Quality of Service Support For Reactive Routing in Mobile Ad Hoc Networks

Navid Nikaein and Christian Bonnet
Institut Eurécom
Sophia Antipolis, France
Firstname.Name@eurecom.fr

Ashish Rastogi
Indian Institute of Technology
Delhi, India
ashishr@cse.iitd.ac.in

***Abstract*— This paper presents a Quality of Service (QoS) model for reactive routing in mobile ad hoc networks, denoted as 2LQoS– two-layered quality of service. We propose network layer metrics and application layer metrics as the additional constraints to the conventional ones to determine paths between a source and a destination. Network layer metrics determine the quality of links in order to generate the paths with good quality. On the other hand, application layer metrics select exactly one path out of the paths with a good quality according to the application requirements. That is why, a two-layered architecture is proposed to deploy quality of service in ad hoc networks.**

***Index Terms*— mobile ad hoc networks, routing, quality of service, metrics.**

I. Introduction

A mobile ad hoc network *manet* consists of a collection of wireless mobile nodes forming a dynamic autonomous network through a *fully mobile infrastructure* [1]. Nodes communicate with each other without the intervention of centralized access points or base stations. In such a network, each node acts both as a router and as a host. Due to the limited transmission range of wireless network interfaces, multiple *hops* may be needed to exchange data between nodes in the network, which is why the literature sometimes uses the term *multi-hop* network for a manet. Therefore, multiple links may be needed to reach the destination from the source. Mobility of the nodes causes link changes. Hence, ad hoc networking becomes a challenging task. Many critical issues need to be addressed such as routing, multicasting, QoS support, and security. This paper presents extensions that facilitate QoS support to applications when a reactive routing protocol is being used by the network.

A routing protocol is the mechanism by which user traffic is directed and transported through the network from the source node to the destination node. Quality of Service means providing a set of service requirements to the flows while routing them through the network. Therefore, Quality of Service routing is a routing mechanism under which paths are determined based on some knowledge of resource availability in the network as well as the quality of service requirements of flows. Therefore, the main objective of QoS routing is to optimize the network resource utilization while satisfying specific application requirements. The core phases in the routing process are:

1) *Path generation*: which generates paths according to the assembled and distributed state information of the network and the application;
2) *Path selection*: which selects appropriate paths based on network and application state information;
3) *Data forwarding*: which forwards user traffic along the selected route.

II. Related Work

The presence of mobility implies that links make and break often and in an indeterministic fashion. This dynamic nature makes routing and consequently QoS support in these networks a challenging task. Further, since the *quality* of mobile nodes (in terms of their connectivity to the network, e.g. enough battery) varies with time, present QoS models for wired networks are insufficient for such networks. *Integrated services* (Intserv) [2] and *Differentiated services* (Diffserv) [3] are the two basic architectures proposed to deliver QoS guarantees in the Internet. A variant of these two architectures: a Flexible QoS Model for Manet (FQMM) [4] has been proposed for ad hoc networks.

- *Integrated services* – Intserv architecture allows sources to communicate their QoS requirements to routers and destinations on the data path by means of a signaling protocol such as RSVP. Hence, Intserv provides per-flow end-to-end QoS guaran-

tees. IntServ defines two service classes: *guaranteed service* and *controlled load*, in addition to the best effort service. The guaranteed service class guarantees to provide a maximum end-to-end delay, and is intended for applications with strict delay requirements. Controlled load, on the other hand, guarantees to provide a level of service equivalent to best effort service in a lightly loaded network, regardless of network load. This service is designed for adaptive real-time applications. As is the case in the Internet, Intserv is not appropriate for mobile ad hoc networks, because the amount of state information increases proportionally with the number of flows, which results in scalability problems.

- *Differentiated services* – Diffserv architecture avoids the problem of scalability by defining a small number of per-hop behaviors (PHBs) at the network edge routers and associating a different Diffserv Code Point (DSCP) in the IP header of packets belonging to each class of PHBs. Core routers use DSCP to differentiate between different QoS classes on per-hop basis. Thus, DiffServ is scalable but it does not guarantee services on end-to-end basis. This is a drawback that hinders DiffServ deployment in the Internet, and remains to be a drawback for manet as well, since end-to-end guarantees are also required in manet. In Diffserv, we can identify three different classes: *expedited forwarding*, *assured forwarding*, and *best effort*. Expedited forwarding provides a low delay, low loss rate, and an assured bandwidth. Assured forwarding provides guaranteed/expected throughput for applications, and best effort which provides no guarantee.
- *FQMM* – a Flexible QoS Model for Manet - is also proposed for mobile ad hoc networks [4]. This model selectively uses the per-flow state property of IntServe, while using the service differentiation of DiffServ. That is to say, for applications with high priority, per-flow QoS guarantees are provided. On the other hand, applications with lower priorities are given per-class differentiation.

In view of the different QoS architectures presented above, the main question is: *How can a routing protocol be extended to support QoS?* For this purpose, we explore some issues in ad hoc routing. One of the issues with routing in ad hoc networks concerns whether nodes should keep track of routes to all possible destinations, or instead keep track of only those destinations that are of immediate interest. A node in an ad hoc network does not need a route to a destination until that destination is to be the recipient of packets sent by the node, either as the actual source of the packet or as an intermediate node along a path from the source to the destination. In the literature related to routing strategies used in mobile ad hoc networks, we find three different classes of routing including *proactive*, *reactive*, and *hybrid* [5]. This paper addresses QoS support for reactive routing protocols. However for the sake of clarity, we describe briefly each of classes.

Protocols keeps track of routes for all destinations have the advantage that communications with arbitrary destinations experience minimal initial delay from the point of view of the application. When the application starts, a route can be immediately selected from the routing table. Such protocols are called *proactive* because they store route information even before it is needed. Certain proactive routing protocols are Destination-Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR) [6], Clusterhead Gateway Switch Routing (CGSR). A comprehensive review of these strategies can be found in [7]. To overcome the wasted work in maintaining unrequired routes, *on-demand*, or *reactive* protocols have been designed. In these protocols, routing information is acquired only when it is actually needed. Reactive routing protocols save the overhead of maintaining unused routes at each node, but the latency for many applications will drastically increase. Most applications are likely to suffer a long delay when they start because a route to the destination will have to be acquired before the communication can begin. Some reactive protocols are Cluster Based Routing Protocol (CBRP) [8], Ad Hoc On-Demand Distance Vector (AODV), Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA), Associativity-Based Routing (ABR), Signal Stability Routing (SSR), Location Aided Routing (LAR) [9]. Hybrid routing protocols aggregates a set of nodes into zones in the network topology. Then, the network is partitioned into zones and proactive approach is used within each zone to maintain routing information. To route packets between different zones, the reactive approach is used. Consequently, in hybrid schemes, a route to a destination that is in the same zone is established without delay, while a route discovery and a route maintenance procedure is required for destinations that are in other zones. The zone routing protocol (ZRP) [10], zone-based hierarchical link state (ZHLS) routing protocol [11], and distributed dynamic routing algorithm (DDR) [12] are three hybrid routing approaches.

This paper presents an overview of the extensions that must be made to reactive routing strategy in an ad hoc network so that Quality of Service guarantees may be in-

corporated in these networks. In section III, we give an intuition for our quality of service model. We suggest an approach which considers the characteristics of manet and tries to emulate both end-to-end service management of Intserv while maintaining the scalability and per-hop service differentiation of Diffserv. In section IV, we describe the basic routing strategies in ad-hoc networks along with suitable extensions so that they may implement QoS. In section IV-B, we illustrate our extensions with the aid of an example. We conclude the paper in section VI.

III. Intuition of 2-Layered QoS Model

With the introduction of real-time audio and video applications, specifically two-way voice communications (example telephony) into mobile ad hoc networks, the communication path that is selected between the nodes has to meet additional constraints (of latency or bandwidth for example). In addition to the destination node, the application must also supply the constraint parameters (i.e., its QoS parameters) to the routing layer so that a suitable path can be found. The routing protocols that support QoS must be *adaptive* to cope with the time-varying topology and time-varying network resources. For instance, it is possible that a route that was earlier found to meet certain QoS requirements no longer does so due to the dynamic nature of the topology. In such a case, it is important that the network intelligently adapts the session to its new and changed conditions.

Unlike fixed networks such as the Internet, Quality of Service support in mobile ad hoc networks depends not only on the *congestion state* but also on the *mobility rate*. This is because *mobility* may result in link failure which in turn may result in a broken path. Furthermore, an ad hoc network potentially has less resources than fixed networks. Therefore, more criterion are required in order to capture the quality of the links between nodes. We propose to measure the *quality of connectivity* and use it in the route discovery procedure. Further, we propose network layer metrics and application layer metrics as the additional constraints to the conventional ones to determine paths between a source and a destination. Network layer metrics determine the quality of connectivity in order to generate the paths with good quality. On the other hand, application layer metrics select exactly one path out of the paths with good quality according to the application requirements. That is why, we propose a two-layered architecture to deploy quality of service in ad hoc networks. In order to incorporate these requirements in the path determination mechanism, the network and application layer requirements must be translated into meaningful metrics. Then, a reasonable combination of these metrics have to be mapped onto QoS classes in such a way that the path computation complexity does *not* make route determination impractical. We believe that this separation is desirable because the quality of service that an application requires depends on the "quality" of the network (see Fig. 1). We propose several network layer metrics to estimate this quality which may be reflected by stability, battery and buffer level of the nodes forming the ad hoc network.

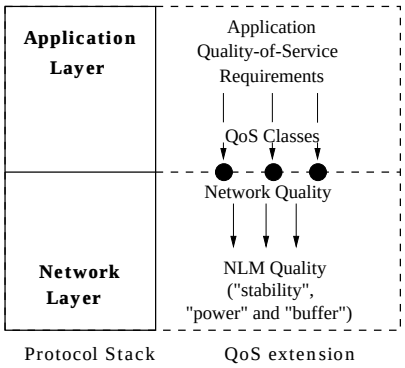

Fig. 1. Global view of our extension proposal

In the process of generating paths from the source to the destination, additional parameters (stability, buffer level and available battery) other than the hop-count are used. The objective of the network layer metrics is to avoid unbalanced network utilization while minimizing the network resource consumption. Therefore, there is a tradeoff between load balancing and resource conservation. The application layer metrics are employed by the *path selection procedure* at the destination node. They attempt to select the most suitable path according to application requirements. At a fundamental level, network layer metrics are node level (*local*) metrics, while application layer metrics are path level (*global*) metrics. In this sense, the network layer metrics become a primary metric and the application layer metrics become a secondary metric.

In order to keep the routing overhead low and support fast routing decisions in QoS routing, we associate a *state* to the available network resources. In the path generation phase, the nodes use this state information to generate paths according to the available network resources. Then in the path selection phase, this state is used in conjunction with the desires QoS class to select the most suitable path according to the application requirements. Furthermore, the state associated with the selected path may be reused at the source node to *shape* packet streams according to the available network resources. The model differentiates services and provides soft guarantees to network re-

sources for an admitted application by using a *class-based weighted fair queuing* (CB-WFQ) at intermediate nodes. Finally, we also consider the possible scenario in which nodes may lie about their states, and propose a mechanism to avoid malicious behavior.

In this section, first we explain how nodes maintain their quality in terms of states in the Network Layer Metrics (III-A) and how these metrics are used and evaluated in the path generation procedure (III-B). Then, we explain the Application Layer Metrics (III-C) that specify the application QoS constraints. Further, we explain how applications are classified into different QoS classes in our model, and then propose a mapping of these classes onto the appropriate network layer metrics(III-D). In (III-E) we describe the shaping procedure. Finally, we explain service differentiation at nodes (III-G) and address issues that prevent malicious behavior of nodes in our model using the status concept and buffer management (III-H).

A. NLM– Network Layer Metrics

As stated earlier, the network layer metrics are used during the path generation procedure in order to indicate the kind of service requirements that can be met by the generated paths. The main objective of NLMs is to provide a trade-off between load balancing and resource conservation. Therefore, they control and maintain the network performance. We define four network layer metrics: *hop count*, *power level*, *buffer level*, and *stability level*. Since hop count is already a minimization metric in all routing protocols, we refer to the other three network layer metrics: power level, buffer level and stability level as the *QoS state*. Note that a *QoS state* is internal to a node and it is periodically evaluated by each node. The *QoS state* of a particular node reveals whether the node is *forced* to be *selfish* or not. In the *selfish* mode, a node ceases to be a router and acts only as a host. We assume that a node periodically broadcasts its network layer metrics to its neighbors in the form of a *beacon*, indicating its presence and its *QoS state*.

1) Hop Count: The hop count corresponds to the number of hops required to a packet to reach its destination. Note that the hop count metric is related to resource conservation, since a path with fewer hops is preferable.
2) Power Level: The power level represents the amount of available battery level (i.e. energy). This metric is related to load balancing. It is translated into a two-bit code that indicates the QoS state of a node in terms of available battery. We classify the QoS state in terms of available battery into *high*, *medium*, *low* and *selfish* states corresponding to each of the four two-bit codes. For example, a *high* QoS state may be indicated if the available battery is between 75% and 100%, and a node may exhibit *selfish* behavior in case its battery is lower than 25%. Intermediate battery levels may be classified into *medium* and *low* states.
3) Buffer Level: The buffer level stands for the available unallocated buffer. Like the power level, this metric is also related to load balancing. It represents a node's internal state, and we assume that a node is capable of determining its state. [1] Note that if the buffer level of a particular node is low, then this implies that a large number of packets are queued up for forwarding, which in turn implies that a packet routed through this node would have to experience high queuing delays. This metric is also translated into a two-bit code which indicates the QoS state of a node in terms of available buffer. Once again, a two-bit code is used to indicate *high*, *medium*, *low* and *selfish* QoS state in terms of the buffer level. A *high* QoS state indicates that the corresponding node no packets queued up for forwarding, while *selfish* QoS state shows that the available buffer is less than 25 percent of its size. Since there is a slight delay between the broadcast of this metric and its use, instantaneous buffer-level may be misleading. Hence, a node should maintain the average buffer-level. Exponentially weighted moving average (EWMA) may be used.
4) Stability Level: We define the connectivity variance of a node with respect to its neighboring nodes over time as the stability of that node. This metric is used to avoid unstable nodes to relay packets. We estimate the stability of a node x as:

$$stab(x) = \frac{|N_{t_0} \cap N_{t_1}|}{|N_{t_0} \cup N_{t_1}|}$$

N_{t_1} and N_{t_0} represent the nodes in the neighborhood of x at times t_1 and t_0 respectively. Note that, $t_1 - t_0$ denotes the time period in which nodes exchange beacons. A node is unstable if a large number of its neighbors change. Further, if most (or all) of the neighbors remain the same at the two times t_1 and t_0, then we call this node stable. Note that $N_{t_1} \cap N_{t_0}$ (the numerator of $stab(x)$) denotes the set of nodes that have remained in the neighbor-

[1] Note that it is trivial to determine the power and buffer level since a node can directly read from its battery and its buffer.

hood of x between times t_0 and t_1. The denominator of $stab(x)$ is a normalization term. A node has *high* stability if none of its neighbors change ($N_{t_1} = N_{t_0}$) , in this case we have $stab(x) = 1$. A node is *unstable* (no stability), if all its neighbors change ($N_{t_1} \cap N_{t_0} = \phi$), in this case we have $stab(x) = 0$. We say that a node has *low* stability if $0 < stab(x) \leq 0.5$ and that it has *medium* stability if $0.5 < stab(x) < 1$. A two-bit code maps the stability to four QoS states of *high*, *medium*, *low* and *no* stability. For the sake of conformity with other metrics, if a node has *no* stability, we say that it has *selfish* stability.

B. NLM Evaluation for Path Generation

In the path generation phase, network layer metrics are propagated through the nodes of generated paths. With the exception of hop count- which is simply the number of hops required to reach the destination, we use concave functions to represent the network layer metrics corresponding to a path given the values of these metrics for individual nodes on that path. Suppose P is a path between source node s and destination node d (that is P is a sequence of (non-repeating) nodes, hence $P =< s, n_1, \ldots, n_k, d >$). One way to estimate the value of these metrics for P is:

$$P.hop = \sum_{n \in P \setminus \{s\}} 1$$

$$P.power = \min_{n \in P \setminus \{s\}} n.power$$

$$P.buffer = (\sum_{n \in P \setminus \{s\}} n.buffer)/P.hop\ count$$

$$P.stability = \min_{n \in P \setminus \{s\}} n.stability$$

The power level of P is represented by the node with the least power on P. Similarly, the stability level of P is represented by the node with the least stability. Indeed, this is appropriate for the route generation procedure, since a route is rendered broken even if one intermediate node has no power or stability. Further, on a path that routes application traffic, all nodes must have have enough battery and stability to support forwarding of the packets of this application. The buffer level of P is estimated as the average over the buffer levels of the intermediate nodes. Paths with higher network layer metrics are preferred in the path selection phase.

Recall that we use a two-bit code to capture the power, buffer and stability levels and classify them into *high*, *medium*, *low* and *selfish* states respectively. In computing the corresponding codes for these metrics on paths, we first map these states to the set $\{3, 2, 1, 0\}$ respectively, evaluate the metrics for the paths as given above, and then unmap the result back to these codes using a ceiling function.

C. ALM– Application Layer Metrics

The application layer metrics are employed by the path selection procedure which is carried out at the destination node. Indeed, they give a *reflection* of paths' class for the application based on the information provided by the network layer metrics. This reflection is a basis to compare the generated path, and select the most suitable one.

- Delay: The delay is the total latency experienced by a packet to traverse the network from the source to the destination. At the network layer, the end-to-end packet latency is the sum of processing delay, packetization, transmission delay, queuing delay and propagation delay. Queuing delay contributes most significantly to the total latency and all other delays are negligible. Hence, it is appropriate to estimate the total latency experienced by a packet by the queuing delay experienced by the packet as it moves from the source to the destination. From the two metrics: *average buffer level* (say $\overline{b}$) and *hop count* (say h), one can estimate the queuing delay experienced by a packet as $h \cdot (r - \overline{b})/c$, where r is the buffer size and c represents the total link throughput. Recall that $\overline{b}$ represents the average unallocated buffer, and hence $r - \overline{b}$ denotes the average buffer occupancy.
- Throughput: The throughput is defined as the rate at which packets are transmitted in the network. It can be expressed as the peak rate or the average rate. Note that the throughput is reduced because of packet loss, that may be caused by link failure due to node mobility and congestion. Assuming that each node's throughput is c, the throughput for an end-to-end connection can be estimated as: $\frac{c}{2h \cdot (r - \overline{b})}$.
- Cost: The cost metric considers the economic aspects of providing services in an ad hoc network. Each node needs incentive to act as a router and forward other nodes' packets. Depending upon the *QoS state* of the node and some other parameters, a node may decide to *charge* a flow that routes packets through it. We estimate the cost metric of a path as an additive function, where each intermediate node

adds its *charge*. The application specifies a cost that it is willing to pay for transmitting its packets, and amongst the paths of a cost lower than the bearable cost, a path is selected based on the other QoS constraints. In our model, we make a simplifying assumption and assume that cost is reflected by the hop count.

D. QoS Classes and Metrics' Mapping

We define three QoS classes for the destination to select the best available path. Class I corresponds to applications that have strong delay constraints, for example applications with real-time traffic such as voice. The corresponding service of this class in Diffserv is referred to as *expedited forwarding* and in Intserv to as *guaranteed service*. The well-known port for this class is VAT. We map this class to the delay metric at the ALM, and to the buffer level and hop count at the NLM. Therefore, the path selection procedure attempts to extract a path that has minimum delay on the basis of the average buffer level and hop count. We assume that queuing delay of a packet is a good estimate of its end-to-end delay. Class II is suitable for applications requiring high throughput such as video or transaction-processing applications. The service of this class is referred to *assured forwarding* in Diffserv and *controlled load* in Intserv. FTP and HTTP are the well-known ports for this class. We map this class to the throughput metric at the ALM, and to the buffer level and hop count at the NLM as in the first class. Finally, Class III traffic has has no specific constraint (best-effort). This class is referred to the *best effort* in both architectures, and is routed on the minimum hop path, like conventional routing protocols. Table I shows the defined QoS classes together with their ALM constraints and the corresponding NLM. A detailed description of how the application layer metrics (ALM) are mapped on to the network layer is given in (III-C).

TABLE I
QOS CLASSES & MAPPING

Class	ALM Constraint	Mapped NLM
Class I	Delay	Buffer & Hop Count
Class II	Throughput	Buffer & Hop Count
Class III	No Constraint	Hop Count

E. Shaping

At this stage, source node knows the QoS state of the selected path to reach the destination. If the QoS state corresponds to the *primary* application requirements, then data transmission occurs without any delay. Otherwise, source node needs to *shape* its traffic. Note that, shaping is the process of delaying or dropping packets within a traffic flow to cause them to conform to the QoS state of the selected path [13]. To decide whether to delay or drop the packets, a node checks the application requirements. If the application is delay sensitive–i.e. class, then the dropping approach may be used. Although this approach implies an increase of loss rate, the probability of the path failure is reduced as it avoids an extra delay. On the other hand, if the application requires low loss rate– i.e. class II, then the delaying approach is more appropriate since the path supports an extra delay caused by this approach.

F. Path Stability Period

The stability level (in the *QoS state*) can also be seen as a widthwise metric since it is considered for all QoS classes. It aims at establishing the most stable path from the source to the destination in order to improve delay performance due to path failure. Stability level metric is used to capture the *duration* for which the communication between the source node and destination node may remain unbroken. This duration is called *stability period*. The rationale relies on the fact that a high stability indicates (with a large probability) a low state of node mobility, while a low stability indicates (with a large probability) a high state of mobility. For a path P between the source node s and the destination node d, one way to estimate the stability period is:

$$SP = P.stability \cdot T$$

where T is the beaconing period for stability evaluation (see III-A). Note that if the stability period of a particular path is equal to the beaconing period T, then this implies that all nodes on this path are stable, and hence the connection is expected to remain unbroken for the entire period T. Stability period if used to estimate the *life* of a discovered path. If the stability period of a path is low, then a new path generation phase is expected to be triggered soon, because it is likely that a link on this path would go down (it has low stability). This is a desired behavior for the delay sensitive application. On the other hand, if the stability period of a path is high, then this path has a long *life*.

G. Service Differentiation in Nodes

In this section, we propose an analogy of DiffServ architecture proposed for the Internet, which extends our model to provide a mechanism that guarantees the network resources for an admitted application on per-hop basis.

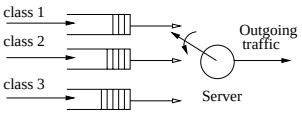

Fig. 2. *Service Differentiation in an Ad hoc Node*

To achieve this, we propose using the *class-based weighted fair queuing* (CB-WFQ) scheduling in ad hoc nodes. Class-based WFQ is the extension of the standard weighted fair queuing (WFQ) [14], [15] functionality to provide support for user-defined traffic classes. In CB-WFQ, a queue is reserved for each QoS class, and traffic belonging to a class is directly forwarded to the queue for that class, see Fig. 2 for details. After packets are assigned to their corresponding queues, they receive prioritized service based on user-configured weights assigned to the queues. We define QoS classes in Tables I. In our approach, classification is performed by a source node. A source node assigns a QoS class to a packet by tagging a (two bit) code to the IP header of each packet belonging to an application. No further classification is required at the intermediate nodes. Upon arrival at an intermediate node, a packet is directly placed to the queue associated to its QoS class Hence, each queue buffers packets belonging to the same *QoS class*. In this model, the packets that reside in the same queue may belong to different applications with the same QoS class.

Finally, the server services packets from different queues based on the priority of the queue, which corresponds to the weights set for each queue in every node. Example of weights for each queue at the node can be set such that, class I service occupies 60% of the CPU times, class II service 30%, and class III service gets 10%. The weights in CB-WFQ are necessary to guarantee minimum bandwidth to each QoS class, this also prevents complete starvation of applications with lower priorities. Furthermore, the unused capacity in CB-WFQ is shared amongst other classes proportional to their weights. Traffic belonging to class I has strong delay constraints, and hence must be forwarded with a priority, and this is captured by service differentiation. Hence, using CB-WFQ, a node guarantees QoS resources of an admitted application through scheduling.

H. Status Concept and Buffer Management

The model we have proposed for QoS support so far considers network layer metrics in the process of path generation. As mentioned earlier, a node broadcasts is *QoS state* (power, buffer and stability level - refer III-A) to its neighbors periodically. The *QoS state* of a node reflects its ability to act as a router. If the *QoS state* of a node is *selfish*, then it ceases to behave as a router and does not take part in the routing protocol.

Since the *QoS state* is propagated by a node itself, it is possible that a node to behaves *maliciously*, and pretends to be in the *selfish* mode. While propagating its *QoS state*, a node may *incorrectly* broadcast poor resources (*selfish QoS state*) that render it *selfish*. In such a case, other nodes in the network will not choose this node as their next hop. Note that the distinction between a *malicious node* and a node *forced* to be in *selfish* mode is entirely personal, and the network has no means to ascertain whether a particular node is lying about its *QoS state*.

We propose a scheme where packets generated by unselfish nodes are preferred during forwarding, and packets generated by nodes that have remained *selfish* for a long time are dropped preferentially. This assures a node that has been in the router mode for a long time that its packets will be forwarded through the network. On the other hand, a node that has been in the *selfish* mode for a long time receives poor service. We introduce a *status* metric with each node that reflects its service to the network. Intuitively, this is explained as follows: If a node has remained in the *selfish* mode for a long time (low *status*), then for this duration, it has not been forwarding other nodes' packets (or it has not been acting as a router.) Hence, it is fair for the network to give poor services to a node that gives it poor service. Note that our scheme discourages nodes to be *malicious* by providing them with an incentive to act as *routers*.

In the path generation process, a node is selected to be in the path for a particular destination based on parameters like *power level*, *buffer level* and *stability*. Intuitively, while generating paths and determining next hops, a node asks itself: *Based on the application requirements, the QoS state of my neighbors and my view of the network, which node must I select as the next hop?* This query facilitates optimization of network resource and QoS support. To avoid *malicious* nodes, and discourage nodes to be *selfish* over long periods of time, we propose an additional query. Upon receiving a packet that a node has to route, we propose that it also asks itself: *Based on the* sta-

tus *of the source node and my available resources, should I forward this packet at all?*

With this intuition for the *status* of a node, it remains to map *status* onto an appropriate metric. We desire that the *status* metric reflect the duration of time a node has been in service to the network (as a router). A node with higher *status* is one that has spent a larger fraction of time servicing the network. One simple way in which a node x may estimate the *status* of node y is by monitoring two variables: the number of beacons received by x from y that reflect y unselfish (call this number $N_u[y]$), and the total number of beacons received by x from y (call this number $N[y]$). The *status* ($R[y]$) of y as perceived by node x is then:

$$R[y] = \frac{N_u[y]}{N[y]}$$

Using this formulation, a node needs to update two entries upon receiving a periodic *QoS state* beacon. Note that $0 \leq R[y] \leq 1$. Hence, the computational overhead placed on a node is not large. Two buffer management schemes are possible. In the first one, an overloaded node may choose to drop the packets of nodes of lower *status*. In the other one, the protocol specifies a threshold, and if the status of a particular sender is below this threshold, then packets from this node are dropped.

IV. 2LQoS Extensions for Reactive Routing Protocols

A. Basic Idea

Reactive route generation (or on-demand route generation) refers to a routing philosophy in which routes are built only when necessary in response to data traffic demands at a source node. In a QoS supported reactive routing strategy, the source nodes broadcasts a *QoS path request* for the destination that it wishes to communicate with. The path request, in addition to carrying the traffic ID of the required flow (source, destination and port number) also carries the desired *QoS class*; the *QoS state*; and the hop count. The QoS class identifies the service that is required by the application (see Table I). It is also used to assign packets to their appropriate queues (refer III-G). As stated earlier, *QoS state* of a path represents the power, buffer, and stability levels of a node (see III-A). The *QoS state* is evaluated and updated as the path request message traverses the network (refer III-B).

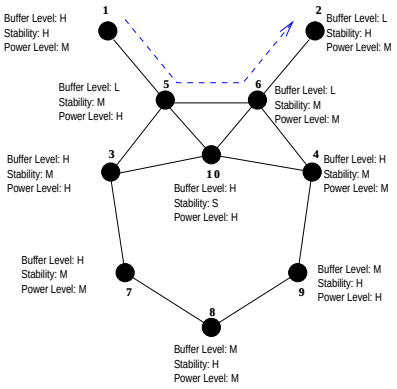

Fig. 3. Example for QoS support reactive strategies. A 10-node topology with the *QoS state* of each node specified.

In a reactive strategy, several paths may be generated (in the path generation procedure) between the source node and the destination node. The destination node uses the *QoS state* of the path request in conjunction with the QoS class of the application to extract the most suitable path according to the application requirements. To give the intuition of our model, for example if the destination receives two paths (P_1 and P_2) with different hop counts (h_1 and h_2, $h_1 > h_2$), different average buffer levels (b_1 and b_2, $b_1 < b_2$), and equal stability level; then it may select P_2 if the application is FTP (since FTP applications require low loss rate), while it may select P_1 if the application is voice (since audio applications require low latency).

B. Example

Consider the topology of a MANET with 10 nodes described in Fig. 3. The topology is modeled as an undirected graph, $G = (V, E)$, where V represents the set of mobile nodes, and E denotes the set of edges. There exists an edge between two nodes if they are in the transmission range of each other (that is the distance between them is less than a fixed radius r.) The *QoS state* for each node are given alongside. H stands for *high*, M stands for *medium*, L stands for *low*, S stands for *selfish* (see III-A). These *codes* indicate the quality of a particular node in terms of the corresponding metrics.

For example, node 8 has *medium* available buffer, *high* stability, and *medium* power level. On the other hand, node 5 has *low* available buffer, *medium* stability and *high* power level. Suppose that the network is routing a flow from node 1 to node 2. Hence, the available buffer levels at the intermediate nodes (node 5 and node 6) are *low*. Next, suppose that an application at node 3 requires to

send data to node 4, and that the QoS requirement of this application is delay constraints along with stability.

We illustrate QoS support in reactive protocols through AODV (ad hoc on-demand distance vector) routing protocol[16]. We assume that each node knows *QoS state* of its neighboring nodes based on the information provided by beacons. When node 3 requires to send application data to node 4, it initiates a route discovery process to locate node 4. This process generates several paths including $< 3, 10, 4 >$, $< 3, 5, 6, 4 >$ and $< 3, 7, 8, 9, 4 >$. The destination node chooses $< 3, 10, 4 >$ which is the freshest and shortest path (see Fig. 3). However, this path is the most unstable path because of the node 10 (selfish state refer III-A) and risks to encounter the path failure during the packet transmission.

If AODV is extended with the proposed QoS, then paths with selfish nodes will be avoided. This means that the route discovery process of AODV will generate two paths including $< 3, 5, 6, 4 >$, and $< 3, 7, 8, 9, 4 >$ (see Fig. 3). At the first stage, this indicates that all the paths via node 10 have been dismissed, hence leading towards load balancing and congestion avoidance. If node 3 desires the best effort service (class III), then node 4 will select the path $< 3, 5, 6, 4 >$. Otherwise, node 4 extracts the NLM of both paths– i.e. hop count = 3, power = *medium*, buffer = *low*, stability = *medium* for the path $< 3, 5, 6, 4 >$; and hop count = 3, power = *medium*, buffer = *low*, stability = *medium* for the path $< 3, 7, 8, 9, 4 >$. Note that, node 4 receives the evaluated and updated NLM (refer III-B). Then, it determines the appropriate metric according to the desired QoS class and computes that metric (refer III-C). Obviously, it selects the path $< 3, 7, 8, 9, 4 >$ if the application requires more than the best effort service. Although the selected path has a larger hop count, unlike the shortest path it meets the application requirements. Therefore, packets belonging to the different QoS classes will be routed differently. Similarly, this leads towards load balancing of the network and improving the performance of AODV. Next, the traffic will be shaped at the source node if the selected path does not meet the application requirements. This avoids network congestion and also improves the performance of AODV. Finally, the path for the priority classes (I & II) will be *renewed* before the path unstability period starts (refer III-F). This is done to avoid path failure as the network topology may change after a certain time, which is a desired property for certain applications. However, nodes may also have unanticipated behavior that may cause path failure. In this case the route maintenance used in AODV is triggered in addition to the previous mechanism.

V. 2LQoS Extension for Proactive Routing Strategies

The 2LQoS strategy of considering network layer metrics while path generation and path selection can also be adopted for routing protocols that use proactive strategy based on link state. In a link state proactive routing protocol, each node maintains a view of the network, and in order to determine paths to destinations, runs the shortest path algorithm on its view of the network topology. If the network layer metrics are embedded in the broadcast message of each node, then a node will maintain a view of the network topology along with the network layer metrics of each node. Thus, it is possible to run the shortest path algorithm with the cost associated with a link depending on these network layer metrics. Depending upon the QoS class of the application layer, the cost associated with a link may be chosen based on one or more network layer metrics.

2LQoS extension for distance vector proactive routing strategies is ambiguous because in the distance vector routing protocol, each node maintains a routing table with exactly one next hop for each destination. Clearly, in order to have support for different application classes, there must be a choice from more than one paths. However, maintaining two paths corresponding to each destination results in extra routing overhead: namely an additional routing table.

VI. Conclusion

This paper have presented a quality of service model for routing in mobile ad hoc network, denoted as 2LQoS– two-layered quality of service. We suggest an architecture that separates network and application layer metrics to achieve two objectives: firstly to avoid unbalanced network resource utilization while minimizing their consumption; and secondly to select a path to meet application requirements. In future work, we will address the performance evaluation of certain well-known routing protocols such as AODV and DSR with and without 2LQoS to make a comparison between them.

Acknowledgment

The authors wish to thank Idirs A. Rai, Yan Moret, and Pietro Michiardi for their useful discussions.

REFERENCES

[1] "MANET - Mobile Ad hoc Network," www.ietf.org/html.charters/manet-charter.html.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," 1994, IETF RFC 1633.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," 1998, IETF RFC 2475.

[4] H. Xiao, W. K. G. Seahand A. Lo, and K. C. Chua, "A flexible quality of service model for mobile ad-hoc networks," in *IEEE VTC2000-spring*, Japon/Tokyo, 2000.

[5] Charles E Perkins, *Ad Hoc Networking*, Addison-Wesley, 2001.

[6] T. Chen and M. Gerla, "Global state routing: A new routing scheme for ad-hoc wireless networks," in *Proc. IEEE ICC'98*.

[7] E. M. Rover and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, 1999.

[8] M. Jiang, J. Li, and Y. C. Tay, "Cluster based routing protocol," IETF Draft, 1999.

[9] Y-B. Ko and N. H. Vaidya, "Location-aided routing in mobile ad hoc networks," *4th Annual International Conference on Mobile Computing and Networking*, MOBICOM'98.

[10] M. R. Pearlman and Z. J. Hass, "Determining the optimal configuration for zone routing protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395–1414, 1999.

[11] M. Joa-Ng and I-Tai Lu, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," *IEEE on Selected Areas in Communications*, vol. 17, no. 8, pp. 1415–1425, 1999.

[12] Na. Nikaein, H. Labiod, and C. Bonnet, "DDR-distributed dynamic routing algorithm for mobile ad hoc networks," in *MobiHOC 2000*. IEEE, Augest 2000, pp. 19–27.

[13] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Network*, pp. 8–18, Match/April 1999.

[14] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proceedings of the Sigcom'89, Symposium on Communications Architecture and Protocols*, 1989, pp. 1–12.

[15] D. Clark, S. Shenker, and L.Zhang, "Supporting real-time applications in an integrated service packet networks: Architecture and mechanism," in *ACM SIGCOM'92*, 1992, pp. 14–26.

[16] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on-demand distance vector routing," IETF Draft, 1999.