

End-to-end versus Explicit Feedback Measurement in 802.11 Networks

Manthos Kazantzidis, Mario Gerla
UCLA Computer Science WAM Lab
kazantz , gerla @ cs.ucla.edu

Abstract

Higher layer protocols in wireless networks need to dynamically adapt to observed network response. The common approach is that each session employs end-to-end monitoring to estimate quantities of interest, like delay, delay jitter and available bandwidth. A less conventional approach is to employ lower layer explicit feedback mechanisms in place or in aid of end-to-end efforts. Available bandwidth measurements are known to follow multi-modal distributions and therefore are especially difficult to measure and filter, even in wired networks. In 802.11-based multi-hop networks obtaining usable end-to-end measurements is questionable. They are affected by a combination of a large number of transient variables due to the virtual carrier sense, head of line problems on each link and mobility. Motivated by this, we are developing a network explicit feedback mechanism. Our study of this accurate network feedback architecture aids in the cost/benefit analysis of an important trade-off: deployment of network support mechanisms for transports and QoS, versus the simple, scalable and easily deployable end-to-end solution. We test our solution in: (i) multimedia adaptation and (ii) measurement based call admission. Loss rates of end-to-end adaptive video and audio connections have been more than 4 times higher than in the network feedback case. A simple call admission strategy has also proved very effective using the feedback. In our experiments it led the network to a maximal performance and stable operating point.

1. Introduction

Adaptation and call admission have often been studied in parallel because of their close relation. Call admission in wireless networks can only be handled through adaptive contracts. A quantity of interest for these functions is available bandwidth [5], [8]. Unfortunately, it is very difficult to measure and filter using end-to-end techniques even in wired networks, because the observed samples follow a multi-modal distribution [1]. Techniques for measuring related quantities, like bottleneck link bandwidth are more feasible

[2], [6], [7], [9]. For 802.11 networks the situation is more difficult since more transient parameters affect the measurements. This is due to mobility, random loss, virtual carrier sense timers and unique neighborhood head-of-line problems.

Traditionally, transport protocols rely on end-to-end measurements since this is a simple, easily deployable, scalable and transparent solution. However, it is understood that in last mile 802.11 networks end-to-end solutions may be less efficient. This motivates us to develop and study an explicit feedback architecture so that the cost/benefit analysis of the trade-off is better understood.

In this architecture we rely on lower layers to provide us with the available bandwidth values. The link layer can accurately monitor its queue utilization and the MAC layer can accurately measure a neighboring (source, destination) pair throughput. The main disadvantage of lower level support of measurements is that nodes in order to operate in the network need to provide that support. This limits scalability, ease of deployment and increases the node cost. Note that the network support requirement may be limited in local or last mile networks by use of middleware architectures and nodes in part of the path may in this way be unaware of it (e.g. [19]). But, how much would be the benefit of deploying such network support versus its cost in networks with wireless links? In setting the grounds to answer this question we explore a lower level support architecture. We then use the explicit measurements for a) multimedia adaptation b) measurement based call admission.

We focus the study on asynchronous 802.11 networks. We find this timely also because of additional pressure for 802.11 to support voice and multimedia calls in the infrastructure-less mode (DCF) due to Bluetooth advent. The DCF mode is the only mode currently supported by 802.11 in the infrastructure-less ad-hoc network and is also the only mode currently implemented in Wavelan even in infrastructure mode (access point support). The 802.11 standard allows for link performance measurements. Here we use the fragment acknowledgement and link-fail message used for unicast traffic in DCF to produce link-by-link (source, destination) pair throughput measurement. Link layer

queue utilization measurements are then combined to produce a permissible throughput measurement, in a suitable way for a wireless multi-hop network (see also contention models in [10]). Propagation of the permissible throughput values can be accomplished using piggybacking to routing messages or as a separate application. The former constrains the two mechanisms, with possibly different cost functions, to work on relevant time scales. Their propagation may be distance vector based, link state based or on demand. In this paper we are using distance vector propagation. In [18] we have developed an event-driven propagation based on AODV [22].

Link bandwidth measurement has been attempted in the past for networks with wireless links (including multi-hop) using a variety of approaches; link or end-to-end, passive or active. However, there is no study that is either a) based on an implementable approach b) has been experimented with in multi-hop networks c) uses MAC layer and link layer features and d) provides a network layer measurement mechanism. In [11] a maximum available bandwidth measurement per node is proposed for QoS support in MANETs, but is only expressed in a formula. Translation of the formula to a distributed algorithm would result in a complex scheme because it would require up-to-date neighborhood knowledge.

In the next section we present our approach for 802.11 links and provide an optional network layer based support (instead of L2) implementation. In III we present the simulations with end-to-end and network feedback and call admission. We conclude in IV.

2. Permissible Throughput Measurement

In multi-hop networks we measure the link performance for each (source, destination) link pair. We define our permissible throughput using a typical residual bandwidth measurement as

$$P(src, dest) = (1 - u) * Throughput(src, dest) \quad (1)$$

where u is the L2 queue utilization.

2.1 Link Throughput Measurement

From (1) in order to calculate the permissible throughput we first need to estimate the link throughput. Each node passively estimates its throughput to each neighbor. The throughput seen by one packet of S bits can be calculated as

$$Throughput_{packet} = \frac{S}{T_{ACK\ reception} - T_{transmission}}$$

where, $T_{ACK\ reception}$ = timestamp of ACK reception

and, $T_{transmission}$ = timestamp of packet transmission

The throughput we measure using this formula includes fading, internal, external noise and contention. If measured above the link layer, it normally contains (and cannot distinguish) queuing time, 802.11 related overhead and actual bit transmission time. In order to understand what we are measuring with the above equation, let us express it in 802.11 terms:

$$Throughput_{packet} = \frac{S}{t_q + (t_s + t_{CA} + t_{overhead}) * R + \sum_{r=1}^R TB_r}$$

where t_q is the L2 queuing time, t_s the transmission time of the S bits, t_{CA} the collision avoidance phase time, $t_{overhead}$ the control overhead time (e.g. RTS/CTS, ACK, Header, 4 propagation delays), R the necessary retransmissions and TB_r the back-off time for retransmission r . This formula reveals some undesirable characteristics that are however common to measurements

- Packet size dependence
- High variance due to random per packet effects.
- Queuing delay has to be properly factored out, if transmission time is marked at or above the link layer

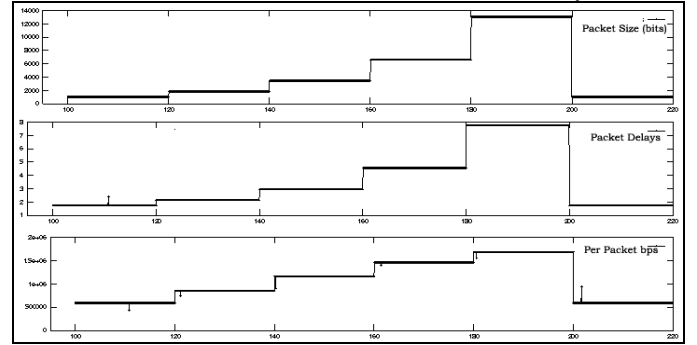


Fig. 1: (top) a CBR source uses different packet. (mid) packet delays (bottom) the throughput observed per packet.

On the other hand, it has significant benefits compared to end-to-end metrics since those problems are only relevant to a single link.

We can identify three distinct areas of operation. A) low load - in this area we have very rare retransmissions, mainly due to random channel errors, the collision avoidance phase is rarely prolonged and the overhead is almost constant and paid once per frame. B) medium load - retransmissions are still rare and the variance in the measurements is introduced mainly by variation of the collision avoidance phase duration. C) high load - collisions and retransmissions, cause high delays and dominate the throughput measurement. Strong hidden terminal effects further complicate filtering in case of multi-hop networks. Detailed specific analyses on 802.11 throughput and delay can be found in [15] and [14].

In fig.1 we show the effect of the packet size on the throughput measurement by simulation. A different packet size is used for every 20 seconds of a one hop connection over a 2Mbps link. The throughput measured as above (with a window of a single packet) starts at barely over 500Kbps (25% of link bandwidth) for a payload of 100 bytes! The different packet sizes are a significant source of noise to the measurements.

When the measurement function is in the MAC layer the per packet overhead may be known (see calculation below) and can be factored out. If not, in order to limit the payload dependence, instead of complicating our design by e.g. using

packet size hashing, we simply deduct a constant c from all delay measurements. c should ideally be the packet overhead time. An a priori global estimated c is a very simple and effective solution for our purpose. Let us define as t_{overhead} an average overhead time of a transmission that succeeds before the 802.11 virtual carrier sense timers expire. Then by using a c equal to t_{overhead} our throughput measurements will be immune to packet size for those cases. At the same time the measurement will be more pessimistic for medium and higher loads (see areas of operation above). An example calculation of c is shown below:

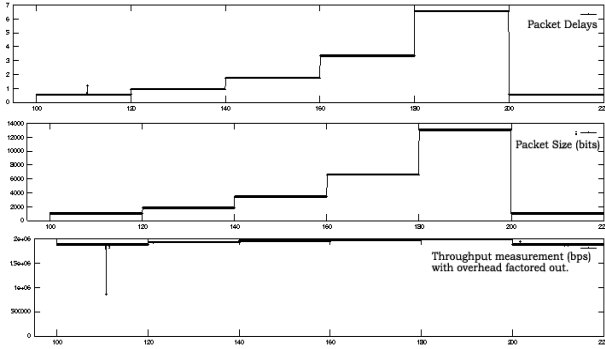


Fig. 2: Under low load the throughput measurement becomes independent of packet size by subtracting a constant c

$$c \approx t_{\text{overhead}} \approx t_{\text{RTS}} + t_{\text{CTS}} + t_{\text{HDR}} + t_{\text{waitACK}} + t_{\text{transmissionoverhead}}$$

It includes the transmission times of the overhead bits, RTS, CTS, Header, ACK, the hold for ACK time (see also [12]) the extra propagation times and SIFS for the RTS and the CTS and one DIFS that the transmission will be delayed etc. (see also [14]). For our simulations we use $c=1200 \mu\text{sec}$ as we calculated using the equation and the default values of the 802.11 Glomosim implementation [20]. The next simulation shows that, in a low load link with no outside contention, the measured throughput now –after subtracting c - is negligibly affected by packet size. Note that deduction of c helps in factoring out packet size dependence but the size of the competing packets is still factored into the measurement, because the collision avoidance phase depends on competing packet sizes. (In fact, [4] uses this observation for a packet size adaptation method.) In general by subtracting the constant c we have limited significantly the packet size dependence for our purposes.

We use a packet window in order to increase statistical robustness of the measurements. A packet window of 32 samples (fragments) is adequate to produce fast enough, noise immune measurements. For an illustration of this see fig. 3 where the high variance of per packets measurements is filtered using the window. The window idle time and window duration are also calculated per window in order to produce the link utilization factor and finally the permissible throughput measurement as

$$\frac{\text{idle time in window}}{\text{window duration}} * \text{Throughput measured} \cdot$$

The use of the packet window, in the case that the function resides in the network layer, requires special handling though. Queuing delays should be appropriately factored out. When a packet has waited in queue, the correct delay contribution can be calculated at the time that the corresponding ACK (or NACK) has been received as $now - T_{\text{previous ACK reception}}$.

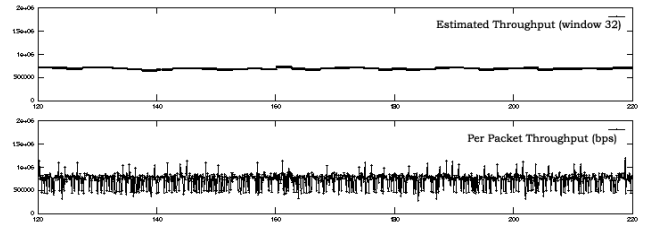


Fig. 3: Window operation

When it has not suffered any queuing the contributed delay is $now - T_{\text{arrival}}$. In the following section we provide and discuss the network layer algorithm that implement these measurements.

One sample is collected per window and it is directed to a digital low pass filter. A Tustin approximation [17] is suitable for this digital sampling of throughput.

2.2 Network Layer Implementation

In this section we propose an implementation that requires minimal support and knowledge of the link layer, and is suitable for the network layer. Errors are introduced in the measurement due to unknown parameters for the network layer. Most errors however can be filtered out or tolerated. One of the main issues dealt with here is the subtraction of queuing delays in the window operation (see fig.4). In order to simplify the implementation description we first introduce the assumption that a single notification per packet is provided, and later we discuss how to deal and/or detect violations of this assumption (802.11 may have duplicate ACKs and L2 drops packets that cause overflow without notification).

The necessary variables and data structures are

- *LastPacketTimestamp*[WINDOW_SIZE] : a vector used to keep the timestamp when a packet leaves the network layer
- *LastPacketPayload*[WINDOW_SIZE]: a vector used to keep the useful bits of the above packets
- *VQ*: Contains the number of packets sent. It is incremented (modulo at least WINDOW_SIZE) each time a packet is sent.
- *NTF*: Contains the number of notifications received. It is incremented (modulo at least WINDOW_SIZE) each time a notification is received,
- *PrevNTFTimestamp*: Contains the timestamp of the last notification,
- *SampleDelays*[RECEIVERS]: Contains the sum of delays measured for packets to RECEIVER,

- *SampleBits[RECEIVERS]*: Contains the sum of bits for the above packets.

We also use *Window_duration*: the duration of the current window and *Idle_time*: the idle time in the current window for utilization measurements.

```

ON_PACKET_SENT()
VQ++;
LastPacketPayload[VQ]= <packet bits>;
LastPacketTimestamp[VQ]= now;
If Queue was empty =(VQ - NTF==1) {
  Channel has been idle since last notification
  Idle_time += now - PrevNTFTimestamp;
  Window_duration+=now-PrevNTFTimestamp;
}

ON_NOTIFICATION_RECEIVED()
NTF++;
If (ACK received)

SampleBits[Notif.source]+=LastPacketPayload[NTF];
This_Packet_Suffered_Queueing =
  (LastPacketTimestamp[NTF]< prevNTFTimestamp);
If (This_Packet_Suffered_Queueing)
  Delay = Now - prevNTFTimestamp - C;
Else
  Delay= Now-LastPacketTimestamp[NTF]-C;
SampleDelays[Notif.Source] += Delay - C;
Window_duration += Delay;

MEASURE()
For All Measured Receivers R:
  Link_Throughput=SampleBits[R]/SampleDelays[R];
  If ( Test_Measurement() ) {
    PermissibleT[R]= idle_time/Window_duration *
      Link_Throughput;
    Update_Tables();
  }

```

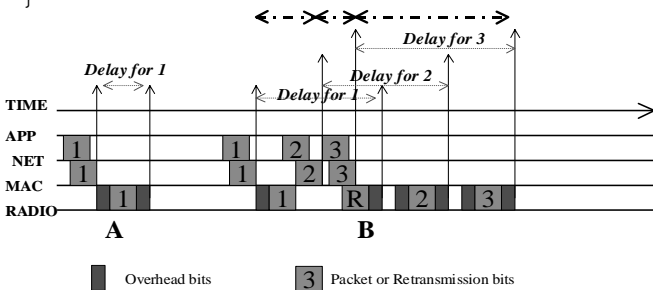


Fig. 4: An example packet arrival/ acknowledgement sequence. The network calculates the contributed delays by using the notification series to emulate the link queue.

Note that the implementation is simple and does not require significant memory. The per-destination vectors can be constant in size regardless of the size of the network while the packet payload and timestamp vectors depend on the window size. At worst, information that cannot be held into the data structures can be discarded since this just deprives the measurement of the current sample.

In order to make the algorithm work efficiently outside the link layer we need to be robust in situations where a notification is sent more than once or when a notification is not sent at all. The former is rare but may occur in 802.11(duplicate acknowledgement), while the latter occurs

when the link layer queues are full. Both of these situations are detectable but only after few packets. Even if sanity checks do not fail for erroneous measurements they can be eventually filtered out.

For the propagation of values, we use a distance vector based propagation that piggybacks its values to existing periodic routing broadcasts. In the propagation we are interested in discovering the minimum permissible throughput along the path (bottleneck link). In order to allow for efficient updating of the tables, we use both the measurement (expressed as negative permissible throughput as a cost) and the hop count to the discovered bottleneck link. This is necessary to detect bottleneck point changes and cancellations. In [18] we extended AODV to support event-driven on demand propagation of measurements.

3. Simulations

3.1 Correctness

In this section we use GlomoSim [20] for two simple simulations that show the correctness of our measurements. We show the measured throughput, the permissible throughput and the consumed, defined as the difference between the two. The latter must always match the source throughput if the measurement is correct. The environment is the same as for the remaining experiments (1Mbps links).

In the first experiment we use a CBR connection of 182Kbps rate using packets of 480 bytes. In fig. 5 we see the throughput measurement to be close to 1Mbps and can even distinguish the periodic broadcast messages in the throughput measurement. Combining with the utilization measurement we get a permissible throughput that is 200Kbps less than the measured throughput. This shows that our measurement is very accurate under low load conditions (the 18Kbps difference is transport overhead, RTP header etc.)

The above experiment was with one hop CBR traffic and consequently no link queuing. In the next experiment we use a video source to examine our VBR case. In fig. 6 we see the source rate averaged every 32 packets. The video traces are taken from the Star Wars trailer [21] encoded in Intel's implementation of H.263 at 100Kbps target rate.

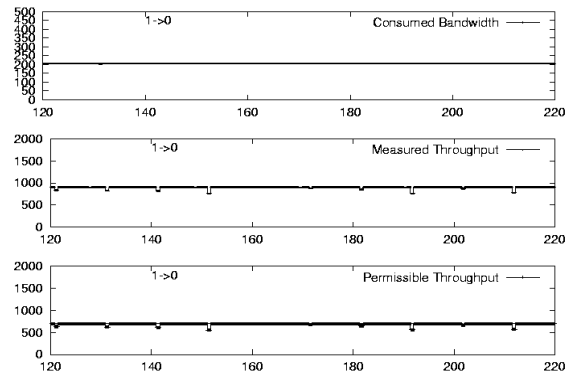


Fig. 5 Exact measurements for CBR

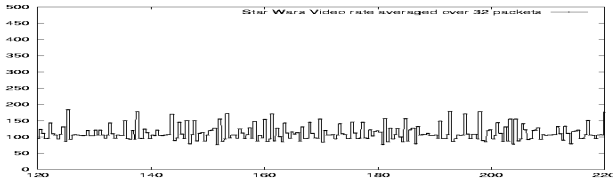


Fig. 6 The VBR source rates in time averaged over 32 packets

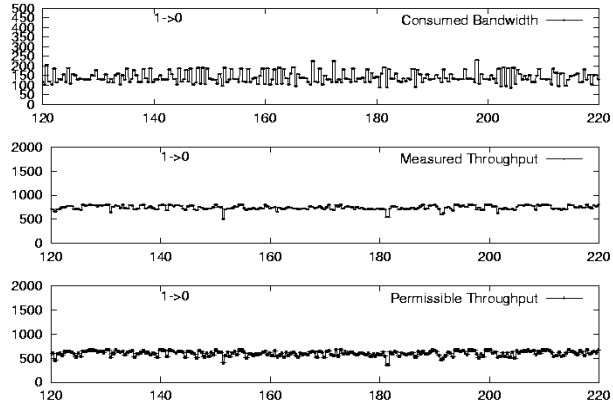


Fig.7 Accurate measurements for VBR traffic.

As we see in fig. 7 the measurement has very similar magnitude and shape and the consumed bandwidth follows accurately the offered traffic. In general these simulations show that we have a measurement accurate, fast and robust. We go on exploring the behavior of such a feedback, and show that its attributes are good enough to perform both adaptation of multimedia sources and call acceptance control.

3.2 Adaptive Multimedia

In this section we use video and audio adaptive connections adapting either to periodic 1 second end-to-end feedback containing the RTP loss, or to the network feedback. The video connections are able to adapt to 5 rates of an H263 encoded video stream. The traces have been gathered from the Star Wars trailer clip encoded in 256, 128, 80, 64 and 48 Kbps average target rates. The smoothing process is limited to one frame time and uniformly spreads the frame bytes into 200 byte payloads into the inter-frame interval. The audio is encoded at 9 encodings of 256 to 8 Kbps (as in [3]). The initial rate is set to a low rate of 64Kbps. When adaptation is based on network feedback the server tries to match half of the reported permissible throughput by choosing the appropriate rate for the CODEC. In general the two mechanisms use values of different nature. We have attempted to use as similar add/drop strategies as possible. In this comparison, we are downgrading the CODEC rate proportionally to the loss rate observed. For example, if we observe a loss of 60% while the maximum tolerable loss is 15% we will attempt to downgrade 4 layers. This is a pessimistic approach that avoids over-sending and limits the loss rates further for the end-to-end case, resulting in a more

fair comparison. The emphasis is in the measurement rather than the add/drop strategy in this way. As we see, even so, the end-to-end loss rates remain high.

In a mesh topology and connectivity of 36 (6 by 6) nodes we distribute connection pairs throughout the network. The hop distribution is (20%, 25%, 25%, 15%, 10%, 5%) for 1 to 6 hops respectively. Each experiment has 12 connections either active throughout the experiment or starting/ending 20, 40 or 60 seconds late/early. 4 experiments that contain adaptive audio and video connections are reported. The aggregate loss percentages are shown in fig. 8 for the network and end-to-end cases. In most cases, network stability has been maintained, but only the network mechanism managed perceptually acceptable loss rates (ranging from 1 to 8 percent). The end-to-end case in a few cases has even failed to uphold its maximum loss rate threshold (15%).

3.3 Call Admission

Next, we use similar scenarios but using higher rate connections so that all connections cannot be allowed in the network even with adaptation. We also use only video connections since passive measurements with VBR sources are more challenging. We do this to test whether the stability of the network will be maintained by a simple call acceptance strategy. On these experiments, the adaptive connections request a rate before entering the network or when a layer addition or drop is performed. Once they start they periodically query the path available bandwidth (through an API). They initially request the rate of the second lower rate (64Kbps plus their overhead). The network allows them to enter if there is at least double the bandwidth requested in order to maintain stability and fairness.

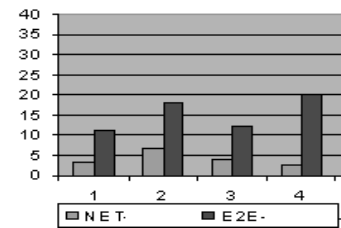


Fig. 8 Overall loss rates (%) per experiment

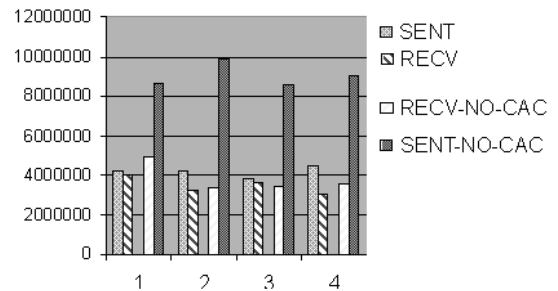


Fig. 9 Total bytes sent vs total bytes received with call acceptance and without.

References

The experiments show clearly that even a simple strategy and implementation of call acceptance based on our measurement can significantly improve the overall network throughput. Fig. 9 shows that the amount of bytes received in the network is equivalent between the cases of call acceptance or no call acceptance. This, and the fact that loss rates for the call-acceptance case adaptive connections were small, shows that the call acceptance process found a good and stable operation point. The network resources were correctly utilized close to their limits. Based on permissible throughput and its propagation and working in parallel with source adaptation, it was precise in finding the network limits, since it managed to sent as much data as would be received.

In all of the above experiments we see that the network feedback mechanism provides strong control capabilities at the end points of the network. The values of permissible throughput propagated are robust and accurate enough for performing not only source adaptation but also call acceptance control.

4. Conclusions

We have successfully developed an architecture that supports accurate permissible throughput explicit feedback to multimedia transports and call admission applications. The motivation is for the architecture to base a cost/benefit analysis of end-to-end measurements versus lower level explicit feedback in last mile 802.11 networks. We also dealt with deployment issues by providing a network layer implementation of the link-by-link measurement. We tackled common measurement problems effectively, by dealing with them on a link-by-link source-destination specific basis. Only simple techniques were necessary to produce a statistically robust, correct and interpretable measurement, in contrast to common end-to-end experience. After making available the bottleneck permissible throughputs at endpoints of a multi-hop ad-hoc network, we compared the network feedback adaptation to end-to-end and found it four times more effective in reducing the RTP loss rates in our experiments. Finally, we used them to direct a simple call acceptance strategy and showed that it very accurately found a stable, high throughput and low loss operating point. Therefore, single link measurements and feedback present a promising approach for soft QoS support in multi-hop ad-hoc networks.

Scalable, simple, deployable and cost-effective end-to-end approaches should be further studied. Direct end-to-end measurements, particularly other than loss observation and 'trial-and-error' based approached, should be evaluated in accuracy and resulting network goodput in networks with wireless links. The architecture studied here may provide a reference for such a comparison.

- [1] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure? In Proceedings of IEEE INFOCOM, April 2001.
- [2] Kevin Lai and Mary Baker, Measuring Bandwidth, *IEEE INFOCOM '99*
- [3] M. Kazantzidis, L. Wang, and M. Gerla, On Fairness and Efficiency of Adaptive Audio Application Layers for Multihop Wireless Networks *IEEE MOMUC'99*
- [4] M. Kazantzidis, I. Slain, T.-W. Chen, M. Gerla, Y. Romanenko Experiments on QoS Adaptation for Improving Enduser Speech Perception over Multihop Wireless Networks *IEEE ICC'99*
- [5] Carter, R.L.; Crovella, M.E. Server selection using dynamic path characterization in wide-area networks. *IEEE INFOCOM '97*
- [6] Berwin A. Turlach. Bandwidth Selection in Kernel Density Estimation: A Review. *CORE and Institut de Statistique*
- [7] Bolot, J.-C. End-to-end packet delay and loss behavior *SIGCOMM '93*.
- [8] Paxson, V. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, vol.7, (no.3) June 1999.
- [9] Net-timer: <http://mosquitonet.stanford.edu/~laik/projects/nettimer/>
- [10] T. Nandagopal, T. Kim, X. Gao and V. Bharghavan, Achieving MAC Layer Fairness in Wireless Packet Networks. *Mobicom 2000*
- [11] Cansever, D.H.; Michelson, A.M.; Levesque, A.H. Quality of service support in mobile ad-hoc IP networks. *MILCOM 1999*
- [12] IEEE Std 802.11 - Wireless LAN Medium Access Control and Physical Layer specifications
- [13] H. Schulzrinne et al., RTP: A Transport Protocol for Real-Time Applications, *RFC 1889* Jan 1996
- [14] Hadzi-Velkov, Z.; Gavrilovska, L. Influence of hidden terminals over the performance of the MAC protocol for IEEE 802.11 wireless LANs. ITG-Fachbericht, (no.157), (*European Wireless '99*)
- [15] Crow, B.P.; Widjaja, I.; Kim, J.G.; Sakai, P. Investigation of the IEEE 802.11 medium access control (MAC) sublayer functions. Proceedings *IEEE INFOCOM '97*
- [16] Bianchi, G. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE JSAC* March 2000. p.535-47.
- [17] D. Poulton, J. Oksman, "Digital filters for non uniformly sampled signals," *NORSIG 2000, Nordic Signal Processing Symposium*, Vildmarkshotellet Kolmarden, Sweden, pp. 421-424, June 2000
- [18] M Kazantzidis, M Gerla, "Permissible Throughput Network Feedback for Adaptive Multimedia in AODV MANETs" *ICC 2001*
- [19] R. Bagrodia, M. Gerla, S. Lu, R. Meyer, D. Valentino, and L. Zhang. "Supporting Nomadic Healers" *UCLA Technical Report 200021*, 2000.
- [20] M. Takai L, Bajaj, R, Ahuja, R, Bagrodia and M. Gerla, "GloMoSim: A Scalable Network Simulation environment" *Technical report 990027*, UCLA, Computer Science Department, 1999.
- [21] www.starwars.com/episode-i/video/epitrailer/epitrailer2.html
- [22] Perkins, C.E.; Royer, E.M. *Ad-hoc on-demand distance vector routing*. Proceedings WMCSA'99.