Ad Hoc Networks 000 (2015) 1–11

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A new packet scheduling algorithm for access points in crowded WLANs

Yi Yao[a,*], Bo Sheng[b], Ningfang Mi[a]

[a] Northeastern University, United States
[b] University of Massachusetts Boston, United States

ARTICLE INFO

Article history:
Received 9 August 2014
Revised 2 May 2015
Accepted 1 June 2015
Available online xxx

Keywords:
Packet scheduling
Access point
Scheduling policy
Shortest job first (SJF)
Fairness

ABSTRACT

Nowadays, Wireless LANs (WLANs) have been densely deployed to provide the last mile delivery of Internet access to mobile clients. As the population of WLAN clients who carry WiFi-enabled devices keeps increasing, WLANs are often crowded and WLAN clients may thus encounter serious performance degradation due to channel contention and interference. Therefore, in this paper we present a new packet scheduling algorithm, named DAT, for access points (APs) in a crowded 802.11 WLAN. Our goal is to improve the performance of *efficiency* (measured by packet response time or throughput) and *fairness* which often conflict with each other. To meet this goal, our solution is to aggregate and balance both performance metrics by enabling an AP to automatically adjust time windows for serving each active WLAN client. Specifically, our algorithm leverages the knowledge of the observed traffic to dynamically shift the weight between efficiency and fairness and strikes to improve the preferred performance metric without excessively degrading the other one. A valid queuing model is designed in this work to evaluate the performance of our new scheduling algorithm. Trace-driven simulations demonstrate that the proposed algorithm successfully balances the trade off between efficiency and fairness in crowded WLANs.

1. Introduction

In recent years, the population of wireless LAN (WLAN) clients who carry WiFi-enabled devices has increasedly exploded. To tackle such a population explosion, WLANs have been densely deployed to provide the last mile delivery of Internet access to mobile clients. For example, wigle.net has reported more than 33 million observed WiFi networks. SkyHook, a WiFi-based localization service, has claimed to have “tens of millions” access points (APs) in its database. With the dense deployment of WiFi infrastructure and the evolution of 802.11 family, WLANs keep playing an extremely important role in serving mobile clients. When more and more people carry WiFi-enabled devices, such as laptop, smartphone, and iPad, WLANs are often crowded, especially at particular locations with special events, e.g., a meeting room for a large conference or a stadium hosting a sports game. Under a heavy traffic load, WLAN clients may encounter serious performance degradation due to channel contention and interference.

This paper focuses on developing a new scheduling algorithm for APs to improve the performance of a crowded WLAN. Specifically, we focus on the packet scheduling of the downlink traffic, i.e., from AP to clients, as it carries the majority of data. Typically, an AP applies the First-In-First-Out (FIFO) scheduling discipline, i.e., the first packet arriving from Internet will be first sent via the wireless channel. FIFO strategy works well under light traffic load, where each downlink packet can be immediately sent to a client with little

* Corresponding author. Tel.: +1 6176801590.
E-mail addresses: yyao@ece.neu.edu (Y. Yao), shengbo@cs.umb.edu (B. Sheng), ningfang@ece.neu.edu (N. Mi).

http://dx.doi.org/10.1016/j.adhoc.2015.06.001
1570-8705/© 2015 Elsevier B.V. All rights reserved.

delay at the AP. However, when the traffic load becomes heavy, downlink packets cannot be delivered right after arrivals at APs. Instead, each AP maintains a queue to buffer the incoming packets from the Internet. Every packet in the queue has different attributes such as packet size and transmitting rate (according to the destination client). The simple FIFO policy, however, ignores all these characteristics and can barely yield the optimal performance. For example, when the AP delivers a packet with low transmitting rate, the response times of all other packets waiting in the queue are increased by the transmission time.

To address the above issue, we design a new AP scheduling algorithm which selects the next to-be-delivered packet under the consideration of two metrics, *link efficiency* measured by throughput and *fairness* among all the clients. When an AP serves multiple clients, the wireless link qualities between each client and the AP are different. In order to improve efficiency, the AP prefers to first send the packets through fastest links. However, other clients with slow links may suffer from starvation, which thus causes non-negligible unfairness. Therefore, we propose a new scheduling scheme, named DAT, that dynamically adjusts the time windows allocated to each client with the goal of achieving the balance between efficiency and fairness.

Our basic idea is to combine Round-Robin, which achieves the best fairness, with an adaptive time window for service. The AP rotates among all active clients and delivers the packets in the buffer for them one after another. Each client is assigned a time window for serving its packets, i.e., during a given time window, the AP continuously sends the packets to a particular client. DAT scheme dynamically adjusts the time window for each client according to the observed efficiency and fairness values. Our goal is to aggregate these two metrics and balance the performance of them. We build a queuing model that captures behaviors of WLAN clients and the AP for performance evaluation. Comprehensive trace-driven simulations are then conducted to evaluate the proposed scheduling algorithm and compare to two classic policies. The simulation results show that our algorithm is achieving a well balance between efficiency and fairness.

The rest of this paper is organized as follows. Section 2 summarizes the prior work and Section 3 presents our new scheduling algorithm. In Section 4, we introduce the queuing model for evaluation. The simulation results are reported in Section 5. Finally, we conclude in Section 6.

2. Related work

Throughput and fairness are traditional metrics for network packet scheduling. They have also been well studied in the WLANs' literature. One direction particularly works on the TCP flows [1–5]. The key problem is to handle head-of-line blocking and the competition between TCP data packets and TCP ACKs, especially when considering the channel errors. While this paper focuses on the MAC layer scheduling, the prior work on TCP flows can certainly be combined with our solution to form a cross-layer scheduling scheme.

Another direction in the prior work is to improve throughput and fairness by managing the whole WLAN such as strategically associating clients with APs [6–11], assigning channels to APs [12–14], or hybrid approaches [15]. The default association in practice is to let a client associate with the closest AP (with the strongest signal strength). This simple policy is not optimal from the perspective of managing a WiFi network consisting of multiple APs. Similarly, the default channel selection of each AP is arbitrary lacking consideration of interference with nearby APs. Most of these prior work requires a central control and coordination among APs. They are also complementary to our work in this paper which handles the packet level scheduling on a single AP. Additionally, AP scheduling for saving power consumption has also been well studied in the literature [16–20]. Their goal is to tune AP scheduling such that clients associated with the AP can stay in the power saving mode as long as possible. This paper targets on the scheduling on a busy AP in a crowded WiFi network, where the power saving mode is not enabled.

Furthermore, some previous work [21,22] proposed to achieve time-based fairness in WLANs via rate adaptation with modifications on 802.11 standards. Their solutions focus on the rate adaptation algorithms and require modifications on 802.11 standards. Our work targets on MAC layer scheduling and we consider the fairness of throughput. Finally, wireless packet scheduling is often studied to provide QoS represented by the IEEE 802.11e standard [23]. In 802.11e, high priority traffic is given shorter delay-related parameters such as contention window (CW) and arbitration inter-frame space (AIFS) so that it has better chance to be sent than low priority traffic. Our scheduler proposed in this paper is also complementary to QoS provision scheme. While 802.11e distinguishes traffics with different priorities, our scheme can be used to schedule the traffics with the same priority.

3. New AP scheduling algorithm: DAT

In this section, we present our solution DAT, which dynamically adjusts the time window occupied by each client to improve performance. We first introduce the system model and problem formulation, and then we describe the algorithm details.

3.1. System model and overview

We consider that an access point (AP) serves n clients, $\{c_1, c_2, \ldots, c_n\}$, in a crowded wireless LAN. We assume that clients are ordered based on their effective downlink rates, considering the MAC layer transmitting rates, acknowledgment, retransmission, and other per-packet overheads. It follows that client c_1 has the slowest link to the AP and c_n is connected with the fastest link. When the AP is over-loaded, the downlink packets may be buffered in a queue at the AP before being sent out. Normally, the queue has a capacity limit indicated by a maximum number of packets that can be held in the buffer. Later in Section 5, we show the evaluation under infinite queue capacity as well.

In this paper, we consider two metrics as the performance objective of an AP scheduling algorithm: efficiency and fairness. The first metric (i.e., efficiency) is measured by the packet response time or alternatively by the throughput. The second metric (i.e., fairness) is measured by Jain's fairness index [24]. Despite that both metrics are critical in the scheduler evaluation, it is often difficult to improve them simultaneously under a particular AP scheduling policy. For example, one can use Round Robin (RR) to achieve the best fairness. By rotating among all active clients, RR always delivers one packet for each client in a round. This policy can certainly avoid starvation, but the efficiency under RR is very poor. In contrast, the other extreme of scheduling (e.g., MaxTP) is to always give higher priority to fast links, i.e., keep sending the available packets to a client which has the highest downlink rate. As a result, the optimal efficiency in terms of packet throughput or packet transmit delay time is achieved under MaxTP while poor fairness often becomes a big problem under this policy because it unfairly treats the clients with slow downlink rates.

How to balance the trade-off between efficiency and fairness is imminently important and challenging in the AP scheduler design. In this paper, we propose a new scheduling algorithm, named DAT, which takes into account both performance metrics (i.e., efficiency and fairness) in the scheduling of downlink packets at the AP and strikes to obtain the fairness and the efficiency close to the optimal results provided by RR and MaxTP, respectively.

Although this paper targets on the scheduling algorithm for a single AP, the solution is also applicable to the practical environment with multiple APs. For the APs that are configured with different wireless channels, their scheduling decisions are individually made and the data communication with their clients do not conflict with each other. For the APs whose communication channels are the same or overlapped, their concurrent wireless signals will cause interference if they are within each other's communication range. According to WiFI's distributed coordination function, each of the contenting APs has the equal opportunity to occupy the channel. Thus, the solution in this paper can be applied by each AP to manage the *effective* channel time obtained by the AP. In fact, each client's downlink traffic has the same probability to be affected by the interference caused by other nearby APs. Therefore, our solution can achieve the same objectives for efficiency and fairness when deployed on multiple APs.

3.2. Algorithm description

Our new algorithm adopts the basic idea of Round Robin by rotating clients to serve. However, we assign *different* service times to each client. Specifically, the AP selects a client c to serve and allocates a time window to delivery client c's packets. That is, the AP keeps sending the packets for client c till the assigned time window is elapsed or there is no more packets available for client c. Then, the AP selects another client and starts delivering its packets. Different from the RR scheme, the duration of the time window for each client is dynamically adjusted across time by considering the trade-off between efficiency and fairness.

Let w denote the minimum time slice (finest granularity), e.g., $w = 0.01$ s. We then consider that the AP selects a window size from a set of k discrete values $\{1w, 2w, 3w, ..., kw\}$ to deliver a client's packets. In our DAT scheme, the AP chooses the best window size for each client from these k values based on the following two target functions.

- *"Relative Efficiency" function:* This function expresses the relative efficiency gain for a particular choice of the window size. Intuitively, if efficiency is the only concern, then a good policy (e.g., MaxTP) should always consider large windows for clients with fast link rates in order to empty the AP's buffer as soon as possible, resulting in short packet response times, high system throughput and high system availability. To characterize the effect of the window size $i \cdot w$, we define the *Relative Efficiency* function as follows:

$$\alpha_i = i \cdot w \cdot \frac{\text{Rate}_c - \overline{\text{Rate}}}{\overline{\text{PackSize}}}, \quad \forall i \in [1, k], \tag{1}$$

where Rate_c represents the link rate of the selected client c, $\overline{\text{Rate}}$ represents the average link rate of all the remaining active clients that have packets in the AP buffer, and $\overline{\text{PackSize}}$ is the mean size of the packets. A higher value of α_i indicates more packets expelled from the queue, thus a better efficiency is achieved. Based on Eq. (1), if $\text{Rate}_c > \overline{\text{Rate}}$, then the AP prefers to allocate a large window (or the largest window if α_i is the only metric) to client c . Otherwise, if α_i becomes negative, the AP would reduce the window size as much as possible. We remark that α_i provides a good indication of efficiency that the policy can achieve when the AP assigns a particular time window to a client.

- *"Expected Fairness" function:* Our second target function is designed to quantify the fairness. As mentioned earlier, we use Jain's fairness index [24] as the metric to measure the fairness of a given scheduling policy. Eq. (2) gives the definition of Jain's fairness index I:

$$I = \frac{\left(\sum_{j=1}^{n} \text{TP}_j\right)^2}{n \cdot \sum_{j=1}^{n} \text{TP}_j{}^2}, \tag{2}$$

where n is the number of active clients and TP_j represents the throughput of client j in a predefined time period. The range of I is between 0 and 1, and a higher value of I indicates a better fairness. The main goal of our second target function, named *Expected Fairness*, is to estimate the packet throughput among all active clients and thus express the performance of fairness. To accomplish it, DAT on-line tracks each client's throughput in the past monitoring window t and uses this information to decide the duration of the time window for the current client. The intuition is that if the client has already received a higher throughput in the previous monitoring window, then a smaller time window should be chosen for that particular client in the next round, and vice versa. Note that the size of monitoring window t is a user-specific parameter and we will describe its setting in Section 5.
Given a client c and a candidate time window $i \cdot w$, we have the following equations to calculate the expected throughput for all n clients if DAT decides to send the packets to client c during the $i \cdot w$ time period:

$$\text{TP}_j = \begin{cases} \dfrac{S_j}{t + i \cdot w}, & \text{for } j \neq c, \\ \dfrac{S_c + s}{t + i \cdot w}, & \text{for } j = c, \end{cases} \tag{3}$$

where S_j represents the total amount of data transmitted to client j during the previous monitoring window t, and s equals to the estimated amount of data that can be transmitted to client c within the $i \cdot w$ time window, i.e., $s = \text{Rate}_c \cdot i \cdot w$. Let $\widehat{\text{Sum}} = \sum_{j=1}^{n} S_j$ and $\widehat{\text{Sum}_2} = \sum_{j=1}^{n} {S_j}^2$. We then express the *Expected Fairness* target function as follows:

$$\beta_i = \frac{(s + \widehat{\text{Sum}})^2}{n(\widehat{\text{Sum}_2 + 2 \cdot s \cdot S_c + s^2})}. \tag{4}$$

Given the above two target functions, DAT further uses the 0-1 scaling technique to scale α_i and β_i for all k candidate time windows (i.e., $1 \le i \le k$) as follows.

$$\alpha_i' = \frac{\alpha_i - \alpha_{\min}}{\alpha_{\max} - \alpha_{\min}}. \tag{5}$$

$$\beta_i' = \frac{\beta_i - \beta_{\min}}{\beta_{\max} - \beta_{\min}}. \tag{6}$$

Then, DAT selects the best window size for the current client based on the following equation:

$$P_i = w_1 \cdot \alpha_i' + w_2 \cdot \beta_i', \tag{7}$$

where w_1 and w_2 are the user-defined weights for α_i and β_i, respectively. We expect that higher P_i will achieve better efficiency/fairness balance. Therefore, the time window which can get the highest value of P_i will then be assigned to the current client. The major steps of DAT are presented in Algorithm 1.

The mainframe of Algorithm 1 is a round-robin process, where a client c is selected to be served by the AP. The first loop (lines 3–6) enumerates all possible window size ($i \cdot w$) for c and calculates the corresponding values of relative efficiency and expected fairness, i.e., α_i and β_i. In the second loop (lines 8–14), the algorithm normalizes α_i and β_i into the range of [0,1], and then uses the aggregate utility function Eq. 7 to pick the optimal window size. Variables P' and i' record the current optimal value of the utility function and the corresponding window size yielding it. As we periodically choose the optimal window size that maximizes utility function based on current system information, DAT cannot be guaranteed to be optimal for a long time without a prior knowledge of future traffic. Moreover, DAT is not optimal for both efficiency and fairness. Instead, our design strikes to achieve a good balance between these two metrics as shown in Eq. (7).

Algorithm 1 DAT scheduling algorithm.

```
1: while number of active clients $n > 0$ do
2:   Choose an active client $c$ based on round-robin rotation
     and set current time as $t_0$;
3:   for $i = 1$ to $k$ do
4:     calculate $\alpha_i$ using Eq. (1);
5:     calculate $\beta_i$ using Eq. (4);
6:   end for
7:   $P' \leftarrow 0, i' \leftarrow 0$;
8:   for $i = 1$ to $k$ do
9:     scale $\alpha_i$ and $\beta_i$ to range [0,1] by 0–1 scaling method;
10:    calculate $P_i$ using Eq.~(??);
11:    if $P_i > P'$ then
12:      $P' \leftarrow P_i$ and $i' \leftarrow i$;
13:    end if
14:  end for
15:  Assign window size $i' \cdot w$ to client $c$;
16:  while current time $t < t_0 + i' \cdot w$ and number of packets from client $c > 0$ do
17:    send a packet to client $c$;
18:    update history information $S_c$, $\widehat{\text{Sum}}$, and $\widehat{\text{Sum}_2}$;
19:  end while
20: end while
```

4. Simulation model

In this section, we present a queuing model built for scheduling evaluation. We consider the circumstance where requests from multiple clients and the corresponding reply packets from the AP are all sent through the same wireless channel. Fig. 1 illustrates the model that captures the behavior observed in the single AP situation and evaluates the performance of different scheduling algorithms under heavy load conditions.

In this model, an infinite queue (Q_0) with n servers is used to emulate the activity of n clients in the system, where each server represents a single client, independently sending requests to the AP. We model the request inter-arrival times as a Markovian arrival process (MAP) [25] for each server in Q_0 such that each client can have different arrival rates and different arrival distributions. The specifications of each request include request arrival times, request sizes in bytes,

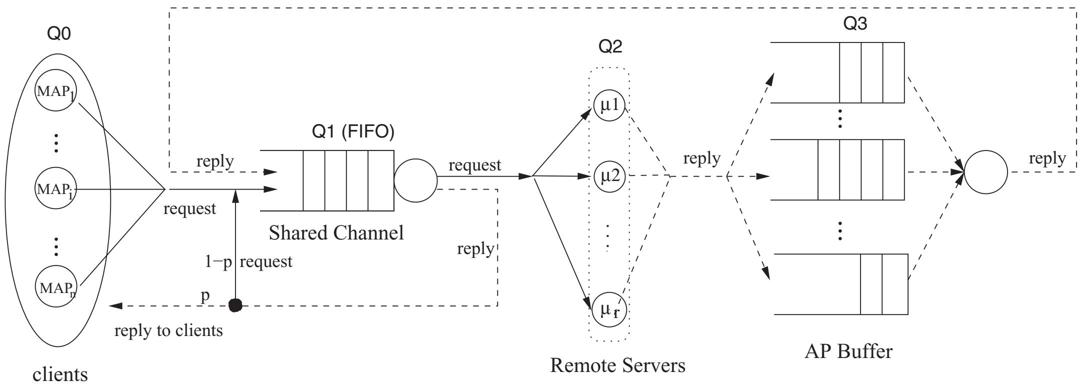

Fig. 1. A queuing model of the single AP wireless network.

client index, and effective uplink/downlink rates. All requests generated by Q_0 are then enqueued to the queue Q_1, waiting for the transmission in the shared wireless channel. That is, Q_1 is adopted to emulate all transmissions from/to the AP through a shared wireless channel. We remark that in the real wireless network, clients usually trigger a retry mechanism if there are simultaneous contentions for the shared channel. Since our focus is on AP's scheduling for downlink packets, we here simply ignore such a retry mechanism and instead transmit these requests in the order of their arrival times using the FIFO discipline at channel queue Q_1.

We further introduce a delay center Q_2 to model the processing times at remote servers, which process a received client request with a fixed time $1/\mu_i$ and then send the corresponding reply data back to the AP. As transmitted packets usually have a limited size, e.g. 1.5 K bytes, if a reply data set is larger than that particular limit, then the original one will be partitioned into several packets and transmitted at the packet level. It follows that instead of having one to one relationship between requests and reply packets, remote servers in the delay center Q_2 might generate m reply packets for each arrived request, where m is determined by the original reply data size and the size limit.

The reply packets received from the remote servers are then queued in the corresponding client buffer in the AP, shown as Q_3 in Fig. 1, waiting for the service or transmission at the shared channel Q_1. When detecting no reply packet waiting or serving at Q_1, the AP chooses a reply packet from the buffer and enqueues it to Q_1 immediately. The selection of the next reply packet to be transmitted is done according to different scheduling disciplines. For example, if the MaxTP policy is considered, then the AP buffer can be implemented as a priority queue based on transmitting rates. Consequently, the AP always selects a reply packet with fastest downlink rates. In real wireless networks, the transmitted reply packets might trigger one or several client requests after some delay time. We capture this behavior by adding a branch probability for reply packets at Q_1: with probability p a completed packet at Q_1 is simply forwarded to its associated client and with probability $1 - p$, a batch of client requests (≥ 1) are sent back to the channel queue.

5. Performance evaluation

In this section, we conduct trace-driven simulations to illustrate the effectiveness of our new scheduling algorithm. We also compare the performance of DAT with respect to efficiency and fairness with the other two classic policies, RR and MaxTP.

5.1. Evaluation settings

We conduct the simulations under the following three scenarios.

- *Base case*: The requests from each client follow a random arrival pattern with the same mean request arrival rate.
- *Burst case*: In this case, we choose the top two fastest clients and introduce idle and bursty periods in their request arrivals. The requests from other clients are the same as in the base case.
- *Uneven case*: In this case, we set the request rates of the top two fastest clients to be five times higher than those of the remaining clients.

Therefore, request inter-arrival times for each client are drawn from a Markovian arrival process (MAP), which has the ability of providing mean and variability at different levels as well as different bursiness profiles. We also use the trace from SIGCOMM 2008 [26] for generating requests and reply packets. In this trace, the mean size of requests is 322 bytes and the mean size of reply packets is 1004 bytes.

Default parameters: In our default setting, there are totally $n = 20$ clients. Each client has the fixed uplink and downlink rates throughout the whole simulation, and the link rate ranges from 100 KB to 1 MB. Without loss of generality, clients with larger index have faster link rates, such that client 20 has the highest uplink and downlink rates among all clients. By default, we set the minimum time slice $w = 0.01$ s, the number of candidate window sizes $k = 10$ and the branch probability $p = 1$. The user-specific parameters are set as follows: $t = 0.5$ s (size of the monitoring window), $w_1 = 1$ and $w_2 = 2$ (weights in Eq. (7)). Sensitivity analysis with varying parameters will be presented in Section 5.3.

For each trace set, we further consider both infinite buffer size situation and finite buffer size situation when measuring performance. With the setting of infinite buffer size, we measure the average response time for efficiency and average Jain's index for fairness. When considering finite buffer size, we additionally measure the number of dropped packets and the corresponding drop ratios.

Fairness measurement: In our simulation, the fairness index is measured across time within a 0.25-s time window. We also tried other time window lengths (e.g., 0.5 s, 1 s), which generate qualitatively the similar results. In each time window, we only consider the active clients for calculating the fairness index. We define a client to be active in a certain time window if it has sent requests during this time window or if it has a pending request (sent in past time windows) that has not been replied yet. In addition, we ignore the time windows with no active clients or only one active client since there is no fairness issue in such a situation.

Scale rate: The efficiency and fairness are two possibly conflicting metrics defined in different domains. In this work, we compare DAT to the optimal solution for each individual metric and try to strike a good balance between the two metrics. To clearly illustrate how DAT algorithm achieves the balance between efficiency and fairness, we further present the relative scale rate using the 0–1 scaling technique as follows: among all the polices (e.g., RR, DAT and MaxTP), we scale the best performance to 1 and the worst performance to 0 and then normalize our DAT's performance between 0 and 1, see Eq. (8).

$$\text{Scale rate} = \frac{|\text{DAT} - \text{Worst}|}{|\text{Best} - \text{Worst}|}. \tag{8}$$

Therefore, a larger-than-0.5 relative scale rate indicates that DAT performs closely to the best policy, e.g., with shorter response time or larger fairness index. If the relative scale rate is smaller than 0.5, then it implies the opposite. If the values with respect to both response time and fairness index are greater than 0.5, then we think that our DAT algorithm

Table 1
Performance under base case, numbers in parentheses are scale rates.

Scenario	Metrics	Policies				
		RR	DAT (w1=1, w2=2)	DAT (w1=1,w2=1)	DAT (w1=2,w2=1)	MaxTP
InfiniteBuffer	RespTime(s)	2.301	1.135 (0.82)	1.041 (0.89)	0.984 (0.93)	0.885
	FairIndex	0.766	0.626 (0.56)	0.604 (0.49)	0.586 (0.43)	0.450
FiniteBuffer	RespTime(s)	0.866	0.714 (0.69)	0.691 (0.80)	0.678 (0.86)	0.647
	FairIndex	0.758	0.641 (0.70)	0.587 (0.56)	0.556 (0.48)	0.369
	DropRatio(%)	2.010	1.012 (0.81)	0.921 (0.88)	0.872 (0.92)	0.773

obtains a well balance between the efficiency and the fairness.

5.2. *Performance improvement*

5.2.1. *Base case*

In this case, each client's request inter-arrival times are generated independently through a 2-state Markovian-Modulated Poisson Process (MMPP), which is a special case of the Markovian Arrival Process (MAP) [25]. The details of MMPP used in our simulations are presented in Appendix A. We stress that distributions of modern network traffic, such as packet and connection arrivals are no longer Poisson distributed [27]. Therefore, we introduce heavy-tailed WLAN packet arrival processes in our base case where the mean request arrival rate of each client c_i is set to $\lambda_i = 1.5\,\text{s}^{-1}$, and the squared coefficient of variation (SCV) of inter-arrival times is equal to 5. We also investigate the impact of the weight settings of w_1 and w_2 in Eq. (7).

The simulation results under different scheduling policies are shown in Table 1 with infinite and finite buffer size (maximum 800 packets in the buffer), respectively. The numbers in parentheses are the relative scale rates of DAT. For DAT, the weights configuration can help adjust the trade-off between efficiency and fairness. Higher weight of Relative Efficiency function yields better response times, i.e., response times close to MaxTP when configure $w_1 = 2$ and $w_2 = 1$. While higher weight of Expected Fairness function gives better fairness among clients. As we believe that a good trade-off should bring both response time and fairness performance close to the optimal value, best trade-off is achieved under the configuration of $w_1 = 1$ and $w_2 = 2$ in this case. With such setting, in both infinite buffer and finite buffer settings, mean response times obtained by DAT algorithm are close to MaxTP while mean fairness indexes are close to RR. For example, compared to RR, when using infinite buffer, DAT improves the efficiency (e.g., response time) by 50%yet only degrades the fairness by 18% . The corresponding scale rates in terms of response time and fairness index are both more than 0.5. Especially, when we have a finite AP buffer, all performance metrics (i.e., efficiency, fairness and drop ratio) under DAT are close to the best ones, where drop ratio is defined as the percentage of packets which are dropped (or rejected) when the AP buffer is full. The optimal weight configurations may change under different traces. We only show the results under default weights configuration, i.e., $w1 = 1, w2 = 2$ in the following experiments due to the lack of space. As shown in the results, DAT can strike good balance between efficiency and fairness for most of the cases under the default setting.

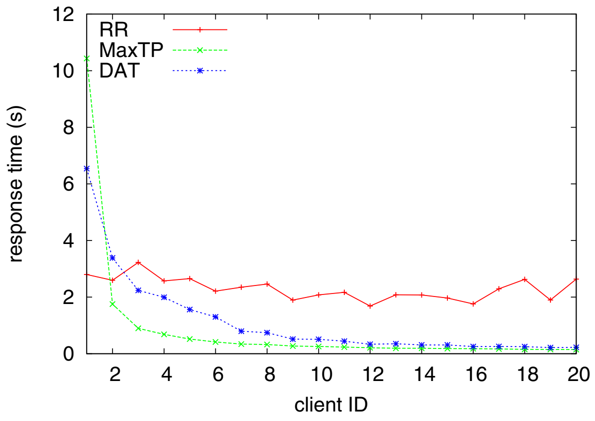

Fig. 2. Average response times of each client for RR, MaxTP, and DAT with $k = 10$ under the base case and infinite buffer size scenario.

Fig. 2 illustrates the average packet response time of each client under different polices. It is clear that all clients have similar performance despite of their link rates under RR policy. On the other hand, the MaxTP algorithm dramatically sacrifices the performance of the client with the lowest link rate, i.e., client 1. DAT balances the performance and fairness between clients by rendering performances related to client link rates.

To better understand how DAT dynamically adjusts the window size for different clients, Fig. 3(a) presents the selection of clients as well as window sizes when we have infinite buffer size and $k = 10$. The solid line represents the rotation among active clients (i.e., from c_1 to c_{20}) and the dashed line shows the time window size (i.e., i from 1 to 10) that DAT chooses for the corresponding client. We observe that during most time periods, DAT attempts to assign large window sizes to fast clients (i.e., with large client index) for improving the efficiency. Yet, after going through several such time periods, when DAT finds that the fairness is seriously degraded, it instead allocates short time windows to fast clients to improve the fairness. Fig. 3(b) further illustrates the performance comparison with respect to Jain's fairness index among the three scheduling policies. We observe that the RR policy achieves the best fairness such that its fairness index is always around 1.0. While, our DAT policy keeps the fairness between RR and MaxTP in most of the time.

To further study the distributions of response times and fairness indexes, we plot the cumulative distribution functions (CDFs) of these two performance metrics in Fig. 4.

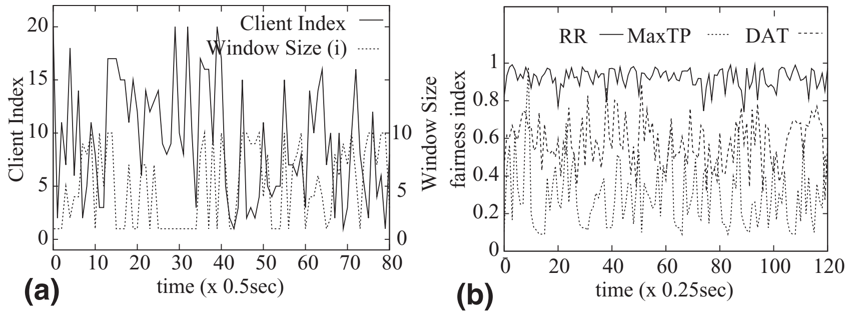

Fig. 3. Illustrating (a) the chosen client index and the corresponding time window size across time, and (b) the fairness index across time for RR, MaxTP, and DAT, where $k = 10$ and buffer size is infinite under base case.

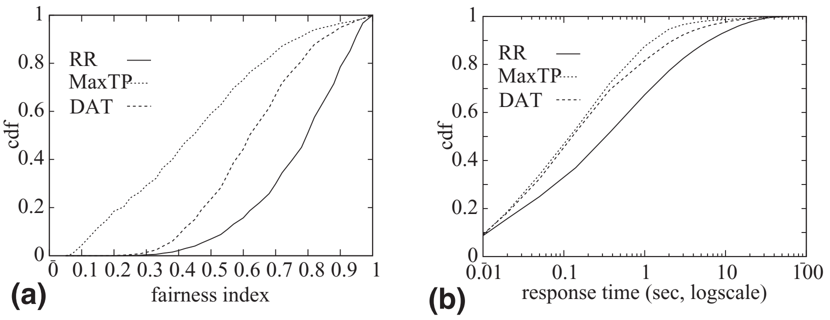

Fig. 4. Illustrating (a) CDFs of fairness index, and (b) CDFs of response time for RR, MaxTP, and DAT, where $k = 10$ and buffer size is infinite under base case.

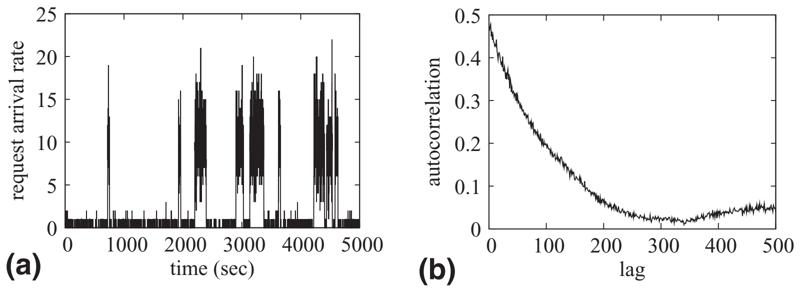

Fig. 5. Illustrating (a) the request arrival trace and (b) the autocorrelation of the trace of a bursty client.

According to the CDFs of fairness index and response time per packet, it is apparent that the DAT policy achieves the fairness close to RR and the efficiency close to MaxTP across all packets. The minimum fairness index under DAT is about 0.3, which is close to the minimum one under RR, while the minimum index in MaxTP is less than 0.1 indicating that most clients are starved during that time period, see Fig. 4(a). More importantly, about 70% of packets under DAT experience similar response times as those under MaxTP, see Fig. 4(b).

5.2.2. *Bursty case*

In our burst case workload, the mean request arrival rate of each client is the same as in the base case. Yet, a bursty access pattern is introduced into the arrivals of two clients with fastest link rates, i.e., client 19 and 20, such that the SCV of their inter-arrival times is equal to 20 and the autocorrelation function (ACF) at lag 1 is equal to 0.47. Therefore, high variability and strong burstiness are injected to the workload of these two clients. Fig. 5 illustrates the arrival rates (i.e., the number of arrivals per second) and the ACF at different lags of a bursty client.

Table 2
Performance under bursty case, numbers in parentheses are scale rates.

Scenario	Metrics	Policies		
		RR	DAT	MaxTP
InfiniteBuffer	RespTime(s)	5.921	4.594 (0.27)	1.052
	RespTime*(s)	1.490	0.849 (1.00)	1.152
	FairIndex	0.759	0.662 (0.72)	0.417
FiniteBuffer	RespTime(s)	0.737	0.623 (0.95)	0.617
	RespTime*(s)	0.658	0.538 (1.00)	0.662
	FairIndex	0.750	0.675 (0.69)	0.507
	DropRatio(%)	3.329	2.866 (0.28)	1.682

As shown in Table 2, all three policies encounter significant performance degradation in terms of response time and drop ratio (when the AP buffer is finite). The efficiency of DAT seems not as good as we observed in the base case. The rela-

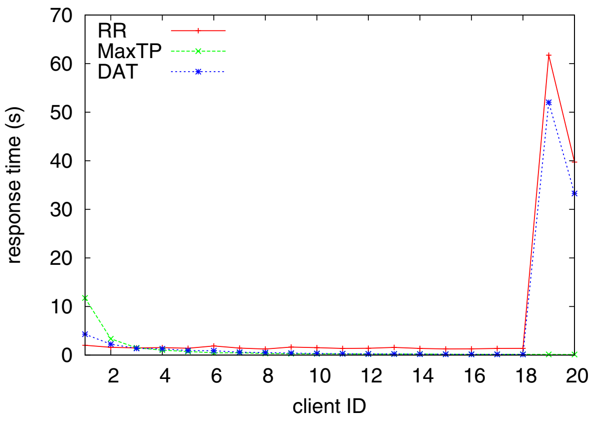

Fig. 6. Average response times of each client for RR, MaxTP, and DAT with $k = 10$ under the bursty case and infinite buffer size scenario.

tive improvement of average response time is only 22% compared to RR. However, if we examine the average response times of those clients without bursty patterns, i.e., client 1–18, then the average response times of both RR and DAT decrease while the average response time of MaxTP on the contrary increases, see the numbers of *Resp** in Table 2. Consequently, DAT becomes the most efficient policy for those non-bursty clients.

Since the bursty patterns are injected into the arrivals of the top two fastest clients, MaxTP achieves good efficiency by giving them high priority, but on the other hand, sacrifices the performance of other clients, resulting in extremely serious unfairness. Fig. 6 further demonstrates such unfairness under MaxTP where it always degrades the performance of clients which have slow link rates. On the other hand, RR and DAT punish only the clients that are responsible for the network congestion, i.e., clients with bursty workloads, and leave other clients unaffected.

Another interesting point in this case is that the mean fairness index value of MaxTP is counter-intuitively better than that in the base case, increasing from 0.369 to 0.507 when the AP buffer is finite, see Tables 1 and 2. In fact, this is caused by the property of Jain's fairness index, which is sensitive to the number of active clients. For example, considering there are now only two active clients, even under the extreme scenario where one client is starved during the whole period, the index value will be equal to 0.5 which is still relatively good. In the bursty case, the AP also experiences more idle periods where the number of active clients is small. In such an idle time period, the difference of fairness index values among the three policies is thus reduced.

5.2.3. Uneven case

Now, we turn to investigate different request arrival rates. In order to keep the overall request arrival rate similar as the previous two cases, we scale the arrival rates of selected clients (e.g., 19, 20) to 6 s^{-1} and decrease the rates of other clients (e.g., 1–18) to 1.2 s^{-1}. In addition, the SCV of all request arrival traces is equal to 5 and no clients have bursty patterns in their arrival flows. The results shown in Table 3 validate that our DAT policy works well in this case, consistently achieving a good balance between efficiency and fairness.

Table 3
Performance under uneven case, numbers in parentheses are scale rates.

Scenario	Metrics	Policies		
		RR	DAT	MaxTP
InfiniteBuffer	RespTime(s)	1.118	0.500 (0.74)	0.287
	FairIndex	0.729	0.625 (0.58)	0.480
FiniteBuffer	RespTime(s)	0.476	0.360 (0.62)	0.289
	FairIndex	0.725	0.650 (0.61)	0.535
	DropRatio(%)	1.091	0.425 (0.68)	0.108

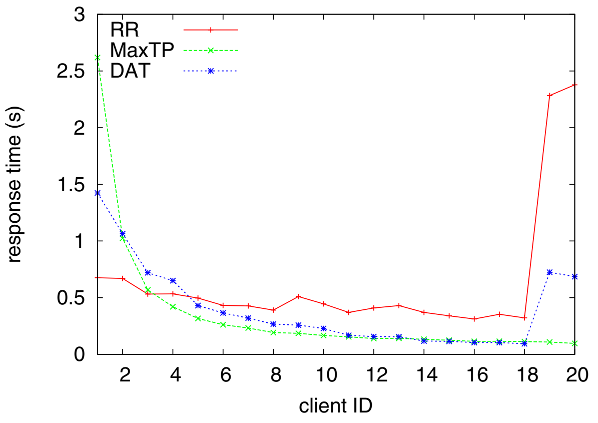

Fig. 7. Average response times of each client for RR, MaxTP, and DAT with $k = 10$ under the uneven case and infinite buffer size scenario.

Fig. 7 further presents the average response time of each client's reply packets. Recall that in our evaluation setting, the link rate linearly increases as the client index increases and clients with larger index always have faster link rates. Thus, we can clearly observe that MaxTP always degrades the performance of one or two clients with slowest link rates. While under RR, clients which are responsible for the buffer congestion suffer significant performance degradation and the other clients have almost the same response times despite of their varying link rates. We also observe that our DAT policy always punishes the clients that cause the buffer congestion in order to improve the efficiency of other clients. On the other hand, DAT strikes to give clients with faster link rates better performance, which fortunately is not too aggressive to degrade the fairness as MaxTP does.

5.3. Sensitivity analysis

Till now, only fixed configurations of the network, i.e., buffer size, population, and branch probability p were tested in the experiments, see Section 5.2. To evaluate the robustness of our algorithm, we investigate the robustness of DAT's performance to a variety of different experiment parameters.

5.3.1. Sensitivity to buffer size

We first focus on investigating the impact of different AP buffer sizes on various AP scheduling algorithms. Recall that both infinite and finite buffer size situations were considered in the previous simulations. Yet, due to physical space limitations in real WLANs, the AP always has a finite amount of

Table 4
Sensitivity analysis to buffer size, client population and branch probability under the base case, where the numbers in parentheses are the relative scale rates over RR and MaxTP.

	Buffer size								
	200			400			800		
	RR	DAT	MaxTP	RR	DAT	MaxTP	RR	DAT	MaxTP
RespTime	0.303	0.277(0.62)	0.261	0.530	0.470(0.60)	0.430	0.866	0.714(0.69)	0.647
FairIndex	0.701	0.662(0.71)	0.540	0.723	0.665(0.74)	0.501	0.748	0.640(0.70)	0.369
DropRatio	7.351	5.312(0.98)	5.265	3.789	2.256(0.98)	2.222	2.010	1.012(0.81)	0.773
	Number of clients								
	20			40			80		
	RR	DAT	MaxTP	RR	DAT	MaxTP	RR	DAT	MaxTP
RespTime	0.530	0.470(0.60)	0.430	0.587	0.534(0.54)	0.488	0.478	0.413(0.63)	0.375
FairIndex	0.723	0.665(0.74)	0.501	0.724	0.644(0.61)	0.520	0.710	0.623(0.68)	0.435
DropRatio	3.789	2.256(0.98)	2.222	1.664	1.258(0.91)	1.220	0.705	0.452(0.88)	0.418
	Branch probability								
	1.0			0.8			0.6		
	RR	DAT	MaxTP	RR	DAT	MaxTP	RR	DAT	MaxTP
RespTime	0.530	0.470(0.60)	0.430	0.734	0.683(0.57)	0.645	0.938	0.905(0.83)	0.898
FairIndex	0.723	0.665(0.74)	0.501	0.747	0.684(0.80)	0.426	0.785	0.699(0.77)	0.406
DropRatio	3.789	2.256(0.98)	2.222	8.522	6.561(0.74)	5.874	15.70	14.16(0.59)	13.10

buffer. It is important to take into account finite buffer capacity as well as the resultant packet drop ratio in the scheduler performance evaluation. Thus, we conduct experiments with various AP buffer sizes but keep the other experiment parameters the same as in the base case. The performance metrics considered here include the average response time, the fairness index, and the packet drop ratio due to the AP finite capacity under three AP scheduling algorithms.

Table 4 shows the simulation results when the AP's maximum capacity is set as 200, 400, and 800, see the "Buffer Size" part. The relative scale rates over RR and MaxTP are also shown in the parentheses in the table. We first observe that under all the three policies the performance in terms of response time improves as the AP buffer size decreases. Yet, such a performance improvement incurs at the cost of increasing packet drop ratio at the AP buffer. Meanwhile, we observe that the performance in terms of fairness index under RR and DAT is quite insensitive to various AP buffer sizes while the MaxTP policy experiences a significant degradation in fairness when the AP buffer size increases. We integrate that when the AP has a larger buffer, more packets from different clients are waiting in the AP buffer. As MaxTP always gives high priority to the faster clients, it becomes highly likely that clients with lower link rates are starved and the fairness consequently becomes worse under the MaxTP policy. More importantly, these results further verify that our DAT policy successfully achieves a good balance between fairness and efficiency under different buffer size conditions such that DAT's performance is close to RR in terms of fairness and to MaxTP with respect to response time and drop ratio.

5.3.2. *Sensitivity to client population*

Now we investigate the sensitivity of DAT to an increased number of clients in a crowded WLAN. This is extremely important to understand the performance benefit of the new technique as the WLAN becomes crowded under a large client population. So, we conduct experiments with three different network populations, i.e., $n = 20$, $n = 40$ and $n = 80$, while keeping fixed the other parameters as the experiment in the base case. In particular, in order to fix the same system load in all experiments, we scale each client's request arrival rate and set the link rates of all clients within the same range. Therefore, the system utilization (i.e., the buffer busy period over the whole time period) is kept around 75% under different experiment configurations. In addition, the AP buffer size is finite with the maximum capacity equal to 400.

The "Number of Clients" part in Table 4 shows the relative scale rates of DAT compared to the RR and the MaxTP policies when we have different numbers of clients in the WLAN. Obviously, the relative scale rates with related to all the three performance metrics (e.g., response time, fairness index and drop ratio) are consistently more than 0.5. These results indicate that our DAT policy always performs closely to the best one and thus well balances the trade-off between efficiency and fairness under different population situations.

5.3.3. *Sensitivity to branch probability*

In real wireless networks, a transmitted reply packet might trigger one or several client requests after a short think time. We capture this behavior in our simulation model by a branch probability p for reply packets at Q_1, see Fig. 1. With probability $1 - p$, a batch of client requests (≥ 1) are sent back to the channel queue. In all the previous experiments, we set $p = 1$, i.e., no client requests are triggered upon a reply packet transmission. We now evaluate a network with various branch probabilities while keeping all the other experiment parameters the same as the base case. In addition, the AP buffer size is set to 400 and the client population is $n = 20$.

Table 4 illustrates the simulation results under the three scheduling policies when the branch probability is $p = 1.0$, $p = 0.8$ and $p = 0.6$, see the "Branch Probability" part. As p decreases, it becomes more likely for a transmitted reply packet to trigger client requests and thus inject additional interactive traffic load into the network. As a result,

Table 5
Computation cost of our DAT algorithm with varying n (number of clients) and k (number of candidate window size).

	$k = 5$	$k = 10$	$k = 20$
$n = 5$	1.438 ms	2.546 ms	4.804 ms
$n = 10$	1.646 ms	2.732 ms	5.014 ms
$n = 20$	2.070 ms	3.194 ms	5.470 ms

non-negligible performance degradations of response time and drop ratio are observed under the three policies. While, our DAT policy consistently achieves the performance closer to the best one, i.e., the relative scale rate is greater than 0.5, across different branch probabilities.

5.4. CPU computation

Finally, we evaluate the computation cost of our proposed scheme. We implement the core algorithm of DAT in a popular commercial wireless router, Linksys WRT54GL. The router is equipped with a 200 MHz CPU, 16 Mb RAM, and running DD-WRT firmware [28]. Our codes are written in C language and cross compiled by the DD-WRT toolchains for MIPS architecture. The resulting binary code is 7.6K bytes. We execute the algorithm with a varying number of clients (n) and candidate window sizes (k). The actual computational costs (in *ms*) are shown in Table 5.

First, we observe that the computation cost is almost proportional to both n and k. However, k appears to be a more dominating factor because it determines the number of iterations in the outer loop in our algorithm (see Algorithm 1). Second, in all the tested cases, the computation cost of our algorithm is a small overhead for the router compared to the window size allocated to each client, e.g., with our default setting of $k = 10$ and $n = 20$, the execution time is less than 3.2 ms. Therefore, our algorithm is certainly feasible for real implementation and deployment.

6. Conclusion and future work

When WLANs become more and more popular, they are often crowded and the clients experience significant performance degradation. This paper attempts to solve the issue by designing a new packet scheduling algorithm on the heavily-loaded APs. Our goal is to improve the efficiency, i.e., reduce the packet response time and increase the throughput, and the fairness, i.e., avoid starvation especially for clients with poor link quality. We proposed a new scheduling algorithm DAT where the AP applies the basic Round-Robin scheme to select a client for service, but each client is allocated a different time window. We show that by dynamically adjusting the window size, DAT can achieve the balance between the efficiency and the fairness. The extensive experimentation carried out in this paper has revealed that the proposed algorithm significantly improves the performance in terms of efficiency and fairness in a crowded WLAN. Sensitivity analysis to AP buffer size, number of clients, branch probability for triggering new requests further proved that the gains of DAT are visible in a variety of different conditions.

Our future work mainly includes two directions. First, we would like to implement the scheduling algorithm on commercial wireless routers and conduct experiments for evaluation. Second, we plan to explore a new algorithm as well as a new model in a setting of multiple APs with possible coordination.

Appendix A

In this paper, we use Markovian Arrival Process (MAP) to express the arrival process to an AP in WLAN. MAPs, introduced by Netus [29], can easily model general distributions and non-renewable features such as autocorrelation of the stochastic process.

As a special case of MAP, a 2-state Markovian-Modulated Poisson Process (MMPP) is formally described by two 2×2 matrices, i.e., $\mathbf{D}_0$ and $\mathbf{D}_1$. Matrix $\mathbf{D}_1$ captures all transitions that are associated with real events in the MAP while matrix $\mathbf{D}_0$ only captures the transitions between states without signifying any real events. All off-diagonal entries of $\mathbf{D}_0$ and all entries of $\mathbf{D}_1$ are non-negative.

As an example, Eqs. (9) and (10) describe the request arrival processes with mean rate of $1.5\,\mathrm{s}^{-1}$. In particular, the MMPP(2) shown in Eq. (9) represents an autocorrelated stochastic process used in the bursty case while the MMPP(2) shown in Eq. (10) represents an independent stochastic process used in the base case.

$$D_0^{(S)} = \begin{bmatrix} -10.00744527 & 0.007445268262 \\ 0.001207011869 & -0.1232041617 \end{bmatrix},$$

$$D_1^{(S)} = \begin{bmatrix} 10 & 0 \\ 0 & 0.1219971498 \end{bmatrix}. \tag{9}$$

$$D_0^{(S)} = \begin{bmatrix} -13.61250000 & 3.612500000 \\ 0.6375000000 & -0.637500000 \end{bmatrix},$$

$$D_1^{(S)} = \begin{bmatrix} 10 & 0 \\ 0 & 0 \end{bmatrix}. \tag{10}$$

References

[1] S. Pilosof, R. Ramjee, D. Raz, R. Ramjee, Y. Shavitt, P. Sinha, Understanding TCP fairness over wireless LAN, in: Proceedings of the IEEE INFOCOM, 2003.
[2] M. Bottigleliengo, C. Casetti, C.-F. Chiasserini, M. Meo, Short-term fairness for TCP flows in 802.11b WLANs, in: Proceedings of the INFOCOM, 2004.
[3] G. Urvoy Keller, A.-L. Beylot, Improving flow level fairness and interactivity in WLANs using size-based scheduling policies, in: Proceedings of the MSWiM'08, 2008.
[4] P. Bhagwat, P. Bhattacharya, A. Krishna, S.K. Tripathit, Enhancing throughput over wireless LANs using channel state dependent packet scheduling, in: Proceedings of the INFOCOM'96, 1996.
[5] P. Bhagwat, P. Bhattacharya, A. Krishma, S.K. Tripathi, Using channel state dependent packet scheduling to improve tcp throughput over wireless lans, Wirel. Netw. 3 (March 1997) 91–102.
[6] A. Balachandran, P. Bahl, G.M. Voelker, Hot-Spot congestion relief in public-area wireless networks, in: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, in: WMCSA '02, 2002, p. 70.
[7] Y. Bejerano, S.-J. Han, L.E. Li, Fairness and load balancing in wireless LANs using association control, in: Proceedings of the MobiCom'04, 2004.
[8] N. Ahmed, S. Keshav, SMARTA: a self-managing architecture for thin access points, in: Proceedings of the CoNext' 06, 2006.

[9] A.P. Jardosh, K. Mittal, K.N. Ramachandran, E.M. Belding, K.C. Almeroth, IQU: practical queue-based user association management for WLANs, in: Proceedings of the MobiCom' 06, 2006, pp. 158–169.
[10] H. Lee, S. Kim, O. Lee, S. Choi, S.-J. Lee, Available bandwidth-based association in IEEE 802.11 wireless LANs, in: Proceedings of the MSWiM '08, 2008, pp. 132–139.
[11] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, D. Towsley, Facilitating access point selection in IEEE 802.11 wireless networks, in: Proceedings of the IMC '05, 2005.
[12] S. Manitpornsut, B. Landfeldt, A. Boukerche, Efficient channel assignment algorithms for infrastructure WLANs under dense deployment, in: Proceedings of the MSWiM '09, 2009, pp. 329–337.
[13] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, S. Ganguly, Distributed channel management in uncoordinated wireless environments, in: Proceedings of the MobiCom '06, 2006, pp. 170–181.
[14] G.K.W. Wong, X. Jia, An efficient scheduling scheme for hybrid tdma and sdma systems with smart antennas in wlans., Wireless Netw. 19 (2) (2013) 259–271.
[15] B.P. Tewari, S.C. Ghosh, A combined frequency assignment and ap scheduling for throughput maximization in ieee 802.11 wlan, in: Proceedings of International Conference on Advances in Mobile Computing and Multimedia, in: MoMM '13, 2013, pp. 133:133–133:142.
[16] A. Enayet, N. Mehajabin, M.A. Razzaque, C.S. Hong, A power-aware distributed wi-fi access point scheduling algorithm, in: Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, in: ICUIMC '13, 2013, pp. 41:1–41:8.
[17] M. Anand, E.B. Nightingale, J. Flinn, Self-tuning wireless network power management, Wirel. Netw. 11 (4) (Jul. 2005) 451–469.
[18] E. Rozner, V. Navda, R. Ramjee, S. Rayanchu, Napman: Network-assisted power management for wifi devices, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, in: MobiSys '10, 2010, pp. 91–106.
[19] F.R. Dogar, P. Steenkiste, K. Papagiannaki, Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, in: MobiSys '10, 2010, pp. 107–122.
[20] J. Manweiler, R. Roy Choudhury, Avoiding the rush hours: wifi energy management via traffic isolation, IEEE Trans. Mobile Comput. 11 (5) (May 2012) 739–752.
[21] B. Sadeghi, V. Kanodia, A. Sabharwal, E. Knightly, Opportunistic media access for multirate ad hoc networks, in: Proceedings of the MobiCom '02, 2002, pp. 24–35.
[22] G. Tan, J. Guttag, Time-based fairness improves performance in multirate WLANs, Proceedings of the AnnualConference on USENIX Annual Technical Conference, 2004.
[23] 802.11e amendment, [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1541572.
[24] R. Jain, D.-M. Chiu, W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, DEC Research Report TR-301 (1984).
[25] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, SIAM, Philadelphia PA, 1999. ASA-SIAM Series on Statistics and Applied Probability
[26] Sigcomm 2008 trace, [Online]. Available: http://www.cs.umd.edu/807projects/wifidelity/sigcomm08_traces/.
[27] V. Paxson, S. Floyd, Wide-area traffic: the failure of poisson modeling, IEEE/ACM Trans. Netw. 3 (3) (1995) 226–244.
[28] DD-WRT firmware, [Online]. Available: http://www.dd-wrt.com.
[29] M.F. Neuts, Structured Stochastic Matrices of M/G/1 Type and Their Applications, Marcel Dekker, New York, 1989.

Yi Yao is a Ph.D. student at Northeastern University, Department of Electrical and Computer Engineering, Boston, MA. He received his M.S. and B.S. in Computer Science from the Southeast University, China, in 2010 and 2007. His current research interests are scheduling, resource management, and cluster computing.

Bo Sheng is an assistant professor in the Department of Computer Science at University of Massachusetts Boston. He received his Ph.D. in computer science from the College of William and Mary in 2010. His research interests include mobile computing, wireless networks, security and cloud computing.

Ningfang Mi is an Assistant Professor at Northeastern University, Department of Electrical and Computer Engineering, Boston, MA. She received her Ph.D. degree in Computer Science from the College of William and Mary, VA in 2009. She received her M.S. in Computer Science from the University of Texas at Dallas, TX in 2004 and her B.S. in Computer Science from Nanjing University, China, in 2000. Her current research interests are performance evaluation, capacity planning, resource management, simulation, data center and cloud computing.