Supporting soft real-time services in MANETs using distributed admission control and IEEE 802.11e technology [*]

Carlos T. Calafate, Pietro Manzoni, and Manuel P. Malumbres
Polytechnic University of Valencia, Valencia, Spain
Phone: +34.96.387.7007 - Fax: +34.96.387.7579
E-mail: {calafate, pmanzoni, mperez}@disca.upv.es

Abstract

QoS support in MANETs is a hard and challenging task due to the intrinsic complexities of these networks. In this work we present a solution called DACME to support real-time services in MANETs based on distributed admission control. Using the IEEE 802.11e MAC technology as our basis for traffic differentiation, we develop a technique based on probes to assess available bandwidth in an end-to-end path, as well as the end-to-end delay and jitter. Results show that our technique is quite promising due to the degree of accuracy achieved estimating the different network parameters, while maintaining acceptable levels of traffic overhead and admission delay. Also, since no demands are imposed on intermediate stations, it can be easily deployed.

1 Introduction

In the Internet domain there are two distinct Quality of Service models, which are known as Integrated Services and Differentiated Services. Mobile Ad hoc Networks (MANETs) differ greatly from the Internet environment, and so the development of a QoS framework for MANETs requires adopting a much more flexible philosophy of cooperation and resource sharing among users.

In MANETs we can devise two main policies for resource management and user control. The first one follows the Master / Slave paradigm, where one person or entity (Master) has practically all the control over the slave devices, both in terms of their components (including hardware and software), as well as control over how users will operate them. However, it can only be deployed in very limited situations. The second policy drops the Master / Slave paradigm, embracing cooperation among equals instead. This cooperation may be based on the willingness to achieve mutual benefit, or enforced through "punishment" of selfish nodes. Anyway, there is always a notion of strong dependency among users and complex interactions in the network, which can be remarkably well described using game theory [1] when nodes behave selfishly.

To the best of our knowledge, previous proposals for QoS support in MANETs require that all MANET stations implement some sort of resource reservation or admission control mechanism, fitting perfectly in the first policy for resource management and user control described before. If only a few stations do not implement these mechanisms, then the whole QoS architecture fails or is hindered. Examples of such proposals are FQMM [2], INSIGNIA [3] and SWAN [4].

FQMM [2] has been presented as a flexible QoS model for MANETs. It proposes a hybrid per-flow and per-class QoS provisioning scheme where the IEEE 802.11 MAC layer is used without changes. In a later work [5] authors find that their proposal fails when UDP traffic gets higher priority than TCP.

Lee et al. [3] proposed INSIGNIA, an in-band signaling system that supports fast reservation, restoration and adaptation algorithms. With INSIGNIA all flows require admission control, resource reservation and maintenance at all intermediate stations between source and destination to provide end-to-end quality of service support. However, Georgiadis et al. [6] show that bandwidth reservation in multihop wireless networks is an NP-complete problem.

Ahn et al. [4] designed SWAN, a stateless network model aiming at providing service differentiation in MANETs. SWAN's admission control mechanism requires all stations to keep track of the MAC's transmission delay of all packets in order to estimate available bandwidth; such estimation, though, can be deviated because an IEEE 802.11 radio performs adaptive rate control and also because stations may dynamically select different RTS/CTS and fragmentation thresholds. Such factors difficult the deployment of SWAN in real-life environments.

In this work we propose a solution which we named Distributed Admission Control for Manet Environments (DACME). DACME offers a new framework for QoS support in MANETs based on MAC-level QoS support offered by the IEEE 802.11e [7] technology. DACME uses differ-

[*]This work was supported by the *Ministerio de Ciencia y Tecnologia* under grant TIC/2003/00339, and the *Junta de Comunidades de Castilla La-Mancha* under grant PBC/03/001.

ent kinds of probes to assess the available bandwidth in the network, as well as the delay and the jitter on an end-to-end path. The implementation and deployment of DACME in real-life MANETs is effective, simple, and without constraints or strong requirements on intermediate stations participating on traffic forwarding tasks.

The rest of this paper is organized as follows: in the next section we expose the core of our proposal. In sections 3, 4 and 5 we present several algorithms to assess the available end-to-end bandwidth, end-to-end delay and jitter respectively. Finally, in section 6 we present our conclusions, as well as reference to future work.

2 The distributed admission control mechanism

The admission control mechanism proposed basically requires running a DACME agent at source and destination nodes. DACME agents communicate in order to assess the current state of the path, and decide if a connection should be accepted or not, or if it should be maintained when communication is in progress. Such agents do not require any intervention from the intermediate nodes except forwarding probe packets as if they were actual data. QoS support is achieved by applications by registering with DACME. It works by altering the application's data stream, setting the IP TOS (Type of Service) packet header field according to the application's QoS requirements. The IEEE 802.11e MAC must then treat those packets according to the service type specified in the IP TOS header field.

The admission control is done in two steps: in the first step the source and destination communicate in order to assess the available bandwidth. For those applications that also have delay and jitter requirements, we include an optional second step to assess the end-to-end delay, the jitter, or both.

Relatively to the first step, the source assesses the available bandwidth by sending probe packets to the destination; these packets are generated back-to-back and should be followed by a probe reply. The source agent keeps a timer to detect probe reply losses.

The destination, upon receiving all the probe packets, is expected to send a single reply packet with the measured value for the available end-to-end bandwidth. This value is defined as:

$$B_{measured} = \frac{8 \times P_{size}}{\triangle t_{rec}} \cdot (N_{packets} - 1) \text{ (bit/s)} \qquad (1)$$

where P_{size} is the size of the data segment of each packet, $\triangle t_{rec}$ is the time interval between the first and the last received packet, and $N_{packets}$ is the number of packets received (not the number of packets sent). In order to achieve more accurate results, this process should be repeated a certain number of times, though not too many times due to mobility related constraints and also to avoid long startup times, as we will see later in section 3.3.

The destination agent must also take into account that some packets may be lost. So, when destination receives the second and following packets it launches an internal timer, using the following expression to obtain the timeout value:

$$T = \frac{T_{last} - T_{first}}{N_{recv} - 1} \cdot (N_{rem} + \varepsilon) + \tau, \qquad (2)$$

where T_{last} and T_{first} are the arrival times of the last and first packet received, N_{recv} is the number of packets currently received, N_{rem} is the number of remaining packets (the total number is indicated by the source), and ε is a fixed number of additional packets (to be defined) used to model a certain degree of tolerance. In the first part of the expression $[\frac{T_{last} - T_{first}}{N_{recv} - 1} \cdot (N_{rem} + \varepsilon)]$ we try to dynamically accommodate the time tolerance to the observed network performance. However, there are situations where we cannot predict the timeout value correctly; an example is a MANET where the routing protocol splits traffic through multiple paths. So, to take into account such situations, we add a small constant time value (τ) to it.

If the application is only bandwidth constrained, the source will then notify it if the connection can currently be admitted or not. If the application also has requirements on end-to-end delay and delay jitter, the DACME source agent will enter into step two and perform new tests to assess the current end-to-end delay and jitter values.

End-to-end delay measurements are done in a ping-pong fashion, in a similar way to the ICMP echo packets exchanged by the "ping" command. The main difference is that a new *echo request* packet is immediately sent after receiving an *echo reply* packet. The purpose is to reduce as much as possible the time used to perform measurements. Also, the *echo reply* packet should have the same length and the same value in the IP TOS field as the *echo request* one.

Relatively to jitter measurements, the source must send packets with the size, IP TOS field value and data rate chosen by the application. The receiving end, aware of the source's packet sending rate by explicit notification, calculates the mean and standard deviation values for the absolute jitter and returns them to the source. Jitter measurements can also be used to obtain an estimate on network congestion by counting the number of packets lost.

After all the required measurements are done, the source agent will decide weather to admit or deny a connection, or it can simply inform the application layer about network status, according to previous negotiations.

3 Available bandwidth estimation

In this section we will tune the admission control mechanism to achieve reliable bandwidth measurements in a short period of time. We consider that if measurements take too long they can be corrupted by mobility, making them unreliable and possibly useless. We are also aware that longer measurements lead to more reliable and accurate results, so we need to find the best trade-off.

In order to obtain precise values we devised a static scenario which allows us to make measurements in a controlled environment. Should this environment be mobile instead of static, we would not be able to perform accurate or

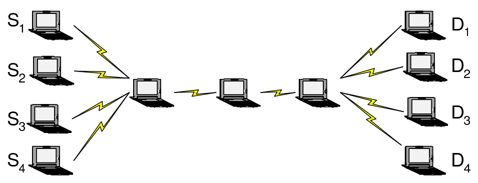


Figure 1. Static scenario

even meaningful comparisons between reference values and those obtained with probes. Apart from other problems, we would have interferences due to routing traffic and variable number of hops. These would cause available bandwidth to vary with time, making any measurements meaningless for our purpose.

The proposed static scenario is presented in figure 1. When choosing it we took into account several factors. First off all, we desired that contention among stations had a clear impact on performance. Our aim was also to create a multi-hop environment, since this is typical of MANETs. That was achieved by including several intermediate stations on the end-to-end path. With this setting we also experience hidden terminal effects and the impact of carrier sensing ranges (considerably larger than data communication ranges).

To conduct our experiments we used the ns-2 simulator [8] with the IEEE 802.11e extentions by Wietholter and Hoene [9]. We set up the radio parameters for IEEE 802.11g. Concerning the IEEE 802.11e MAC, it was configured according to [7]. Our measurements were made over a period of 60 seconds and averaged over twenty simulations runs. We use static routing so that routing actions do not interfere with data traffic. The purpose is to make reliable measurements in a steady-state environment.

Relatively to the sources of traffic, source/destination pair (S_1, D_1) is used by reference streams of different categories to generate data at a very high rate, so that the source's network queue is always full; it is also used to perform measurements using probes. The three remaining source/destination pairs are used to generate different levels of background traffic by varying the data generation rate. The purpose is to saturate the network gradually. These three background sources generate traffic with negative-exponentially distributed inter-arrival times for all Access Categories available in IEEE 802.11e, which are *Voice*, *Video*, *Best-effort* and *Background*.

Concerning the RTS/CTS and fragmentation mechanisms, they are turned off. We set the packet size to 512 bytes for all sources, including probing packets.

To obtain reference bandwidth values we perform different simulation experiments where we set source S_1 to generate no traffic, Voice test traffic and also Video test traffic. In figure 2 we show how the aggregated background traffic (BG) and the test traffic change by varying the background load. We have a level of confidence of 99% that the mean value will be within 1% of all points represented.

From figure 2 we see that as the background load increases, the bandwidth share obtained by Voice traffic is higher, especially in saturation. So, similarly to what was found in [10], we should set the probes always to the same priority to avoid that a source accepts connections of higher priority that would degraded the performance of its own ongoing connections with lower priority. This is known as the stolen bandwidth problem. Therefore, we will perform all our bandwidth measurements using probes set to the Video AC.

3.1 Probe size tuning

In this section we will find the optimal number of packets per probe taking into account the trade-offs between the accuracy of bandwidth measurements and probing time. We use the values for the Video test traffic depicted in figure 2 as reference, and we test different probes with sizes (number of packets) equal to 2, 3, 4, 5, 10, 20 and 50.

The results of our experiments show that the probing process tends to over-estimate the available bandwidth slightly. Also, the degree of accuracy achieved with different probe sizes can vary greatly. In table 1 we assess this accuracy using the peak value of the average normalized standard deviation found among tests under high congestion.

As the normalized standard deviation shows, the measurement results tend to spread more as the probe size decreases. We conclude that tested probe sizes inferior to 10 can lead to significant errors, and so should be avoided. We also conclude that several consecutive probe cycles are required in order to achieve an accurate estimate of available bandwidth.

Table 1 also shows that, under network saturation, a single cycle time can reach relatively high values (more than one second for a probe size of 50). Moreover, there are further problems under saturation, such as a probe becoming unusable due to an excessive number of packet losses. Taking into account all the results found in this section, we consider that setting the number of packets per probe to 10 is a reasonable choice, and so we will use 10-packet probes from now on.

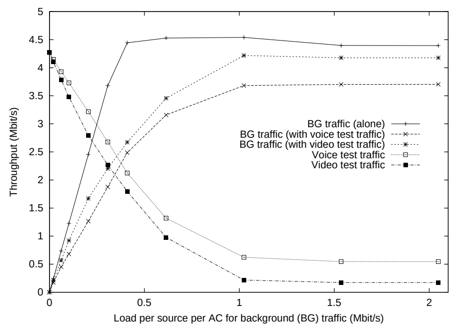


Figure 2. Measured available bandwidth varying the background traffic

Table 1. Peak values for the average normalized standard deviation and single cycle times under high congestion

Probe size (num. packets)	2	3	4	5	10	20	50
Peak value (normalized)	2.66	2.25	2.39	1.78	0.53	0.37	0.21
Single cycle time (s)	0.41	0.42	0.45	0.46	0.53	0.67	1.09

Table 2. Peak and mean relative errors using different probe set sizes

Probe set size	Mean error (%)	Peak error (%)
2	3.62	132.00
3	2.69	106.07
4	2.57	73.54
5	2.55	57.95
6	2.53	55.00
7	2.52	50.95
8	2.51	50.01

3.2 Improving the estimation of available bandwidth

In this section we will improve the bandwidth estimation process since observation of the results obtained using $B_{measured}$ (1) showed that the sample mean was a biased estimator for the available bandwidth. So, we need to correct the measured values. With that aim, we introduce a corrected estimator for available bandwidth:

$$B_e = \alpha \cdot \mu_e + \beta \cdot \sigma_e \tag{3}$$

where μ_e is the sample mean, σ_e is the standard deviation of the sample and parameters (α, β) are used for curve fitting purposes using least square error regression. We obtain optimum values for $(\alpha, \beta) = (0.9430, -0.3794)$, and the degree of accuracy achieved is excellent.

3.3 Tuning the number of probe cycles

In this section we will tune the number of probe cycles in order to find a good balance between bandwidth estimation accuracy and admission control time. We shall refer to this number as the probe set size (in contrast to probe size, which refers to the number of packets per probe). Using the bandwidth estimator developed in the previous section (B_e), we study the mean and peak estimation errors when varying the probe set size.

Table 2 shows that both mean and peak errors tend to reach a steady value after a given probe set size. In fact, we observe that the peak error between probe sets of sizes 5 and 6 decreases by less than 3%, and that the mean error decreases by less than 0.03%. In contrast, the peak error decreases by more than 15% between probe sets with size 4 and 5.

We also consider the total admission control time using different numbers of probe sets in our study. We take into consideration the probability for a probe to become unusable, so that our admission time estimates are correct.

We verify that, under small/moderate congestion, the admission control time in our scenario using a probe set size of 5 is very low (under 300 ms), never reaching values above 3.5 seconds even when the network suffers high levels of traffic congestion. Taking into consideration the results found, we consider that using probe sets sized 5 is a reasonable choice, offering a good balance between bandwidth estimation accuracy and admission control time.

4 End-to-end delay estimation

In this section we will present the algorithm used to estimate end-to-end delay values at admission time. Such measurements can be useful to applications that have strict end-to-end requirements and that, without bounds on delay, cannot meet the purpose they were designed for.

We start by finding what is the minimum number of consecutive ping-pong probes that offers an acceptable degree of accuracy on end-to-end delay estimation. With that purpose we again take the scenario of figure 1 for analysis. We set sources S_2, S_3 and S_4 to generate moderate background traffic with negative-exponentially distributed inter-arrival times at 200 kbit/s per AC per source; source S_1 is used to obtain the reference values for different source rates with either Voice or Video traffic at constant bit-rate.

According to the measurements performed using the bandwidth estimation tool developed in section 3, the available bandwidth is of 2.796 Mbit/s.

Using these reference values, we now proceed by performing end-to-end delay estimation through probes. The process consists, as referred in section 2, of performing several consecutive ping-pong tests without allowing any wait interval between the time a reply packet arrives and a new request packet is sent. This is to accelerate the probing process as much as possible.

After the test ends, we use these values to obtain a low-rate estimate for end-to-end delay. We call it low-rate estimate since, as we will discuss next, it needs to be adjusted according to the ratio between requested and available bandwidth.

To obtain values directly related to the actual traffic, probe priority is set either to the Voice or Video access categories. In figure 3 we show the mean and standard deviation values for an increasing number of consecutive ping probes.

As expected, as the number of consecutive ping probes increases, the mean value tends to increase (congestion increase) and the standard deviation to decrease. Taking into consideration both accuracy and the time to accomplish this task, we decide to use 3 consecutive ping probes for further study. The total delay in this step using the current configuration is around 7 milliseconds, a very low value.

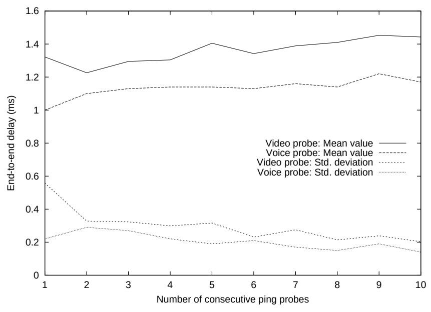


Figure 3. Mean and standard deviation values for Voice and Video end-to-end delay probes

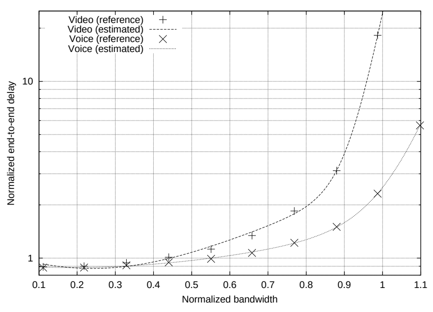


Figure 4. Reference and estimated end-to-end delay values

As referred previously, the end-to-end delay estimation value found through ping probes is a low-rate estimate. This means that it can accurately estimate the delay if the application rate is small when compared to the estimated available bandwidth value. We wish to find a method to estimate the end-to-end delay values for medium/high traffic bandwidths. With that purpose, we first normalize both application bandwidth and end-to-end delay values, and then we employ curve-fitting using the function:

$$D_e(x) = e^{\alpha \cdot x + \beta} + \gamma + \eta \cdot x + \vartheta \cdot x^2 \tag{4}$$

where x is the normalized bandwidth value and parameters $(\alpha, \beta, \gamma, \eta, \vartheta)$ are used for curve fitting purposes. We have tested other types of functions, but this was the one which offered the best results. The value of x is obtained by the ratio between the application's chosen data rate and the bandwidth estimated through the probing process. Relatively to D_e values, these are normalized delay values obtained dividing the different end-to-end delay values by the low-rate delay estimation (obtained by the ping-pong probing process). The normalization of both bandwidth and end-to-end delay values has the purpose of finding values for the different parameters that are independent of the scenario used, so that we can find a general trend of the behavior of delay for both access categories when varying relative bandwidth usage. Using estimator D_e we can then estimate the end-to-end delay in a system by multiplying this normalized delay value found by the low-rate value obtained in the ping-pong process to de-normalize it.

Figure 4 allows a visual comparison between the reference delay values and the estimated delay curves for Voice and Video traffic.

As a final remark, we should point out that this method is only required when the application traffic is waiting on queue, either because initial admission control is being performed, or because route breaks caused the application to stop transmission. During normal transmission measurements can still be made, though no bandwidth-dependent adjustments are required.

5 Jitter estimation

In this section we will detail the jitter estimation process. We wish to obtain not only the mean absolute jitter value, but also the standard deviation for the jitter.

We use the same scenario configuration of the previous section in order to obtain reference values for the jitter. We also use that scenario to generate jitter probes and perform measurements. These probes consist, as referred before, of sending some packets of the same QoS class and at the same rate as the application traffic, and the collection of jitter statistics by the agent in the destination, which are returned to the source in a single packet.

The minimum number of packets required to obtain mean and standard deviation jitter values is 3, though many more should be sent to resemble actual traffic.

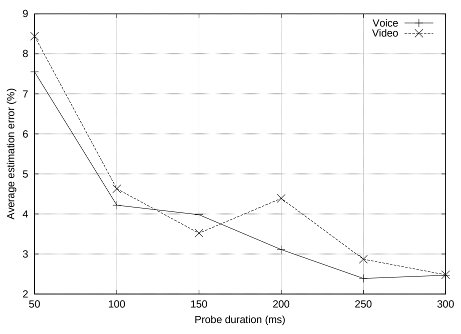
Contrarily to the previously proposed probing processes for available bandwidth and end-to-end delay estimation, in the jitter estimation process we will not find the minimum number of packets that offers good accuracy for all traffic rates; instead, we will find the smallest time interval for probing that offers the desired accuracy, which is the opposite strategy. This way we bound the delay associated with the jitter probing test, making it independent of the application's data rate. An interval range between 50 and 300 ms seems reasonable and is acceptable as admission control delay, and so we will use it for further analysis.

Our objective is to find the minimum interval of time that offers a reasonable accuracy when estimating the mean absolute value and the standard deviation for the jitter.

In figure 5 we show the relative error estimating the mean absolute value and the standard deviation for the jitter. Again, longer probe duration intervals are related to smaller estimation errors.

From the results found previously we consider that a probe duration of 250 milliseconds is enough to achieve accurate and consistent estimations for the mean absolute jitter value, as well as for the standard deviation. Relatively to measurements when traffic streaming from the source has already started, it is not possible to implement this method and it should be replaced by actual traffic measurements in the destination. When re-routing is performed the proposed method can be applied, penalizing the connection with a small extra time after the path is again available.

With this last step we conclude the tuning of the admission control system, having therefore set the complete



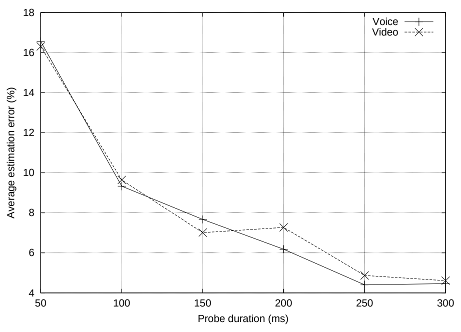


Figure 5. Average estimation error for the absolute mean (top) and the standard deviation (bottom) for Voice and Video traffic varying the probe size

framework required for DACME agents to be implemented and configured properly.

6 Conclusions and future work

In this paper we presented DACME, a solution for supporting soft real-time services in MANETs. Our proposal uses distributed admission control techniques and imposes very few requirements on MANET nodes.

The core of our contribution consisted of algorithms based on probing that are used to determine the end-to-end available bandwidth, delay and jitter. The measurements made using probes are used by DACME agents to decide whether to admit or maintain connections based on their QoS requirements. Preliminary results show that our distributed admission control technique is able to offer reliable end-to-end measurements, and that the time spent in that process is typically low.

As future work we plan to evaluate the effectiveness of our proposal in typical mobile MANET environments by implementing a version of our DACME agent for the ns-2 simulator.

References

[1] A. Urpi, M. Bonuccelli, and S. Giordano. Modelling cooperation in mobile ad hoc networks: a formal description of selfishness. In *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, March 2003.

[2] Hannan Xiao, Winston K.G. Seah, Anthony Lo, and Kee Chaing Chua. A flexible quality of service model for mobile ad-hoc networks. In *IEEE 51st Vehicular Technology Conference Proceedings*, volume 1, pages 445–440, Tokyo, 2000.

[3] S.B. Lee, A. Gahng-Seop, X. Zhang, and Andrew T. Campbell. INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks. *Journal of Parallel and Distributed Computing (Academic Press) , Special issue on Wireless and Mobile Computing and Communications*, 60(4):374–406, April 2000.

[4] G-S. Ahn, A. T. Campbell, A. Veres, and L. Sun. Supporting service differentiation for real-time and best effort traffic in stateless wireless ad hoc networks (SWAN). *IEEE Transactions on Mobile Computing*, September 2002.

[5] Hannan Xiao, Kee Chaing Chua, Winston K.G. Seah, and Anthony Lo. On service prioritization in mobile ad-hoc networks. In *IEEE International Conference on Communications (ICC 2001)*, Helsinki, Finland, June 2001.

[6] Leonidas Georgiadis, Philippe Jacquet, and Bernard Mans. Bandwidth Reservation in Multihop Wireless Networks: Complexity and Mechanisms. In *International Conference on Distributed Computing Systems Workshops (ICDCSW'04)*, Hachioji - Tokyo, Japan, March 2004.

[7] IEEE 802.11 WG, IEEE 802.11e/D8.0, ”Draft Amendment to Standard for Telecommunications and Information exchange between systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements”, February 2004.

[8] K. Fall and K. Varadhan. ns notes and documents. The VINT Project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, February 2000.

[9] Sven Wietholter and Christian Hoene. Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26. Technical Report TKN-03-019, Telecommunication Networks Group, Technische Universitat Berlin, November 2003.

[10] Lee Breslau, Ed Knightly, Scott Shenker, Ion Stoica, and Hui Zhan. Endpoint Admission Control: Architectural Issues and Performance. In *Proceedings of ACM Sigcomm 2000*, Stockholm, Sweden, September 2000.