Implementation and Demonstration of a Credit-based Home Access Point

Choong-Soo Lee
clee7@gustavus.edu
Department of Mathematics & Computer Science
Gustavus Adolphus College
800 West College Avenue, St. Peter, MN 56082, USA

Mark Claypool
claypool@cs.wpi.edu
Robert Kinicki
rek@cs.wpi.edu
Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609, USA

ABSTRACT

The increasing availability of high speed Internet access and the decreasing cost of wireless technologies has increased the number of devices in the home that wirelessly connect to the Internet. While home user applications often have different network requirements, the wireless access point (AP) typically gives them all the same treatment. It has been shown that applications that are sensitive to delay, such as VoIP, remote login and online games suffer degraded performance when running concurrently with applications that expand to fill the available capacity, such as file sharing and downloading. Unfortunately, there are few mechanisms available at the AP to mitigate these effects other than for users to explicitly classify traffic based on port numbers or host IP addresses. This work proposes a novel home access point called *CHAP* that features credit-based queue management designed to eliminate the need for explicit classification and configuration of per-application quality. CHAP is implemented as a Linux queuing discipline and compared with a traditional AP for Web browsing and online game activities. The comparisons demonstrate the merits of our approach.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication; Packet-switching networks*

General Terms

Design, Performance

1. INTRODUCTION

Wireless access points (APs) allow home users to share one Internet connection simultaneously and to network multiple devices at home. Devices such as desktop and laptop computers, gaming consoles, Voice-over-IP (VoIP) phones, video streaming servers, cell phones and PDAs run a wide variety of applications that connect to the Internet. Each application has its own Quality-of-Service (QoS) requirements, with some applications degrading the performance of others by increasing loss or latency or restricting bandwidth on the shared Internet link.

In particular, congestion queue build up along the network path results in an increase in latency for network applications. In a typical home Internet connection setup, there are three places where the queue is most likely to become congested: the ISP gateway, the broadband connection modem, or the wireless AP. Wireless APs offer capacities up to 54 Mbps (802.11g) or 70 Mbps (802.11n) to connected devices. The bandwidth between the wireless AP and the broadband modem is typically 100 Mbps to 1 Gbps. The bandwidth along the link between the broadband modem and the ISP gateway depends on the Internet subscription, with maximums differing from one country to another. However, ISPs in general are providing increasingly faster broadband access to homes worldwide. For example, Hong Kong and Japan have deployed residential connections of 1 Gbps while Sweden tested 40 Gbps connections to select homes. High bandwidths to homes make the wireless APs the most likely source of the bottleneck link. Maier *et al.* suggest that wireless networks may already be the cause of the performance bottleneck in home networks [3].

Solutions proposed and implemented by wireless AP manufacturers provide basic QoS support. Some APs statically prioritize a physical Ethernet port for a specific device connected to that port or prioritize flows based on their transport protocol type and port number. However, the average home user has difficulty understanding and configuring even basic features of a wireless AP without adding the burden of having to configure an AP with port and flow information. Even for experienced users, these solutions have limitations. For port-based priorities, users need to know the protocol and ports for the applications they run and new, unclassified applications are unable to receive appropriate treatment.

The limitations of static QoS identification have prompted research in automatic classification of flows to provide for specific QoS, removing the need for users to configure their wireless APs to prioritize traffic. However, in addition to the limited ability to classify new applications, changes in the wireless network connectivity can bring down the overall performance and quality of the home network if flows with poor connectivity are given higher priority. In this situation, it is better not to give priority to flows with poor connectivity despite their classification.

To address the shortcomings of either static or dynamic flow classification and treatment while still providing varied per-application QoS, we propose a Credit-based Home Access Point (CHAP) that uses a new queue management scheme based on per-flow credits in deciding which packets to send. The primary goal of CHAP is to improve overall application quality in home networks with minimal configuration and no explicit classification. CHAP takes advantage of the relationship between delay tolerance and bandwidth

MM'10, October 25–29, 2010, Firenze, Italy.

usage of users' activities. This relationship is mapped such that flows that use less bandwidth have more credits than flows that use more bandwidth. CHAP uses wireless channel usage time as the credit cost metric for each flow. Thus, given the same bandwidth, flows with good connectivity use less credits than flows with poor connectivity.

Our NS-2 [4] simulation results [2] show that CHAP provides improved performance for delay sensitive applications while maintaining overall network performance similar to that of a traditional FIFO AP. This paper extends these results by presenting an evaluation of an implementation of CHAP, showing results from a case study along with viewable demonstrations[1]. The results show that in the presence of congestion, CHAP: 1) provides Web response times superior to FIFO; and 2) provides better latencies for online games than does FIFO. These application quality improvements are accomplished without per-application configuration and while providing performance similar to FIFO with respect to traditional application throughputs and overall network performance.

The rest of this paper is organized as follows: Section 2 provides the derivation and some details of the CHAP mechanisms; Section 3 describes the set of case studies that evaluate CHAP and includes comparisons of the performance of CHAP with FIFO; and Section 4 summarizes our findings and possible future work.

2. PROPOSED APPROACH

2.1 Bandwidth and Delay for Residential Activities

Applications run in the home can be broadly categorized based on the user activities that they support. Popular user activities include Web browsing, audio and video streaming, voice over IP (VoIP), instant messaging, online games, email and file downloading. Typically, applications that support a specific activity share common characteristics in terms of their bandwidth usage and delay tolerance.

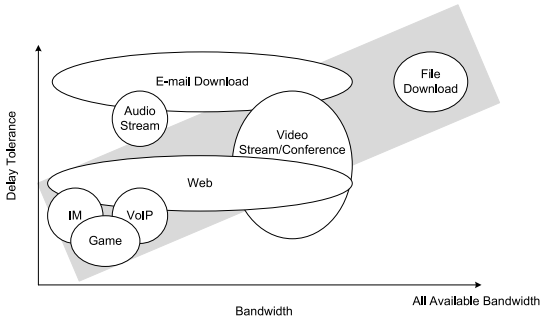


Figure 1: Characteristics of Network Applications

Figure 1 depicts the bandwidth usage and delay tolerance of various user activities. As the figure suggests, there is a proportional relationship between each activity's bandwidth usage and its delay tolerance, indicated by the angled gray rectangle. The more bandwidth an activity uses, the more delay tolerant it is and vice versa.

2.2 CHAP Algorithm

Typical operating system (OS) schedulers favor interactive processes that are not CPU-bound. Our proposed approach, Credit-based Home Access Point (CHAP), draws upon the similarity of process scheduling by an OS to packet transmission scheduling by an AP, exploiting the relationship between delay tolerance and bandwidth usage shown in Figure 1. Since the AP is responsible for routing traffic to the correct host within the home network, the typical home AP keeps a Network Address Translation (NAT) table with a 5 tuple per entry: source address, source port, destination address, destination port and protocol. This same information can be used to identify flows. Bandwidth usage can be observed easily at the AP because every packet that arrives and leaves the home network goes through the AP. CHAP then uses per-flow credits to track bandwidth use and improve overall application quality.

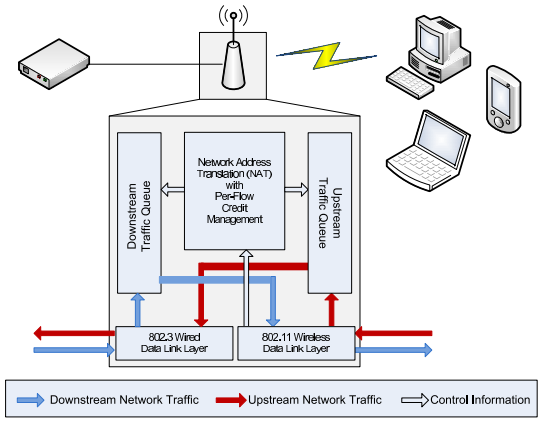


Figure 2: Block Diagram of CHAP

Figure 2 shows the components of CHAP inside an AP, with the solid arrows indicating network traffic in and out of the home network. In CHAP, unlike in a traditional AP, the downstream and upstream queues get control information from the enhanced NAT component with credits to prioritize the packets in the queue. In addition, the NAT component gets transmission time information from the 802.11 data link layer to update the credits.

To adjust the credits for each flow, CHAP uses the channel usage time at the wireless data link layer which provides a combination of its bandwidth usage and wireless connectivity. The more frames a flow transmits, the more credits it uses and the fewer credits it has remaining. Similarly, the more time the frames take to transmit, the more credits it uses and the fewer credits the flow has remaining. Therefore, in general, flows that drain credits more slowly have higher priority. This mechanism ensures nodes with poor connectivity, even those with low application-level bandwidth (i.e., delay sensitive applications), do not degrade performance for the entire wireless network.

When all active flows exhaust their allocated credits, CHAP initiates a credit boost for these flows. The credit boost formula is based on a previous version of the Linux process scheduler: $\alpha_i' = \frac{\alpha_i}{2} + I$, where i is the flow index, α_i is the credit of flow i, and I is the increment. The parameter I is in units of time, consistent with the unit of credits. This boost occurs when every flow with data in the queue has 0 (or fewer) credits and is applied to *every* active flow, those with credits or not.

Algorithm 1 summarizes the dequeue function for downstream traffic in pseudo code format. dequeue() selects and returns the first packet from the flow with the most credits in the queue. Step 1 finds the flow with the most credits. If a credit boost is needed, Step 3 boosts credits for all active flows. Steps 5-6 dequeue the first packet from the flow selected in Step 1. Step 7 returns the packet for transmission. After packet transmission, the wireless data link

[1]http://perform.wpi.edu/chap/

Algorithm 1 CHAP::dequeue()

1: find k such that $\alpha_k = max[\alpha_1, ..., \alpha_n]$ and $k \in$ set of backlogged flows
2: **if** $\alpha_k \leq 0$ **then**
3: $\alpha_i = \frac{\alpha_i}{2} + I$ for all $i \in$ active flows
4: **end if**
5: p = the first packet from flow k in the queue
6: remove p from queue
7: return p
... $\alpha_k = \alpha_k - cost$

layer reduces the transmitted flow's credits by the cost (i.e., the channel usage time). Further CHAP details and, in particular, the enqueue() operation can be found in our earlier work [2]. They are not presented here due to space constraints.

3. PERFORMANCE

This section compares CHAP against FIFO for scenarios where Web browsing and online game activities compete for network resources with a bulk file download. In these scenarios, the home user would typically favor browsing and gaming over a bulk file download which can happen in the background. CHAP is implemented as a Linux Queuing Discipline (qdisc) and FIFO, the default queue management for wireless home APs, serves as a performance baseline.

3.1 Linux Implementation

Linux allows users to connect multiple network interfaces to create a bridge. Each network interface can be associated with a qdisc using Traffic Control (TC) to manage its queue. Outgoing packets for a network interface are first queued by the associated qdisc before being passed to the network interface device queue. CHAP is implemented using the Stochastic Fair Queue (SFQ) qdisc as the base. CHAP hashes each incoming packet based on flow determined by source address, destination address and protocol type, and keeps track of the flow's corresponding credits.

Because of difficulties with the existing Linux codebase[2] for the host AP and madWifi driver in an IEEE 802.11 infrastructure network, we emulated IEEE 802.11g frame transmission using a kernel timer interrupt to schedule packet departures. The IEEE 802.11 delay of each packet is calculated by:

$$d = DIFS + \frac{p_{data} + p_{header}}{R_{data}} + SIFS + \frac{p_{ack}}{R_{basic}} \quad (1)$$

where d is the emulated delay, p_{data} is the packet size, p_{header} is the IEEE 802.11 header size (34 bytes), p_{ack} is the ACK frame size (14 bytes), R_{data} is the data rate (54 Mbps for IEEE 802.11g) and R_{basic} is the basic rate (1 Mbps). The minimum value of d is 177 μs when p_{data} is 0 bytes and a p_{data} of 1500 bytes results in a d of 399 μs.

Our implementation and wireless emulation has been validated[3] against the simulations from our earlier work [2]. For the case studies, an additional wireless-emulation delay of 2 ms is added to ensure the Linux bridge acts as the bottleneck to the home network. The additional delay limits the incoming bandwidth to about 5 Mbps, which is less than the home ADSL's downstream bandwidth of 7 Mbps.

[2]Kernel panics and frequent disassociation of wireless hosts.
[3]Not shown here due to lack of space.

3.2 Setup

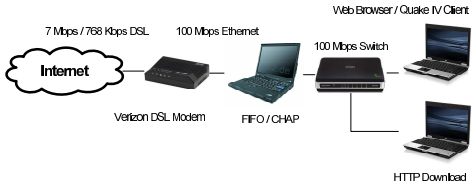


Figure 3: Case Study Network Topology

Figure 3 depicts the experimental setup used to represent a home network. The Internet is connected by a 7 Mbps downstream and a 768 Kbps upstream ADSL connection. The Linux bridge with FIFO and CHAP connect the home network and the ADSL modem. Both FIFO and CHAP allow up to 350 packets in queue. One computer runs one of the two delay sensitive applications: *Firefox* for Web browsing and *Quake IV* for online gaming. The other computer runs Firefox for a bulk file download of the Debian Linux DVD ISO. Each case study tests the FIFO and CHAP queue in the absence of the bulk file download and then tests them again with the competing bulk file download.

3.3 Case Study 1: Web Browsing

Four popular news Web sites – Google News, Yahoo, Wall Street Journal (WSJ) and CNN – were used for the Web browsing tests. Table 1 indicates the composition of each Web site in terms of its total size in bytes and the number of text and non-text objects. CNN is the heaviest in terms of the total size while WSJ has the most non-text objects. Google News is the lightest with respect to size and the total number of objects. Web response time is used to evaluate the performance of the Web browsing activity. Web response time is the time a user must wait after clicking on a link or entering a URL until the entire Web page, text plus images, is retrieved. Lifehacker.com's modified version of Webmonkey's Browser Load Time Stopwatch[4] is used to time the load time for each Web page.

Table 1: Webpage Statistics

Website	# of Text Objs	# of Non-text Objs	Size (KB)
Google News	2	4	155
Yahoo	6	42	322
WSJ	19	132	750
CNN	33	103	1142

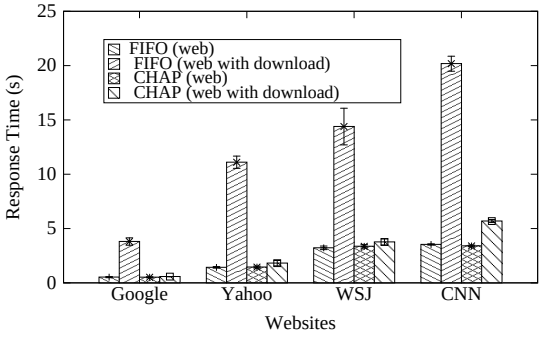
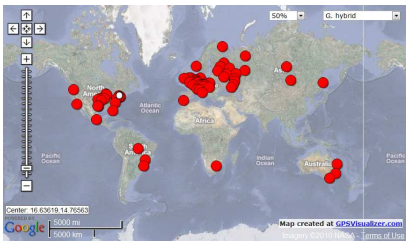


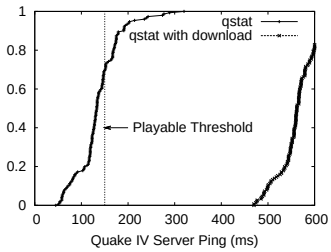
Figure 4: Web Response Time

Figure 4 compares average Web response time for five visits to each of the four Web sites. All these measurements were com-

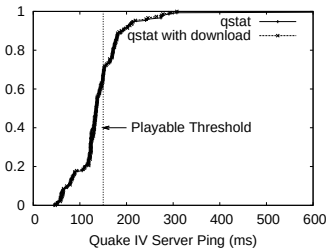
[4]http://cache.lifehacker.com/assets/resources/stopwatch.php



(a) Quake IV Server Locations



(b) CDF of Server Pings (FIFO)



(c) CDF of Server Pings (CHAP)

Figure 5: Quake IV Server Locations and CDFs of Quake IV Server Pings

pleted in a single two-hour session to keep the Web site components shown in Table 1 relatively static. The error bars in the figure represent the standard deviation of the measured samples. In the absence of the bulk file download, all the Web sites loaded in less than 5 seconds for both FIFO and CHAP. Google News loaded the fastest in 0.525 s and 0.514 s, Yahoo in 1.43 s and 1.45 s, Wall Street Journal in 3.22 s and 3.36 s, and CNN in 3.54 s and 3.40 s using FIFO and CHAP respectively. When loading Web sites while downloading the Linux DVD ISO, the Web response times for FIFO increased by 400% to 800%. Google News finished loading in 3.81 s (725%), Yahoo in 11.11 s (778%), WSJ in 14.40 s (447%) and CNN in 20.18 s (570%) for FIFO. On the contrary, CHAP managed to keep the Web response time low and close to the Web response time in the absence of the bulk file download. Google News loaded in 0.574 s (112%), Yahoo in 1.81 s (125%), WSJ in 3.8 (112%) and CNN in 5.69 s (167%). Videos of loading CNN have been recorded to further illustrate the differences in browsing performance under these conditions[5].

3.4 Case Study 2: Quake IV

Quake IV is used to evaluate performance for online games. Quake IV is a first person shooter, a genre of fast paced, highly interactive online games. Previous work has demonstrated that Quake requires round-trip times of about 150-180 ms or lower for users to find it playable [1]. Quake measures server "ping" times as the averaged round-trip times from the game client to the Quake servers. We use *qstat*[6] to contact the Quake IV master server for a list of active game servers and to act as a game client in obtaining the ping times to each server by sending a single probe packet. The IP addresses returned by qstat are mapped to longitudes and latitudes according to GeoIP[7]. Figure 5a depicts the geographical locations of active Quake IV servers around the world in March 2010, with the white circle indicating Worcester, MA, USA.

Figures 5b and 5c provide the CDFs of all the server ping times recorded using FIFO and CHAP in the absence and presence of file bulk download. When there is no competing download, the median server ping times are 133 ms and 135 ms for FIFO and CHAP, respectively. About 70% of server ping times are under the playable threshold of 150 ms for both FIFO and CHAP. When there is a competing download, the CDF of server ping times shifts to the right for FIFO, as indicated by the dotted line. The median server ping time increases by 429 ms to 562 ms, and none of the server ping times are below the playable threshold of 150 ms. CHAP, however, is able to keep server ping times comparable to the ping times in the absence of bulk file download. The median ping time stays the same at 135 ms and about 70% of server ping times are still below the playable threshold. Available video recordings further illustrate differences in game performance under these conditions[8].

[5]http://perform.wpi.edu/chap/

[6]http://www.qstat.org/

[7]http://www.maxmind.com/app/ip-location

4. CONCLUSIONS

This paper introduces CHAP, a queue management technique for wireless access points (APs), designed specifically to improve application QoS over home wireless networks. CHAP retains and manages credits for all flows passing through the AP, decrementing credits as a function of wireless channel usage time, and boosting credits when all active flows run out of credits. CHAP has the potential to increase the quality of delay sensitive activities while preserving the quality of delay insensitive activities. We are unaware of other access point queue management techniques that can improve quality of delay sensitive applications without using a priori configuration and/or flow classifications.

CHAP has been implemented in a Linux testbed, enabling performance evaluation with actual user activities. Specifically, case studies involving Web browsing and a networks game (Quake IV) individually competing against a bulk file download are used to compare CHAP against FIFO. The results suggest CHAP provides significantly lower loading times for Web browsing and faster response times for Quake IV when compared with FIFO.

Our ongoing work involves evaluating CHAP under a broader range of conditions and circumstances including a wider range of of clients and QoS measures such as jitter, the impact of varying frame error rate and AP queue size, the presence of upstream traffic and consideration of other background traffic such as Peer-to-Peer (P2P) applications. A prototype implementation using a Linux-based firmware (DD-WRT) for commercial APs is planned to validate the emulation and to understand hardware nuances.

5. REFERENCES

[1] G. Armitage. An experimental estimation of latency sensitivity in multiplayer Quake 3. In *11th IEEE International Conference on Networks (ICON 2003)*, Sydney, Australia, September-October 2003.

[2] C.-S. Lee, M. Claypool, and R. Kinicki. A Credit-based Home Access Point (CHAP) to Improve Application Performance on IEEE 802.11 Networks. In *Proceedings of ACM Multimedia Systems (MMSys)*, Scottsdale, AZ, USA, February 2010.

[3] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the ACM Internet Measurement Conference (IMC)*, Chicago, IL, USA, November 2009.

[4] U. of California Berkeley. The Network Simulator - ns-2. Online at http://www.isi.edu/nsnam/ns/.

[8]http://perform.wpi.edu/chap/