A framework to deploy bandwidth constrained applications in IEEE 802.11-based ad hoc networks*

Carlos T. Calafate, Pietro Manzoni y Manuel P. Malumbres
Dept. de Informática de Sistemas y Computadores
Facultad de Informática
Universidad Politécnica de Valencia
E-mail: {calafate, pmanzoni, mperez}@disca.upv.es

Abstract

Pervasive computing environments are characterized by the formation of spontaneous networks, also known as ad hoc networks. QoS support in ad hoc networks is a hard and challenging task due to the intrinsic complexities of these networks. In this work we present a solution - DACME - to support bandwidth constrained applications in IEEE 802.11 based ad hoc networks. We propose using the combination of distributed admission control and the IEEE 802.11e MAC technology as our basis for traffic differentiation. Experimental results show that with DACME we can greatly improve the support of multimedia applications in ad hoc networks with little overhead.

1 Introduction

The transition towards ubiquitous computing environments requires improving the currently available technologies to offer more intelligent applications, more powerful terminals and also more flexible and self-structuring networks.

Mobile ad hoc networks (MANETs) are the most natural solution to provide support when a plethora of devices must communicate among themselves in a same environment, especially when the wireless medium is the predominant channel of communication. The deployment of a QoS framework in such a pervasive environment is an important issue to address due to the increasing demand for video and voice communication, and also to improve the support for bandwidth constrained or time-critical applications.

In ad hoc networks we can devise two main strategies to achieve QoS support. The first one consists of configuring all devices pertaining to the MANET to actively participate in the QoS tasks; all devices must then assess the available resources and perform resource reservations or some sort of admission control, thereby contributing to the overall QoS framework. The second strategy consists of reducing the efforts of the MANET terminals as much as possible, so as to avoid collapsing those devices with limited power and resources with QoS-related tasks. According to this second strategy, only the source and destination are required to actively participate in providing end-to-end QoS.

To the best of our knowledge, previous proposals for QoS support in ad hoc networks require that all network stations actively participate in the QoS-support efforts, implementing some sort of resource reservation or admission control mechanism. Therefore, these fit perfectly in the first of the two strategies outlined previously. 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 [6], INSIGNIA [10] and SWAN [5].

FQMM [6] has been presented as a flexible QoS model for ad hoc networks. It proposes a hybrid per-flow and per-class provisioning

*Este trabajo se ha realizado en el ámbito de los proyectos TIN2004-03678-C02-01 del *Ministerio de Educación y Ciencia* y PBC/03/001 de la *Junta de Comunidades de Castilla La-Mancha*, estando parcialmente financiado por el *Programa de Incentivo a la Investigación de la Universidad Politécnica de Valencia*.

scheme, so that traffic of the highest priority is given per-flow provisioning, while other category classes are given per-class provisioning.

Lee et al. [10] 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.

Ahn et al. [5] designed SWAN, a stateless network model aiming at providing service differentiation in ad hoc networks. 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.

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 ad hoc networks based on MAC-level QoS support offered by the IEEE 802.11e [7] technology. DACME uses probes to assess the available bandwidth in the network, and performs admission control based on such measurements. The purpose of DACME is offering a solution whose implementation and deployment in real-life ad hoc networks is effective, simple, and without constraints or strong requirements on intermediate stations participating on traffic forwarding tasks. The only requirement imposed by our technique is that the different terminals conforming the MANET are able to offer the basic level of QoS support as dictated by the IEEE 802.11e standard. This is in clear contrast will all the QoS frameworks referred before. In the context of pervasive computing environments, DACME's functionality could be applied to test several alternative paths, providing a method to determine which paths are able to offer the desired amount of bandwidth.

The rest of this paper is organized as follows: in the next section we expose the core of our proposal. In section 3 we detail the algorithm used to assess the available end-to-end bandwidth. In section 4 we evaluate DACME in both static and mobile ad hoc networks. Finally, in section 6 we present our conclusions, as well as references to future work.

2 The distributed admission control mechanism

The admission control mechanism proposed basically consists of DACME agents running at the source and destination nodes. Both agents communicate in order to assess the current state of the path and decide when a connection should be accepted, maintained or dropped. Such agents do not require any intervention from intermediate nodes besides forwarding probe packets as if they were real data. QoS support requires applications to register with DACME, which is responsible for setting the IP TOS (Type of Service) packet header field according to the QoS required by the application. The IEEE 802.11e MAC must then treat those packets accordingly.

In figure 1 we present the functional block diagram of the DACME agent. The steps to follow are the following: initially an application registers with the DACME agent by indicating the source and destination port numbers, the destination IP address and the QoS parameters.

The *QoS measurement module* assesses the available bandwidth by sending probe packets to the destination; we set the probe packets to the Video Access Category independently of the type of service registered by the application (see section 3).

Probe packets are generated back-to-back and should be followed by a probe reply. The source agent keeps a timer to be able to react in case the probe reply is never received. So, after sending the last packet of the probe, it sets the timer to be triggered after a fixed time interval (PROBE_RESPONSE_TIME = 0.5s). If no probe reply is received, the source considers the path to be down and schedules a new probing cycle after INTER_PROBE_TIME (3s), to which we add a jitter of ± 0.5s to remove possible negative effects that could arise from probe synchronization.

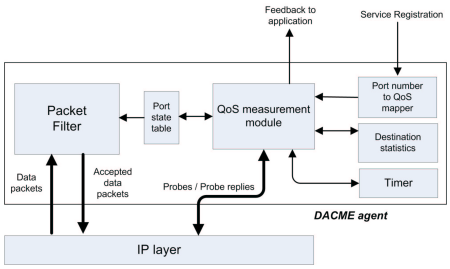

Figure 1: Functional block diagram of the DACME agent

The DACME agent in the destination, upon receiving the probe, will obtain a measure of available end-to-end bandwidth. This value is defined as:

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

where P_{size} is the size of the data segment of each packet, Δt_{rec} is the time interval between the the first and the last packet arriving, and $N_{packets}$ is the number of packets received (not the number of packets sent). This method presents some resemblance with the one proposed in [3] for the Internet, though in that work authors used echo packets instead. They did not have to deal with radio interference problems or different Access Categories at the MAC level.

The DACME agent in the destination will then send this estimate value back to the source. However, this destination agent must also accommodate to the possibility that not all probe packets arrive. So, when the destination receives the first packet it merely updates the current sequence number. When the second or the following packets are received, it continuously updates an internal timer setting it to be triggered after:

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

where T_{last} and T_{first} are the times of arrival of the last and first packet received, N_{recv} is the number of packets currently received, N_{rem} is the number of packets that remain (not received yet), and ε is a fixed number of additional packets used to model a certain degree of tolerance; in our experiments we empirically set it to 3. While in the first part of the expression we try to accommodate dynamically to the state of congestion of the network, 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 also add τ, a small constant time value; in out experiments it is set to 50 ms according to the results we found in [1]. If the timer goes up and the destination did not receive more than half of the packets, it notifies such occurrence to the source (null probe).

The DACME source agent, when receiving the probe reply packet, will collect the $B_{measured}$ values sent by the destination agent and will use it to decide whether to admit the connection or not, updating the *Port state table* accordingly. As we will show in section 3.2, these bandwidth measurements need to be corrected to make such short term bandwidth measurements match long term bandwidth measurements. Taking into account the need to correct this bandwidth deviation, we propose a strategy to perform probabilistic admission control as described in algorithm 1.

Algorithm 1 Probabilistic admission control mechanism

After receiving a probe reply **do** {
correct the bandwidth estimation
if *(there is a level of confidence of 95% that available bandwidth* $\geq$ *requested bandwidth)*
then *accept the connection*
else if *(there is a level of confidence of 95% that available bandwidth* $<$ *requested bandwidth)*
then *drop the connection*
else if *(number of probes used* $<$ *maximum allowed)*
then *send a new probe*
else *maintain the previous path state* }

This algorithm allows reducing the number of probes required to perform a decision to a value as low as two probes; it occurs often in those situations where it becomes quickly evident that the available bandwidth is either

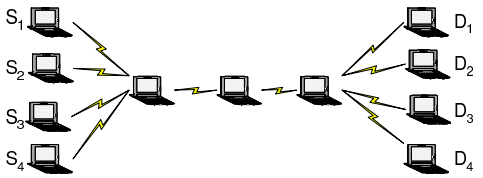

Figure 2: Static scenario

much higher or much lower than the requested one. The maximum number of probes allowed per cycle is set to five, according to the analysis performed in [2]. If after sending five probes still no decision can be reached, we maintain the previous path state; that way, if a connection is waiting for admission it will remain blocked, and if it is active it will remain active. Such criteria aims at reducing the entropy in the system.

It should be noticed that the DACME agent or the application itself should always reserve some extra bandwidth to cope with network bandwidth fluctuations, routing data and probes from other sources.

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. Therefore, the most important thing at this initial stage is to measure the different degrees of accuracy achieved by the different probe sets, and tune them for optimal performance.

In order to obtain precise values we devised a static scenario which allows us to make measurements in a controlled environment.

The static scenario we selected is presented in figure 2. When choosing it we took into account several factors. First off all, we wanted to make evident the impact of contention among stations. This was achieved by aggregating several stations located within data communication range (traffic sources and first intermediate station). Our aim was also to create a multi-hop environment, since this is typical of ad hoc networks. 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 [11]. We set up the IEEE 802.11 radio 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 simulation runs. Due to the nature of the network (static), the chosen simulation time is more than enough to provide stable, long-term measurements. 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 gradually saturate the network. 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*.

The RTS/CTS and fragmentation mechanisms are turned off. We set the packet size to 512 bytes for all sources, including probe 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 Video test traffic. In figure 3 we show how the aggregated background traffic (BG) and the test traffic change

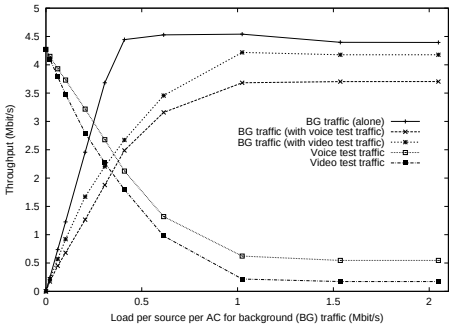

Figure 3: Measured available bandwidth varying background traffic

by varying the background load. By repeating the experiments several times with the same parameters we have a confidence level of 99% that the mean value will be within 1% of all points represented.

From figure 3 we see that if all bandwidth is available (background traffic load is null) there is not much difference between the throughput achieved with the Voice and Video test traffics. However, as the background load increases, it becomes evident that the bandwidth share obtained by Voice traffic is higher, especially in saturation. So, similarly to what was found in [9], we should set the probes always to the same priority to avoid that a source accepts high priority connections 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 3 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 under high congestion.

As the normalized standard deviation shows, the measurements tend to spread more as the probe size decreases. We conclude that 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.

3.2 Improving the estimation of available bandwidth

In this section we will improve the bandwidth estimation process since observation of the bandwidth measurement results obtained ($B_{measured}$) showed that the sample mean was a biased estimator for the available bandwidth. We now present further insight into this phenomena by presenting the discrete probability distribution for the probing process. We obtain this distribution by splitting the range of each probing process into fifteen intervals of equal length. We use three distinct background traffic levels to analyze the behavior under low, average and high congestion. Results are shown in figure 4.

The arrow/letter pairs refer to available bandwidth measurements made with real traffic, and are used as a reference for the comparison. As it can be seen, the three probability distributions are not centered around the reference values (H, A and L), which explains why their mean is superior to the real 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

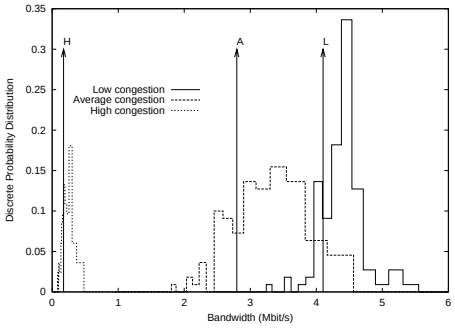

Figure 4: Discrete probability distribution for the probing process under low, average and high levels of congestion

measurements. Also, we notice that average levels of congestion tend to favor lower values for kurtosis.

We find that the mode and the median also over-estimate the available bandwidth. So, we need to correct the measured values. With this purpose we introduce a corrected estimator for available bandwidth:

$$B_e = \alpha \cdot \mu_e + \beta \cdot \sigma_e \quad (3)$$

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. Each sample consists of $B_{measured}$ values (bit/s). We apply a curve fitting algorithm to find the optimum values for parameters α, β based on least square error regression, and the degree of accuracy achieved is excellent.

4 DACME performance evaluation in a static scenario

In this section we will analyze the performance of DACME using a simulated ad hoc network environment. Such analysis required the implementation of DACME for the simulation platform of our choice, the ns-2 discrete event simulator [8]. In that platform we have changed the structure of mobile nodes so that DACME agents are able to interrupt the flow of data packets, and perform admission control tasks as desired.

In the following sections we will study the behavior of DACME in a static environment, and we will then proceed by presenting improvements and doing performance analysis in fully mobile ad hoc network environments.

4.1 Behavior with several sources and no mobility

Mobile ad hoc networks are characterized by the requirement to adapt not only to congestion changes, but also to mobility of the nodes that conform it. In this section, however, we will consider that the nodes in the ad hoc network are not moving, so that the behavior of DACME can be easily evaluated and understood.

To perform our evaluation we setup a 1900x400 squared meters scenario with 50 nodes. All nodes are equipped with an IEEE 802.11g/e interface and the radio range is of 250 meters. The position of nodes is random, and the average number of hops between nodes is 4. Concerning routing, we use static routing at this stage.

DACME agents handle five CBR traffic sources sending data at a rate of 1 Mbit/s. All packets are set to the Video Access Category. Concerning background traffic, it consists of four sources, each sending negative-exponentially distributed traffic at a rate of 50 packets per second in all four Access Categories defined in IEEE 802.11e.

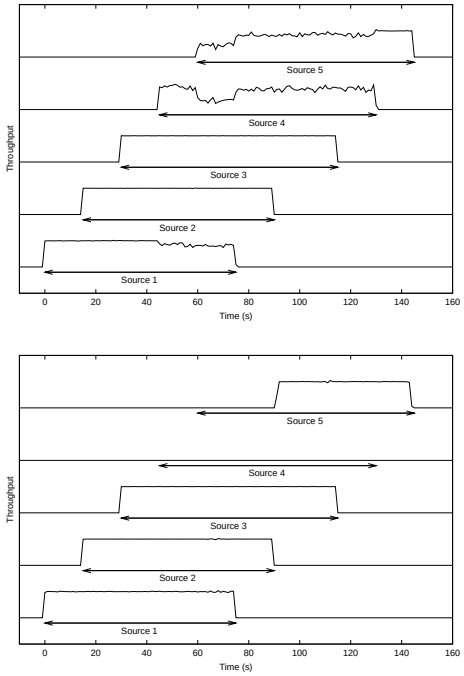

Figure 5: Throughput values for different sources with (bottom) and without (top) DACME

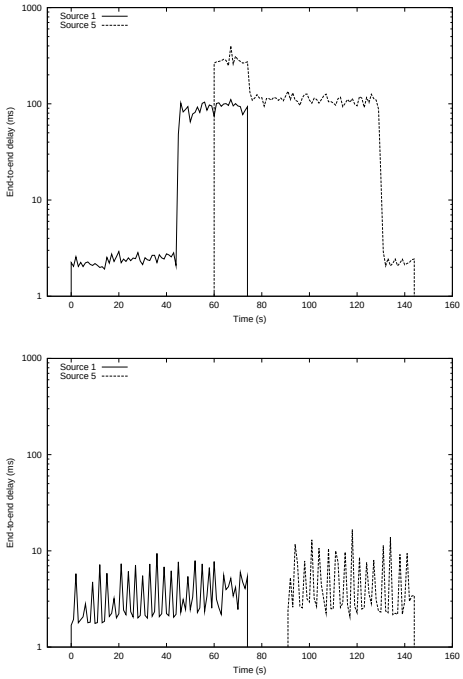

Figure 6: End-to-end delay values for sources 1 and 5 with (bottom) and without (top) DACME

Relatively to CBR sources, the first source starts at the beginning of the simulation, and a new source is started every 15 seconds, until all five sources are active. Afterwards, they are turned off in the same order as they were turned on.

In figure 5 we show the throughput for each source (maximum is 1 Mbit/s), and the arrows indicate the periods of activity for each source. We can see that when DACME is not used, sources 1, 4 and 5 suffer from throughput degradation. Such degradation would result in a quality drop if we were in presence of, for example, video traffic. What we want is to offer excellent quality to the streams allowed to flow through the ad hoc network; when that is not possible, such streams must not be allowed to access the network. We see that with DACME our aim is correctly achieved. Now, source 4 is never allowed to transmit since the DACME agent verifies that there is not enough bandwidth at any time throughout its period of activity. Relatively to source 5, we verify that it is allowed to transmit as soon as source 2 stops transmitting, which indicates that the algorithm proposed for DACME is able to react relatively quick to changing network conditions.

In terms of end-to-end delay, figure 6 shows the improvements obtained with DACME for sources 1 and 5, which are the most representative. We can see that when DACME is not used the end-to-end delay can reach very high values (close to 500 ms). Such high values are not desirable, and are related to high congestion. We observe that when using DACME the end-to-end delay values are always kept low (usually less that 10ms), though we can observe a periodic variability. Such occurrence is directly related to the probing process, periodically repeated every 3 seconds (plus jitter), and therefore cannot be avoided. We should

point out that such variability depends on the path congestion and on the application's data rate.

In terms of DACME performance we observe that DACME traffic generated between 32 and 64 kbit/s of overhead per source.

5 DACME performance evaluation in a dynamic scenario

In the previous section we considered ad hoc network stations to be static. This is not the typical behavior in ad hoc networks, though. The issue of mobility is usually handled by the routing protocol. However, we consider that the DACME agent, and therefore the application that relies on it, can benefit from obtaining awareness of the state of a path as seen by the routing protocol. Therefore, we improved the implementation of DACME so that the packets that are passed between the IP layer and the QoS measurement module (see figure 1) are not only DACME *probe/probe reply* packets, but also routing packets. The selected routing protocol for applying this method was AODV [4].

Relatively to the integration with DACME, we consider that the DACME agent should make new measurements as soon as a new path is established. This way it can assess if the new path can sustain the desired QoS.

The integration of DACME with AODV proposed in this section will be referred to as DACME-AODV from now on.

5.1 Performance in a typical mobile ad hoc network environment

In this section we will assess the effectiveness of both DACME and DACME-AODV in a typical mobile ad hoc network environment. With that purpose we use ns-2 to simulate an environment where 50 nodes move at a constant speed of 5 m/s in a 1900x400 squared meters scenario according to the random waypoint mobility model. As previously, radio interfaces are IEEE 802.11g/e enabled and the radio range is 250 meters, leading to an average of 4 hops between nodes.

Relatively to traffic, we have four background sources that are generating negative-exponentially distributed traffic in the *Video*, *Best Effort* and *Background* MAC Access Categories. Contrarily to section 4.1, we do not set the same data rate to the different MAC Access Categories. So, we set 50% of the traffic to the Video AC, and the remaining 50% is evenly divided among the Best Effort and Background ACs. Relatively to the Voice AC, the only Voice traffic belongs to DACME sources. The difference is due to the need to perform routing tasks; since routing traffic is set to the Voice AC (highest), and since Voice sources usually generate low data rates, we limit the amount of Voice traffic to avoid routing misbehavior.

Concerning the data sources under study (regulated by DACME), these consist of four video streams and three voice streams. The video sources send CBR traffic at 1 Mbit/s using 512 byte packets. Voice sources are VoIP streams simulated using a Pareto On/Off distribution with both burst and idle time set to 500 ms; the shaping factor used is 1.5 and the average data rate is of 100 kbit/s. Relatively to start and end times for the different sources, the first video source is started at the beginning of the simulation, and then every 15 seconds a new data source becomes active, alternating between voice and video sources. Each source is active for two minutes.

Figure 7 shows the improvements in terms of video goodput and voice packet losses using DACME and DACME-AODV compared to a solution where DACME is not used (turned off). We can observe that when DACME is not used the average goodput for the different video sources drops steadily with increasing congestion. By using DACME the average goodput is maintained steady because sources are only allowed to transmit if the DACME agent verifies that the available bandwidth is enough. Obviously, as the congestion level increases the amount of DACME-regulated traffic accepted into the network decreases. From figure 7 we can also observe that DACME offers great improvements in terms of voice packet losses; notice that DACME-

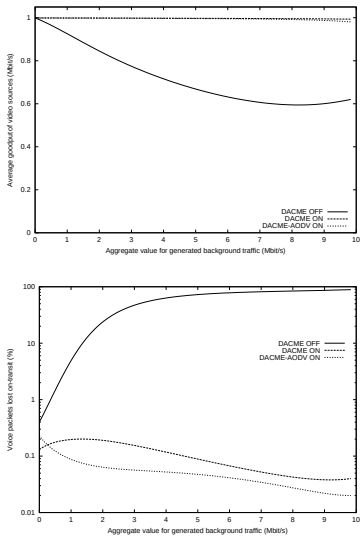

Figure 7: Improvements in terms of video goodput (top) and voice packet losses (bottom) by using DACME and DACME-AODV

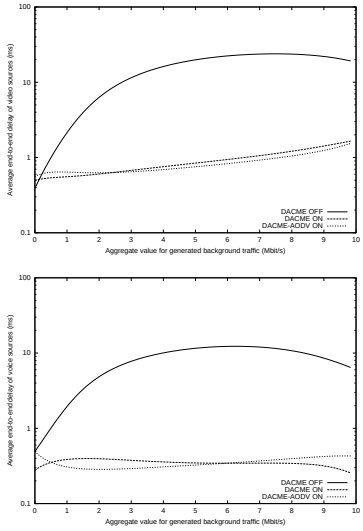

Figure 8: Average end-to-end delay values for video (top) and voice (bottom) sources.

AODV offers a better performance than the default DACME implementation.

We now proceed to evaluate the performance achieved in terms of end-to-end delay. The results are shown in figure 8. As expected, the end-to-end delay values increase with increasing background traffic. We see that when using DACME the end-to-end delay for both Voice and Video sources is greatly improved. Comparing DACME with DACME-AODV we see that, in terms of video traffic, DACME-AODV performs better than plain DACME for moderate to high congestion levels; in terms of voice traffic, DACME-AODV performs better than plain DACME for low to moderate congestion levels.

Relatively to routing traffic, simulation results show that, by maintaining the network congestion under control, DACME is able to avoid congestion-related routing misbehavior. In terms of routing overhead, this translates in up to 12 times less routing packets flowing in the network by using DACME.

The results found in this section lead us to conclude that DACME agents can improve the QoS offered to bandwidth constrained applications, and that they can clearly avoid the waste of resources by interrupting communication when the minimum QoS requirements are not met.

6 Conclusions and future work

In this paper we presented DACME, a solution for supporting bandwidth constrained applications in ad hoc networks. Our proposal uses distributed admission control techniques and imposes very few requirements on ad hoc network nodes. With our technique we expect to solve some of the problems encountered in previous proposals, focusing on ease of implementation and deployment, while not imposing any strong demands on intermediate forwarding stations.

The core of our contribution consisted of an agent that performs admission control based on probe measurements. Simulation results show that DACME is a reliable admission control solution at different levels of congestion. Overall it improves performance, avoids routing misbehavior and avoids wasting MANET resources. We observe that enhancing DACME with routing awareness further improves the support of bandwidth con-

strained applications.

As future work we plan to enhance DACME with end-to-end delay and delay jitter probing capabilities, as well as integrating DACME with a multipath-enabled version of the Dynamic Source Routing (DSR) protocol.

References

[1] Carlos T. Calafate, M. P. Malumbres, and P. Manzoni. Improving H.264 real-time streaming in MANETs through adaptive multipath routing techniques. In *IEEE Workshop on Adaptive Wireless Networks, Globecom 2004*, Dallas, Texas, USA, December 2004.

[2] Carlos T. Calafate, Pietro Manzoni, and Manuel P. Malumbres. Supporting soft real-time services in MANETs using distributed admission control and IEEE 802.11e technology. In *The 10th IEEE Symposium on Computers and Communications*, June 2005.

[3] Robert Carter and Mark Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report 1996-006, 15, 1996.

[4] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003. Work in progress.

[5] 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.

[6] 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.

[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] 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.

[10] 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.

[11] 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.