On Providing Soft-QoS in Wireless Ad-Hoc Networks

Andrea Zanella, Daniele Miorandi, Silvano Pupolin, Paolo Raimondi
University of Padova – Department of Information Engineering
Via Gradenigo 6/B, 35131 Padova, Italy
e–mail: {andrea.zanella}@dei.unipd.it
phone: +39 049 827 7656 fax: +39 049 827 7699

Abstract

In this paper we present a soft–QoS scheme for wireless networks based on the well–known ad–hoc distance vector (AODV) routing algorithm. We show how AODV may be easily modified to support (in a statistical sense) QoS provisioning over multihop wireless networks. Basically, each route discovery packet gathers an estimate of the delay and bandwidth along the path from source to destination. Then, according to the soft–QoS requirements of the application, the destination chooses the connection path (if any) which best satisfies the service constraints.

The proposed scheme does not rely on priority–based scheduling schemes or complex resource reservation mechanisms. Soft–QoS guarantees are, instead, provided by means of a statistical and distributed Call Admission Control algorithm. In order to verify the effectiveness of the proposed algorithm, we finally apply the scheme to a Bluetooth scatternet scenario and provide simulation results.

Keywords

ad-hoc networks, AODV, QoS, routing

I. Introduction

Wireless ad–hoc networks have been gathering more and more attention in the last period, by virtue of their capability of providing wireless and mobile connectivity at different levels. Generally speaking, a wireless ad–hoc network consists of a set of peer mobile nodes interconnected by multi–hop communication paths. Network organization and management are distributed among all the nodes, with no central controller, no predefined topology and no fixed support [1]. In this scenario, supporting Quality of Service (QOS) connections for multimedia applications represents a very challenging issue.

A general framework for QoS supporting encompasses several different phases and mechanisms [2]. First, the source selects the desired service level and issues a connection–request specifying the required QoS parameters. Then, the network verifies whether the request can be accepted without violating the service-requirements of any other active data flow. This task is performed by QoS–routing and Call–Admission–Control (CAC) mechanisms. QoS–routing aims at finding a *feasible* path between source and destination, i.e., a route that best satisfies the resource requirements. For wired network, the QoS-routing problem is also known as the Constrained Shortest Path Routing Problem, which is NP–complete. Several heuristics have been proposed for its solutions, most of which require central knowledge of the network topology and involve rather sophisticate algorithms, as in [3], [4]. In ad–hoc network scenario, however, such conditions are hardly satisfied and low–complexity distributed solutions are preferable. In this case, the routing algorithm needs to collect the state information about the intermediate links along the path and make them available at the destination side, where the Call–Admission–Control procedure will determine whether the connection request should be accepted or rejected. Once a request is accepted, resources are reserved along the path to guarantee the target service level required by the new traffic flow. Resource reservation is usually performed by the MAC protocol and requires node cooperation and signalling [5], [6]. Finally, data transmission can be started.

Ideally, the service provided to the connection should not be affected by the traffic dynamics of other connections sharing the common links. Such contractual arrangements are usually referred to as *Hard–QoS* guarantees. Hard–QoS, however, badly fits into the framework of wireless networks, where scarce transmission capability, unreliability of the radio links and nodes mobility make rather impossible to provide Hard–QoS guarantees. Furthermore, many multimedia applications do not require Hard–QoS constraints, since the applications may work even if, for short periods of time, QoS requirements are not fully satisfied. The service requests of such type of applications are sometime referred to as *Soft–QoS* guarantees [5].

In this paper we propose a very simple mechanism to provide basic QoS support on low–profile ad–hoc networks, such as, for instance, multi–hop sensor networks. In this context, nodes are usually battery driven and, consequently, their functionalities are very limited. Hence, it may be convenient to move the complexity of QoS–management to the border nodes. The proposed solution is based on a simple modification of the well–known ad–hoc distance vector (AODV) routing algorithm [7], which enables the provisioning of (Soft) QoS guarantees. Although the idea of integrating QoS in flooding–based route discovery algorithms is not new [8], in most existing schemes routing is decoupled from resource reservation, which is deferred to the MAC layer. Conversely, we propose to combine routing and resource reservation in a single multi-path message pass from the source to the destination. Furthermore, in our model nodes are supposed to support only basic functionalities, such as link state monitoring and routing, while flows differen-

This work was partially supported by MIUR within the framework of the PRIMO project FIRB RBNE018RFY

tiation policies, priority–based scheduling or explicit resource management schemes are not required. Soft–QoS guarantees are, instead, provided by means of a distributed Call Admission Control algorithm, which achieves a sort of *statistical resource reservation*. Nodes are simply required to provide dynamic estimation of their available bandwidth and packet forwarding delay in terms of second order statistics. This information is collected by the routing algorithm and made available at the destination node, which will determine whether the connection request should be accepted or rejected. In case of acceptance, the destination node computes the *resource bounds* for each node in the path, i.e., the residual resources that each node should guarantee in order to respect the service level required by the new connection. Resource bounds are, hence, back propagated along the path and stored in the routing table of each node. Successive connection requests that would exceed the resource bounds of a node are not propagated by that node. This mechanism assures that, even in absence of hard resource reservation, an accepted flow does get the required service without violating the QoS-levels negotiated by the established connections. Such a simplicity is obtained to the detriment of a lower utilization of the radio resources. Indeed, in case more connections share a common path, they will experiment the same service characteristics, since flows differentiation is not supported. This means that any connections will receive the same service as required by the connection with the most strict QoS-requirements. Consequently, connections with looser QoS constraints may get much more resources than needed.

To show the effectiveness of the proposed algorithm, a Bluetooth scatternet has been implemented and the scheme has been thoroughly tested by using a commercially available simulation tool [9]. The Bluetooth scatternet network well fits the low–profile ad–hoc network scenario we envisioned, since Bluetooth nodes are usually characterized by limited computational capabilities.

The remainder of this paper is organized as follows. Section II provides a broad characterization of soft–QoS provisioning in ad–hoc networks. The Call Admission Control algorithm is described in details in Section III. The modified AODV routing algorithm is briefly presented in Section IV, while in Section V we describe the simulation scenario and discuss simulation outcomes. Finally, Section VI concludes the paper with a summary of the results.

II. Soft–QoS provisioning

We assume that the service levels required by the applications can be mapped into three Soft-QoS parameters, namely:

- *Minimum peak band*: B_r;
- *End to end delay*: D_r;
- *Target satisfaction*: ξ_r.

The minimum peak bandwidth determines the minimum throughput that should be guaranteed through the entire path from source to destination. The end–to–end delay refers to the maximum sustainable packet delay, i.e., the maximum delay that can incur between the packet generation and its delivery to the final destination. According to the hard-QoS paradigm, such QoS constraints should be fulfilled for the entire duration of the connection. The third parameter, namely the *target satisfaction*, is introduced to relax these commitments. The target satisfaction, indeed, defines the percentage of packets that the application wishes to be served within the QoS constraints. Basically, the target satisfaction is the one–complement of the Soft–index, which is usually defined in the literature as the maximum tolerable QoS–outage probability. Note that $\xi = 1$ corresponds to hard-QoS requests, while $\xi = 0$ corresponds to pure best–service requests.

Let us denote by $P = \{p_1, \ldots, p_N\}$ the multihop path that goes through the N links $p_1, p_2, \ldots, p_N$. Furthermore, let us denote by b_i the residual (unused) bandwidth of link i and by d_i the forwarding delay introduced by the link. Thus, the overall bandwidth and delay along the path P can be defined as follows:

$$B_P = \min_{p_j \in P} \{b_{p_j}\} \tag{1}$$

$$D_P = \sum_{p_j \in P} d_{p_j} \; . \tag{2}$$

Because of the network dynamics, such metrics are expected to change over time. Therefore, we model the residual bandwidth and forwarding delay of each link as random variables. Assuming statistical independency among the links, the probability that the path bandwidth B_P is below the requested value B_r can be expressed as:

$$\Pr\left[B_P > B_r\right] = \prod_{p_j \in P} \Pr[b_{p_j} > B_r] \; . \tag{3}$$

Analogously, the complete statistical description of the path delay D_P can be derived as

$$H_{D_P}(s) = \prod_{p_j \in P} H_{d_{p_j}}(s) \; ; \tag{4}$$

where $H_u(s)$ denotes the Moment Generating Function for the random variable u, defined as $H_u(s) = E[e^{su}]$ where $E[\cdot]$ indicates the statistical expectation operator.

Therefore, for a Bandwidth–constrained connection request to be accepted along a path P, it must be

$$\Pr\left[B_P \geq B_r\right] \geq \xi_r \; ; \tag{5}$$

while, for a Delay–constrained connection request, we have

$$\Pr\left[D_P \leq D_r\right] \geq \xi_r \; . \tag{6}$$

Clearly, different target satisfaction indexes may be provided for bandwidth and delay constraints to allow finer tuning of the required service levels.

III. Call Admission Control algorithm

The exact evaluation of (5) and (6) requires the destination to be acquainted with the complete statistical descriptions of delay and bandwidth of each node along the path. However, in many cases, the statistical distribution of such parameters can be rather safely approximated by a Gaussian distribution. Under this hypothesis, and assuming independency among nodes statistics, the path delay turns out to be a Gaussian variable, which is completely characterized by its second order statistics. Let m_x and σ_x^2 be the statistical average and variance of the random variable x, respectively. Thus, we can write

$$m_{D_P} = \sum_{p_i \in P} m_{d_{p_j}} \; ; \tag{7}$$

$$\sigma^2_{D_P} = \sum_{p_j \in P} \sigma^2_{b_{p_j}} \; . \tag{8}$$

Therefore, the path bandwidth and path delay statistics can be expressed as follows

$$P\left[B_P > B_r\right] = \prod_{p_j \in P} Q\left(\frac{B_r - m_{b_{p_j}}}{\sigma_{b_{p_j}}}\right) ; \quad (9)$$

$$P\left[D_P \le D_r\right] = 1 - Q\left(\frac{D_r - m_{D_P}}{\sigma_{D_P}}\right) ; \quad (10)$$

where Q represents the complementary distribution function of a Gaussian random variable with mean zero, and variance one. Call–Admission Control (CAC) can now be performed by checking conditions (5) and (6).

In the absence of adequate resource reservation schemes, we need to determine, for each node, the minimal residual resources that should be guaranteed in order to avoid violations of existing QoS agreements.

To this end, any time a new connection request is accepted, the nodes compute the *resource bounds* as described in the following.

Bandwidth–constrained requests. Let us assume that condition (5) is satisfied for a connection with peak bandwidth B_r and target satisfaction ξ_r through a path P. Furthermore, let $\hat{\xi}$ be the probability that the bandwidth D_P along the path P exceeds the request B_r. In other words, $\hat{\xi}$ is the actual bandwidth satisfaction provided by the path P, as given by (9). Hence, the Soft–QoS request would be satisfied even if the average residual bandwidth for the generic link j along the path were reduced to the value $\hat{m}_{b_j}$ given by:

$$\hat{m}_{b_j} = B_r - \sigma_{b_j} Q^{-1}\left[\frac{\xi}{\hat{\xi}} Q\left(\frac{B_r - m_{b_j}}{\sigma_{b_j}}\right)\right] ; \quad (11)$$

where $Q^{-1}[x]$ is the inverse Q function, so that $Q^{-1}[Q(x)] = x$. Note that (11) assumes the variance of the link bandwidth does not change and, hence, may overestimate the bandwidth bound. A corrective coefficient should be added to limit such an error.

Delay–constrained requests. Analogously to the previous case, let D_r and ξ_r be the maximum delay and satisfaction index for an accepted connection request. Then, let $\hat{\xi}$ be the actual delay satisfaction provided by the path P, as given by (10). Hence, the Soft–QoS request would be satisfied even if the average residual path delay were increased to the value $\hat{m}_{D_P}$ given by:

$$\hat{m}_{D_P} = D_r - \sigma_{D_P} Q^{-1}\left[1 - \hat{\xi}\right] . \quad (12)$$

This delay bound, which refers to the entire path, has now to be partitioned among the links composing the path. The fraction of extra–delay margin assigned to each link is inversely proportional to the link delay. Hence, for the generic link j in the path, the delay bound will be given by

$$\hat{m}_{d_j} = m_{d_j} + \frac{1/m_{d_j}}{\sum_{k \in P} 1/m_{d_k}} \left(\hat{m}_{D_P} - m_{D_P}\right) . \quad (13)$$

Once again, (12) and (13) assume delay variances do not change and, hence, may overestimate the delay margins.

Before a connection is definitely accepted, the tightest resource bounds among the nodes in the path are made available at the source. This bounds are translated in terms of maximum traffic rate that the source is allowed to inject into the network.

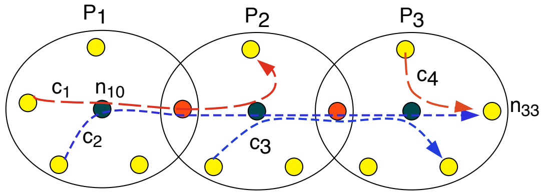

Fig. 1. Scatternet topology and target connections.

The source is then demanded to fulfil this limit or refuse the service.

IV. Path creation and maintenance

The path creation process is highly inspired to AODV, with some simple improvements to permit QoS check and validation. The algorithm makes use of route request packets (RREQ) to discover a path toward the destination and route reply packets (RREP) to fix the selected path.

When a node requires a connection, it specifies four fields, namely the minimum bandwidth, maximum packet delay, and two *target satisfaction–indexes* for the previously defined parameters. These data are embedded in the route discovery packet (RREQ). Each node keeps an estimation (in terms of estimated average and variance) of the bandwidth and delay a packet undergoes when travelling through that node in a store–and–forward manner. The route discovery packet (RREQ) contains information related to the minimum bandwidth and the path delay encountered along that portion of the path.

Bandwidth information is expressed as partial bandwidth satisfaction, whose value is updated at each hop by using (9). Before propagating the message, intermediate nodes check whether Bandwidth–condition (5) is violated. In this case, the path is declared not feasible and the connection request is not further propagated.

Delay information consists of the mean and variance of the delay along the path. Such values are updated at each hop, until the connection requests is received by the destination node, where condition (10) will be verified.

The destination node, upon reception of the route discovery packets, chooses the best path among those that satisfy the QoS requirements. Hence, a route reply packet (RREP) is sent backward along the selected path, to the source. The RREP packet carries information that allows the nodes along the path to compute the excess delay and bandwidth they can still accept without violating the connection–requests. Such resource bounds are stored in the routing table, together with other information related to tthat specific flow. Each table entry is associated to a timer that is restarted any time the table entry is fetched to forward a new packet. In case the timer expires, the entry is definitely cleared. The RREP packet, furthermore, carries the maximum sustainable traffic rate. This information is directly derived from the resource bounds stored in each node along the backward path. Once the source node receives the RREP, it is required to respect the limit imposed by the maximum sustainable traffic rate or to refuse the connection.

V. Simulations results

To prove the effectiveness of the algorithm we developed a Bluetooth scatternet simulator with OPNet [9]. Bluetooth networking is based on small network structures, called Piconet [10]. Within each piconet, a node acts as master while the other node (no more than 7) act as slaves. The master controls the channel access by means of a pure round robin polling policy. Direct communication can occur between master and slaves only: inter-slave traffic is, then, delivered through the master unit. Time is organized in slots of 0.625 ms that are used alternatively for uplink and donwlink transmissions in a Time Division Duplex fashion. Piconets ideally operate on separate physical channels. Inter–piconet communication, hence, is obtained by means of shared units that are connected in a time–division fashion to more piconets. Such special units are generally referred to as *gateways* and the resulting network structure is called *scatternet*.

We simulated a network with three piconets connected in a loop–free fashion, as depicted in Figure 1. We have considered a linear topology in order to avoid the presence of multiple paths between pair of nodes. In this way, we can better analyze the behaving of the Call Admission Control procedure, decoupling it from the QoS–routing that, however, would achieve fair load balancing among the various gateways in the scatternet. Piconets 1 and 3 have four slaves and a single gateway, which forwards traffic from and to Piconet 2. Piconet 2 has three slave units and shares the gateways with the two adjacent piconets. In this scenario, every gateway spends cyclically switch between the two piconets it belongs to, spending an equal time of 50 slots within each piconet. Gateways represent a severe limit to the scatternet performance, especially in terms of maximum delay. For this reason, we have considered only delay–constrained requests in our simulations. Furthermore, Poisson traffic sources have been considered in all the simulations. Each source uses a mix of baseband data packet types, for an average payload of approximately 1500 bits per packet.

As a first step, we verify the validity of the gaussian assumption for the link delay and path delay statistics. To this aim, we have established a traffic connection across the scatternet from a slave in Piconet 1 to a slave in Piconet 3. The connection path is shown by the dotted line marked as c_2 in Figure 1. In each piconet we have established local slave–to–slave connections among the remaining slave units. Local connections generate packets with a rate $\lambda_0 = 6.4$ packets per second (pck/s), while two different generation rates have been considered for the target connection $c2$, namely $\lambda_1 = 8$ pck/s and $\lambda_1 = 56$ pck/s.

Figure 2 show the graphs for the packet delay statistics at node 10 and node 33. The dotted curve gives the real packet delay distribution. The curve marked with $\times$ is the Gaussian CDF obtained by considering the mean and variance delay measured at the destination. Finally, the curve marked with $\bullet$ is the CDF of Gaussian random variable with mean and variance corresponding to the estimation of the path delay statistics gathered by the RREQ packet.

We can see that the Gaussian approximation is fairly close to the empirical delay CDF, in particular for low traffic rates. The gap between empirical and approximated CDF curves tends to become larger for long–distance connection with high traffic. This is probably due to the increasing statistical correlation among the delays introduced by successive hops. Furthermore, we may observe that the accuracy of the estimated delay

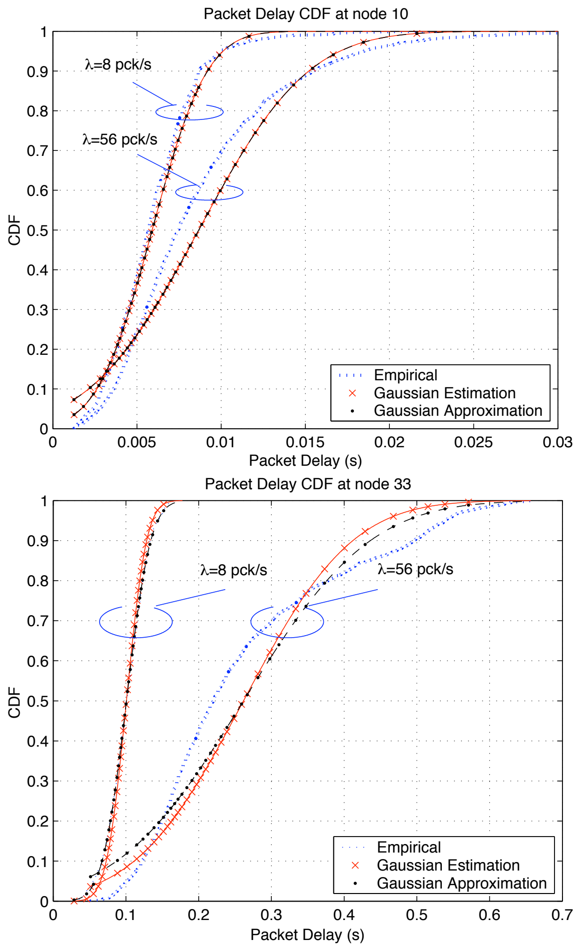

Fig. 2. Cumulative Distribution Functions for the packet delay at node 10 (upper graph) and node 33 (lower graph).

statistics gets worse as the number of hops increases. This is due both to the accumulation of the estimation errors introduced by each node and to the increasing correlation among links delays. Nevertheless, the relative error committed by considering the estimated Gaussian CDF instead of the empirical is rather limited.

In order to verify the effectiveness of the proposed algorithm, we consider a specific simulation scenario. We focus our attention on four reference connections, whose paths are sketched in Figure 1 and denoted as c_1, c_2, c_3 and c_4. Connections c_1 and c_4 require a maximum delay of $D_r = 0.05$ s with a target satisfaction of $\xi = 0.2$. Connections c_2 and c_3 require a maximum delay of $D_r = 0.2$ s with a target satisfaction of $\xi = 0.9$. The average traffic rate is set to 20 kbit/s for c_1 and c_3 connections, to 30 kbit/s for c_2 and to 60 kbit/s for c_4. These reference connections are established at the beginning of the simulation and never released. After 20 seconds of simulated time, nodes start generating connection requests to random destination nodes, at an average rate of 1 requests per second per node. QoS requirements are randomly chosen, while the desired transmission rate is randomly selected in the interval from 5 kbit/s to 20 kbit/sec. If the connection request is accepted, the source will start transmitting packets with a rate that is the minimum between the desired rate and the maximum sustainable traffic rate specified by the network. On average, connections remain active for 10 seconds, then stop.

Figure 3 shows the dynamics over time of the *satisfaction* for the reference connections. Satisfaction is evaluated as the fraction of packets delivered within the maximum delay constraint over the total number of packets transmitted within a time window of 15 s. Figure 4 shows the dynamics of the packet delays experienced by each flow. Values have been averaged over a window of 20 packets. As we can see, new connections may determine fluctuations on the delays experienced by the reference connections. Nevertheless, the CAC and statistical resource reservation schemes guarantee that such fluctuations remain within the Soft-QoS limits. In some cases, however, the approximations introduced in the derivation of the resource bounds may lead to sporadic QoS–requirement violations. An example can be seen in Figure 3, where for a short period of time, the satisfaction of connection c_3 decreases below the target satisfaction value of 0.9.

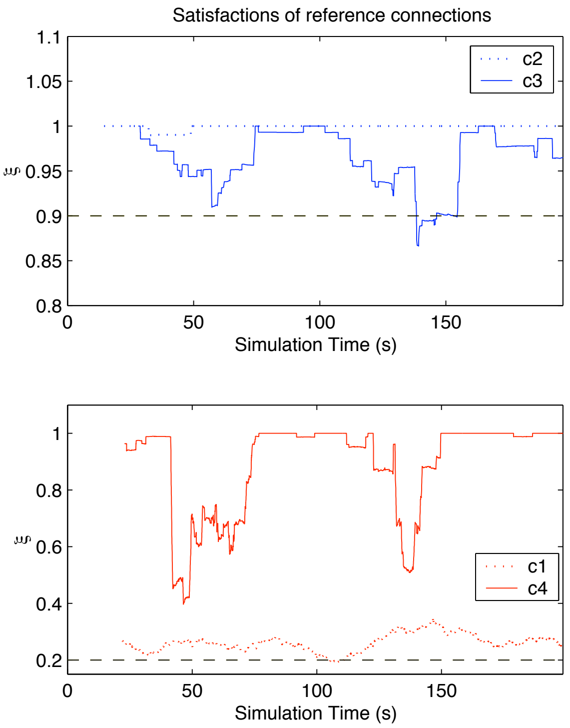

Fig. 3. Simulation results: satisfaction index dynamics.

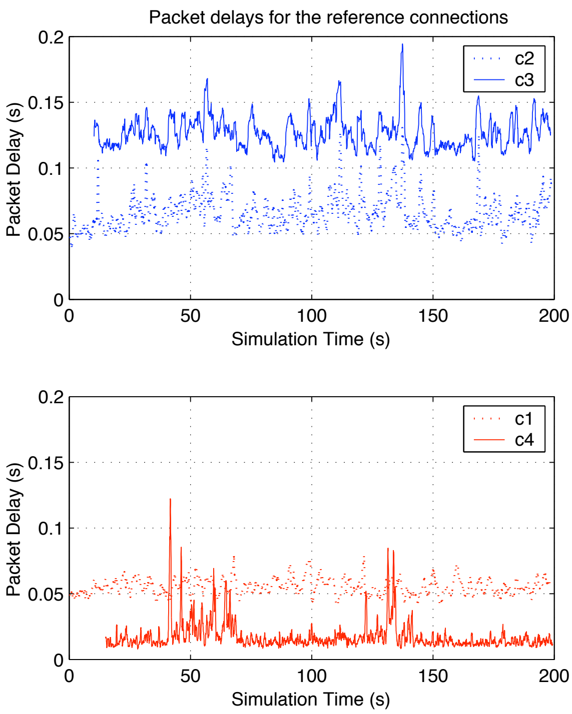

Fig. 4. Simulation results: packet delay dynamics.

VI. Conclusions

In this paper we present a scheme to provide basic QoS support on wireless ad–hoc networks. The scheme encompasses a Call Admission Control procedure that acts on the basis of statistical information gathered by the route discovery packet along its way to the destination node. A connection request is accepted if the resources available along the path can satisfied the QoS requirements with a probability that is higher than the target satisfaction specified by the source. Once the connection is accepted, the source is acquainted with maximum data rate that can be injected into the network without disrupting existing QoS–agreements.

Simulation results show that this method can provide Soft-QoS guarantees without using a reservation protocol. Such a simplicity is paid, though, in terms of lower utilization of the radio resources and higher rate of connection request rejection.

References

[1] M. Frodigh, P. Johansson, and P. Larsson, “Wireless ad hoc networking – the art of networking without a network,” *Ericsson Review*, no. 4, pp. 248–263, 2000.

[2] W. Pattara-Atikom and P. Krishnamurthy, “Distributed mechanisms for Quality of Service in wireless LANs,” *IEEE Wireless Communications*, pp. 26–34, June 2003.

[3] L. L. H. Andrew and A. A. N. A. Kusuma, “Generalised analysis of a QoS-aware routing algorithm,” in *Proc. Globecom*, Sidney, Australia, 1998.

[4] S. Siachalou and L. Georgiadis, “Efficient QoS routing,” in *IEEE INFOCOM*, 2003.

[5] W. K. Wong and H. Zhu, “Soft QoS provisioning using Token Bucket Fair Queueing scheduling algorithm,” *IEEE Wireless Communications*, pp. 8–16, June 2003.

[6] J. Tsai and M. Gerla, “Multicluster, mobile, multimedia radio network,” *ACM-Baltzer Journal of Wireless Networks*, vol. 1, no. 3, pp. 255–265, 1995.

[7] C. E. Perkins and E. Royer, “Ad–hoc on demand distance vector Routing,” in *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA*, Feb. 1999.

[8] P. Mohapatra, J. Li, and C. Gui, “QoS in mobile ad hoc networks,” *IEEE Wireless Communications*, pp. 44–52, June 2003.

[9] *Opnet modeler user manual*, OPNET Technologies. [Online]. Available: http://www.opnet.com

[10] J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen, “Bluetooth: Vision, goals, and architecture,” *ACM Mobile Computing and Communications Review*, vol. 2, no. 4, pp. 38–45, Oct 1998.