

# Bandwidth Calculation in a TDMA-based Ad Hoc Network

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

## Abstract

Bandwidth calculation for Quality-of-Service (QoS) routing in an ad hoc network employing Time-Division-Multiple-Access (TDMA) is studied. Certain constraints of TDMA transmission in a wireless network requires careful scheduling among the nodes in order to achieve conflict-free operations. These constraints also make the calculation of the end-to-end bandwidth along a path non-trivial. These calculations are essential for QoS routing which requires a certain amount of bandwidth available on a route. We prove the problem of calculating the maximal end-to-end bandwidth along a given a path in a TDMA network is NP-complete, and develop an efficient bandwidth calculation scheme. We also show how the bandwidth calculation scheme can be used with the Ad-hoc On-demand Distance Vector protocol (AODV) to perform QoS routing.

## 1 Introduction

We consider the problem of quality of service (QoS) routing and bandwidth calculation in a TDMA-based ad hoc network. An ad hoc network is an autonomous network consisting of a set of wireless, potentially mobile, communication nodes, which do not rely on any predefined or fixed infrastructure. Such a network can be rapidly deployed in an area, and their survivability and flexibility makes them ideal for battlefield communications and disaster relief operations. These network have gained much attention in recent years, with a lot of work done on routing in these networks. In particular, the Ad-hoc On-Demand Distance Vector routing protocol(AODV) [1], the Dyanmic Source Routing protocol(DSR) [2], and the Temorary-Ordered RoutingAlgorithm(TORA) [3] all demonstrated the ability to route connectionless data packets efficiently through the network. They are designed primarily to carry best-effort traffic, and at the center of their design is the connectivity between the nodes, or the network topology as a whole. Little attention is paid to the amount of available bandwidth of these links, or the end-to-end quality of service delivered to the upper layers. A recent review of routing protocols for mobile ad hoc networks can be found in [4]. Only recently have people turned their attentions to establishing quality of service routes in ad hoc networks [5, 6, 7, 8, 9, 10, 11, 12, 13]. QoS routing requires not only to find a route connecting the source to the destination, but the route can satisfy the end-to-end QoS requirement, often given in terms of bandwidth

and delay. Quality of service is more difficult to guarantee in a wireless network than in a fixed, wireline network, because the bandwidth resource is usually shared among adjacent nodes due to the broadcast nature of the wireless medium, which is subject to collision and loss, and the potential topological change caused by nodal movement. This requires extensive collaboration between the nodes, both to establish and to maintain the QoS route and secure the necessary resources. Among the QoS routing protocols proposed so far, some of them use generic QoS measures and are not tuned to a particular kind of link layer [9, 10, 13]. Some are designed for a MAC layer which uses spread spectrum for transmission in a time slotted channel [5, 6, 11]. Different MAC layer models have different constraints for conflict-free transmissions, and the algorithm generated for one kind of MAC layer does not generalize to others easily. So far no work has been done to study QoS routing in a flat-architected, TDMA-based ad hoc network. TDMA transmission is more demanding than spread spectrum techniques, because without using different transmission code, transmissions from different nodes are more likely to collide. Hence more coordinations among the nodes are required. QoS routing in TDMA-based network is the subject of this paper.

In the following sections, a distributed algorithm for calculating the end-to-end bandwidth for a route in a TDMA-based ad hoc network is presented, and how the algorithm can be used in conjunction with an existing routing scheme, namely Ad hoc on-Demand Distance Vector Routing (AODV) protocol [1], to perform quality-of-service (QoS) routing, is shown. Here the QoS of a route is measured in terms of the amount of bandwidth, measured by the number of time slots per frame, assigned to a route. If the mobility factor can be neglected, once the time slots are assigned, the route will enjoy conflict free transmissions. This guarantees the throughput and delay of the QoS route. When nodes start to move, the conflict-free property no longer holds, therefore the QoS of a route is subject to degradation. However, with the proposed algorithm, the QoS route can be quickly restored, provided there is sufficient bandwidth. This way it can handle nodal mobility to some degree.

## 2 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 (or each undirected link can be viewed as two directed links in opposite directions). The routing protocol will only use bi-directional links, therefore any unidirectional links are neglected. Each 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  $S = \{s_1, s_2, \dots, s_M\}$ . The transmission schedule of node  $n_i$  is defined as the set of slots in which it transmits ( $TS_i$ ), and the set  $R_i^k$  which is its transmission target nodes 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. 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  $(n_i \rightarrow n_j)$ , and  $(n_i \rightarrow n_j)^k$  if we want to emphasis 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 to the nodes 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 node  $n_j \in R_i^k$ ,  $NB_j \cap TN^k = \{n_i\}$  and  $n_j \notin TN^k$ .

In other words, if node  $n_i$  transmits to node  $n_j$  in slot  $s_k$ , node  $n_j$  itself does not transmit and node  $n_i$  is the only transmitting neighbor of  $n_j$  in that slot <sup>1</sup>. For a node  $n_i$ , we can define the following sets:

$$SRT_i = \{s_j \in S : s_j \notin TS_i, s_j \notin RS_i, s_j \notin \cup_{n_k \in NB_i} RS_k\},$$

$$SRR_i = \{s_j \in S : s_j \notin TS_i, s_j \notin RS_i, s_j \notin \cup_{n_k \in NB_i} TS_k\}.$$

Respectively, 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.

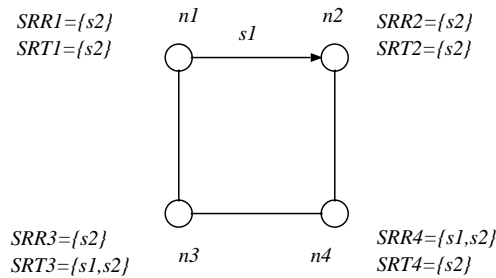


Figure 1: Transmission in a TDMA-based wireless network. Suppose there are 2 slots,  $S = \{s_1, s_2\}$  in the network. If the current transmission schedule is  $(n_1 \rightarrow n_2)^1$ ,  $SRR \neq SRT$  for nodes  $n_3$  and  $n_4$ .

The scenario considered here is connection-oriented traffic, each connection is also called a session. A request to setup a connection is given in terms of  $\langle source\_id, destination\_id, required\_bandwidth \rangle$ . We assume a session requires constant bandwidth, even itself is not necessarily CBR. If a slot only carries the traffic for the session it is assigned to, none of the transmission slots in the current schedule can be used for a new

<sup>1</sup>This marks the difference between the TDMA network studied here and the slotted CDMA network studied in [5, 6, ?]. In a slotted CDMA network which assumes unlimited number of transmission code, it does not require other neighbors of node  $n_j$  to be silent in the same time slot in order for  $n_j$  to receive successfully from  $n_i$ . Different transmissions can use different transmission codes, and the only constraint is that a node cannot receive while it is transmitting.

session. From the prospective of a new session, the sets  $\{SRT_i\}$  and  $\{SRR_i\}$  represent all the constraints presented by the current transmission schedule  $TS$ . 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 task of QoS routing is to find a route with sufficient bandwidth, and to determine the set of transmission slots used by each node along the route to carry the traffic <sup>2</sup>. This is no trivial, because even to calculate the maximal available bandwidth along a *given* route is NP-complete. We start from the calculation of the end-to-end bandwidth along a given path <sup>3</sup>.

### 3 The bandwidth calculation problem (*BWC*)

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

In a TDMA network  $G = (N, L)$ , given the current, conflict free schedule  $TS$ , for a given path  $P$  (without loss of generality let  $P = \{n_m, n_{m-1}, \dots, n_1, n_0\}$ , and  $(n_i, n_{i-1} \in L)$ , where  $n_m$  is the source and  $n_0$  is the destination), find the sets  $TS_i^P, n_i \in P$ , 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 of route  $P$

$$BW(P) = \min_i |TS'_i|, n_i \in P$$

is maximized. The set  $TS_i^P$  is the set of slots where node  $n_i$  along  $P$  transmit to  $n_{i-1}$  to carry the traffic for the session and a transmission in  $TS^P$  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 maximal bandwidth path  $P$  can support.

**Theorem:** The sets  $\{TS'_i\} = \{TS_i \cup TS_i^P\}$  are conflict free iff  $TS_i^P \subset LB_i = (SRT_i \cap SRR_{i-1})$ , and  $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 1, i \pm 2, n_i, n_j \in P$ .

**Proof:** Given the current schedule  $\{TS_i\}$  is conflict free, the new transmission schedule  $TS_i \cup TS_i^P$  is conflict free iff every new transmission in  $TS_i^P$  does not conflict with current transmissions in  $\{TS_i\}$ , and

<sup>2</sup>The job of the QoS routing protocol stops at determining these transmission sets. 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 running at MAC layer.

<sup>3</sup>The terms *path* and *route* are used interchangeably in this paper.

does not conflict with other new transmissions in  $\{TS_i^P\}$ . For a transmission  $(n_i \rightarrow n_{i-1})^k \in \{TS_i^P\}$ , it does not cause interference to current transmissions in  $\{TS\}$  iff  $s_k \in SRT_i$ , and it does not suffer interference from current transmissions in  $\{TS\}$  iff  $s_i \in SRR_{i-1}$ . Therefore  $TS$  and  $TS^P$  do not conflict with each other iff  $TS_i^P \subseteq (SRT_i \cap SRR_{i-1}) = LB_i$ . The set  $LB_i$ , called the link bandwidth for  $(n_i \rightarrow n_{i-1})$ , is the set of slots allowed by  $TS$  for transmission from  $n_i$  to  $n_{i-1}$ . In order for the new transmission  $(n_i \rightarrow n_{i-1}) \in TS^P$  not to interfere with other transmissions in  $TS^P$ , it is necessary and sufficient that any new transmissions of nodes  $n_{i-2}, n_{i-1}, n_{i+1}, n_{i+2}$  do not take place in the same slot. Thereof the condition  $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 1, i \pm 2, n_i, n_j \in P$ . Q.E.D.

As far as the  $BWC$  is concerned, the constraint presented by the current schedule is nothing but the sets  $\{LB_i : n_i \in P\}$ . Therefore in the calculation of  $BW(P)$ , we can ignore the rest of the network and consider only  $\{LB_i : n_i \in P\}$  instead. In the rest of the paper, only the interesting parts of the network  $G$  and the current schedule  $TS$  are represented, namely the path  $P$  and the sets  $\{LB_i : n_i \in P\}$ .

**Theorem:** The problem  $BWC$  is NP-complete.

**Proof:** This can be proven by reducing another NP-complete problem to the problem  $BWC$ . The problem we choose is to calculate the end-to-end bandwidth along a given path in a slotted CDMA wireless network. This problem, termed  $BWC_c$ , is described as following:

In a slotted CDMA wireless network with  $K$  slots,  $CS = \{s_1, s_2, \dots, s_K\}$ . For a path  $CP = \{cn_m, cn_{m-1}, \dots, cn_0\}$ , where  $cn_m$  is the source and  $cn_0$  is the destination, a node  $cn_i \in CP$  can only transmit or receive in a set of slots  $CS_i \subseteq CS$ , and a node cannot transmit or receive simultaneously. If multiple neighbor nodes of  $cn_i$  transmit in the same slot,  $cn_i$  can receive successfully from one of them. Find the set of slots  $TS_i^{CP} \subseteq CS_i$  with which the end-to-end path bandwidth

$$BW_c(CP) = \min_i |TS_i^{CP}|, n_i \in CP$$

is maximized. This problem is NP-complete [5]. Node  $cn_i$  can only transmit to  $cn_{i-1}$  in slots  $TS_i^{CP} \subseteq CLB_i = CS_i \cap CS_{i-1}$ , and  $TS_i^{CP} \cap TS_{i\pm 1}^{CP} = \emptyset$ . The difference from  $BWC$  and  $BWC_c$  is that in a CDMA network, nodes can transmit with different codes (assuming infinite number of codes are available), and can receive the desired signal successfully in the presence of multiple incoming signals (with different codes), while a TDMA network does not have this feature. We now show an instance of  $BWC_c$  can be reduced to an instance of  $BWC$ .

For a path  $CP = \{cn_m, cn_{m-1}, \dots, cn_1, cn_0\}$  in the slotted CDMA network, construct the following path  $P$  in a TDMA network:  $P = \{n_{2m}, n_{2m-1}, n_{2m-2}, \dots, n_1, n_0\}$ , where for each pair of adjacent nodes  $(n_i \rightarrow n_{i-1})$ ,

the link bandwidth  $LB_i$  is given by

$$LB_i = \begin{cases} CLB_{i/2}, & i = 2k; \\ S_o, & i = 4k + 1; \\ S_e, & i = 4k - 1, k \text{ is an integer.} \end{cases}$$

where  $S_o = \{s_{K+1}, s_{K+2}, \dots, s_{2K}\}$  and  $S_e = \{s_{2K+1}, s_{2K+2}, \dots, s_{3K}\}$ . Clearly this transformation can be done in polynomial time. An example of the transformation is given in Figure 2.

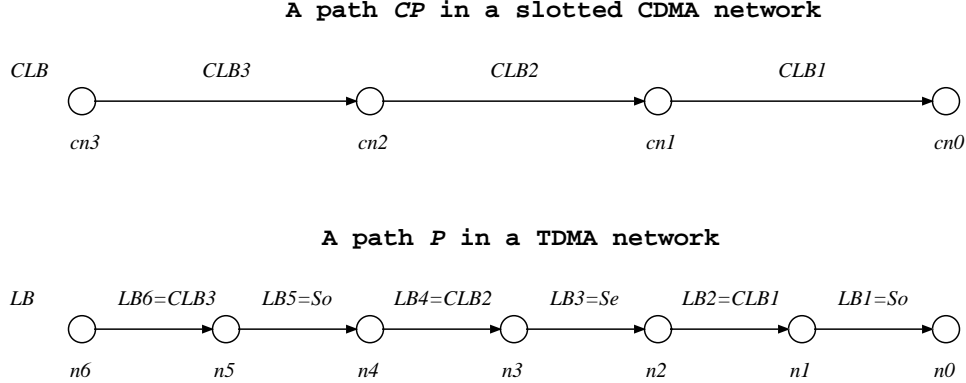


Figure 2: Conversion of bandwidth calculation for a path  $CP$  in a slotted CDMA network to bandwidth calculation for a path  $P$  in a TDMA network.

The total number of time slots in the TDMA network is three times that of the slotted CDMA network, and the path is twice as long. In the TDMA network, the three sets of time slots,  $S_i = CS = \{s_1, \dots, s_k\}$ ,  $S_o$  and  $S_e$ , are mutually disjoint. For a TDMA node  $n_i, i = 2k$ , the constraint  $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 2, i \pm 1$  reduces to  $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 2$ . Similarly for a node  $n_i, i = 4k \pm 1$ , the constraint  $TS_i^P \cap TS_j^P = \emptyset, j = i \pm 2, i \pm 1$  reduces to null. Because a node  $n_i, i = 4k \pm 1$  can transmit in all its  $K$  time slots without interfering with others, they are never the minimizer of  $BW(P)$ . The path bandwidth is therefore given by

$$BW(P) = \min_{n_i \in P} TS_i^P = \min_{i \in \{2, 4, \dots, 2m\}} TS_i^P.$$

If the set of transmission slots  $\{TS_i^P\}$  achieves maximal bandwidth along  $P$  in the TDMA network, the set of transmission slots  $\{TS_i^{CP}\} = \{TS_{2i}^P\}$  achieves maximal bandwidth along  $CP$  in the CDMA network, and vice versa. As a consequence, if we can find the optimal sets  $\{TS_i^P\}$  in the TDMA network with polynomial complexity, we can find the optimal sets  $\{TS_i^{CP}\}$  in the slotted CDMA network. This is unlikely because the problem  $BWC_c$  is NP-complete. Therefore  $BWC$  is NP-complete. Q.E.D.

## 4 A bandwidth calculation algorithm

Because the maximal path bandwidth for a given path is intractable, we seek heuristic alternatives to approximate the optimal solution. Instead of searching over the entire path, the algorithm only performs

localized search which ends up to suboptimality. The attraction of this algorithm lies in its distributed nature and simple, iterative calculations. Its performance will be compared with an upper bound of the end-to-end bandwidth, and we will see it produces acceptable results. Two versions of the algorithms will be presented. The forward algorithm (*FA*) iterates over the hops from the source to the destination, and the backward algorithm (*BA*) iterates from the destination to the source. The term forward and backward only refer to the direction the iteration is carried out, and both the forward algorithm and the backward algorithm calculate the bandwidth from the source to the destination along the same path. Both will be used later. The forward algorithm is presented first:

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

1. If  $m = 1$ ,

$$PB_1^0 = LB_1;$$

2. If  $m = 2$ ,

$$(PB_1^0, PB_2^0) = BW_2(LB_1, LB_2);$$

3. If  $m \geq 3$ ,

$$(PB_m^{m-2}, PB_{m-1}^{m-2}) = BW_2(LB_m, LB_{m-1});$$

for  $k = m - 3$  to 0 do

$$(PB_{k+3}^k, PB_{k+2}^k, PB_{k+1}^k) = BW_3(PB_{k+3}^{k+1}, PB_{k+2}^{k+1}, LB_{k+1});$$

end;

The end-to-end bandwidth of the path  $P$  is given by  $BW(P) = BW(FP^0) = |PB_1^0|$ . The functions  $BW_1$ ,  $BW_2$  and  $BW_3$  are presented in the appendix. The function  $BW_1$  simply outputs the  $n$  elements with the lowest indices (or randomly) of the input <sup>4</sup>. The function  $BW_2$  outputs two disjoint sets of the same size, and each output set is a subset of the corresponding input. The size of the output,  $|BW_2(IN_2, IN_1)| = |OUT_2| = |OUT_1|$ , is maximized. The function  $BW_3$  requires two of the inputs,  $IN_2$  and  $IN_3$ , to be disjoint and have the same size. The output of  $BW_3$  are 3 disjoint sets with the same size, and the size of the output,  $|BW_3(IN_3, IN_2, IN_1)| = |OUT_3| = |OUT_2| = |OUT_1|$ , is maximized. The algorithms  $BW_2$  and  $BW_3$  are optimal, and the forward algorithm, taking advantages of  $BW_2$ ,  $BW_3$ , is in fact a greedy scheme which seeks locally maximal bandwidth from the source to the next hop, given the sets of slots used to reach the current

---

<sup>4</sup>Our simulations did not show significant difference between choosing the lowest elements or choosing randomly.

node. During the  $k$ th iteration, the partial path extends one hop beyond the previous path  $FP^{k+1}$  to reach  $n_k$ . Only the set of slots on the three links closest to the end  $n_k$  are required, 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  $PB_{k+1}^{k-2} = PB_{k+1}^j$  for  $j \leq k-2$ . The path bandwidth of the  $FP^k$  is determined by the three links closest to the end, and is non-increasing as  $FP^k$  extends towards the destination  $n_0$ . The computation cost at each iteration is constant, thus the computation cost for the entire path is proportional to the length of the path. Because at each node only local maximizer is sought, it does not always provide the globally maximal bandwidth, except for  $m < 3$ . Figure 3 shows an example of the  $FA$  algorithm.

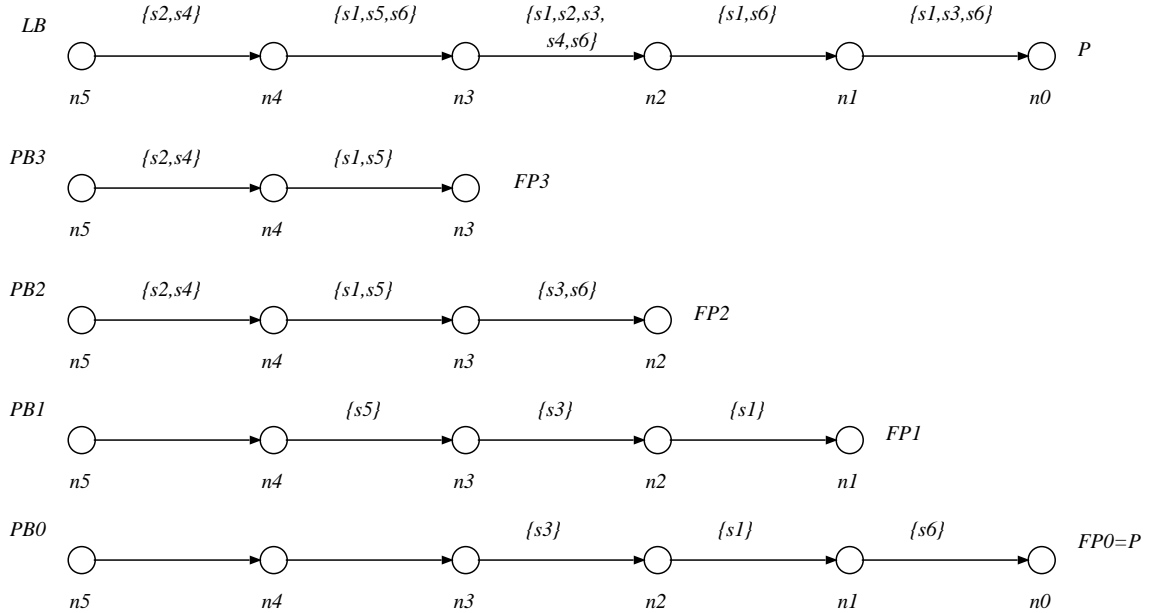


Figure 3: The bandwidth on path  $P$  as calculated by the forward algorithm ( $FA$ ).

We study the performance of the bandwidth calculation algorithm with simulations. Because the maximal end-to-end bandwidth is intractable, we compare the bandwidth obtained with  $FA$  with an upperbound of the end-to-end bandwidth. The upperbound ( $UB$ ) is obtained by observing that the the bandwidth of the entire path  $P$  cannot be higher than the bandwidth on just portion of the path,  $PP_k^3 = \{n_{k+3}, n_{k+2}, n_{k+1}, n_k\}$ . An upperbound is obtained as

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

where the bandwidth on the path  $PP_k^3 = \{n_{k+3}, n_{k+2}, n_{k+1}, n_k\}$  can be calculated with integer linear



programming

$$BW(PP_k^3) = \max B$$

*s.t.*

$$C_{12}^1 + C_{12}^2 \leq C_{12},$$

$$C_{13}^1 + C_{13}^3 \leq C_{13},$$

$$C_{23}^2 + C_{23}^3 \leq C_{23},$$

$$C_{123}^1 + C_{123}^2 + C_{123}^3 \leq C_{123},$$

$$B - C_{12}^1 - C_{13}^1 - C_{123}^1 \leq E_1,$$

$$B - C_{12}^2 - C_{23}^2 - C_{123}^2 \leq E_2,$$

$$B - C_{13}^3 - C_{23}^3 - C_{123}^3 \leq E_3,$$

$$C_{123} = |LB_{k+1} \cap LB_{k+2} \cap LB_{k+3}|,$$

$$C_{12} = |LB_{k+1} \cap LB_{k+2} \cap \overline{LB_{k+3}}|,$$

$$C_{13} = |LB_{k+1} \cap \overline{LB_{k+2}} \cap LB_{k+3}|,$$

$$C_{23} = |\overline{LB_{k+1}} \cap LB_{k+2} \cap LB_{k+3}|,$$

$$E_1 = |LB_{k+1} \cap \overline{LB_{k+2}} \cap \overline{LB_{k+3}}|,$$

$$E_2 = |\overline{LB_{k+1}} \cap LB_{k+2} \cap \overline{LB_{k+3}}|,$$

$$E_3 = |\overline{LB_{k+1}} \cap \overline{LB_{k+2}} \cap LB_{k+3}|.$$

The variables  $B$ ,  $C$  and  $E$  are non-negative integers. The  $BW(PP_k^3)$  is calculated with CPLEX [14], a commercial mathematical programming software. Usually integer linear programming problems are computational intensive, but the calculation of  $BW(PP_k^3)$  is relatively inexpensive thanks to the limited number of the variables. 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 each link ( $n_{k+1} \rightarrow n_k$ ), i.e. the link bandwidth  $LB_k$ , is modeled as an i.i.d. 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 (the average cardinality of the link bandwidth) is  $E[|LB_k|] = p_a * S$ . Tables 1,2,3 compare the path bandwidth obtained with the  $FA$  algorithm and the upperbound. Of course, the true maximal bandwidth lies between the upper bound and the bandwidth obtained using the  $FA$  algorithm, and these three are not far apart from each other. By comparing Tables 1,2 and 3, we can see that the relative difference between  $FA(P)$  and  $UB(P)$  is not very sensitive to the path

length  $M$  and the total number of slots  $S$ . Based on the simulations we conclude that the  $FA$  algorithm is an efficient scheme with acceptable performance.

$E[ LB_k ]$	$FA(P)$	$UB(P)$
2.5	0.37	0.39
5.0	1.93	2.21
7.5	3.03	3.66
10.0	4.18	5.12
12.5	4.91	6.08
15.0	5.57	6.92
17.5	6.20	7.42
20.0	6.76	7.86
22.5	7.10	7.99
25.0	8.00	8.00

Table 1: Comparisons of the bandwidth calculation algorithm ( $FA$ ) and the upperbound ( $UB$ ). The average number of available slots on each hop is given  $E[|LB_k|]$ . The path length is  $M = 10$  and the total number of slots is  $S = 25$ . The results are obtained by averaging 100 different trials.

$E[ LB_k ]$	$FA(P)$	$UB(P)$
2.5	0.19	0.21
5.0	1.30	1.55
7.5	2.47	3.21
10.0	3.57	4.71
12.5	4.45	5.87
15.0	5.16	6.78
17.5	5.79	7.03
20.0	6.45	7.66
22.5	6.96	8.00
25.0	8.00	8.00

Table 2: Comparison of  $FA$  and  $UP$  for long paths. The length is  $M = 20$  hops and the total number of slots is  $S = 25$ .

Often a bandwidth requirement of  $R$  slots is known when the bandwidth is calculated. The QoS route can only be established along path  $P$  if the bandwidth requirement can be met ( $BW(P) \geq R$ ) and the transmission slots are identified and reserved. After it has been confirmed by the  $FA$  algorithm that there is sufficient bandwidth along path  $P$ , the time slots used on each hop ( $TS_i^P, n_i \in P$ ) need to be identified. Because  $R \leq BW(P) = |PB_1^0| \leq |PB_k^0|$ , for  $k \geq 1$ , there is sufficient slots along the entire path  $P$  to support the bandwidth of  $R$ . The transmission sets for each hop to support a QoS route along  $P$  with bandwidth requirement of  $R$  can be determined by

$$TS_k^P = BW_1(PB_k^0, R) = BW_1(PB_k^{k-3}, R), \quad k = m, m-1, \dots, 4,$$

$E[ LB_k ]$	$FA(P)$	$UB(P)$
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 3: Comparison of  $FA$  and  $UB$  for high number of time slots. The path length is  $M = 10$  hops and the total number of slots is  $S = 40$ .

$$TS_k^P = BW_1(PB_k^0, R), \quad k = 3, 2, 1.$$

After  $TS_k^P$  is determined, the underlying slot assignment protocol can be used to reserve these time slots at MAC layer.

The bandwidth calculation can also be initiated from the destination and iterated backward to the source. This version of the algorithm is the backward algorithm ( $BA$ ):

Define  $PB_i^k$  as the set of slots used on link  $(n_i \rightarrow n_{i-1})$  to support the partial path  $BP^k = \{n_k, n_{k-1}, \dots, n_0\}$ . Note that  $BP^k$  is from an intermediate node  $n_k$  to the destination node, and  $BP^m = P$ .

1. If  $m = 1$ ,

$$PB_1^1 = LB_1;$$

2. If  $m = 2$ ,

$$(PB_1^2, PB_2^2) = BW_2(LB_1, LB_2);$$

3. If  $m \geq 3$ ,

$$(PB_1^2, PB_2^2) = BW_2(LB_1, LB_2);$$

for  $k = 3$  to  $m$  do

$$(PB_{k-2}^k, PB_{k-1}^k, PB_k^k) = BW_3(PB_{k-2}^{k-1}, PB_{k-1}^{k-1}, LB_k);$$

end;

The end-to-end bandwidth of path  $P$  is given by  $BW(P) = BW(BP^m) = |PB_m^m|$ . On average, the  $FA$  and the  $BA$  schemes have the same performance, but their results for a given path may be different. Both

schemes will be useful in the QoS routing schemes in the following section.

## 5 QoS routing with AODV protocol

QoS routing in an ad hoc network requires finding a route from a source to a destination with sufficient bandwidth. The bandwidth calculation scheme presented above only provides a method to calculate the maximal bandwidth for a *given* route. It is *not* a routing protocol, and needs to be used together with a routing protocol in order to perform QoS routing. The routing protocol chosen here is the Ad hoc on-Demand Distance Vector Routing protocol (AODV) [1]. AODV is a pure on-demand routing protocol and uses a broadcast route discovery mechanism. It relies on dynamically establishing routing table entries. Destination sequence number is used to maintain the most recent routing information between nodes. The following short description of AODV is from [15]. The reader is referred to [1] for more details of AODV.

### 5.1 The AODV protocol

The Ad-Hoc On Demand Distance Vector (AODV) routing algorithm is a routing protocol designed for ad-hoc mobile networks. It is an on demand algorithm, meaning that it builds routes between nodes only as desired by source nodes. It maintains these routes as long as they are needed by the sources. AODV uses sequence numbers to ensure the freshness of routes. It is loop-free, self-starting, and can also build multicast routes.

AODV builds routes using a route request/route reply query cycle. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node receiving the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it unicasts a RREP back to the source. Otherwise, it rebroadcasts the RREQ. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it.

As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller

hop count, it may update its routing information for that destination and begin using the better route.

As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically traveling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

## 5.2 QoS routing with AODV

Currently AODV provides some minimal controls to enable the nodes to specify Quality of Service parameters, namely maximal delay and minimal bandwidth, that a route to a destination must satisfy [13]. These QoS parameters, however, are genetic and their calculations depend on specific networks. In a TDMA based ad hoc network, the bandwidth can be calculated using the scheme developed in Section 4.

Two approaches can be used to build QoS routes. In the first approach, bandwidth calculation is decoupled from route discovery. With this approach, a route is provided by the (original) AODV protocol without considering the bandwidth constraint. The bandwidth calculation is performed on this route to check whether it has enough bandwidth. This can be called the decoupled approach, because the interaction between routing and bandwidth calculation is minimal. In the second approach, called integrated approach, the bandwidth calculation is performed in conjunction with route discovery, and the routing protocol will try to find a path with sufficient bandwidth. The decoupled approach is presented first.

### 5.2.1 The decoupled approach

In the original AODV protocol, the only information used to build a route is the network topology. Hop count and route freshness (sequence number) are used as the route selection criteria, and no bandwidth information is taken into account. A route found by AODV may not have enough bandwidth. When the interest is to find a route with bandwidth  $R$ , the minimal requirement is to determine whether there is sufficient bandwidth along the path found by AODV. Only a path with sufficient bandwidth can be accepted. In the decoupled approach, a path connecting the source with the destination is first found with the AODV protocol without considering the bandwidth requirement. Its bandwidth is calculated afterwards. If the bandwidth requirement cannot be met, the path will be dropped and will not be used for the session.

Therefore bandwidth calculation is used for admission-control purpose only. The calculation can be initiated by either the source node in the forward direction all the way to the destination, or by the destination node in the backward direction all the way to the source. To reduce the delay, the bandwidth is calculated with the *BA* algorithm as the RREP packet is unicasted back from the destination to the source<sup>5</sup>. Assume the RREQ packet travels from the source ( $n_m$ ) to the destination ( $n_0$ ) through a path  $P = \{n_m, n_{m-1}, \dots, n_2, n_1, n_0\}$ . When the RREQ packet arrives at the destination ( $n_0$ ), node  $n_0$  transmits a RREP packet to its upstream neighbor  $n_1$  and starts the bandwidth calculation. As the RREP packet travels along the reverse path and propagates from node  $n_{k-1}$  to  $n_k$ , node  $k$  passes the information  $\langle PB_{k-2}^{k-1}, PB_{k-1}^{k-1}, SRR_{k-1} \rangle$  to  $n_k$ , in addition to the usual information carried in a RREP packet. Node  $n_k$  performs the following calculation:

$$LB_k = SRT_k \cap SRR_{k-1},$$

$$(PB_{k-2}^k, PB_{k-1}^k, PB_k^k) = BW_3(PB_{k-2}^{k-1}, PB_{k-1}^{k-1}, LB_k).$$

If  $BW(BP^k) = |PB_k^k| \geq R$ , node  $n_k$  propagates the RREP one hop closer to the source. If  $BW(BP^k) = |PB_k^k| < R$ , the bandwidth requirement can not be met, and node  $n_k$  drops the RREP. This way if the RREP reaches at the source without being dropped, there is sufficient bandwidth on the path  $P$ . The phase when the initial RREQ is broadcasted in the network and eventually reaches the destination can be called the RREQ phase, and the phase when the RREP travels along the reverse path from the destination to the source can be called RREP phase. On receiving the RREP packet, the source node can send a RESV packet along the path to inform the nodes to reserve the corresponding transmission slots  $TS_k^P$ . The RESV packet transmitted from node  $n_{k+1}$  to  $n_k$  carries the following information:  $\langle source\_addr, dest\_addr, session\_id, TS_{k+2}^P, TS_{k+1}^P \rangle$ . The triplet  $\langle source\_addr, dest\_addr, session\_id \rangle$  uniquely identifies a session. Node  $n_k$  chooses its transmission slot  $TS_k^P$  as

$$TS_k^P = BW_1(PB_k^k \cap \overline{TS_{k+1}^P} \cap \overline{TS_{k+2}^P}, R),$$

and sends a similar RESV packet to its downstream neighbor  $n_{k-1}$ . It is the job of the underlying MAC layer (slot reservation protocol) that reserves the set of slots  $TS_k^P$ . When the RESV reaches the destination, a QoS route with bandwidth  $R$  has been established from the source to the destination. (The source node can start to transmit its data packets as soon as it receives the RREP packet. Before all the slots along the path are reserved at the MAC layer, the packets may be transmitted with a best-effort approach.) This third phase can be called the RESV phase.

<sup>5</sup>This requires the RREP packet to be generated by the destination node only. It disallows a node other than the destination to reply to a RREQ. However, an intermediate node with a fresh route to the destination can speed up the route discovery and reduce the bandwidth consumption by unicasting the RREQ to the destination node instead of broadcasting the RREQ to its neighbors.

It is required that nodes along the path do not move, and the set of available slots ( $SRT_k, SRR_k$ ) do not change, if the path can be setup successfully. The modified AODV protocol has the same susceptibility to nodal movement as the original protocol, but it is also susceptible to changes in the current transmission schedule ( $SRR_k, SRT_k$ ). If ( $SRR_k, SRT_k$ ) on the path change between RREP and RESV, the available bandwidth for the path may change and the QoS path may not be established successfully. A way to prevent the unexpected change of time slots is to use soft-state to protect these slots temporarily<sup>6</sup>. When node  $n_k$  calculates  $PB_k^k$ , a timer is initiated. Before the timer expires, the slots of  $PB_k^k$  cannot be assigned to other sessions. If the RESV packet arrives before the timer expires, these slots are readily available and can be reserved for this session. An important parameter is the timeout time  $t_e$  for this timer on the link ( $n_k \rightarrow n_{k-1}$ ). It should be long enough to allow the RESV packet to arrive, but not too long to hold these time slots unnecessarily. Note that the interval between RREP and RESV is the longest for the last link of the path ( $n_1 \rightarrow n_0$ ), and decreases progressively for the hops closer to the source. If the time for the RREP and RESV packet to travel one hop can be considered as a constant  $t_{hop}$ , the time  $t_e$  should be at least  $2 * t_{hop} * (m - k)$  for the link ( $n_k \rightarrow n_{k-1}$ ). The longer the path, the longer this timeout period. If the network is too large, the path can be too long and this time becomes intolerable. Therefore this soft-state approach is not suitable for long paths.

The bandwidth calculation can also be performed with the *FA* if the source node already has a recent entry in its routing table to the destination. No path discovery over the whole network is required for this case. Still, the source node needs to send a RREQ towards the destination via the path in its routing table in order to calculate the bandwidth along this path. Hop by hop, the RREQ is forwarded towards the destination. On receiving the RREQ, an intermediate node  $n_k$  calculates the bandwidth on the partial path the RREQ has traveled so far (from the source to itself), using the *FA* scheme. The calculation is identical to the calculation in the integrated approach, so we refer to the next subsection for the details. After node  $n_k$  calculates  $PB_{k+1}^k$ , it can protect these time slots with a soft-state. If a RREP arrives before the timer expires, the slots  $TS_{k+1}^P \subseteq PB_{k+1}^k$  can be identified and reserved. When *FA* is used, a link closer to the source has to wait longer for the RREP to arrive. The timer of the soft-state on link ( $n_{k+1} \rightarrow n_k$ ) should be at least  $2 * t_{hop} * k$  long.

Due to the minimal coupling between routing and bandwidth calculation (admission-control), this approach works with other routing protocols as well. A protocol which provides multiple route might be a better choice than AODV. With AODV, the destination node only processes the first RREQ packet it receives. Often this RREQ travels along a path shorter than the paths traveled by other RREQ packets that arrive later.

---

<sup>6</sup>A soft-state is a timer-based state, which put the resource/state of a node on hold temporarily. The resource or state is released when the timer expires.

If routing protocol can provide multiple paths, the bandwidths for these multiple paths can be calculated in the RREP phase, and the chance of finding a path with sufficient bandwidth is enhanced. If more than one paths are found to satisfy the bandwidth requirement, the source node can choose the one with the highest bandwidth (or alternatively use a benchmark which combines available bandwidth and path length) and send the RESV packet along this path <sup>7</sup>. When the soft-states associated with the time slots on the other paths expire, these time slots become available for other sessions.

### 5.2.2 The integrated approach

The integrated approach follows the practice proposed in [13]. The bandwidth calculation is carried out in the RREQ phase in conjunction with route discovery. This way in the RREQ phase the protocol will try to find a path with sufficient bandwidth (not to check whether the path has sufficient bandwidth *after* it has been found). Bandwidth can be calculated on the partial path  $FP^k$  as the RREQ's are propagated, and when the destination node is reachable, at least one of them is forwarded towards the destination. Because when a RREQ propagates, only the reverse path from the source is setup (corresponds to  $FP$  in Section 4), the forward algorithm ( $FA$ ) can be used to calculate the bandwidth on the path from the source to the current node (the  $FP$  the RREQ has traversed so far). Without loss of generality, assume the RREQ has traveled along a path  $FP = \{n_m, n_{m-1}, \dots, n_{k+1}\}$ , and is being forwarded by node  $n_{k+1}$  to  $n_k$ . The bandwidth can be calculated as follows:

As node  $n_{k+1}$  transmits a RREQ to its neighbor  $n_k$ , it appends the following information to the RREQ packet:  $\langle PB_{k+3}^{k+1}, PB_{k+2}^{k+1}, SRT_{k+1} \rangle$ . Node  $n_k$  calculates

$$\begin{aligned} LB_{k+1} &= SRT_{k+1} \cap SRR_k, \\ (PB_{k+3}^k, PB_{k+2}^k, PB_{k+1}^k) &= BW_3(PB_{k+3}^{k+1}, PB_{k+2}^{k+1}, LB_{k+1}). \end{aligned}$$

The reason that the calculation  $(PB_{k+3}^k, PB_{k+2}^k, PB_{k+1}^k)$  is done by node  $n_k$ , not  $n_{k+1}$ , is to allow node  $n_{k+1}$  to broadcast the same RREQ packet to all its neighbors. This reduces the computation and the bandwidth consumption, because otherwise node  $n_{k+1}$  needs to calculate the bandwidth  $(PB_{k+3}^k, PB_{k+2}^k, PB_{k+1}^k)$  for each of its neighbor and sends the RREQ packet individually. After calculating the bandwidth on the partial path from the source node to itself, node  $n_k$  will propagate the RREQ to its neighbors only if  $BW(FP^k) = |PB_{k+1}^k| \geq R$ . If the required bandwidth  $R$  cannot be satisfied on this path, the RREQ

---

<sup>7</sup>This is to assume that  $BA$  is used. If  $FA$  is used, the destination node chooses the best route and sends along it a RREP packet.



packet will be dropped at  $n_k$ . This way if a RREQ reaches the destination via a path  $P$ , there is sufficient bandwidth along this path. The destination node  $n_0$  responds by sending a RREP packet along the path  $P$  towards the source. As the RREP packet travels the path  $P$  in the reverse direction, transmission slots along the links can be reserved. On receiving RREP, node  $n_k$  calculates

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

It then passes the information  $(TS_{k+1}^P, TS_k^P)$  to its upstream neighbor  $n_{k+1}$  along with the RREP packet. When the RREP reaches the source, every link  $(n_k \rightarrow n_{k-1})$  on path  $P$  has chosen its transmission slots  $TS_k^P$  and a QoS path with bandwidth  $R$  has been setup. Because the transmission time slots  $TS_k^P$  are identified with the RREP packet, no RESV packet is necessary. The path is established in two phases, and the source node can start to transmit data packets to the destination when it receives the RREP.

Compared with the previous approach, the bandwidth is calculated when the AODV protocol is searching for a path from the source to the destination. If a path does not have enough bandwidth, a RREQ cannot reach the destination via this path. Therefore a path with insufficient bandwidth is never produced. Note that no soft-state is used in this approach. The reason is that the transmission set  $PB_{k+1}^k$  is calculated during the RREQ phase, when the RREQ packet is propagated throughout the network in a non-directional manner. Most of the RREQ packets are forwarded on paths not leading to the destination. If soft-state is used to protect the time slots  $PB_{k+1}^k$  during the RREQ phase, the flooding nature of the RREQ packet can freeze time slots over the entire network, not just those on a path from the source to the destinations. The consequence of not protecting the slots with soft-state is that the sets  $SRR_k, SRT_k$  may change before the RREP packet arrives, and the path may no longer have enough bandwidth to support the session. It is clear that the longer the path, the higher the probability that these slots may change during the interval between the RREQ packet and the RREP packet. As a consequence, the applicability of this scheme is limited in terms of the size of the network.

### 5.2.3 Maintenance of active paths

The transmission slots  $TS_k^P$  for each link on path  $P$  can be protected with soft-state. These soft-states are refreshed each time a packet is received and transmitted for the session, and expire when no packet arrives for a certain period of time. The slots  $TS_k^P$  are released when the associated soft-state expires. This prevents a session from locking up resources forever, and prevents time slots from being locked unnecessarily when a route breaks. If a source node does not have packet to transmit for sometime, but still needs to keep its route, it can transmit dummy packets and refresh the soft-states periodically. When node  $n_k$  on a path  $P$

finds the link to its downstream neighbor  $n_{k-1}$  broken, it can try to restore the downstream path locally by sending out a RREQ packet. If the path can be restored this way, the transmission can continue without the source node being aware of the link break.

If the path cannot be restored by node  $n_k$ , it can send a route error message (RERR) towards its upstream neighbors. The RERR invalidates the current path and releases the transmission slots  $TS_i^P$  between node  $n_k$  and the source  $n_m$ . For a node downstream of the failed link (between  $n_k$  to the destination  $n_0$ ), the transmission slots  $TS_i^P$  are released when no data packets arrive and the soft-state expires. When the source node receives the RERR packet, if it still requires a path to the destination, it can transmit a RREQ to start a new path discovery phase using the same mechanism as before. The new RREQ can carry the same *session\_id*. If this RREQ arrives at a node  $n_i$  downstream of the broken link before its  $TS_i^P$  expires, it serves both as a message to release the transmission slots  $TS_i^P$  associated with the old path and as a message to explore a new path. When the transmission is complete and the route is no longer needed, the source node can either transmit an explicit release packet along the path to release the time slots, or simply do nothing and let the soft-states on the links expire and release the slots automatically.

### 5.3 Discussions of the two approaches

The decoupled approach and the integrated approach each has its own advantages and disadvantages. In the decoupled approach, no attempt is made to find a route with sufficient bandwidth in the first place, and bandwidth is calculated only to check the acceptability of the route. Therefore the chance of finding the right route with the decoupled approach is less than that with the integrated approach, and it helps to use a routing protocol which provides multiple paths. The attraction of the decoupled approach is that no modification is required for the existing routing protocols, and the possibility to use soft-state to protect the time slots<sup>8</sup>. In the integrated approach, every RREQ arriving at the destination indicates a path with sufficient bandwidth, and it is more likely to provide an acceptable path than the decoupled approach. But the flooding-like nature of the RREQ packet makes it impossible to use soft-state to protect the transmission slots, and these slots are subject to change before the path can be setup. Neither approach is perfect, and their performances are likely to degrade when more nodes try to establish their QoS routes simultaneously. In both approaches it is possible that two simultaneous route setup requests will block each other, but if the two routes are setup one after another, at least one of them will be successful. This is because neither approach attempts to coordinate different route setup procedures. (This is not a problem for setup of best-effort routes, where bandwidth is not a constraint. The problem arises for QoS routing because of the potential

---

<sup>8</sup>Still, soft-state is not suitable for very long routes.

competition of the limited bandwidth resources). The performance of these two different routing approaches will be studied in the future.

## 6 Conclusions and future work

The problem of QoS routing in a TDMA-based ad hoc network is studied. We have shown that to calculate the maximal path bandwidth in a TDMA-based ad hoc network is NP-complete. An efficient bandwidth calculation scheme has been developed (with the FA version and the BA version), and its performance is studied with simulations. The bandwidth calculation scheme can be implemented in a distributed manner, and can be used in conjunction with the AODV protocol to perform QoS routing. A decoupled approach and an integrated approach are proposed, and their properties are discussed. The performance of these two approaches will be studied as part of the future work.

## 7 Appendix

This section provides the functions  $BW_1, BW_2$  and  $BW_3$  used by the bandwidth calculation scheme.

```

function (OUT) = BW1(IN, n)
    assert(n ≤ |IN|);
    choose the n elements with the lowest index in IN as OUT;
    (or choose the n elements in IN randomly)
    return.

```

```

function (OUT2, OUT1) = BW2(IN2, IN1)
    C = IN1 ∩ IN2;
    E1 = IN1 - C;
    E2 = IN2 - C;
    if |E2| ≥ |IN1|
        OUT2 = BW1(E2, |IN1|)
        OUT1 = IN1;
    return;
    elseif |E1| ≥ |IN2|
        OUT1 = BW1(E1, |IN2|);
        OUT2 = IN2;

```

```

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

```

*function*  $(OUT_3, OUT_2, OUT_1) = BW_3(IN_3, IN_2, IN_1)$

```

    assert( $|IN_3| = |IN_2|$  &  $IN_2 \cap IN_3 = \emptyset$ );

```

```

     $C_{21} = IN_2 \cap IN_1$ ;

```

```

     $C_{31} = IN_3 \cap IN_1$ ;

```

```

     $E_1 = IN_1 - C_{21} - C_{31}$ ;

```

```

     $E_2 = IN_2 - C_{21}$ ;

```

```

     $E_3 = IN_3 - C_{31}$ ;

```

```

    if  $|E_1| \geq |IN_2|$ 

```

```

         $OUT_1 = BW_1(E_1, |IN_2|)$ ;

```

```

         $OUT_2 = IN_2$ ;

```

```

         $OUT_3 = IN_3$ ;

```

```

        return;

```

```

    elseif  $|E_3| \geq |BW_2(IN_2, IN_1)|$ 

```

```

         $(OUT_2, OUT_1) = BW_2(IN_2, IN_1)$ ;

```

```

         $OUT_3 = BW_1(E_3, |OUT_1|)$ ;

```

```

        return;

```

```

    elseif  $|E_2| \geq |BW_2(IN_3, IN_1)|$ 

```

```

         $(OUT_3, OUT_1) = BW_2(IN_3, IN_1)$ 

```

```

         $OUT_2 = BW_1(E_2, |OUT_1|)$ ;

```

```

        return;

```

```

    else

```

```

         $T = \text{floor}(|IN_3 \cup IN_2 \cup IN_1|/3)$ 

```

```

         $C_{31}^3 = BW_1(C_{31}, T - |E_3|)$ ;

```

```

         $C_{31}^1 = C_{31} - C_{31}^3$ ;

```

$$C_{21}^2 = BW_1(C_{21}, T - |E_2|);$$

$$C_{21}^1 = C_{21} - C_{21}^2;$$

$$OUT_1 = BW_1(E_1 \cup C_{21}^1 \cup C_{31}^1, T);$$

$$OUT_2 = E_2 \cup C_{21}^2;$$

$$OUT_3 = E_3 \cup C_{31}^3;$$

*return.*

## References

- [1] Elizabeth M. Royer Charles Perkins and Samir R. Das. Ad Hoc On-Demand Distance Vector routing. In *Internet-Draft, draft-ietf-manet-aodv-06.txt*, Work in Progress, 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] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, April:46–55, 1999.
- [5] J. Tsai T. Chen and M. Gerla. Qos routing performance in multihop, multimedia, wireless networks. In *Proceedings of IEEE ICUPC'97*.
- [6] Y.-C. Hsu and T.-C. Tsai. Bandwidth routing in multihop packet radio environment. In *Proc. 3rd Int. Mobile Computing Workshop*, 1997.
- [7] A. Michail, W. Chen and A. Ephremides. Distributed routing and resource allocation for connection-oriented traffic in ad-hoc wireless networks. Proc. of the Conference on Information Sciences and Systems (CISS-98), March 1998.
- [8] A. Michail and A. Ephremides. A distributed routing algorithm for supporting connection-oriented service in wireless networks with time-varying connectivity. Proceedings of the 3rd IEEE International Symposium on Communications and Control, June 1998.
- [9] S-B Lee and A. T. Campbell. INSIGNIA: In-band signalling support for QoS in mobile ad hoc networks. Proceedings of the 5th Intl. Workshop on Mobile Multimedia Communication, 1998.

- [10] S. Chen and K. Nahrstedt. Distributed Quality-of-Service in Ad-Hoc Networks. *IEEE J. Sel. Areas Commun.*, SAC-17(8), 1999.
- [11] 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.
- [12] D. H. Cansever, A. M. Michelson and A. H. Levesque. Quality of Service Support in Mobile ad-hoc IP Networks. In *Proc. MILCOM*, 1999.
- [13] 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, 2000.
- [14] Using the CPLEX Callable Library. CPLEX Optimization, Inc., Version 4.0.
- [15] <http://alpha.ece.ucsb.edu/~eroyer/aodv.html>.