Technical Research Report

QoS Routing for Mobile Ad Hoc Networks

by Chenxi Zhu, M. Scott Corson

CSHCN TR 2001-18
(ISR TR 2001-28)

QoS routing for mobile ad hoc networks

Chenxi Zhu and M. Scott Corson
Institute for Systems Research, University of Maryland
College Park, Maryland 20742
(czhu,corson@isr.umd.edu)

Abstract— **A Quality-of-Service (QoS) routing protocol is developed for mobile ad hoc networks. It can establish QoS routes with reserved bandwidth in a network employing TDMA. An efficient algorithm for calculating the end-to-end bandwidth on a path is developed and used together with the route discovery mechanism of AODV to setup QoS routes. Simulations show that the QoS routing protocol can produce higher throughput and lower delay than its best-effort counterpart.**

I. Introduction

The problem of Quality-of-Service (QoS) routing for mobile ad hoc networks is studied. Most routing protocols for mobile ad hoc networks, such as AODV [1], DSR [2], and TORA [3], are designed without explicitly considering quality-of-service of the routes they generate. QoS routing in ad hoc networks has been studied only recently [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. QoS routing requires not only to find a route from a source to a destination, but the route must satisfy the end-to-end QoS requirement, often given in terms of bandwidth or delay. Quality of service is more difficult to guarantee in ad hoc networks than in other type of networks, because the wireless bandwidth is shared among adjacent nodes and the network topology changes as the nodes move. This requires extensive collaboration between the nodes, both to establish the route and to secure the resources necessary to provide the QoS. The ability to provide QoS is heavily dependent on how well the resources are managed at the MAC layer. Among the QoS routing protocols proposed so far, some use generic QoS measures and are not tuned to a particular MAC layer [8], [9], [12]. Some use CDMA to eliminate the interference between different transmissions [4], [5], [10], [13]. Different MAC layer have different requirements for successful transmissions, and a QoS routing protocol developed for one type of MAC layer does not generalize to others easily. So far no work has been done on QoS routing in a flat-architectured, TDMA-based ad hoc network. TDMA transmission is more demanding than CDMA, because transmissions are more likely to interfere. Hence more coordinations among the nodes are required. In this paper we develop a QoS routing protocol for ad hoc networks using TDMA. The object is to establish bandwidth guaranteed QoS routes in small networks whose topologies change at low to medium rate. The protocol is based on AODV, and builds QoS routes only as needed. We assume the application is session-oriented and requires constant bandwidth. A session specifies its QoS requirement as the number of transmission time slots it needs on its route. The QoS routing protocol will both find the route and the slots for each link on the route. We begin with the problem of calculating the available bandwidth on a given route and develop an efficient algorithm. We then use this algorithm in conjunction with AODV to perform QoS routing. At last we study the performance of this QoS routing protocol with simulations and compare it with the original best-effort AODV protocol.

II. The network model

An ad hoc network is modeled as a graph $G = (N, L)$, where N is a finite set of nodes and L is a set of undirected links. The routing protocol will only use bi-directional links, so any unidirectional links are omitted. A node n_i has a set of neighbors $NB_i = \{n_j \in N : (n_i, n_j) \in L\}$. The bandwidth is partitioned into a set of time slots $S = \{s_1, s_2, ..., s_M\}$ which consists a frame. The transmission schedule of node n_i is defined as the set of slots TS_i in which it transmits, and the set of nodes R_i^k which is its transmission target set (receivers) in slot s_k, $s_k \in TS_i$, $R_i^k \in NB_i$. With an abuse of notation we will use TS_i to refer to both the transmission slots set and the transmission targets set for these slots. The set $RS_i = \{s_k \in S : n_i \in R_j^k, n_j \in NB_i\}$ is the set of slots where node n_i is required to receive from its neighbors. Let $TN^k = \{n_i \in N : s_k \in TS_i\}$ be the set of nodes transmitting in slot s_k. A transmission from node n_i to node n_j is labeled as $(n_i \rightarrow n_j)$, or $(n_i \rightarrow n_j)^k$ when we want to emphasize it takes place in slot s_k. The schedule of the entire network TS is the collection $\{TS_i : n_i \in N\}$. The transmission slots can be assigned by some TDMA slot assignment protocol running at the MAC layer. The details of the slot assignment protocol is not important at the moment, but we assume the following conflict-free property always holds:

If a node n_i transmits in slot s_k ($n_i \in TN^k$), for every

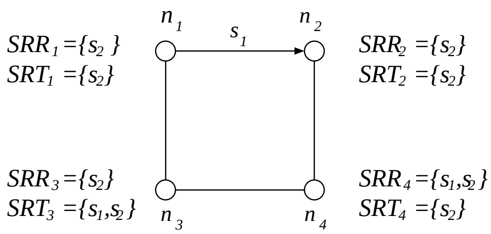

Fig. 1. SRR and SRT for TDMA transmissions. There are 2 slots, $S = \{s_1, s_2\}$. If the current transmission schedule is $(n_1 \to n_2)^1$, $SRR \neq SRT$ for nodes n_3 and n_4.

node $n_j \in R_i^k$, $NB_j \cap TN^k = \{n_i\}$ and $n_j \notin TN^k$.

In other words, when node n_i transmits to n_j in slot s_k, n_j itself does not transmit and n_i is the only transmitting neighbor of n_j in that slot. We define the following sets for a node n_i: $SRT_i = \{s_k \in S : s_k \notin TS_i, s_k \notin RS_i, s_k \notin \cup_{n_j \in NB_i} RS_j\}$, $SRR_i = \{s_k \in S : s_k \notin TS_i, s_k \notin RS_i, s_k \notin \cup_{n_j \in NB_i} TS_j\}$. These are the set of slots when node n_i can transmit without causing interference to its current receiving neighbors (SRT_i), and the set of slots when node n_i can receive without suffering interference from its current transmitting neighbors (SRR_i), given the current transmission schedule TS. The sets SRT_i and SRR_i are not necessarily the same. This is illustrated in the Figure 1. The traffic is session-oriented, where each unidirectional session is also called a flow. A request to setup a QoS route for a session is given in terms of $< Source\_Addr, Dest\_Addr, Flow\_ID, Bandwidth >$. We assume a session requires constant bandwidth and tells the routing protocol how many slots it needs. When a QoS route is established for a flow, new slots need to be reserved on the route. These reservations must be conflict-free. From the prospective of finding a QoS route, the sets $\{SRT_i\}$ and $\{SRR_i\}$ represent all the constraints presented by the current transmission schedule TS, because they dictate what slots are in use and what slots are available. For this reason we also express the transmission schedule as $TS = \{SRT_i, SRR_i, n_i \in N\}$. Given the requirement to establish a session, the QoS routing protocol needs to find a route with sufficient bandwidth, and to determine the set of transmission slots used by each link on the route [1]. This is not easy, because even to find out the maximum available bandwidth along a *given* route is NP-complete. Without causing confusion the terms $path$ and $route$ are used interchangeably. We start from the calculation of the end-to-end bandwidth for a given route.

[1]The job of the QoS routing protocol stops at determining these transmission slots. How the nodes negotiate with each other to ensure these slots are assigned to the corresponding transmitters and are respected by their neighbors is the job of the underlying slot assignment protocol at MAC layer.

III. Calculation of Path Bandwidth

To provide a bandwidth of R slots on a path P, it is necessary that every node along the path find at least R slots to transmit to its downstream neighbor, and these slots do not interfere with other transmissions. Because of these constraints, the end-to-end bandwidth on the path is not simply the bandwidth on the bottleneck link. The path bandwidth calculation problem, termed BWC, can be formulated as follows:

In a network $G = (N, L)$, given the current, conflict-free schedule TS, for a given path P (without loss of generality let $P = \{n_m \to n_{m-1} \to ... \to n_1 \to n_0\}$, $(n_i, n_{i-1}) \in L$, $i = m, m-1, ..., 1$, n_m is the source and n_0 is the destination), find the sets TS_i^P, $n_i \in P \cap \overline{n_0}$, where $TS_i \cap TS_i^P = \emptyset$, the sets $\{TS_i' = TS_i \cup TS_i^P\}$ still satisfy the conflict-free property, and the end-to-end bandwidth on P

$$BW(P) = \min_i |TS_i^P|, n_i \in P \cap \overline{n_0}$$

is maximized. The set TS_i^P is the set of slots where node n_i along P transmits to n_{i-1} to carry packets for the flow, and a transmission in $TS^P = \{TS_i^P : n_i \in P \cap \overline{n_0}\}$ can be called a new transmission or a transmission of P. A transmission in the current schedule TS is called a current transmission. The objective is to find a set of new transmission slots for each node along P so that these transmissions are conflict-free, and the path bandwidth is maximized. We want to find out the maximum available bandwidth of P.

Proposition: Given the current transmission schedule TS is conflict-free, transmission schedule $\{TS_i' = TS_i \cup TS_i^P\}$ is conflict-free iff $TS_i^P \subseteq LB_i = SRT_i \cap SRR_{i-1}$, and $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 1, i \pm 2, n_i, n_j \in P \cap \overline{n_0}$.

Theorem: The problem BWC is NP-complete.

Their proves can be found in [14].

Because the maximum bandwidth for a given path is intractable, we seek alternatives approximating the optimal solution. Instead of searching for the global maximum, the algorithm developed here only searches for local maximum which ends up to sub-optimality. The attraction of this algorithm is that its simple, iterative calculation is well matched to the route discovery mechanism of AODV. The version presented here is termed forward algorithm (FA), because for a path $P = \{n_m, n_{m-1}, ..., n_0\}$, it iterates over the hops from the source n_m to the destination n_0:

Define PB_i^k as the set of slots used on link ($n_i \to n_{i-1}$) to support path $FP^k = \{n_m \to n_{m-1} \to ... \to n_k\}$. Note that FP^k is the partial path of P starting from the source and extends to node n_k, and $FP^0 = P$.

1. If $m = 1$,
$$PB_1^0 = LB_1; \quad (1)$$
2. If $m = 2$,
$$(PB_2^0, PB_1^0) = BW_2(LB_2, LB_1); \quad (2)$$
3. If $m \geq 3$,
$$(PB_m^{m-2}, PB_{m-1}^{m-2}) = BW_2(LB_m, LB_{m-1}); \quad (3)$$
for $k = m-3$ to 0 do
$$\begin{aligned}&(PB_{k+3}^k, PB_{k+2}^k, PB_{k+1}^k)\\ &= BW_3(PB_{k+3}^{k+1}, PB_{k+2}^{k+1}, LB_{k+1});\end{aligned} \quad (4)$$
end;

The available bandwidth on path FP^k is given by
$$BW(FP^k) = |PB_{k+1}^k|. \quad (5)$$
The end-to-end bandwidth of path $P = FP^0$ is
$$BW(P) = BW(FP^0) = |PB_1^0|. \quad (6)$$

Functions BW_1, BW_2 and BW_3 are given in the Appendix. The FA is in fact a greedy scheme which seeks local maximal bandwidth from the source to the next hop, given the sets of slots used to reach the current node. After an iteration, the partial path extends one hop closer to the destination, from FP^{k+1} to FP^k. Only the set of slots on the three links closest to the end n_k are required for the input, and only two of the output variables, PB_{k+2}^k and PB_{k+1}^k, are needed for the next iteration. Because the information required for each iteration is limited and local, the algorithm lends itself easily to distributed implementation. Note that for the link $(n_{k+1} \rightarrow n_k)$, only three sets of slots, $PB_{k+1}^k \supseteq PB_{k+1}^{k-1} \supseteq PB_{k+1}^{k-2}$, are calculated. This is sufficient because transmissions of links further downstream do not interfere with transmissions of $(n_{k+1} \rightarrow n_k)$, therefore $PB_{k+1}^j = PB_{k+1}^{k-2}$ for $0 \leq j < k-2$. The path bandwidth $BW(FP^k) = |PB_{k+1}^k|$ is determined by the three links closest to node n_k, and is non-increasing as FP^k extends towards the destination n_0. Figure 2 shows an example of the FA algorithm.

To evaluate FA, we compare it with an upper bound (UB) for the end-to-end bandwidth on path P with simulations. The upper bound is derived in the appendix. The simulation is carried out on a path with length of M hops. There are total S slots, and the availability of each slot at link $(n_k \rightarrow n_{k-1})$, i.e. LB_k, is modeled as an i.i.d.

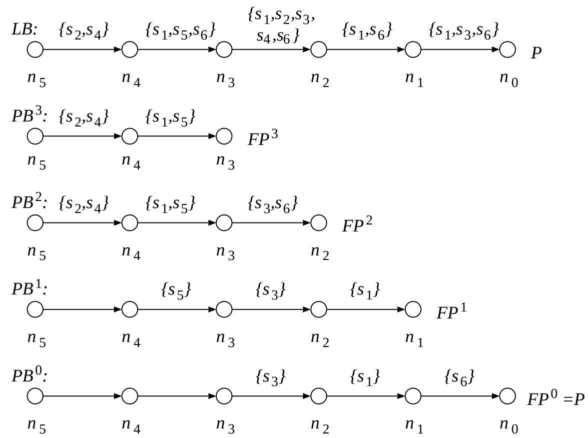

Fig. 2. Bandwidth of a path P calculated by FA.

Bernoulli random variable with probability p_a. The current traffic load on the path is varied by adjusting p_a. The average number of available slots on a link is $E[|LB|] = p_a * S$. Tables 1 compares the bandwidths calculated by FA and UB for a path of 10 hops and 40 slots. The results are averaged over 100 different trials. We found FA and UB are not far from each other, and their relative difference is not sensitive to the path length M or the number of slots S. Therefore FA is an efficient algorithm.

IV. The QoS routing protocol

QoS routing requires finding a route from a source to a destination with required bandwidth. The bandwidth calculation scheme developed above only provides a method to calculate the available bandwidth for a *given* route. It is *not* a routing protocol, and needs to be used together with a routing protocol to perform QoS routing. The routing protocol chosen here is AODV [1]. AODV is a pure on-demand routing protocol and uses a broadcast route dis-

$E[\|LB\|]$	FA	UB
4.0	1.30	1.40
8.0	3.48	3.91
12.0	5.74	6.80
16.0	7.17	8.87
20.0	8.39	10.29
24.0	9.59	11.42
28.0	10.36	12.06
32.0	11.15	12.71
36.0	11.96	13.00
40.0	13.00	13.00

TABLE I
Comparison of FA and UB.

covery mechanism. It relies on dynamically establishing routing table entries. The reason for selecting AODV is that its route discovery mechanism matches the bandwidth calculation scheme very well and is suitable for bandwidth constrained routing. Like AODV, the QoS routing protocol also works on an on-demand basis. A node does not keep routing or bandwidth information it does not need. Currently AODV provides some minimal control to enable nodes to specify Quality of Service parameters, namely maximal delay or minimal bandwidth, that a route to a destination must satisfy [12]. These QoS parameters, however, are generic and their calculations depend on specific networks. The QoS measure used here is bandwidth. In a TDMA network, the bandwidth can be calculated using the FA in the RREQ phase in conjunction with route discovery. Bandwidth is calculated on its path as a RREQ packet is forwarded hop by hop. To find the available bandwidth on a path requires the calculation to be done all the way from end to end. This excludes any node other than the destination to generate a RREP. As a RREQ is forwarded hop by hop and leaves behind a path FP, the available bandwidth for FP is calculated. If a node finds that FP cannot meet the required bandwidth, it drops the RREQ. No RREP is generated for this path. If a RREQ reaches the destination via a path P, a route satisfying the bandwidth requirement is found.

When a source node wants to setup a QoS route for a flow to a destination, it sends a RREQ as it starts the route discovery. The RREQ carries the flow information. A partial path from the source, FP, is set up as the RREQ propagates from the source. The FA is used to calculate the bandwidth on the partial path FP the RREQ has traversed so far. Without loss of generality, assume the source node is n_m, the destination node is n_0, and a RREQ has traveled along a path $FP^{k+1} = \{n_m \rightarrow n_{m-1} \rightarrow ... \rightarrow n_{k+1}\}$, and is being forwarded by node n_{k+1} to its neighbors. As node n_{k+1} transmits the RREQ packet, it appends the following information to the RREQ packet: $< PB^{k+1}_{k+3}, PB^{k+1}_{k+2}, SRT_{k+1} >$. Suppose an one-hop neighbor of n_{k+1}, n_k, receives the RREQ. It calculates:

$$LB_{k+1} = SRT_{k+1} \cap SRR_k, \tag{7}$$

$$(PB^k_{k+3}, PB^k_{k+2}, PB^k_{k+1}) = BW_3(PB^{k+1}_{k+3}, PB^{k+1}_{k+2}, LB_{k+1}). \tag{8}$$

For $k = m-1$ or $k = m-2$, it uses $PB^{m-1}_m = LB_m$ or $(PB^{m-2}_m, PB^{m-2}_{m-1}) = BW_2(LB_m, LB_{m-1})$ in the place of Equation 8. The reason that this calculation is done by node n_k, not n_{k+1}, is to allow node n_{k+1} to broadcast a RREQ packet to all its neighbors. This reduces the computation and the bandwidth consumption, otherwise node n_{k+1} needs to calculate the bandwidth for each of its neighbors and sends the RREQ packet individually. After calculating the bandwidth on the partial path FP^k from the source node to itself, node n_k propagates the RREQ to its neighbors only if $BW(FP^k) = |PB^k_{k+1}| \geq R$. In the meantime, the field $< PB^{k+1}_{k+3}, PB^{k+1}_{k+2}, SRT_{k+1} >$ in the RREQ is replaced by $< PB^k_{k+2}, PB^k_{k+1}, SRT_k >$. Node n_k also sets up an entry for this QoS route and sets the associated state to REQ, indicating it has processed and forwarded the request, but the QoS route has not been established yet. More details about the states associated with a QoS route will be given later. If the required bandwidth R cannot be satisfied on this path, the RREQ packet will be dropped at n_k. No entry will be setup in this case. If a node drops the RREQ packet, it will process the next RREQ packet it receives, even with the same $Broadcast\_ID$. The next RREQ comes from a different neighbor and may have traveled via a path with more bandwidth. The next RREQ is dropped if a RREQ satisfying the bandwidth requirement has been processed and forwarded, i.e. the state of the route is REQ [2]. If a RREQ is forwarded hop by hop without being dropped and reaches the destination n_0 via a path $P = \{n_m \rightarrow n_{m-1} \rightarrow ... \rightarrow n_1 \rightarrow n_0\}$, after the destination calculates and verifies $BW(P) = BW(FP^0) = |PB^0_1| \geq R$, a QoS route P from the source to the destination has been found. The destination node n_0 responds by sending a RREP packet along the path P in the reverse direction. It records the neighbor from which it receives the RREQ as its upstream neighbor on P (so does every other node on P) and sends the RREP to this node. This ensures the RREP and the RREQ packets travel on the same path in opposite directions. The transmission slots $TS^P_i, n_i \in P \cap \overline{n_0}$ will be determined and reserved as the RREP is forwarded towards the source n_m. The destination n_0 calculates the slots used on the last hop ($n_1 \rightarrow n_0$)

$$TS^P_1 = BW_1(PB^0_1, R), \tag{9}$$

and appends TS^P_1 to the RREP packet it sends to n_1. If multiple RREQ arrives at the destination, the first RREQ satisfying the bandwidth requirement is replied and the others are neglected. The reason for the destination not to wait for more RREQs (thus more QoS routes are found and it can choose the best of them) but to use the first QoS route it becomes aware of is to reduce the delay of route

[2] In the original AODV protocol, a node always processes and forwards the first RREQ it receives with a $Broadcast\_ID$ and drops the others in order to control the number of RREQs circulating in the network. With QoS constraint, the first RREQ which satisfies the bandwidth requirement is processed and forwarded and the others are dropped.

discovery. This is suboptimal in the sense that other routes might be shorter or have higher bandwidth. As the RREP packet travels towards the source, transmission slots along the path are determined and reserved and the QoS route is established. The RREP packet transmitted from node n_{k-1} to n_k carries the information $< TS_k^P, TS_{k-1}^P >$. Note that the set of transmission slots TS_k^P on link ($n_k \to n_{k-1}$) is determined by the receiver n_{k-1}. When node n_k receives the RREP, it calculates

$$TS_{k+1}^P = BW_1(PB_{k+1}^k \cap \overline{TS_k^P} \cap \overline{TS_{k-1}^P}, R). \tag{10}$$

After replacing $< TS_k^P, TS_{k-1}^P >$ in the RREP with $< TS_{k+1}^P, TS_k^P >$, n_k passes the RREP to its upstream neighbor n_{k+1}. It also changes the state of the QoS route from REQ to $RESV$. For n_k, the transmission slots TS_k^P can now be reserved. When the RREP reaches the source, every link on path P has found its transmission slots, and a QoS path with bandwidth R has been set up.

In the original AODV protocol, active routes are protected with soft-state. A timer is associated with an active route at a node, and is refreshed each time the route is used to forward a packet. When a route has not been used for sometime, its entry in the routing table is deleted as the timer expires. This ensures every route in the routing table is fresh. Soft-state can also be used with a QoS route. We now describe the soft-states used by the QoS routing protocol. The state of a QoS route at a node can be one of the followings:

1. $NONE$: This node does not have an entry for the QoS route;
2. REQ: A RREQ to set up the QoS route has been processed, but the QoS route is not established yet. No slots are reserved. A node at REQ state will not process or forward any new RREQ packet it receives for the same flow with the same $Broadcast\_ID$;
3. $RESV$: The QoS route has been set up and is used to forward data packets. A node at $RESV$ state will not process or forward any RREQ or RREP packet for the same flow;
4. BRK_U: The QoS route is broken at upstream of this node and is under repair;
5. BRK_D: The QoS route is broken at downstream of this node and is under repair;

Transitions among these states are triggered by events such as receiving or transmitting a packet, or expiration of the timer associated with the state. The conditions and operations associated with these transitions are defined below:

1. $NONE \to REQ$: An entry for a QoS route is setup when the source of the flow sends a RREQ, or when a non-source node receives and forwards a RREQ, or when the destination receives a RREQ and verified there is sufficient bandwidth on the route. A node records the neighbor from which it receives the RREQ as its upstream neighbor on the route. The length of the timer is set to $Route\_setup\_time$.
2. $REQ \to NONE$: The entry for the QoS route is deleted when the timer expires and no route is setup;
3. $REQ \to RESV$: The state becomes $RESV$ when the destination sends out a RREP, or a node on the route, including the source, receives a RREP. An intermediate node also updates the RREP packet and forwards it to the upstream neighbor. It records the neighbor from which it receives this RREP as its downstream neighbor on the route. The length of the timer is reset to $Route\_setup\_time$.
4. $RESV \to RESV$: The state $RESV$ is refreshed when the route is used to transmit a data packet belonging to this flow. The timer is reset to $Route\_life\_time$. Once a route is setup, it is used during the lifetime of the session, unless it breaks due to some topological change. In order not to disturb the packet flow, a QoS route is not changed as long as the required QoS is satisfied;
5. $RESV \to BRK\_U$: The $RESV$ state becomes BRK_U when no data packet arrives for $Route\_life\_time$ and the timer expires. This implies the QoS route is broken at the upstream. The timer is set to $Route\_setup\_time$.
6. $BRK\_U \to RESV$: The QoS route which was broken at upstream is restored. The timer is set to $Route\_setup\_time$. This could happen for three cases. The first case, a data packet belonging to this flow arrives, indicating the QoS route from the source to the current node has been restored. The second case, a node n_k receives a RREQ packet from node $n_{k+1'}$ [3]. After calculating the bandwidth of the path $FP^{k'}$ along which this RREQ traveled from the source to itself, and verifying there is enough bandwidth on this path, it sends out a RREP back to $n_{k+1'}$, even it may not be the destination. Note that node $n_{k+1'}$ is not its upstream neighbor n_{k+1} on the original QoS route (n_{k+1} will reply, rather than forward the RREQ if it receives one). The state transits to $RESV$ when this node sends the RREQ and the timer is set to $Route\_setup\_time$. If this node is the destination, this is identical to the initial route discovery phase. If this node is not the destination, this can be called a local reply. Note that in the initial route discovery

[3] Here we use prime ($'$) to indicate the path the new RREQ has traversed.

phase, only the destination can send a reply. What makes the local reply feasible here is that the part of the original QoS route from this node to the destination (BP^k) still exists, although most likely every downstream node is also at BRK_U state. When the RREP reaches the source, a QoS route is setup between the source and the current node. This, together with the part of the original route from the current node to the destination, restores the entire route. Local reply reduces the delay to restore a broken route. A node sending a local reply also sends a route hold packet (RT_HLD) towards the destination. On receiving the RT_HLD, nodes at the downstream also transit to $RESV$ (this is the third case), so the QoS route at the downstream side is reinstated.

A potential problem for allowing any BRK_U node to locally reply the RREQ is that more than one routes can be built. This happens when more than one BRK_U node send out local replies. Although these routes do not form a loop (they are all from the source to the destination), this is apparently redundant. Which route will be used depends on which RREP reaches the source first. When a node in BRK_U sends a local reply, it may temporarily have two upstream neighbors: the one it sends the local RREP to and the one on the original QoS route. The route from the original neighbor cannot be deleted at this moment, because one of its upstream neighbors could also send a reply (and assume the original down stream route is still good). This route may still be used. As data packets start to flow on one of the routes, they will refresh the $RESV$ states on that particular route. Others routes will time out and be deleted. As a result, route redundancy is only temporary and there is only one QoS route per flow after the states stabilize.

7. $BRK\_U \rightarrow NONE$: The route is deleted at this node if it cannot be restored when the timer expires. The slots TS_k^P are released;
8. $RESV \rightarrow BRK\_D$: When a node finds the link to its downstream breaks, the route breaks and it transits to BRK_D. At the same time it sends a route error packet (RERR) towards the source. A node also transits from $RESV$ to BRK_D when it receives a RERR packet from its downstream neighbor. As the RERR packet is forwarded from the broken link towards the source, every node in this part of the route becomes BRK_D. The timer is set to $Route\_setup\_time$.
9. $BRK\_D \rightarrow REQ$: If this node is the source, it sends out a new RREQ as soon as it receives the RERR and transits to REQ. If this node is not the source, it becomes REQ when it receives (from $n_{k+1'}$) and forwards a RREQ packet. Suppose this node is n_k, and its upstream (downstream) neighbor on the original QoS route is n_{k+1} (n_{k-1}). The transmission slots on link ($n_{k+1} \rightarrow n_k$) is TS_{k+1}^P and on link ($n_k \rightarrow n_{k-1}$) is TS_k^P. It is possible that $n_{k+1'}$ and n_{k+1} are not the same. When processing the RREQ, node n_k uses

$$SRR'_k = SRR_k \cup TS_{k+1}^P, \tag{11}$$
$$SRT'_k = SRT_k \cup TS_k^P \tag{12}$$

in the place of SRR_k and SRT_k. Although slots TS_{k+1}^P and TS_k^P are reserved on the old route, they can be used on the new route as well. The timer is set to $Route\_setup\_time$;
10. $BRK\_D \rightarrow NONE$: The QoS route entry is deleted if no RREQ arrives before the timer expires. The slots TS_k^P are released.
11. $RESV \rightarrow NONE$: When transmission of the session is complete and the QoS route is not needed anymore, the source node sends a route release packet (RT_RLS) to release the route P and the slots TS^P.

$Route\_setup\_time$ and $Route\_life\_time$ should reflect the dynamics of the QoS routing protocol. The timer is set to $Route\_setup\_time$ for route discovery and route repair. It should be long enough for a packet to be transmitted back and forth on the route. $Route\_life\_time$ should be in the order of data packet arrival interval, because on an established route data packets flow regularly and the timer is refreshed by every packet. This allows quick detection once the route breaks and the data packet flow stops. Because soft-states are used and transitions can be triggered by timers, under no circumstances does a node keeps a route forever. Eventually all states become $NONE$, the QoS route is deleted and the time slots are released.

A. An example of route setup and route repair

Figure 3 provides an example of the setup and the repair of a QoS route. Suppose node n_4 wants to setup a QoS route to n_0. It starts the route discovery by transmitting a RREQ. The RREQ packet is forwarded throughout the entire network (Figure 3.a). For simplicity, we assume there is enough bandwidth on every link so the RREQ packet is not dropped. On receiving and forwarding the RREQ, every node sets up an entry for the route and sets the associated soft-state to REQ. When the RREQ reaches the destination n_0 via a path $P = \{n_4 \rightarrow n_3 \rightarrow n_2 \rightarrow n_1 \rightarrow n_0\}$, n_0 sends a RREP to n_4 in the opposite direction of P (Figure 3.b). The state at n_0 becomes $RESV$. On receiving RREP, nodes on P determines and reserves transmission

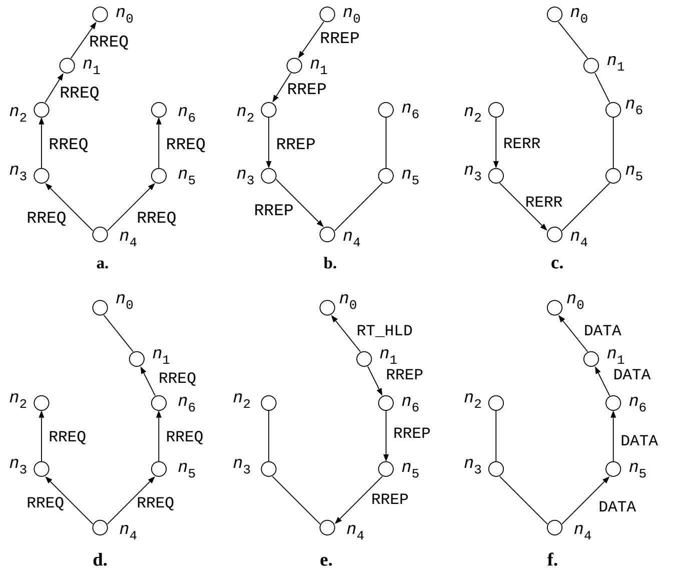

Fig. 3. An example of route setup and route repair with the QoS routing protocol. A arrows the direction of a packet transmission.

slots TS^P. Their states transit to $RESV$. A QoS route P is established. As data packets sent by n_4 travel along P, the $RESV$ states of the nodes on P are refreshed periodically. For a node not on P (n_5, n_6), the route entry is deleted when no RREP packet is received before the timer expires. Suppose sometime later a node n_1 on P moves from the vicinity of n_2 to the vicinity of n_6. The link between n_1 and n_2 breaks and a new link appears between n_1 and n_6. Assume the link between n_1 and n_0 is not affected by this movement. The node upstream of the broken link (n_2) detects its next hop node (n_1) is gone and sends a RERR packet back to the source (Figure 3.c). Nodes n_2, n_3 and n_4 become BRK_D. In the meanwhile, nodes downstream of the broken link (n_1, n_0) time out when they do not receive data packets of the flow for $Route\_life\_time$ and transit to BRK_U. When the source node n_4 receives the RERR packet, it sends out a new RREQ and starts a new round of route discovery (Figure 3.d). Every node which either does not have an entry for the QoS route (n_5, n_6), or where the route state is BRK_D (n_3, n_2) receives and forwards the RREQ. Their states become REQ. When the RREQ reaches n_1 via $FP' = \{n_4 \rightarrow n_5 \rightarrow n_6 \rightarrow n_1\}$, if the soft-state BRK_U at n_1 has not expired, n_1 generates a local reply and sends out the RREP back to the source in the reverse direction of FP' (Figure 3.e). The state at n_1 becomes $RESV$. At the same time n_1 sends a route hold packet (RT_HLD) to its downstream neighbor n_0. Node n_0 also becomes $RESV$. As the RREP is forwarded back to n_4, every node on FP' (n_6, n_5, n_4) determines and reserves their transmission time slots. Their states become $RESV$. The route is restored when the RREP arrives at n_4. The soft-states at nodes n_2, n_3 time out and their route entries are deleted. As data packets flow through this new route $\{n_4 \rightarrow n_5 \rightarrow n_6 \rightarrow n_1 \rightarrow n_0\}$ (Figure 3.f), the $RESV$ state at every node on the route is being refreshed periodically.

V. Simulations results

The performance of the QoS routing protocol is studied with simulations. The QoS routing protocol has been implemented with ns [15]. The implementation is based on the AODV module contributed by the MONARCH group from CMU, and the QoS routing functions are added. In additional to building QoS routes, the protocol also builds a best-effort route when it learns such a route. The best-effort route is used when a QoS route is not available. The Evolutionary-TDMA scheduling protocol (E-TDMA) [16]) developed by the same authors is used at the MAC layer. It is a distributed protocol which dynamically generates and updates TDMA transmission schedules among the nodes. Transmission rate is 1 Mbps. There are 40 slots in a frame, and a slot carries 32 bytes of information. A packet needs to be transmitted in multiple slots if it cannot fit in one slot. An information slot is equivalent to 18 kbps. In a control epoch, nodes contend for a permission in a 2-hop neighborhood for making new slot reservations, and those which succeed can reserve new slots. The control epoch runs at a frequency of 17 Hz and consumes 14% of the bandwidth itself. Thanks to contention, E-TDMA's operation is limited by the nodal density rather than the network size. Details of E-TDMA can be found in [16]. In the simulations, $Rount\_setup\_time$ = 1000 ms and $Route\_life\_time$ = 200 ms. A mobile ad hoc network of 25 nodes is generated in an area of 1000 m by 1000 m. The transmission range of a node is 250 m. A modified "way-point" movement model is used to model the random movement of the nodes [17]. In the beginning, the nodes are randomly placed in the area. Each node remains stationary for a pause time, the duration of which follows an exponential distribution with a mean of 10 seconds. The node then chooses a random point in the area as its destination and starts to move towards it. The speed of the movement follows an uniform distribution between 0 and the maximal speed v. Network mobility is varied when we change v. Different network scenarios for v= 0, 5, 10 m/s are generated. The scenario $v = 0$ represents a static network with no link change. At $v = 10$ m/s, on average a node experiences a link change every 5 seconds. After reaching a destination, a node pauses again and starts to move towards another destination as previously described. This process is repeated for the duration of the simulation (300 seconds). The only constraint of the movement pattern is that it does not cause network partitions, so there is always a route from a source to a destination and no packet is dropped because the destination is unreachable. All dropped packets are due to network congestion or temporary route failure. When the movement pattern is generated, caution is taken to prevent network partition. If a partition occurs, the node causing the partition randomly picks another destination and starts to move towards it. The node does not pause in this case. An example of this network is a group of soldiers moving on foot in a loose formation. Changes in their relative positions are modeled by this movement pattern. In order for the leader to issue command to his soldiers, no one is allowed to stray away, therefore no partition occurs in the network. User traffic is generated with CBR sources, where the source and the destination of a session are chosen randomly among the nodes. During its lifetime of 30 seconds, a CBR source generates 20 packets per second. A CBR source does not adjust its transmission depending on the network congestion, and all 600 packets are always transmitted irrespective of how many of them get through. The size of a CBR packet is 64 bytes, and it becomes 84 bytes after an IP header is added. A packet is transmitted in three time slots. The starting time of a session is randomly chosen between 0 to 270 seconds, so a session always ends naturally by the end of the simulation. The offered traffic load is varied by increasing the number of CBR sessions generated during the simulation from 20 to 360. Ten different traffic patterns are generated and their simulation results are averaged. We measure the number of packets received by the destinations and the average packet delay. We also measure the number of sessions that are serviced and average packet delay for these serviced sessions. A session is called ”serviced” if at least 90% packets are received by the destination. This is a measurement of the quality-of-service provided to the end user (the application layer).

The QoS routing protocol is compared with the original, best-effort (BE) AODV protocol. Figures 4 and 5 show the packet throughput and the average packet delay under different traffic loads and node speeds. Under light traffic, packet throughput and packet delay are very close for the two protocols, because they often use same routes. The advantage of QoS routing protocol becomes apparent when traffic gets heavy. With the BE protocol, a node has one active route to a destination and uses it for all the packets to the destination. As the network traffic becomes heavy, this route becomes heavily loaded, causing packets to be delayed and dropped. The average packet delay increases significantly under heavy traffic. On the other hand, the QoS routing protocol tries to find and use routes satisfying bandwidth constraints for different flows, even between the same pair of source and destination. Two QoS routes may share the same path, but the protocol will ensure enough bandwidths are reserved on this path to accommodate both flows. The traffic load is more balanced this way. The average packet delay increases with offered load slowly with the QoS routing protocol. When the nodal speed v increases, the throughput of both protocols drops. Mobility affects network throughput at both the MAC layer and the routing layer. At the MAC layer, it takes time for E-TDMA to resolve the collisions caused by node movement and to reserve new slots. Essentially a protocol like E-TDMA which is based on establishing reservation has only limited capability to handle network mobility and is best for a static network. At the network layer, it takes time for the routing protocol to re-establish a route when it breaks. For the QoS routing protocol, the packet throughput drops roughly by 15% at v=5 m/s and by 30% at v=10 m/s, compared with $v = 0$. Nodal mobility also increases the average packet delay. The average packet delay nearly doubles at v=10 m/s. Interestingly, when we compare the two routing protocols under mobility, the advantage of QoS routing increases. An explanation is as follows: because the QoS routing protocol uses different QoS routes for individual flows, when one of the QoS routes breaks, only this QoS route is repaired. Other are not affected. Packets of the flow on the broken route are temporarily forwarded using the best-effort route, which may coincide with one of the other QoS routes. There is more route redundancy with QoS routing. In the BE protocol, when the only route to a destination breaks, all packets addressed to this destination are delayed or dropped. It can be expected that a best-effort routing protocol which finds multiple routes will be better than AODV in this aspect.

When the two protocols are compared at the session level (Figures 6 to 8), in the static network both can service almost all the sessions up to 150 sessions. After that the BE protocol degrades until the session good-put drops to about 100. In the meanwhile the QoS routing protocol continues to service more sessions. Average packet delay for serviced sessions is relatively stable in both protocols (usually below 150 ms, which can be tolerated by many real-time applications). Note that the relative performance of the two protocols in terms of session good-put is very different from that of packet-throughput. With the BE protocol, all the packets are treated alike and transmitted on a first-in-first-out (FIFO) bases. Packets from different sessions are equally vulnerable to being dropped. When more sessions are transmitted at the same time, packets are

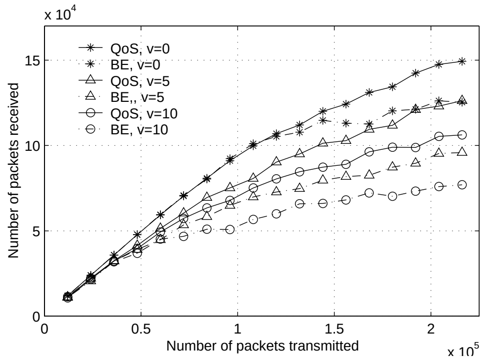

Fig. 4. Packet throughput for $v = 0, 5, 10$ m/s.

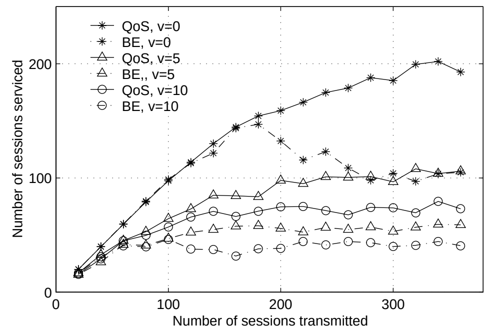

Fig. 6. Session good-put for v=0, 5, 10 m/s.

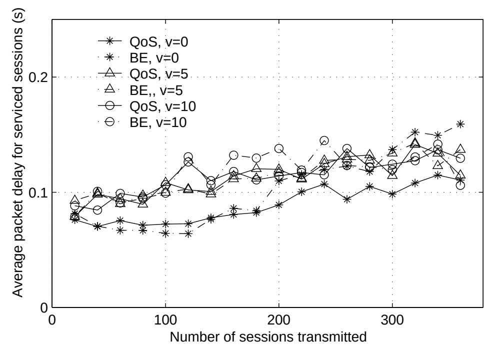

Fig. 7. Average packet delay for serviced sessions.

dropped from all of them and fewer sessions deliver 90% of their packets. With the QoS routing protocol, it is possible to distinguish packets from different sessions. Priority can be given to a packet transmitted on its QoS route before a packet transmitted on a best-effort routed. With the QoS routing protocol the capacity reaches about 200 sessions. When nodes start to move, the session good-put for both protocols decreases significantly. Figure 8 shows that the probability for a session not serviced increases with the nodal speed v. For the QoS routing protocol, session good-put drops to 1/2 and 1/3 at $v = 5$ and 10 m/s respectively compared with $v = 0$. Once a route breaks, before it can be restored, the flow suffers significant degradation, even its packets are transmitted on a best-effort route. The QoS routing protocol offers little protection when this happens until a new QoS route is found for the flow. Because of the bandwidth constraint, a QoS route is not always restored. For $v = 0$, packets from serviced sessions consist of most of the packets received; as v increases, their portion decrease rapidly, indicating many sessions suffer from route failures during transmission. How to protect a flow when its QoS route breaks needs further investigation.

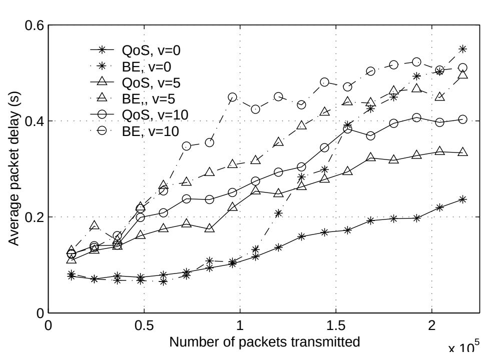

Fig. 5. Average packet delay for $v = 0, 5, 10$ m/s.

VI. Discussions of the QoS and BE protocols

The original AODV protocol is designed for reacting quickly to topology changes in the network. It is very flexible when looking for a route and handles node mobility well. When nodes move very fast, topology could change so quickly that one is lucky to find a route at all, no to mention any QoS. Whether QoS can be achieved in a highly mobile network is questionable. At each node, there is at most one route to any given destination, and this route is changed when a fresher route, or sometimes a shorter route, is known. All the packets addressed to that destination are sent through this route, causing congestion on this route under heavy traffic. This leads to "hot spot" in the network where packets are delayed and dropped.

The QoS routing protocol builds individual QoS routes for different flows, even between the same source and destination. Packets transmitted on QoS routes are guaranteed of bandwidth. When an area of the network is congested, a new QoS route is likely to be built around it rather than through it, providing a way for load balancing. However, a RREQ to set up a QoS route has to reach the destination before it can be replied. A QoS RREQ often travels further than a BE RREQ. In the worst case a QoS RREQ is flooded in the entire network, generating much overhead. Because

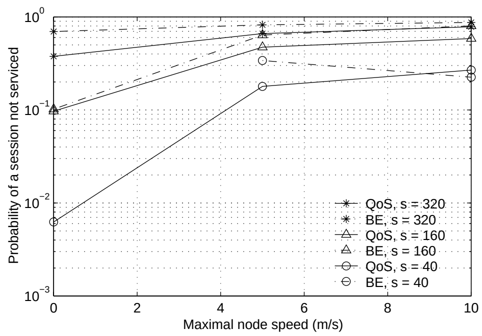

Fig. 8. Probability that a session is not serviced. Load s is the number of sessions transmitted in 300 seconds.

of the requirement for bandwidth reservation, a QoS route is harder to construct than a best-effort route. A long QoS route is more difficult to build and to maintain than a short one, especially under mobility. As nodes move faster and the network topology changes more frequently, it becomes more and more difficult to do QoS routing. All these suggest that the QoS routing protocol is only good for short routes and in networks of low mobility. Consequently QoS routes should be built and used as complement to, not substitute for, best-effort routes.

Another advantage of the QoS routing protocol is related to the E-TDMA protocol used at the MAC layer, where a slot is reserved at a delay cost. Because contention is used for reserving a slot, it works the best when the reservation request is light. More route change requires more reservation and leads to longer reservation delay. Because route change is less frequent with QoS routing, E-TDMA works better for QoS than for BE routing protocol. However, these are characteristic of E-TDMA and may not be true if other protocols are used.

A major criticism of this QoS routing protocol is that it is designed without considering the situation when multiple QoS routes are being setup simultaneously. A route request is processed under the assumption that it is the only one in the network at the moment. When multiple routes are being setup simultaneously, they each reserve their own transmission time slots. When they cross, they may compete for the same set of slots and interfere with one another. It is possible that two QoS routes will block each other when they are trying to reserve the same time slots simultaneously; but if the two requests come one after another, one of them will be successful. This is because no attempt is made to coordinate different route requests. This is not a problem for the BE protocol, because no resource reservation is necessary and two routes can simply cross each other. However, the use of soft-states ensures there will not be deadlocks between the two competing QoS routes. If two QoS routes cannot be fully established because they are blocking each other, both will be deleted. How to setup QoS routes when there are multiple competing requests needs further study.

VII. Conclusion

An on-demand QoS routing protocol based on AODV is developed for TDMA-based mobile ad hoc networks. It can build a QoS route from a source to a destination with reserved bandwidth. We developed a distributed algorithm for calculating the end-to-end bandwidth on a path efficiently. This bandwidth calculation algorithm is integrated into the AODV protocol in search of routes satisfying the bandwidth requirements. The QoS routing protocol can also restore a route when it breaks due to some topological change. Therefore it can handle some degree of network mobility. Its performance is compared with that of the original AODV protocol with simulations. The simulation results show that the QoS routing protocol can produce higher throughput and lower delay than the best-effort protocol. It works the best in small networks or short routes under low network mobility.

Appendix

A. Functions BW_1, BW_2 and BW_3

$function\ (OUT) = BW_1(IN, n)$
  $assert(n \le |IN|);$
  choose n elements from IN randomly as OUT;
  $return.$

$function\ (OUT_2, OUT_1) = BW_2(IN_2, IN_1)$
  $C = IN_1 \cap IN_2;$
  $E_1 = IN_1 \cap \overline{IN_2};$
  $E_2 = IN_2 \cap \overline{IN_1};$
  $if\ |E_2| \ge |IN_1|$
    $OUT_2 = BW_1(E_2, |IN_1|)$
    $OUT_1 = IN_1;$
    $return;$
  $else\ if\ |E_1| \ge |IN_2|$
    $OUT_1 = BW_1(E_1, |IN_2|);$
    $OUT_2 = IN_2;$
    $return;$
  $else$
    $T = floor(|IN_1 \cup IN_2|/2)$
    $C_2 = BW_1(C, T - |E_2|);$
    $C_1 = C \cap \overline{C_2};$
    $OUT_1 = BW_1(C_1 \cup E_1, T);$
    $OUT_2 = BW_1(C_2 \cup E_2, T);$
    $return.$

$function\ (OUT_3, OUT_2, OUT_1) = BW_3(IN_3, IN_2, IN_1)$
  $assert(|IN_3| = |IN_2|\ \&\&\ IN_2 \cap IN_3 = \emptyset);$
  $C_{21} = IN_2 \cap IN_1;$
  $C_{31} = IN_3 \cap IN_1;$
  $E_1 = IN_1 \cap \overline{C_{21}} \cap \overline{C_{31}};$
  $E_2 = IN_2 \cap \overline{C_{21}};$
  $E_3 = IN_3 \cap \overline{C_{31}};$

$$
\begin{aligned}
&if\ |E_1| \geq |IN_2| \\
&\quad OUT_1 = BW_1(E_1, |IN_2|); \\
&\quad OUT_2 = IN_2; \\
&\quad OUT_3 = IN_3; \\
&\quad return; \\
&else\ if\ |E_3| \geq |BW_2(IN_2, IN_1)| \\
&\quad (OUT_2, OUT_1) = BW_2(IN_2, IN_1); \\
&\quad OUT_3 = BW_1(E_3, |OUT_1|); \\
&\quad return; \\
&else\ if\ |E_2| \geq |BW_2(IN_3, IN_1)| \\
&\quad (OUT_3, OUT_1) = BW_2(IN_3, IN_1) \\
&\quad OUT_2 = BW_1(E_2, |OUT_1|); \\
&\quad return; \\
&else \\
&\quad T = floor(|IN_3 \cup IN_2 \cup IN_1|/3) \\
&\quad C_{31}^3 = BW_1(C_{31}, T - |E_3|); \\
&\quad C_{31}^1 = C_{31} \cap \overline{C_{31}^3}; \\
&\quad C_{21}^2 = BW_1(C_{21}, T - |E_2|); \\
&\quad C_{21}^1 = C_{21} \cap \overline{C_{21}^2}; \\
&\quad OUT_1 = BW_1(E_1 \cup C_{21}^1 \cup C_{31}^1, T); \\
&\quad OUT_2 = E_2 \cup C_{21}^2; \\
&\quad OUT_3 = E_3 \cup C_{31}^3; \\
&\quad return.
\end{aligned}
$$

B. *An upper bound of the end-to-end bandwidth*

An upperbound on a path $P = \{n_m \to ... \to n_0\}$ is obtained by observing that the bandwidth of the entire path cannot be higher than the bandwidth on portion of the path which consists of three adjacent links on P, $PP_k^3 = \{n_{k+3} \to n_{k+2} \to n_{k+1} \to n_k\}$. The upperbound is given by

$$UB(P) = \min_k BW(PP_k^3), k = 0, 1, ..., m-3,$$

where the bandwidth $BW(PP_k^3)$ from n_{k+3} to n_k is calculated with integer linear programming

$$
\begin{aligned}
&BW(PP_k^3) = \max B \\
&s.t. \\
&C_{12}^1 + C_{12}^2 \leq C_{12}, \\
&C_{13}^1 + C_{13}^3 \leq C_{13}, \\
&C_{23}^2 + C_{23}^3 \leq C_{23}, \\
&C_{123}^1 + C_{123}^2 + C_{123}^3 \leq C_{123}, \\
&B - C_{12}^1 - C_{13}^1 - C_{123}^1 \leq E_1, \\
&B - C_{12}^2 - C_{23}^2 - C_{123}^2 \leq E_2, \\
&B - C_{13}^3 - C_{23}^3 - C_{123}^3 \leq E_3, \\
&C_{123} = |LB_{k+1} \cap LB_{k+2} \cap LB_{k+3}|, \\
&C_{12} = |LB_{k+1} \cap LB_{k+2} \cap \overline{LB_{k+3}}|, \\
&C_{13} = |LB_{k+1} \cap \overline{LB_{k+2}} \cap LB_{k+3}|, \\
&C_{23} = |\overline{LB_{k+1}} \cap LB_{k+2} \cap LB_{k+3}|, \\
&E_1 = |LB_{k+1} \cap \overline{LB_{k+2}} \cap \overline{LB_{k+3}}|, \\
&E_2 = |\overline{LB_{k+1}} \cap LB_{k+2} \cap \overline{LB_{k+3}}|, \\
&E_3 = |\overline{LB_{k+1}} \cap \overline{LB_{k+2}} \cap LB_{k+3}|.
\end{aligned}
$$

The variables B, C and E are non-negative integers.

REFERENCES

[1] C. Perkins, E. M. Royer and S. R. Das. Ad Hoc On-Demand Distance Vector routing. In *Internet-Draft, draft-ietf-manet-aodv-06.txt*, July 2000.

[2] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editor, *Mobile Computing*. Kluwer Academic Publ., 1996.

[3] V. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proc. INFOCOM*, Kobe, Japan, April 1997.

[4] J. Tsai T. Chen and M. Gerla. QoS Routing Performance in Multihop, Multimedia, Wireless Networks. In *Proc. of IEEE ICUPC*, 1997.

[5] Y.-C. Hsu and T.-C. Tsai. Bandwidth routing in multihop packet radio environment. In *Proc. 3rd Int. Mobile Computing Workship*, 1997.

[6] A. Michail, W. Chen and A. Ephremides. Distributed routing and resource allocation for connection-oriented traffic in ad-hoc wireless networks. In *Proc. of the Conference on Information Sciences and Systems (CISS-98)*, 1998.

[7] A. Michail and A. Ephremides. A distributed routing algorithm for supporting connection-oriented service in wireless networks with time-varying connectivity. In *Proceedings of the 3rd IEEE International Symposium on Communications and Control*, 1998.

[8] S. Lee and A. T. Campbell. INSIGNIA: In-band signalling support for QoS in mobile ad hoc networks. In *Proc. of the 5th Intl. Workshop on Mobile Multimedia Communication*, 1998.

[9] S. Chen and K. Nahrstedt. Distributed Quality-of-Service in Ad-Hoc Networks. *IEEE J. Sel. Areas Commun.*, SAC-17(8), 1999.

[10] C. R. Lin and J.-S. Liu. QoS Routing in Ad Hoc Wireless Networks. *IEEE J. Sel. Areas Commun.*, SAC-17(8):1426–1438, 1999.

[11] D. H. Cansever, A. M Michelson and A. H. Levesque. Quality of Service Support in Mobile ad-hoc IP Networks. In *Proc. MILCOM*, 1999.

[12] Elizabeth M. Royer Charles Perkins and Samir R. Das. Quality of Service for Ad Hoc On-Demand Distance Vector Routing. In *Internet-Draft, draft-ietf-manet-aodvqos-00.txt*, Work in Progress, July 2000.

[13] C. R. Lin. On-demand QoS routing in multihop mobile netowrks. In *Proc. of Infocom*, 2001.

[14] C. Zhu and M. S. Corson. Bandwidth calculation in a tdma-based ad hoc network. In *Proc. of ATIRP*, 2001.

[15] The VINT Project: A collaboratoin between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PAR. *The NS manual*, May 2001.

[16] C. Zhu and M. S. Corson. A new protocol to schedule TDMA transmissions in for mobile ad hoc networks. submitted to INFOCOM 2002.

[17] 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 protocol. In *Proc. of MOBICOM*, 1998.