

# Application-Aware Service Differentiation in PAWNs

Hanping Lufei, Sivakumar Sellamuthu, Sharun Santhosh, and Weisong Shi

Department of Computer Science  
Wayne State University  
{hlufei, siva, sharun, weisong}@wayne.edu

## Abstract

*We have witnessed the increasing demand for pervasive Internet access from public area wireless networks (PAWNs). The diverse service requirements from end users necessitate an efficient service differentiation mechanism, which should satisfy two goals: end-user fairness and maximizing the utilization of wireless link. However, we found that the existing best-effort based service model is not enough to satisfy either goal. In this paper, we have proposed an application-aware service differentiation mechanism which takes both application semantics and user requirements into consideration. The results show that our proposed method outperforms two other bandwidth allocation approaches, best effort and static allocation, in terms of both client fairness and wireless link bandwidth utilization, especially in heavy load environments.*

## 1 Introduction

As the rapid growth of the deployment of public area wireless networks (PAWNs) [2], the *diverse service requirements of end users* become an important issue that need to be addressed. A *service differentiation and access control* mechanism is necessary in such an environment to ensure genuine users receive the services they have paid for, to protect them from malicious users, and to efficiently use the network bandwidth resources.

Several access control or bandwidth allocation algorithms have been proposed. RSVP [11] is a signaling protocol to reserve resources at all the routers along the path. Several measurement-based admission control algorithms [8] can provide a soft guarantee. Although these previous results are promising, few of them take the application level information into consideration. The result is either lower link utilization or unfair service differentiation. Therefore, we envision that an efficient service differentiation mechanism should satisfy two goals: *end user fairness*

and *maximization of wireless link utilization*.

To achieve both goals, in our proposed approach *Application-Aware Service Differentiation (AASD)*, bandwidth is not allocated to clients statically based on their reserved bandwidth only. Instead, application level information, such as requested object size, available Web server bandwidth, and user reserved bandwidth are taken into consideration comprehensively. Even during the course of processing requests, bandwidth adjustment may be made in accordance with variations in measured Internet path bandwidth. We felt that by adaptively varying the bandwidth allocated to users in the last-mile wireless hop, the network can accommodate more users while at the same time increase the likelihood of admitting users at pre-negotiated service levels. All our results validate this intuition.

The service differentiation algorithm was implemented and evaluated under two different scenarios: *fixed upstream bandwidth* and *dynamic upstream bandwidth*. The results show that in comparison with two other bandwidth allocation approaches, best effort and static access control algorithms, our proposed method outperforms them in terms of both client fairness and wireless bandwidth utilization. For example, when the access point hosts 120 clients with different reserved bandwidths that follow a normal distribution, the utilization of our approach is twice of that of static allocation method and 50% more than best effort approach while at the same time satisfies 90% of clients with 95% of their individual reserved bandwidth. Furthermore, we proposed an exponential weighted queue optimization technique to adapt bandwidth allocation dynamically to bandwidth variations between access points and web servers.

The rest of the paper is organized as follows. Two performance metrics related to service differentiation algorithm are described in Section 2.1. Section 2.2 presents two other access control algorithms for comparison purpose. The design details of the AASD algorithm is presented in Section 3. Section 4 reports the details of performance evaluation, including experimental platforms, performance analysis, and implications. Related work and conclusion remarks

are listed in Section 5 and Section 6 respectively.

## 2 Background

A common access scenario in public-aware wireless network includes a set of wireless users sharing one wireless access point. The wireless access point in turn connects to the Internet through wired connection. There are two important metrics to be concerned. First, The investor cares about how many percentage of the wireless resource can be used in order to get the maximal profit. Second, for the end users the matters is the reserved bandwidth they paid can be satisfied to the most extend.

### 2.1 Performance Metrics

**Fairness** — Fairness tries to guarantee that each user should get the bandwidth he deserved no more and no less. Fairness is a relative concept. Assume user A sends out  $n$  connections,  $bw_i$  is the bandwidth of connection  $i$ ,  $BW_{reserv}^A$  is the reserved bandwidth of user A. We define the fairness as following:

$$\text{Fairness of A} = \frac{\sum_{i=1}^n bw_i}{n \cdot BW_{reserv}^A} \quad (1)$$

**Utilization** — Utilization shows the usage of wireless resources. For a given number of users, say  $n$  users, in time interval  $T_{interval}$ , user  $i$  receives  $size_i$  bytes in total and the specific wireless link bandwidth of the access point is  $BW_{ap}$ , we can calculate the utilization by the following equation:

$$\text{Utilization} = \frac{\sum_{i=1}^n size_i}{T_{interval} \cdot BW_{ap}}. \quad (2)$$

### 2.2 Non-Application Aware Methods

For comparing purpose, we also implement two other service differentiation algorithms: *best effort* and *static allocation*:

1. *Best Effort with no Reservation*: Strictly speaking there is no algorithm. When the access control program receives a request from the client, it just sets up a connection to the destination, then sends back the response to the client. None of the bandwidth allocation algorithms are applied. With increasing number of clients the wireless link resource will finally get saturated and the throughput of each flow will become too small to be tolerated by the users.
2. *Static Access Control with User Reservation*: The main principle of this algorithm is try to statically allocate

the total bandwidth to the clients. In this model an access point serves  $n$  client each associated with a reserved bandwidth. Let  $B_i$  denotes the reserved bandwidth associated with client  $i$  ( $i \in [1, N]$ ), where  $N$  is the maximum number of clients. Let  $B_{total}$  be the total wireless bandwidth. In order to satisfy each client total wireless bandwidth is allocated statically to the incoming clients. When the total upstream bandwidth is used up the new incoming client will not get served. Thus the following observation holds.

$$\sum_{i=1}^{n+1} B_i \geq BW_{total} \geq \sum_{i=1}^n B_i, \quad (3)$$

In this approach, the fairness can be satisfied maximally but the utilization is badly affected. Once the bandwidth has been statically assigned to a client, no matter what he does this part can not be used by anyone else even if the client is idle all the time.

## 3 Application-Aware Service Differentiation

Based on the observations of the two previous access control algorithms, we propose an application aware access control with adaptive allocation algorithm. First let us start our discussion with the behavior of wireless users.

### 3.1 User Behavior

Our proposed algorithm is inspired by previous studies on the mobile user behavior of Web surfing. Balachandran *et al.*'s recent study found that major part of the mobile user traffic is HTTP traffic [4]. At the same time, study from Barford and Crovella [5] found that for web surfing not much time is spent on data transfer, on the contrary most of the time the user is idle and this is refereed to as user 'think' time. Borrowed the term from [5], a user equivalent (UE) is defined as a single process in an endless loop that alternates between making requests for online content, and lying idle. Each UE is therefore an ON/OFF process; Statistical models show Active OFF Times follow the Weibull distribution and Inactive OFF Times follow the Pareto distribution. For a given time  $\tau = 10$ ,  $\alpha = 1.5$  and  $k = 1$  in Pareto distribution.

$$\int_1^{\tau} \alpha k x^{-\alpha+1} dx = 1 - \tau^{-1.5} = 99.68\% \quad (4)$$

It means 99.68% of users have 1 to 10 average seconds idle time between their clicks, this is users' 'think' time. In the thinking time period the bandwidth allocated to the user is wasted. In order to utilize the wasted bandwidth the application-aware service differentiation algorithm is proposed.

## 3.2 Design

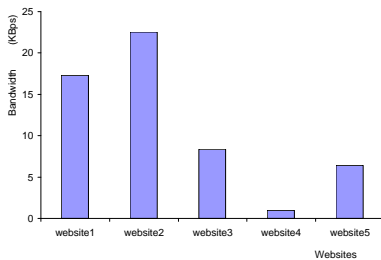
The goal of a service differentiation algorithm would be to efficiently allocate wireless bandwidth to the users, and at the same time to satisfy both the fairness and the utilization metrics.

### 3.2.1 Overview

Our design differs from existing resource allocation and access control algorithms in the following three ways: First, *application-awareness*, application level demands, such as object size, Web server bandwidth etc., are taken into consideration. An acceptance decision is not based only on the user reserved bandwidth but also on the application specific parameters predicted by our application-aware algorithm. Second, *dynamic server bandwidth adaptation*, our algorithm monitors (passively) the current Internet path bandwidth (of different servers) and take bandwidth history for each Web server into consideration to get a more stable and predictable bandwidth value for the next connection. Third, *discrete service differentiation*, the AASD algorithm distinguishes different levels of Web server bandwidth service queues. This method improves the utilization of the total bandwidth dramatically. Theoretically the more differentiation service queues we have the better is the performance.

### 3.2.2 Multiple Service Queues

Usually, the available bandwidth between an access point and a fixed Web server is dynamically changing. Thus, using one queue for multiple clients and multiple servers probably can not fill up the total wireless upstream bandwidth, given the fact of the head of queue blocking effect [13]. Hence, in AASD, we divide total upstream bandwidth into discrete service queues based on different individual bandwidth limits, as shown in Figure 2. However, how to allocate servers into these service queues is a problem.



**Figure 1. The top five Web server bandwidth accessed by a medium-size research institution.**

To address this problem, we examined the top five Web servers accessed by the users from a medium-size research institution. Figure 1 shows a snapshot of the available bandwidth of these Web sites. We found that the following observation can be hold: basically, the ratio of bandwidth between Web servers follows exponential distribution [16], and this relationship can be used to decide the allocation of multiple service queues. We call these queues as exponential weighted queues (EWQs). Suppose there are  $n$  queues the bandwidth ratio of these queues would be

$$1 : 2 : 2^2 : 2^3 : \dots : 2^n$$

Another reason to chose EWQ is to take care of service downgrade dynamically. If the Internet bandwidth of an incoming request later turns out to be much lower than the bandwidth specified by this queue, GetURL module can move this request to an appropriate queue to avoid jamming the following requests in the original queue.

For each individual queue, it is different from the traditional first in first out (FIFO) queue. For any individual client, its request can be inserted into any position in the queue. There is a GETURL module that only takes out the head of queue and processes the request, so that at any instant moment there is only one request is being served for each queue. For each service queue, each entry includes three important elements:  $T_{delay}^i$  is the maximum delay time before serving this request.  $T_{download}^i$  is the estimated download time of the requested object.  $URL$  is the location of the object as shown in the Figure 2. The entries are increasingly ordered by  $T_{delay}^i$  so that the following formula holds.

$$T_{delay}^i \leq T_{delay}^{i+1}, i = 1, 2, \dots, n \quad (5)$$

Based on the statistical result of web object size distribution most of the web page frame is no more than 20KB [7], with high probability, the following formula holds.

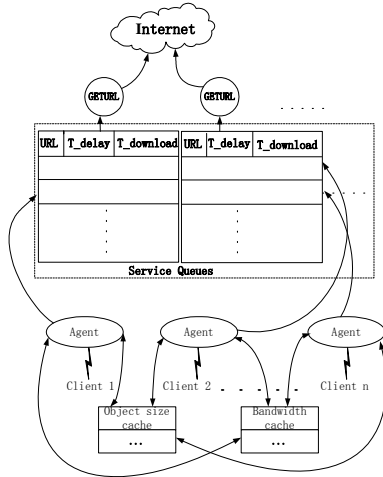
$$T_{delay}^i + T_{download}^i \leq T_{delay}^{i+1} \quad (6)$$

It means downloading object  $i$  probably will not delay serving  $(i + 1)^{th}$  request. Even if the inequation doesn't hold, the *GetURL* module will dynamically migrate the request to an appropriate lower bandwidth service queue.  $T_{delay}^i$  and  $T_{download}^i$  are calculated by the following two formulas.  $S_{obj}$  is the object size,  $B_{user}$  is the user bandwidth and  $B_{svr}$  is the Web server bandwidth.

$$T_{delay}^i = \frac{S_{obj}^i}{B_{user}} - T_{download}^i, \quad (7)$$

$$T_{download}^i = \frac{S_{obj}^i}{B_{svr}}, \quad (8)$$

Simply speaking, as long as the queues are not empty the wireless link will be fully utilized. So our application aware algorithm combines multiple loosely loaded user sessions into one (maybe more) tightly heavy loaded service queue to improve utilization and maintain fairness at the same time.



**Figure 2. The basic model of the application-aware service differentiation algorithm.**

### 3.2.3 Two Important Modules

Due to the space limit, we briefly explain two important modules here. More details can be found in the technical version of this paper [10]. The Agent module in Figure 2 handles each client request, checks the request object size and server bandwidth from the cache first, inserts the request in the appropriate position of the proper queue, sends back response and updates the object size and server bandwidth if necessary. For the object size cache studies show that the distribution of Web object size follow the Pareto distribution, most of the html files are below 50KB [7] respectively. The object size cache is maintained to store size information of objects that have already been handled by the agent. The agent searches for the size of the requested URL in the cache. If the size is not known the default size is used and the actual size will be updated to the cache after the *GetURL* module retrieves the object. Next time when the same URL is requested the accurate size value can be obtained.

The *GetURL* module is in charge of grabbing the head of queue, getting URL object, probing Internet bandwidth etc., as shown in Figure 2. At the beginning a default bandwidth value is used. Later on the Internet path bandwidth will be calculated from the latest bandwidth that each connection observed and the bandwidth value of the immediate

previous connection. The new server available bandwidth ( $B_{i+1}$ ) will be calculated as follows:

$$B_{i+1} = \frac{B_{latest} + B_i}{2}, \quad (9)$$

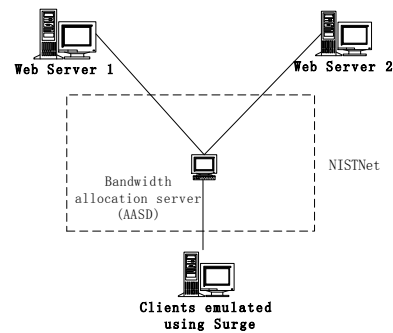
where  $B_{latest}$  represents the latest bandwidth measured between bandwidth allocation server and the specific Web server. This feedback based adaptive bandwidth measurement mechanism makes the curve of bandwidth more fluent and stable.

## 4 Performance Evaluation and Analysis

In this section, we first briefly describe the experimental platforms, then compare different algorithms in terms of fixed and dynamic Internet bandwidth. After that we will discuss effects of available server bandwidth and queue optimization on our algorithm.

### 4.1 Experimental Setup

We set up four machines and run web servers, allocation algorithms and clients on these machines separately, as shown in Figure 3, where the two Web servers are running on two SUN UltraSparc2 dual-processor running Solaris 5.8, bandwidth allocation server is running on Intel-based Linux box running RedHat Linux 8.0 (P4 2.2GHz with 512 MB SDRAM), clients are emulated on another Linux box with the same configuration. All these four machines are connected by a 100Mbps fast switched Ethernet. Generally our experimental platform consists of the following parts.



**Figure 3. Experimental setup.**

*Hardware Configuration* — As shown in Figure 3, we set up two Apache [1] web servers with different bandwidths connected to our bandwidth allocation server where our algorithms will be deployed. All the web traffic generated by clients will be processed by the bandwidth allocation server by applying the appropriate bandwidth allocation algorithm. We evaluate different service differentiation algorithms in the context of two scenarios: *fixed upstream*

*bandwidth and dynamic upstream bandwidth.* During the experiments, the fixed upstream bandwidth is set to 6Mbps, while the dynamic upstream bandwidth is set by using the available bandwidth extracted from the wireless LAN traces from ACM SIGCOMM'01 [4]. Both the fixed and dynamic upstream bandwidths are emulated by using NISTNet [12], a network emulation package developed by NIST.

*Simulation of User Behavior* — Surge [5] is a realistic Web workload generation tool which mimics a set of real users accessing a server. It generates references matching empirical measurements of server file size distribution, request size distribution, relative file popularity, embedded file references, temporal locality of reference and idle periods of individual users. We use Surge to generate the client requests. The number of clients we simulated using Surge is between 1 to 200.

*Distribution of User Reserved Bandwidth* In the real world, different users have different reserved bandwidths. So in our experiment we use two distributions, one is uniform distribution; all users have the same bandwidth set to 100Kbps. Another one is normal distribution; that most of the users have bandwidth set to 80Kbps or 120Kbps, while small part of users have 40Kbps or 160Kbps reserved bandwidth. All the results below are for normal distribution client bandwidth, for uniform distribution we get more or less similar results.

## 4.2 Fixed Upstream Bandwidth

First, we compare these three service differentiation algorithms in the context of ideal fixed upstream bandwidth case. We set the bandwidth between Web server one (WS1) and the bandwidth allocation server as half of the bandwidth between Web server two (WS2) and the bandwidth allocation server. We also regulate the total wireless bandwidth as 6Mbps, a reasonable approximation of a 802.11b [6] network.

Figure 4 shows the evaluation results of different algorithms in the fixed upstream bandwidth scenario, with the user reserved bandwidth follows a normal distribution. First, let's have a look at the number of accepted clients in Figure 4(a). Although the best effort algorithm will accept any incoming clients without any rejection, the individual user reserved bandwidth suffers. In the static access control algorithm, once the remaining wireless link bandwidth is used up by incoming clients, about 60 clients in our experiment, all new clients will be refused unless one of the previous clients exits from the network. For the AASD, the curve tells us that it serves more clients than the static method although it has to reject new comers when the bandwidth gets saturated.

For the utilization of the wireless link, as shown in Figure 4(b), in the static access control case, the utilization

value doesn't change to much since the number of accepted clients remains unchanged. But for the best effort and the application aware approaches, the utilization curves are both increasing in a similar fashion, which means the application aware access control does almost as good as the best effort in terms of utilization.

As far as the user bandwidth satisfaction is concerned, Figure 4(c) shows that average bandwidth deteriorates dramatically with the increase of client numbers. On the other hand, average bandwidth is abnormally high in light load scenario. In Figure 4 (d) the x axis is the percentage of satisfaction of user reserved bandwidth, defined as *Fairness* in equation 1, while the y is the cumulative number of clients. We can see that in the scenario with 60 clients active involving, the application aware approach performs as good as the static access control, as 90% of clients experience about 95% fairness. In the 160 clients scenario, where the number of clients increase 167%, AASD still can satisfy 60% of clients with 90% fairness. AASD has the potential to hold more users than static access control algorithm, while at the same time guarantees the reserved bandwidth.

Note that we also evaluated these algorithms in the context of other user reserved bandwidth distributions, such as the uniform distribution, and got similar results [10].

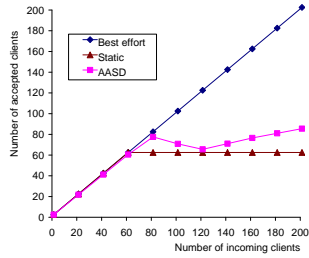
In summary, the best effort approach can reach the high utilization of wireless bandwidth but with poor user bandwidth satisfaction. On the contrary, the static access control has the good user bandwidth fairness but with the fair wireless link utilization. The proposed application aware approach takes both advantages so that it can maximally guarantee user reserved bandwidth and improve the utilization of wireless link bandwidth as efficiently as possible at the same time.

## 4.3 Dynamic Upstream Bandwidth

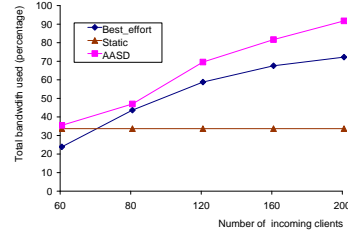
In addition to the fixed upstream scenario, we also evaluate the proposed algorithm under the circumstance of dynamic changing environment, which is collected from ACM SIGCOMM 2001 conference [4]. For simplicity, we calculate the average bandwidth ever 100 seconds, over a 10 minute duration. Totally we have six different upstream bandwidths, as shown in Figure 5.

From Figure 6 it is easy to see that the application aware approach has the potential to accept more clients than fixed upstream bandwidth scenario as shown in Figure 4(a). This is because in most of the time the bandwidth from the trace is greater than the bandwidth used in the fixed bandwidth scenario.

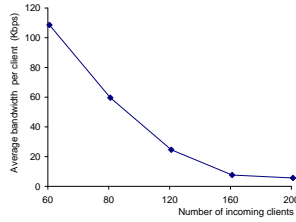
For the utilization of wireless link bandwidth, generally it is less than the fixed Internet bandwidth scenario because most of the time the Internet bandwidth fluctuates below the value of fixed bandwidth. AASD is still able to follow the



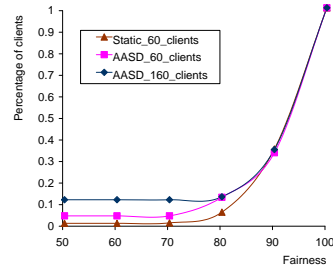
(a) The number of accepted clients



(b) The utilization of wireless link

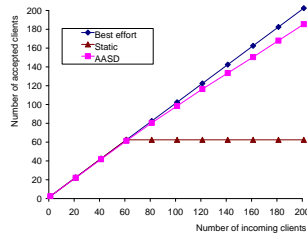


(c) Average user bandwidth (best effort)

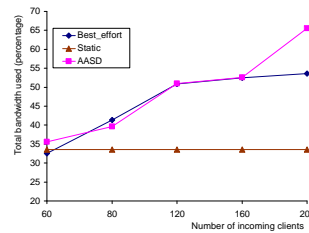


(d) The CDF of fairness

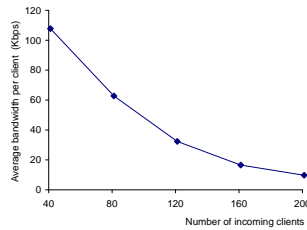
**Figure 4. Evaluation results of different algorithms in the fixed upstream bandwidth scenario, with normal user reserved bandwidth distribution: (a) the number of accepted clients, (b) the utilization of wireless link, (c) the average user bandwidth of the best effort approach, and (d) the CDF of fairness.**



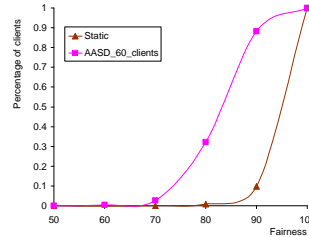
(a) The number of accepted clients



(b) The utilization of wireless link

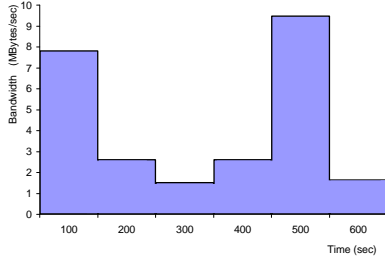


(c) Average user bandwidth (best effort)



(d) The CDF of fairness

**Figure 6. Evaluation results of different algorithms in the dynamic upstream bandwidth scenario, with normal user reserved bandwidth distribution: (a) the number of accepted clients, (b) the utilization of wireless link, (c) the average user bandwidth of the best effort approach, and (d) the CDF of fairness.**



**Figure 5. A discrete 10-minute Internet bandwidth trace from ACM SIGCOMM'01 [4].**

best effort curve closely as can be seen in Figure 6(b).

In terms of user bandwidth fairness, Figure 6(d) shows the application aware approach is a little bit worse than the static access control by about 10%. One of the reason is, if the Internet bandwidth changes fast our approach will exhibit a delay to accommodate the new Internet bandwidth. On the other hand if the Internet bandwidth drops out of the range of a service queue the request have to be transferred to another queue (see the design of AASD 3, which makes results in request thrashing between queues. These factors affect user reserved bandwidth satisfaction. But generally the 10% is in an acceptable range.

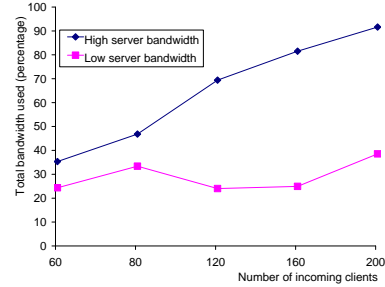
Again, we also evaluated these algorithms in the context of the uniform user reserved bandwidth distribution, and got similar results [10].

#### 4.4 Effect of Available Server Bandwidth

We also notice that available server bandwidth has great effect on the the utilization of wireless bandwidth, as shown in Figure 7. We compare utilization under two server bandwidth scenarios. The low server bandwidth scenario sets server bandwidth to 2Mbps and 4Mbps for each webserver. While the high server bandwidth scenario sets server bandwidth to 4Mbps and 8Mbps. No matter how hard the algorithm tries, the total utilization is still not high if the Internet server bandwidth is low. An increase in server bandwidth can dramatically improve the total utilization.

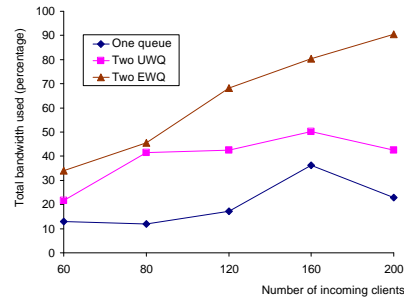
#### 4.5 Effect of Queue Optimization

As we discussed in Section 3, multiple queues, EWQs in our algorithm, are useful to improve the utilization of upstream stream bandwidth. To show the effect of our design, we set up an experiment which compares EWQ with two other queue algorithms. In our settings, we use  $n = 2$ , the two other queue algorithms are: *one queue only*, and *two equally distributed queues* (UWQ). We found the EWQ approaches outperforms other two queue algorithms in terms



**Figure 7. The utilization of wireless link for different server bandwidths.**

of the utilization of wireless bandwidth as shown in Figure 8. It is obvious that the performance is improved when two queues are used than that of one queue is used. However, it is worth noting that the performance is not proportional to the number of queues because more queues will definitely add more overhead to the system and in turn put penalty on the total performance. Based on the variation of Web server bandwidths we found two or three queues is the optimal choice in practice.



**Figure 8. The utilization of wireless link for different queue implementations.**

## 5 Related Work

Extensive research has been done on the subject of providing quality of service over a network. RSVP [11] is a signaling protocol that reserves resources on all the routers along the path. Though it guarantees quality of service, it exhibits significant scalability problems. Admission control in integrated service (IntServ) architecture: flows must request services from the network and are accepted or rejected depending on the availability of resources. Differentiated services (Diffserv) is another approach to provide QoS. It is based on DS field in the packet header, priority scheduling and buffering. Upon overload in a given service class all

flows in that class suffer a degradation of service. End point admission control [8] combines Diffserv superior scalability and IntServ superior quality of service. Measurement based admission control algorithms provide a soft guarantee. Most previous work on wireless QoS work has focused on the MAC layer [14]. All these approaches do not take application-level demands into consideration.

Vaidya, Bahl and Gupta [15] developed a fully distributed algorithm for scheduling packet transmissions such that different flows are allocated bandwidth in proportion of their weights. Qiu, Bahl and Adya [9] evaluated several first-hop allocation schemes using traces collected from a popular Web site. Their results show that the scheme which takes into account both the application data rate and available Internet path bandwidth yields the best performance. But they did not propose their own bandwidth allocation algorithm. Our evaluation results show that adapting to variations in server bandwidth plays an important role in service differentiation, as this scheme gets the best results.

In Balachandran *et al.*'s recent work [3], they proposed a hot-spot releasing approach that when a user requests service from the network in an overloaded region, the wireless network tries to adapt itself to handle the user service request by readjusting the load across its APs. This approach is different from ours: they deal with the load balance between multiple APs, for each AP no special method is proposed, while we try to make one AP serves more clients than usual and each client can still get the reserved bandwidth so they do not need to reluctantly move to other place to get the service. But we believe these two approaches compliment very well, and the combination of them will provide a more comprehensive solution for this kind of questions related to wireless resource utilization and service differentiation.

## 6 Summary and Future Work

This paper proposed an application-aware service differentiation mechanism in public-area wireless network, aiming to satisfy two goals *end user fairness* and *maximization of wireless link utilization*. The unique feature of our algorithm is both the application semantics, client requirements and the varying individual server bandwidths are taken into consideration. The performance evaluation shows that AASD outperforms other algorithms in terms of both performance metrics. Furthermore, an exponential weighted queue optimization technique applied to AASD allows it to adapt to constantly changing bandwidth environments. Our next step is to extend the AASD approach for other non-HTTP based application traffic, such as SSH, Telnet and email.

## References

- [1] Apache HTTP Server Project.
- [2] P. Bahl, W. Russell, Y. Wang, A. Balachandran, G. Voelker, and A. Miu. PAWNS: Satisfying the need for ubiquitous secure connectivity and location services. *IEEE Personal Communications Magazine*, 9(1):40–48, Feb. 2002.
- [3] A. Balachandran, P. Bahl, and G. Voelker. Hot-spot congestion relief and service guarantees in public-area wireless networks. In *Proceedings of WMCSA 2002*, June 2002.
- [4] A. Balachandran, G. Voelker, P. Bahl, and V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *Proceedings of ACM SIGMETRICS 2002*, June 2002.
- [5] P. Barford and M. E. Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of Performance '98/ACM SIGMETRICS '98*, July 1998.
- [6] IEEE. IEEE std 802.11 — wireless LAN medium access control (mac) and physical layer (phy) specifications, 1997.
- [7] B. Krishnamurthy and J. Rexford. *Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement*. Addison-Wesley, Inc, 2001.
- [8] L. Breslau, E.W. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *SIGCOMM*, pages 57–69, 2000.
- [9] L. Qiu, P. Bahl, and A. Adya. The effect of first-hop wireless bandwidth allocation on end-to-end network performance. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 85–93, 2002.
- [10] H. Lufei, S. Sellamuthu, S. Santhosh, and W. Shi. Application-aware service differentiation in pawns. Technical Report CS-MIST-TR-2003-009, Department of Computer Science, Wayne State University, Aug. 2003.
- [11] L. Zhang, S. Deering, and D. Estrin. RSVP: A new resource ReSerVation protocol. *IEEE network*, 7(5):8–?, September 1993.
- [12] Nist network emulation tool (nistnet).
- [13] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *ACM First Workshop on Hot Topics in Networks (HotNets-1)*, Oct. 2002.
- [14] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Mobile Computing and Networking*, pages 87–98, 2000.
- [15] N. H. Vaidya, P. Vahl, and S. Gupta. Distributed fair scheduling in a wireless lan. In *Proc. of the 6th ACM SIGMOBILE International Conference on Mobile Computing and Networking (MobiCom'00)*, Aug. 2000.
- [16] Z. Zhu, Y. Mao, and W. Shi. Workload characterization of uncacheable web content — and its implications for caching. Technical Report CS-MIST-TR-2003-003, Department of Computer Science, Wayne State University, May 2003.