QoS Support in Mobile Ad-hoc Networks

Apurv Gupta and Dheeraj Sanghi
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
email: dheeraj@iitk.ac.in

ABSTRACT

In this paper, we present a framework for providing guaranteed Quality of Service in ad-hoc networks given reasonable limits on rate of change of topology. This is achieved at the expense of higher network resources than required by the flow specification itself. We quantify these extra resources and suggest changes that need to be incorporated in the buffer requirements and the play out times. Central to providing QoS guarantees is the concept of Local route repair and buffer flush. The onus for restoring a failed in-use path is on the upstream node of the failed link. The route restoration time after a failure has important consequences on the performance of the proposed scheme and so we propose to add this functionality to the QoS module in general. We also suggest a protocol to reduce this latency under the assumption of slow rate of change of topology. The analysis can be extended to slightly relaxed assumptions on rate of change of topology. But when the topology changes too fast, it shall not be possible for any protocol with reasonable overhead to provide QoS guarantees.

INTRODUCTION

In mobile ad-hoc networks (MANETs), the topology may vary with time owing to the movement of nodes and lack of an infrastructure network. Most of the work till date has concentrated on efficient light weight routing protocols for MANETs [1, 2, 4, 6, 7, 9, 10, 11]. Since the path may change many times in the lifetime of a single session, it seems difficult to provide QoS guarantees in such an environment. We however argue that given reasonable limits on the rate of change of topology, QoS guarantees can be maintained with a reasonable certainty.

If the topology is changing so fast that a path may become invalid in the duration of a few round trip times, then it seems impossible to provide QoS guarantees without having a reservation for each flow on almost all nodes and using some flooding like algorithm for routing.

In real life adhoc networks, it is not unrealistic to assume a limit on the rate of change of topology. When, for example, an adhoc network is set up for some civilian communication system, one person may move at a time. It is unlikely, that many people change places at the same time. Consider the scenario where an adhoc network is set up between the laptops of students across several hostels. Here the rate of change of topology is restricted. Such an assumption allows us to build efficient mechanisms for providing QoS guarantees. QoS guarantees are maintained at the expense of utilization of more network resources than the flow would have otherwise required.

Some work has been done in this area by Lee and Campbell [8]. An Inband signalling is used as a light weight protocol for resource reservation. The concept of QoS module is used that is responsible for QoS routing, flow establishment and flow maintenance. Our approach can be considered an extension of this work. There are, however, differences in flow re-establishment mechanisms and we focus on how higher reservations can be used to support QoS.

MODEL

QoS support would typically be required for multimedia applications in MANETs like voice communications. We briefly explain the model here.

Mobile adhoc networks comprise of a number of mobile nodes connected by wireless links forming arbitrary time varying topologies [3]. Mobile nodes function as nodes as well as routers. They support packet store-and-forward services to neighbbring nodes. In order to support real time services, all intermediate nodes co-operate to support bandwidth requirement and a bound on the total delay.

We assume that there is a mechanism creating these flow states and maintaining them. QoS routing is also required and this can be accomplished by the routing module [5] or by a separate QoS module in conjunction with the routing module [8]. A light-weight resource reservation protocol is also required. We favour piggybacking of the reservation protocol information with the data packets where ever possible to reduce overhead.

0-7803-5893-7/00/$10.00 © 2000 IEEE

The flow states established must be soft state because of its ability to withstand failures and route changes. The soft state timer is refreshed at receipt of every packet. When the timer expires, the state is deleted. Fresh states are created on receipt of a packet for a flow not already having a reservation.

A key concept in providing QoS is local route repair. When a link of an in-use path fails, the node detecting the failure must take local action to repair the path locally instead of generating an error message back to the source.

We assume that the topology is changing slowly enough for QoS guarantees to be maintained. By this we mean that no two links of a path fail at the same time, and the time between two successive failures on the same path is greater than some limit (time for route repair and buffer flush, as described later).

APPROACH

The lightweight mechanisms developed in [5, 8] address some of the problems in providing QoS guarantees. However, some issues still need to be addressed.

The difficulty in providing QoS comes mainly from link failure of an in-use path for a flow. This is a much more frequent phenomenon in adhoc networks than in other kinds of networks.

This failure is detected by the upstream node of the failed link. Link failure information can be had from some underlying protocol. The IEEE 802.11 link layer protocol supports this operation, for instance.

It is not advisable that the error message for this failure be propagated towards the source. Instead a local route repair must be undertaken by the node that detects route failure. This node is expected to buffer the packets it has received in the meanwhile. Dropping these packets may result in unacceptable degradation of the QoS of the application.

Let τ_f denote the time taken for link breakage detection and connection reestablishment (the route repair time). Also let (PCR, SCR, MBS) denote the traffic specification of the flow. This tuple denotes the peak cell rate (PCR), sustainable cell rate (SCR) and the maximum burst size (MBS).

Once the flow is re-established, normal transmission can be resumed. But this gives rise to two potential problems. First, the play out buffer at the receiver may become empty. Second, the buffered packets at the node need to be dealt with.

The first problem is addressed by adjusting the initial play out latency.

$$L_{playout} = L_{Jitter} + \tau_f, \quad (1)$$

where L_{Jitter} is the jitter latency.

This approach has a subtle problem. If there are more than one failures during a session on the same path (another failure while the effects of the first failure are still being handled) then the play out latency has to be adjusted to

$$L_{playout} = L_{Jitter} + k * \tau_f, \quad (2)$$

where k is the maximum number of failures that can occur in close vicinity of each other during the session. This much of an initial delay may be unacceptable in case of interactive multimedia applications.

To address the second problem, our strategy will be to transmit the buffered packets at a higher rate R' compared with the normal rate R, as soon as the route-repair is accomplished. R' should be fast enough that we clear our buffers before the next failure occurs. The transmission at rate R' takes place till the last packet accumulated before route restoration has been transmitted.

We discuss the choice of R' later. We term this operation as *buffer flush*. The time taken for emptying this accumulated buffer x is given by

$$x = \frac{\tau_f}{\frac{R'}{R} - 1}. \quad (3)$$

An additional buffer requirement of $\tau_f * R$ will ensure that during the route repair period, packets are not lost.

Traffic Profile

Result of the buffer flush is such that the traffic profile through a link could be different from that sent by the sender, not only due to jitter introduced, but also due to link failure and restoration.

Based on the values chosen for τ_f and R', etc., we need to find a way to map source traffic description to real traffic profile through a link. The resource reservation and admission control will be done based on this profile. Also when the adhoc network borders an infrastructure network, this mapping may be applied at the ingress node.

The mapping is simple for a CBR source. The rate of transmission, which would have been R, drops to zero for

τ_f time and then is raised to R' for time x. Thereafter, the rate resorts back to R. $R' * x$ is equal to $R * (x + \tau_f)$. This can be thought of as a VBR source with

$$(PCR, SCR, MBS) = (R', R, R * \frac{\tau_f}{\frac{R'}{R} - 1}) \quad (4)$$

Similarly, a VBR source with (PCR, SCR, MBS) as traffic specification will be mapped to (PCR', SCR', MBS').

Clearly $PCR' = R'$ and $SCR' = SCR$. To find MBS', the worst case scenario occurs when a burst arrives just after the link fails. The incoming rate cannot be larger than PCR. Hence the time to empty the buffer is simply $x \leq \frac{\tau_f}{\frac{R'}{PCR} - 1}$ and therefore $MBS' \leq \frac{\tau_f}{\frac{R'}{PCR} - 1} * R'$. Therefore we get

$$(PCR, SCR, MBS) = (R', SCR, R' * \frac{\tau_f}{\frac{R'}{PCR} - 1}) \quad (5)$$

A New Route Restoration Mechanism

As is clear from the discussion in the previous sections, the the value of τ_f has important consequences on the performance of the proposed scheme. Hence it is of importance to reduce this value. Some protocols support the operation of local route repair and some do not.

One of the methods could be to query a route to the destination. This however has its drawbacks. This may take much more time if the protocol requires that only the destination generate a reply. The newly computed route may be entirely or significantly different from the previous path. This leads to waste of network resources and also reduces the probability of successful reservations.

We would like to reduce the latency of restoration and also the new route must have most nodes in common with the older path. For this purpose we design a new algorithm, which works independently of the underlying routing protocol.

The idea is to find a path NOT to the destination, but to ANY of the few downstream nodes of the path. From then on the connection is routed on the older path. Several modifications to the resource reservation protocol need to be made in order to enable this scheme to work.

Each node must be aware of a fixed number, say n, of the downstream hops in the path. Number n can be a design parameter, but value of 3 or 4 will suffice. On a route failure, a special Route Repair Request packet (RRReq) is sent to a limited area. This area is limited in terms of number of hops and the model from the RDMAR protocol may be used to compute the number of hops (the TTL field). The purpose of this packet is to request a path to ANY of the nodes listed there. Other nodes send a Route Repair Reply (RRRep) packet listing a path. The best path along these is used.

As an illustration, consider the case shown in Figure 1. Node D detects a link failure and requests a path to either G or I. It gets a reply from F and a new path is then set up.

The information about the next few downstream hops is furnished by a periodic reverse routed packet from the receiver. On a route repair operation, node D must also request a fresh reverse route packet (RevR) from the receiver. The purpose of this is to furnish the modified information to each intermediate node. The propagation of RevR packet is stopped as soon as an upstream node beyond D finds it to be the same as the information it previously had. One may note that maintaining such information enables route optimizations.

Another observation is that there may be temporary loops in the path selected by our protocol. These loops are removed as soon as the buffer flush is over. The onus for doing this is on the node (say C) where the stream enters the loop. For example, in Figure 2, node C will remove the loop.

The node C must not forward the packets on the next hop in the loop, but must buffer them and apply a buffer flush once the buffer flush of A is over. This cannot possibly result in a buffer overflow at node C as the buffer flush of A is complete in time $\tau_f * \frac{R'}{R}$ and the buffer accumulation at C is $\tau_f * \frac{R}{R'} * R$ which is less than B. (since $R' > R$).

Another fact to be considered is that of limit on the maximum number of times a route can fail. The sender must be informed of a local route repair and the number of additional hops introduced in the path. This responsibility is also on node that detects route failure. The sender then knows that when the play out buffer to handle route failures becomes insufficient. At that instant, it must try to find a new path which may probably have fewer nodes.

A local route repair may not be effective if there are very few nodes in vicinity of node detecting failure. Under such conditions, the route repair fails abruptly. But we expect such cases to be very rare. Such cases can be reduced with the help of cached routes. If we permit nodes to supply RRRep packets based on their cached routes, then the probability of route restoration failure is reduced. However, there arises a potential problem of looping. This can be avoided by nodes sending a reply only if the route is different from the path being repaired.

DISCUSSION

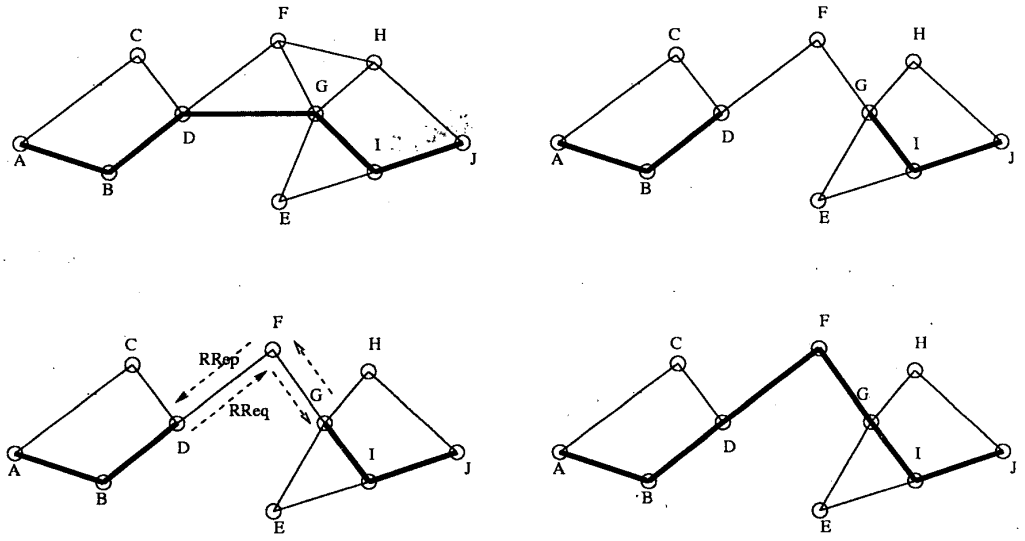

Figure 1: Route Restoration

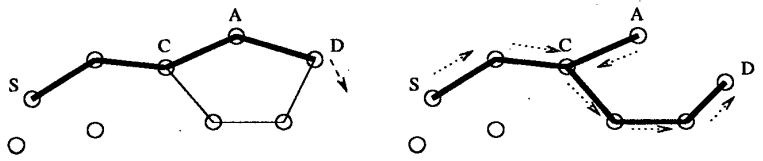

Figure 2: Loop in Repaired Path

The choice of rate R' is a design parameter that affects the performance of our scheme. A large value reduces the time for buffer flush, thereby increasing the probability of correctness of our assumptions regarding the slow rate of change of topology. It would also reduce the buffer requirements and initial play out latency. On the other hand, a larger R' also means a larger reservation of resources, leading to poorer utilization of network resources. R' must be so chosen as to complete buffer flush before another route failure, if any estimate is available of this time. R' may be chosen as $\beta * R$ for simplicity, where β is a constant.

Our approach will work well if the following conditions are satisfied. First, at any time, the number of buffer flushes that have not completely reached the destination node must not exceed a reasonable limit, k. (Our analysis is only for k equal to one, but small values of k can be tolerated.) Second, the local route repair succeeds within the estimated time bounds. In this case, the packet loss ratio is expected to be very low and the timely play out of the data packets can be ensured.

It may be noted that our guarantees are probabilistic and subject to our assumptions. If the assumptions are violated, e.g., if a network partition takes place, no QoS may be supported. This can't be avoided even in wired networks.

The mechanism of buffer flush increases the probability of successful delivery compared to the best effort service. The limits on the buffer sizes serve the purpose of packet dropping whenever the conditions are violated. In the extreme case when a route fails in time less than τ_f, we can't really repair a route and there can be no QoS guarantees in this case.

CONCLUSION

In this paper, we have presented a scheme to support QoS guarantees in mobile adhoc networks. Our scheme works well if the rate of topology is slow enough to permit a route repair and buffer flush to be undertaken before another link fails in the path. We have suggested a different route repair algorithm. This is coupled with computation of effective traffic profile, and reserving resources based on this new profile. This results in greater than necessary reservation. That is the cost to be paid for having QoS in the adhoc networks. If the rate of change of topology is very fast, we argued that it doesn't make sense to try to provide QoS guarantees. The applications must live with best effort service and would necessarily have to be adaptive. However, our concept of a buffer flush is still applicable and would improve the performance of the system. Some of the proposed mechanisms need to be checked for effectiveness using a simulation before being adopted.

REFERENCES

[1] G. Aggelou and R. Tafazolli. RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks. In *Proc. of The Second ACM Int'l Workshop on Wireless Mobile Multimedia*, August 1999.

[2] J. Broch, D. B. Johnson, and D. A. Maltz. The dynamic source routing protocol for mobile adhoc networks. Mobile Computing, 1996.

[3] M. S. Corson and A. T. Campbell. Towards supporting quality of service in mobile adhoc networks. In *Proc. of 1st Conf. on Open Architechture and Network Programming*, April 1998.

[4] J. J. Garcia-Luna-Aceves, M. Spohn, and D. Beyer. Source-tree routing in wireless networks. In *Proc. of the Seventh Annual Int'l Conf. on Network Protocols*, 1998.

[5] M. Gerla and J. T. Tsai. Multicluster, mobile, multimedia radio networks. *Wireless Networks*, 1(3), 1995.

[6] Z. J. Haas and M. R. Pearlman. The zone routing protocol for highly reconfigurable ad-hoc networks. In *Proc. of ACM SIGCOMM Conf.*, August 1998.

[7] M. Jiang, J. Li, and Y. C. Tay. Cluster based routing protocol (CBRP). draft-ietf-manet-cbrp-spec-01.txt, August 1999. Work in progress.

[8] Seoung-Bum Lee and A. T. Campbell. INSIGNIA: Inband signalling support for QoS support in mobile adhoc networks. In *Proc. of 5th Intl. Wkshop on Mobile Multimedia Communication*, October 1998.

[9] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. of IEEE INFOCOM Conf.*, April 1997.

[10] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers. In *Proc. of ACM SIGCOMM Conf.*, 1994.

[11] C. E. Perkins, E. M. Royer, and S. Das. Adhoc on-demand distance vector (AODV) routing. draft-ietf-manet-aodv-06.txt, July 2000. Work in progress.