Supporting QoS Traffic at the Network Layer in Multi-hop Wireless Mobile Networks

Gokce Gorbil
Imperial College London
Dept. of Electrical and Electronic Engineering
Intelligent Systems and Networks Group
London, UK
Email: g.gorbil@imperial.ac.uk

Ibrahim Korpeoglu
Bilkent University
Dept. of Computer Engineering
Ankara, Turkey
Email: korpe@cs.bilkent.edu.tr

***Abstract*—Supporting real-time and quality-of-service (QoS) traffic in multi-hop wireless mobile networks is challenging due to the high level of dynamism involved. In this paper, we propose a network layer solution in the form of a hybrid routing protocol to enable QoS traffic support in this class of networks. Our proposed protocol combines link state topology updates, source routing and on demand link cost dissemination to concurrently support multiple classes of QoS and normal flows. Our protocol provides for QoS traffic by intelligent path selection at the source nodes based on the required QoS parameters/levels and dynamically adapting the paths as network topology and conditions change. This solution does not require any other layers/components in the network stack to be QoS-aware and is therefore readily deployable over existing networks. We present experimental results from a simulation study on the performance of our protocol. Our results show that the proposed solution can provide efficient QoS traffic support in small-to-medium sized mobile networks, where up to 90% improvement in QoS metrics are observed in certain experiments.**

***Index Terms*—multi-hop wireless mobile ad hoc networks; quality-of-service; real-time traffic; routing protocol; link state; on demand**

I. Introduction

Wireless mobile networks have proliferated and seen widespread real-life deployment as wireless NICs with ad hoc networking support have become commonplace. This proliferation has been accompanied by more stringent demands from users and applications, including support for real-time and QoS traffic such as VoIP and multimedia streaming. However, QoS support in wireless networks has been challenging due to the dynamic environment. This is especially apparent in multi-hop networks: recent experiments with real-time applications over MANETs have shown deficiencies in their capabilities to support real-time traffic [1].

In this paper, we propose a network layer solution in the form of an intelligent and adaptive routing protocol to concurrently support multiple classes of QoS traffic over multi-hop wireless mobile networks. This solution does not require any other layers/components in the network stack to be QoS-aware. The advantages of this approach are decreased complexity, and ease of configuration and deployment on current architecture. The disadvantages are possibly weaker QoS support since the solution is not integrated in the protocol stack and providing only *soft QoS guarantees*. Despite these disadvantages, our protocol shows promising improvements in QoS in our experimental study.

Our proposed routing protocol is a hybrid protocol using event-based link state message dissemination to enable routing of non-QoS traffic and for the dissemination of topology information, and an on demand directed cost dissemination mechanism to enable routing of QoS traffic. QoS traffic is supported via intelligent path selection at the source node and dynamic path adaptation as the network topology and conditions change. The details of our protocol is further explained in Sec. III while Sec. IV provides experimental results from a simulation study on the performance of the proposed protocol.

II. Related Work

There is extensive research on providing QoS in wireless networks; we provide a brief overview of some of the more related works here. QOLSR [2] provides QoS extensions to the OLSR protocol [3]. It enables multi-metric QoS routing via multi-paths using a link state mechanism; multi-point relays are employed to reduce protocol overhead [4]. Similar methods to make link-state routing scale for ad hoc networks are presented in [5]. QOLSR work is continued in [6] by taking into account the interferences caused by a flow on its own path during routing decisions. MP-DSR [7] is a QoS-aware multi-path extension to DSR [8]. It uses end-to-end reliability (e2eRel) as its QoS metric, where e2eRel is calculated from link availabilities. Similarly, AODVM [9] is a multi-path extension to the AODV protocol [10] adopting end-to-end reliability as its QoS metric. Unlike MP-DSR, however, AODVM is proposed for heterogeneous ad hoc networks where some nodes are expected to be more reliable than others. Chen et al. propose an on-demand, link-state, multi-path QoS routing protocol for MANETs in [11]. They aim to satisfy bandwidth requirements of flows by using multiple paths between source and destination. Their proposed protocol reactively collects

978-1-4577-9538-2/11/$26.00 ©2011 IEEE

link-state information from source to destination in order to construct a partial topology at the destination, which chooses multiple paths from the source to itself that can collectively satisfy the bandwidth requirement. Other related work on QoS routing in wireless mobile networks include [12]–[15].

III. Protocol Specification

We propose a network layer solution in the form of a multi-hop hybrid routing protocol named *Elessar*[1] in order to enable real-time applications and QoS traffic over wireless mobile ad hoc networks. We do not assume any specific medium access control (MAC) scheme so the routing protocol can be used with any MAC protocol, including the IEEE 802.11x family. We do not require resource reservation but Elessar can easily accommodate a reservation mechanism at the network layer for reserving resources along least-cost paths found by its route discovery mechanism.

Elessar combines link-state message dissemination with an on demand link cost mechanism, where the on demand mechanism is activated when there are QoS flows in the network. This hybrid approach is a mix of proactive and reactive routing, where the proactive component serves to quickly find shortest paths for normal flows and to react to changes in the network, and the reactive component enables smart routing of QoS flows based on additional information received (i.e. network information on QoS metrics such as delay, loss rate, available bandwidth, etc.) while incurring a lower overhead. Our proposed solution can support normal and different types of QoS traffic concurrently. Elessar is self-aware as it proactively monitors the network and its own performance (i.e. the paths) and reacts to changing conditions in the network by selecting better/more suitable paths. The protocol consists of the components described below.

A. Neighborhood Beaconing

Neighborhood beaconing enables a wireless node in the network to learn of its one-hop neighbors through periodic locally broadcast *beacons*. A node which receives a beacon sends back an acknowledgment (ACK). Each node maintains a neighbor list, where each entry is time-stamped by the last received update. These time-stamps are used in automated removal of nodes from the neighbor list due to node mobility and failure (i.e. a disgraceful leave). Neighborhood beaconing also handles the join and departure procedures of nodes that want to join/leave the network through the use of explicit *node join* and *leave* messages sent by the joining/departing node. Such explicit messages, limited with a hop count of 1, enable the exchange of control information such as bootstrap data.

Monitoring the network status via active probing of links to one-hop neighbors is also part of the neighborhood beaconing mechanism. Each beacon serves as a link probe, both updating the liveness of the responding neighbors and serving to measure the round-trip times (RTTs) and packet loss rates on the links. From these observations, other QoS metrics can be derived, such as delay jitter and available bandwidth. Link costs are kept as exponentially weighted moving-point averages at each node: $c_i = \alpha \cdot v_i + (1 - \alpha) \cdot c_{i-1}$, where c_i is the average cost after probe i and v_i is the measured cost value by probe i. We have chosen $\alpha = 0.75$ based on initial experiments so nodes remember their past but give priority to recent events. Note that if the underlying MAC protocol provides a similar periodic beacon service, then Elessar may decrease the number of beacon messages to reduce related overhead.

B. Topology Dissemination

In Elessar, topology dissemination is achieved via network-wide link state (LS) messages. Instead of the more usual periodic LS updates, this mechanism operates in an *event-driven* manner where neighborhood changes of nodes constitute the events of interest. We aim to achieve a balance between LS overhead and accuracy of topology representations at network nodes with this approach. Each network node maintains its own local view of the network, which is used in route discovery for normal and QoS traffic. These local representations are maintained by the LS updates, and delays and losses during topology dissemination cause intermittent inconsistencies in these local representations, which inadvertently lead to errors in path finding and the maintenance of the shortest or least-cost paths for flows. Note that during the network-wide dissemination of LS messages, Elessar may utilize optimized methods provided by the underlying MAC layer; if such mechanisms are not available, then efficient variants of flooding or gossiping may be employed (e.g. [16]–[18]).

Each LS message contains information (e.g. the IDs) on the sender's one-hop neighbors; link cost information is not included in LS messages for several reasons: (i) This allows us to reduce overhead by keeping network probing inactive if neighborhood information is provided by the MAC layer or use a high beacon period (e.g. 3 sec.); (ii) Not including link costs lowers LS message size; (iii) Link costs are not needed in normal operation mode.

Elessar utilizes two approaches to further decrease LS overhead: (i) incremental LS messages, and (ii) reverting to a periodic scheme when the event rate is high. A normal (whole) LS message contains information on all of the sender's neighbors whereas an incremental message contains information regarding those changes that have occurred since the last update. Although the incremental scheme lowers LS overhead, it may increase inaccuracy in local topology representations due to message delivery failures as LS messages are sent unreliably. In order to strike a balance, Elessar combines incremental and normal link-state messages: after every K incremental LS message, a node sends a normal LS message; the protocol parameter K controls the balance between representation accuracy and LS overhead, with increasing values of K decreasing overhead and increasing inaccuracy.

When mobility rates in the network are very high, many LS messages will be created due to the high event rate. In these

[1]The name is derived from the pronunciation of the abbreviation LSR, standing for link-state routing.

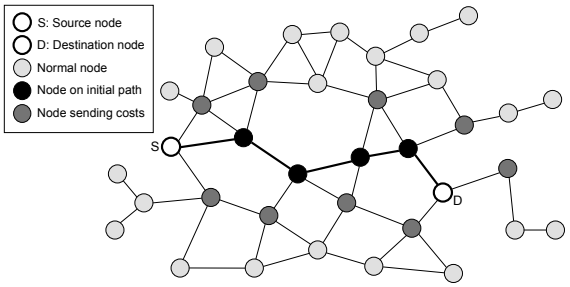


Fig. 1. Directed cost dissemination, hop limit = 1.

cases, Elessar reverts to periodic LS messages: this decision is local to each node and is based on the node's event (and LS) creation rate R, measured in *msg/min*. When R is greater than a threshold τ, a node would switch to periodic LS with a set period T; when $R < \tau$, it would switch back to event-based LS. We provide values for τ and T in Sec. IV.

C. Directed Cost Dissemination

Directed cost dissemination (DCD), together with topology dissemination, are the mechanisms that enable intelligent path selection to support QoS traffic. While a node s learns the current status of the network topology via LS updates, DCD informs s of the *link costs* with the appropriate metric(s) for the QoS flows originating from s. A node activates DCD only when it has QoS flows since normal traffic is routed based on hop count. The activation of DCD only for the duration of QoS flows aids in using network bandwidth more efficiently.

The DCD mechanism operates as follows: when a node s has QoS traffic to send to a destination node d, it immediately starts sending the packets over an initial path P that may be non-optimal, therefore decreasing the route acquisition delay; P is found via normal route discovery. s also initiates the DCD mechanism by sending a message m reliably to d over P. This message informs nodes that it visits about the new QoS traffic and starts a limited flooding controlled by a small hop limit h_m along P. A node n which receives m first forwards m to the next hop along P and starts sending its link costs (with the appropriate QoS metric(s)) *directly* to s periodically using a path found by normal route discovery. If n does not currently have information about its link costs, then it also starts the necessary network probing mechanism. n also decreases h_m and if $h_m \geq 0$, it sends m to all its neighbors.

An example of this procedure with $h_m = 1$ is shown in Fig. 1. In this figure, all black and dark-grey nodes have been informed of the QoS flow from s to d and are now sending their link costs to s. Periodic reception of link costs from a portion of the network, in addition to topological updates received via the LS mechanism, enables s to select a more suitable path for the flow. When the QoS flow terminates, s sends an explicit stop message to inform all cost sending nodes to stop DCD.

m contains the source and destination node IDs, expected length/duration of the QoS flow (if known), the required QoS metric(s) for the flow, the period for the cost messages (p_c), and the hop limit h_m, which controls the extent of the limited flooding that m initiates. Therefore, h_m greatly influences protocol overhead due to DCD by controlling the number of nodes which send their costs to s. It should be noted that the potential of finding a better path from s to d decreases by decreasing h_m. We investigate the influence of h_m in Sec. IV.

D. Route Discovery

Elessar uses *source routing* for the routing of all packets. A path for a packet or flow is found by running a shortest path finding algorithm at the source node s using its local representation of the network and this source route is then embedded into the header of each packet of the flow. Intermediate nodes forward a packet along its route by examining the source route in the header and sending the packet to its next hop, which is a neighbor of the current node. If an error occurs during this process (e.g. current node is no longer connected to the next hop in the packet's route), route maintenance is activated to repair the broken route. Source routing is employed since it places minimal burden on intermediate nodes for routing and allows routing to be trivially loop-free unless routes are broken.

Elessar employs variants[2] of Dijkstra's shortest path algorithm as the local path finding algorithm. For normal traffic, link costs are not required and the algorithm finds the shortest (i.e. minimum-hop) route from s to d; for QoS traffic, link costs are used and a least-cost path is found. If there is not enough information (i.e. link costs) to find a least-cost path, then the shortest path is used.

A *route cache* is employed to decrease computational load for route discovery. Route cache entries are valid as long as the topology when they were discovered remains the same. Therefore, all entries in the route cache are flushed when an LS update is received or a routing error occurs that activates route maintenance. In order to keep processing overhead low at s, each cost message does not trigger a new route discovery. Instead, the route cache is emptied periodically, which forces route discovery for all traffic flows originating from s. Since route discovery operates on the updated local representation of the network, the algorithm returns the most optimal route at the time.

E. Route Maintenance

Route maintenance repairs broken routes and informs the source node s of a routing failure so it can find a different path. Route maintenance is initiated when an intermediate node i encounters a problem in forwarding a packet towards its destination d using source routing. The unavailability of a next-hop node is discovered via the neighborhood beaconing mechanism and causes a link-state update, which will eventually reach s and trigger the discovery of a new path, fixing the broken route. Since this may take some time, i finds an alternate route (using local route discovery) from itself to d

[2]We have modified the original algorithm so it can handle non-additive costs (e.g. loss rate) and find the *maximum* cost path (e.g. for finding the path with the greatest available bandwidth).

and forwards packets using this patched route to reduce the number of lost messages. If i cannot find such a route, then all messages destined to d are dropped at i (i.e. they are lost). In this case, i sends an explicit *route failure* message to s to facilitate quicker recovery; this message follows the *reverse* path from s to i to inform upstream nodes of the problem and tell them to stop forwarding packets belonging to the broken flow until a new path is provided.

If the broken route was used for a QoS flow, then the patched route is no longer guaranteed to satisfy the QoS requirements of the flow, since node i can only find a minimum-hop path from itself to d as it does not receive the link costs. Therefore, for QoS flows which are broken, regardless of whether node i is able to repair the route or not, an explicit *route failure* message is generated and sent to s. For QoS flows, the only node that can effectively repair a broken route is s due to its reception of cost messages. The route failure message in this case causes upstream nodes to stop forwarding packets only if i was not able to repair the path.

IV. Experimental Results on Protocol Performance

We have implemented Elessar in the OMNeT++ discrete-event network simulator [19]. All layers of the ISO protocol stack (including physical and link layers) were simulated for a more realistic evaluation; the IEEE 802.11b MAC protocol is employed at the link layer and parameters for the wireless NICs are set using common settings across commercially available NICs. The signal attenuation threshold is -70 dBm, path loss coefficient is 2.5, MAC bitrate is 11 Mbps, transmit power is 20 mW, radio SNR threshold is -40 dBm, and radio sensitivity is -85 dBm. With the physical radio properties and wireless channel characteristics used in the simulations, each node has an effective communication range around 50m.

We use τ = 20 *msg/min* and T = 3 *sec* in the topology dissemination mechanism; these values are an educated guess from experience with preliminary experiments. While these values may seem high, it should be noted that source nodes rely on LS messages to have an up-to-date view of the network, which is critical for route finding performance. Therefore, although high values of T (and conversely low values of τ) lower LS overhead, they affect routing performance adversely. A similar effect can be observed due to K, which is discussed below. Since T and τ provide an upper bound on LS overhead, they do not affect protocol performance as much as other parameters such as K or h_m.

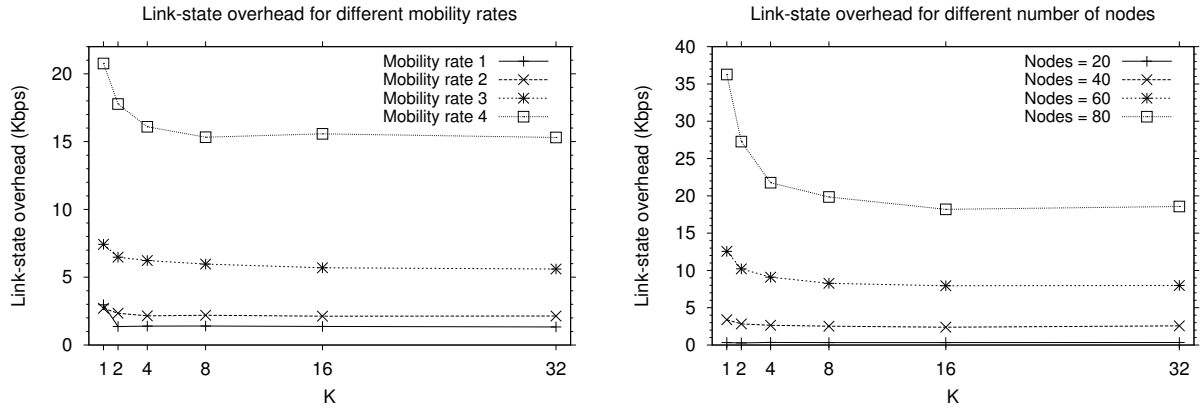
For simulations on normal (QoS) mode performance, a total of 15 (25) random traffic flows are created and maintained in the network. Each flow is between a random source-destination pair, with characteristics (e.g. packet creation rate, duration) chosen from uniform distributions based on parameters given in Tab. I. When a flow terminates, a new random flow is created, so the number of flows in the network is maintained. For QoS simulations, all traffic flows are of the same QoS type. Each experiment is executed for a simulation time of 11 minutes, where the first minute is reserved for initial set-up of the experiment. Each data point in the provided results is an average of 30 simulation runs unless stated otherwise. 95% confidence intervals are provided where appropriate. The random waypoint mobility model is employed to simulate node mobility with different rates as given in Tab. I. A routing protocol is expected to perform well in all circumstances, even when users may behave erratically. Although random waypoint is not a realistic movement pattern for most mobile entities, it provides a valuable benchmark of the protocol in unusual cases. Mobility rates 3 and 4 are especially high in order to observe performance in very dynamic networks. Note that with such high mobility, we would expect some of the nodes to switch to the periodic LS scheme.

Parameter	Value (normal)	Value (QoS)
No. of flows	15	25
Packet payload length	[40,600] bytes	[40,600] bytes
Packet creation rate	[20,200] pkts/sec	[20,200] pkts/sec
Flow duration	[2,200] sec	[5,200] sec
Mobility Rate	**Node Speed**	**Waiting Time**
Mobility Rate 1	[1,2] m/sec	[5,10] sec
Mobility Rate 2	[2,4] m/sec	[5,10] sec
Mobility Rate 3	[5,10] m/sec	[0,5] sec
Mobility Rate 4	[10,20] m/sec	[0,0] sec

TABLE I
Traffic flow parameters and mobility rates used in the simulations.

Figure 2 shows how incremental messaging can decrease link-state overhead under various network conditions. Figure 2a presents results with different mobility rates for a network of 40 nodes in a 200x200 m^2 area. We fix the mobility rate (to 2) and the area size and vary the number of nodes from 20 to 80 in Fig. 2b. We can observe that the incremental messaging mechanism is able to lower LS overhead significantly, especially in dense networks or when mobility rates are high. The ratio of the LS overhead to generated data packets in Fig. 2 range from 0.5% in the lowest to 13% in the highest.

Figure 3 presents the effects of the protocol parameter K on routing performance for normal traffic under a 200x200 m^2 area with changing mobility rates and number of nodes; each data point is an average of 60 simulation runs. It is observed that increasing values of K decrease routing performance as shown by the decreasing ratio of successful route lookups and the increasing ratio of lost data packets. Since LS updates are sent unreliably and are subject to errors and losses, lost LS messages when we use a high K value increase the probability and duration of inconsistencies between the current network and local representations at nodes. Such inconsistencies eventually lead to false negatives during route lookups (i.e. not being able to find a route even though one exists) and to the loss of data packets (either at the sources due to route lookup failure or at intermediate nodes when broken routes cannot be repaired). Considering its effects on routing performance as well as LS overhead, setting K to a value between 4 and 8 seems to be the best choice for the considered network settings.



(a) Mobility rates (b) Number of nodes

Fig. 2. Effect of K on link-state overhead, for different mobility rates and number of nodes.

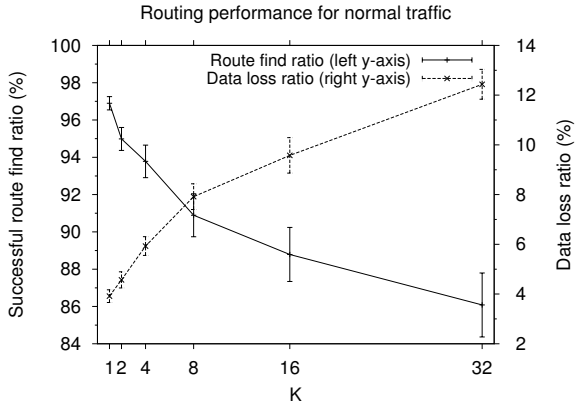


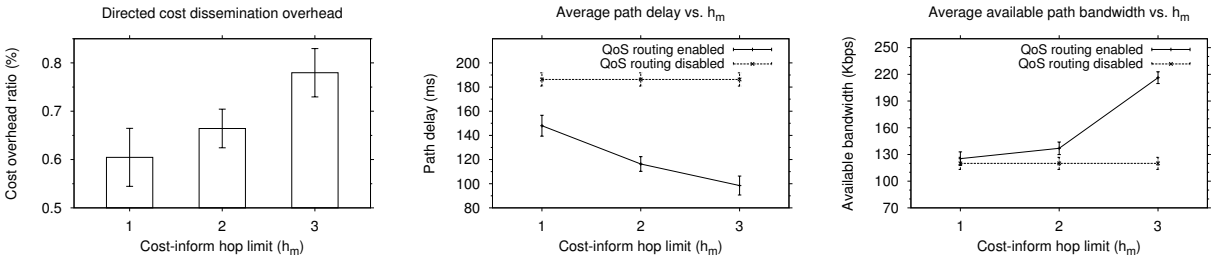
Fig. 3. Route finding performance and data loss ratio for different values of K.

Figures 4a and 5a present the ratio of cost messages to generated data packets under different h_m and p_c values. These experiments involve a network of 40 nodes in a 200x200 m^2 area, mobility rate = 2 and p_c = 2 sec (for Fig. 4a) and h_m = 2 (for Fig. 5a). Interesting to note is that while h_m for the initial limited flooding affects cost overhead, its effects are much more subdued compared with the effect of the cost period p_c. This is due to the fact that the influence of h_m depends on the network topology and population density whereas p_c's effect is more general and does not depend on population density. While the overhead due to DCD is quite low (below 1.3%), it constitutes between 8% to 24% of the total protocol overhead (cost and LS) and therefore the utility of controlling DCD overhead via careful selection of h_m and p_c should not be underestimated.

As discussed in Sec. III, p_c and h_m affect routing performance for QoS traffic: this is depicted in Fig.s 4b and 5b, which show the average end-to-end round-trip path delays for different values of h_m and p_c. They have the same simulation configuration as Fig.s 4a and 5a. These figures also present a comparison between QoS routing and non-QoS routing. p_c and h_m do not affect non-QoS routing since the DCD mechanism is disabled, so we observe a constant non-QoS performance with varying p_c and h_m. It can be seen that in QoS routing, DCD together with intelligent route selection can decrease path delay significantly where average improvements between 17% to 50% can be observed over the LS-only non-QoS routing mechanism. Similar results for the available bandwidth (residual capacity) QoS metric are given in Fig.s 4c and 5c. With this QoS metric, Elessar finds and maintains paths where the end-to-end available capacity of the path is the maximum among all possible. Improvements over non-QoS routing are between 3% and 80% in Fig. 4c and between -25% to 91% in Figure 5c. DCD parameters are seen to play a more important role with the available bandwidth (avBW) QoS metric; we see that with a long p_c, Elessar performs *worse* in QoS routing compared to non-QoS routing. With a high cost period, sources do not get link costs (in this case, avBW information) in a timely manner, causing route adaptation to lag behind current network conditions. With the added overhead of DCD, which is not incurred with non-QoS routing, this causes inadequate performance of QoS routing. This is an important indication that selection of protocol parameters is challenging and must consider the current network conditions (such as node density, mobility rate) as well as the types of supported QoS traffic.

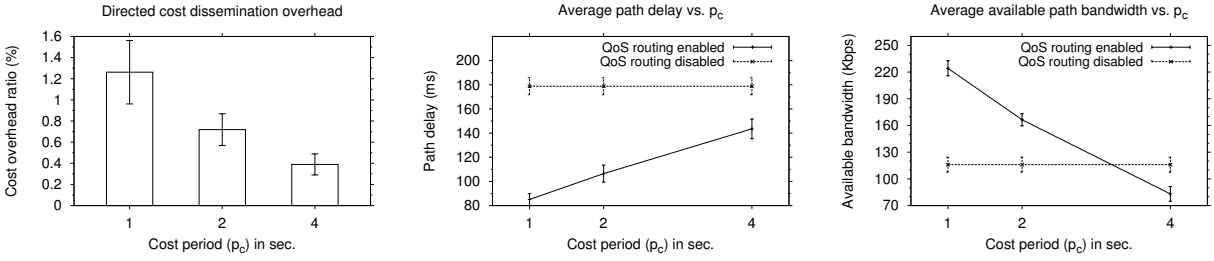
V. Conclusions and Future Work

We have presented a network-layer solution that enables support for concurrent and multiple QoS traffic types over multi-hop wireless mobile ad hoc networks. The proposed solution is in the form of an adaptive hybrid routing protocol that combines an event-driven link-state mechanism with on demand cost dissemination to intelligently select and maintain paths for QoS flows under dynamic network conditions. Our experimental results show that this solution can effectively support different types of QoS traffic flows while keeping



(a) DCD overhead (b) Average end-to-end round-trip path delay (c) Average available path bandwidth

Fig. 4. Effect of cost-inform hop limit (h_m) on protocol performance.



(a) DCD overhead (b) Average end-to-end round-trip path delay (c) Average available path bandwidth

Fig. 5. Effect of cost period (p_c) on protocol performance.

protocol overhead at reasonable levels. We plan to investigate further methods for lowering protocol overhead and develop an extension for multipath QoS routing in the future.

ACKNOWLEDGMENTS

This research has been supported under the TUBITAK CARREER EEEAG104E28 grant.

REFERENCES

[1] T. Karapantelakis and G. Iacovidis, "Experimenting with real time applications in an IEEE 802.11b ad hoc network," in *Proc. IEEE Conf. on Local Computer Networks*, 2005, pp. 554–559.

[2] H. Badis and K. A. Agha, "QOLSR, QoS routing for ad hoc wireless networks using OLSR," *European Transactions on Telecomm.*, vol. 15, no. 4, 2005.

[3] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proc. IEEE INMIC'01*, December 2001, pp. 62–68.

[4] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot, "Performance of multipoint relaying in ad hoc mobile routing protocols," in *NETWORKING'02*, April 2002, pp. 387–398.

[5] C. Santivanez, R. Ramanathan, and I. Stavrakakis, "Making link-state routing scale for ad hoc networks," in *Proc. ACM Intern. Symp. on Mobile Ad Hoc Networking and Computing*, October 2001, pp. 22–32.

[6] D.-Q. Nguyen and P. Minet, "QoS support and OLSR routing in a mobile ad hoc network," in *5th Intern. Conf. on Networking and the Inter. Conf. on Systems*, April 2006.

[7] R. Leung, J. Liu, E. Poon, A. Chan, and B. Li, "MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks," in *Proc. 26th Annual IEEE Conf. on Local Computer Networks*, 2001, pp. 132–141.

[8] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The dynamic source routing protocol for multihop wireless ad hoc networks," in *Ad Hoc Networking*. Addison-Wesley, 2001, ch. 5, pp. 139–172.

[9] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A framework for reliable routing in mobile ad hoc networks," in *Proc. IEEE INFOCOM'03*, March 2003, pp. 270–280.

[10] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc. 2nd IEEE W'shop on Mobile Computing Systems and Applications*, February 1999, pp. 90–100.

[11] Y.-S. Chen, Y.-C. Tseng, J.-P. Sheu, and P.-H. Kuo, "On-demand, link-state, multi-path QoS routing in a wireless mobile ad-hoc network," *Computer Communications*, vol. 27, no. 1, pp. 27–40, 2004.

[12] P. Mohapatra, J. Li, and C. Gui, "QoS in mobile ad hoc networks," *IEEE Wireless Comm.*, vol. 10, no. 3, pp. 44–52, June 2003.

[13] D. D. Perkins and H. D. Hughes, "A survey on quality-of-service support for mobile ad hoc networks," *Wireless Commun. and Mobile Computing*, vol. 2, no. 5, pp. 503–513, Sep. 2002.

[14] R. Asokan, "A review of quality of service (QoS) routing protocols for mobile ad hoc networks," in *Proc. Inter. Conf. on Wireless Communication and Sensor Computing*, January 2010, pp. 1–6.

[15] L. Chen and W. Heinzelman, "A survey of routing protocols that support QoS in mobile ad hoc networks," *IEEE Network*, vol. 21, no. 6, pp. 30–38, 2007.

[16] R.-S. Chang and H.-C. Ting, "Improving the performance of broadcasting in ad hoc wireless networks," in *Proc. 8th Inter. Conf. on Parallel and Dist. Systems*, June 2001, pp. 69–74.

[17] M. Onus, A. Richa, K. Kothapalli, and C. Scheideler, "Efficient broadcasting and gathering in wireless ad-hoc networks," in *Proc. 8th Inter. Symp. on Parallel Architectures, Algorithms and Networks*, Dec. 2005.

[18] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, pp. 111–122, 2002.

[19] A. Varga, "OMNeT++," *IEEE Network Interactive*, vol. 16, no. 4, July 2002, in the column Software Tools for Networking.