

Impact of Upper Layer Adaptation on End-to-end Delay Management in Wireless Ad Hoc Networks

Wenbo He Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Siebel Center, 201 N. Goodwin Ave.
Urbana, IL 61801

Abstract

*A good amount of research has been developed to support QoS issues in IEEE 802.11 ad hoc networks, such as QoS routing, MAC layer QoS support, and cross-layer QoS design. However, QoS solution at upper layers for real-time multimedia applications is overlooked. This paper investigates impact of the adaptation mechanisms at application layer and middleware layer on end-to-end delay management. Upper layer adaptation is a localized method with small overhead, and the adaptation mechanism is hardware independent. The application layer adaptor is to dynamically change the requirement levels based on end-to-end QoS measurement. The middleware adaptor is to dynamically adjust the service classes for applications by feedback control theory. We use real IEEE 802.11 ad hoc network environment to evaluate the impact of upper layer adaptation, and conclude that the upper layer adaptation for end-to-end delay is efficient in many scenarios, but it is not enough for contention scenarios, where lower layer scheduling should be adopted.*¹

1. Introduction

An ad hoc network is a dynamic multi-hop wireless network that consists of a small group of hosts. Multiple hosts share the wireless link capacity in the vicinity of each other. There exists great uncertainty about the available bandwidth at each hop. Therefore, it is hard to provide guarantee on end-to-end delay for real time audio applications over ad hoc networks. In this paper, we address the end-to-end delay and jitter control for audio applications. We adopt upper layer adaptation to provide more flexibility for applications, and use feedback control method to dynamically change priority of real time multimedia flows.

In this paper, we aim to take the advantage of upper layers adaptation. A real time multimedia application usually has QoS requirements on two quantifiable metrics, end-to-end latency and its variance (jitter). Depending on network conditions, applications may have multi-level QoS requirements. For example, when we make phone call using VoIP technology, we prefer to have acceptable voice quality rather than drop the phone line, if the network condition is bad. On the other hand, if the network condition is good, we prefer high voice quality as much as possible. We deploy application layer adaptation to select an acceptable end-to-end delay requirement from multiple delay levels according to the network traffic load. With the selected end-to-end delay requirement, we utilize feedback control in middleware to prioritize multimedia packets. Therefore, the multimedia applications strive to meet the delay requirement. Our upper layer adaptation requires the support of network layer service differentiation.

This paper addresses the delay control problem, not the utilization problem. Under the same utilization level, a real-time multimedia traffic is able to gain more bandwidth, therefore smaller delay. So we do not show the increases of the total utilization in this paper, but address the end-to-end delay of multimedia traffic.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 shows the framework of upper layers adaptation mechanism, and illustrates the function of each component in the framework. Section 4 shows the experimental results and the performance evaluation. Section 5 concludes the paper.

2. Related Work

The existing methods to deal with QoS issues in wireless networks are mainly focused on MAC layer scheduling. [18], [2], [4], [17] study the MAC layer scheduling under IEEE 802.11 DCF to provide differentiated delays. The evaluation of the IEEE 802.11e draft has been done in [5]

¹ This work is supported by Motorola URP and Vodafone Fellowship

[9]. However, the MAC layer solutions are not appropriated to support end-to-end application-specific QoS requirements. [10] [3] address the end-to-end QoS support in wireless network by adapting traditional QoS model for wireless networks. Both approaches use per-application admission, which causes the poor scalability and suffers improper admission control due to the time varying characteristics of the highly dynamic wireless environments. Since no single layer in the protocol stack can guarantee end-to-end QoS, cross-layer design is necessary to achieve the end-to-end QoS support. [21] tries to maximize system-wide utility while meeting all resource constraints by local adaptation. [14] addresses the middleware adaptation for end-to-end delay in ad hoc network. [15] [19] are pretty much focused on providing service differentiation or fairness scheduling at MAC layer. The upper layer QoS support is lacking of attention. However, the right place to map the end-to-end QoS requirements to performance metrics is the upper layers (application layer and middleware layer).

To control the performance measurement such as end-to-end delay and jitter for real-time multimedia applications, we adopt feedback control theory in middleware design. Control theory has been applied to many areas. The metrics being controlled could be utilization, delay, throughput and power consumption. In real-time domain, feedback control theory was introduced as in [13] [11] [16].

To enforce the service differentiation, we adopt waiting time priority (WTP) scheduler at network layer. [6] introduces the proportional differentiation model in wireline networks, which offers relative differentiated services between a small number of service classes with different service quality in queueing delay and packet losses. [20] addresses the proportional service differentiation in terms of throughput in WLANs. The proportional delay differentiation (PDD) model has the property that for any pair of classes i and j , in the time interval $(t, t + \tau)$,

$$\frac{\bar{d}_i(t, t + \tau)}{\bar{d}_j(t, t + \tau)} = \frac{\delta_i}{\delta_j} \quad (1)$$

where $\bar{d}_i(t, t + \tau)$ is the average delay for class i , and δ_i is the delay differentiation parameter (DDP) for class i . The gap between absolute QoS guarantee and relative differentiation is bridged by dynamic service class selection [7]. In proportional differentiation, even though the service quality of each class may vary with traffic loads change, the quality ratio between each pair of classes remains constant. And such a quality ratio can be achieved by setting the service differentiation parameters. To meet the absolute service quality requirement of an application, the application can dynamically select the appropriate service classes. This is the basic idea of our network layer scheduler design.

3. Upper Layer Adaptation Framework

Figure 1 shows the framework of our upper layer adaptation. Five components are involved: network layer scheduler, delay monitor, classifier, priority adaptor and requirement adaptor.

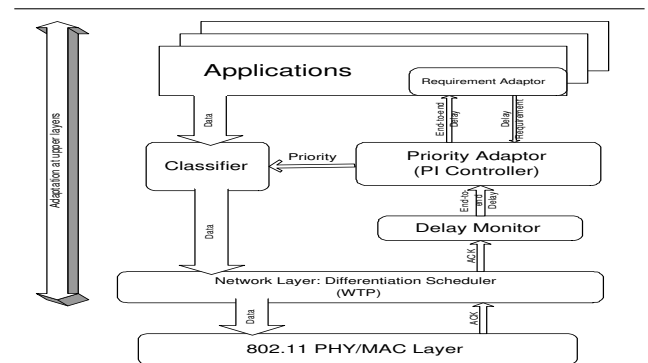


Figure 1. Network stack of the cross-layer design scheme

3.1. Differentiation Scheduler at Network Layer

In this paper, we use waiting time priority (WTP) scheduler to approximate PDD model. In the WTP scheduling algorithm, the *normalized head waiting time* of class i at time t is defined as $\tilde{w}_i(t) = w_i(t)/\delta_i$, where $w_i(t)$ is the actual waiting time of the head packet in service class i . The WTP scheduler selects the packet from class j with the maximum normalized head waiting time $j = \arg \max_{i \in B(t)} \tilde{w}_i(t)$ to serve, where $B(t)$ stands for the set of classes at time t waiting to be served. [8] shows that the waiting time priority (WTP) scheduler is one of the algorithms to approximate PDD.

In our implementation of WTP scheduler, a packet in service class i is attached with a class parameter p_i , which is interpreted as priority of service class i . p_i is determined dynamically by the middleware. When a packet needs to be transmitted, the network layer scheduler selects the *packet* with the maximum normalized waiting time

$$packet = \arg \max_{p \in P(t)} w_p(t) p_i \quad (2)$$

where $P(t)$ is the set of packets waiting to be served in time t and $w_p(t)$ is the waiting time of a packet p . The application packet with a larger priority p_i is assigned more service. By increasing the priority of a multimedia flow, the delay ratio of the multimedia flow to best-effort flows will

decrease. Therefore, we can control the average end-to-end delay by adjusting the priority of the multimedia flow.

3.2. Delay Monitor

The delay monitor measures the average round trip delay incurred to deliver multimedia packets in the ad-hoc network for each application. The end-to-end latency contains the delay introduced by traversing the entire protocol stack at the end nodes. The delay manager is placed in middleware is because that the end-to-end latency caused by traversing the upper layers (network and above) at the end nodes can be ignored. The sender attaches timestamps when sending the packets to the destination. As the sender gets ACK from the destination, it retrieves the sending timestamp, and compares it with the current timestamp, so that a sender can obtain the round-trip delay d_i for packet i . We take an average of N round-trip delay measurements (d_1, d_2, \dots, d_N) , and have $d_{avg} = \frac{1}{2N} \sum_{i=1}^N d_i$ as the measured value to estimate end-to-end delay, which is used by the *Priority Adaptor* to update the service priority appropriately.

3.3. Classifier

The classifier in the middleware determines the service classes for packets according to their priorities. The goal of the *Classifier* is to map the priorities to service classes in order to keep a small number of service classes. Each service class is represented by the class parameter, which is a number obtained by rounding up the priority in a certain range. The possible priorities generated by *Priority Adaptor* can be divided into L ranges, R_1, R_2, \dots, R_L . If the priority attached to a packet by the *Priority Adaptor* is in the range R_i , the service parameter assigned to the packet will be represented by the largest priority in the range R_i . Though the priorities can take a large range of values, the number of service classes can be small. Usually we choose a small number of service classes for easy deployment and efficient network management.

3.4. Requirement Adaptor

The multimedia application is intended to be real-time and interactive, which means QoS requirement on end-to-end latency. The variance of end-to-end delay implies jitter in the multimedia presentation. A multimedia application usually has QoS requirements on two quantifiable metrics, end-to-end latency and its variance (jitter). For example, in telephony, one-way delay requirement is from 25ms to 400ms, depending on the requirement of voice quality and existence of echo canceller, and jitter is set to be 20ms.

When delay of an application is small, the application requires more bandwidth, and the delay is more sensitive to bandwidth change on wireless links. Intuitively, minimizing end-to-end latency is at the cost of jitter. However, to get good qualities of the multimedia applications, both requirements on end-to-end delay and jitter should be satisfied.

Higher QoS requirement implies better quality of multimedia presentation. In this paper, the QoS requirement is given by the expectation of the end-to-end delay. Users prefer the best quality of the multimedia delivery as network load condition permits. When the best quality is not guaranteed, users may prefer to tolerate quality degradation a little bit rather than drop the applications. In our model, an application defines multiple quality levels in the form of acceptable delay expectations. $D = \{d_1 < d_2 < \dots < d_m\}$ denotes the set of the multiple delay levels, and j_{req} denotes required jitter. The goal of the application layer QoS adaptation is in charge of selecting a delay expectation d_{req} from the set D according to network load condition. Different delay expectations in set D result in different audio or video qualities. Since deterministic end-to-end QoS guarantees over contention based (e.g. IEEE 802.11) wireless links are impossible, we use statistical criteria to select the delay requirement.

With d_{req} and j_{req} , we can define d_{min} and d_{max} as the lower and upper bounds of the desired delay range. We desire that end-to-end delay of multimedia packets falls into the range $[d_{min}, d_{max}]$, where $d_{min} = d_{req} - \frac{j_{req}}{2}$ and $d_{max} = d_{req} + \frac{j_{req}}{2}$. The probability $P = Prob(d \in [d_{min}, d_{max}])$ means the probability that QoS requirement is satisfied, which indicates the quality of the playback of multimedia application. Multimedia applications can tolerate a small percent, say 5%, of packets which fall out of the delay range $[d_{min}, d_{max}]$. The requirement adaptation at application layer relaxes the requirement on the delay expectation if $P < 95\%$. In our application model, if given d_{req}, j_{req} and D , the chosen delay requirement should satisfy

$$d_{req} = \min \{d \in D \mid Prob(d \in [d_{min}, d_{max}]) > 95\%\} \quad (3)$$

By equation (3), the application layer selects the highest possible requirement from the requirement set D to adapt the network load.

3.5. Priority Adaptor

To design the *Priority Adaptor*, we need first to establish the open-loop dynamic model for the mapping between priority and end-to-end delay. Using the standard system identification techniques [12], we can determine the model and

relationship between priority and end-to-end delay. After we understand the control model, we design the PI (Proportional and Integral) controller to adjust the priorities of the application to control the end-to-end delay around the required delay level.

The goal of middleware adaptation is to dynamically adjust the priority of a multimedia application, thus make the multimedia application strive to meet the end-to-end delay requirement. In this section, we focus on the design of middleware *Priority Adaptor*. PID controller is a widely used controller for dynamic systems to maintain the output level at the pre-determined value (reference value), where PID stands for Proportional, Integral, Derivative. A proportional (P) controller has the effect of reducing the rise time and decreasing the steady-state error. An integral (I) controller has the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative (D) controller has the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. However, if the system noise is large, the derivative controller will decrease the stability of the system. For the system under investigation (multimedia over wireless ad hoc networks), both system noise and the measurement noise are large. The system noise is due to the random workload in the adhoc network, and the nature of randomized algorithm in MAC layer protocols. The measurement noise comes from the measured data we get from the delay monitor. The measurement noise of the system is caused by a large range of round trip delay in the wireless ad hoc networks. Due to the large noise in the system, the D controller will introduce the undesired oscillation to the system. So we choose the combination of PI controller and do not consider D controller in the middleware adaptation.

To give a good end-to-end delay performance of the system, the design goals of the closed-loop system are shown in Table 1.

Desired System Properties	Controller Design Criteria
Stability	Poles within the unit-circle
Accuracy of control (Zero steady-state error)	Use of Integral (I) control
Fast Response and Less oscillation	Reasonable parameters of PI controller

Table 1. The desired properties and the controller design criteria

Figure 2 shows the whole control loop including *Priority Adaptor* (C), the open loop wireless multihop system

(G), and the *Delay Monitor* (M). To calculate all parameters of the *Priority Adaptor* C as a PI controller, we need to determine the wireless end-to-end system as an open loop system model G . The *Delay Monitor* detects the end-to-end delay, $d(k)$ of the system. Then the *Priority Adaptor* compares it with the delay requirement ref_{delay} given by the application, and adjusts the priority of the packets to be sent of the multimedia flow. Let $e(k) = d(k) - ref_{delay}$, where ref_{delay} is the desired level of end-to-end delay. If $e(k)$ can be kept around zero, the end-to-end delay of the system will be stabilized around the desired level.

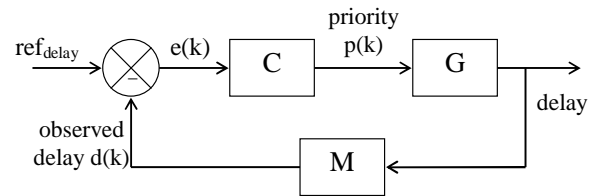


Figure 2. The block diagram of the closed-loop control system for end-to-end delay

We obtain the parameters of the *Priority Adaptor* in three steps: (1) Model identification of G ; (2) Deriving parameters of *Priority Adaptor* C ; and (3) Stability analysis of the design.

3.5.1. Model Identification It is difficult to obtain the model of G using first-principles due to the complexity of the multi-hop ad hoc wireless network system. We treat the wireless network system as a black-box and then infer the model from externally observable metrics. The process to infer the open-loop system model is model identification. We use a 802.11 wireless adhoc test-bed to gather data for the model identification. Note that in model identification, we look at the priority as the system input and the end-to-end delay as output. On the other hand, in the priority adaptor design, the priority is output of the controller C and the end-to-end delay is the input.

In order to develop an adaptive priority adjustment algorithm to control the end-to-end delay, we need first to understand how priority affects the end-to-end delay under a certain traffic load. In this section, we will show how to get the system model G by model identification.

In the model identification, we use difference equation with unknown parameters as the dynamic model between the input (priority) and the output (end-to-end delay). Such a model estimates the mathematical relationship between the input and the output of the system. We then use a pseudo-random digital white noise generator to stimulate the system by assigning random priorities to multimedia packets and observe the end-to-end delay during a certain

time period. We choose a sampling interval of 0.5 second. So we get a priority and delay pair every 0.5 second. With the data we obtained in the experiments, we can apply the auto-regressive (AR) model to get the mathematical relationship from priority to end-to-end delay. We notice that the first order AR model provides reasonably good prediction for end-to-end delays with different priorities. The first order model is written as:

$$d(k+1) = b_0 p(k) + a_1 d(k) \quad (4)$$

where $d(k)$ and $p(k)$ are the end-to-end delay and priority at time k respectively. The relation of $d(k)$ and $p(k)$ in equation (4) is independent of the number of nodes on the route as long as the route between two end nodes are fixed.

The z-transform is used to take discrete time domain signals into a complex-variable frequency domain to simplify the calculation. It plays a similar role to what the Laplace Transform does in the continuous time domain. Like the Laplace, the z-transform gives a simpler way to solve problems and design discrete time applications. The corresponding z-transform of the transfer function from $p(k)$ to $d(k)$ in equation (4) is

$$G(z) = \frac{d(z)}{p(z)} = \frac{b_0}{z - a_1} \quad (5)$$

where b_0 and a_1 are to be determined in the model identification. We use 50 input-output pairs of $(p(k), d(k))$ to identify the model (i.e. coefficients b_0, a_1). The first 25 samples are used for identification, and the remaining 25 samples for validation. We determine the parameters that $b_0 = 0.02425$ and $a_1 = 0.2514$. Figure 3 shows the comparison of the measured output and model predicted output from the above model on the validation data.

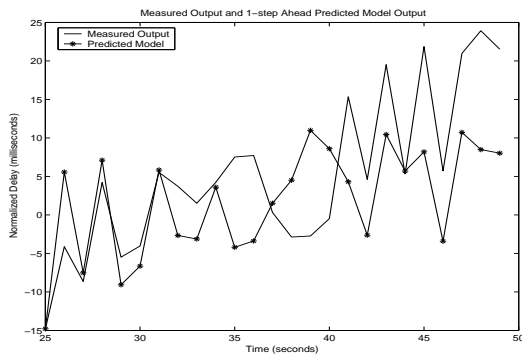


Figure 3. Comparison of measured output vs. model prediction

3.5.2. Design of PI Controller for Priority Adaptor The form of PI controller is given below. The parameters k_p and k_i are to be determined.

$$p(k+1) = k_p e(k) + k_i \sum_{t=0}^k e(t) \quad (6)$$

Then,

$$p(k) = k_p e(k-1) + k_i \sum_{t=0}^{k-1} e(t) \quad (7)$$

If we subtract (7) from (6), we have

$$p(k+1) - p(k) = k_p (e(k) - e(k-1)) + k_i e(k) \quad (8)$$

Then we get

$$p(k+1) = p(k) + (k_p + k_i)e(k) - k_p e(k-1) \quad (9)$$

The z-transform of controller (9) is given as

$$C(z) = \frac{(k_p + k_i)z - k_p}{z(z-1)} \quad (10)$$

From the model identification, we know the open loop model $G(z)$ is given as

$$G(z) = \frac{d(z)}{p(z)} = \frac{0.02425}{z - 0.2514} \quad (11)$$

According to discrete control theory, the performance of a system is implied by the poles of its closed loop transfer function $\frac{C(Z)G(z)}{1+C(Z)G(z)}$. We take the advantage of Root Locus, which is a graphical technique that plots the traces of poles of a closed-loop system on the z-plane as the controller parameters change. So with the Root Locus diagram, we can determine parameters k_p and k_i , in order to achieve the design goal listed in Table 1. We choose $k_i = 1.51$ and $k_p = 1.85$, and the resulting controller is given as

$$p(k+1) = p(k) + 3.36e(k) - 1.85e(k-1) \quad (12)$$

3.5.3. Stability Analysis We adopt Bode plot to show the stability of the designed close-loop system. Bode plot is a representation of the system's response to sinusoidal inputs at varying frequencies, which shows the magnitude and phase differences between the input and output sinusoids. Bode plot tells us stability of the close-loop system according to information of the open-loop system. The transfer function of the open-loop system is CG , so the transfer function of the close-loop system is $\frac{CG}{1+CG}$. Note as the denominator tends to zero, the system is not stable (goes to unbounded). So in that case that $|CG| = -1$, the system is not stable. Based on this, gain margin and phase

margin are defined. The gain margin with the unit decibel(dB) and phase margin with the unit degree are two values to measure how "close" a system is to crossing the boundary between stability and instability. The gain margin is how much $|C(j\omega)G(j\omega)|$ is below 0dB in Bode plot when $\arg(C(j\omega)G(j\omega)) = 180^\circ$. The phase margin is how many degrees $\arg(C(j\omega)G(j\omega))$ is above -180° when $|C(j\omega)G(j\omega)| = 1(0db)$. The Bode plot in Figure 4 shows the relative stability of the system [1]. The values of phase margin and gain margin of the system are given in the Bode plot, they are around 160 degree and 25 db respectively. Due to the large value of the phase and gain margin, the designed system is stable even under dynamically changed background traffic.

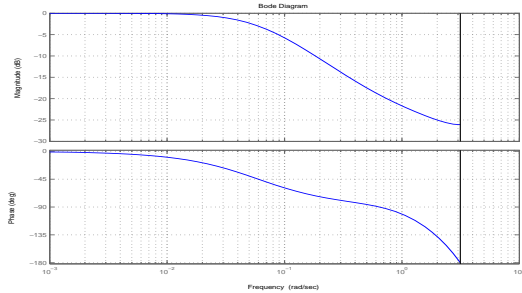


Figure 4. Bode plot of the closed-loop system

4. Experiments and Evaluation

4.1. Experiment Setup

The experiments for model identification and system evaluation are performed on our wireless ad hoc testbed, which operates on real IEEE 802.11 environment. We designed two scenarios to evaluate the impact of feedback control framework on end-to-end delay provision in wireless ad hoc networks. We adopt *javax.sound.sampled.AudioFormat* to specify data format in audio streams. IBM T30 laptops serve as wireless nodes in the ad hoc network. In below figures, the *time of packet play back (packets)* on the x-axes means the time at which the number of packets which are played at the receiver side. The delay values measured when a certain packet is received (and play) are shown in these figures.

Scenario 1:

Figure 5 shows the experiment setup for scenario 1. In the experiment, Machine 1 sends multimedia traffic (audio application) to Machine 3, where Machine 2 serves as a router

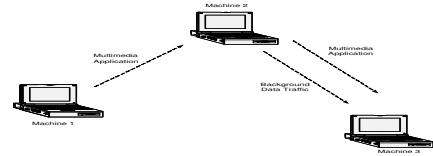


Figure 5. Experiment setup over 802.11b wireless ad hoc network for Scenario 1

of the multimedia traffic. At the same time, Machine 2 generates UDP background traffic to Machine 3, with data rate 100KB/s. The audio sampling rate is 256Kbps.

Scenario 2:

Figure 6 shows the experiment setup for scenario 2. In this case, Machine 1 sends multimedia flow 1 and background data flow 1 to Machine 3, Machine 2 sends multimedia flow 2 and background data flow 2 to Machine 4. Machine 5 serves as the router. The background data rate is 20KB/s. The audio sampling rate is 256Kbps.

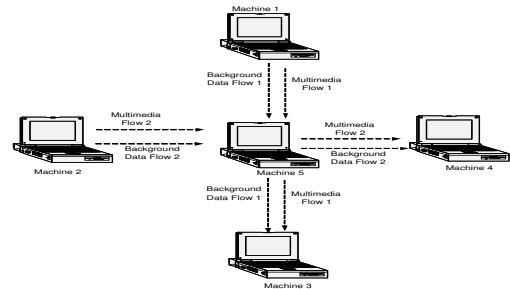


Figure 6. Experiment setup over 802.11b wireless ad hoc network for Scenario 2

4.2. Efficiency of Priority Adaptation

Scenario 1:

First, we do the experiment, in which no middleware adaptation is adopted. In this case, the background UDP traffic starts after the multimedia application sends 170 packets. Under the background traffic at the rate of 100 KB/s. We use UDP protocol for multimedia packets transmission. Usually, when the end-to-end delay is larger than a threshold, we consider the packet get lost. But here, to emphasize on end-to-end delay (without interference of packet-dropping), we assign large buffer to intermediate nodes. The end-to-end delay as shown in Figure 7 tends to be very large.

We then deploy the PI controller as in equation (12) on our test bed. The targeted end-to-end delay is 60 millisecond ($ref_{delay} = 60$). Figure 8 shows the resulting end-to-

end delay under the middleware priority adaptation. In this case, the background data traffic starts after the multimedia application sends around 60 packets.

The experiment shows that the PI controller is able to quickly converge the end-to-end delay of a multimedia application to a desired level, when there exist other applications which compete with the multimedia application for the network resources in wireless adhoc environment. We also notice that the delay of most of the multimedia packets is in the range from 40 to 80 ms. Therefore, the middleware adaptation method described in this paper also achieves the latency and playback jitter control for multimedia applications.

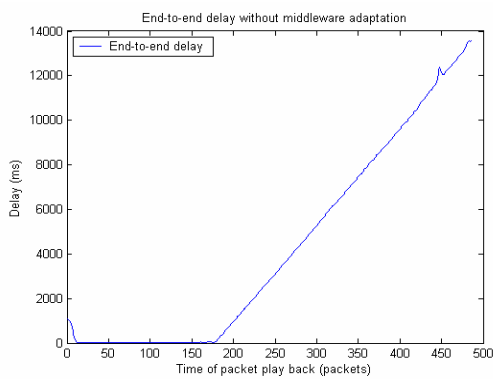


Figure 7. Scenario 1: End-to-end delay with background traffic without priority update

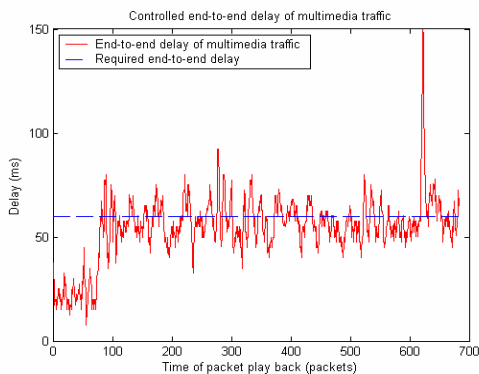
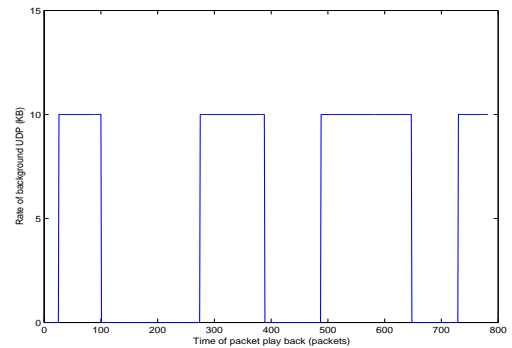
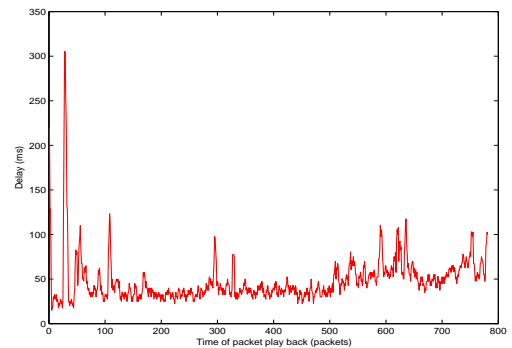


Figure 8. Scenario 1: End-to-end delay with background traffic under PI controller of priority update

We study the performance of the algorithm under dynamic background traffic, where the intermediate nodes see dynamic traffic, as Figure 9. We use on-off UDP traffic to simulate the variable background traffic as shown in Figure 9:(a). The background traffic is generated at the constant rate of 100 KB/s when the traffic is on. Figure 9:(b) shows the actual delay for an audio application under the on-off UDP background traffic. The audio sampling rate is 256Kbps.



(a) Variable background UDP traffic



(b) Delay under variable background traffic

Figure 9. Delay of the multimedia flow under the variable background traffic

Scenario 2:

We set the expected end-to-end delay (delay reference) of multimedia flow 1 be 100ms, and the end-to-end delay of multimedia flow 2 be 200ms. With no priority adaptation, the end-to-end delay tends to go to infinite as shown in Figure 10.

With middleware priority update, the end-to-end delays of multimedia flows are shown in Figure 11. You may notice the spikes in Figure 11. When network load turns to heavier, some packets suffers very large delay. We adjust priority of packets when we observe the large delay, so that the later

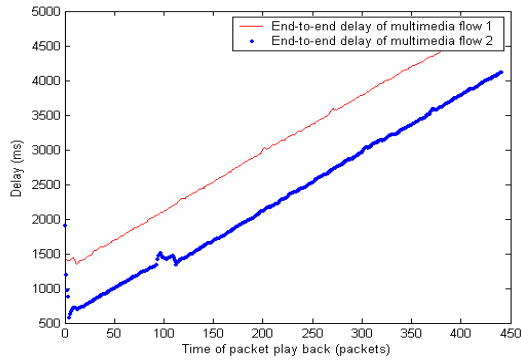


Figure 10. Scenario 2: End-to-end delay with background traffic without priority update

packets will encounter small delay. The spikes are formed in this way that some packets get large delay in a very short time.

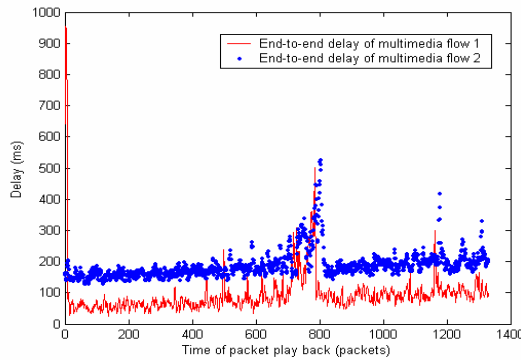


Figure 11. Scenario 2: End-to-end delay with background traffic under PI controller of priority update

4.3. End-to-end Delay Control for Multiple Multimedia Flows

We study the end-to-end delay of multiple multimedia applications under the adaptation mechanism proposed in this paper. In experiment setup Scenario 1, let two multiple multimedia flows send from Machine 1 to Machine 3, so they compete for the limited resources in the wireless ad hoc network. Users can specify different end-to-end delay requirement for different multimedia applications. In the experiment, we set up the required end-to-end delay of flow

1 as 60ms, and the delay of flow 2 as 120ms. In this case, background UDP traffic is active during the whole experiment. Figure 12 shows the end-to-end delay of multimedia flows, and we can conclude that the designed PI controller is capable to control the end-to-end delays of multiple multimedia flows. We notice that if the classifier in the middleware takes the priority directly as the service class parameter, we can study the relationship of delay and priorities of the application.

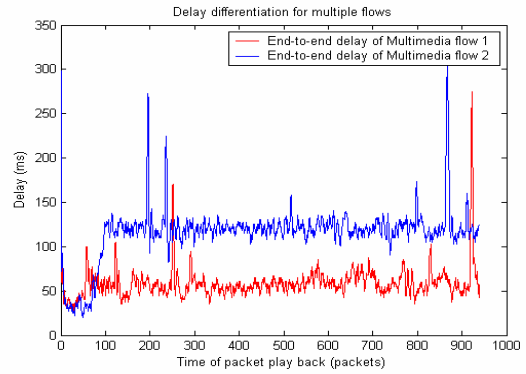


Figure 12. Controlled end-to-end delay for multiple multimedia applications

4.4. The Effect of Classifier

To have efficient network management and easy deployment, the number of service classes are usually small. So we categorize the priorities generated by *Priority Adaptor* in groups. Each group represents a service class, and the highest priority in the group is assigned as the parameter of the service class. To show the effect of the *Classifier*, we use the scenario 1 to setup the experiment, and let two multimedia flows send from Machine 1 to Machine 3. The required end-to-end delays of flow 1 and flow 2 are 60ms and 120ms respectively. The background UDP traffic is active during the whole experiment period. In the experiment, the middleware *Classifier* provides 20 service classes (groups of priorities). Figure 13 shows the effect of *Classifier* on end-to-end delays. Comparing the experiment results in Figure 12, we can observe that the controlled end-to-end delay under the *Classifier* is smaller than the controlled delay without it. This effect is easy to understand since we select the largest priority in group as the class parameter for each service class.

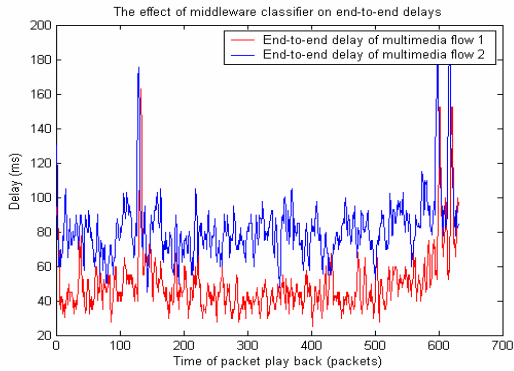


Figure 13. The effect of classifier on end-to-end delays

5. More Discussions

Our paper presents a feedback control framework for end-to-end delay management embedded in upper layers of the protocol stack. Our design on requirement adaptor and priority adaptor are decoupled, so dynamically changed delay requirements does not affect the stability of control loop in the middleware layer (priority adaptor).

The experiments show that when background data and multimedia application share one or more nodes on their routes, the cross-layer control framework yields very good results on service differentiation and end-to-end delay support. However, we want to also stress that in some cases (as shown in Figure 14) that service differentiation provided by the upper layer cooperation cannot fully differentiate flows or provide delay support. In cases shown in Figure 14, multimedia flow (say flow 1) and background flow (say flow 2) do not share common intermediate node on their routes. Hence, network layer service differentiation cannot provide support for those flows. Therefore, the MAC layer service differentiation scheduling scheme (e.g. IEEE 802.11e) must be adopted together with upper layer cooperation for end-to-end QoS support in wireless ad hoc networks.

References

[1] <http://www.library.cmu.edu/ctms/ctms/freq/freq.htm>.
 [2] I. Aad and C. Castelluccia. Differentiation mechanisms for IEEE 802.11. In *INFOCOM*, pages 209–218, 2001.
 [3] G.-S. Ahn, A. T. Campbell, A. Veres, and L.-H. Sun. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (swan). *IEEE Transactions on Mobile Computing*, 1(3):192–207, 2002.
 [4] A. Banchs, M. Radimirsch, and X. Perez. Assured and expedited forwarding extensions for IEEE 802.11 wireless LAN. In *IEEE/IFIP IWQoS*, pages 237–246, 2002.

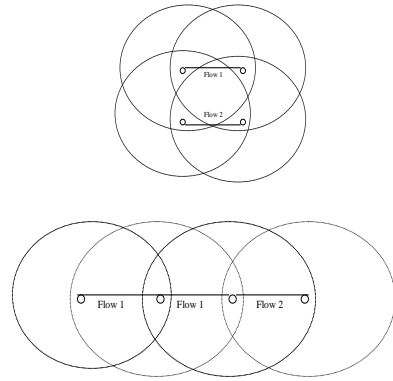


Figure 14. Cases that MAC layer service differentiation scheduler must be adopted

[5] W. T. Chen, B. B. Jian, and S. C. Lo. An adaptive retransmission scheme with qos support for the ieee 802.11 mac enhancement. In *IEEE VTC*, Spring 2002.
 [6] C. Dovrolis and P. Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network*, 13(5), 1999.
 [7] C. Dovrolis and P. Ramanathan. Dynamic class selection: From relative differentiation to absolute QoS. In *Proceedings of ICNP*, Nov. 2001.
 [8] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *SIGCOMM*, pages 109–120, 1999.
 [9] A. Grilo and M. Nunes. Performance evaluation of IEEE 802.11e. In *IEEE PIMRC*, pages 511–517, 2002.
 [10] S.-B. Lee, G.-S. Ahn, X. Zhang, and A. T. Campbell. Insignia: an ip-based quality of service framework for mobile ad hoc networks. *J. Parallel Distrib. Comput.*, 60(4):374–406, 2000.
 [11] X. Liu, X. Zhu, S. Singhal, and M. Arlitt. Adaptive entitlement control to resource containers on shared servers. In *9th IFIP/IEEE International Symposium on Integrated Network Management*, 2005.
 [12] L. Ljung. *System Identification: Theory for the User* (2nd Edition), 1999.
 [13] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. A feedback control approach for guaranteeing relative delays in web servers. In *IEEE Real-Time Technology and Applications Symposium (RTAS)*, June 2001.
 [14] C. S. Ong, Y. Xue, and K. Nahrstedt. A middleware for service adaptation in differentiated 802.11 wireless networks. In *Workshop on Coordinated Quality of Service in Distributed Systems (COQODS)*, pages 314–329, Singapore, 2004.
 [15] D. Qiao and K. G. Shin. Achieving efficient channel utilization and weighted fairness for data communications in ieee 802.11 wlan under the dcf. In *The Tenth International Workshop on Quality of Service (IWQoS'2002)*, Miami, Florida, 2000.

- [16] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher. Queuing model based network server performance control. In *23th IEEE Real-Time Systems Symposium (RTSS)*, December 2002.
- [17] S. T. Sheu and T. F. Sheu. A bandwidth allocation/sharing/extension protocol for multimedia over ieee 802.11 ad hoc wireless lans. *IEEE JSAC*, 19(10):2065–2080, October 2001.
- [18] J. L. Sobrinho and A. S. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *IEEE JSAC*, 17(8):1353–1368, August 1999.
- [19] N. H. Vaidya, P. Bahl, and S. Gupta. Distributed fair scheduling in a wireless LAN. In *Mobile Computing and Networking*, pages 167–178, 2000.
- [20] Q. Xue and A. Ganz. Proportional service differentiation in wireless lans using spacing-based channel occupancy regulation. In *ACM Multimedia*, November 2004.
- [21] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets. Design and evaluation of a cross-layer adaptation framework for mobile multimedia systems. In *SPIE/ACM Multimedia Computing and Networking Conference*, 2003.