Cross Layer Design for IEEE 802.11 WLANs: Joint Rate Control and Packet Scheduling

Qiuyan Xia, Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Kowloon, Hong Kong, China
{xiaqy, hamdi}@cs.ust.hk

Abstract

IEEE 802.11 Wireless Local Area Networks (WLANs) are widely deployed nowadays. With traditional layered architecture, current WLAN adopts functional layer partitioning and aims at optimization in individual layers. However, in a highly dynamic and media sharing wireless environment, the capacity enhancement at physical layers may not necessarily benefit the overall system performance. Moreover, in a multiuser setting, throughput can be increased substantially if partial knowledge of the channel is known. In this paper, we address the issue of cross layer design in the proposed "Weighted Fair Scheduling based on Adaptive Rate Control" (WFS-ARC) framework, where the PHY layer knowledge is shared with the MAC/LLC layer. It can make good matching of instantaneous channel conditions of multiple users with resource allocation to each user.

1. Introduction

The IEEE 802.11 standard [5] has emerged as a prevailing standard in the market for WLANs. The current PHY layer extensions provide multiple data rates: for example, 802.11a [6] defines 8 different data rates ranging from 6 up to 54 Mbps. Typically, higher data rates require higher SNR to maintain a certain BER; on the other hand, lower data rates can ensure a small BER, but the achieved throughput is also small. In wireless systems, the propagation environments vary over time and space due to such factors as signal attenuation and fading, motion of objects, interference and so on, causing variations in the received SNR. As a result, no data rate can be optimal under all scenarios. Moreover, the standard does not specify when and how to switch among different rates. As the multirate enhancements are PHY layer protocols, MAC layer rate control mechanism is required to exploit this capability, by tuning the user-available MAC layer parameter, the data rate, to current channel condition. However, the transmitter only has limited, indirect feedback such as ACKs and retry counts under the current 802.11 implementation. Therefore, a rate control algorithm is required to be adaptive and simple enough.

While the rate control scheme at the MAC layer aims to tune the data rate to the channel conditions and maximize the throughput of an individual layer, the overall system performance may not necessarily be optimized. Moreover, in a multiuser setting, different users undergo different channel gains, which results in the phenomenon of multiuser diversity. However, in the common infrastructure WLANs, the multiuser diversity is rarely exploited and sometimes even degrades the system performance. For example, FIFO queueing at the Access Point (AP) may give rise to the Head-of-Line Blocking problem, as the HOL packet transmitted through a low quality radio channel consumes lots of air time and prevents other transmissions occurring over good channels, which results in low channel utilization. Taking account of these effects, a transmission scheduler at the LLC layer is required to exploit the multiuser diversity, by opportunistically selecting a feasible user with a good channel. Also, it should balance the system throughput and fairness requirements among "good" and "bad" users.

Applying the above ideas to our proposed cross layer design, the WFS-ARC approach dynamically adjusts the data rate parameter at the MAC layer, based on channel state feedback provided by the PHY layer. On top of the MAC layer, an LLC layer scheduler is implemented to opportunistically schedule the packet transmission to the most promising user and also satisfy the fairness constraint. We remark that while exploring a much richer interaction between parameters across layers, we should not destroy the necessary independence and integrity of individual layers, which may lead to various undesirable consequences.

The rest of the paper is organized as follows. We briefly introduce the IEEE 802.11 standard and related work in section 2. In section 3, we first give an overview of our cross layer design framework and explain how it functions. Based

Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)
0-7695-2421-4/05 $20.00 © 2005 IEEE

on that, we present our adaptive rate control algorithm and the optimization model for weighted fairness scheduling in detail. Section 4 evaluates the simulation performance of our approach. Finally, this paper concludes with section 5.

2. Background and related work

2.1. The IEEE 802.11 WLAN

The IEEE 802.11 [5] standard specifies the Medium Access Control (MAC) and Physical (PHY) layers for a WLAN system. There are currently three PHY layer extensions: 802.11b, 802.11a, and 802.11g, all providing multiple data rates. The common MAC layer defines rules for orderly access to the shared medium in support of the Logical Link Control (LLC) layer. Two medium access mechanisms are defined in 802.11: the Distributed Coordination Function (DCF) is a mandatory, contention-based protocol; the Point Coordination Function (PCF) is a priority-based, contention-free protocol. The DCF implementation is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism. The basic access mode works as follows: before a station starts a frame transmission, it checks the medium status by carrier sensing. If the medium is idle, the transmission may proceed; if the medium is sensed busy, the station defers its transmission until the medium is determined to be idle for the DIFS interval and a random backoff procedure is invoked. For each successful reception of a frame, the receiver immediately acknowledges the frame reception by sending an ACK frame. The DCF also defines an optional RTS/CTS mechanism. Throughout this paper, we present the cross layer design for 802.11a WLAN working under the basic access mode. Our approach can easily be extended to WLANs with other PHY layers such as 802.11g. Also, we focus on infrastructure networks that use Access Points (APs) for all communications between stations.

2.2. Related work

2.2.1. Rate control algorithm. The Auto Rate Fallback (ARF) algorithm [16] in Lucent's WaveLAN-II devices, is one of the few used in commercial WLAN products. It works as follows: if 2 consecutive ACKs are not received correctly, the sender drops the transmission rate for the second retry of the current packet as well as the subsequent transmissions to the next lower data rate, and starts a timer; if 10 consecutive ACKs are received successfully, the transmission rate is raised to the next higher data rate and the timer is cancelled. Obviously, this scheme is easy to deploy with existing 802.11 devices since it requires no changes to the standard. In the Receiver Based Auto Rate (RBAR) protocol [4], the receiver estimates the channel conditions using a sample of instantaneously received signal strength at the end of RTS reception, and then feeds back the selected data rate to the sender in CTS. Similar ideas can be found in [15]. However, standard modification is required. In all schemes, only time diversity of single channel is considered. By tuning data rate to the time-varying channel conditions, these methods mitigate channel variations rather than utilize them. The multiuser diversity is completely ignored.

2.2.2. Packet scheduling algorithm. In an infrastructure WLAN, usually the AP maintains a single FIFO queue for all traffic flows. Since the HOL packet may be either transmitted at a low rate, or retransmitted many times before it gets through or is finally dropped, it prevents other packets in the FIFO queue from being transmitted, leading to very low channel utilization. Note that the instantaneous channel gains from the AP to different clients are independent; however, due to the HOL blocking, variations in channel quality can not be utilized.

The Channel State Dependent Packet Scheduling (CSDPS) scheme proposed in [3] addresses this problem. The main idea is that, when a wireless channel is observed at the "bad" status, it switches to transmit on other "good" links, so that the HOL effect can be removed. However, this method has some limitations. First, it uses a link status monitor to continuously track the channel quality. Such a mechanism is not available in the current 802.11 WLANs due to implementation costliness. Second, the binary channel model assumed is too simple to account for the realistic time-varying wireless environment.

There are other scheduling schemes proposed; however, some are designed specifically for HDR or CDMA systems [10]; others do not take advantage of a rate adaptive MAC [7]. Again, few explicit models or solutions are proposed for jointly using the information from different layers. Note that in the infrastructure WLANs, the overall system performance (e.g., the aggregate throughput) is more desirable than single user performance. On the other hand, as described in [13], there also exists a fundamental tradeoff between throughput and fairness. Therefore, the fairness constraint should be satisfied while trying to improve the system performance.

3. WFS-ARC framework

3.1. Cross layer design

In wireless networks, where channel quality can vary dramatically in both time and frequency, knowledge of the channel state can be exploited to significantly improve performance [2]. For example, modern wireless interface cards usually employ rate adaptation methods. Moreover, higher error rates or lower data rates at the PHY layer affects higher-layer performance. In a multiuser setting, as

the number of users in a system increases, the probability that one user has a very good channel also increases. Exploiting this diversity results in a total system throughput gain that increases with the number of users, since it provides a user with an opportunity to transmit data to one of its neighbors with good channel quality before those with bad channel quality. However, this must be balanced with higher-layer issues (e.g., fairness). All of these coupling effects demonstrate the need to consider higher-layer issues jointly with the PHY layer issues. Note that cross layer design does not mean getting rid of protocol layers, or integrating all layers [8]. Instead, since there is direct coupling between the PHY layer and upper layers, the inter-layer coupling can be exploited for further optimization, instead of a sub-optimal solution resulting from several local optimizations in each layer. We must consider carefully which layers should respond to channel variations, and what layers should be jointly designed or optimized [1].

Our solution serves to provide some insights regarding the design of cross layer paradigms. We are trying to demonstrate that significant performance gain can be achieved by jointly considering several layer issues in an integrated framework. Our goal is to utilize the multirate PHY to produce a rate adaptive MAC, and exploit the multiuser diversity to produce a MAC assuming a multiuser PHY, so that the MAC can take advantage of both time and location dependent diversities to address the channel vulnerability and the HOL blocking, while considering fairness issues among multiple users.

We focus primarily on the downlink model, a situation where the arriving traffic is buffered at the AP until it is transmitted, and resources are allocated as a function of each flow's channel state. The AP acts as a centralized controller and makes all resource allocation decisions. It maintains a set of queues for each back logged flows at the LLC layer (Figure 1). We assume saturated traffic here, so packets arriving when queues are full are dropped. In our WFS-ARC scheme, LLC/MAC/PHY layers work cooperatively to optimize bandwidth allocation, while satisfying the fairness requirement. We note that the proposed framework is quite general and can accommodate a variety of physical layer models and wireless network topologies.

There are three functional blocks in the proposed cross layer framework (Figure 1). The rate controller at the MAC layer adjusts the data rate parameter used by the PHY layer. Its goal is to dynamically adjust the data rate according to the channel quality variations and therefore, improve the channel efficiency. It collects statistics of ACKs and retry counts, for different downlink nodes. Based on the collected information, the AP estimates the downlink channel quality and selects the data rate which is "best" among all possible rates for each active client.

Suppose there are N users in total in the system, each

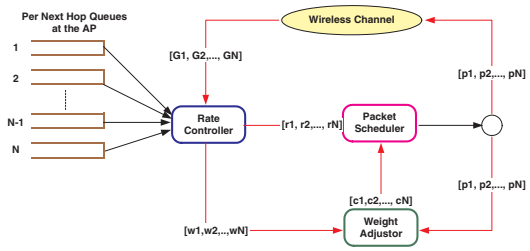

Figure 1. WFS-ARC framework

with a weight w_i. The weight vector $\vec{w}$ can be constant or dynamically updated. We define the Contention Period (CP), as the time duration lasting for a frame's transmission and reception, including backoff, deferring, and ACK duration. The scheduling decision is made at the beginning of each CP. The rate controller collects the transmission statistics, such as ACKs and retry counts to measure the channel qualities. The output of the rate controller is a vector of data rates, $\vec{r}(k) = [r_1(k), r_2(k), \ldots, r_N(k)]$, for active nodes at the contention period k. Based on the rate vector, the expected goodput vector during k_{th} CP $\vec{G}(k) = [G_1(k), G_2(k), \ldots, G_N(k)]$ can be computed approximately using the formulas (1), (2) and (3):

$$G = \frac{Data\,payload\,length}{ETT}, \quad where \tag{1}$$

$$\begin{aligned} ETT = \quad & Backoff + DIFS + PHY\,ov + MAC\,ov + \\ & MAC\,data\,duration + SIFS + ACK, \end{aligned} \tag{2}$$

$$MAC\,data\,duration = \frac{MAC\,data\,payload\,length}{data\,rate \quad r} \tag{3}$$

The goodput vector $\vec{G}(k)$, together with a control vector $\vec{c}(k) = [c_1(k), c_2(k), \ldots, c_N(k)]$ are fed to the scheduling block, which is positioned on top of the MAC layer. The scheduler selects the most promising user to transmit a packet to, based on a scheduling policy. The scheduling decision is given by the vector $\vec{p}(k) = [p_1(k), p_2(k), \ldots, p_N(k)]$. Define $p_i(k)$ $(1 \leq i \leq N)$ as follows (i.e., one by one scheduling):

$$p_i(k) = \begin{cases} 1 & user_i\,is\,scheduled\,in\,k_{th}\,CP, \\ 0 & otherwise. \end{cases} \tag{4}$$

Following the scheduling decision, a packet is transmitted to the wireless channel, and the transmission result is collected at the rate controller for rate adaptation in the next CP. The decision vector $\vec{p}(k)$ also serves as an input to the weight adjustor, which adaptively updates the control vector $\vec{c}(k)$, so that the long term expected goodput of each user approaches the assigned weight.

3.2. Adaptive rate control algorithm

In this section, we propose a simple but efficient ARF-like rate control algorithm implemented in the rate controller. The basic ideas included are: 1) multirate retry; 2) AIMD adjustment of the Success Threshold (ST).

The multirate retry mechanism is used to react quickly to the short-term channel variations and reduce fluctuations in the long-term data rate. The idea is that, each next-hop FIFO $queue_i$ is associated with a long-term rate r_i. When the HOL packet of $queue_i$ is going to the air, the rate controller fills in the retry series, in the transmission descriptor for that packet, with the current value of r_i and corresponding lower retry rates than r_i. If the first attempt fails, retries at lower rates automatically take place without changing r_i. Therefore, small-scale variations can be quickly resolved to avoid premature long-term rate adjustments.

We adjust the long-term rate r_i by a threshold scheme similar to ARF. It has been shown that a fixed value of ST is very sensitive to the changing speed of link quality [4]. Intuitively, it is better to increase ST so that we can minimize the undesired rate increments for slow fading channels; on the other hand, when the channel condition is fluctuating rapidly, it is critical to locate an optimal rate and stay there as long as possible. Here we propose a dynamic ST scheme based on the transmission history. If the previous ST has been reached, followed by a rate increment, we conclude that the channel quality is improving. Since the data rate is already higher now, we increase ST to stay in the current high rate as long as possible. If a transmission failure immediately follows the rate increment, we also increase ST to discourage premature rate increments. For the Failure Threshold (FT), a high value may degrade the system performance, since too many transmissions failed before the data rate is reduced. On the contrary, it is very likely that a single failure is due to fast fading, which should be handled by the multirate retry strategy, while bursty failures are due to stable deterioration in channel quality. Therefore, ARC requires that once 2 consecutive packets are retransmitted, the data rate is reduced as well as ST, to encourage potential rate increments. Specifically, the ST value is adjusted in the interval of $[ST_{min}, ST_{max}]$, using a similar way as the one of TCP's AIMD congestion window adjustment. Algorithm 1 illustrates the ideas presented above.

Algorithm 1 Adaptive Rate Control

```
ARC with Multi-rate Retransmission and AIMD ST Adjustment.
while receiving a packet P dequed from one of the FIFO queues do
  NextHop = daddr(p);
  P.dataRate = NextHop.dataRate;
  send(P);
  if recvACK() then
    if retry == 0 then
      Success++; Failure = 0;
      if Success ≥ ST then
        NextHop.dataRate++;
        ST+ = α; Recovery = 1;
      end if
    else if retry > 0 then
      Success = 0; Failure++;
      oldRate = NextHop.dataRate;
      if Failure ≥ FT then
        NextHop.dataRate--;
      end if
      if Recovery == 1 then
        ST+ = α;
      else if oldRate < NextHop.dataRate then
        ST/ = β;
      end if
      Recovery = 0;
    end if
  end if
  if ACKtimeout() then
    P.dataRate--;
    if retry < retryLimit then
      retry++; retransmit(P);
    else
      Failure++; Success = 0;
      Recovery = 0; drop(P);
    end if
  end if
end while
```

3.3. Weighted fair scheduling: optimization model

Our focus in this section is on systems where packets destined for each downlink user are queued at the AP and all flows have infinite backlogs of bits. The task of the scheduler at the AP is to schedule a packet transmission to one of the most profitable users, in the hope that such a transmission maximizes the system performance. One greedy scheduling policy is to always choose the user with the best channel quality. However, in the case where one user always has a good channel, other users will suffer from starvation. Therefore, we need to consider a general tradeoff model which maximizes an utilization function subject to the assigned fairness constraint.

3.3.1. Packet scheduler. Following the framework described in section 3, we present our packet scheduler formulation. Without loss of generality, the aggregate goodput (application layer throughput) is used as the system performance metric. Consider N users in total (not including the AP) in the system. Each one operates on the same PHY layer with the same rate set $R = \{R_1, R_2, ... R_M\}$. The objective of the scheduler is to choose one $user_i$ out of N at the beginning of the k_{th} CP, based on the output $\vec{r}(k) = [r_1(k), r_2(k), ..., r_N(k)]$, of the rate controller, where $r_i(k) \in R$ is the current data rate of $user_i$. The scheduler output $p(\vec{k})$ (defined in (4)) states that if $user_i$ is scheduled, $p_i(k) = 1$; otherwise, $p_i(k) = 0$. Since only one user is scheduled in one CP, it follows that $\sum_{i=1}^{N} p_i(k) = 1$. The expected goodput of the scheduled $user_i$ in the k_{th} CP is $g_i(k)$. Note that $g_i(k) = g(r_i(k), s_i(k), x_i(k))$, where $s_i(k)$, $x_i(k)$ are the packet length and channel condition respec-

tively, at the $user_i$ during the k_{th} CP. However, as we do not have the exact knowledge of $x_i(k)$, we can only compute the goodput using (1) approximately. Following this, the goodput in the current CP can be represented as $G(k) = \sum_{i=1}^{N} p_i(k)g_i(k)$. In the long run, the system goodput is given by:

$$G = E[G(k)] = E[\sum_{i=1}^{N} p_i(k)g_i(k)].$$

Alternatively, let $G_i = E[p_i(k)g_i(k)]$ be the long term goodput of $user_i$. The system goodput can be rewritten as:

$$G = \sum_{i=1}^{N} E[p_i(k)g_i(k)] = \sum_{i=1}^{N} G_i.$$

The optimization model is to maximize the system goodput, given the assigned weight vector $\vec{w}$, i.e.,

$$\max \quad G = \sum_{i=1}^{N} G_i, \quad s.t. \tag{5}$$

$$\frac{G_i}{w_i} = \frac{G_j}{w_j}, \quad \forall 1 \leq i, j \leq N. \tag{6}$$

Using a similar technique as the one in [11], (5) and (6) can be transformed to the following equivalent problem:

$$\max \quad Z = \frac{G_i}{w_i}, \quad (1 \leq i \leq N). \tag{7}$$

Since all $(\frac{G_i}{w_i})'s$ are identical, once again, the objective function in (7) is equivalent to:

$$\max \quad Y = (\sum_{i=1}^{N} c_i w_i)Z, \quad where$$

$$c_i \geq 0, \quad Z = \frac{G_i}{w_i}, \quad (1 \leq i \leq N).$$

It follows that the above problem can be rewritten as:

$$\max \quad Y = \sum_{i=1}^{N} c_i G_i, \quad s.t. \tag{8}$$

$$\frac{G_i}{w_i} = \frac{G_j}{w_j}, \quad \forall 1 \leq i, j \leq N. \tag{9}$$

Therefore, the original problem defined in (5) and (6) is equivalently transformed to the optimization problem defined in (8) and (9), where $\vec{c}$ is a vector with non-negative values. Suppose we are able to choose an appropriate control vector $\vec{c}^*$, such that it can drive $(G_i)'s$ to satisfy (9), then we only need to consider the optimization model

$$\max \quad Y = \sum_{i=1}^{N} c_i^* G_i. \tag{10}$$

Based on the above analysis, define the scheduling policy at the LLC layer of the AP to be:

$$S^*(\vec{G}(k)) = \arg\max_i \; c_i^* G_i(k), \tag{11}$$

such that in each CP, the weighted goodput is maximized. Obviously, the policy $S^*(\vec{G}(k))$ yields a solution to the problem defined in (10).

3.3.2. Weight adjustor. Note that we introduce the control vector $\vec{c}^*$, which is dependent on the distribution of $\vec{G}$, to approach the fairness constraint. However, since the full knowledge of channel conditions is not known prior, a fixed $\vec{c}^*$ obtained at the very beginning and leading to the optimal value is not available. Rather, we define an online updating process to dynamically adjust the control vector $\vec{c}^*$, which is the task of the weight adjustor introduced here. In a special case where c_i^* is always 1 for all $i's$, the scheduling policy reduces to the greedy one mentioned previously, i.e., the one that always schedules the user with the largest goodput in the current CP computed by (1).

The weight adjustor estimates the control vector using a standard stochastic approximation algorithm [10][11], introduced by Robbins and Monro for finding the solution to the equation $f(x^*) = 0$ when f is observed with error $y^k = f(x^k) + \xi^k$, where $\{\xi^k\}$ is a sequence of random errors. In this paper we assume $\{\xi^k\}$ are independent with mean zero and bounded variances. The idea of the Robbins-Monro algorithm is to iterate the sequence $x^{k+1} = x^k - a^k y^k$ until convergence [9] [17]. In our case, define

$$f_i(\vec{c}) = \frac{G_i}{\sum_{i=1}^{N} G_i} - \frac{w_i}{\sum_{i=1}^{N} w_i}, \quad i = 1, ..., N.$$

We use the stochastic approximation to generate a sequence of iterations $\vec{c}^1$, $\vec{c}^2$, ..., that converges to $\vec{c}^*$. In each iteration, the scheduling policy is given by:

$$S^k(\vec{G}(k)) = \arg\max_i \; c_i^k G_i(k).$$

Hence, we need an estimation of y^k at $f(\vec{c}^k)$. Note that in each CP, we have a noise observation of $f_i(\vec{c}^k)$, i.e.,

$$y_i^k = p_i(k) - \frac{w_i}{\sum_{i=1}^{N} w_i}, \quad i = 1, ..., N,$$

where $p_i(k)$ is defined in (4). The observation error is:

$$E[\xi_i^k] = E[p_i(k) - \frac{G_i}{\sum_{i=1}^{N} G_i}] = 0.$$

Therefore, we can use the stochastic approximation algorithm to adaptively find $\vec{c}^*$ by

$$c_i^{k+1} = c_i^k - a^k y_i^k,$$

where the step size a^k should be appropriately chosen to converge to zero, e.g., $a^k = 1/k$. Algorithm 2 shows the structure of WFS.

Algorithm 2 Weighted Fair Scheduling

Weighted Fair Scheduling: $\frac{G_i}{w_i} = \frac{G_j}{w_j}, \quad \forall 1 \leq i, j \leq N.$
while receiving a packet P from the upper layer **do**
  $Enque(P)$;
  if not blocked **then**
    return $P = Dequeue()$;
  end if
end while

void $Enque(P)$:
$qid = findEnqueID(P.daddr())$;
$flowQueue[qid].enque(P)$;

Packet* $Dequeue()$:
if has not dequed any packet **then**
  $initwi()$;
else
  $updatewi()$ using stochastic approximation;
end if
for all $NextHop_i$ **do**
  $G_i =$ compute $EGi(r_i, s_i)$ by (1);
  $WeightedG[i] = c_i G_i$;
end for
$qid = selectBest(WeightedG[])$;
return $flowQueue[qid].deque()$;

Table 1. MAC/PHY and control parameters

$[CW_{min}, CW_{max}]$	[15, 1023]
SlotTime, SIFSTime	$9\mu s, 16\mu s$
BasicRate	6Mbps
PHY overhead	$20\mu s$
ARC (ST_{min}, ST_{max}, FT)	(8, 50, 2)
ARC (α, β)	(16,2)

4. Performance evaluation

In this section, we present the performance evaluation of the proposed WFS-ARC framework in $ns2$ [12]. Table 1 summarizes some parameters used in the simulation. For more realistic consideration, we make necessary changes to the original $ns2$ MAC/PHY layers and incorporate the Ricean Fading channel model introduced in [14]. Saturated CBR traffic is used. We run each simulation for at least 100 seconds with data payload size of 1000 Bytes.

4.1. ARC performance

Here we consider the single receiver case and evaluate the performance ARC. In the first case, suppose the only static user is far away from the AP. Without rate adaptation, the AP always sends packets at a specified data rate, which may either cause lots of transmission errors or a waste of bandwidth due to a too low rate. In our scenario, the optimal rate is 18 Mbps (Figure 2). With ARC, the AP can tune to the best data rate (Figure 3). Now suppose the single user is moving towards the AP. The channel condition is improving and the packet error rate at 18 Mbps is lower than in the previous case. Without ARC, the maximum goodput is approximately 12.8 Mbps (Figure 4). On the other hand, ARC can quickly tune to the improving channel quality and reach the maximum 54 Mbps (Figure 5).

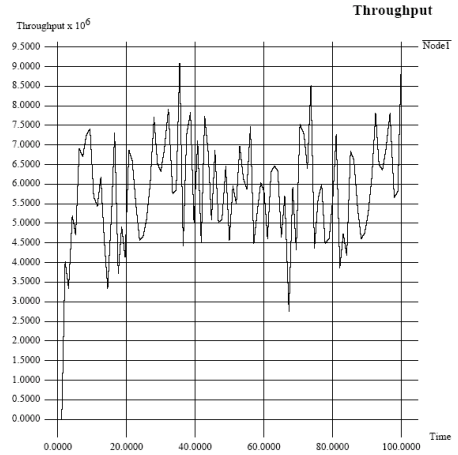

Figure 2. 802.11a/18: single static user

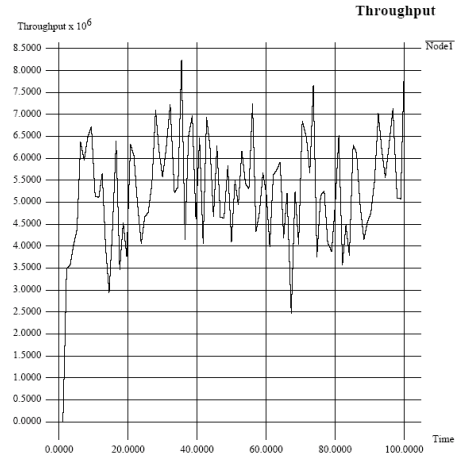

Figure 3. 802.11a/ARC: single static user

4.2. WFS-ARC performance

Here we study the performance of 802.11/ARC and 802.11/WFS-ARC. We first give a simple two-user scenario, where $user_1$ is close to the AP and $user_2$ is far away

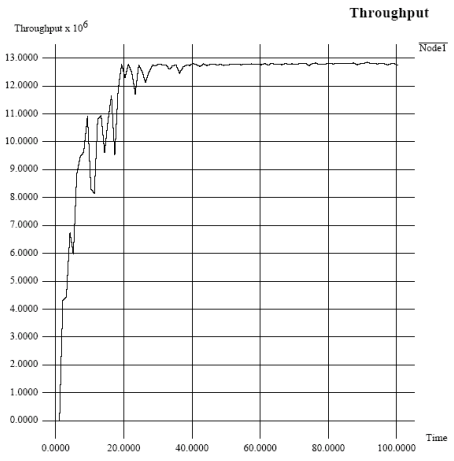

Figure 4. 802.11a/18: single moving user

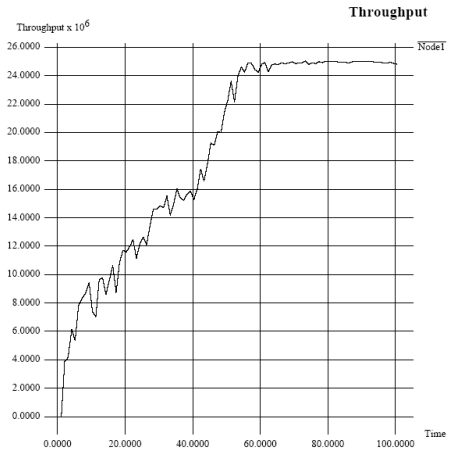

Figure 5. 802.11a/ARC: single moving user

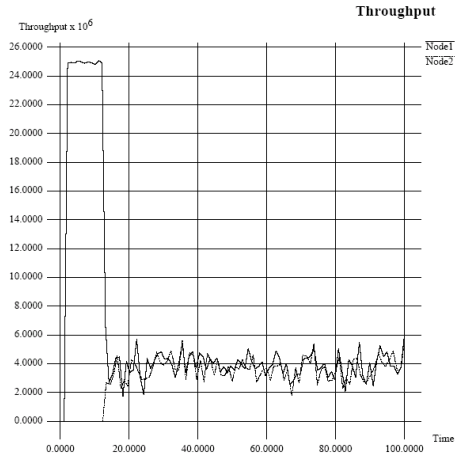

Figure 6. 802.11a/ARC: two static users

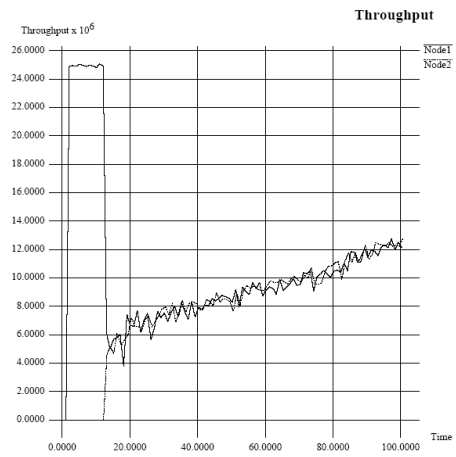

Figure 7. 802.11a/ARC: two moving users

from it. The AP starts the flow of $user_1$ first and later of $user_2$. With ARC, though the AP can always transmit to $user_1$ at 54 Mbps and can only send at 18 Mbps to $user_2$, the goodput of the two users are approximately the same (Figure 6). This can be explained by the fact that the two users have roughly the same opportunity to access the channel, but the slower user consumes more radio time and leads to very low channel utilization. This phenomenon is more evident in another two-user case, where $user_2$ is moving towards the AP. The goodput of the two users are still approximately the same throughout the time (Figure 7). However, both are increasing since the channel quality of $user_2$ is now improving and the time fraction it takes up is less compared to the previous case. In Figure 8, the weights for $user_1$ and $user_2$ in previous two-static-user case (Figure 6) are set to be 2.4 and 1 respectively, taking account of different channel qualities. We can see that our framework converges quickly to the assigned weights and the system goodput is improved by 50% while not sacrificing fairness too much. The improvement is even more if we set the weights to be 5 and 1 for $user_1$ and $user_2$ (Figure 9). Both results reflect the well-known throughput and fairness tradeoff: the higher the system throughput is, the lower the fairness is. Figure 10 plots the aggregate goodput of both schemes as a function of the number of users, assuming saturated downlink CBR traffic. For 802.11/ARC, the system performance is greatly dependent on the behavior of the "worst user". For 802.11/WFS-ARC, with properly assigned weights (e.g., the weights satisfying the QoS requirement), significant system goodput gain can be achieved.

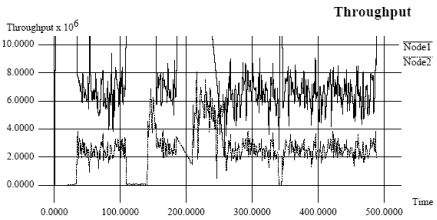

Figure 8. 802.11a/WFS-ARC (2.4:1)

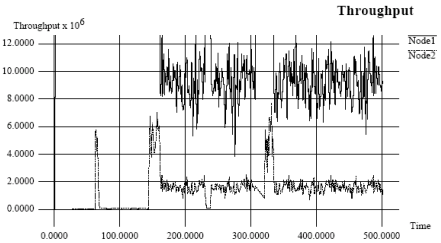

Figure 9. 802.11a/WFS-ARC (5:1)

5. Conclusions

In this paper, we propose a cross layer design, WFS-ARC, for joint rate control and packet scheduling, so that the LLC/MAC layers can exploit the multirate PHY layer capability and the multiuser diversity. The problem is modeled to maximize the system goodput with a rate adaptive MAC layer, while satisfying the assigned fairness constraint. We study the saturation behavior in different cases and the simulation results demonstrate that through well-designed cooperation of different layers, superior performance gain can be achieved. This scheme can be easily adopted by the sate-of-the-art IEEE 802.11 AP products, since it can be implemented in the device driver and no modification to the hardware is required. It should also be easy to extend this to Ad Hoc networks.

References

[1] ICC Panel on Defining Cross-layer Design in Wireless Networking, May 2003.

[2] R. A. Berry and E. M. Yeh. Cross-Layer Wireless Resource Allocation. *IEEE Trans. Signal Processing*, 21:59–68, Sept. 2004.

[3] P. Bhagwat, P. Bhattacharya, A. Krishna, and S. Tripathi. Enhancing Throughput over Wireless LANs using Channel State Dependent Packet Scheduling. In *Proc. IEEE INFOCOM'96*, pages 1133–1140, San Franciso, CA, Mar. 1996.

[4] G. Holland, N. Vaidya, and P. Bahl. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *Proc. ACM MOBICOM'01*, pages 236–251, Rome, Italy, July 2001.

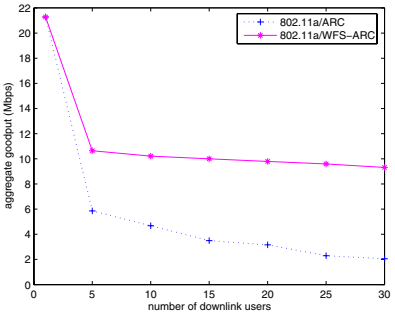

Figure 10. Aggregate goodput of multiple users: 802.11a/ARC vs. 802.11a/WFS-ARC

[5] IEEE Std 802.11-1999. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Aug. 1999.

[6] IEEE Std 802.11a-1999. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band, Sept. 1999.

[7] L. B. Jiang and S. C. Liew. An Adaptive Round Robin Scheduler for Head-of-Line-Blocking problem in Wireless LANs. The Chinese University of Hong Kong.

[8] V. Kawadia and P. Kumar. A Cautionary Perspective on Cross-Layer Design. *IEEE Trans. Wireless Commun.*, pages 3–11, Feb. 2005.

[9] H. Kushner and G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, New York, 1997.

[10] X. Liu, E. Chong, and N. Shroff. Transmission Scheduling for Efficient Wireless Utilization. In *Proc. IEEE INFOCOM'01*, pages 776–785, Anchorage, Alaska, Apr. 2001.

[11] Y. Liu and E. Knightly. Opportunistic Fair Scheduling over Multiple Wireless Channels. In *Proc. IEEE INFOCOM'03*, pages 1106–1115, San Franciso, CA, Mar. 2003.

[12] NS2. The Network Simulator2, 2003.

[13] D. Pong and T. Moors. Fairness and Capacity Trade-off in IEEE 802.11 WLANs. In *Proc. IEEE LCN'04*, pages 310–317, Tampa, Florida, Nov. 2004.

[14] Ricean Fading. Additions to the NS network simulator to handle Ricean and Rayleigh fading, 2000.

[15] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. OAR: An Opportunistic Auto-Rate Media Access Protocol for Ad Hoc Networks. In *Proc. ACM MOBICOM'02*, Atlanta, GA, Sept. 2002.

[16] A. van der Vegt. Auto Rate Fallback Algorithm for the IEEE 802.11a Standard. Technical report, Utrecht University, 2002.

[17] I. J. Wang, E. K. P. Chong, and S. R. Kulkarni. Weighted Averaging and Stochastic Approximation. *Math. Control Signals Systems*, 1(10):41–60, 1997.