], [15]. That is, the physical layer can provide the service of the slotted channel. Only one data packet can be transmitted in each data slot.

The information concerning available bandwidth between two nodes is critical. It is used to select a route that satisfies the QoS requirement. In addition, it is also used to determine whether a new connection is allowed into the network. As we mentioned, the topology is changed dynamically. Although the end-to-end bandwidth converges to the correct answer, it may do so slowly. Therefore, the "current bandwidth" kept in a

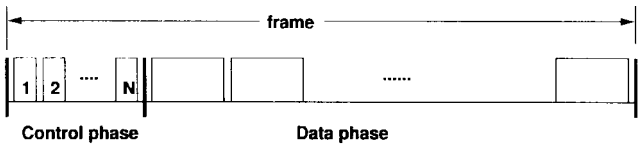



Fig. 3. Frame structure.

node may not react to the "real bandwidth" because of hop-by-hop propagation delay. If the current bandwidth is less than the real bandwidth (i.e., "underestimated"), we may miss some feasible paths, and the system utilization is low. Otherwise (i.e., "overestimated"), the call setup along a selected path may fail at some link because of the lack of enough bandwidth. The system then has to backtrack to free all reserved slots and to choose another route. Observe that this backtracking will degrade the system performance. If we have the bandwidth information at the beginning of a call setup, we can avoid the failure of the call setup at some intermediate node on a given route. Routing with a QoS indication is thus needed in order to efficiently manage bandwidth resources.

### *B. Bandwidth Calculation*

The transmission time scale is organized in frames, each containing a fixed number of time slots. The entire network is synchronized on a frame and slot basis. The frame/slot synchronization mechanism is not described here, but can be implemented with techniques similar to those employed in the wired networks (e.g., "follow the slowest clock" [17]) and properly modified to operate in a wireless mobile environment. Propagation delays will cause imprecision in slot synchronization. However, slot guard times (fractions of a microsecond) will amply absorb propagation delay effects (in the order of microseconds). Each frame is divided into two phases, namely, the control phase and the data phase, as shown in Fig. 3. The size of each slot in the control phase is much smaller than the one in the data phase. The control phase is used to perform all the control functions, such as slot and frame synchronization, power measurement, code assignment, VC setup, slots request, routing table (loop-free DSDV routing algorithm [19] in this paper), etc. The amount of data slots/frame assigned to a VC is determined according to a QoS requirement.

As depicted in Fig. 3, the control phase uses pure TDMA with full power transmission in a common code. That is, each node takes turns to broadcast its information to all of its neighbors in a predefined slot, such that the network control functions can be performed distributively. We assume the information can be heard by all of its adjacent nodes. In a noisy environment, where the information may not always be heard perfectly at the adjacent nodes, an acknowledgment scheme is performed in which each node has to ACK for the last information in its control slot. By exploiting this approach, there may be one frame delay for the data transmission after issuing the data slot reservation.

Ideally, at the end of the control phase, each node has learned the channel reservation status of the data phase. This information will help one to schedule free slots, verify the failure of reserved slots, and drop expired real-time packets.

The data phase must support both VC and datagram traffic. Since real-time traffic (which is carried on a VC) needs guaranteed bandwidth during its active period, bandwidth must be preallocated to the VC in the data phase before actual data transmission. That is, some slots in the data subframe are reserved for VC's at call setup time.

Because only adjacent nodes can hear the reservation information and the network is multihop, the free slots recorded at every node may be different. We define the set of the common free slots between two adjacent nodes to be the *link bandwidth*. Consider the example shown in Fig. 4 in which $C$ intends to compute the bandwidth to $A$. We assume the next hop is $B$. If $B$ can compute the available bandwidth to $A$, then $C$ can use this information and the "link bandwidth" to $B$ to compute the bandwidth to $A$.

We define the *path bandwidth* (which can be called *end-to-end bandwidth*) between two nodes, that are not necessarily adjacent, to be the set of available slots between them. If two nodes are adjacent, the path bandwidth is the link bandwidth. Consider the example in Fig. 4, and assume that one hop distance is between $A$ and $B$. If $C$ has free slots {1, 3, 4}, and $B$ has free slots {1, 2, 3}. Then the *link bandwidth* between $C$ and $B$ is {1, 3}. This means that we can only exploit slots 1 and 3 for packet transmission from $C$ to $B$. Thus, if a VC session needs more than two slots in a time frame, then it will be rejected to pass through (C, B). We can observe that $link\_BW(A,B) = free\_slot(A) \cap free\_slot(B)$. The definition of $free\_slot(X)$ is the slots which are not used by any adjacent host of $X$ to receive or to send packets from the point of view at node $X$. Next, we can further employ link bandwidth to compute end-to-end bandwidth. End-to-end bandwidth can provide us with an indication of whether there exits a QoS route between a given source–destination pair. We will use the following four cases as examples to show how to calculate the path bandwidth.

*Case 1:* Assume the link bandwidth of both $(A,B)$ and $(B,C)$ are the same, say {1, 2, 3, 4}, as in Fig. 5. If $C$ uses slots 1, 2 to send packets to $B$, then $B$ can only use slots 3, 4 to forward packets to $A$. This is because $B$ cannot be in transmitting mode and listening mode simultaneously. So the path bandwidth from $C$ to $A$, denoted as path_BW(C, A), can be {1, 2}, and its size is two. In this case, four free slots can only contribute two slots for path bandwidth. Namely, $\lfloor 4/2 \rfloor = 2$. Similarly, if there are only three free slots on both links, then the size of path bandwidth is $\lfloor 3/2 \rfloor = 1$.

*Case 2:* Assume link_BW(A, B) $= \{2,3\}$ and link_BW(B, C) $= \{1,2,3,4\}$, as in Fig. 6. Namely, link_BW(A, B) $\subset$ link_BW(B, C). If $C$ uses slot 2, then $B$ cannot use slot 2 any more. So in this case, $C$ should first use slots in link_BW(B, C) $-$ link_BW(A, B) $= \{1,4\}$ to maximize system utilization. Therefore, if $C$ uses slots 1, 4, then $B$ can use slots 2, 3. So path_BW(C, A) $= \{1,4\}$, and its size is two. Similarly, we can use the same way to process the case of link_BW(A, B) $\subset$ link_BW(B, C). In this case, $B$ must use slots in "link_BW(A, B) $-$ link_BW(B, C)" first.

*Case 3:* If link_BW(A, B) $\cap$ link_BW(B, C) $= \emptyset$, no conflict will occur. Fig. 7 shows this example. $C$ can choose

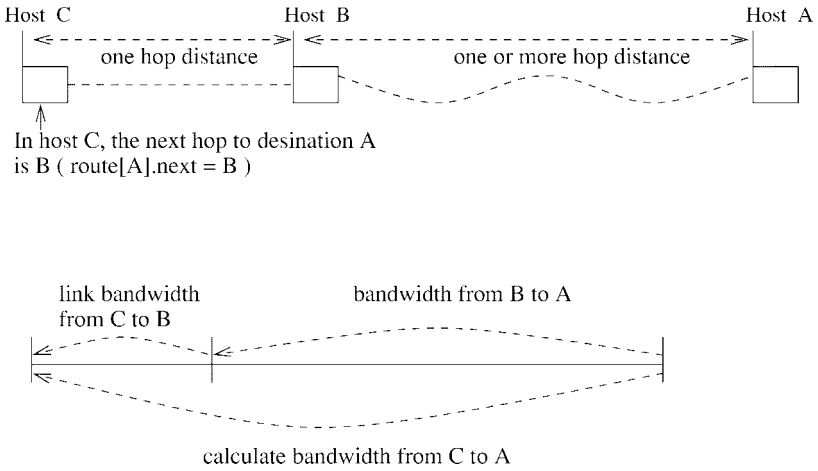



Fig. 4. End-to-end bandwidth calculation.

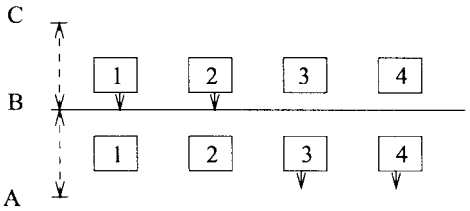



Fig. 5. Equal case.

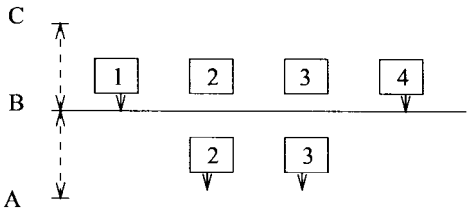



Fig. 6. Containing case.

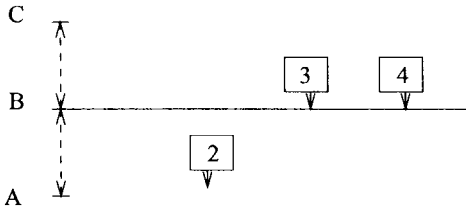



Fig. 7. Exclusive case.

either slots 3 or 4, and $B$ chooses slot 2. So path_BW(C, A) $= \{3\}$, and its size is one.

*Case 4:* This is a general case, as shown in Fig. 8. We will find any general case can be regarded as a combination of the previous three cases. Follow the slot assignment policy in Case 2. We assign slot 9 to $C$ and slot 1 to $B$ first. Fig. 9 shows the slots left. Next, we assign slot 10 to $C$ and slot 4 to $B$. Fig. 10 shows only slots {5, 6, 7, 8} left (slot 4 cannot be used by $C$ any more since $B$ is in transmitting mode). At present, this is the same situation as in Case 1. So $C$ can be assigned slots 5, 6, and $B$ is assigned slots 7, 8, as shown in Fig. 11. Here we let the path_BW(C, A) $= \{5, 6, 9, 10\}$. That is, $C$ can use slots {5, 6, 9, 10} to send packets to $B$, and

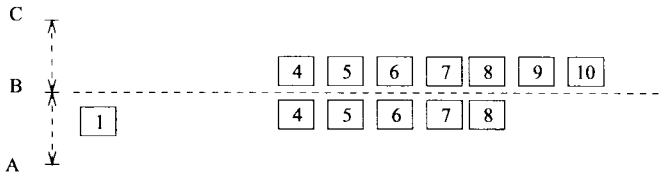



Fig. 8. General case.

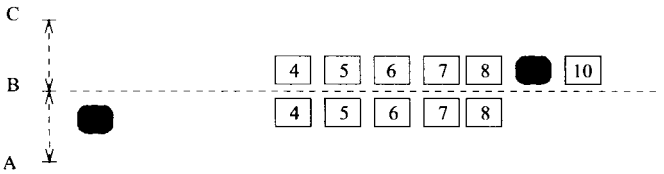



Fig. 9. Step 1 bandwidth calculation in Host C.

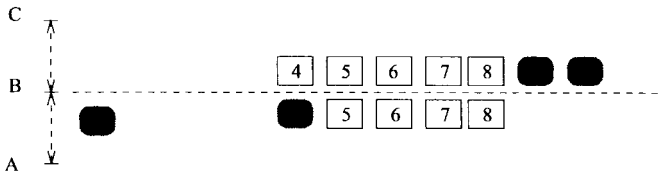



Fig. 10. Step 2 bandwidth calculation in Host C.

then $B$ uses {1, 4, 7, 8} to forward packets to $A$. The size of path bandwidth from $C$ to $A$ is four.

The last case is a general case. Observe that it is, in fact, a combination of Cases 1–3. Our slot assignment policy is to consider the slots not in link_BW(B,C) $\cap$ link_BW(A,B) first, until one of the special cases (i.e., Cases 1–3) occurs. The detail of the bandwidth calculation algorithm is shown in Fig. 12. To further generalize Case 4, if the distance between $A$ and $B$ is no longer one hop, as shown in Fig. 4, the same algorithm can still work. $C$ can calculate path_BW(C, A) by using link_BW(C, B) and path_BW(B, A) by using the algorithm in Fig. 12.

In general, to compute the available bandwidth for a path in a time-slotted network, one not only needs to know the available bandwidth on the links along the path, but also needs to determine the scheduling of the free slots. Resolving slot scheduling at the same time as available bandwidth is searched

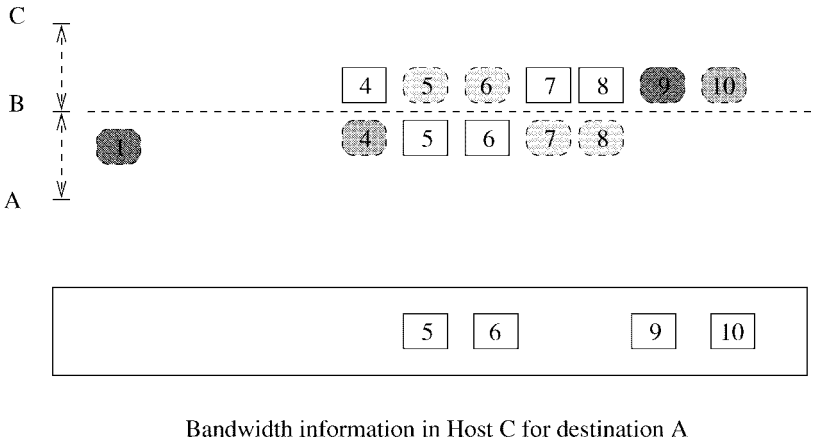



Fig. 11. Final result of bandwidth in Host C.

**Path Bandwidth Calculation Algorithm:**

when receive routing table from neighbor:
(bandwidth information is embedded in the routing table)

```
for all host i do
{
 if (i==sender)
 {
   link_BW = free_slots & free_slots of sender;
   route[sender].bandwidth = link_BW;
 }
 else if (route[i].next == sender)
 {
   common_BW = link_BW & route[i].bandwidth of sender;
   common_BW_size  = size(common_BW);
   difference_BW1  = size(link_BW ^ common_BW); /* the symbol ^ means XOR */
   difference_BW2  = size(route[i].bandwidth of sender ^ common_BW);

   if (difference_BW1 <= difference_BW2)
   {
     route[i].bandwidth_size = difference_BW1;
     remain_BW_size  = difference_BW2-difference_BW1;
   }
   else  /* difference_BW1 > difference_BW2 */
   {
     route[i].bandwidth_size = difference_BW2;
     remain_BW_size = difference_BW1-difference_BW2;
   }

   if (remain_BW_size > 0  &&  common_BW_size > 0)
   {
     if (common_BW_size <= remain_BW_size)
       route[i].bandwidth_size = route[i].bandwidth_size + common_BW_size;
     else  /* comm_BW_size> remain_BW_size */
     {
       rout[i].bandwidth_size = route[i].bandwidth_size + remain_BW_size;
       common_BW_size = floor((common_BW_size - remain_BW_size)/2);

       if (common_BW_size > 0)
          route[i].bandwidth_size=route[i].bandwidth_size + common_BW_size;
     }
   }
 }
}
```

Fig. 12. Path bandwidth calculation algorithm.

on the entire path is equivalent to solving the satisfiability problem (SAT), which is known to be NP-complete [9]. We use the heuristic approach to assign slots, as discussed in Case 4. Compared with [13], our algorithm in Fig. 12 is correct, simple, and efficient. In this algorithm, we only compute the size of the path bandwidth. Observe that the information of the end-to-end bandwidth is useful when a new VC session comes in the system. The system can immediately determine whether the VC traffic flow can be accepted at the beginning of the connection request according to the bandwidth requirement and available path bandwidth. Actually, this QoS indication enables more efficient call admission control.

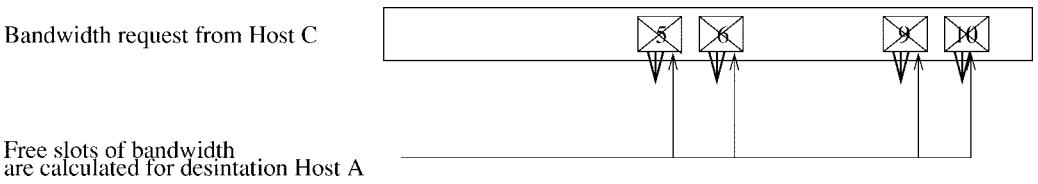



Fig. 13. Bandwidth information in Host C.

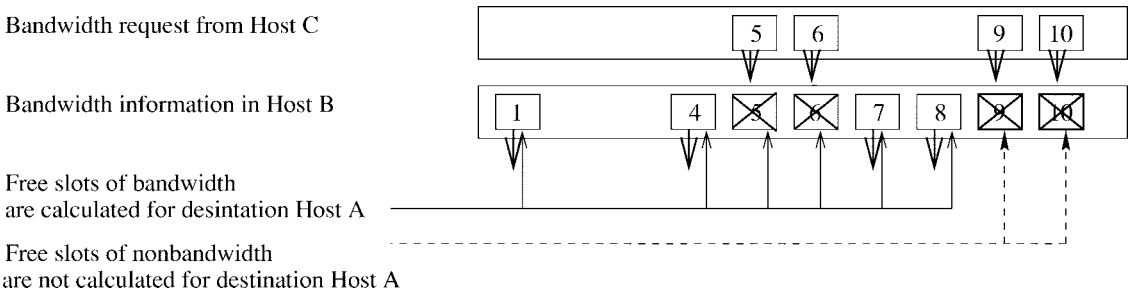



Fig. 14. Bandwidth information in intermediate Host B.

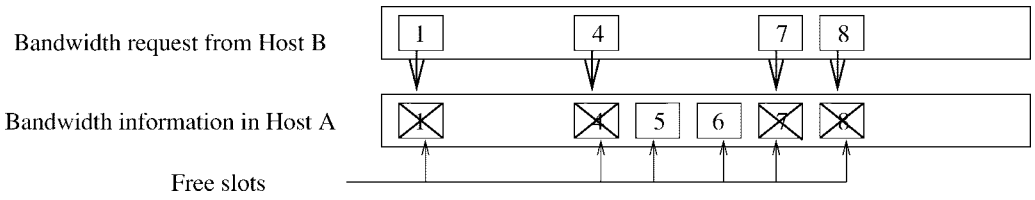



Fig. 15. Bandwidth information in destination Host A.

### C. Slot Assignment

We have already described how to calculate path bandwidth. In this section, we will discuss how to do the slot assignment. Once a call is accepted, the system needs to allocate slots to the VC hop-by-hop along the path to the destination. Each host must run a slot assignment algorithm. The algorithm in Fig. 12 only calculates the size of path bandwidth by periodically exchanging the path bandwidth information. However, it does not tell us how to assign the available slots efficiently during the call setup in each host. We use Fig. 8 as an example to describe how the slot assignment algorithm works. For a given path, the source node, immediate nodes, and the destination node will do different work.

*Source Node:* According to link_BW(C, B) and path_BW(B, A), we can compute the path bandwidth $path\_BW(C, A) = \{5, 6, 9, 10\}$ as we mentioned in the previous section (Fig. 11). Thus, the caller $C$ can reserve any of these slots. Assume the new VC session from $C$ to $A$ needs four data slots in each time frame (i.e., the QoS requirement). Thus {5, 6, 9, 10} are reserved, and we must remove these slots from the path bandwidth in the other destination entries in the routing table. For example, for the destination $E$, if $path\_BW(C, E)$ contains slots 5, 6, 10 and the size of the path bandwidth is five, then $C$ must remove slots 5, 6, 10 away, and the size becomes two $(5 - 3 = 2)$. Fig. 13 shows slots {5, 6, 9, 10} are reserved and then marked.

*Intermediate Nodes:* The $path\_BW(B, A) = \{1, 4, 5, 6, 7, 8\}$. When $B$ receives the reservation packet from $C$, it must check if its slots {5, 6, 9, 10} are free. If so, then it can receive data packets from $C$. Notice that $B$ must remove these slots from the path bandwidth in each destination entry in the routing table. In addition, $B$ must check if there are free slots which can be used for forwarding packets to next hop (i.e., $A$). In this case, slots {1, 4, 7, 8} are available. Thus, $B$ reserves them (shown in Fig. 14). In the meantime, $B$ must delete these slots from the path bandwidth field in the routing table. If any one slot in {5, 6, 9, 10} is busy or if there are no enough free slots (four slots in this case) to forward the packets, this reservation will fail. At this time, $B$ must reject the reservation request. Because $C$ has already reserved slots {5, 6, 9, 10}, $B$ needs to send a control packet, say *RESET*, to $C$ to ask to free these slots. These checking operations have to be done because the topological change may affect the free slots.

*Destination Node:* When the destination $A$ receives the reservation packet from the previous hop (i.e., $B$), it only reserves slots {1, 4, 7, 8} for receiving data packets from $B$, as shown in Fig. 15. These slots must be deleted from the path bandwith field in the routing table. Like intermediate nodes, if $A$ finds that any one slot in {1, 4, 7, 8} is not free, it must send the RESET message back to free those reserved slots hop-by-hop.

In the previous example, we discuss how to reserve the bandwidth. When every host receives a reservation packet, it must check if the slots that the sender will use for transmitting packets are free and find if there are free slots that can be reserved for forwarding packets. If both are satisfied, the reservation can be passed to the next hop. Finally, the destination sends a *REPLY* backward to the source to acknowl-

**Part 1: Source node (issuing a call request)**

```
if (available bandwidth < request bandwidth)
  return;
else
{
  mark all needed free slots;
  remove these needed slots from every path bandwidth;
  send the reservation packet;
}
```

**Part 2: Intermediate nodes and destination node**

```
if (I am the destination)
{
  check if slots which are used to receive packets are free;
  if TRUE
  {
    remove these needed slots from every path bandwidth;
    send REPLY to the source;
  }
  else
    send RESET along the path back to the source to free all reserved bandwidth;
}
else /* I am the intermediate node */
{
  check if slots which are used to receive packets are free; /*incoming bandwidth check*/
  if TRUE
  {
    check if bandwidth to destination is enough; /*outgoing bandwidth check*/
    if TRUE
    {
      remove all needed slots for incoming and outgoing from the available slots;
      forward the reservation packet to the next hop;
    }
    else
      send RESET along the path back to the source to free all reserved bandwidth;
  }
  else
    send RESET along the path back to the source to free all reserved bandwidth;
}
```

Fig. 16. Reservation algorithm.

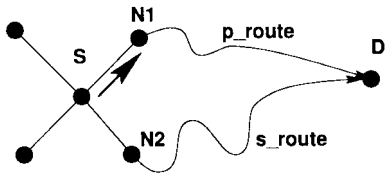



Fig. 17. Standby routing.

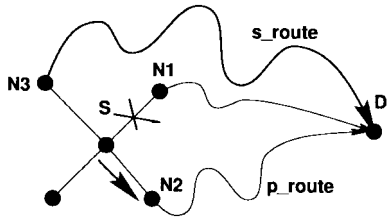



Fig. 18. The primary route fails, and the standby route becomes the primary route. Another standby route is constructed.

edge having set up the connection. The reservation algorithm is shown in Fig. 16. In the algorithm, when the call setup fails, it is necessary to free all reserved slots along the path backward to the source. This operation is very important since the bandwidth in the wireless network is quite limited. A topological change may make it impossible to send either a REPLY or a RESET back along the path that has been established so far. Thus, if an intermediate node does not receive a REPLY by a predefined time, the reserved slots will be freed automatically.

### *D. Rerouting When the Path Broken*

During the active period of a connection, a topological change may destroy the VC. The connection control must reroute or reestablish the VC over a new path. Therefore, it is important for a routing scheme to support alternative paths in a mobile environment. In such an environment, routing optimality is of secondary importance. The routing protocol must be capable of finding new routes quickly when a topological change destroys existing routes. To this end, we propose to maintain secondary paths that can be used immediately when the primary path fails [15], [16]. In Fig. 17, each node uses the primary route (i.e., the DSDV route) to route its packets. When the first link (s, N1) on the path fails, the secondary path (s, N2) becomes the primary path, and another standby path (s, N3) will be computed, as shown in Fig. 18. It is worth emphasizing that these routes must use different immediate successors to avoid failing simultaneously.

The secondary (standby) route is easily computed using the DSDV algorithm. Referring to Fig. 17, each neighbor of node $S$ periodically informs $S$ of its distance to destination $D$.

```
Routing Table Maintenance Algorithm

When receiving a routing table from a neighbor:
for all i /* i means ID */
{
 if (i != sender &&
   sender routing table[i].next != myself &&
   my routing table[i].next != sender &&
   my routing table[i].next != sender routing table[i].next)
 {
   if (sender != my routing table[i].next3)
   {
     calculate bandwidth to destination host i using the same algorithm in DSDV.

     if (calculate_BW > bandwidth from original next2)
     {
       new next2 = sender;
       update next2 bandwidth information;
     }
   }

   if (sender != my routing table[i].next &&
      sender != my routing table[i].next2)
   {
     calculate bandwidth to destination host i using the same algorithm in DSDV.

     if (calculate_BW > bandwidth from original next3)
     {
       new next3 = sender;
       update next3 bandwidth information;
     }
   }
 }
}
```

Fig. 19. The standby routing algorithm.

```
Construct a QoS path

if (bandwidth of next > required bandwidth)
{
  update bandwidth information of next;
  send this call;
}
else if (bandwidth of next2 > required bandwidth)
{
  copy next2 bandwidth to next bandwidth;
  copy next2 to be next;
  update next2 bandwidth information;
  send this call as using the original DSDV method;
}
else if (bandwidth of next3 > request bandwidth)
{
  copy next3 bandwidth to next bandwidth;
  copy next3 to be next;
  update next3 bandwidth information;
  send this call as using the original DSDV method;
}
```

Fig. 20. Construct a QoS path for a new call.

The neighbor with the shortest distance yields the primary route. The runnerup yields the secondary route. This scheme guarantees that the first link is different for the two paths. Furthermore, the standby route computation requires no extra table, message exchange, or computation overhead. Like the DSDV (or primary) route, we must compute bandwidth information from all neighbors to determine the standby routes. The algorithm in Fig. 19 maintains the routing table (two alternative routes in the algorithm, i.e., $\mathrm{next2}$ and $\mathrm{next3}$); $\mathrm{next2}$ has larger bandwidth than $\mathrm{next3}$. The "$\mathrm{next}$" in the algorithm means the primary route. It is notable that the primary route is shortest, but is not necessary to have the largest bandwidth.

When a host generates a new call, it uses the algorithm in Fig. 20 to construct the path. In the algorithm, we will choose the route that satisfies the QoS requirement in order of precedence $\mathrm{next}$, $\mathrm{next2}$, and $\mathrm{next3}$. The chosen route will be the primary route. That is, the $\mathrm{next}$ entry in the routing table may be changed depending on the QoS requirement. After choosing the primary route, the source node will send out a call setup message to $\mathrm{next}$. When receiving the message, the $\mathrm{next}$ node will run the protocol in Fig. 21 to reserve bandwidth for the new call. When a topological change destroys the primary route, as in Fig. 18, node $s$ will try to rebuild a new path immediately, using either $\mathrm{next2}$ or $\mathrm{next3}$ if the QoS is satisfied. Thus, a new route from the breakpoint will be established by sending call setup message hop-by-hop to the destination.

## III. System Performance

The environment that we consider consists of 20 mobile hosts roaming uniformly in a 1000 $\times$ 1000 ft$^2$ area. Each node moves randomly at uniform speed. Radio transmission range is 400 ft. That is, two nodes can hear each other if their distance is within the transmission range. Data rate is 4 Mbit/s. In our experiments, the channel quality may affect the packet transmission. That is, the noise in the channel may cause errors in packets. The channel quality specified by the bit error rate is uniform in all of the experiments. Because the VC traffic is delay sensitive rather than error sensitive, packets are therefore not ACK'ed. A coding scheme is assumed to be running in the system to do the forward error correction. In the experiments, we will pay more attention to the effect of mobility to the system performance.

In each time frame (Fig. 3), the data slot in the data phase is 5 ms, and the control slot in control phase is 0.1 ms. Channel overhead (e.g., code acquisition time, preamble, etc.) is factored into control/data packet length. We assume there are 16 data slots in data phase. So the frame length is $20 * 0.1 + 16 * 5 = 82$ ms. Since the number of data slots is less than the number of nodes, nodes need to compete for these data slots. The source–destination pair of a call is randomly chosen, and their distance must be greater than one. Once a call request is accepted on a link, a transmission window (i.e., data slots) is reserved (on that link) automatically for all the subsequent packets in the connection. The window is released when either the session is finished or the RESET packet is received (Fig. 16). Conceptually, this scheme is an extension of packet reservation multiple access (PRMA) [12] to the multihop environment.

There are three types of QoS for the offered traffic. $\mathrm{QoS}_1$, $\mathrm{QoS}_2$, and $\mathrm{QoS}_4$ need one, two, and four data slots in each frame, respectively. The total simulation time is $10^6$ ms. A new call is generated every cycle (82 ms). Each call duration is an exponential distribution with the mean value 180 s. The interarrival time of packets within a $\mathrm{QoS}_1$ session is an exponential distribution with 100 ms on average. Similarly, the mean values of the interarrival time for $\mathrm{QoS}_2$ and $\mathrm{QoS}_4$ are 50 and 25 ms, respectively. The maximal queueing delay of a data packet within a node is set to four frame lengths (328 ms). Namely, if a packet stays in a node more than that time, it will be dropped.

**Bandwidth reservation in the intermediate node**

```
if (sender using first next for path building)
  the same action as original DSDV;
else
{ /* sender initial this call by next2 or next3 */

 if (bandwidth calculated from next > request bandwidth)
 {
   reserve bandwidth using next;
   forward this call setup message to next;
 }
 else if (bandwidth calculated from next2 > request bandwidth)
 {
   copy next2 bandwidth to be next bandwidth;
   copy next2 to be next;
   reserve bandwidth;
   forward this call setup message as original DSDV method;
 }
 else if (bandwidth calculated from next3 > request bandwidth)
 {
   copy next3 bandwidth to be next bandwidth;
   copy next3 to be next;
   reserve bandwidth;
   forward this call setup message as original DSDV method;
 }
 else /* no bandwidth */
   backward "RESET" message to clean existing path bandwidth in other hosts reserved for this path;
```

Fig. 21. Bandwidth reservation in the intermediate nodes.

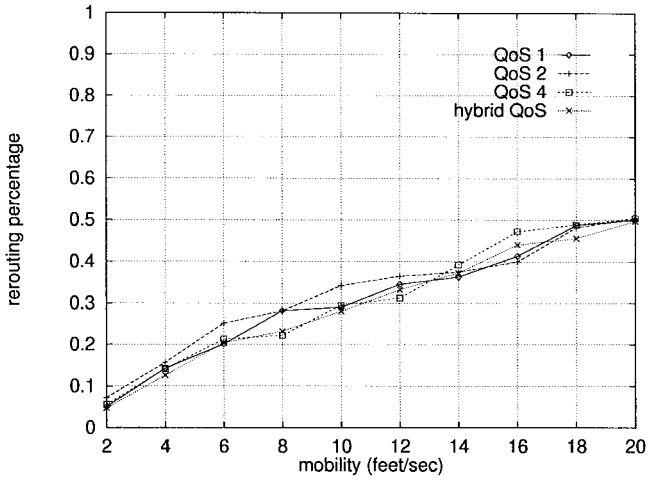



Fig. 22. The percentage of calls to be rerouted.

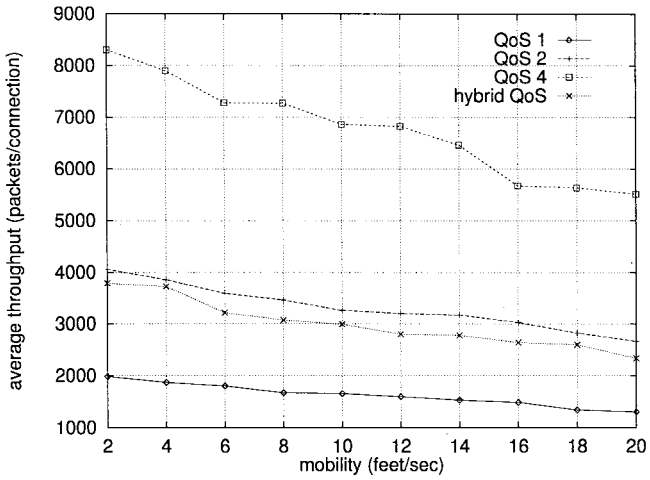



Fig. 23. Average throughput of different QoS's.

In the first experiment, we consider the effect of variable mobility on the rerouting due to a broken path. If any one of the links on the path is broken, the VC over the path needs to be rerouted. Fig. 22 presents the simulation result. The curve $\mathrm{QoS}_i$ means QoS is uniform for all traffic flows. Hybrid QoS means different QoS traffic flows in the system. At the call setup, each source–destination pair can randomly determine its QoS type with the uniform distribution that will not be changed during the active period. Observe that the percentage of calls that need to be rerouted during their active periods increases as the mobility is increasing. That is, high mobility causes paths to be broken frequently. When mobility is 20 ft/s, about 50% of the connections need to be rerouted. It is notable that the result is independent of the QoS of the traffic flows. This is because what we measure is the fraction of connections that have already received a QoS route and need to be rerouted during their active periods.

The second experiment is to find the average throughput. Recall that packets are not ACK'ed, and every packet is sent exactly once. Thus, there are no duplicates in our system. In Fig. 23, we can find that the throughput of each connection decreases as the mobility increases. High mobility makes frequent rerouting and thus results in more end-to-end transmission delay and more packet loss (over the upper bound of the queueing delay at each node). In addition, observe that the high QoS connection has high throughput on average because of the high input rate. In addition, slot reservation makes the input packet flow have lower queueing delay to avoid the packet loss. Note that the throughput of hybrid traffic is similar to $\mathrm{QoS}_2$ traffic.

The average hop delay is shown in Fig. 24. Since path length is not the same for all packets due to rerouting or different VC sessions, in this paper we show the average hop delay instead of end-to-end delay for those packets that can

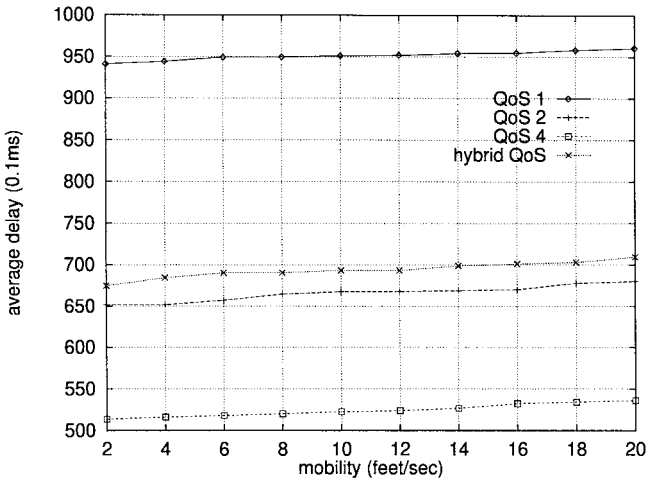



Fig. 24. Average hop delay.

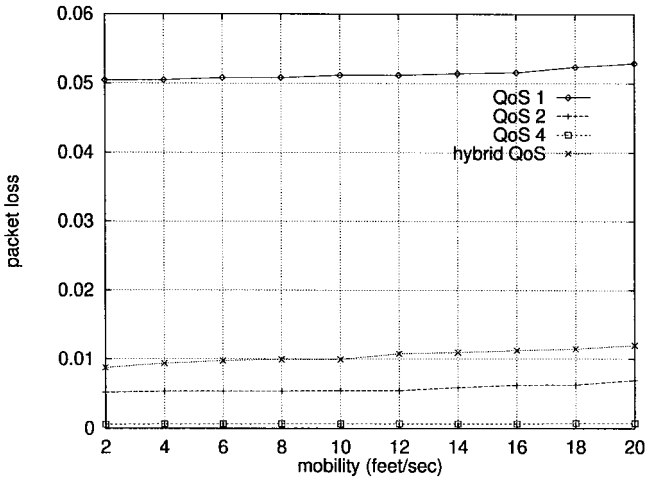



Fig. 26. Packet loss per host.

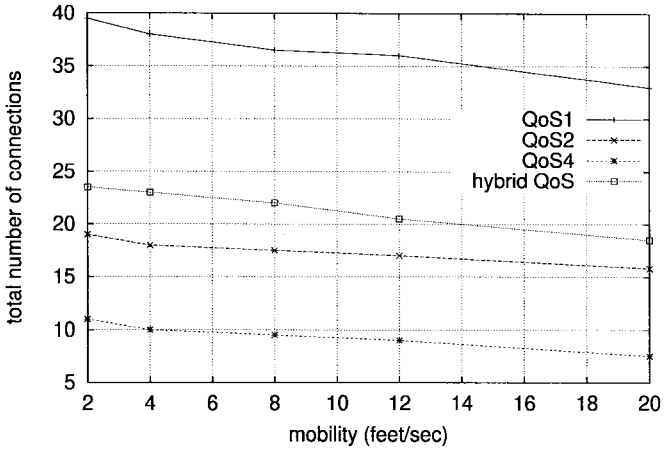



Fig. 25. The total number of connections.

reach the destinations (those packets may come from different VC sessions). The hop delay is computed from the end-to-end delay divided by the path length. Even if we know how to manage the acceptance of the VC's at call setup time using QoS routing, we can experience network congestion due to the dynamics of mobility and traffic patterns. Thus, in our simulation, we apply the concept of selective packet dropping that is successfully used in ATM. This operation of congestion control can reduce the end-to-end delay. In addition, hop-by-hop slot reservation can also limit the queueing delay within a host.[1] We can observe that the delay is stable. For example, the $QoS_1$-VC has a stable hop delay of about 95 ms, which is close to the frame length (82 ms). $QoS_2$ and $QoS_4$ also have stable delays of about 66 and 53 ms, respectively. When the QoS is stringent (i.e., more slots are allocated per frame), a packet has a higher probability to be transmitted sooner. So the delay is lower. Mobility only makes the delay increase slightly.

[1]There are two factors to limit the delay within a host. First, for a host, the input rate is always less than the processing rate. For example, the mean input rate of $QoS_1$ is 1 packet/100 ms, and the mean processing rate is 1 packet/82 ms. Second, we set an upper bound of the queueing delay for each packet within a host.

Fig. 25 presents the supportable amount of VC's of different QoS traffic. The current system bandwidth is 16 data slots in each time frame. There are about 33 connections of $QoS_1$ traffic simultaneously in the system at a mobility of 20 ft/s (16 connections for $QoS_2$ and seven connections for $QoS_4$). Mobility will decrease the supportable amount of connections.

Fig. 26 reports the packet loss for varying mobility. The maximal queueing delay of a packet within a node is limited to four frame lengths (328 ms). If a packet stays in a node more than 328 ms, it will be dropped. Packets are served in first-in/first-out (FIFO) order. Fig. 26 appears to show that at a mobility of 20 ft/s, the packet loss is about 5.3% (for $QoS_1$) or less (0.7% for $QoS_2$, 0.1% for $QoS_4$, and 1.2% for hybrid QoS). This loss rate is particularly low. The mobility slightly increases the packet loss rate. Consider the M/M/1 queue $T = 1/(\mu - \lambda)$ where $T$ is the system time, $\mu$ is the processing rate, and $\lambda$ is the arrival rate. If $\mu' = 2\mu$ and $\lambda' = 2\lambda$, then $T' = T/2$. Thus, the system time of $QoS_2$ is one half of the system time of $QoS_1$. This is the reason why $QoS_1$ has the highest packet loss rate among all traffic flows. For the traffic with high QoS, the packet loss rate is low because its system time is small. Therefore, the system time has a lower probability to be over 328 ms. Fig. 26 presents the same result as the queueing analysis. The queueing model can also explain why $QoS_4$ has the largest throughput (Fig. 23), and $QoS_1$ has more hop delay (Fig. 24).

The following set of experiments is to assess the improvement introduced by the "standby" routing feature, i.e., the availability of an alternate route in case the preferred route fails. This feature is of critical importance when stations are mobile. In the experiments, the total simulation time is $10^8$ ms. A new call is generated every two cycles ($2*82$ ms). If no data packet is sent over the reserved slots for ten cycles ($10*82$ ms), the reserved slots will be released. There are four experiments to be done. In the first one, we evaluate the successful probability of constructing a VC through each route by exploiting the path bandwidth information and slot allocation algorithm under the condition of mobility. Each node is considered to run the algorithm in Fig. 20 to set up a new call. Fig. 27 shows that at a mobility of 20 ft/s, for example, 98% of calls that use

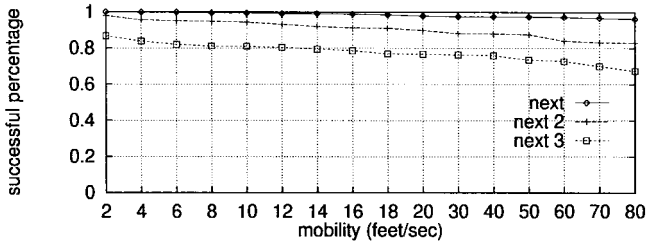



Fig. 27. The reliability of different routes for the QoS requirement.

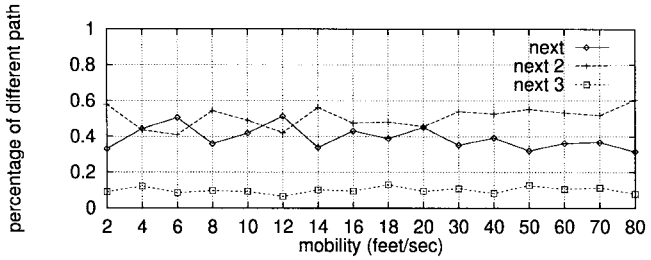



Fig. 28. Route selection at the source for a given VC.

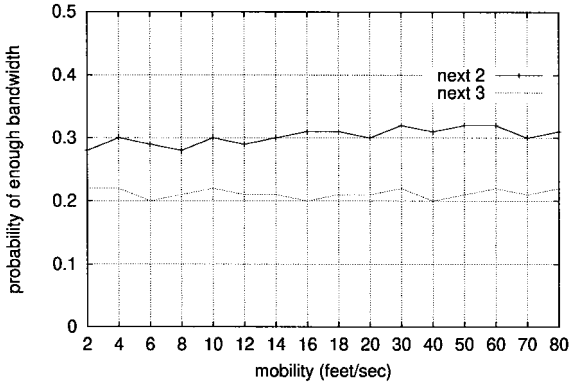



Fig. 29. Probability of the breakpoint having QoS alternative routes.

next (primary route) at the source node can set up the QoS-VC successfully, and 2% will fail because of outdated bandwidth information. Because of mobility (i.e., topological change), the path bandwidth information is changed dynamically. If a node does not receive the newest bandwidth information, then the QoS-VC setup may fail at some intermediate node because of lack of bandwidth. Fig. 27 presents the effect of the "possible" outdated bandwidth information on the primary route (next) and the standby routes (next2 and next3). We can observe that no matter which route is selected at the source, we still have high probability (for example, 98% for next, 90% for next2, and 77% for next3 at a mobility of 20 ft/s) to construct a QoS-VC successfully. That is, the effect of mobility on the route selection that establishes a VC is not too strong.

For a given VC of a call, it may be constructed by a different route at the source. According to our algorithm for constructing a QoS path (Fig. 20), in Fig. 28 near 30–50% of VC's are setup through the primary route (i.e., next) under different mobility. Similarly, 40–60% of VC's are through next2. From this result, we can find the standby route is particularly useful. The primary route is the shortest path calculated by the DSDV algorithm. However, if all source–destination pairs only consider the shortest path, there will be some hot spots that lack enough bandwidth. Once a call request is passing through those nodes, it will be rejected. Thus, this is the reason why there are only 30–50% of VC's that can pass through the primary route. However, because of the existence of standby routes, the VC traffic load can be evenly distributed among the network to avoid through these hot spots. Note that there is only about 10% of VC's using next3. This means that if next and next2 do not have enough bandwidth, there is a small probability for next3 to have enough bandwidth. Actually at this time, the system is saturated.

When a link of a VC is broken, the new VC can be constructed from the "breakpoint" (like node $s$ in Fig. 18) if there exists enough bandwidth in a standby route. In case of no bandwidth in any standby route, back pressure can be exerted to stop traffic flow from the upstream node and to backtrack to some intermediate node that has a QoS route to the destination. In the worst case, a new VC will be reconstructed from the source node. All reserved slots by the old VC will be released hop-by-hop. Fig. 29 shows the probability of finding a feasible alternate route at the breakpoint according to the current bandwidth information before the new call setup begins. At a mobility of 20 ft/s for example, there is a probability of 0.3 for next2 (0.21 for next3), which has enough bandwidth to the destination at the breakpoint. According to our standby routing protocol, whether there exists a feasible alternate route depends on the set of neighbors. However, the node speed (i.e., quick topological change) is not necessary to result in a greater chance of having "good" neighbors who have larger bandwidth. Therefore, the mobility does not affect the probability. Fig. 30 shows the probability of a successful call setup given a supposedly feasible route (i.e., either next2 or next3) at the breakpoint to the destination. The next2 path can have a probability of more than 0.9 to set up a new VC in low mobility. In high mobility, the probability is still more than 0.8. Observe that in high mobility, there is a lesser chance of a successful call setup. This is because when the system is saturated, the node speed (i.e., quick topological change) does not cause an intermediate node between the breakpoint to the destination to see another "good" neighbor who has enough bandwidth. Combine the results in Figs. 29 and 30. We can find there is little probability (about 0.2) of having another QoS route at the breakpoint. However, the backtracking increases the probability. That is, if we consider the set of nodes from the source to the breakpoint along the path, the probability of having a QoS route at any one of these nodes will be much higher than the case of just considering the breakpoint. We must note that there is no extra communication cost to maintain the standby routes.

In the last experiment, we intend to assess how useful the bandwidth information is that is obtained from the bandwidth calculation algorithm presented in Fig. 12. We can exploit this QoS indication to determine if a new call can be accepted or not. This information lets us foresee whether a QoS-VC can be established along a given route before the call setup

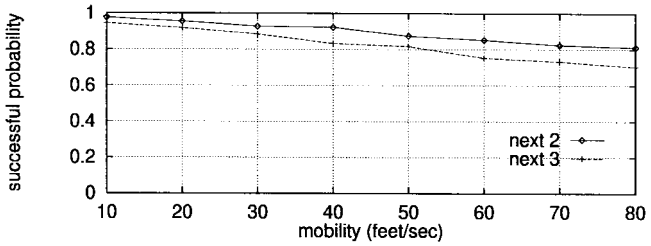



Fig. 30. The performance of the standby routing at the breakpoint.

| mobility | with bandwidth information | bandwidth info under-estimate | without bandwidth information |
|---|---|---|---|
| 2 feet / sec | 1.9 | 0.2 | 650.3 |
| 4 feet / sec | 2.1 | 0.3 | 667.5 |
| 8 feet / sec | 2.1 | 0.3 | 661.6 |
| 12 feet / sec | 2.6 | 0.3 | 656.7 |
| 20 feet / sec | 2.8 | 0.5 | 686.1 |

Fig. 31. The average number of blocked calls in "DSDV+ reservation algorithm."

begins. If we only use the DSDV algorithm and the reservation algorithm (Fig. 16), then a new call may be blocked in some intermediate node that is saturated. No source can construct a VC via the saturated node until one of the VC's over the node ends its transmission, and the bandwidth becomes available.

Mobile nodes exchange bandwidth information periodically. The data is propagated hop-by-hop and cannot reach all nodes immediately. In the following experiment, we compare the call blocking rate of two systems that are running the same routing algorithm (i.e., DSDV) and the reservation algorithm in Fig. 16. But only one of both has the bandwidth information in the routing table. In addition, we also consider the case in which bandwidth information shows that there is no bandwidth, but the new call still can be set up. That is, the bandwidth is "underestimated" (the current bandwidth is less than the real bandwidth). The simulation results in Fig. 31 report that the system with bandwidth information obviously has much better performance. These call blocking rates are for input traffic during $10^6$ ms of the simulated time. In this experiment, we run 100 simulations (each of length $10^6$ ms) with different initial topologies to compute the averages. The call generating rate is one call every two frames (i.e., $2 * 82 = 164$ ms). Thus, there are $(1/164) * 10^6 = 6098$ calls generated during $10^6$ ms. Observe that about 11% calls will be blocked if there is no bandwidth information. However, only from two to three calls of the 6098 calls will be blocked if the source node has the bandwidth information. This information lets the source node determine if a new call should be blocked. In addition, this information is seldom "underestimated" by our algorithm. That is, the "reliability" of the information is high. Thus, this knowledge enables more efficient call admission control.

## IV. Conclusions

In this paper, we propose a novel QoS routing protocol that contains bandwidth calculation and slot reservation for multihop mobile networks. It can be applied to two important scenarios: multimedia ad hoc wireless networks and multihop extension wireless ATM networks. Specially, the bandwidth information can be used to assist in performing the handoff of a mobile host between two ATM base stations. Furthermore, it enables more effective call admission control. In the case of ATM interconnection, ATM-VC service can be extended to the wireless networks with possible renegotiation of QoS parameters at the gateways (base stations). In the performance experiments, traffic flows with different QoS types are considered. Simulation results suggest distinct performance advantages of our protocol calculating the bandwidth information. Furthermore, "standby" routing enhances the performance in the mobile environment. Finally, the comparison of the call blocking rate of both systems, "DSDV+ reservation algorithm" with and without bandwidth calculation, illustrates the importance of bandwidth routing to the system with a QoS requirement.

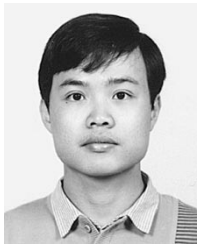
## Acknowledgment

The authors would like to thank M. Gerla at the Computer Science Department, University of California, Los Angeles, for his invaluable suggestions for this work.

## References

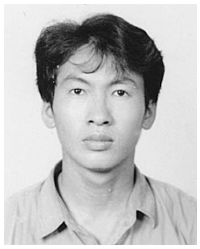
[1] A. Acampora, "Wireless ATM: A perspective on issues and prospects," *IEEE Personal Commun. Mag.*, pp. 8–17, Aug. 1996.
[2] I. F. Akyildiz, W. Yen, and B. Yener, "A new hierarchical routing protocol for dynamic multihop wireless networks," *Proc. IEEE INFOCOM'97*, pp. 280–287.
[3] A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Kleinrock, J. Short, and J. Villasenor, "Adaptive mobile multimedia networks," *IEEE Personal Commun. Mag.*, pp. 34–51, Apr. 1996.
[4] R. Bagrodia and W. Liao, "Maisie: A language for the design of efficient discrete-event simulations," *IEEE Trans. Software Eng.*, pp. 225–238, 1994.
[5] D. J. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. Commun.*, vol. COMM-29, pp. 1320–1332, Nov. 1981.
[6] N. Bambos and G. J. Pottie, "On power control in high capacity cellular radio networks," in *Proc. IEEE GLOBECOM'92*, pp. 863–867.
[7] I. Chlamtac and S. Kutten, "On broadcasting in radio networks problem analysis and protocol design," *IEEE Trans. Commun.*, vol. COMM-33, pp. 1145–1158, Dec. 1985.
[8] I. Chlamtac and S. S. Pinter, "Distributed nodes organization algorithm for channel access in a multihop dynamic radio network," *IEEE Trans. Comput.*, vol. C-36, no. 6, pp. 728–737, 1987.
[9] M. R. Garry and D. S. Johnson, *Computers and Untractability*. San Francisco, CA: Freeman, 1979.
[10] M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *ACM-Baltzer J. Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.
[11] M. Gerla, "Routing and flow control in ISDN's," in *Proc. ICCC'86*, pp. 643–647.
[12] D. Goodman and R. A. Valenzuela, K. T. Gayliard, and B. Ramamurthi, "Packet reservation multiple access for local wireless communications," *IEEE Trans. Commun.*, vol. 37, pp. 885–890, Aug. 1989.
[13] Y.-C. Hsu and T.-C. Tsai, "Bandwidth routing in multihop packet radio environment," in *Proc. 3rd Int. Mobile Computing Workshop*, Mar. 1997.
[14] L. Hu, "Distributed code assignments for CDMA packet radio networks," *IEEE/ACM Trans. Networking*, pp. 668–677, Dec. 1993.
[15] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 1265–1275, Sept. 1997.

[16] _____, "Real-time support in multihop wireless networks," *ACM-Baltzer J. Wireless Networks*, to be published.
[17] Y. Ofek, "Generating a fault tolerant global clock using high-speed control signals for the MetaNet architecture," *IEEE Trans. Commun.*, vol. 42, no. 5, pp. 2179–2188, 1994.
[18] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless network," in *Proc. IEEE INFOCOM'97*.
[19] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM '94*, pp. 234–244.
[20] D. Raychaudhuri, "Wireless ATM networks: Architecture, system design and prototyping," *IEEE Personal Commun.*, pp. 42–49, Aug. 1996.



**Chunhung Richard Lin** was born in Kaohsiung, Taiwan. He received the B.S. and M.S. degrees from the Department of Computer Science and Information Engineering, National Taiwan University, in 1987 and 1989, respectively, and the Ph.D. degree from the Computer Science Department, University of California, Los Angeles (UCLA), in 1996.

Since 1996, he has been on the Faculty in the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. His research interests include the design and control of personal communication networks, mobile multicasting, protocol design and implementation for a mobile integrated services wireless radio network, and distributed simulation.



**Jain-Shing Liu** was born in Taipei, Taiwan, in 1970. He received the M.S. degree from the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan in 1997. Currently, he is working toward the Ph.D. degree in the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan.

His current research interests include ad hoc wireless network protocol design, cellular communication systems, mobile computing, and high-speed networking.