Admission Control in Time-Slotted Multihop Mobile Networks

Chunhung Richard Lin

***Abstract*—The emergence of nomadic applications have recently generated a lot of interest in next-generation wireless network infrastructures [e.g., the 3G enhanced general packet radio services (EGPRS), UMTS/IMT-2000, etc.] which provide differentiated service classes. So it is important to study how the quality of service (QoS), such as packet loss and bandwidth, should be guaranteed. To accomplish this, we develop an admission control scheme which can guarantee bandwidth for real-time applications in multihop mobile networks. In our scheme, a host need not discover and maintain any information of the network resources status on the routes to another host until a connection request is generated for the communication between the two hosts, unless the former host is offering its services as an intermediate forwarding station to maintain connectivity between two other hosts. This bandwidth guarantee feature is important for a mobile network (e.g., wireless LAN, EGPRS, etc.) to interconnect wired networks with QoS support (e.g., ATM, Internet, etc.). Our connection admission control scheme can also work in a stand-alone mobile *ad hoc* network for real-time applications. This control scheme contains end-to-end bandwidth calculation and bandwidth allocation. Under such a scheme, the source (or the ATM gateway, or enhanced serving GPRS supporting node) 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 the case of ATM interconnection, the bandwidth information can be used to carry out an intelligent handoff between ATM gateways and/or to extend the ATM virtual circuit service to the mobile network with possible renegotiation of QoS parameters at the gateway (e.g., enhanced gateway GPRS supporting node). We examine via simulation the system performance in various QoS traffic flows and mobility environments. Simulation results suggest distinct performance advantages of our protocol calculating the bandwidth information. Furthermore, an "on-demand" feature enhances the performance in the mobile environment because the source can keep more connectivity with enough bandwidth to a receiver in the path-finding duration. Simulation experiments show this improvement.**

***Index Terms*—Admission control, bandwidth routing, call blocking rate, multihop mobile networks, quality of service, slot assignment.**

I. Introduction

A MULTIHOP mobile wireless network can be a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. Mobile hosts communicate with each other using multihop wireless links. Each mobile host in the network also acts as a router, forwarding data packets for other nodes. This kind of network can be implemented over the wireless local area network or the cellular networks (e.g., general packet radio services (GPRS), UMTS, or IMT-2000). A central challenge in the design of this mobile network is the development of dynamic routing protocols that can efficiently find routes between two communicating nodes. The routing protocol must be able to keep up with the high degree of node mobility that often changes the network topology drastically and unpredictably.

A wireless network is often internetworking with 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 which have already appeared in the wireless extensions of the wired ATM networks [1], [14], [20]. Most of them focus on the cellular architecture for wireless PCN (personal communication networks) supported by ATM backbone infrastructures. In this architecture, all mobile hosts in a communication cell can reach a base station (i.e., ATM switch) 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. The problem of interconnecting of 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 VC (virtual circuit) into the wireless network and to carry out an intelligent handoff between ATM gateways or wireless network gateways by selecting the gateway offering 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], [16]. The key to the support of QoS reporting is QoS routing, which provides path bandwidth information at each source. Prior research efforts in multihop mobile networks have not fully addressed this problem.

For a network to deliver QoS guarantees, it must reserve and control resources. The main objective of the routing algorithms presented here is to set up connections in a network. The setup process includes a reservation of network resources for the connection. If these resources are not available, the setup fails, or in other words, the connection is blocked. Our aim is to minimize the blocking probability for a connection. A major challenge in multihop multimedia networks is the ability to account for resources so that bandwidth reservation (in a deterministic or statistical sense) can be placed on them. We note that in single

Manuscript received December 1, 2000; revised June 1, 2001. This work was supported in part by the Ministry of Education, Taiwan, under Contract 89-H-FA07-1-4 "Learning Technology."

The author is with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Taiwan (e-mail: lin@mail.nsysu.edu.tw; lin@cse.nsysu.edu.tw).

Publisher Item Identifier S 0733-8716(01)08483-9.

hop wireless networks (e.g., cellular 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 hop-distance path to the destination, but also the available bandwidth on it. A VC can be accepted if not only it has enough available bandwidth, but also it can not disrupt the existing QoS VCs.

Here we consider "bandwidth" as the QoS parameter (thus omitting signal to interference ratio (SIR) packet loss rate, etc.). This is because a bandwidth guarantee is one of the most critical requirements for real time applications. "Bandwidth" in time-slotted network systems can be measured in terms of the amount of "free" slots. Before a connection is set up, a route must be selected. The routing decision can influence the blocking probability. For example, there are two routing algorithms which differ in the length of the selected routes. On grounds of the consumption of network resources, usually the algorithms which wastes resources have the lower probability of successfully finding sufficient resources for more connections. Thus, the quality of the routing decision also affects the blocking probability in the network. The goal of our QoS routing is to find the shortest path among all paths on which the available bandwidth 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 also have to determine the scheduling of free slots. Though some algorithms were proposed to solve this QoS routing problem, unfortunately they may only work in some special environments [2], [8], [13].

We propose a call admission control which is based on on-demand routing protocol for QoS support in multihop mobile networks. Without maintaining any routing information and exchanging any routing table periodically, a route (VC) with QoS requirements is created on-demand. We consider different QoS traffic flows in the network to evaluate the performance of our scheme. The remainder of this paper is organized as follows. Section II describes the QoS and the end-to-end bandwidth calculation. In Section III, we present the main design principle of the on-demand QoS routing. Section IV presents the simulation experiments and results obtained, and finally Section V concludes the paper.

II. QoS Definition and Bandwidth Calculation

Lin and Liu [9] proposed a new bandwidth routing scheme which contains bandwidth calculation and reservation for mobile *ad hoc* networks. In this protocol, the bandwidth information is embedded in the routing table. By exchanging the routing table, the end-to-end bandwidth of the shortest hop-distance path for a given source-destination pair can be calculated. If there is not enough bandwidth over the shortest path, the call request will be blocked. However, not enough bandwidth over the shortest path does not mean that there does not exist any bandwidth route in the network. That is, there may be a route which meets the bandwidth requirement but is not the shortest

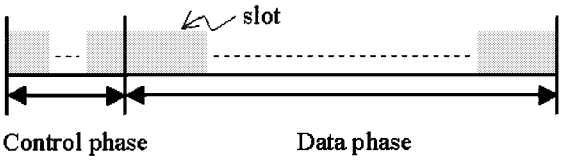

Fig. 1. TDMA time frame structure.

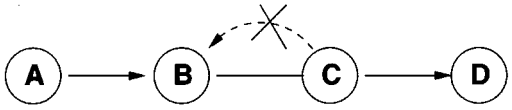

Fig. 2. No collision at node B within the CDMA system.

in hop distance. Therefore, this protocol may miss some feasible bandwidth routes and the blocking probability is high. In our protocol, we would like to route and reserve resources for a connection in a way that: 1) minimizes the blocking probability by attempting several routes in parallel; 2) always considers the feasible route with the minimal cost (the shortest route); 3) selects a route with a cost that is close to the min-cost feasible route; and 4) limits the flow of information and the use of computing resources in that process.

As was the network environment discussed in [9], 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. Namely, time is divided into slots, which are grouped into frames. The frame/slot synchronization mechanism is not described here but can be implemented in the mobile *ad hoc* networks with techniques similar to those employed in the wired networks, e.g., "follow the slowest clock" [12], properly modified to operate in a wireless mobile environment. If the infrastructure is the cellular architecture like enhanced general packet radio services (EGPRS) or IMT-2000, etc., the synchronization is achieved by the radio network controller. Propagation delays will cause imprecision in slot synchronization. However, slot guard times (fractions of microsecond) will amply absorb propagation delay effects (in the order of microseconds). Each time a frame is divided into two phases: a control phase and data phase. The size of each slot in the control phase is much smaller than the one in the data phase. The TDMA time frame structure is shown in Fig. 1. 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, etc. The amount of data slots per frame assigned to a VC is determined according to the bandwidth requirement.

We assume a TDMA within our network; a code division multiple access (CDMA) is overlaid on top of the TDMA infrastructure. Namely, multiple sessions can share a common TDMA slot via CDMA. In this case, the near–far problem and related power control algorithm become critical to the efficiency of the channel access [3]. An ideal code assignment scheme [9] is assumed running in the lower layer of our system, and all spreading codes are completely orthogonal to each other. Thus the hidden terminal problem can be avoided. Consider the example illustrated in Fig. 2. C can use the same slots as A to send packets to D encoded by a different code without any collision

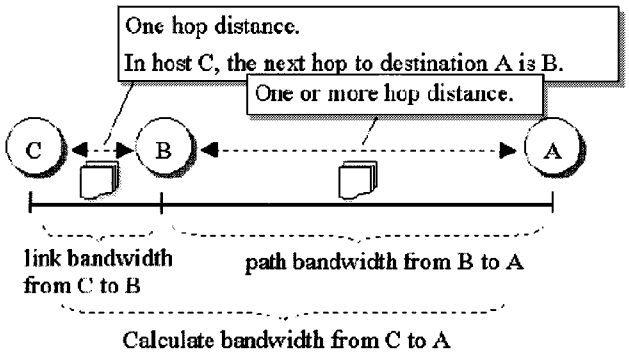

Fig. 3. Bandwidth information calculation overview.

at B. It is notable that this case is assumed 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.

As depicted in Fig. 1, the control phase uses pure TDMA with full power transmission in a common spreading 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 distributedly. We assume the information can be heard by all of its adjacent nodes. In a noisy environment in which the information may not always be heard perfectly at the adjacent nodes, an acknowledgment scheme is performed in which each node has to acknowledge 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 failure of reserved slots, and drop expired real time packets.

The data phase must support both virtual circuit and datagram traffic. Since real time traffic (which is carried on a VC) needs guaranteed bandwidth during its active period, the 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 VCs at call setup time.

Because only adjacent nodes can hear the reservation information, and the network is a 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. 3 in which C intends to compute the bandwidth to A. We assume the next hop is B. By using our end-to-end bandwidth calculation scheme, 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* (also called *end-to-end bandwidth*) between two nodes, which 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. 3 and assume that A is one hop distance from 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

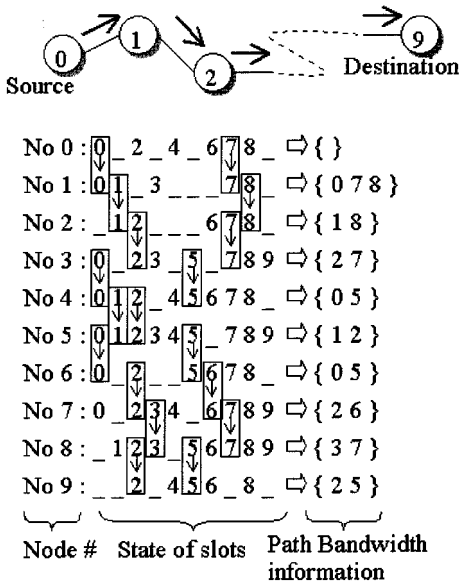

Fig. 4. An example for bandwidth calculation and reservation. The number of data slots is 10. "_" means a reserved slot by the other connections. Bandwidth requirement is two data slots per frame.

that we can only exploit slot 1 and slot 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 $\text{link\_}BW(P, Q) = \text{free\_slot}(P) \cap \text{free\_slot}(Q)$. $\text{free\_slot}(X)$ is defined to be 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 the end-to-end bandwidth. This information can provide us an indication of whether there is enough bandwidth on a given route between a source-destination pair. We will use the example illustrated in Fig. 4 to describe how to calculate the path bandwidth.

In Fig. 4, the source node (node 0) delivers packets to the destination node (node 9) through nodes 1 to 8. We assume there are ten data slots in the data phase. The notation "_" means the slot has been reserved and is not available. The $\text{free\_slot}(0)$, for example, is $\{0, 2, 4, 6, 7, 8\}$, and $\text{free\_slot}(1)$ is $\{0, 1, 3, 7, 8\}$. Obviously, $\text{link\_}BW(1, 0) = \text{path\_}BW(1, 0) = \{0, 7, 8\}$. $\text{path\_}BW(2, 0)$ can be calculated from $\text{link\_}BW(2, 1) = \{1, 7, 8\}$ and $\text{path\_}BW(1, 0)$ according to the bandwidth calculation algorithm illustrated in Fig. 5,[1] and is equal to $\{1, 8\}$. Recursively, $\text{path\_}BW(3, 0)$ can be obtained from $\text{link\_}BW(3, 2)$ and $\text{path\_}BW(2, 0)$, and is equal to $\{2, 7\}$. Finally, $\text{path\_}BW(9, 0)$ is achieved from $\text{link\_}BW(9, 8)$ and $\text{path\_}BW(8, 0)$ and is equal to $\{2, 5\}$. The main principle of computing the $\text{path\_}BW(P, Q)$ is to consider the slots not in $\text{path\_}BW(P, R) \cap \text{link\_}BW(R, Q)$ first, and then the common ones of both set. For example, $\text{path\_}BW(0, 7) = \{2, 6\}$ and $\text{link\_}BW(7, 8) = \{2, 3, 6, 7, 8, 9\}$. Thus, node 8 must choose slots from $\{3, 7, 8, 9\}$ first to forward data from node 7 to the next hop, and node 7 can only choose slots from $\{2, 6\}$ to transmit data to node 8. But the total number of slots chosen

[1]Our previous work in [9] discussed this algorithm in more detail for how to obtain $\text{path\_}BW(i, 0)$ from $\text{link\_}BW(i, i-1)$ and $\text{path\_}BW(i-1, 0)$.

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. 5. Path bandwidth calculation algorithm.

by each node has to be equal. In this case, node 8 chooses $\{3, 7\}$ and node 7 chooses $\{2, 6\}$ (actually, node 8 can choose any pair from $\{3, 7, 8, 9\}$, not necessary $\{3, 7\}$). That is, $\text{path\_}BW(0, 8) = \{3, 7\}$. Finally, $\text{path\_}BW(0, 9) = \{2, 5\}$. This means that node 9 can only reserve either data slot 2 or 5 or both for any new call from source node 0, even though node 9 is recording the slot $\{2, 4, 5, 6, 8\}$ to be free currently. Thus, if the bandwidth requirement of a new call from node 0 to node 9 through the above path is more than two data slots per frame (say QoS > 2), then our admission control will reject this call.

After calculating the end-to-end bandwidth, we need to reserve the data slots from the destination (node 9) hop-by-hop backward to the source (node 0). If $\text{QoS} = 2$, node 9 reserves slot 2 and 5; node 8 reserves slot 3 and 7; node 7 reserves slot 2 and 6, etc.[2] This reservation is not released until the end of the session. On completing the reservation, node 0 begins transmitting datagrams. For details of the reservation see the algorithm we presented in [9].

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. To resolve slot scheduling at the same time as available bandwidth is searched on the entire path is equivalent to solve the satisfiability problem (SAT) which is known to be NP-complete [10]. We use a heuristic approach to assign slots as discussed in [9]. In this bandwidth calculation algorithm, we only compute the size of the path bandwidth. Observe that the information of the end-to-end bandwidth is useful for admission control when a new VC session comes in the system. The admission control can immediately determine whether the VC traffic flow can be accepted at the beginning of connection request according to the bandwidth requirement and available path bandwidth. Actually, this QoS indication enables a more efficient call admission control. It reduces the possibility of the failure of the call setup.

[2]It is notable that in Fig. 4, nodes 0 and 2 are hidden from each other, but they can respectively transmit data packets to node 1 and node 3 at data slot 7 simultaneously without collision at node 1. This is because we assume nodes 0 and 2 use a different spreading code in the CDMA channels as illustrated in Fig. 2.

III. The On-Demand Routing With Bandwidth Constraint

In order to minimize the blocking probability, we need to attempt several routes in parallel for a connection. Additionally, we should always consider the feasible route with the minimal or close to minimal cost (the shortest route). But the flow of control information and the use of computing resources in this process should be limited. In the case of multihop networking, most routing protocols for packet radio networks can be categorized as being *before-demand* (e.g., DSDV [15]) or *on-demand* protocols (e.g., DSR [5], TORA [23] and LMR [24]), or some combination thereof (e.g., AODV [14]). Before-demand protocols compute and maintain routes even if nodes are not actively transmitting packets. Generally, each host needs to maintain a distance–vector-based routing table. In contrast, on-demand protocols compute routes only when necessary. In [9], the shortest path is the only candidate when searching for a feasible bandwidth route. It extends the before-demand routing protocol to further include the information of bandwidth in the routing table. Because of "before-demand" bandwidth calculation, a host can decide either to accept or to reject a new call immediately without any delay. However, in order to minimize the blocking rate, the candidates for the bandwidth route should no longer be the shortest path only. Due to the rapidly changing availability of resources and the processing delay, it is difficult and impractical to use the before-demand routing approach to maintain the pool of candidates for each source-destination pair. In addition, the aforementioned bandwidth calculation scheme is obviously more suitable for on-demand routing. On-demand routing can also save the control messages for maintaining unactive routes. Therefore, we believe on-demand routing is a better choice, and we develop an on-demand routing protocol which not only can search for several routes in parallel for a connection, but also can incorporate our bandwidth calculation scheme. Because it is on-demand, there will be delay for the virtual circuit setup.

A. Route Discovery

Like those on-demand routing protocols [e.g., *ad-hoc* on-demand distance vector routing (AODV) [14], dynamic source routing (DSR) [5], [18], temporally ordered routing algorithm (TORA) [23], and lightweight mobile routing (LMR) [24]], our protocol conforms to a pure on-demand rule. We neither maintain any routing table nor exchange routing information periodically. When a source node wants to communicate with another node for which it has no routing information, it floods a route request (RREQ) packet to its neighbors. If the topology exists, a route from the source to the destination RREQ will find (record) it. In our protocol, all packets contain following uniform fields:

$$\langle \text{packet\_type}, \text{source\_addr}, \text{dest\_addr}, \text{sequence\#}, \text{route\_list}, \text{slot\_array\_list}, \text{data}, TTL \rangle .$$

All packet types which we define are shown in Table I. We use ⟨source_addr, sequence#⟩ to uniquely identify a packet. Here sequence# is monotonically increasing, which can be used to supersede stale cached routes. route_list records the routing information; slot_array_list records the status of slot assignment on the route. When any host receives an RREQ, it will perform the following operations.

TABLE I
All Packet Types and Their Functions

Packet Type	Function
ROUTE_REQUEST (RREQ)	Send to discover route
ROUTE_REPLY (RREP)	Send to reserve route
RESERVE_FAIL	Nack for unsuccessful reservation
ROUTE_BROKEN	Nack for route broken
CLEAN_RREQ	Clean surplus RREQs
NO_ROUTE	Nack for finding no route
DATA	Use to transport datagram

1) If the pair ⟨source_addr, sequence#⟩ for this RREQ was seen recently, discard this redundant request packet and do not process it further.
2) Otherwise, if the address of this node appeared in the route_list in the RREQ, we drop this RREQ (do not rebroadcast it) and do not process it further.
3) Otherwise, *a*) calculate the bandwidth from the source to this node following the algorithm discussed in Section II and record the status of the available data slots to the slot_array_list. We will drop this RREQ if the result does not satisfy the QoS requirement, and do not process it further. Note that we do not modify the state of the data slots at this time. *b*) Decrement *time-to-live* (TTL) by one. If TTL counts down to zero, we drop this RREQ and do not process it further. The purpose of the TTL field is to prevent packets from ending up in infinite loops, which can occur during routing transients. There may exist a very long path which satisfies the bandwidth requirement. However, this path will be difficult to be maintained within a dynamic environment. In addition, unlimited packet flooding will deteriorate the network performance. The use of a TTL can control the flooding traffic. *c*) Append the address of this node to the route_list to track the route which the packet has traversed, and rebroadcast this request if this node is not the destination.

Consider the example in Fig. 4. route_list in RREQ received by node 9 will be (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and the slot_array_list will be [(1, {0, 7, 8}), (2, {1, 8}), (3, {2, 7}), (4, {0, 5}), (5, {1, 2}), (6, {0, 5}), (7, {2, 6}), (8, {3, 7}), (9, {2, 5})]. Each component in the slot_array_list contains the host ID and the set of available time slots. As is to be expected, a destination node will receive more than one RREQ. Every RREQ packet indicates a unique feasible QoS path from the source to the destination. Thus, the destination node can keep more than one path. Multiple connectivity between a source-destination pair can provide a more robust packet delivery. This is especially important in a multihop mobile network. In order to reduce the overhead of flooding, the destination node can broadcast a CLEAN_RREQ packet to clean RREQ packets that are still roaming around the network after receiving enough paths.

B. Route Reservation

When the destination node receives one RREQ packet from the source node, it returns a route reply (RREP) packet by unicasting back to the source following the route recorded in the route_list. Our protocol uses symmetric links between neighboring nodes. It does not attempt to follow paths between nodes when one of the nodes cannot hear the other one; however, we may include the use of such links in future enhancements. As an RREQ travels from the source to the destination, it automatically sets up the reverse path from the destination back to the source. To set up a reverse path, a node records the address of the neighbor from which it received the copy of the RREQ. From the RREQ packets, we can obtain the state of the data slots. According to the information recorded within the RREQ, the destination can set up a bandwidth route and reserve resources (slots) hop-by-hop backward to the source.

Using the source routing algorithm, we copy the fields ⟨route_list, slot_array_list⟩ from RREQ to RREP. As the RREP traverses back to the source, each node along the path reserves those free slots which were calculated in advance. When the source receives an RREP, the end-to-end bandwidth reservation is successful, and the VC is established. Then, the source node can begin transmitting datagrams. It is notable that this establishment protocol for a VC connection from the source to the destination is two-way handshaking. When a new call request arrives, the call admission control drives this establishment protocol. This new call will not be accepted until the reservation process is successfully completed.

C. Unsuccessful Reservation

When the RREP travels back to the source, the reservation operation may not be successful. This may result from the fact that the slots which we want to reserve are occupied a little earlier by another VC or the route breaks. If this is the case, we must give up the route. The interrupted node sends a NACK (i.e., RESERVE_FAIL) back to the destination, and the destination restarts the reservation process again along the next feasible path (note that in the route discovery process, each RREQ which arrives at the destination piggybacks a feasible bandwidth route). All nodes on the route from the interrupted node to the destination must free the reserved data slots when receiving RESERVE_FAIL. If there is no VC that can be setup along all feasible bandwidth routes, the destination broadcasts another NACK (i.e., NO_ROUTE) to notify the source. Upon receiving NO_ROUTE, the source can either restart the discovery process if it still requests a route to the destination or reject the new call. In addition to NO_ROUTE arrival, if there is no response back to the source before the timeout occurs, the source can also reperform the route discovery operation.

Once a VC is established, the source can begin sending datagrams in the data phase. At the end of the session, all reserved slots must be released. These free slots will be contended by all new connections. However, if the last packet is lost, we will not know when the reserved slots should be released. This issue will be discussed next.

D. Connection Breakage

During the active period of a connection, a topological change may destroy a VC. The connection control must reroute or reetablish the VC over a new path. When a route is broken, the breakpoints send a special NACK (i.e., ROUTE_BROKEN) to the source and the destination. That is, once the next hop becomes unreachable, the breakpoint which is near the source sends an unsolicited NACK to the source, and the other breakpoint does so to the destination. Each node along the path relays this ROUTE_BROKEN to its active neighbors and so on. Furthermore, they release all reserved slots for this connection and drop all data packets of this connection which are still waiting for sending in the queue. Upon receiving the ROUTE_BROKEN, the source restarts the discovery process to reestablish a VC over a new path, and the destination only drops the ROUTE_BROKEN (the main purpose of the travel of the ROUTE_BROKEN from the breakpoint to the destination is to release the reserved slots). This procedure is repeated until either the completion of data delivery or timeout. If a timeout occurs, the source stops any data delivery for this session.

The wireless channels have been shown to be distinct and time-varying for each user, depending on their location. Thus, in addition to mobility to break a link, noise may also cause two "adjacent" hosts not to communicate to each other owing to the high packet-error rate on the wireless link. When receiving packets with errors, the receiver just drops them and the sender does not retransmit these packets. In accordance with our bandwidth reservation scheme, the data packet arrival rate of a connection of an intermediate host is the same as the leaving rate. The incoming packets are forwarded within the time period of one TDMA time frame if the next hop is still within the transmission range and the wireless link to the next hop is stable. (A "stable" link means that the signal-to-noise ratio at the receiver is greater than a threshold.) If all wireless links of a virtual circuit are all stable, each intermediate host can constantly have data packets to send in each time frame. Let each reserved data slot be assigned a timer. When a data packet is transmitted on a reserved data slot, its timer is refreshed. Packet loss (because of link breakage or wireless link keeping a high packet-error rate) may cause the timeout. Once a timeout occurs, the reserved time slot is automatically released. A link with a high packet-error rate causes the packets to be dropped. All its downstream links, thus, will free their reserved slots later because of the timeout. For simplicity, in the simulation we let the value be uniform for all wireless links. We let the value be ten times the frame length in our system. Namely, if a reserved slot keeps idle for ten frames, then it will be freed. The value of the timeout will affect the performance of the incomplete ratio and the loss rate of a complete connection. Both will be defined and discussed in the simulation.

If a link on the VC is broken owing to mobility before the last packet reaches the destination to complete a session, the last packet may be suspended within some intermediate node. In this situation, some resources are occupied by this VC and cannot be released for the others. However, the aforementioned timeout scheme for each reserved slot is working in this situation. That is, if a reserved slot is not used to deliver data packets for a couple of frame time periods and timeout occurs, this slot is freed automatically. Such free slots will be fully utilized by the other new connections. Fig. 6 summarizes the operation of our admission control over the on-demand routing algorithm.

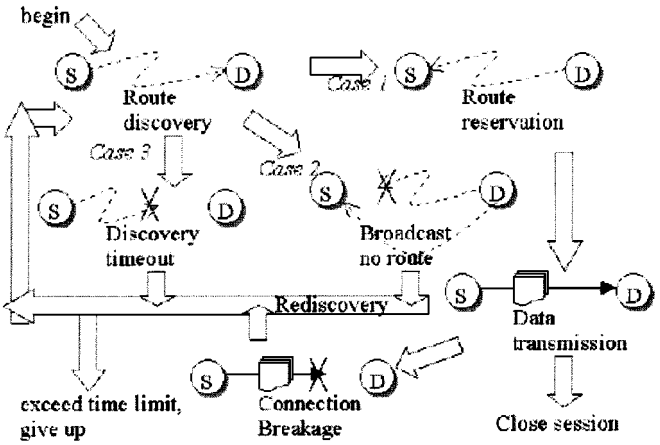

Fig. 6. Overview of the on-demand routing algorithm.

IV. Performance Evaluation

To evaluate the performance of our admission control scheme, we consider the environment which consists of 20 mobile hosts roaming uniformly in 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 2 Mb/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 experiments. Because the VC traffic is delay sensitive rather than error sensitive, packets are not ACKed. A coding scheme is assumed running in the system to do the forward error correction. In the experiments, we will pay more attention to the effect of mobility upon the system performance.

In each time frame (Fig. 1), 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 $20 \cdot 0.1 + 16 \cdot 5 = 82$ 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 (i.e., a link which the RREP passes through), 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 NACK packet (RESERVE_FAIL and ROUTE_BROKEN) is received. Conceptually, this scheme is an extension of packet reservation multiple access (PRMA) [19] to the multihop environment. In addition, in our simulation, for each source-destination pair, the destination keeps three different bandwidth routes which are piggybacked within the first three RREQs arriving at the destination. The default value of TTL is 19. This is the worst case since there are 20 mobile hosts in the system. Table II lists the values of the parameters used in the simulation.

There are three types of QoS for the offered traffic. QoS_1, QoS_2, and QoS_4 need one, two, and four data slots in each frame, respectively. For each simulation result, we consider 100 different topologies and run 10 000 frame times (i.e., 10 000 $*$ 82 ms) for each topology. The interarrival time of two calls is an exponential distribution with the mean value 10 cycles (820 ms). Each session length of QoS_1, QoS_2, and QoS_4 is 100, 200, and 400 packets, respectively. All traffic is assumed the constant bit rate. The interarrival time of packets within QoS_1 is one cycle (82 ms). Similarly, the interarrival times for QoS_2 and QoS_4 are 41 and 21 ms, respectively.

TABLE II
Values of Parameters in the Simulation

Number of nodes	20
Mean of interarrival time of calls	10 cycles
Input queue length	3000
Default TTL	19
Number of data slots	16
Route discovery timeout	(number of nodes * 2) cycles
Total routes kept for each S-D pair	3
Route reservation timeout	Default TTL*(number of route kept*2+1) cycles
Discovery operations	only operate twice for each S-D pair at most

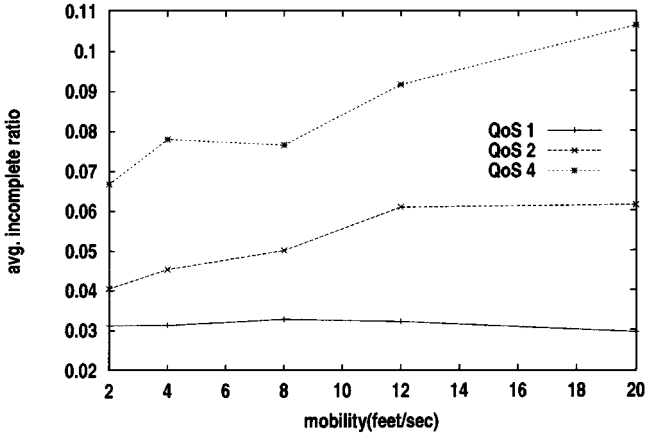

Fig. 7. Incomplete ratio of different QoSs.

In the first experiment, we consider the effect of variable mobility on the incomplete connection ratio. A call may not be completed due to the mobility. If the last packet can arrive at the destination, this session is defined to be complete. Otherwise, it is incomplete. Some data packets of a complete session may be lost because of the topological change. The VC of a connection may be reestablished during the active period and the last packet finally can get to the destination. This is also considered as a complete session. In our simulation, we only allow the source to reperform the route discovery operation once. If the session still cannot be completed, it will be rejected. Fig. 7 illustrates the result. Observe that high mobility causes a path to be broken frequently. When mobility is 20 ft/s for example, QoS_4, QoS_2, and QoS_1, respectively, have 10.5%, 6%, and 3% sessions which cannot be completed. QoS_1 almost keeps a constant incomplete ratio when mobility is less than 20 ft/s. As is to be expected, because the bandwidth route of the lower QoS traffic can be more easily reestablished, the mobility cannot affect it as much as the higher QoS traffic.

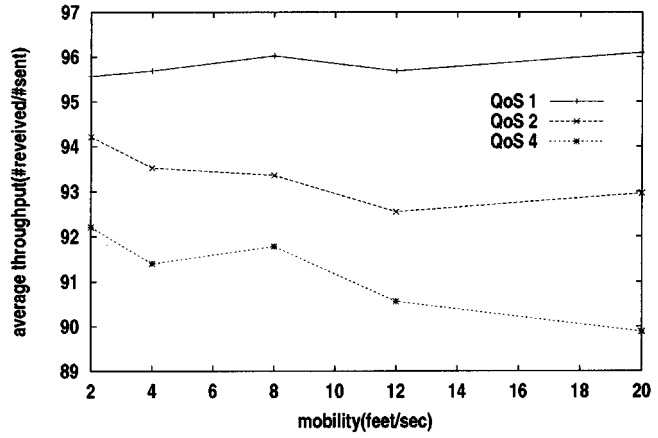

Fig. 8. Average throughput (%) of different QoSs.

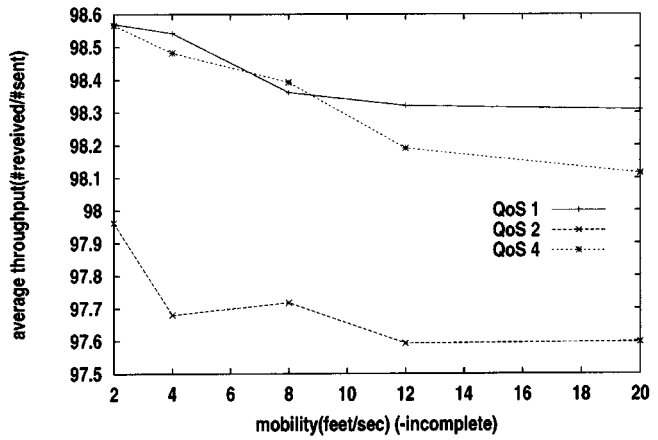

Fig. 9. Average throughput (%) of different QoSs excluding incomplete connections.

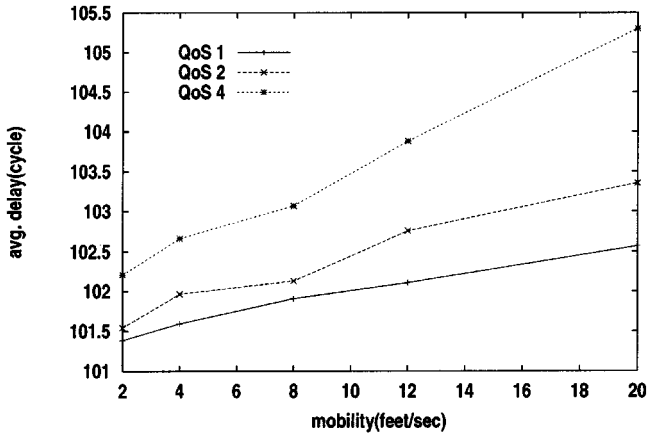

Fig. 10. Average session delay.

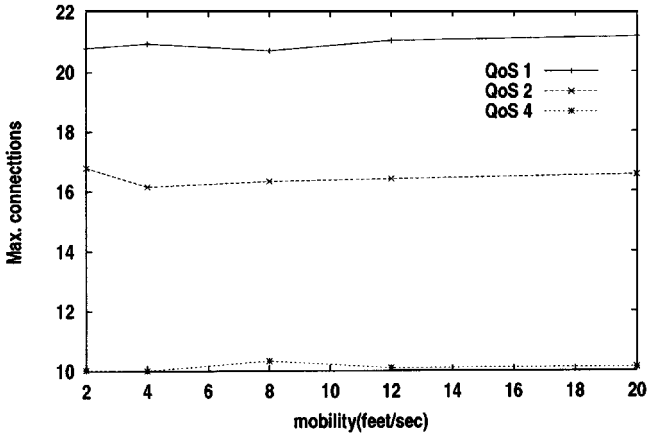

Fig. 11. The maximal number of connections.

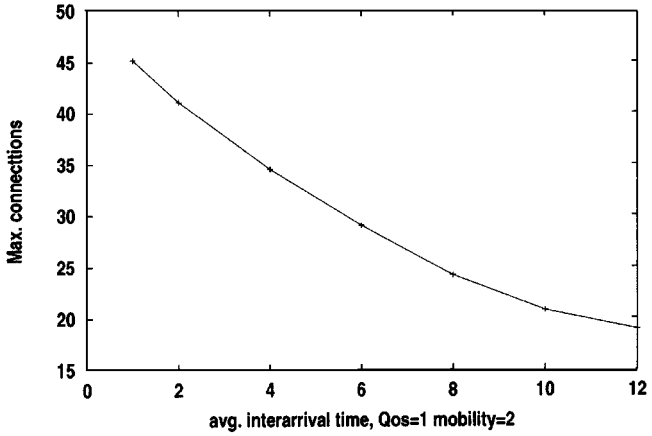

Fig. 12. The maximal number of connections for QoS_1 traffic.

Figs. 8 and 9 show the average throughput of different QoSs. In Fig. 8, we consider both complete and incomplete connections, and in Fig. 9, we only consider the complete ones. The throughput changes slowly and gradually with respect to mobility. In Fig. 8, the mobility does not affect the throughput of all connections crucially. However, in Fig. 9, the throughput is more obviously affected by the mobility. In both figures, the traffic with a lower QoS requirement outperforms the one with a higher QoS requirement. As a general observation, high mobility makes rerouting, thus results in more end-to-end transmission delay and more packet loss. Because the high QoS traffic needs more bandwidth, it is more difficult to find a feasible route when rerouting. In addition, in the rerouting duration, the higher QoS traffic will have more packet loss.

Consider the *complete* connections. We define the *session delay* of a connection to be the interarrival time of the first data packet and the last one. Fig. 10 illustrates the session delay at the destination side. According to our simulation parameters, the session delays at the source side for these three kinds of QoS traffic are all 100 frame time (cycles). If there is no VC reestablishment, the session delay at the destination side also should be 100 cycles. From Fig. 10, we can find the average session delay at the destination side more than 100 cycles. The higher QoS requirement will have more overhead as the mobility increases. This overhead results from the connection reestablishment. When the mobility and QoS level are higher, a connection will spend more time to recover from the path breakage. Fig. 11 shows the maximal number of connections of different QoSs which can be supported by current system resources. There are about 21 connections of QoS_1 traffic simultaneously in the system, and 17 for QoS_2 and 10 for QoS_4. For QoS_1 traffic at a mobility of 2 ft/s, Fig. 12 presents the maximal number of connections for varying mean interarrival times of

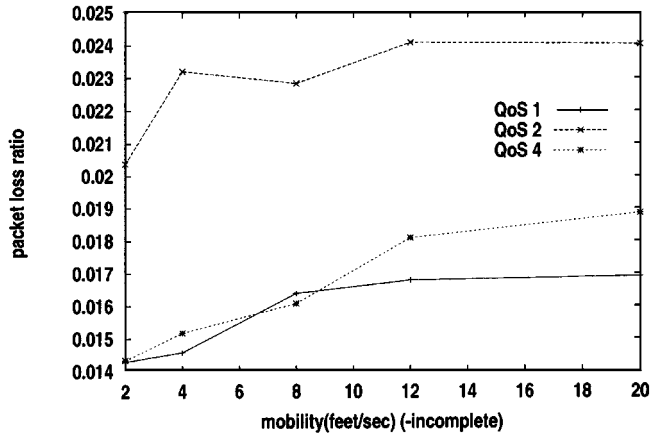

Fig. 13. The average packet loss for "complete" connections.

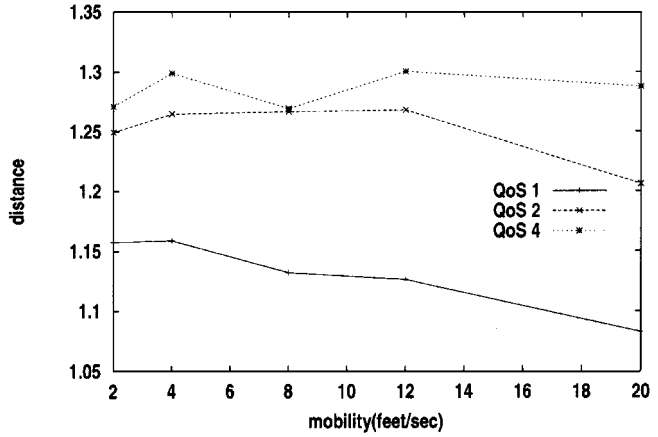

Fig. 14. The average value of d for different QoSs.

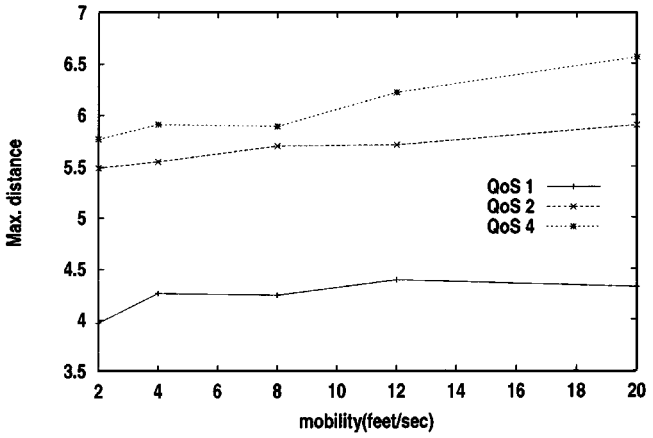

Fig. 15. The maximal value of d for different QoSs.

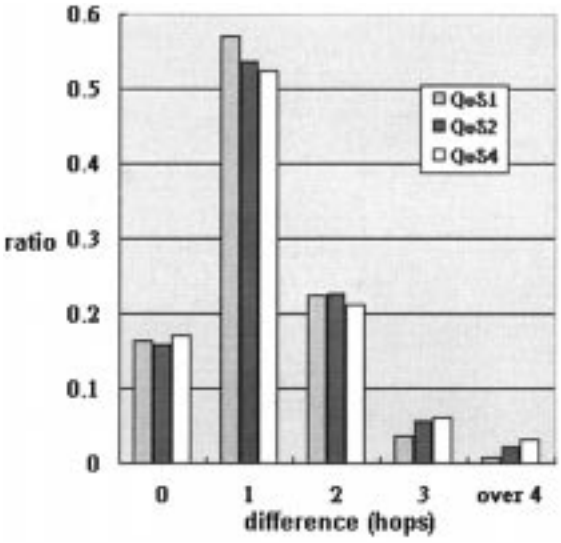

Fig. 16. Difference between the number of hops each packet took to reach its destination and the optimal number of hops required.

calls. There can be up to 45 simultaneously active connections in the system as the mean value is one cycle.

Fig. 13 reports the packet loss rate of all *complete* connections of different QoSs for varying mobility. For these complete connections, we can find that the mobility does not affect this result too much. The packet loss is about 2.4% or less. This loss rate is particularly low. The mobility slightly increases the packet loss rate. The traffic witha lower QoS requirement has a smaller packet loss rate. This is because the lower QoS requirement can make it easier to reconstruct a new VC when the original VC fails.

In the last experiment, we would like to illustrate the performance of the blocking probability. We compare the hop length of the bandwidth route created by our protocol with the optimal shortest path. We let d denote the difference between the shortest path length and the length of the bandwidth route actually taken by data packets. A difference of 0 means the packet took a shortest path, and a difference greater than 0 indicates the number of extra hops the packet took. Fig. 14 shows the average value of d. We can find that the average hop distance of our bandwidth routes is very close to the optimal one. In Fig. 15, we record the average maximal value of d. Observe that the lower QoS traffic has statistically significant optimality in length of the routes with respect to node mobility rate. Fig. 16 further illustrates the percentage of each value of d. We can find that only about 16% of the accepted calls took the shortest paths (i.e., $d = 0$). This means that the bandwidth routes of most of the accepted calls (up to 84%) are not shortest. Therefore, the acceptance rate of the solution proposed in [9] is only 16% of ours. That is, the blocking probability of the solution in [9] is much higher than ours. Our solution can further find those nonshortest bandwidth routes to accept the other 84% calls.

In Sections III-A and III-B, we describe the whole route discovery and reservation processes. A destination node may receive more than one RREQ, and each RREQ packet indicates a unique feasible bandwidth route from the source to the destination. In our experiment, we assume that only the first three routes will be kept by the destination node for route establishment, and the other feasible routes will be dropped. In the duration of the route reservation, the destination will take the first route to reserve the slots hop-by-hop backward to the source. If the reservation cannot succeed because either the route no longer exists or the resources have been occupied, the destination will pick up the next route to reperform the reservation operation. If all three routes cannot complete the hop-by-hop reservation, the call will be rejected. From Fig. 17, we find that most of VCs (near 90%) can be established in the first reservation operation. For the same mobility, if the QoS level is higher, there are more chances for a

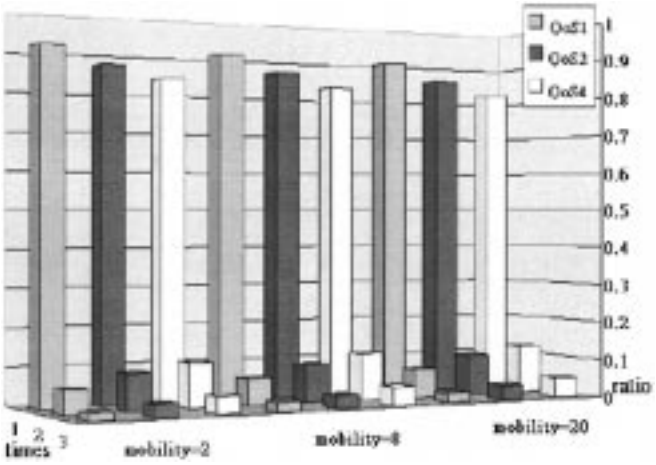

Fig. 17. The reservation operations to be performed to establish a QoS route.

VC to be built in the second or third run. This is because the low QoS VCs are established more easily. Furthermore, consider the effect of mobility (mobility = 2, 8, and 20) upon the same traffic type. This effect is small. The percentage of the VCs, which can be built in the first run of reservation, is low when mobility is high.

V. Conclusion

In summary, we have presented an admission control over an on-demand routing protocol which is suitable for use with multihop mobile networks. Our protocol is more powerful in the resource management than the work in [9]. Thus we can accept more calls in the network according to the simulation results. During the route discovery process, the RREQ packets are used not only to find paths between the source-destination pair, but also to calculate bandwidth hop-by-hop. If there is not enough bandwidth to satisfy the bandwidth requirement at any intermediate node, the route is dropped. Thus, when an RREQ packet arrives at the destination, the route piggybacked on the RREQ packet must have satisfied the end-to-end bandwidth requirement. However, the route may not be the shortest in hop length. In the route reply process, the route reservation is made hop-by-hop backward from the destination to the source. Our admission control 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. In the case of ATM interconnection, ATM virtual circuit service can be extended to the wireless networks with possible renegotiation of QoS parameters at the gateways. In the performance experiments, traffic flows with different QoS types are considered. Finally, more than 60% bandwidth routes created by our protocol are very close to the shortest paths (i.e., less than or equal to one hop difference).

References

[1] A. Acampora, "Wireless ATM: A perspective on issues and prospects," *IEEE Personal Commun.*, vol. 3, pp. 8–17, Aug. 1996.

[2] I. F. Akyildiz, W. Yen, and B. Yener, "A new hierarchical routing protocol for dynamic multihop wireless networks," in *Proc. IEEE INFOCOM'97*, 1997.

[3] N. Bambos and G. J. Pottie, "On power control in high capacity cellular radio networks," in *Proc. IEEE GLOBECOM '92*, 1992, pp. 863–867.

[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LAN's," in *Proc. SIGCOMM '94*, Aug. 1994, pp. 212–225.

[5] J. Broch, D. B. Johnson, and D. A. Maltz, "The dynamic source routing protocol for mobil *ad hoc* networks," *Internet-Draft, Draft-ietf-manet-dsr-00.txt*, Mar. 1998.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless *ad hoc* network routing protocols," in *Proc. ACM/IEEE MobiCom '98*, Oct. 1998, pp. 85–97.

[7] T.-W. Chen, M. Gerla, and J. T.-C. Tsai, "QoS routing performance in a multi-hop, wireless networks," in *Proc. ICUPC '97*, 1997.

[8] Y.-C. Hsu and T.-C. Tsai, "Bandwidth routing in multihop packet radio environment," in *Proc. 3rd Int. Mobile Computing Workshop*, Mar. 1997.

[9] C. R. Lin and J.-S. Liu, "QoS routing in *ad hoc* wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1426–1438, Aug. 1999.

[10] C. R. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 1265–1275, Sept. 1997.

[11] G. Lin and C.-K. Toh, "Performance evaluation of a mobile QoS adaptation strategy for wireless ATM networks," in *Proc. QoS Mini Conf. Conjunction IEEE ICC'99*, June 1999.

[12] Y. Ofek, "Generating a fault tolerant global clock using high-speed control signals for the MetaNet architecture," *IEEE Trans. Commun.*, vol. 42, pp. 2179–2188, 1994.

[13] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. INFOCOM '97*, 1997.

[14] C. E. Perkins, "*Ad hoc* On demand distance vector (AODV) routing," *Internet-Draft, Draft-ietf-manet-aodv-00.txt*, Nov. 1997.

[15] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. SIGCOMM '94*, Aug. 1994, pp. 234–244.

[16] D. Raychaudhuri, "Wireless ATM network: Architecture, system design and prototyping," *IEEE Personal Commun.*, vol. 3, pp. 42–49, Aug. 1996.

[17] W. Su, "Bandwidth allocation strategies for wireless ATM networks using predictive reservation," in *Proc. IEEE Globecom '98*, 1998.

[18] S. Das, C. Perkins, and E. Royer, "Performance comparison of two on-demand routing protocols for *ad hoc* networks," in *IEEE Infocom 2000*, Mar. 2000, pp. 3–12.

[19] D. Goodman, 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.

[20] B. Subbiah, "Transport architecture evolution in UMTS/IMT-2000 cellular networks," in *IEEE Globecom 2000*, Nov. 2000.

[21] Z. Jiang, L. F. Chang, and N. K. Shankaranarayanan, "Providing multiple service classes for bursty data traffic in cellular networks," in *IEEE Infocom 2000*, Mar. 2000.

[22] J. Cai, L. F. Chang, K. Chawla, and X. Qui, "Providing differentiated services in EGPRS through packet scheduling," in *IEEE Globecom 2000*, Nov. 2000.

[23] V. D. Park and M.S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *IEEE Infocom'97*, Apr. 1997.

[24] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *ACM-Baltzer J. Wireless Networks*, vol. 1, pp. 61–81, Jan. 1995.

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.

He joined National Chung Cheng University in Taiwan in 1996. Since August 2000, he has been with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan. His research interests include the design and control of personal communication networks, protocol design and implementation for differentiated/integrated services in mobile wireless networks, and distributed simulation.