

End-to-End Delay Control of Multimedia Applications over Multihop Wireless Links

WENBO HE and KLARA NAHRSTEDT
University of Illinois at Urbana Champaign
and
XUE LIU
McGill University

The proliferation of multimedia applications over mobile, resource-constrained wireless networks has raised the need for techniques that adapt these applications both to clients' Quality of Service (QoS) requirements and to network resource constraints. This article investigates the upper-layer adaptation mechanisms to achieve end-to-end delay control for multimedia applications. The proposed adaptation approach spans application layer, middleware layer and network layer. In application layer, the requirement adaptor dynamically changes the requirement levels according to end-to-end delay measurement and acceptable QoS requirements for the end-users. In middleware layer, the priority adaptor is used to dynamically adjust the service classes for applications using feedback control theory. In network layer, the service differentiation scheduler assigns different network resources (e.g., bandwidth) to different service classes. With the coordination of these three layers, our approach can adaptively assign resources to multimedia applications. To evaluate the impact of our adaptation scheme, we built a real IEEE 802.11 ad hoc network testbed. The test-bed experiments show that the proposed upper-layer adaptation for end-to-end delay control successfully adjusts multimedia applications to meet delay requirements in many scenarios.

Categories and Subject Descriptors: D.4.8 [Operating Systems]: Performance; H.3.4 [Information Storage and Retrieval]: Systems and Software—*Performance evaluation*

General Terms: Performance, Design

Additional Key Words and Phrases: End-to-end delay QoS; wireless ad hoc networks

ACM Reference Format:

He, W., Nahrstedt, K., and Liu, X. 2008. End-to-end delay control of multimedia applications over multihop wireless links. *ACM Trans. Multimedia Comput. Commun. Appl.* 5, 2, Article 16 (November 2008), 20 pages. DOI = 10.1145/1413862.1413869 <http://doi.acm.org/10.1145/1413862.1413869>

1. INTRODUCTION

Pervasive and ubiquitous multimedia communications are in great demand recently, such as in emergency medical systems and disaster relief operations. The advance of wireless technologies facilitates

Part of this work was presented at the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06). © IEEE 2006.

Authors' addresses: W. He, K. Nahrstedt, Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, IL 61801; X. Liu, School of Computer Science, McGill University, Montréal, Que. H3A 2A7, Canada.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2008 ACM 1551-6857/2008/11-ART16 \$5.00 DOI 10.1145/1413862.1413869 <http://doi.acm.org/10.1145/1413862.1413869>

ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 5, No. 2, Article 16, Publication date: November 2008.

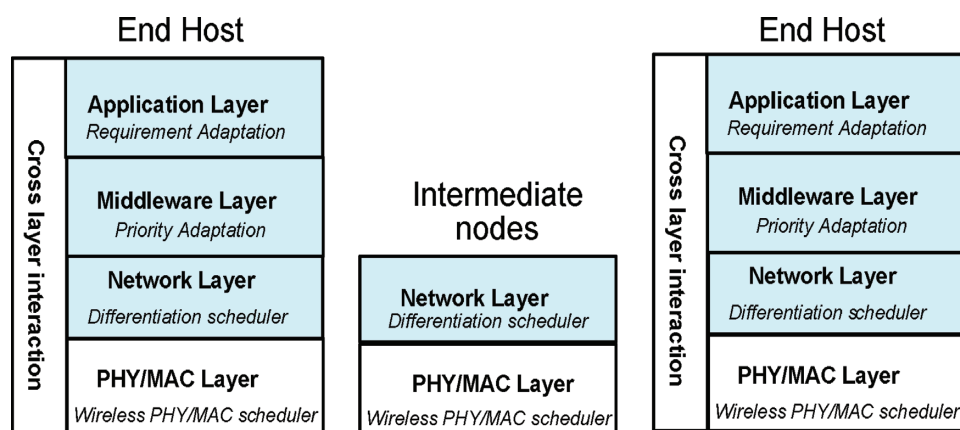


Fig. 1. Network architecture of the end-to-end delay management system.

multimedia applications in ubiquitous environments. We observe that wireless networks are accommodating more and more emerging multimedia traffic. However, communications over wireless links are subjected to channel fading, multipath fading, and interference from background noise and neighbors, which makes the delay of wireless communications unpredictable and hard to control. Moreover, a multimedia application may take multiple hops from the source to its destination. To achieve smooth delivery of multimedia contents, end-to-end delay control in multihop wireless network is important for multimedia applications.

Figure 1 illustrates the network layer stack of the end-to-end delay control system, where a multimedia flow is generated by an end host, is forwarded by intermediate nodes in the wireless ad hoc network, and finally reaches the destination host. In wireless ad hoc networks, multiple hosts share the wireless link capacity in the vicinity of each other. Accordingly, the bandwidth of a wireless link is limited, shared, unpredictable. Also, the channel capacities and error rates are time-varying in comparison with wired networks. Hence, end-to-end delay control is challenging in wireless ad hoc networks.

Wireless network standards typically focus on media access control (MAC) layer and physical (PHY) layer signaling, leaving crucial end-to-end delay control to end users (or application implementers). However, PHY/MAC layer scheduling mechanisms are not proper to achieve end-to-end QoS support, because these mechanisms only control link layer behavior and achieve one-hop QoS management. In this paper, we address the end-to-end delay and jitter control for multimedia applications over multihop wireless links via upper-layer adaptation, where network, middleware, and application layers are involved. Upper-layer adaptation is independent of lower-layer hardware and protocols, hence is able to provide flexibility for end users.

A 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 multilevel 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 dynamically according to observed

end-to-end delay. Therefore, with the assistance of network layer service differentiation scheduler, the multimedia applications can be adapted to network conditions, and meet the delay requirement.

The rest of the article is organized as follows. Section 2 reviews the related work. Section 3 shows the framework of upper-layer adaptation mechanism, and illustrates the function of each component in the framework. Section 4 describes the test-bed setup and experiment scenarios, and then evaluates the performance of upper-layer adaptation. Section 5 concludes the paper.

2. RELATED WORK

Most multimedia applications are delay-sensitive and require quality of service (QoS) guarantees. There are many studies focusing on throughput guarantees of wireless ad-hoc networks [Li 2005; Luo et al. 2000], but they do not provide delay guarantees for multi-hop wireless ad-hoc networks. In approaches proposed in Barry et al. [2001], Ahn et al. [2002a], Lee et al. [2000], and Ahn et al. [2002b], admission control was used to provide delay guarantees based on delay measurement. However, Yang and Kravets [2004] showed that they are not successful because the delay of a particular flow is affected by the packet scheduling between the competing flows at neighboring nodes. Hence a newly arrived flow affects the delay of all of the nearby flows. As a result, measurement based admission control delay guarantee protocols are difficult to design without taking into account the admitting new flows that may degrade the delay of existing QoS flows.

Many current researches on QoS management in wireless networks are primarily focused on MAC layer scheduling. Because of the wide availability of IEEE 802.11 [IEEE Computer Society] technology, many studies discussed on providing delay guarantees in ad hoc networks based on IEEE 802.11. Sobrinho and Krishnakumar [1999], Aad and Castelluccia [2001], Banchs et al. [2002], and Sheu and Sheu [2001] study the MAC layer scheduling under IEEE 802.11 DCF to provide differentiated delays for single-hop wireless applications. However, these approaches are not appropriate to provide end-to-end delay guarantees for multihop applications.

Other schemes or protocols, including IEEE 802.11e [Mangold et al. 2002; Chen 2002; Grillo and Nunes 2002; Vaidya et al. 2000] explore mechanisms to adjust the Contention Window size (CW) or adjusting the Transmit Opportunity (TXOP) time interval length. These parameters affect the delay of a flow. Hence these protocols can provide delay differentiation by allocating different CW sizes or TXOP lengths to different classes of flows. However, this type of approach only provides proportional delay differentiation but does not guarantee the actual delay of an individual flow.

While most research on delay guarantee in wireless ad hoc network focused on single layer such as link-layer (e.g., scheduling) or network layer (e.g., routing), real-world applications need an integrated cooperation among different layers to achieve such guarantee. Our work differs from most previous research in that it exploits the cooperation among upper layers (application layer, middleware layer, and network layer) to achieve end-to-end delay control for multimedia applications. Our approach also has the advantage that it is independent from the link (MAC) layer protocols, which is usually designed to achieve delay differentiation for single-hop applications. Hence our approach can be used on top of a variety of lower-level protocols.

Our approach utilizes online feedback control to dynamically adjust service priority levels. Control theory has recently been applied to computer systems for resource management and performance control [Diao et al. 2005; Li and Nahrstedt 1999; Hellerstein et al. 2004; Hellerstein 2004; Karamanolis et al. 2005]. The application areas include Web server performance guarantees [Abdelzaher et al. 2002], dynamic adjustment of the cache size for multiple request classes [Lu et al. 2004], guaranteed relative delays in Web servers by connection scheduling [Lu et al. 2001], CPU and memory utilization control in Web servers [Diao et al. 2002], and to adjust the resource demands of virtual machines based on resource availability [Zhang et al. 2005]. Li et al. [Li and Nahrstedt 1999] first proposed a middleware

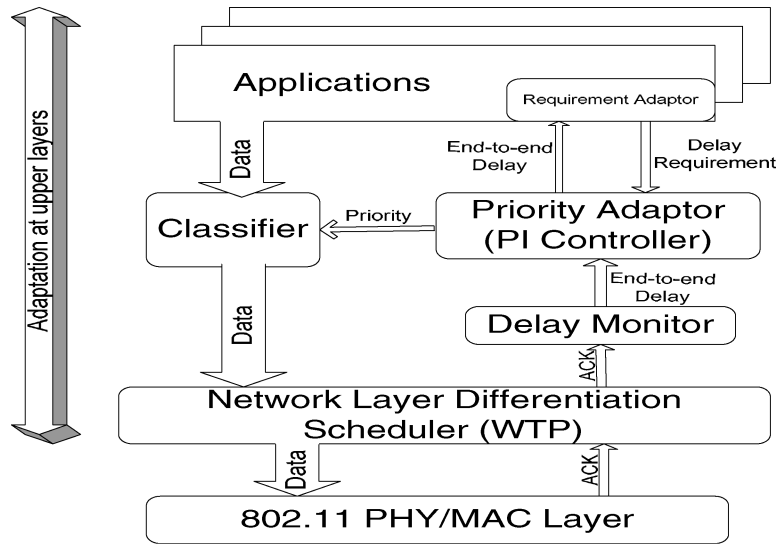


Fig. 2. Network stack of the cross-layer design scheme.

framework using feedback control for QoS adaptation. Their work uses fuzzy controller while in this paper we use adaptive controller that dynamically adjusts the parameters of the controller, so that the controller is able to adapt with wide-range of network load conditions.

To the best of our knowledge, this work is among the first to provide end-to-end delay guarantee using coordinated upper-layer adaptation for multimedia applications over multihop ad hoc wireless networks.

3. UPPER-LAYER ADAPTATION FRAMEWORK

Figure 2 shows the framework of our upper-layer adaptation. The components involved in our design are waiting time priority (WTP) scheduler (at network layer), delay monitor (at middleware layer), classifier (at middleware layer), priority adaptor (at middleware layer) and requirement adaptor (at application layer). These components in the upper-layer adaptation framework cooperate seamlessly to make multimedia applications meet end-to-end delay requirement. We will explain the function of each individual component in this framework in the following subsections.

3.1 Differentiation WTP Scheduler

Our upper-layer adaptation requires the support of network layer service differentiation. To enforce the service differentiation, we adopt waiting time priority (WTP) scheduler at network layer. Dovrolis and Ramanathan [1999] introduce 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. Xue and Ganz [2004] address 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 . 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 quality of service requirement of an application, the application must be able to dynamically select the appropriate service classes. This is the basic idea of our network layer scheduler design.

In this article, we use waiting time priority (WTP) scheduler to approximate the 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. Proposition 1 in the appendix shows that the waiting time priority (WTP) scheduler is one of the algorithms to approximate PDD for data-intensive and interleaved multimedia flows.

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 time stamps when sending the packets to the destination. While receiving a packet, the receiver acquires the attached time stamp, and sends a middleware ACK back to the sender with the acquired time stamp. As the sender gets ACK from the receiver, it retrieves the sending time stamp, and compares it with the current time stamp, 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. Note that we use middleware ACK instead of MAC layer ACK, because we want to measure end-to-end delay instead of delay at a single hop. To reduce the overhead introduced by middleware ACKs, we don't want to send an ACK for each received packet. Rather we can combine time stamps of multiple packets in a single ACK.

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 article, the QoS requirement is given by the expectation of the end-to-end delay. Users prefer the best quality of the multimedia delivery that 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 10%, 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 $P < 90\%$. In our application model, if given d_{req} , j_{req} , and D , the chosen delay requirement should satisfy

$$d_{req} = \min \{d \in D \mid Prob(d \in [d_{min}, d_{max}]) > 90\%\}. \quad (3)$$

The application layer selects the highest possible requirement satisfying Equation (3) from the requirement set D to adapt to the network load. Our design of application layer requirement adaptor is inspired by Chebyshev's inequality [Chung 2000; Feller 1971]:

$$Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}, \quad (4)$$

where X is a random variable with expected value μ and variance σ^2 . We look X as measured delay and μ as average delay under certain network condition. In steady state, we hope $\mu = d_{req}$ and $Pr(|X - \mu| \geq j_{req}) \leq p$ with $j_{req} = k\sigma$ and $p = \frac{1}{k^2}$. If the network condition gets worse, X will increase. Thus, $Pr(|X - d_{req}| \geq j_{req}) > p$. If lower layer adaptors cannot bring $Pr(|X - d_{req}| \geq j_{req})$ down in a short period, then we need to adjust d_{req} or j_{req} . Here we adjust the former.

Table I. The Desired Properties and the Controller Design Criteria

Desired System Properties	Controller Design Criteria
Stability	Place poles within the unit-circle
Accuracy of control (Zero steady-state error)	Adopt Integral (I) control
Fast response and less oscillations	Select reasonable parameters of PI controller

Note that if Equation (3) is not satisfied, then it implies that by adjusting only the priority (even using the highest priority), we still cannot keep the QoS level satisfying the given requirement. Hence, it is time for requirement adaptor to update its parameters. It is not necessary to guarantee that the requirement adaptor never decreases the requirement before priority adaptor reaches the highest priority. Both requirement adaptor and priority adaptor dynamically select their parameters. Our concern is that the QoS requirement can be satisfied for a long run.

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 system identification techniques [Ljung 1999], we determine the model which captures the relationship between priority and end-to-end delay. After we obtain 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 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 predetermined 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 ad hoc 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 I. Figure 3 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. Note that the control interval is not a fixed time slot. The controller is event-driven controller. Whenever the delay monitor gets an estimated delay, the controller updates the priority.

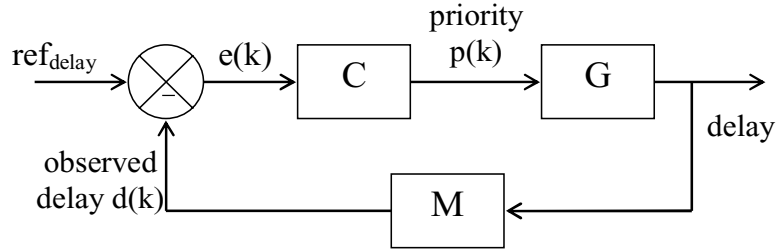


Fig. 3. The block diagram of the closed-loop control system for end-to-end delay.

We obtain the parameters of the Priority Adaptor in four steps: (1) Model identification of G for a fixed load condition. (2) PI Controller design of the Priority Adaptor C based on off-line model identification. (3) Stability analysis of the PI controller. (4) Design of adaptive controller which dynamically changes parameters of the controller, so that the controller is able to adapt with different network load conditions.

3.5.1 Off Line Model Identification. It is difficult to obtain the model of G using first principles due to the complexity of the multihop 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 an 802.11 wireless ad hoc 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:

$$d(k+1) = b_0 p(k) + a_1 d(k), \quad (5)$$

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 (5) 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

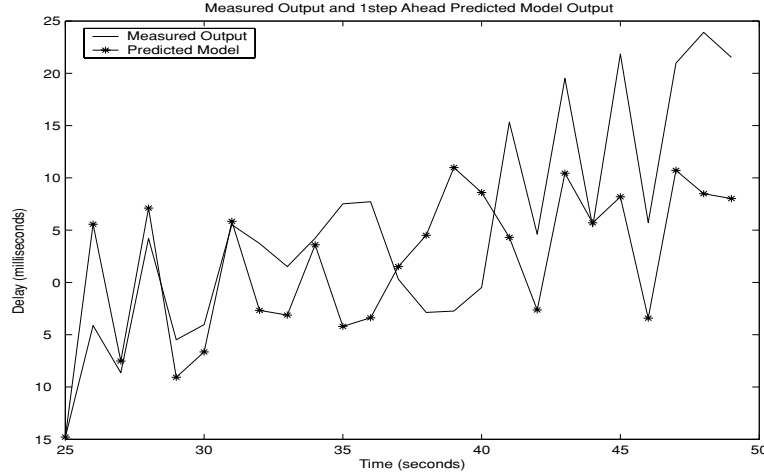


Fig. 4. Comparison of measured output vs. model prediction.

$d(k)$ in Equation (5) is

$$G(z) = \frac{d(z)}{p(z)} = \frac{b_0}{z - a_1}, \quad (6)$$

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 4 shows the comparison of the measured output and model predicted output from the above model on the validation data.

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

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

Then,

$$p(k) = k_p e(k-1) + k_i \sum_{t=0}^{k-1} e(t). \quad (8)$$

If we subtract (8) from (7), we have

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

Then we get

$$p(k+1) = p(k) + (k_p + k_i)e(k) - k_p e(k-1). \quad (10)$$

The z-transform of controller (10) is given as

$$C(z) = \frac{(k_p + k_i)z - k_p}{z(z-1)}. \quad (11)$$

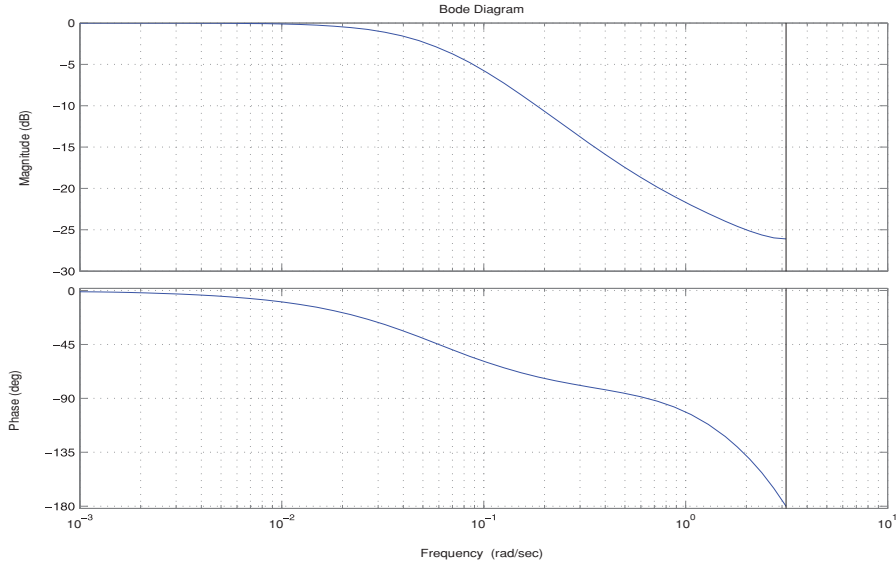


Fig. 5. Bode plot of the closed-loop system.

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

$$G(z) = \frac{d(z)}{p(z)} = \frac{0.02425}{z - 0.2514}. \quad (12)$$

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 I. We choose $k_i = 1.51$ and $k_p = 1.85$, and the resulting controller is given as

$$p(k+1) = p(k) + 3.36e(k) - 1.85e(k-1). \quad (13)$$

3.5.3 Stability Analysis of PI Controller. 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 5 shows the relative stability of the system. 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 values of the phase and gain margin, the designed close-loop system is stable even under dynamically changed background traffic.

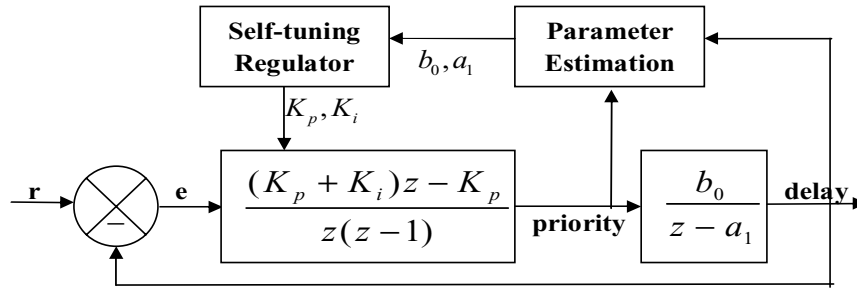


Fig. 6. Block diagram of a self-tuning adaptor.

3.5.4 Adaptive Controller Design. We have described the design of PI controller for priority adaptor, which is determined off-line based on the dynamic input-output pairs via system identification. Data collected in the system identification experiments are obtained with fixed load condition. In a real network environment, it is difficult to find a single linear model characterizing the systems behavior under all load conditions. Hence, we need an adaptive scheme which is able to adapt its behavior according to changing load and network condition. In control theory, this goal is achieved through adaptive controller. We implemented a simple self-tuning regulator on our test-bed, where we assume the system model has a fixed structure, but the parameters are time-varying depending on load and network condition. Figure 6 shows a block diagram for the self-tuning adaptor to control priorities of multimedia applications. Comparing Figure 6 with Figure 3, we add two blocks: an online estimation of identified parameter, and self-tuning regulator. In Figure 6, we have $M = 1$, $C = \frac{(K_p + K_i)z - K_p}{z(z-1)}$ and $G = \frac{b_0}{z - a_1}$.

4. EXPERIMENTS AND EVALUATION

4.1 Experiment Setup

The experiments for model identification and system evaluation are performed on our wireless ad hoc test bed, which operates on real IEEE 802.11b 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. *AudioFormat* takes several parameters, wherein *sampleRate* represents the number of samples per second, and *sampleSizeInBits* represents the number of bits in each sample. In our implementation, we take *sampleRate* = 32000, and *sampleSizeInBits* = 8. Hence audio traffic rate is 256Kbps. We use PCM (pulse-code modulation) as the encoding technique to generate data of audio streams, and the packetization interval is 20ms. IBM T30 laptops with 1.8-GHz P4-M processor and 256MB RAM serve as wireless nodes in the ad hoc network. In the following figures, the *time of packet playback (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 7 shows the experiment setup for scenario 1. In the experiment, Laptop 1 sends multimedia traffic (audio application) to Laptop 3, where Laptop 2 serves as a router of the multimedia traffic. At the same time, Laptop 2 generates UDP background traffic to Laptop 3, with data rate 100Kbps. We set a fixed priority for the background traffic, say 1. The audio sampling rate is 256Kbps.

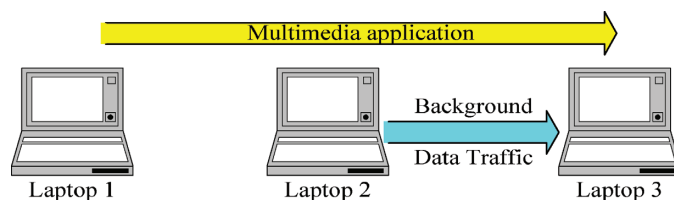


Fig. 7. Experiment setup over 802.11b wireless ad hoc network for Scenario 1.

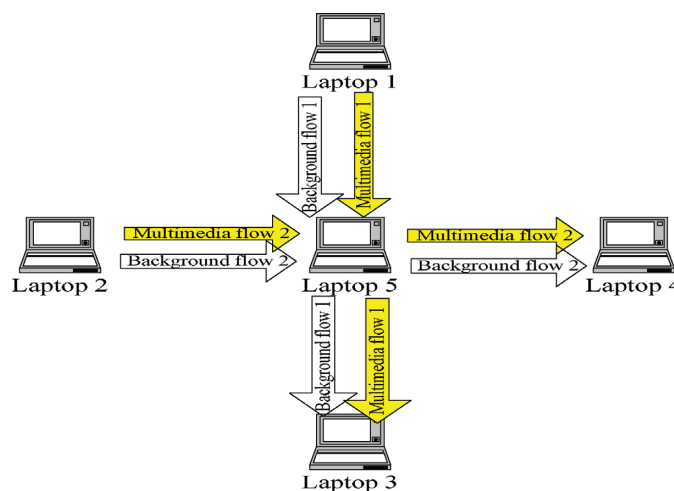


Fig. 8. Experiment setup over 802.11b wireless ad hoc network for Scenario 2.

Scenario 2. Figure 8 shows the experiment setup for scenario 2. In this case, Laptop 1 sends multimedia flow 1 and background data flow 1 to Laptop 3, Laptop 2 sends multimedia flow 2 and background data flow 2 to Laptop 4. Laptop 5 serves as the router. The background data rate is 20 Kbps. We set priority of the background traffic as 1. The audio sampling rate is 256 Kbps.

4.2 Efficiency of Priority Adaptation

Scenario 1. First, we do the experiment, in which no middleware adaptation is adopted, under the background traffic at the rate of 100 Kbps. In this case, the background UDP traffic starts after the multimedia application sends 170 packets. 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 9 tends to be very large.

We then deploy the adaptive controller on our test bed. The targeted end-to-end delay is 60 milliseconds ($ref_{delay} = 60$). Figure 10 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 90th percentile of end-to-end delay is about 70 milliseconds.

The experiment shows that the adaptive 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 ad hoc environment. We also notice that

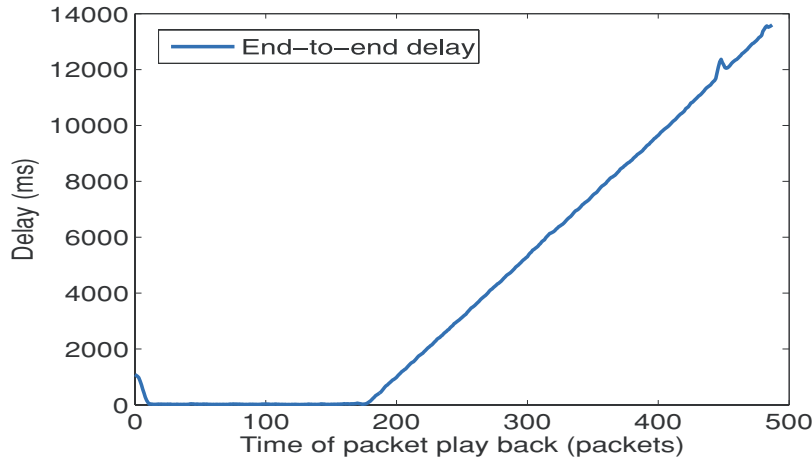


Fig. 9. Scenario 1: End-to-end delay with background traffic without priority update.

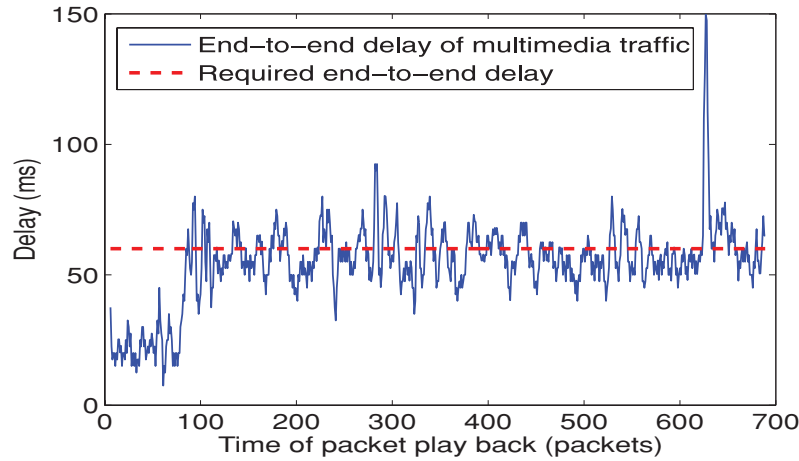
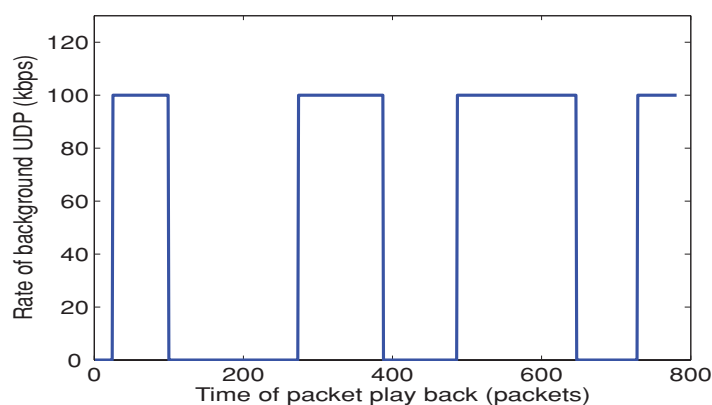


Fig. 10. Scenario 1: End-to-end delay with background traffic under adaptive controller of priority update.

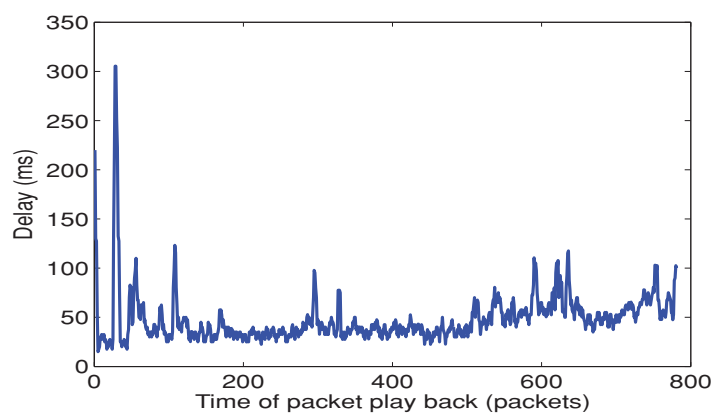
the delay of most of the multimedia packets is in the range from 40 to 80 ms. Therefore, the middleware adaptation method described in this paper also achieves the latency and playback jitter control for multimedia applications.

We study the performance of the algorithm under dynamic background traffic, where the intermediate nodes see dynamic traffic, as in Figure 11. We use randomly generated on-off UDP traffic to simulate the variable background traffic as shown in Figure 11(a). The background traffic is generated at the constant rate of 100 Kbps when the traffic is on. Figure 11(b) shows the actual delay for an audio application under the on-off UDP background traffic. The audio sampling rate is 256 Kbps, and the target delay is 60 ms.

Scenario 2. We set the expected end-to-end delay (delay reference) of multimedia flow 1 be 100 ms, and the end-to-end delay of multimedia flow 2 be 200 ms. With no priority adaptation, the end-to-end delay tends to go to infinite as shown in Figure 12.



(a) Variable background UDP traffic



(b) Delay under variable background traffic

Fig. 11. Delay of the multimedia flow under the variable background traffic.

With middleware priority update, the end-to-end delays of multimedia flows are shown in Figure 13. You may notice the spikes in Figure 13. When network load turns to heavier, some packets suffer very large delay. We adjust priority of packets when we observe the large delay, so that the later packets will encounter small delay. The spikes are formed in this way that some packets get large delay in a very short time.

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 multimedia flows send from Laptop 1 to Laptop 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 14 shows the end-to-end delay of multimedia flows, and we can conclude that the designed adaptive controller is capable to control the end-to-end delays of multiple multimedia flows. We notice that if the classifier in the middleware

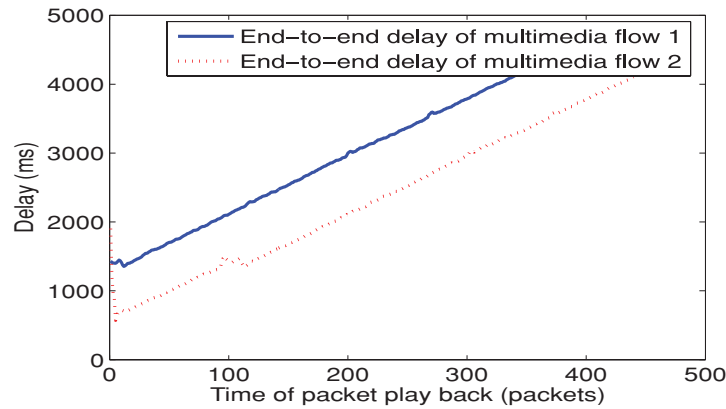


Fig. 12. Scenario 2: End-to-end delay with background traffic without priority update.

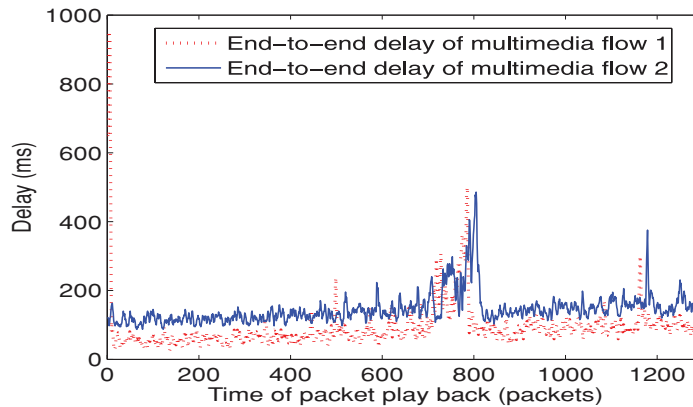


Fig. 13. Scenario 2: End-to-end delay with background traffic under adaptive controller of priority update.

takes the priority directly as the service class parameter, we can study the relationship of delay and priorities of the application.

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 Laptop 1 to Laptop 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 15 shows the effect of Classifier on end-to-end delays. Comparing the experiment results in Figure 14, 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.

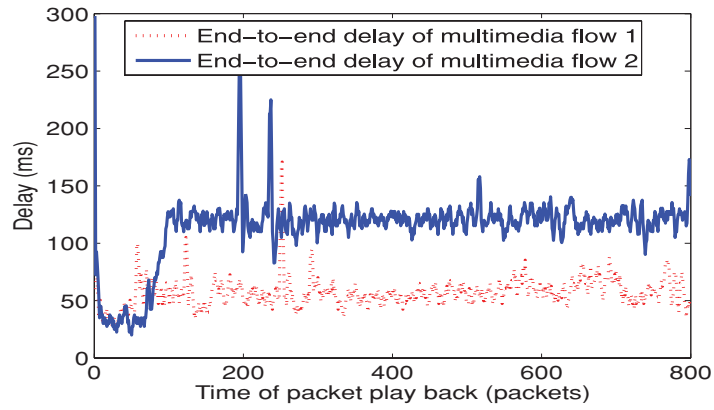


Fig. 14. Controlled end-to-end delay for multiple multimedia applications.

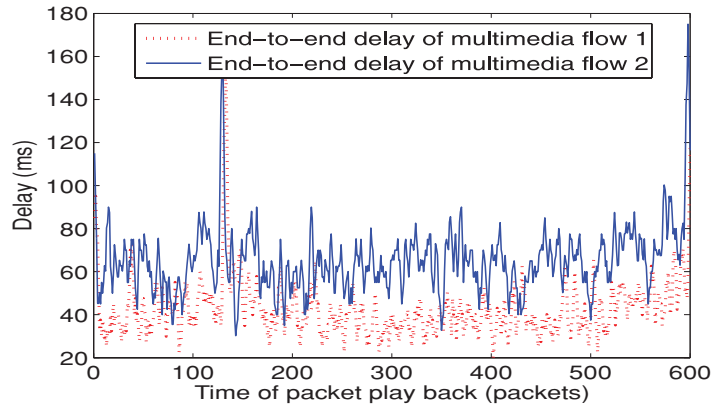


Fig. 15. The effect of classifier on end-to-end delays.

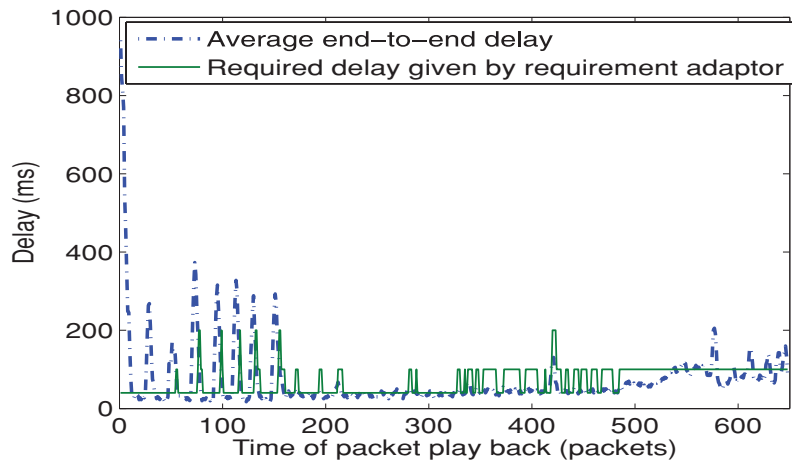


Fig. 16. Delay with Requirement adaptation.

4.5 The Effect of Requirement Adaptor

Consider the scenario 1, and we set the multiple delay levels as $D = \{40ms, 100ms, 200ms\}$ and $j_{req} = 40ms$. When the wireless links are heavily loaded, the application relaxes the delay requirement temporarily, and when the load condition improves, the application updates the delay requirement to higher level. Consequently, the dynamic adjustment of requirement and the priority makes the delay of audio application fulfill the highest delay requirement (smallest delay) as much as possible. Figure 16 shows the effect of requirement adaptation.

5. CONCLUSIONS

This article 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 middle-layer (priority adaptor). The upper-layer adaptation is independent of PHY/MAC layer protocols. Hence, our upper-layer adaptation scheme can be widely used on different PHY/MAC hardware and protocols.

APPENDIX

Assume the class i is assigned with class parameter p_i , the scheduling algorithm in (2) is equivalent to select the packet in class i to serve if $i = \arg \max_{j \in B(t)} w_j(t) p_j$. Let $n_i(t, \tau)$ be the number of packets of class i received and queued by network layer WTP scheduler in time period $[t, t + \tau]$, and $m_i(t, \tau)$ be the number of packets of class i served by the scheduler in time period $[t, t + \tau]$. Denote $a_i(t)$ be the arrival time of the last served packet from service class i before time t , $b_i(t)$ be the arrival time of the packet which is first served from service class i after time t , and $s_i(t)$ be the service begin time of the last packet served from class i before time t . Consider any pair of service classes i and j , ($i, j \in B(t)$ and $i \neq j$), the packets from classes i and j interweave with each other.

LEMMA 1. *For data-intensive and interleaved flows, we have $(t - a_i(t))p_i \simeq (t - a_j(t))p_j$.*

PROOF. By the scheduling algorithm (2), we have

$$(t - a_i(t))p_i > (t - b_j(t))p_j \quad (14)$$

and

$$(t - a_j(t))p_j > (t - b_i(t))p_i. \quad (15)$$

If $s_i(t) > s_j(t)$, then last packet served by the scheduler is from class i , we have:

$$(t - a_j(t))p_j > (t - a_i(t))p_i. \quad (16)$$

Combining Equations (14) and (16), we have

$$(t - a_j(t))p_j > (t - a_i(t))p_i > (t - b_j(t))p_j. \quad (17)$$

For data-intensive and interleaved flows, the packet arrivals from class j are close enough, so that $a_j(t) \simeq b_j(t)$. Similar to the packet arrivals from packet i , we have $a_i(t) \simeq b_i(t)$. Therefore, $(t - a_j(t))p_j \simeq (t - b_j(t))p_j$ and $(t - a_i(t))p_i \simeq (t - b_i(t))p_i$. Thus, by Equation (17), we have the approximation that $(t - a_i(t))p_i \simeq (t - a_j(t))p_j$.

Similarly, if $s_i(t) < s_j(t)$, we can get the same result.

LEMMA 2. For data-intensive and interleaved flows, during the period $[t, t + \tau]$, $(1 - \frac{m_i(t, \tau)}{n_i(t, \tau)})p_i \simeq (1 - \frac{m_j(t, \tau)}{n_j(t, \tau)})p_j$.

PROOF. By Lemma 1, we have

$$(t + a_i(t))p_i = (t + a_j(t))p_j \quad (18)$$

and

$$(t + \tau + a_i(t + \tau))p_i = (t + \tau + a_j(t + \tau))p_j. \quad (19)$$

From Equations (18) and (19), we have

$$\frac{a_j(t + \tau)p_j - a_i(t + \tau)p_i}{p_j - p_i} = t + \tau \quad (20)$$

and

$$\frac{a_j(t)p_j - a_i(t)p_i}{p_j - p_i} = t. \quad (21)$$

$n_i(t, \tau)$, the arrival packets of class i being queued during $[t, t + \tau]$ and the packets being served $m_i(t, \tau)$ satisfies

$$\frac{n_i(t, \tau)}{\tau} \simeq \frac{m_i(t, \tau)}{a_i(t + \tau) - a_i(t)}. \quad (22)$$

Substituting Equation (22) into (20)–(21), we have

$$\left(1 - \frac{m_i(t, \tau)}{n_i(t, \tau)}\right)p_i \simeq \left(1 - \frac{m_j(t, \tau)}{n_j(t, \tau)}\right)p_j. \quad (23)$$

PROPOSITION 1. For data-intensive and interleaved flows, with the WTP scheduler described in (2), the delay at each hop approximates proportional delay differentiation (PDD) model.

PROOF. Based on Equation (23), we have

$$\frac{n_i(t, \tau) - m_i(t, \tau)}{n_j(t, \tau) - m_j(t, \tau)} = \frac{p_j n_i(t, \tau)}{p_i n_j(t, \tau)}. \quad (24)$$

Let $\lambda_i(t, \tau)$ be the average arrival rate of class i within the time period $[t, \tau]$. Then $\lambda_i(t, \tau) = \frac{n_i(t, \tau)}{\tau}$. By Little's Law, we know that

$$\lambda_i(t, \tau) \times \bar{d}_i(t, \tau) = \bar{L}_i(t, \tau), \quad (25)$$

where $\bar{d}_i(t, \tau)$ and $\bar{L}_i(t, \tau)$ are average delay and average queue length of class i packets within the time period $[t, \tau]$. Since $n_i(t, \tau) - m_i(t, \tau)$ is the number of backlogged packets from class i , the ratio of average queue length is given by

$$\frac{\bar{L}_i(t, \tau)}{\bar{L}_j(t, \tau)} = \frac{n_i(t, \tau) - m_i(t, \tau)}{n_j(t, \tau) - m_j(t, \tau)}. \quad (26)$$

From (24), (25), and (26) we have

$$\frac{\bar{d}_i(t, \tau)\lambda_i(t, \tau)}{\bar{d}_j(t, \tau)\lambda_j(t, \tau)} = \frac{n_i(t, \tau) - m_i(t, \tau)}{n_j(t, \tau) - m_j(t, \tau)} = \frac{p_j n_i(t, \tau)}{p_i n_j(t, \tau)}. \quad (27)$$

Thus,

$$\frac{\bar{d}_i(t, \tau)n_i(t, \tau)}{\bar{d}_j(t, \tau)n_j(t, \tau)} = \frac{p_j n_i(t, \tau)}{p_i n_j(t, \tau)}. \quad (28)$$

Therefore,

$$\frac{\bar{d}_i(t, \tau)}{\bar{d}_j(t, \tau)} = \frac{p_j}{p_i}. \quad (29)$$

REFERENCES

- AAD, I. AND CASTELLUCCIA, C. 2001. Differentiation mechanisms for IEEE 802.11. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*. 209–218.
- ABDELZAHER, T., SHIN, K., AND BHATTI, N. 2002. Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Trans. Parallel Distrib. Syst.* 13.
- AHN, G.-S., CAMPBELL, A., VERES, A., AND SUN, L.-H. 2002a. Swan: Service differentiation in stateless wireless ad hoc networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.
- AHN, G.-S., CAMPBELL, A. T., VERES, A., AND SUN, L.-H. 2002b. Supporting service differentiation for real-time and best-effort traffic in stateless wireless ad hoc networks (swan). *IEEE Trans. Mobile Comput.* 1, 3, 192–207.
- BANCHS, A., RADIMIRSCH, M., AND PEREZ, X. 2002. Assured and expedited forwarding extensions for IEEE 802.11 wireless LAN. In *Proceedings of the IEEE Annual Workshop on Quality of Service*. 237–246.
- BARRY, M. G., CAMPBELL, A. T., AND VERES, A. 2001. Distributed control algorithms for service differentiation in wireless packet networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies*.
- CHEN, W. T., JIAN, B. B., AND LO, S. C. 2002. An adaptive retransmission scheme with qos support for the IEEE 802.11 Mac enhancement. In *Proceedings of the IEEE VLSI Test Symposium*.
- CHUNG, K. L. 2000. *A Course in Probability Theory Revised*, 2 ed. Academic Press.
- DIAO, Y., GANDHI, N., HELLERSTEIN, J., PAREKH, S., AND TILBURY, D. 2002. MIMO control of an apache web server: Modeling and controller design. In *Proceedings of the American Control Conference (ACC)*.
- DIAO, Y., HELLERSTEIN, J., PAREKH, S., GRIFFITH, R., KAISER, G., AND PHUNG, D. 2005. A control theory foundation for self-managing computing systems. *IEEE J. Select. Areas Comm.* 23, 12, 2213–2222.
- DOVROLIS, C. AND RAMANATHAN, P. 1999. A case for relative differentiated services and the proportional differentiation model. *IEEE Netw.* 13, 5.
- FELLER, W. 1971. *An Introduction to Probability Theory and Its Applications*, 2 Ed. Vol. 2. Wiley.
- GRILO, A. AND NUNES, M. 2002. Performance evaluation of IEEE 802.11e. In *Proceedings of the IEEE International Symposium on Personal Indoor and Mobile Radio Communication*. 511–517.
- HE, W. AND NAHRSTEDT, K. 2006. Impact of upper layer adaptation on end-to-end delay management in wireless ad hoc networks. In *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS06)*.
- HELLERSTEIN, J. L. 2004. Designing in control engineering of computing systems. In *Proceedings of the American Control Conference*.
- HELLERSTEIN, J. L., DIAO, Y., PAREKH, S., AND TILBURY, D. M. 2004. *Feedback Control of Computing Systems*. IEEE Press/Wiley Interscience.
- IEEE COMPUTER SOCIETY. 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
- KARAMANOLIS, C., KARLSSON, M., AND ZHU, X. 2005. Designing controllable computer systems. In *Proceedings of the USENIX Workshop on Hot Topics in Operating Systems*. 49–54.
- LEE, S.-B., AHN, G.-S., ZHANG, X., AND CAMPBELL, A. T. 2000. Insignia: an ip-based quality of service framework for mobile ad hoc networks. *J. Parallel Distrib. Comput.* 60, 4, 374–406.
- LI, B. 2005. End-to-end fair bandwidth allocation in multi-hop wireless ad hoc networks. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS'05)*. 471–480.
- LI, B. AND NAHRSTEDT, K. 1999. A control-based middleware framework for quality-of-service adaptations. *IEEE J. Select. Areas Comm.* 17, 9, 1632–1650.
- LJUNG, L. 1999. *System Identification: Theory for the User* (2nd Edition).
- LU, C., ABDELZAHER, T., STANKOVIC, J., AND SON, S. 2001. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*.

- LU, Y., ABDELZAHER, T., AND SAXENA, A. 2004. Design, implementation, and evaluation of differentiated caching services. *IEEE Trans. Parallel Distrib. Syst.* 15, 5 (May).
- LUO, H., LU, S., AND BHARGHAVAN, V. 2000. A new model for packet scheduling in multihop wireless networks. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking*.
- MANGOLD, S., CHOI, S., MAY, P., KLEIN, O., HIERTZ, G., AND STIBOR, L. 2002. IEEE 802.11e wireless lan for quality of service. In *Proceedings of the Conference on European Wireless*.
- SHEU, S. T. AND SHEU, T. F. 2001. A bandwidth allocation/sharing/extension protocol for multimedia over IEEE 802.11 ad hoc wireless lans. *IEEE J. Select. Areas Comm.* 19, 10, 2065–2080.
- SOBRINHO, J. L. AND KRISHNAKUMAR, A. S. 1999. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *IEEE J. Select. Areas Comm.* 17, 8, 1353–1368.
- VAIDYA, N. H., BAHL, P., AND GUPTA, S. 2000. Distributed fair scheduling in a wireless LAN. In *Proceedings of the Annual Conference on Mobile Computing and Networking*. 167–178.
- XUE, Q. AND GANZ, A. 2004. Proportional service differentiation in wireless lans using spacing-based channel occupancy regulation. In *ACM Multimedia*.
- YANG, Y. AND KRAVETS, R. 2004. Throughput guarantees for multi-priority traffic in ad hoc networks. In *Proceedings of the International Conference on Mobile Ad hoc and Sensor Systems (MASS)*.
- ZHANG, Y., BESTAVROS, A., GUIRGUIS, M., MATTA, I., AND WEST, R. 2005. Friendly virtual machines: leveraging a feedback-control model for application adaptation. In *Proceedings of the 1st International Conference on Virtual Execution Environments (VEE)*, M. Hind and J. Vitek, Eds. ACM, 2–12.

Received March 2007; revised August 2007, October 2007; accepted February 2008