

Chapter 1

Delay Management in Wireless Networks

Wenbo He and Yuan Xue and Klara Nahrstedt

Abstract A good amount of research has been developed to support QoS issues in IEEE 802.11 ad hoc networks. In this chapter, we address QoS management issues through multiple layers for real-time multimedia applications. We investigate the adaptation mechanisms at middleware layer and MAC layer to dynamically adjust the service classes for applications by feedback control theory. Based on our investigation, we propose a cross-layer end-to-end delay management framework for multimedia traffic over multi-hop wireless networks.

1.1 Introduction

A key vision of next-generation wireless systems is to support new emerging distributed multimedia services such as voice-over-IP and video-on-demand anytime anywhere. Some applications (e.g. multimedia applications) are very sensitive to end-to-end delay provided by the communication environment. The control of end-to-end delay over 802.11 wireless networks presents a number of technical challenges. First, wireless networks are typically bandwidth constrained in comparison with their traditional wireline peers. Second, the time-varying error characteristics and time-varying channel capacity at the physical layer makes it difficult, if not impossible to provide hard end-to-end delay guarantees. Third, user mobility may

Wenbo He
Department of Computer Science, 3101 Siebel Center, 201 N. Goodwin Ave., Urbana, IL 61801
e-mail: wenbohe@uiuc.edu

Yuan Xue
383 Jacobs Hall, Vanderbilt University, VU Station B 351824, Nashville, TN 37235 e-mail:
yuan.xue@vanderbilt.edu

Klara Nahrstedt
Department of Computer Science, 3104 Siebel Center, 201 North Goodwin Avenue, Urbana, IL
61801 e-mail: (klara@cs.uiuc.edu)

induce signal fading that in turn can trigger rapid degradation in the delivered service quality.

A real time multimedia application usually has QoS requirements on end-to-end latency. For example, in telephony, one-way delay requirement is from 25ms to 400ms, depending on the requirement of voice quality and existence of echo canceler. It is challenging to manage end-to-end latency to meet the QoS requirement in wireless multi-hop networks.

In the following sections, we will present a cross-layer delay management framework based on the findings in [1, 2]. This framework provides comprehensive and flexible control of end-to-end latency over wireless ad hoc networks as shown in Fig. 1.1.

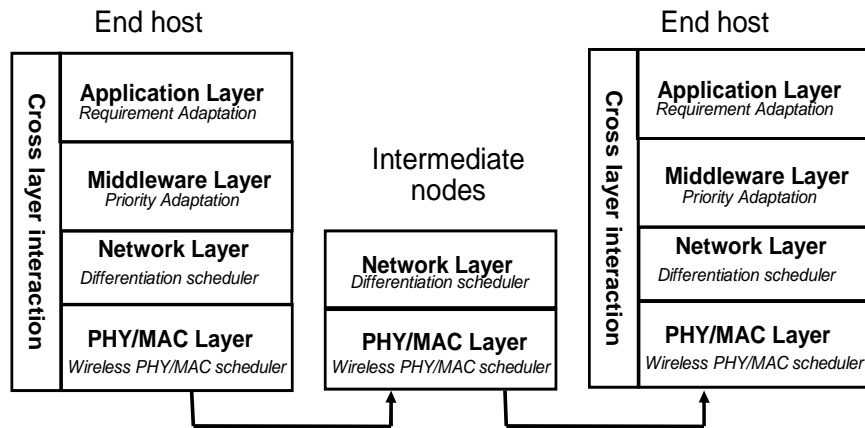


Fig. 1.1 Communication over multi-hop wireless ad hoc networks

1.2 Background

Current efforts on developing delay management and QoS support for wireless networks are mostly focused at the MAC layer. For example, the work of [3, 4] focused on differentiated MAC layer scheduling under IEEE 802.11 DCF. The work of [5, 6, 7] studied the fair scheduling in wireless networks. These MAC layer solutions support delay management within one hop and lack the architectural flexibility to accommodate end-to-end application-specific QoS requirements in time-varying wireless networking environments.

Some other efforts provide an end-to-end delay support by adapting traditional QoS models for wireless network. For example, based on the IntServ model, IN-SIGNIA [8] utilizes resource signaling protocol to reserve per-flow resources. Alternatively, SWAN [9] follows the absolute DiffServ QoS model by defining two

service classes: *real-time* and *best-effort* traffic. A measurement-based admission control is performed to support real-time traffic while the ECN bit is marked to reflect the network support for best-effort traffic. Both these approaches use per-application admission control, thus they have poor scalability and may suffer false admission due to the imprecise resource estimation in the highly dynamic wireless environments.

In light of the limitations of existing approaches, it is clear that a complete and efficient end-to-end delay management architecture for wireless network would require (1) lightweight and scalable QoS support with minimum or no negotiation overheads; (2) agile and responsive delay management tools to monitor network resources and make appropriate adaptation decisions for applications so that their individual delay requirements can be satisfied.

1.3 Cross-layer Delay Management Architecture

The cross-layer delay management framework is presented in Fig. 1.2. The architecture operates from the MAC layer up to the middleware layer. At the MAC and network level, packets from different service classes are processed differently via per-hop forwarding mechanisms (e.g., packet scheduling and queue management) to achieve proportional delay differentiation (PDD). At the middleware layer, a monitor component monitors the performance of applications. Based on the monitoring results, it performs appropriate service class adaptation so that different applications are able to meet their required end-to-end latency specifications.

1. *At the middleware layer in the wireless end hosts:*

- Based on the observed end-to-end delay and the delay specifications of the applications, the *Priority Adaptor* decides the appropriate priority for each application and notifies the *Classifier*. The *Classifier* in the middleware determines the service classes for packets according to their priorities. 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.
- The packets from applications are delivered through the middleware layer, where the *Classifier* marks the packets with their corresponding service classes by mapping the priorities to 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 *Monitor* monitors the performance of each service class and notifies the *Priority Adaptor* of the observed violations of end-to-end delay.

2. *At the network layer in the routing nodes:*

- The *Queue Management* component allocates buffer spaces and marks or drops packets. It deals with packet loss rate differentiation.

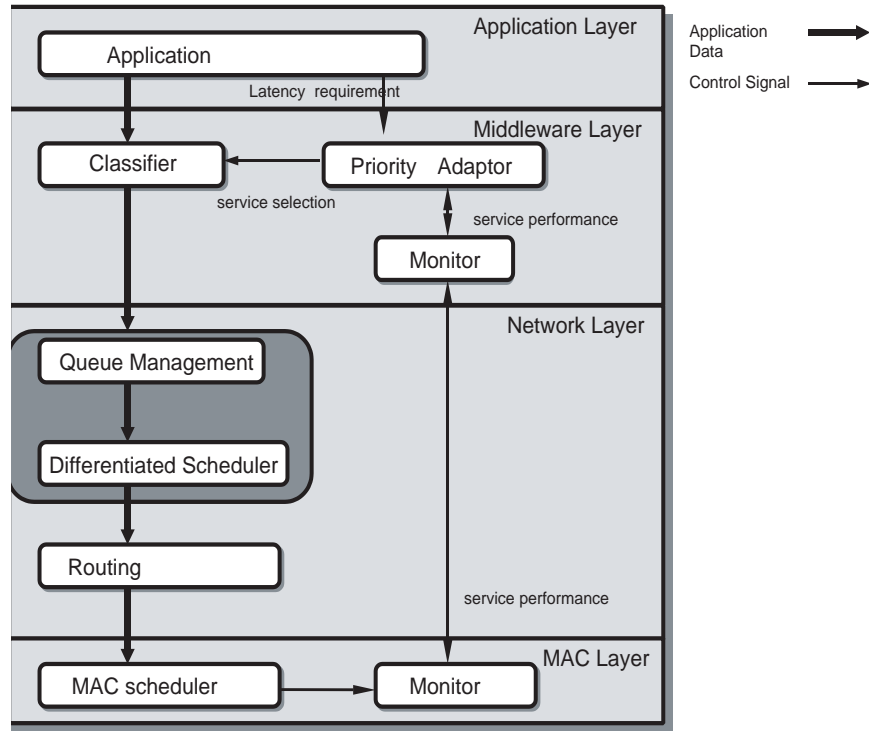


Fig. 1.2 End-to-end Delay Management Architecture

- The *Differentiated Scheduler* selects a packet to transmit. It performs packet-level QoS enforcement, allocates bandwidth for different flows and provides delay differentiation.
3. *At the MAC layer:*
- The MAC layer scheduler coordinates with the differentiation scheduler at the network layer to provide timely channel access for packets from different wireless nodes. Since MAC layer and network layer schedulers are tightly coupled with each other, we propose a cross layer design in the architecture.

1.4 Priority Adaptation in Middleware

1.4.1 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 because the end-to-end latency caused by traversing the upper layers (network and above) is small at the end nodes and can be ignored. The sender attaches time-stamps when sending the packets to the destination. As the sender gets ACK from the destination, 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.

1.4.2 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.

1.4.3 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 [10], we determine the model which captures the relationship between priority and end-to-end delay. After we obtain the control model, we design 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 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 ad hoc network, and the nature of randomized algorithms 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 1.1.

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

Table 1.1 The desired properties and the controller design criteria

Figure 1.3 shows the whole control loop including *Priority Adaptor* (C), the open loop wireless multi-hop 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.

We obtain the parameters of the *Priority Adaptor* in three 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) Design of adaptive controller

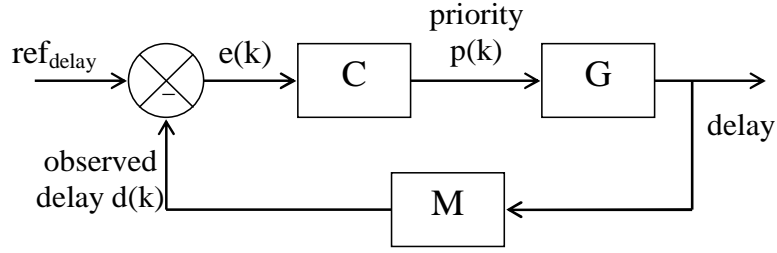


Fig. 1.3 The block diagram of the closed-loop control system for end-to-end delay

which dynamically changes parameters of the controller, so that the controller is able to adapt with different network load conditions.

1.4.3.1 Off-line 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 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:

$$d(k+1) = b_0 p(k) + a_1 d(k) \quad (1.1)$$

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 (1.1) is independent of the number of nodes on

the route as long as the route between two end nodes are fixed. Note that the units of parameters in the model are casted to make both sides of equation have consistent units, e.g. the unit of b_0 is second, a_1 is a constant.

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 (1.1) is

$$G(z) = \frac{d(z)}{p(z)} = \frac{b_0}{z - a_1} \quad (1.2)$$

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 $a_1 = 0.2514$.

1.4.3.2 PI Controller Design

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

$$p(k+1) = k_p e(k) + k_i \sum_{t=0}^k e(t) \quad (1.3)$$

Then,

$$p(k) = k_p e(k-1) + k_i \sum_{t=0}^{k-1} e(t) \quad (1.4)$$

If we subtract (1.4) from (1.3), we have

$$p(k+1) - p(k) = k_p (e(k) - e(k-1)) + k_i e(k) \quad (1.5)$$

Then we get

$$p(k+1) = p(k) + (k_p + k_i)e(k) - k_p e(k-1) \quad (1.6)$$

The z-transform of the priority controller (1.6) is given as

$$C(z) = \frac{(k_p + k_i)z - k_p}{z(z-1)} \quad (1.7)$$

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

$$G(z) = \frac{d(z)}{p(z)} = \frac{0.02425}{z - 0.2514} \quad (1.8)$$

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

$$p(k+1) = p(k) + 3.36e(k) - 1.85e(k-1) \quad (1.9)$$

1.4.3.3 Adaptive Controller Design

We have described the design of the 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 system's 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 1.4 shows a block diagram for the self-tuning adaptor to control priorities of multimedia applications. Comparing Figure 1.4 with Figure 1.3, we add two blocks: an on-line estimation of identified parameter, and self-tuning regulator. In Figure 1.4, we have $C = \frac{(k_p+k_i)z-k_p}{z(z-1)}$ and $G = \frac{b_0}{z-a_1}$.

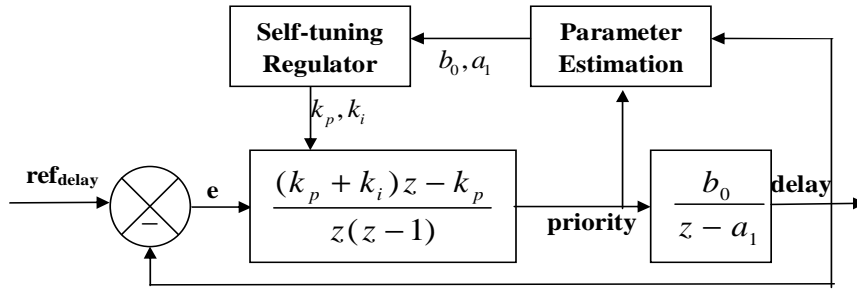


Fig. 1.4 Block diagram of a self-tuning adaptor

1.5 Cross-layer Proportional Delay Differentiation Scheduler

The middleware layer priority adaptation requires the support of service differentiation from the lower level of the network stack. To enforce the service differentiation in wireless network, a coordination between network layer and MAC layer is needed. In this section, we present a cross-layer delay differentiation scheduling scheme. Here we adopt a proportional delay differentiation model.

The model of the proportional service differentiation was first introduced as a per-hop-behavior (PHB) for DiffServ in wireline networks [11]. It states that certain class performance metrics should be proportional to the differentiation parameters. In particular, if we consider the case of delay differentiation, in a network with C service classes the proportional delay differentiation model imposes the following constraints for all pairs of classes.

$$\frac{\bar{d}_i(t, t + \tau)}{\bar{d}_j(t, t + \tau)} = \frac{\delta_j}{\delta_i}, \text{ for all } i \neq j \text{ and } i, j \in \{1, 2, \dots, C\} \quad (1.10)$$

where δ_i is the service differentiation parameter for class i , and $\bar{d}_i(t, t + \tau)$ is the average delay for class i , ($i = 1, 2, \dots, C$) in the time interval $(t, t + \tau)$, where τ is the monitoring time scale.

The basic idea of proportional differentiation is that, even though the actual quality level of each class may vary with traffic loads, the quality ratio between classes should remain constant in various time-scales. In addition, such a quality ratio can be controlled by setting the service differentiation parameters, which provides flexible class provisioning and management. Under certain conditions (*i.e.*, the network is well provisioned), applications with absolute delay requirements can select appropriate service classes to meet their requirements [12], even though the network offers only relative differentiation.

One of the packet scheduling algorithms that can realize the proportional delay differentiation model in a short time-scale is the *waiting time priority* (WTP) scheduler [13]. In this algorithm, a packet is assigned with a weight, which increases proportionally to the packet's waiting time. Service classes with higher differentiation parameters have larger weight-increase factors. The packet with the largest weight is served first in non-preemptive order. Formally, let $wt_{pkt}(t)$ be the waiting time of a packet pkt of class i at time t , we define its *normalized waiting time* $\hat{wt}_{pkt}(t, i)$ at time t as follows.

$$\hat{wt}_{pkt}(t, i) = wt_{pkt}(t) \cdot \delta_i \quad (1.11)$$

The normalized waiting time is then used as the weight for scheduling. The packet with the largest weight is then selected by the WTP scheduler for transmission. Formally, at time t it will transmit the packet pkt which satisfies

$$pkt = \arg \max_{pkt \in \mathcal{P}} \hat{wt}_{pkt}(t, i) \quad (1.12)$$

where \mathcal{P} is the set of backlogged packets. It is shown that WTP scheduler is able to approximate the proportional delay differentiation model in wireline networks under heavy traffic condition [13].

Here we introduce the *proportional service differentiation model* into the domain of wireless LAN. Different from wireline networks, where flows from the same router contend for the same wireline link, in wireless LANs not only do the flows originating from the same node contend with each other, the flows from different nodes also contend for the same wireless channel. To extend the concept of proportional service differentiation to the wireless LAN, the flows among different pairs of nodes are considered. Specifically, our proportional delay differentiation model for wireless LANs states that the relation Eq. (1.10) holds for all flows within the wireless LAN no matter whether they originate from the same node or not.

As a result of the distributed medium sharing, packet scheduling needs the cooperation among all the nodes. This is in contrast to wireline networks where packets that need to be scheduled originate from the same router, and hence the packet scheduling decision can be made by the router itself only considering its own packets. We argue that delay differentiation in wireless LANs can only be achieved through a *joint packet scheduling* at the network layer and distributed coordination at the MAC layer. Therefore, we present a *cross-layer waiting time priority scheduling (CWTP) algorithm* which is able to achieve proportional delay differentiation in wireless LANs.

The *cross-layer waiting time priority scheduling algorithm (CWTP)* divides the scheduling task into two parts which are performed at two layers in the network stack. At the network layer, *intra-node* scheduling at node n selects a packet pkt_n^* with the longest normalized waiting time, *i.e.*, a packet pkt_n^* which satisfies

$$pkt_n^* = \arg \max_{pkt \in \mathcal{P}_n} \hat{w}t_{pkt}(t, i) \quad (1.13)$$

where \mathcal{P}_n is the set of all backlogged packets at node n . At the MAC layer, *inter-node* scheduling selects packet pkt^* among pkt_n^* , which satisfies

$$pkt^* = \arg \max_{pkt_n^* \in \mathcal{N}} \hat{w}t_{pkt_n^*}(t, i) \quad (1.14)$$

where \mathcal{N} is the set of wireless nodes.

Such an intra- and inter-node scheduling algorithm can fit well the environment of wireless LANs. In particular, the intra-node scheduling can be implemented via network layer packet scheduling at each individual node and the inter-node scheduling can be implemented via media access control (MAC) which coordinates packet transmissions among nodes. Fig. 1.5 illustrates such a cross-layer scheduling architecture. In this architecture, the packet scheduler at the network layer and the distributed coordination function at the MAC layer are coordinated using normalized packet waiting time $\hat{w}t$ as a cross-layer signal.

At MAC layer, in order to transmit the packet with larger normalized waiting time before the ones with smaller normalized waiting time, we map the normalized waiting time $\hat{w}t$ to the backoff time b via function $b = \Phi(\hat{w}t)$. In [14], we present two

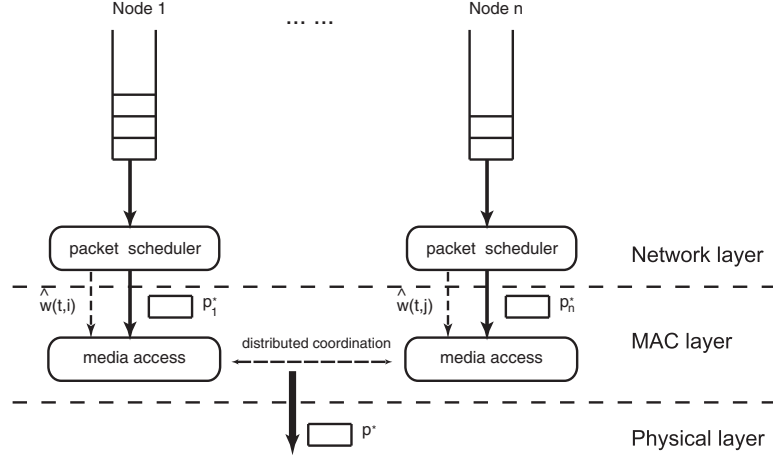


Fig. 1.5 Cross-layer architecture

mapping schemes, namely linear mapping and piece-wise linear mapping schemes to implement the function $\Phi(\hat{w}t)$.

In linear mapping scheme, the normalized waiting time of a packet is mapped to its MAC layer backoff time via a linear function. Formally, let us consider a linear function $\phi(x) : \mathfrak{R}^+ \rightarrow \mathfrak{R}$,

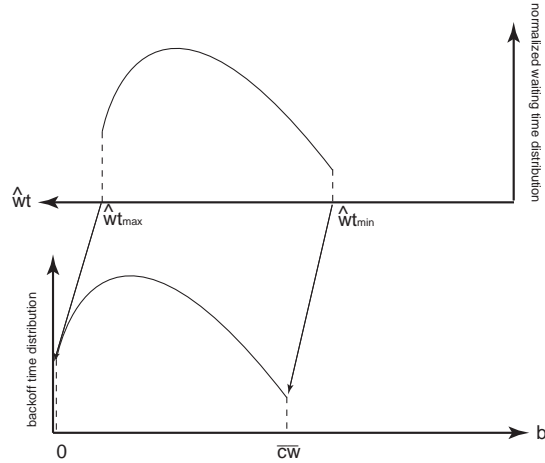
$$\phi(x) = \beta - \alpha \cdot x \quad (1.15)$$

where $\alpha, \beta > 0$ are parameters of this linear function. To ensure it to be a non-negative integer, the backoff time b (in number of time slot) of a packet with normalized waiting time $\hat{w}t$ is chosen as follows,

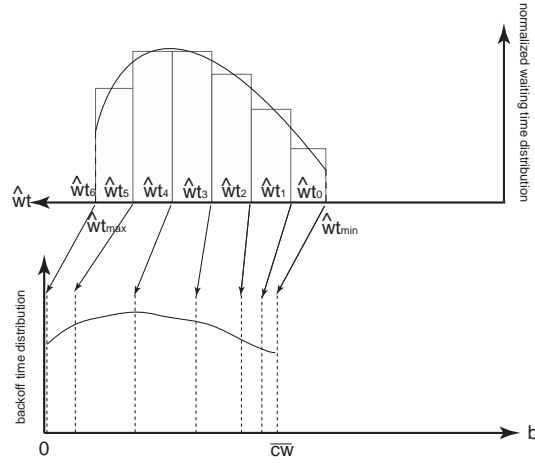
$$b = \Phi(\hat{w}t) = \lceil [\phi(\hat{w}t)]^+ \rceil \quad (1.16)$$

where $[x]^+ = \max(0, x)$ and $\lceil \cdot \rceil$ is the ceiling operation. These two operations round up the value of $\phi(\hat{w}t)$ to a non-negative integer. It is obvious that α and β determines the effectiveness of the mapping function, and thus the performance of the cross-layer scheduling algorithm. We present a dynamic tuning algorithm of α and β . Let $\bar{c}\bar{w}$ be the expected value of contention window under IEEE 802.11 DCF without differentiation. The backoff time b is uniformly chosen from $[0, \bar{c}\bar{w})$. Let $\hat{w}t_{max}$ and $\hat{w}t_{min}$ be the maximum and minimum normalized waiting time respectively. Preferably, the maximum normalized waiting time $\hat{w}t_{max}$ can be mapped to the smallest backoff time (0) for efficient channel utilization; and $\hat{w}t_{min}$ can be mapped to $\bar{c}\bar{w}$ for similar contention behavior as IEEE 802.11 without differentiation.

Linear mapping scheme neglects the fact that the distribution of the normalized waiting time can be non-uniform. If there is a higher density over a certain interval of time, then it will increase the possibility of packets with different normalized waiting time being mapped into the same backoff time. It can also increase the pos-



(a) Linear mapping



(b) Piecewise linear mapping

Fig. 1.6 Linear mapping and piecewise linear mapping: a comparison.

sibility of packet collision at the MAC layer. To address above problems, we present a piecewise linear mapping algorithm which considers the effect of normalized waiting time distribution. In piecewise linear mapping algorithm the normalized waiting times $\hat{w}t$ are divided into L intervals of equal lengths defined by points $\hat{w}t_{min} = \hat{w}t_0, \hat{w}t_1, \hat{w}t_2, \dots, \hat{w}t_L = \hat{w}t_{max}$. During each interval, function $\Phi_i(\hat{w}t) = \lceil [\beta_i - \alpha_i \cdot \hat{w}t]^+ \rceil$ will be used for the mapping. Fig. 1.6 compares these two mapping algorithms.

We simulate the CWTP algorithm under both linear mapping and piecewise linear mapping schemes on a variety of network settings in ns-2 [15]. In the simulation, the number of nodes (N) is a parameter to show how CWTP scales to the network

size. Each node in the wireless LAN sets up a connection. The transmission rate of each flow is configured to give the network an aggregated load about 1500Kbps.

We first show the impact of network size on CWTP algorithm. In this experiment, 2 service classes with $\frac{\delta_2}{\delta_1} = 2$ are supported in the network. Fig. 1.7 shows the differentiation index I with different numbers of nodes in the network. The differentiation index (I) is defined as the ratio of the average delay of the two service classes. That is

$$I = \frac{\bar{d}_1}{\bar{d}_2} \quad (1.17)$$

where \bar{d}_i is the expected packet delay of service class i . This metric shows the effectiveness of the service differentiation – how close the differentiation result matches the differentiation goal. Ideally, in these experiments $I = 2$. We observe that both linear mapping and piecewise linear mapping schemes can lead the CWTP scheduling algorithm to achieve a delay differentiation index very close to the target value, when the network size is relatively small (the number of nodes $N < 20$). When the network size is large (e.g. $N = 50$), the piecewise linear mapping scheme performs much better than the linear mapping scheme.

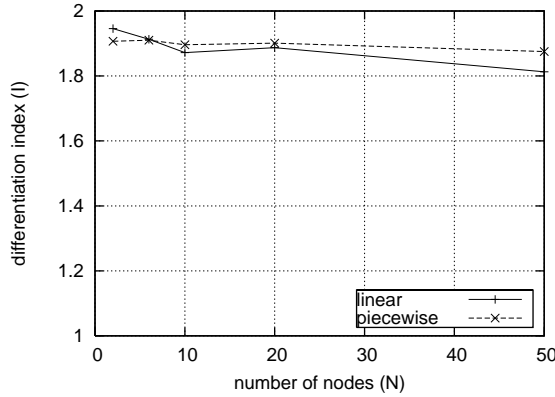


Fig. 1.7 Differentiation Index.

In Fig. 1.8, we show the instantaneous delay behaviors under these two schemes when $N = 10$. From these results, we observe that piecewise linear mapping scheme gives much more consistent and smooth delay behavior than linear mapping scheme. This is because with the consideration of normalized waiting time distribution, piecewise linear mapping significantly reduces the possibility of packet collision at the MAC layer.

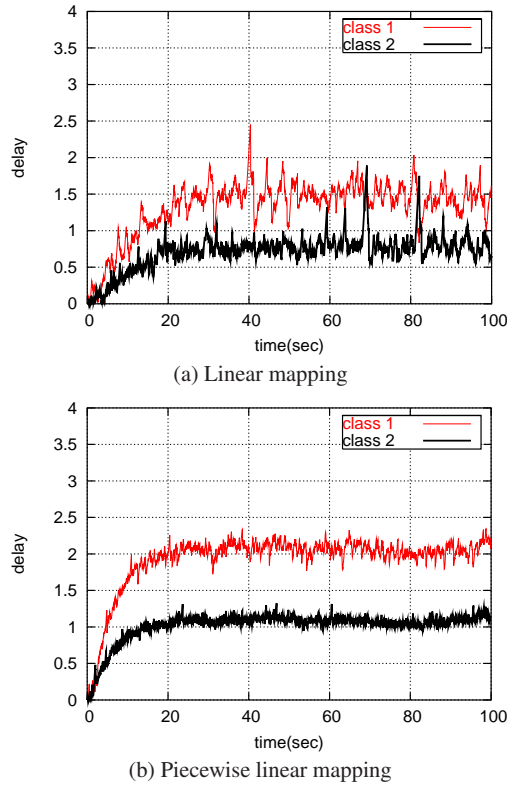
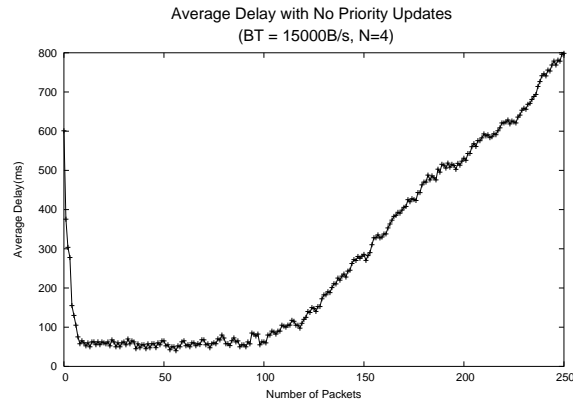


Fig. 1.8 Instantaneous delay behavior.

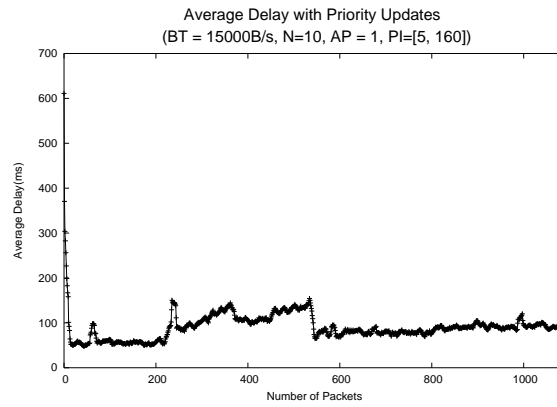
1.6 Evaluation and Implementation

We show the performance of the adaptation service integrated with the delay differentiation service over an IEEE 802.11-based wireless ad-hoc testbed implementation. In the experiment, we first start an audio application which has a QoS requirement in terms of maximum packet delivery delay. Then background UDP traffic with 15000 Bytes/s is started. From the results in Fig. 1.9, we see that the average delay increases quickly from 70ms to 800ms without service differentiation and adaptation. Using the service adaptation policy in the example and the underlying delay differentiation support, we observe that the average delay for the audio application was successfully bounded to < 150 ms.

Next, we use the testbed to evaluate end-to-end delay of multihop QoS (audio) traffic. In the experiment, the audio sampling rate is 256Kbps, and background UDP traffic is 100000Bytes/s. There are two hops between audio source and destination. We enable the PI controller as in equation (1.9) on our test bed. The targeted end-to-end delay is 60 millisecond ($ref_{delay} = 60$). Figure 1.10 shows the resulting end-



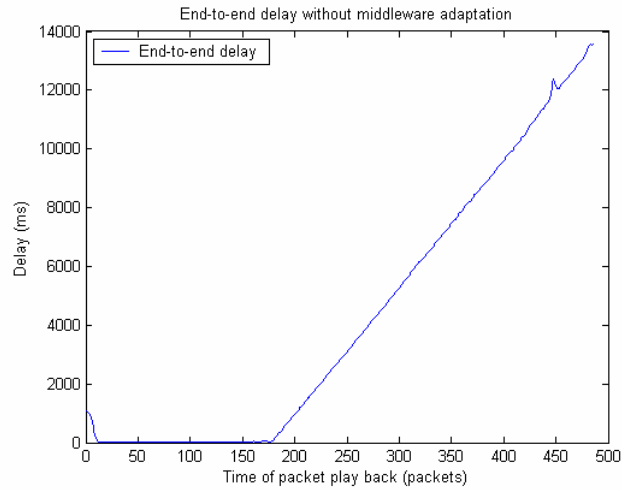
(a) Without service adaptation and differentiation



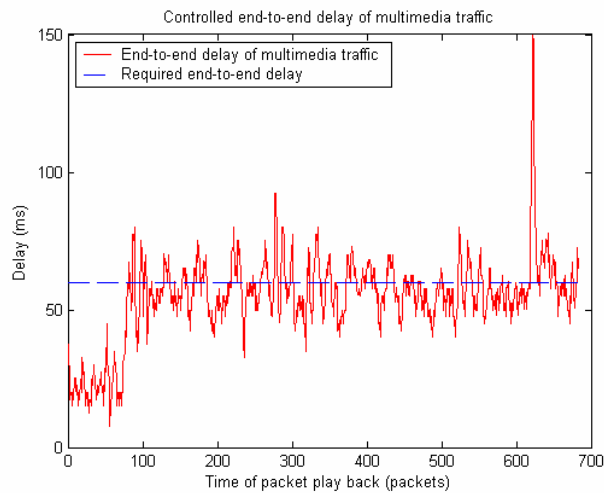
(b) With service adaptation and differentiation.

Fig. 1.9 Performance Comparison.

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 background traffic which compete with the application with QoS requirement for the network resources in wireless ad hoc environment. We also notice that the delay of most of the audio traffic is in the range from 40 to 80 ms. Therefore, the middleware adaptation method also achieves the latency and playback jitter control, which is very important for multimedia applications.



(a) End-to-end delay without priority update



(b) End-to-end delay under PI controller of priority update

Fig. 1.10 Instantaneous delay behavior.

1.7 Thoughts for Practitioners

In the cross-layer end-to-end delay control framework, the middleware-layer adaptation in Section 1.4 and network/MAC layer scheduling algorithm co-design in Section 1.5 are localized methods with small overhead. The middleware layer adaptation has the advantage that the adaptation mechanism is hardware independent, so users do not have to modify the hardware to achieve the QoS requirement. Further,

for wireless network with location-dependent contention, the cross-layer support at the network and MAC layer should be adopted.

1.8 Directions for Future Research

As the heterogenous networking environment where wireless communication and wired links coexist becomes popular, we will consider the QoS support in such a heterogeneous environment as a future research direction.

1.9 Conclusions

In this chapter, we have shown a multi-layer adaptation scheme for end-to-end delay control in multi-hop wireless networks. Based on this cross-layer design, we can control the end-to-end delay given sufficient network bandwidth.

1.10 Questions and Answers

1. What is a waiting time priority scheduler?
A: In this scheduling algorithm, a packet is assigned with a weight, which increases proportionally to the packet's waiting time. Service classes with higher differentiation parameters have larger weight-increase factors. The packet with the largest weight is served first in non-preemptive order.
2. Why joint packet scheduling at the network layer and distributed coordination at the MAC layer is needed?
A: Different from wireline networks, where flows from the same router contend for the same wireline link, in wireless LANs not only do the flows originating from the same node contend with each other, the flows from different nodes also contend for the same wireless channel. As a result of the distributed medium sharing, packet scheduling needs the cooperation among all the nodes. This is in contrast to wireline networks where packets that need to be scheduled originate from the same router, and hence the packet scheduling decision can be made by the route itself only considering its own packets.
3. Describe the function of each component in the cross-layer delay management architecture:
A: At middleware layer in the end hosts, the Adaptor decides the appropriate service class for each application and notifies the Classifier. The packets from applications are delivered through the middleware layer, where the Classifier marks the packets with their corresponding service class. The Monitor monitors the per-

formance of each service class and notifies the Adaptor of the observed violations of end-to-end delay.

At network layer in routing nodes, the Queue Management component allocates buffer spaces and marks or drops packets. It deals with packet loss rate differentiation. The Differentiated Scheduler selects a packet to transmit. It performs packet-level QoS enforcement, allocates bandwidth for different flows and provides delay differentiation.

At MAC layer, a scheduler coordinates with the differentiation scheduler at the network layer to provide timely channel access for packets from different wireless nodes.

4. Why piecewise linear mapping algorithm performs better than linear mapping algorithm.

A: Linear mapping scheme neglects the fact that the distribution of the normalized waiting time can be non-uniform. If there is a higher density over a certain interval of time, then it will increase the possibility of packets with different normalized waiting time being mapped into the same backoff time. It can also increase the possibility of packet collision at the MAC layer.

5. Why Priority Adaptor is in the form of PI controller?

A: 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 multimedia traffic over wireless ad hoc networks, 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 algorithms 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 use D controller in the middleware adaptation.

6. What is the major advantage of middleware adaptation and what is its disadvantage?

A: The middleware layer adaptation has the advantage that the adaptation mechanism is hardware independent, so users do not have to modify hardware to achieve QoS requirement. However, it is not enough for contention scenarios, where lower layer (e.g. Network/MAC layer) scheduling should be adopted.

7. What is the purpose of model identification in middleware Priority Adaptor design?

A: 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. However, it is difficult to obtain the relationship between end-to-end delay and a given traffic load 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.

8. Why the number of service classes cannot be very large?

A: Though the priorities can take a large range of values, the number of service classes can be small. This is because at the network layer, the Queue Management component maintains several queues representing different service classes. The Differentiated Scheduler selects the packet with the largest normalized waiting time to serve/forward. The Differentiated Scheduler only needs to compare the normalized waiting time of packets in the head of queues. For the sake of efficiency in network layer, usually we choose a small number of service classes.

References

1. W. He and K. Nahrstedt, "Impact of upper layer adaptation on end-to-end delay management in wireless ad hoc networks," in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS06)*, April 2006.
2. Y. Xue, K. Chen, and K. Nahrstedt, "Distributed end-to-end proportional delay differentiation in wireless lan," in *IEEE International Conference on Communications (ICC)*, 2004.
3. I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," in *Proc. of IEEE INFOCOM*, 2001.
4. D. Gu and J. Zhang, "Qos enhancement in ieee 802.11 wireless local area networks," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 120 – 124, June 2003.
5. H. Luo, S. Lu, and V. Bharghavan, "A New Model For Packet Scheduling in Multihop Wireless Networks," in *Proc. of ACM Mobicom*, 2000, pp. 76–86.
6. N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed fair scheduling in a wireless lan," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 167–178.
7. D. Qiao and K. Shin, "Achieving Efficient Channel Utilization and Weighted Fairness for Data Communications in IEEE 802.11 WLAN under the DCF," in *Proc. of The Tenth International Workshop on Quality of Service (IWQoS)*, 2002.
8. S.B. Lee, A. Gahng-Seop, X. Zhang and A.T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks," *Journal of Parallel and Distributed Computing, Special issue on Wireless and Mobile Computing and Communications*, vol. 6, no. 4, pp. 374–406, 2000.
9. G.-S. Ahn, A. Campbell, A. Veres, and L. Sun, "Supporting Service Differentiation for Real-Time and Best Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)," *IEEE Tran. on Mobile Computing*, Sep 2002.
10. L. Ljung, "System Identification: Theory for the User (2nd Edition)," 1999.
11. C. Dovrolis and P. Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model," *IEEE Network*, vol. 13, no. 5, pp. 26–34, 1999.
12. C. Dovrolis and P. Ramanathan, "Dynamic Class Selection: From Relative Differentiation to Absolute QoS," in *IEEE International Conference on Network Protocols*, 2001.
13. C. Dovrolis, P. Ramanathan, and D. Stiliadis, "Proportional differentiated services: Delay differentiation and packet scheduling," *IEEE/ACM Transactions on Networking*, vol. 10, pp. 12–26, Feb 2002.

14. Y. Xue, K. Chen, and K. Nahrstedt, "Achieving proportional delay differentiation in wireless lan via cross-layer scheduling," *Journal of Wireless Communications and Mobile Computing, special issue on Emerging WLAN Technologies and Applications*, vol. 4, no. 8, pp. 849–866, 2004.
15. "The Network Simulator - ns-2." [Online]. Available: <http://www.isi.edu/nsnam/ns/>