WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2004; **4**:835–848
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.258

A framework for efficient resource management in IEEE 802.11 WLANs

Liqiang Zhang*,† and Sherali Zeadally

High-Speed Networking Laboratory, Department of Computer Science, Wayne State University, 5143 Cass Avenue, State Hall, Detroit, MI 48202, U.S.A.

Summary

Currently deployed IEEE 802.11 WLANs work mostly with distributed coordination function (DCF) mode at the media access control (MAC) layer, which does not provide quality of service (QoS) support. The upcoming IEEE standard 802.11e achieves service differentiation by assigning different channel access parameters (CAPs) to different traffic classes at the MAC layer. However, such relative differentiation does not yield QoS guarantee. In practice, appropriately selecting CAPs *a priori* is difficult. Time-varying traffic loads also make the use of fixed CAPs, inefficient for both QoS support and channel utilization. In this work, we propose a novel architecture called HARMONICA, in which the access point (AP) dynamically selects the best CAPs for each traffic class to optimally match their QoS requirements. We present and discuss a simple admission control (AC) mechanism used by HARMONICA to avoid congestion. Our simulation results demonstrate that under an interference-free environment, HARMONICA can guarantee the QoS for all traffic classes while simultaneously achieving quasi-optimal channel utilization. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: quality of service; multimedia; IEEE 802.11; WLANs; admission control; adaptation algorithms; channel access parameters

1. Introduction

The IEEE 802.11 WLANs have recently achieved big success in the market of internet access. They have been widely deployed at homes, in offices and public hot spot locations. Most currently installed IEEE 802.11 WLANs operate using distributed coordination function (DCF) mode [17], which employs a carrier sense multiple access with collision avoidance (CSMA/CA) mechanism. As the basic access method of IEEE 802.11 standard, DCF has advantages of simplicity and robustness. However, DCF does not provide any quality of service (QoS) guarantee, since each station has only one first-in-first-out (FIFO) queue at the media access control (MAC) layer, and all stations contend to access the shared channel with same channel access parameters (CAPs) [17]. The lack of QoS support makes DCF inefficient to support emerging QoS-sensitive multimedia applications such as audio and video. Previous research efforts have investigated various approaches on DCF to achieve service differentiation either station-based or

*Correspondence to: Liqiang Zhang, High-Speed Networking Laboratory, Department of Computer Science, Wayne State University, 5143 Cass Avenue, State Hall, Detroit, M.I. 48202, U.S.A.
†E-mail: liqiang@cs.wayne.edu

Contract/grant sponsors: Sun Microsystems, Palo Alto; Microsoft Corporation, Seattle; Ixia, Corporation, Calabasas.

queue-based [1,12,24,28]. In the upcoming standard IEEE 802.11e [20], DCF is updated by enhanced distributed channel access (EDCA). With EDCA, each station can have up to four queues, mapped to different traffic classes. Service differentiation can be achieved by assigning different CAPs to different traffic queues. However, in practice, appropriate distribution of CAPs among different traffic classes to achieve target QoS is a challenging issue. Dynamic load in traffic classes also makes it difficult to provide guaranteed QoS when fixed CAPs are used.

Moreover, since DCF is a contention-based protocol, it suffers from collisions in the shared channel, which leads to poor resource utilization. Several research efforts have investigated various approaches to enhance the throughput of DCF by controlling the number of collisions and idle slots. One common approach is to dynamically tune one of the CAPs—the contention window (CW) size [2,7,8]. However, this approach relies on accurate estimation of active station numbers and distribution of frame lengths at run time, which may be quite difficult to realize in practice. Furthermore, such approach is not suitable for EDCA, which have multiple queues in each station, and these queues may have different CAPs.

Considering the error-prone, time-varying characteristics of wireless channels, providing hard QoS guarantee in IEEE 802.11 WLAN is an 'impossible task'. Even under the assumption of interference-free, providing QoS guarantees while simultaneously achieving optimal channel utilization is still challenging. In this work, we focus on IEEE 802.11e-based WLANs working in BSS mode [17]. We propose a novel architecture called HARMONICA (enhanced QoS support with admission control for IEEE 802.11 contention-based access), in which the access point (AP) exploits a link-layer quality indicator (LQI) to dynamically adjust CAPs to ensure the QoS of applications while simultaneously achieving quasi-optimal channel utilization. By periodically sampling LQI for each traffic class and utilizing two adaptation algorithms executing over different time scales, the AP selects CAPs that best match the QoS requirements of each traffic class and channel contention level. The AP then broadcasts the CAPs to all the stations in the wireless cell through beacon frames. A relative_adaptation algorithm aims at adjusting the relative differences between the CAPs among the different classes for the purpose of QoS guarantee. While a base_-adaptation algorithm is used to synchronously adapt the CAPs of all the classes (increase all or decrease all) to achieve high channel utilization. HARMONICA also employs a simple and flexible admission control (AC) mechanism to avoid congestion. Our simulation results have demonstrated that under an interference-free environment, the HARMONICA can effectively guarantee the QoS of different applications and achieving quasi-optimal aggregate throughput. The overhead (communication and computation) introduced by the HARMONICA is quite low, and the HARMONICA can be easily implemented on 802.11e-based products.

The rest of this paper is structured as follows. In Section 2, we briefly review related works and present the novel contributions of this paper. Section 3 describes the design and implementation issues of the HARMONICA architecture, including the LQI, two adaptation algorithms and the AC mechanism. Performance evaluations of our proposal are presented and discussed in Section 4. Finally, in Section 5, we make some concluding remarks.

2. Related Works and Novelty of Contributions

The lack of QoS support in IEEE 802.11 DCF has been reported by several research efforts [22]. Several service differentiation mechanisms have been proposed for DCF. Most of them achieve their goals by assigning different MAC layer parameters or backoff algorithms to different traffic classes [1,12,24,28]. The tunable parameters include those for contention windows (CWMin—minimal size of the contention window and CWMax—maximal size of the contention window), inter-frame space (IFS) and maximum frame length [1,24,28]. In the upcoming IEEE 802.11e standard, CWMin, CWMax and IFS are used to differentiate access categories [20] in EDCA mode. The service differentiation ability of an earlier version of EDCA (called EDCF[‡]) has been evaluated in several works [10,15,23]. However, service differentiation can provide service prioritization but not QoS guarantee. When the traffic load in each class changes, fixed CAPs assignment is not efficient both for QoS and resource utilization.

Several analytical models for DCF have been proposed [2,7,8,29]. Bianchi [2] models the binary backoff counter behavior of a tagged station as a discrete Markov chain model to analyze the maximum and

[‡]In earlier versions of 802.11e drafts (4.0[19] or earlier), EDCF (enhanced DCF) was used as the nomenclature for the contention-based channel access method.

saturation throughput performance of DCF. Cali *et al.* [7,8] derive a theoretical throughput bound by approximating the DCF with a *p*-persisted version of IEEE 802.11 protocol. Both Bianchi's and Cali's works reveal that to achieve maximum throughput, the initial CW's size (e.g. CWMin) should be dynamically tuned depending on active station numbers. Their computation of optimal CW's size also relies on accurate estimation of the frame length distribution (for calculation of average collision length.[§] Moreover, the above approaches are not suitable for EDCA, in which each station may have multiple queues, and these queues are associated with different CAPs.

Bononi *et al.* [5] propose a novel approach called asymptotically optimal backoff (AOB) to achieve run-time optimization of IEEE 802.11 WLAN performance. In AOB, each station performs an additional control (beyond carrier sensing and backoff algorithm) before any transmission attempt. This control is based on a new parameter named Probability of Transmission (P_T). The run-time optimal value of P_T for each transmission is calculated using the following formula:

$$P\_T(\text{opt}\_S\_U, S\_U, N\_A) = 1 - \min\left(1, \frac{S\_U}{\text{opt}\_S\_U}\right)^{N\_A} \tag{1}$$

where, S_U is the current slot utilization,[¶] opt_S_U is the optimal slot utilization to achieve maximum channel utilization and N_A is the number of unsuccessful transmission attempts of current frame. The values of S_U and N_A can be easily measured at run time. Through analysis and simulation, Bononi *et al.* reveal that the value of opt_S_U is independent of active station numbers. Thus the AOB mechanism achieves its goal without the need to know how many stations are active at run time. However, the calculation of opt_S_U still relies on the estimation of frame length distribution.[‖] It is worth noting that the AOB mechanism can also be extended to support service differentiation by introducing a priority level in the above formula [5]. As we discussed previously, pure service differentiation is not enough for QoS guarantee.

Probing-based AC [3,6,13,21] has been widely discussed as a promising admission control approach for DiffServ [4] QoS model. Besides its advantages of scalability and simplicity, probing-based AC has shortcomings of set-up delay, stealing bandwidth, false admission, thrashing and inaccuracy etc. [3,6]. Veres *et al.* [28] propose a novel distributed AC approach for IEEE 802.11 WLANs, which utilizes a virtual MAC (VMAC) algorithm and a virtual source (VS) algorithm to locally estimate achievable service quality. The advantage of this approach is that it does not introduce probing overhead into network, and it can be used in both ad hoc and BSS mode. However, since it is virtual, it only considers the effect of existing flows on the incoming flow, not the effects of the incoming flow on existing flows, which may introduce inaccuracy in making admission decisions.

Our previous work [31] proposed smart EDCF (SEDCF) to enhance the QoS support in IEEE 802.11 WLANs. This work differs from Reference [31] in several aspects. (1) In HARMONICA, we utilize a LQI, instead of MACQI [31] as the QoS feedback. The LQI measures drop rates, link-layer end-to-end delay and throughputs for specific classes at the link layer. A direct mapping between LQI and the QoS experienced at application layer is easily set up. (2) The HARMONICA utilize a base_adaptation algorithm to achieve quasi-optimal channel utilization, which was not achieved in Reference [31]. (3) By exploiting the information provided by LQI, a simple and flexible AC algorithm is introduced in the HARMONICA, which does not exist in our previous work.

The novel contributions of the HARMONICA are summarized as follows:

1. By utilizing LQI (locally measured at the AP) to conduct adaptations, we eliminate the necessity of explicitly gathering the experienced QoS informations from all the terminal stations. The validation of taking LQI as the QoS indicator for the whole cell is discussed later in this paper.
2. We achieve QoS guarantee and throughput optimization by exploiting two simple adaptation algorithms and one simple AC mechanism. Our approach avoids the need to build a complex, computation-intensive and parameters-relying model (if such a model is accomplishable) to achieve the same goal.

[§]The length of one collision is the transmission time of the frame whose length is the largest among all the frames involved in that collision.

[¶]Slot utilization is defined differently from channel utilization [5].

[‖]In Bononi *et al.*'s work, they assume the length of messages is geometrically distributed with a parameter q, the average length of collisions is then calculated based on this assumption (fixed with time changing). However, in real case, the distribution of the message/frame lengths may be time-variant also traffic-dependent.

3. HARMONICA: Design and Implementation Issues

3.1. Overview of the HARMONICA Architecture

The HARMONICA architecture is shown in Figure 1. As mentioned earlier, this architecture is for IEEE 802.11 WLANs operating in BSS mode, in which the AP plays the role of a bridge connecting the wireless cell and the wired network (e.g. internet). The core of HARMONICA includes three components, which all reside on the AP:

1. LQI monitor (LQIM), which periodically samples the LQI;
2. adaptation control unit (ApCU), which executes two adaptation algorithms based on LQI to get best CAPs for each traffic class; and
3. admission controller (AdC), which makes admission decisions based on the current LQI and admission requests.

The HARMONICA architecture supports flexible admission policies, which can be defined in policy control unit (PCU). The PCU also defines various thresholds, which are used by LQIM and AdC to judge the measured QoS. The adaptation execution unit (AEU) is rather simple, it just updates the CAPs for all traffic classes based on the calculation results of ApCU. The AEU resides on both the AP and terminal stations. Whenever a terminal station launches a new application (uplink**), the application is first classified to be best-effort or real-time flow. Best-effort flows are always automatically admitted, while real-time flows have to go through the AC process, which is taken care of by the admission control agent (ACA). The ACA sends admission request with QoS requirements of the flow to ACU of the AP and waits for the response. If the real-time flow is admitted, the response will include a class number (1–7), which will be used by the marker unit to mark the packets of the admitted flows. The packets can be marked by utilizing the differentiated services code point (DSCP) in the IP header [25]. The marked packets are sent to IEEE 802.11e MAC layer, which maintains four queues corresponding to up to four traffic classes (including best-effort flows). AC also works on downlink traffic as shown in Figure 1. It involves the cooperation of AdC and bridge unit (BU).†† If the downlink flows are from wired network (i.e. internet), the QoS requirement of these flows can be achieved by some mechanisms (such as RSVP [27,32]), which is beyond this work.

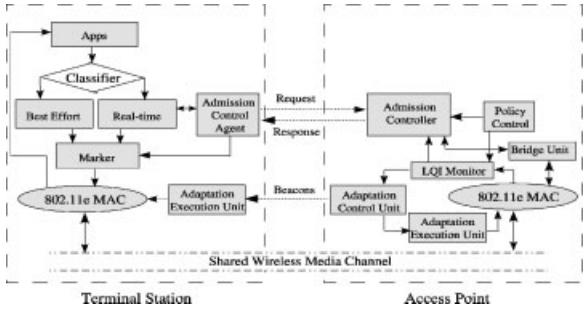

Fig. 1. The HARMONICA architecture.

3.2. The Link-Layer Quality Indicator (LQI)

LQI constitutes of the following parameters: drop rate, link-layer end-to-end delay and throughput. The former two parameters are sampled over a shorter time scale and are used by relative_adaptation algorithm. Throughput is sampled over a relatively longer time scale providing information for the base_adaptation algorithm. We discuss the validation of using LQI as the adaptation indicator at the end of this section.

3.2.1. Drop rate

The drop rate is sampled for each real-time traffic class from downlink traffic only. Packets are commonly dropped at two places: the queue/buffer between the LLC sublayer and MAC sublayer (we call this queue/buffer interface queue (IFQ) in the remainder of this paper) because of buffer overflow, and the MAC sublayer caused by retransmissions exceeding the maximal limits. The drop rate dr[i] is simply calculated using following formula:

$$\mathrm{dr}[i] = \frac{(\mathrm{IFQ\_drop\_number}[i] + \mathrm{MAC\_drop\_number}[i])}{\mathrm{llc\_sent\_number}[i]} \tag{2}$$

**We denote uplink traffic as the traffic sent from terminal stations to the AP, while the downlink traffic as the traffic sent from the AP to terminal stations.

††We use BU to abstract the bridging functions of the AP, which includes forwarding the outgoing, incoming and intra-cell traffic.

where, i is the index of traffic class, and llc_sent_number[i] is the total number of packets sent down from LLC sublayer for class i. Each drop_rate[i] is sampled for every F_{sw1} (the factor of the sampling window size) beacon periods. We define an estimator of exponentially weighted moving average (EWMA) to smooth the estimated value. At each sampling point, the estimator is updated according to following formulas:

$$\begin{cases} \text{dr\_curr\_avg}[i] = \text{dr\_last\_avg}[i] \times (1-\alpha) \\ \qquad\qquad + \text{dr}[i] \times \alpha \\ \text{dr\_last\_avg}[i] = \text{dr\_curr\_avg}[i] \end{cases} \tag{3}$$

where, α is the smoothing factor deciding the agility/stability of adaptations.

Adaptations need high and low thresholds. For this purpose, we defined two thresholds of drop rate for each real-time traffic class, namely, dr_thr_high[i] and dr_thr_low[i].

3.2.2. Link-layer end-to-end delay

The link-layer end-to-end delay (L_delay) is sampled for each real-time traffic class from downlink traffic only. We define the L_delay as the duration between the time point (t_1) when a frame is put into IFQ and the time point (t_2) when the ACK[‡‡] corresponding to that frame is received at the MAC layer. We calculate L_delay only for successfully transmitted frames. Since t_1 and t_2 can be simply sampled at link layer (in real implementation, these two values can be achieved by updating the NIC driver/firmware), L-delay can be easily calculated.

L_delay is a good indicator to application layer end-to-end delay, which can be shown by simple analysis. Please check appendix for details.

The QoS requirement of real-time applications in terms of delay can be defined by the following expression:

$$\Pr\{\text{D}[i] > \text{D\_thr}[i]\} \leq \text{D\_Pr\_thr}[i] \tag{4}$$

where, Pr{} means probability of {}, D[i] is the packet delay for traffic class i, D_thr[i] and D_Pr_thr[i] are the delay threshold and the maximum probability of exceeding it for traffic class i. To use the above expression to describe the QoS quality observed at the link layer, we periodically sample the rate of packets whose L_delay is larger than a high delay threshold. The rate is calculated using the following formula:

$$\text{d\_pr\_high}[i] = \frac{\text{d\_cross\_high\_number}[i]}{\text{mac\_sent\_number}[i]} \tag{5}$$

where d_cross_high_number[i] is the number of packets whose delay is larger than the high delay threshold d_thr_high[i] in class i, and mac_sent_number[i] is the total number of packet successfully transmitted in class i. Similarly, we also record the rate of packets whose L_delay is smaller than a low delay threshold for the purpose of adaptation:

$$\text{d\_pr\_low}[i] = \frac{\text{d\_cross\_high\_number}[i]}{\text{mac\_sent\_number}[i]} \tag{6}$$

where d_cross_low_number[i] is the number of packets whose delay is smaller than the low delay threshold d_thr_low [i] in class i. similar to the drop rate calculation, we also use EWMA to define two estimators for d_pr_high[i] and d_pr_low[i]:

$$\begin{cases} \text{d\_pr\_high\_curr\_avg}[i] = \text{d\_pr\_high\_last\_avg}[i] \\ \qquad \times(1-\alpha) + \text{d\_pr\_high}[i] \times \alpha \\ \text{d\_pr\_high\_last\_avg}[i] = \text{d\_pr\_high\_curr\_avg}[i] \\ \text{d\_pr\_low\_curr\_avg}[i] = \text{d\_pr\_low\_last\_avg}[i] \\ \qquad \times(1-\alpha) + \text{d\_pr\_high}[i] \times \alpha \\ \text{d\_pr\_low\_last\_avg}[i] = \text{d\_pr\_low\_curr\_avg}[i] \end{cases} \tag{7}$$

Two thresholds, namely, d_pr_thr_high[i] and d_pr_thr_low[i] are defined for the above two estimators. It is worth noting that these two estimators are also updated for every F_{sw1} beacon periods.

3.2.3. Throughput

We sample the throughput over a slower time scale, for every F_{sw2} (similar as F_{sw1}, F_{sw2} defines another sampling window size) beacon periods, where $F_{sw2} > F_{sw1}$. Two kinds of throughput are sampled at the AP. One is the aggregate throughput (Agg_-throughput) of all the traffic classes, including downlink and uplink traffic. The other one is the throughput for best-effort traffic (BE_throughput), also including downlink and uplink traffic. Agg_throughput is input to the base_adaptation algorithm, and BE_throughput is used for AC purposes.

[‡‡]Here ACK is the acknowledgment for a successfully transmitted frame defined in IEEE 802.11 MAC layer [17], not the ACK packet defined for TCP protocol.

3.2.4. The validation of using LQI as the indicator to adaptations

As discussed in Reference [31], the validation of using LQI as the indicator to adaptations is based on the observation that, for each traffic class, the QoS experienced by the traffic sent/forwarded by the AP provides a lower bound for the QoS experienced by the traffic sent from terminal stations. This is because in BSS mode all the traffic has to go through the AP, which consequently has much higher load than other stations. More detailed analysis can be found in Reference [31].

3.3. Adapation Algorithms

3.3.1. Relative_adaptation

The relative_adaptation algorithm takes the result of LQIM (only estimations of drop rate and L_delay are used by this algorithm) as input and is executed by the AP for every F_{sw1} beacon periods. It adapts the allocation of wireless channel resources by adjusting three CAPs, namely, CWMin, CWMax and Arbitration Inter Frame Space (AIFS) [20]. Since the smaller the CAPs of one traffic class the larger the chance of this traffic class getting access to the shared media, the relative resource allocations for different traffic classes can be controlled by increasing or decreasing the relative difference of their CAPs. The basic principle of this algorithm is to correctly select the best differences between the CAPs of different classes, so that resources can be optimally allocated according to the QoS requirements of each traffic class. To guarantee the QoS for real-time applications, this algorithm needs to work together with our AC mechanism.

A skeleton of the relative_adaptation algorithm is shown in Figure 2. We briefly explain how this algorithm works. Every time the algorithm is executed, it checks the QoS of all real-time classes in order of increasing priorities.

1. If it finds that the QoS of one class (assume it is class i) is worse than its lower thresholds, the execution checks the QoS of all the classes (real-time only) with lower priority than class i in order of: if it finds one class (assume it is class j) has QoS better than its high thresholds, the CAPs of class j will be increased; if none of such classes can be found, the algorithm directly decreases the resources for best-effort traffic by increase its CAPs.

```
relative_adaptation() {
    for (int i = 1; i < plevels; i++) {          //plevels is the number of total traffic calsses,
                                                 //and class 1 has the highest priority.
        if (the QoS of class i indceated by LQI is worse than its low thresholds ) {
            for (int ii = plevels-1; ii > i; ii--) {
                if (the QoS of class ii indceated by LQI is better than its high thresholds) {
                    increase_parameters(ii);
                    return;
                }
            }
            increase_parameters(plevels);
            return;
        } else if (the QoS of class i indceated by LQI is better than its high thresholds) {
            for (int ii = i+1; ii < plevels; ii++) {
                if (the QoS of class ii indceated by LQI is worse than its low thresholds) {
                    decrease_parameters(ii);
                    return;
                }
            }
            decrease_parameters(plevels);
            return;
        }
    }
}
```

Fig. 2. Skeleton of the relative_adaptation algorithm.

2. If the QoS of one class (assume it is class k) is found better than its higher thresholds, it checks the QoS of all the classes (real-time only) with lower priority than class k in order of decreasing priorities: if one class (assume it is class l) is found having worse QoS than its lower thresholds, the CAPs of class l will be decreased; if none of such classes exists, the algorithm gives more resources to best-effort class by decreasing its CAPs.

As shown in Figure 2, we adjust the CAPs by calling the two functions: increase_parameters() and decrease_parameters(). Due to the space limitation, we cannot explain all the details. However, some basic rules to be followed by these two functions are listed below.

1. We use multiplicative increase multiplicative decrease (MIMD) for these two functions respectively. We compared the performance of MIMD with additive increase additive decrease (AIAD), and found MIMD works better than AIAD in most cases. The experience of selecting the scalers for increasing/decreasing is discussed lately in Section 4.2.
2. We use static priority strategy. When we adjust the CAPs, the CAPs for a class with higher priority class are always kept no larger than the CAPs for a class with lower priority, which means the CAPs of $(i-1)$th class and $(i+1)$th class are the bounds for the CAPs of ith class. Besides this, they should fall into the corresponding value range defined in 802.11e standard.
3. When we increase the CAPs, we always try to increase the parameters of CW (CWMin and CWMax) first; if they reach the bounds, we

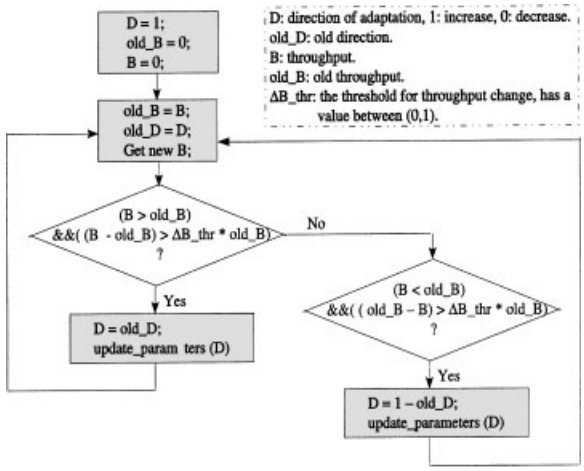

Fig. 3. Flow chart of the base_adaptation algorithm.

increase the AIFS. However, when we decrease the CAPs, we always try to decrease the AIFS first, if it hits the bound, we decrease the CWMin and CWMax.

3.3.2. *Base_adaptation*

We use a simple base_adaptation algorithm to adapt the CAPs to achieve quasi-optimal aggregate throughput. In this algorithm, we only adjust the values of CWMin and CWMax. It takes the aggregate throughput of LQI as input and is executed by the AP for every F_{sw2} beacon periods. The base_adaptation executes over a longer time scale than relative_adaptation, so that every time the CAPs are adjusted to achieve higher throughput, the relative_adaptation algorithm has enough time to adapt the values best for the QoS required also. The flow chart of this algorithm is shown in Figure 3. The base-adaptation algorithm increases or decreases the CAPs synchronously for all classes, depending on the last adaptation direction and its effect on achieved throughput. The update_parameters() function also utilizes MIMD to increase/decrease the CAPs, it increases (or decrease) the CAPs of all the traffic classes at the same time (this is different from relative_adaptation, in which only one class's CAPs are adjusted at one time) and the CAPs follow the same requirement on bounds, which is discussed in previous subsection. The base_-adaptation algorithm is self-corrected, whenever it makes a wrong decision (on the direction of adaptations), it will make corrections in the following round (after F_{sw2} beacon periods). One exception is, when the network has a light load, the varying traffic may mislead the adaptations, and cause the algorithm to take several rounds to find the optimal value. However, for the network with light load, throughput optimization is unimportant if not meaningless at all.

One important point is that, sometimes, we have to make a tradeoff between the goals of QoS guarantee and achieving optimal aggregate throughput. Considering the following example, there are just two traffic classes in a wireless cell, one is real-time, the other is BE; and the total load of these two kinds of traffic is larger than the capacity of the cell. We also assume that the real-time traffic has smaller packet size on an average than the BE traffic. From the view of QoS guarantee, the real-time traffic should have protected resource sharing. While from the view of maximum aggregate throughput, all the resources should be given to the BE traffic. In our work, we try to maximize the aggregate throughput only when the QoS is guaranteed.

3.4. Admission Control

For real-time applications, if expected limits on drop rate[§§] and delay bound cannot be guaranteed, the applications should be rejected since admitting real-time applications without QoS guarantee only leads to wasting of resources. While best-effort applications can always be admitted. Since the HARMONICA architecture is class-based, the QoS bounds for each real-time traffic class can be pre-defined in PCU by setting various thresholds. Whenever a new real-time application requires admission, the AdC will select a traffic class that best matches its QoS requirements (in terms of drop rate limits and delay bounds), then an AC process will executes taking its input from LQIM and the throughput requirement (req_throughput) of the flow.

The AC process relies on AC policies, which is decided by the QoS goal we want to achieve. Besides the QoS guarantee for real-time applications, we may want to guarantee a minimal bandwidth (BE_Min) for BE traffic class. To achieve this goal, the following two rules should not be violated after admitting a new real-time flow (assume it is classified in class *i*):

[§§]The throughput requirement of real-time applications generally depends on what kind of codec they use. Instead of using throughput as a QoS metric for real-time applications, we believe using drop rate is more practical and flexible. However, the throughput requirement should still be included in admission request so that admission decision can be made.

(1) the QoS of each real-time traffic class should be guaranteed;
(2) BE_throughput $\geq$ BE_Min.

To accurately make an AC decision, we need to ensure that the following three requirements are satisfied:

(A) the relative_adaptation has reached a stable state;
(B) BE_throughput$-$Req_throughput $\geq$ BE_Min;
(C) the bandwidth of Req_throughput in BE class can be 'translated' into class i without loss.

Among these three requirements, (A) to make sure that at current time the QoS of all the real-time classes are well supported, so the amount of bandwidth expressed by BE_throughput$-$BE_Min is what we can really take advantage of; (B) and (C) to ensure that after this flow is admitted, we can still guarantee the minimal amount throughput for BE traffic, while sustaining the QoS of all real-time classes. However, satisfying (A) introduces delay to the admission process. (C) is hard to be guaranteed, although class i has a higher priority than BE class, the throughput also depends on the average frame length.

To balance the complexity and accuracy, we take an alternative approach by adding a margin for Req_throughput. The AC process of HARMONICA simply checks the following inequality:

$$\text{BE\_throughput} - \text{Req\_throughput} \times F_{\text{intra-cell}} \times F_{\text{margin}} \geq \text{BE\_Min} \quad (8)$$

where, F_{margin} is the factor of margin, $F_{\text{intra-cell}}$ is 2 for intra-cell flows,*** and 1 for uplink or downlink flows. The value of F_{margin} can be roughly estimated using the following formula:

$$F_{\text{margin}} = \frac{\text{L\_BE}}{\text{L\_Req}} \quad (9)$$

where, L_BE is the average frame length for best-effort traffic, while L_Req is the average frame length for the requested traffic flow. L_BE can be sampled in LQIM, and L_Req can be sent to the AP with admission requests. It is worth noting that estimating the mean frame length is much easier than estimating the average collision length discussed previously in Section 2. If one real-time flow is wrongly admitted (the AP find this when the BE_throughput falls below the BE_Min), the AP can choose to drop the last admitted flow by explicitly informing the corresponding node through admission response messages.

***One intra-cell flow in fact contains one uplink flow and one downlink flow.

4. Performance Evaluations and Discussions

To evaluate the performance of our proposed QoS approach, we have implemented HARMONICA using Berkeley NS V2.1b7 simulator [26]. We compare the performance of HARMONICA in terms of QoS guarantee and aggregate throughput to that of EDCA using fixed CAPs. We select different CAPs for EDCA in three cases: (1) the first case follows the default parameter set given in IEEE 802.11e draft v5.0 [20]; (2) the second case is an extreme case, in which the parameters for each class differ significantly, aiming at very strict service differentiations; (3) and the third case explores the parameters, by which we try to provide quite loose service differentiations. For simplicity, we call these three cases EDCA-default, EDCA-strict and EDCA-loose respectively.

4.1. Simulation Scenario and Traffic Models

We have conducted simulations with various scenarios. Due to the space limitations, we report the results of one scenario in this paper. The scenario contains 100 terminal stations and one AP. All of them operate on IEEE 802.11a PHY mode-6 [18] sharing a clear (interference free) wireless channel. The related PHY/MAC parameters (excluding CAPs) are listed in Table I. We construct three types of traffic flows

Table I. IEEE 802.11a PHY/MAC parameters used in simulations.

Parameters	Values
Data rate	36 Mbits/s
Modulation	16-QAM
MAC header	28 bytes
Preamble length	16 μs
PLCP header length	4 μs
Slot time	9 μs
Short inter-frame space	16 μs
Clear channel assurement (CCA) time	3 μs
RxTx turnaround time	1 μs
RTS threshold	3000 bytes
Fragmentation threshold	3000 bytes
Long retry limit	4
Short retry limit	7
Beacon period	100 ms

Table II. Flow characteristics and channel access parameters (CAPs) for the three traffic classes.

Traffic class	Characteristics	CAPs (EDCF-default/EDCF-strict/EDCF-loose)		
		CWMin	CWMax	AIFS (μs)
1	ITU-T G.729A VoIP stream, Packet size = 60 bytes, Packet interval = 20 ms, Bit rate = 24 kbits/s	31/7/15	63/31/63	43/43/43
2	H.263 Video trace, VBR traffic with target bit rate of 256 kbits/s	63/63/23	127/127/127	43/124/43
3	Exponential on/off traffic, Packet size = 368 bytes, Average burst time = 500 ms, Average idle time = 500 ms, Burst rate = 200 kbits/s	127/127/31	1023/1023/1023	52/484/52

between the AP and terminal stations, namely, VoIP streams, video streams, and BE traffic flows. We model the VoIP traffic source using ITU-T G.729A codec [16], and assume that the audio stream is transmitted over RTP/UDP/IP. This model generates 60-byte messages (40 Bytes overhead introduced by RTP/UDP/IP) periodically with an interval of 20 ms yielding a bit rate of 24 kbits/s. For the video traffic source, we use a trace of real H.263 video stream captured in an office environment. This trace produces a variable bit rate (VBR) traffic with target bit rate as 256 kbits/s. More information on this video trace can be found in Reference [14]. We model the BE traffic source using an exponential on/off model, which generates packets at a constant burst rate (200 kbits/s) during 'on' period, and no packets during 'off' period; burst times and idle times are taken from exponential distributions, both of them have the average of 500 ms. The packet size of BE traffic is set to 368 bytes, which is the mean packet length of real LAN traces [15]. The characteristics of these three types of traffic are summarized in Table II.

The distribution of the traffic flows is shown in Figure 4. There is one pair (one downlink, one uplink) of BE flows between each terminal station (node 1 ∼node 100) and the AP, starting from simulation time 0.0 s and stopping at 80.0 s. VoIP and video flows are created dynamically. Starting from 10.0 s, for every 1.0 s, there are one pair VoIP flows and one pair of video flows launched. These real-time flows stop at

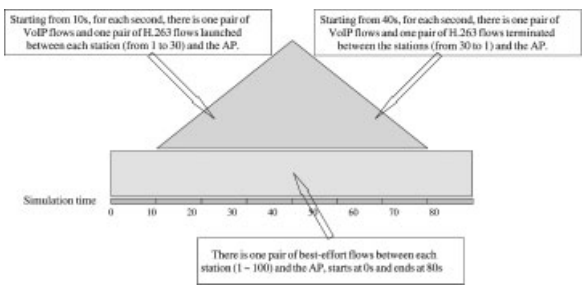

Fig. 4. Dynamic traffic load of the simulation scenario.

(80.0 s—their launching time). There are 30 pairs of VoIP flows and the same number of video flows are launched between nodes 1 ∼30 and the AP. The CAPs for the three cases of EDCA testing are listed in Table II. We use the CAPs of EDCA-default as the initial values to evaluate the HARMONICA architecture.†††

The main performance parameters for voice and video traffic are well-known to be packet transfer latency and packet loss. In order to preserve the end-user-perceived quality, the target end-to-end QoS for voice/video applications can be set as follows: (1) for voice streams, the maximum one-way latency is 150 ms and the maximum packet loss is 5%; (2) for video streams, 400 ms as the maximum one-way latency and 5% as the maximum packet loss rate [9,11]. Since the WLANs represent only the last hop of an end-to-end connection, we have chosen to select more stringent requirements. In our experiments, we use $\Pr\{\text{delay} > 30\,\text{ms}\} \leq 0.05$ and $\Pr\{\text{delay} > 80\,\text{ms}\} \leq 0.05$ to bound the one-way latency for voice and video respectively. While for BE traffic no limits on drop rate and latency are defined. However, in this case, we try to guarantee a minimal aggregate bandwidth for BE traffic. We set this minimal bound as 1 Mbits/s. We introduce a new concept called useful throughput to demonstrate the QoS experienced at each flow. The difference between useful throughput and throughput is that useful throughput only counts the packets whose latency is less than some limit. We denote useful throughput as BW_{limit} in this context. As discussed previously, we investigate $\text{BW}_{30\text{ms}}$, $\text{BW}_{80\text{ms}}$, and $\text{BW}_{+\infty}$ for class 1, 2, 3 respectively.

4.2. Simulation Results and Discussions

Since flows in same class and direction experience similar QoS and the downlink traffic has a lower QoS

†††This is not a necessary choice, we can also use other initial values, since the HARMONICA is auto-adaptive.

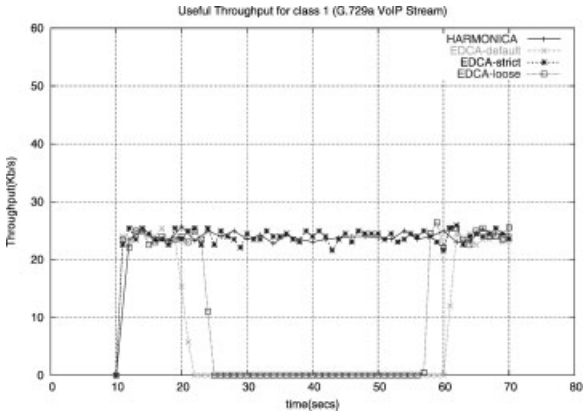

Fig. 5. Useful throughput of the flows in class 1.

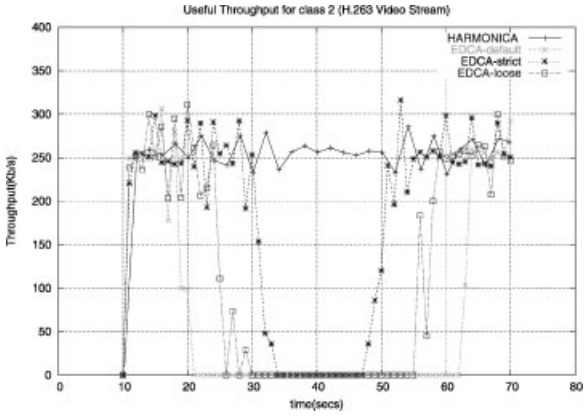

Fig. 6. Useful throughput of the flows in class 2.

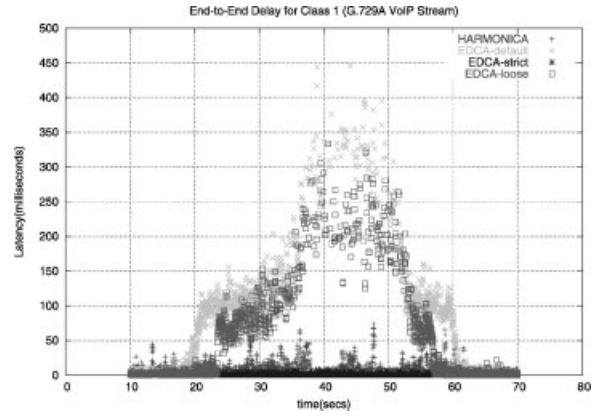

Fig. 7. End-to-end delay for class 1.

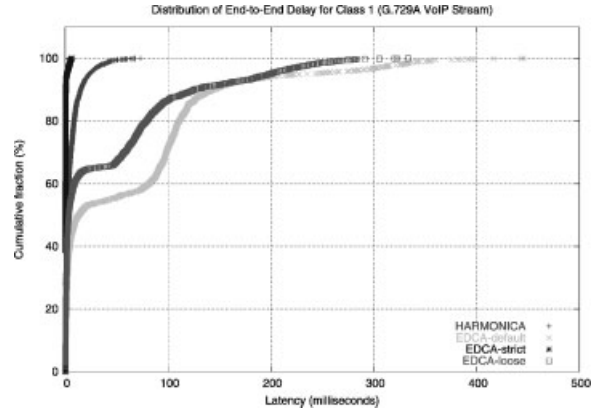

Fig. 8. Delay distributions for class 1.

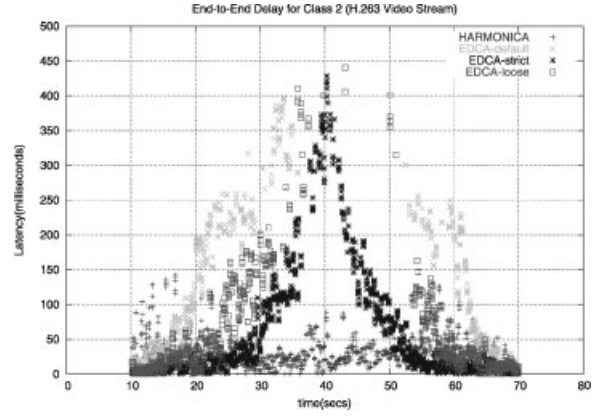

Fig. 9. End-to-end delay for class 2.

bound for each class, instead of presenting the QoS of all the flows, we only report the QoS of one downlink flow for each class. In this case, we select the flows from the AP to node 1, since these flows cover the longest running period among all other flows. The useful throughputs of real-time classes are demonstrated in Figures 5 and 6 respectively. The end-to-end delay and distributions of delay of class 1 (G.729A VoIP streams) are shown in Figures 7 and 8, while Figures 9 and 10 give the delay information for class 2 (H.263 Video streams). We also give the aggregate useful throughputs for the BE class in Figure 11. We present the aggregate useful throughputs of the whole network cell in Figure 12. To compare the performance of HARMONICA to that of EDCA, we give the results for HARMONICA and three EDCA cases (EDCA-default, EDCA-strict, EDCA-loose) in each Figure.

Let us briefly explain the results for the HARMONICA and the three cases of EDCA. From Figures 5 to 10, we can see that only HARMONICA can guarantee the QoS of all the VoIP and video streams, while all the three cases of EDCA failed to do this. The EDCA-strict case guarantees well the QoS of VoIP flows but performs poorly to bound the QoS for video streams

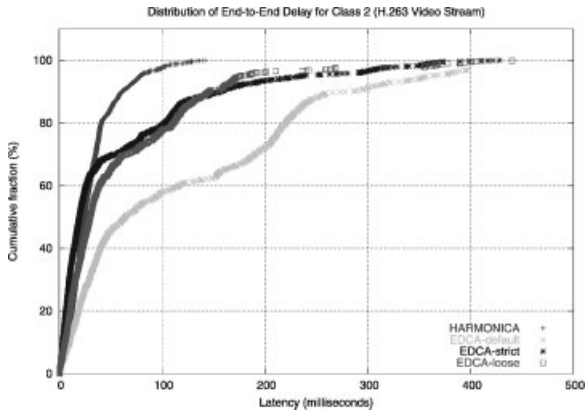

Fig. 10. Delay distributions for class 2.

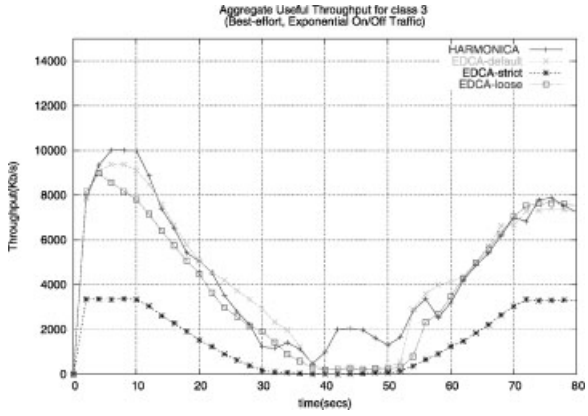

Fig. 11. Aggregate useful throughput of the flows in class 3.

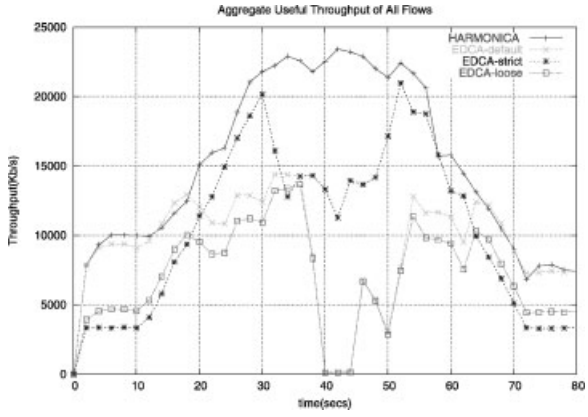

Fig. 12. Aggregate useful throughput of the whole cell.

and gets much lower throughput for BE traffic than all other cases. The EDCA-default and EDCA-loose do not guarantee the QoS for both and VoIP and video streams. The goal of QoS guarantee is achieved in HARMONICA by two mechanisms. During 10.0 s–30.0 s and 50.0 s–70.0 s, the HARMONICA mainly relies on the relative_adaptation algorithm to satisfy the QoS requirements of real-time applications. The three EDCA cases cannot achieve this because they use fixed CAPs. During 30.0 s–50.0 s, the admission control mechanism of HARMONICA helps to avoid congestion by rejecting incoming real-time flows. From our results, we find that HARMONICA starts to deny incoming video streams from 30.0 s and VoIP flows from 31.0 s. And at about 37.0 s, HARMONICA dropped the last admitted (at 31.0 s) VoIP flow since the BE_throughput has fallen down below 1 Mbits/s (as shown in Figure 11). Figure 12 clearly shows that HARMONICA, which benefits from the relative_adaptation as well as the base_adaptation, achieves higher aggregate throughput than all other cases.

We have also conducted extensive simulations to investigate the effects of the various values of several factors on performance. Such factors includes α, F_{sw1}, F_{sw2} and the adaptation scalers for both adaptation algorithms. Due to the space limitation, we cannot present all the results in this work. Generally, selecting different values for these factors is to make the tradeoffs between agility and stability. For α, the values between 0.5 and 0.8 are better than others. A good range for F_{sw1} is 1–5, and that for F_{sw2} is $2 \times F_{sw1} - 5 \times F_{sw1}$. We have used 0.5, 1 and 5 for each of these factors in the reported simulations. In terms of the scalers for relative_adaptation and base_adaptation, we found that the values between 1.2 and 1.5 do not significantly affect performance. The values in this range are better than other values for the final results. We have used 0.02 as the value for ΔB_thr (used in base_adaptation) in the simulations reported above. A good range for this threshold is found to be 0.01–0.03.

The computation overhead of the HARMONICA is quite low. The relative_adaptation algorithm has the highest time complexity among all computations involved in HARMONICA. Its complexity is $\bigcirc(n^2)$ in terms of the number of classes, and $\bigcirc(1)$ in terms of the number of flows. Since the HARMONICA supports up to four classes, the time complexity is bounded. The broadcasting of CAPs does not introduce extra communication overhead into IEEE 802.11e-based products since the 802.11e beacons already contain a defined QoS parameter set, which includes CAPs for each traffic category. HARMONICA can be implemented in IEEE 802.11e-based products by upgrading the driver/firmware of the NICs.

5. Conclusions

In this paper, we proposed a novel architecture called HARMONICA to achieve QoS guarantee and quasi-optimal channel utilization in an IEEE 802.11e-based WLAN. It achieves its goals by utilizing two adaptation algorithms and one simple admission control scheme. The simulations conducted using NS-2 simulator demonstrate that, under an interference-free environment, HARMONICA can well satisfy the QoS requirements for different traffic classes while simultaneously achieving higher aggregate throughput than EDCA with fixed CAPs. One advantage of this architecture is that it provides the ability to control the resource allocation from the AP. Although HARMONICA is currently designed for an interference free environment, it can also be extended to enable deployment in noisy environments by enhancing some mechanisms to adapt to interferences.

Acknowledgements

We are grateful to the anonymous reviewers for their constructive suggestions, which helped to improve the quality of this paper. We thank the guest editor Sunghyun Choi for his time, encouragements and support throughout the revision process. We would also like to thank Xiaoning He of NTT-DoCoMO laboratory (U.S.A.), who provided us assistance with the 802.11e drafts. Liqiang Zhang was supported by a Wayne State University Thomas C. Rumble Fellowship. This work has also been supported by grants from Sun Microsystems (Palo Alto), Microsoft Corporation (Seattle) and Ixia Corporation (Calabasas). We are also grateful to Farshad Fotouhi and Monica Brockmeyer for making the laptops available to us to conduct our simulation experiments.

Appendix: The Relation Between L_delay and A_delay

We denote the application layer end-to-end delay as A_delay, it can be expressed as following formula:

$$\begin{aligned} \text{A\_delay} = {} & T_{(\text{app}-\text{llc})@\text{sender}} + T_{(\text{llc}-\text{mac})@\text{sender}} \\ & + T_{\text{mac}@\text{sender}-\text{mac}@\text{receiver}} \\ & + T_{(\text{mac}-\text{app})@\text{receiver}} \end{aligned} \tag{10}$$

where, $T_{(\text{app}-\text{llc})@\text{sender}}$ is the delay when a packet travels from the application layer to the LLC layer at the sender, $T_{(\text{llc}-\text{mac})@\text{sender}}$ is the buffering delay at the sender's IFQ, $T_{\text{mac}@\text{sender}-\text{mac}@\text{receiver}}$ is the delay for transmitting the packet from the sender to the receiver (including deferring, backoff and propagation delay, etc.) and finally $T_{(\text{mac}-\text{app})@\text{receiver}}$ is the delay from the MAC layer to application layer at the receiver. The L_delay (link layer end-to-end delay) can be expressed by the following formula:

$$\begin{aligned} \text{L\_delay} = {} & T_{(\text{llc}-\text{mac})@\text{sender}} + T_{\text{mac}@\text{sender}-\text{mac}@\text{receiver}} \\ & + \text{SIFS} + T_{\text{ack}-\text{propgation}@\text{phy}} \end{aligned} \tag{11}$$

where, short inter frame space (SIFS) is standard-defined duration between the receiver as it gets a frame and sends back the ACK, and $T_{\text{ack}-\text{propgation}@\text{phy}}$ is the physical layer propagation time for an ACK frame. Using the above two equations, the difference of A-delay and L-delay is calculated as:

$$\begin{aligned} \text{A\_delay} - \text{L\_delay} = {} & T_{(\text{app}-\text{llc})@\text{sender}} \\ & + T_{(\text{mac}-\text{app})@\text{receiver}} \\ & - \text{SIFS} - T_{\text{ack}-\text{propgation}@\text{phy}} \end{aligned} \tag{12}$$

For current WLANs (with bandwidthmuch less than 1 Gbits/s), the wireless link is still the bottleneck for the end-to-end communications [30]. The values of $T_{(\text{app}-\text{llc})@\text{sender}}$ and $T_{(\text{mac}-\text{app})@\text{receiver}}$ are very small (less than several ms[‡‡‡]) and are almost constant for each packet transmission in current terminal stations (PC, laptop or even PDA). SIFS and $T_{\text{ack}-\text{propgation}@\text{phy}}$ are small constants (i.e. for IEEE 802.11a [18] WLAN, the two values are respectively 9 μs and less than 40 μs). So the value of A_delay − L_delay can be seen as a small variable.

References

1. Aad I, Castelluccia C. Differentiation mechanisms for IEEE 802.11. In *Proceedings of IEEE INFOCOM 2001*, Alaska, USA, April 2001.
2. Bianchi G. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications* 2000; **18**(3): 535–547.
3. Bianchi G, Capone A, Petrioli C. Throughput analysis of end-to-end measurement-based admission control in IP. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
4. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An Architecture for Differentiated Services. RFC 2475, Dec. 1998.

[‡‡‡]We use ms to denote milliseconds, and μs to denote microseconds.

5. Bononi L, Conti M, Gregori E. Run-time optimization of IEEE 802.11 wireless LANs performance. *IEEE Transactions on Parallel and Distributed Systems* 2004; **15**(1): 66–80.
6. Breslau L, Knightly E, Shenker S, Stoica I, Zhang H. Endpoint admission control: architecture issue and performance. In *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.
7. Cali F, Conti M, Gregori E. IEEE 802.11: design and performance evaluation of an adaptive backoff mechanism. *IEEE Journal on Selected Areas in Communications* 2000; **18**(9): 1774–1786.
8. Cali F, Conti M, Gregori E. Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Transactions on Networking* 2000; **8**(6): 785–799.
9. Chen T, Gerla M, Kazantzidis M, Romanenko Y, Slain I. Experiments on QoS adaptation for improving end user speech perception over multi-hop wireless networks. In *Proceedings of ICC'99*, Vancouver, Canada, June 1999.
10. Choi S, Prado JD, Shankar SN, Mangold S. IEEE 802.11e contention-based channel access (EDCF) performance evaluation. In *Proceedings of IEEE ICC'03*, Anchorage, Alaska, USA, May 2003.
11. Coverdale P. ITU-T Study Group 12: multimedia QoS requirements from a user perspective. Workshop on QoS and user perceived transmission quality in evolving networks, Dakar, Senegal, October 2001.
12. Deng J, Chang RS. A priority scheme for IEEE 802.11 DCF access method. *IEICE Transactions on Communications* 1999; **E82-B**(1): 96–102.
13. Elek V, Karlsson G, Ronngren R. Admission control based on end-to-end measurements. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
14. Fitzek F, Reisslein M. MPEG-4 and H.263 video traces for network performance evaluation. *IEEE Network* 2001; **15**(6): 40–53.
15. Grilo A, Nunes M. Performance evaluation of IEEE 802.11e. In *Proceedings of IEEE PIMRC'02*, Lisboa, Portugal, September 2002.
16. ITU-T. G.729 Annex A: reduced complexity 8 kbits/s CS_A-CELP speech codec. *ITU-T Recommendation G.729 Annex A*, November 1996.
17. IEEE 802.11 WG. Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11*, 1999.
18. IEEE 802.11 WG. 802.11a, Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications, September 1999.
19. IEEE 802.11 WG. Draft Supplement to Part 11: wireless medium access control (MAC) and physical layer (PHY) specifications: MAC enhancements for quality of service (QoS). *IEEE 802.11e/Draft 4.0*, February 2003.
20. IEEE 802.11 WG. Draft supplement to part 11: wireless medium access control (MAC) and physical layer (PHY) specifications: MAC enhancements for quality of service (QoS). *IEEE 802.11e/Draft 5.0*, July 2003.
21. Kim M, Noble B. Mobile network estimation. In *Proceedings of ACM SIGMOBILE 2001*, Rome, Italy, July 2001.
22. Lindgren A, Almquist A, Schel O. Quality of service schemes for IEEE 802.11 wireless LANs—an evaluation. *Journal of Mobile Networking and Applications (MONET)* 2003; **8**(3): 223–235.
23. Mangold S, Choi S, Hiertz GR, Klein O, Walke B. Analysis of IEEE 802.11e for QoS support in wireless LANs. *IEEE Wireless Communications Magazine* 2003; **10**(6): 40–50.
24. Ni Q, Romdhani L, Turletti T, Aad I. QoS issues and enhancements for IEEE 802.11 Wireless LAN. *INRIA Research Report*, No. 4612, November 2002.
25. Nichols K, Blake S, Baker F, Black D. Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 headers. *IETF RFC 2474*, December 1998.
26. The Network Simulator—ns-2, http://www.isi.edu/nsnam/ns.
27. Tseng C, Lee G, Liu R, Wang T. HMRSVP: a hierarchical mobile RSVP protocol. *Wireless Networks* 2003; **9**(2): 95–102.
28. Veres A, Campbell AT, Barry M, Sun L. Supporting service differentiation in wireless packet networks using distributed control. *IEEE Journal on Selected Areas in Communications* 2001; **19**(10): 2081–2093.
29. Wu H, Peng Y, Long K, Cheng S, Ma J. Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement. In *Proceedings of IEEE INFOCOM 2002*, New York, USA, June 2002.
30. Zeadally S, Zhang L. Enabling gigabit network access to end users. In *Proceedings of the IEEE, Special Issue in Gigabit Wireless Design Challenges* 2004; **92**(2): 340–353.
31. Zhang L, Zeadally S. A novel approach to enhance QoS support over wireless LANs. In *Proceedings of WPMC'03*, Kanagawa, Japan, October 2003.
32. Zhang L, Zeadally S. Extending RSVP to IEEE 802.11 wireless local area networks. In *Proceedings of WPMC'03*, Kanagawa, Japan, October 2003.

Authors' Biographies

Liqiang Zhang received his B.S. degree in computer science from the China University of Mining and Technology in 1993 and his M.S. degree in electrical engineering from the China Academy of Telecommunication Technology (CATT) in 1996. He is currently working toward the Ph.D. in the Department of Computer Science at Wayne State University, Detroit, MI, U.S.A. From 1996 to 2000, he participated in several research and development projects on data communications at CATT. He has published several refered papers on computer networks. His research interests include quality of service over wired and wireless networks, wireless sensor networks and embedded systems. He was the winner of the 2003 Thomas C. Rumble University Graduate Fellowship.

Sherali Zeadally received his B.A. and M.A. degrees in computer science from University of Cambridge, Cambridge, U.K., and the Ph.D. in computer science from University of Buckingham, Buckingham, U.K., in 1996. In March 1996, he joined the Department of Electrical Engineering at the University of Southern California (USC), Los Angeles, where he was an assistant professor until August 1999. While at USC, he was also one of the principal investigators of the Integrated Media Systems Center (IMSC), an NSF Engineering Research Center, where he led the networking research group from 1996 to 1999. He joined the Department of Computer Science at Wayne State University, Detroit, MI, as an assistant professor in August 1999. He is the founder and director of the

High-Speed Networking Laboratory and the Computer Networking Research Group at Wayne State University. His research in computer networking has been funded by NSF and several companies, including HP, Microsoft, Compaq, Altheon, IBM, Sun Microsystems and others. He currently serves on the editorial boards of several international journals. His research interests include high-speed networks, quality of service, operating systems, wireless and personal area networks, middleware and performance analysis of network systems. He is the recipient of two Outstanding Teaching Awards (in 2001 and 2003) from the College of Science at Wayne State University, the highest award given by the College of Science for excellence in teaching. He is a fellow of the British Computer Society.