2010-06-23

A Credit-based Home Access Point (CHAP) to Improve Application Quality on IEEE 802.11 Networks

Choong-Soo Lee
Worcester Polytechnic Institute

A Credit-based Home Access Point (CHAP) to Improve Application Quality on IEEE 802.11 Networks

by

Choong-Soo Lee

A Dissertation
Submitted to
The Academic Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment
of the Requirements for the Degree
Doctor of Philosophy

in

Computer Science

by

June 2010

Committee:

Professor Mark Claypool - Computer Science, Worcester Polytechnic Institute

Professor Robert Kinicki - Computer Science, Worcester Polytechnic Institute

Professor Craig Wills - Computer Science, Worcester Polytechnic Institute

Professor Carey Williamson - Computer Science, University of Calgary

Abstract

Increasing availability of high-speed Internet and wireless access points has allowed home users to connect not only their computers but various other devices to the Internet. Every device running different applications requires unique Quality of Service (QoS). It has been shown that delay-sensitive applications, such as VoIP, remote login and online game sessions, suffer increased latency in the presence of throughput-sensitive applications such as FTP and P2P. Currently, there is no mechanism at the wireless AP to mitigate these effects except explicitly classifying the traffic based on port numbers or host IP addresses. We propose CHAP, a credit-based queue management technique, to eliminate the explicit configuration process and dynamically adjust the priority of all the flows from different devices to match their QoS requirements and wireless conditions to improve application quality in home networks. An analytical model is used to analyze the interaction between flows and credits and resulting queueing delays for packets. CHAP is evaluated using Network Simulator (NS2) under a wide range of conditions against First-In-First-Out (FIFO) and Strict Priority Queue (SPQ) scheduling algorithms. CHAP improves the quality of an online game, a VoIP session, a video streaming session, and a Web browsing activity by 20%, 3%, 93%, and 51%, respectively, compared to FIFO in the presence of an FTP download. CHAP provides these improvements similar to SPQ without an explicit classification of flows and a pre-configured scheduling policy. A Linux implementation of CHAP is used to evaluate its performance in a real residential network against FIFO. CHAP reduces the web response time by up to 85% compared to FIFO in the presence of a bulk file download. Our contributions include an analytic model for the credit-based queue management, simulation, and implementation of CHAP, which provides QoS with minimal configuration at the AP.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Affordable prices for wireless Access Points (APs) and increases in wireless link capacities have driven homes, academic institutions and businesses to deploy Wireless Local Area Networks (WLANs). The proposed IEEE 802.11n standard supports up to 600 Mbps in the PHY layer [1]. IEEE 802.11n wireless APs from Linksys claim to support up to 270 Mbps [2]. Increases in wireless link capacities enable most, if not all, applications that were only possible over wired network links to run over wireless. Such popularity in wireless networks has also increased the use of IEEE 802.11 on portable devices such as cell phones, PDAs and portable game devices.

Wireless APs have become popular in networking multiple devices at home and allowing multiple individuals to share a single household Internet connection while doing different activities simultaneously. People use not only desktop and laptop computers, but also many other devices with network capabilities. These devices include gaming consoles such as the Sony PlayStation and Microsoft Xbox, portable gaming consoles such as the Sony PSP and Nintendo DS, Voice-over-IP (VoIP) phones, video streaming servers such as Slingbox and LocationFree, cell phones such as iPhone and Blackberry, PDAs, and more. Many applications that are run on these devices have different Quality-of-Service (QoS)

requirements. For example, a user playing a game on a Sony Playstation 3 wants low latency to enjoy the online game. A user downloading a high-definition (HD) quality movie trailer seeks consistent, high throughput. Multiple activities done over the same Internet connection may cause congestion and degrade application performance for some of the users. A typical example is when a user tries to use VoIP while another user downloads a huge file, the VoIP suffers from choppy sound and an increase in delay.

An increase in latency for delay sensitive applications is an indication that there is a queue buildup somewhere along the network path. Figure 1.1 depicts a typical network setup at home. The broadband Internet service provider (ISP) provides access to their network through a gateway. The ISP also provides a modem based on the connection type, typically connected to a wireless AP. In some cases, the modem and the wireless AP are combined into one device. There are three places of interest for congestion: ISP gateway, modem and wireless AP. The link between the ISP gateway and modem depends on the Internet subscription and connection type. The link between the modem and the wireless AP is 100 Mbps to 1 Gbps Ethernet. Devices at home connect to the wireless AP at speeds up to 270 Mbps over 802.11n. The queue is most likely to build up over a bottleneck link and this depends on the connection capacity of the Internet from the ISP. Currently in the US, the highest capacity Internet access offered is Fiber Optic Service (FiOS) from Verizon [3]. Verizon currently offers up to 50 Mbps downstream and 20 Mbps upstream and plans to increase their capacity up to 100 Mbps. With this setup, the ISP gateway is most likely the source of congestion.

However, ISPs are trying to push faster broadband access to homes worldwide. Some Asian countries already provide 1 Gbps to residences. Hong Kong Broadband Network

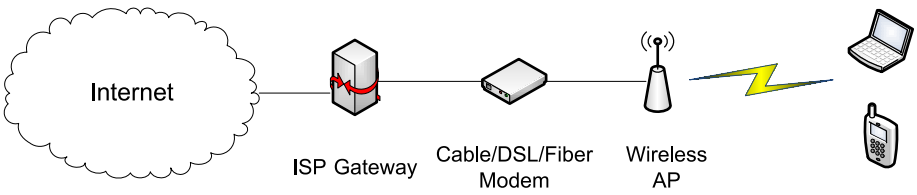


Figure 1.1: Typical Internet setup for home

(HKBN) has provided a 1 Gbps symmetric service for the residential market since April 2005 [4]. Approximately 800,000 households, out of a total of 2.2 million households in Hong Kong, are wired to receive the service [5]. KDDI Corp in Japan has launched a fiber-optic communications service with downstream and upstream speeds up to 1 Gbps since October 2008 [6]. Both SK Broadband and Korea Telecom (KT) in South Korea have been offering 100 Mbps for homes for years [7]. Europe is also moving towards increased broadband service to homes. CityNet in Amsterdam has tested 1 Gbps fiber to residences in September 2008 [8]. Sweden successfully tested a 40 Gbps Internet connection to a home [9]. With such high bandwidths available to homes, it is likely that the wireless AP will become the source of the bottleneck. Historically, wireless communication capacity has been lower than wired capacity and will likely remain thus. Moreover, devices in wireless networks share the same wireless spectrum as the medium for data transmission and reception.

Calvert *et al.* argue that problems in home networks need to be solved in middle devices rather than end hosts [10]. The authors list five difficulties users face in home networks: provisioning, topological complexity, troubleshooting, security, and composition. With a diverse set of devices connected in home networks, provisioning is essential to provide QoS for all applications running on the devices. Misconfiguration of the network can cause degradation in application quality and/or prevent users from accessing the Internet. In addition, it is difficult for users to understand the physical and logical topology of home networks. Increasing complexity in home network topologies requires extra security awareness and configuration from users. Most home users are relatively unsophisticated users, from a networking perspective, and troubleshooting problems in home networks is not an easy task for them. Home users also need to be aware of security issues within and outside home networks. The current Internet architecture lacks support for providing simple security policies. The rich variety of networked devices makes composition an important issue. Networked devices need to co-exist in home networks and provide compatibility with each other. Based on these difficulties, the authors propose six requirements for the "smart mid-

dle" device: self-configuring/self-administering, secure by default, explicit user interface, compatibility with existing external TCP/IP-based applications, application-independence and support for composition. Self-configuring/self-administering takes the burden off home users. Default security provides sufficient security for home networks. Explicit user interface makes it easy for users to interact with and control home networks. Existing TCP/IP applications must seamlessly work with the "smart middle" device. The "smart middle" device must be independent of applications, providing a robust and evolvable network. Ultimately, the "smart middle" device must support composition to allow devices to interconnect within home networks and to connect to the Internet. Wireless APs are "smart middle" devices with a potential to meet most, if not all, requirements.

There are approaches implemented by the manufacturers for wireless APs to provide basic QoS support. Some wireless APs support prioritizing physical Ethernet ports for a specific device connected over a cable. They also support prioritizing flows based on their protocol type and transport layer port for known traffic. While these approaches may be easy and satisfactory for some users who know how to configure wireless APs, typical Internet users have considerable difficulty understanding and configuring APs. Even with experienced users, these solutions have some limitations. For port-based priorities, users need to know the protocols and ports that applications use. It is also difficult for new applications to receive appropriate treatment. Moreover, the traffic that goes through the wireless APs is dynamic and is not always the same mix of applications. These limitations have prompted research in classification of flows to provide specific QoS needs for them.

Classification of flows removes the need for users to configure their wireless APs to prioritize their traffic. Instead wireless APs examine the flows and classify them automatically. However, there are limitations to current classification approaches due to the dynamic nature of wireless conditions. Once classification of flows takes place, flows are treated according to the class in which they are placed. For example, VoIP flows get VoIP treatment while video flows get video treatment. However, if the VoIP client is connected over wireless, the client condition might change based on physical location or interference.

If the connectivity gets worse and the AP tries to give priority to the VoIP flow, everyone's performance suffers due to the AP trying to forward traffic to the client with a poor wireless connectivity. In such a situation, it is recommended to move away from giving priority to flows with poor connectivity [11].

Therefore, this dissertation focuses on the importance of dynamic wireless conditions and application quality improvement. Credit-based queue management that classifies flows implicitly based on the flows' characteristics and changes priority of flows dynamically according to wireless connectivity is proposed to be deployed in wireless APs at home. By utilizing credit-based queue management, delay sensitive flows can benefit from lower latency and wireless APs can dynamically adjust priority to different flows to give higher overall QoS in home networks.

1.2 The Dissertation

The goal of the dissertation is to improve overall quality of applications in a home network by using credit-based queue management at the wireless AP. Figure 1.2 depicts the proposed module and its interaction between layers. The network layer queue, which is represented by a white block, is managed by the proposed credit queue management. The data link layer provides feedback for the network layer queue to help out with credit calculation. As discussed in Section 1.1, the queue is most likely built up on the downstream traffic and not on the upstream traffic because both downstream and upstream bandwidth offered by ISPs exceed the wireless bandwidth. Therefore, the upstream queue is not to be controlled by credit-based queue management.

The assumption is that the wireless link at home is the bottleneck connection in an end-to-end network path. The more a flow uses the network, the more credits it drains while the less a flow uses the network, the fewer credits it drains. Prioritizing the flow with the highest number of credits lowers the latency for that flow. The credit is in units of time and the deduction is based on the transmission time of the corresponding frames through the wireless data link layer. In addition, typical bandwidth usage of applications

characterize flows adequately for an implicit classification. Delay-sensitive flows such as for VoIP, online games and remote login sessions have relatively low bandwidth while delay-insensitive flows such as for file transfer and video streaming tend to use higher bandwidth. The inverse relationship between delay requirement and bandwidth usage fits nicely with credit-based schemes.

It is known that the overall performance of wireless networks can be severely decreased when having one node with poor wireless connectivity. Therefore, if there are delay-sensitive applications running on a node with a poor connection, it may not be best to give priority to those applications [11]. Explicit classification methods and scheduling algorithms would continue to give delay-sensitive flows priority, decreasing overall performance of the wireless network. However, credits based on transmission time naturally reduce the credits of flows with poor wireless connectivity, resulting in lower priority for those flows.

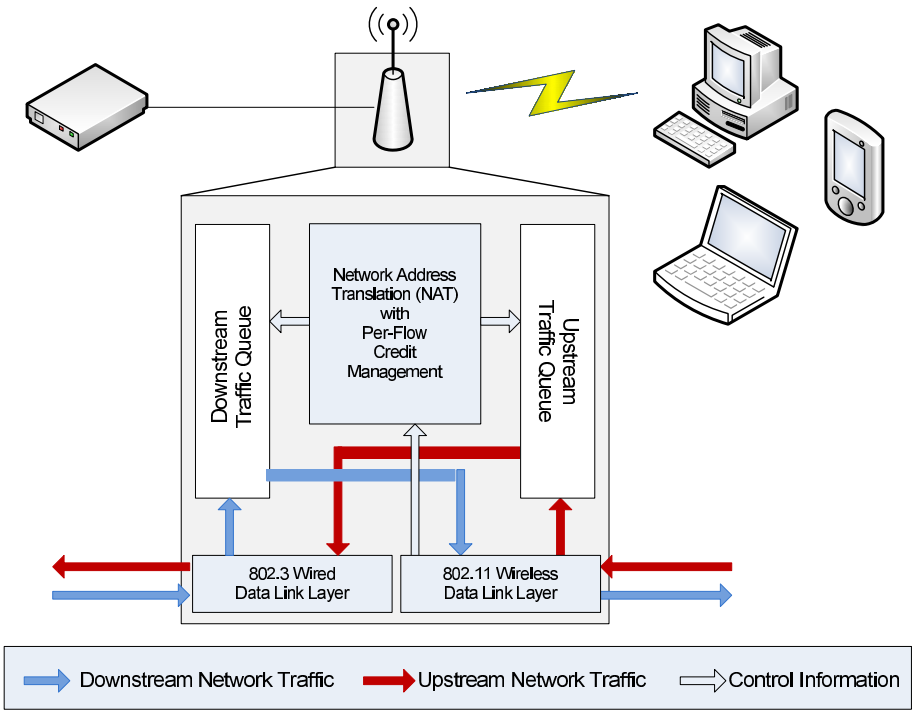


Figure 1.2: The Block Diagram of CHAP

Figure 1.2 depicts CHAP components involved inside a wireless access point. Down-

stream traffic into the home network follows light arrows while upstream traffic into the Internet follows dark arrows. Downstream traffic comes into the wireless access point through the IEEE 802.3 Ethernet data link layer and is passed to the downstream traffic queue. A packet from this queue is selected based on credit information kept along with Network Address Translation (NAT) information that the wireless AP tracks. Once the packet is selected, it is sent to the IEEE 802.11 wireless data link layer to be transmitted. When transmission completes, the IEEE 802.11 wireless data link layer relays transmission time information to the NAT component to update the credit. Upstream traffic comes in through the IEEE 802.11 wireless data link layer and goes to the upstream traffic queue. Most likely, there is no queue build-up on the upstream.

The dissertation consists of the following components:

- Model. The range of credits of flows are estimated based on their bandwidth usage. Then, a delay model for a probabilistic priority discipline queue is used to analyze the average queueing delay of flows based on their range of credits. The model shows that CHAP can reduce delay for flows with relatively lower bandwidth usage compared to the FIFO queueing discipline.

- Validation. The mathematical model requires validation. CHAP is implemented and evaluated using Network Simulator (NS2) [12] and used to validate the model. Performance data from the simulations using the scenarios in the mathematical model are compared to the results of the model.

- Evaluation. Once the model is validated, a wide range of simulation scenarios is run using NS2 to provide performance analysis of CHAP under various conditions. The difference in scenarios include dynamic wireless conditions (devices with good and poor wireless connectivity), number of devices, and different mixes of user activities. The quality of these activities is calculated using well-known quality metrics and compared to performance with a traditional First-In-First-Out (FIFO) and Strict Priority Queue (SPQ) queue access point.

- Implementation. CHAP is implemented as a queueing discipline (Qdisc) in Linux as a proof of concept. The Linux machine is modified to act as a bridge, emulating an IEEE 802.11g network in a 100 Mbps Ethernet network. The implementation is validated against simulation results, and a few case studies are conducted in a real environment to demonstrate the efficacy of CHAP.

1.3 Contributions

The main contribution of this dissertation is the design and evaluation of the Credit-based Home Access Point (CHAP) to improve the quality of Internet applications in wireless home networks. The specific contributions include:

- The design and configuration of credit-based queue management for wireless home access points. Our Credit-based Home Access Point (CHAP) provides better quality for applications in home networks with minimal configuration and without changes to end hosts. We provide an evaluation of CHAP in comparison with First-In-First-Out (FIFO), the default queue management in APs, as the baseline, and Strict Priority Queue (SPQ) as the best case in most cases, given an explicit, static classification and scheduling algorithm. Through a simulation study, we demonstrate that CHAP provides simple and effective application quality improvement. The results also show that CHAP provides significant improvement for delay-sensitive applications while maintaining the quality for delay-insensitive applications, and adapts to dynamic wireless conditions over a wide range of realistic traffic mixes and loads.

- Analytical modeling of CHAP. We model CHAP's behavior by examining the stable credit ranges for flows and applying the known queueing model for a Probabilistic Priority Discipline Queue (PPDQ). The model demonstrates how average queueing delays of flows are related to the ratio of their rates. We also analyze the parameter I, the only configuration variable of CHAP, and recommend a suitable value for I.

- The implementation and evaluation of the CHAP system. We implement CHAP as

a queueing discipline (Qdisc) in Linux. Evaluation in a home network demonstrates significant improvement in quality for Web browsing and online games over FIFO in the presence of a delay-insensitive application.

- Validation of NS2 simulation. We mirror the same set of scenarios in NS2 and in the real world. The comparison of results shows that NS2 simulations provide similar trends as the experiments. The gap between the simulation and experimental results suggests TCP protocol implementation differences and provides evidence of unsimulated factors such as system level overheads and interference in wireless networks.

- The design and implementation of a video streaming application in NS2. The video streaming application uses video traces [13, 14] and sends frames at the encoded rate over UDP. Moreover, the video streaming application is capable of adding Frame Error Correction (FEC) data for more robust streaming.

- Simulation and analysis of TCP variants. TCP NewReno [15], BIC [16], CUBIC [17] and Compound [18] are simulated to demonstrate their differences in congestion window, queue size and throughput. It is important to understand the differences in TCPs because modern operating systems default to different congestion control mechanisms. Simulation results demonstrate that CUBIC provides the most stable throughput with a small queue limit while Compound provides the lowest queue size with a large queue limit.

1.4 Roadmap

The remainder of this dissertation is organized as follows: Chapter 2 provides background knowledge to the work in this dissertation; Chapter 3 discusses related research in the areas of QoS management, traffic classification, and credit-based queue management; Chapter 4 describes the approach; Chapter 5 presents the analytical model of the approach; Chapter 6 describes validation of the analytical model presented in Chapter 4 and evaluation of the approach through simulations; Chapter 7 presents details of the Linux implementation,

validation and case studies; and Chapter 8 concludes this dissertation research and lists possible future work.

Chapter 2

Background

This chapter reviews fundamental techniques and terminologies referenced in the dissertation. Section 2.1 reviews user activities such as Web browsing, file downloads and audio/video streaming, along with their network characteristics. Section 2.2 reviews wireless network techniques, including general characteristics of wireless media and IEEE 802.11 Wireless LAN (WLAN). Section 2.3 describes process scheduling techniques and their relevance to packet scheduling techniques. Section 2.4 introduces a taxonomy for active queue management techniques. Section 2.5 reviews all the metrics used to measure the quality of applications tested in simulation and experiment.

2.1 User Activities and Their Network Characteristics

Users engage in a variety of activities on their network devices over the Internet at home. Common activities include emailing, instant messaging, listening to online radio, watching online video clips, talking with/without video, browsing the Web, playing online games, and downloading files. Each activity may take place over multiple applications. For example, instant messaging is done over many different applications such as America Online (AOL), Yahoo, Google, and MSN. Video streaming can be done over Windows Media Player, YouTube, RealPlayer, and QuickTime Player. Despite the differences in applications, the applications supporting each user activity tend to share common characteristics.

Network characteristics of interest are bandwidth usage, duration, number of flows, and burstiness with respect to bandwidth and delay requirements. Bandwidth usage and delay requirements are particularly important to our approach.

Online games Video games have been around as a form of electronic entertainment since the 1960s. Some games in the 1980s were online through Bulletin Board Systems (BBS), but these were limited to those who had modems and phone lines to connect to such services. As the Internet became popular in the 1990s, computer games started connecting players across the world. In 1996, Quake was the pioneer in the computer gaming industry to connect players over the Internet. Ever since, many different genres of games use the Internet to connect their players across the world. Console games were initially not networked but Sony PlayStation and Microsoft XBox were able to offer online services to their customers by introducing network adapters. Now Sony and Microsoft offer PlayStation Network [19] and XBox Live [20], respectively, for games to be played online on their consoles. Nintendo also offers online game play to Wii players. In addition to computers and consoles, there are portable gaming devices such as Sony PlayStation Portable (PSP) and Nintendo DS that take advantage of the wireless network to connect players in the same vicinity.

There are many genres of online video games: First Person Shooter (FPS), Real Time Strategy (RTS), Massively Multiplayer Online (MMO), Turn-based Strategy, Puzzles, etc. A popular game in the FPS genre is Halo 2 by Bungie on the XBox gaming console. Zander and Armitage examine the characteristics of Halo 2 [21] network traffic [22]. Halo 2 players can act as both a server and a client. Each gaming console can support up to four players. Zander and Armitage find that the packets from client consoles to the server console are consistent in packet size while the packets from the server console to client consoles have wider variation [22]. The packet sizes also scale with the number of players in the game and also on one console. The client consoles show mean bandwidth of 25 Kbps with four players, which is the maximum number of players on one console. The server console shows mean bandwidth of 55 Kbps with 12 players. The mean inter-arrival times for both the

server console and client consoles are 40 ms with small standard deviations.

Starcraft [23] is a popular game in the Real Time Strategy (RTS) genre, although now over 10 years old. Dainotti, Pescape, and Ventre measure the network traffic characteristics of Starcraft [24]. Starcraft is a PC game and can only support one player on one machine. Players connect to BattleNet [25] to find other players and then once the game session starts, each player communicates with other players without going through BattleNet, using the network model described in [26]. Starcraft supports up to eight players in a game session, using UDP packets. The number of packets exchanged scales with the number of players in the game session. Over 90% of the outbound and inbound packets are less than 25 bytes. Over 90% of the inter-departure times are less than 150 ms. Over 90% of the inter-arrival times are less than 100 ms.

World of Warcraft (WoW) [27] is the most popular game in the Massively Multiplayer Online (MMO) genre. A survey [28] done by MMOGCHART.COM [29] shows that 62.2% of MMO players are subscribed to World of Warcraft as of April 2008. They analyze the network traffic of World of Warcraft [30]. Svoboda, Karner, and Rupp define one application data as a collection of packets, meaning that application data may not fit in one packet. However, over 90% of downstream WoW data are less than 1500 bytes, implying that most of the application data can fit in one packet. Every upstream application datum is less than 250 bytes. Over 60% of inter-data time is less than 250 ms. Over 90% of the downstream bandwidth is less than 15 Kbps while over 90% of the upstream bandwidth is less than 5 Kbps.

The network traffic characteristics from Halo 2, Starcraft, and World of Warcraft can be used to generalize the characteristics of online game network traffic. They all use small amounts of bandwidth, both downstream and upstream. Their inter-arrival and inter-departure times are mostly constant. Due to their interactive nature, their delay tolerance is low. Armitage demonstrates that Quake 3 requires round-trip times of about 150-180 ms or lower for users to find it playable [31].

Voice over IP Voice over IP (VoIP) technology surfaced in 1995 when a small company called Vocaltec released the first Internet phone software. It ran on a home PC and ran much like any PC phone software today. In 1998, VoIP traffic reached approximately 1% of all voice traffic in the United States and jumped to 3% by 2000 with the help of Cisco and Lucent manufacturing equipment designed to route and switch VoIP traffic. Currently there are home devices that are built to support popular VoIP services such as Skype [32].

Sharafeddine, Riedl, Glasmann, and Totzke examine the network characteristics of VoIP applications [33]. Two common voice coders mentioned are G.711 and G.723.1 [34]. G.711 uses 64 Kbps of bandwidth and its frames are 0.125 ms long while G.723.1 uses 6.3 Kbps or 5.3 Kbps and its frames are 30 ms long. Measurements from IP phones and VoIP gateways show that both protocols produce constant bit rate traffic with low standard deviation for inter-arrival times. G.711 uses packet sizes of 200 bytes with a mean inter-arrival time of 20 ms. This yields an average bit rate of 80 Kbps. G.723.1 uses packet sizes of 64 bytes with mean inter-arrival time of 30 ms. This results in an average bit rate of 17 Kbps. Therefore, VoIP applications have low bandwidth requirement but delay tolerance is low due to its interactive nature. The action of a user on each end has to reach the other party in a reasonable amount of time to have a smooth interactive session. VoIP quality is good with 150 ms of latency or lower and it gets poor as it goes beyond 400 ms, according to the specification for premium IP service [35].

Many instant messenger clients support video communication on top of audio. Normally users can see each other through Web cams while they are talking to each other online via audio and/or instant messages. This is a case of video streaming both upstream and downstream. Just like voice communication, the delay tolerance is low for a video conference while the bandwidth requirement is higher than for audio.

Audio and video streaming Many users stream audio and video online. By audio and video streaming, we mean primarily one-way traffic from a server to the users in a non-interactive session. As the Internet bandwidth has increased for home users, it has become easier to stream audio and/or video without having to download and store the

media on the computer.

Research in 2001 [36] by MacInTouch [37] showed that Real Player [38] dominates the market share. The number of unique users of Real Player was more than Windows Media [39] and QuickTime [40] combined. Real Player had about 30% of the market share while Windows Media and QuickTime are only around 10%. However, more recent research [41] by Research and Markets [42] shows that Windows Media has grown to over 50% of the market share followed by Flash [43]. Real Player is down to 9.3% while Quicktime to about 2%. An analysis by *compete* in 2008[1] shows that 49.0% of Web video viewing visits are to YouTube.

Windows Streaming Media (WSM) and Real media have different network characteristics from Flash and QuickTime. WSM and Real media are transmitted at a certain rate either over TCP or UDP [44]. Depending on the available bandwidth, both have mechanisms to adapt their rate. QuickTime can be downloaded using HTTP or streamed using RTP/RTSP [45]. Flash offers three modes of video delivery: embedded video, progressive download, and streaming delivery. A Flash object includes a video in the embedded video mode, while it includes a link to a video in the progressive download mode. In these modes, the video is downloaded to the user over an HTTP connection. A Flash object streams a video using HTTP in the streaming delivery mode. Because of our definition of audio/video streaming, the one-way delay is not a factor in QoS. Therefore, latency tolerance for audio/video streaming traffic is relatively high. Audio streaming uses low bandwidth such as 64-192 Kbps while video streaming can use up to 3.7 Mbps (YouTube 1080p HD content[2]).

As cable companies reach millions of homes to provide more channels than traditional broadcast services, many home users view programs on available channels on their subscription plan. However, it is hard for people to watch the programs once they are away from home on business or vacation. Sling Media developed SlingBox [46] and Sony developed LocationFree [47] as a solution for these customers. These products can deliver TV

[1] http://blog.compete.com/2008/09/04/online-video-share-july-youtube-myspace-blinkx-crackle/
[2] http://www.digitalsociety.org/2009/11/youtube-will-support-1080p-3-7-mbps-next-week/

channels from the home to the Internet. In other words, users can stream cable channels over the IP network to their Internet location outside the home. Orb Networks developed Orb to deliver media files on a computer over the Internet [48]. These home streaming systems are applications with video streaming where the network traffic is mostly upstream and not downstream.

Web browsing Web browsing is perhaps the most popular activity users engage in at home. The Web, also known as the World Wide Web (WWW), was primarily adopted for university-based research early on with the Hypertext Transfer Protocol (HTTP) and Gopher protocol to transfer a file rather than common Hypertext Markup Language (HTML) [49]. Early browsers include Mosaic [50] and Netscape, which has become FireFox/Mozilla today. Commercialization of the Web started in 1996 and continues to grow. Between 1999 and 2001, many startup companies ventured into the WWW during the "Dot-Com" boom. The introduction of Web 2.0 allows browsers to share and exchange information in an ad hoc fashion through technologies such as Asynchronous Javascript and XML (AJAX). Popular browsers today include Internet Explorer [51], FireFox [52], Chrome [53], Safari [54], and Opera [55].

Charzinski analyzes Web data sets collected using Tcpdump [56] on a FreeBSD computer connected to a local area network with client computers and their connection to the Internet [57]. Their traces show that about 50-60% of objects can be transferred in one or two IP data packets. This implies that most of the TCP connections used for Web object transfers do not leave the slow start phase. However, today, the introduction of AJAX has increased the amount of data transferred for a single Web page. AJAX-enabled Web pages keep polling the server for new content without the user moving on to a different Web page. Schneider, Agarwal, Alpcan, and Feldmann show that the amount of data transferred can be as much as 10 MB per connection [58]. The bytes transferred over one HTTP connection for AJAX Web pages is significantly higher than that for non-AJAX Web pages. The data analysis from [58] shows that 81.8% of the HTTP payload was less than 10 KB for complete HTTP traffic while 39.6% of the Google Maps [59] traffic was less than 10

KB. Therefore, we can characterize Web traffic as having relatively low bandwidth but sometimes downloading a large amount of data on AJAX-type Web pages. Web browsing is an interactive activity for users where latency tolerance is relatively low [60]. Because users can take time to read the Web pages, the traffic can appear to be bursty at times as the users click on a link to move to another Web page. Authors show that the inter-request time ranges from 10 ms to 10 seconds for a complete HTTP traffic trace [58].

Downloading files File downloads and emails are also common user activities at home. Users download documents, pictures, videos, and music over the Internet. File downloads are commonly done over TCP connections for reliable transfer of data with various types of application protocols.

Emails are commonly accessed through Post Office Protocol (POP) or Internet Message Access Protocol (IMAP) servers, but Web versions of email are popular today. Services such as HotMail [61], GMail [62], and Yahoo Mail [63] offer email access through browsers. Web emails can be characterized as similar to the Web traffic explained above. However, if users are using an e-mail application to download the e-mails and attachments, the traffic resembles that of a file download.

Users prefer to complete downloads quickly but the latency tolerance is high because they can do other activities while waiting for the download. There is no absolute bandwidth requirement for file download, but this application tries to use all the available bandwidth on the network.

Peer-to-peer networking Traditionally, file transfer was done over HTTP and File Transfer Protocol (FTP), but Peer-to-Peer (P2P) protocols have grown to support file downloads. A P2P survey [64] lists P2P protocols such as Freenet [65], Gnutella [66], FastTrack/KaZaA [67], BitTorrent [68], and Overnet [69]/eDonkey [70]. Network traffic on P2P networks differs greatly from traditional file transfers. As the name suggests, files are transferred from one peer to another. P2P applications not only download parts of a file that a user wants, but also upload files or parts of the file that have been downloaded

already. Most P2P applications use all the available bandwidth for both downloading and uploading and open up hundreds of connections. On a flow-by-flow basis, flows may look like they are using a small amount of bandwidth, but the P2P application may be using a large amount of bandwidth overall. Basher *et al.* analyze Web and Peer-to-Peer traffic [71]. Although P2P applications are designed to share relatively large files and over 80% of flows are larger than 10 KB, there is a small fraction of packets that are less than 5 KB, which carry control information. P2P applications use multiple flows as shown in [72] and [73]. P2P flows transferred data with a mean of 362.40 KB and a median of 1.17 KB while lasting for a mean of 123.54 seconds and a median of 24.80 seconds. Once again, since this is under the category of a file download, delay tolerance is high and all the available bandwidth is used.

Instant messaging Users at home started online instant messaging in the early 1990s through major online services such as America Online (AOL), Prodigy and CompuServe. Instant messaging has become popular with the launch of ICQ developed by Mirabilis in 1996 [74]. ICQ client software allows users to connect to ICQ servers and to communicate with other people online. The major online services adopted the ICQ model and introduced their own instant messaging software. Billions Connected[3] identifies Microsoft Network (MSN) [75], America Online (AIM) [76], ICQ [74], Yahoo [77], Jabber [78], Google (GTalk) [79], and QQ [80] as the most popular instant messaging clients around the world. According to their research in July 2008, about 39% of users in the United States use AIM, 28% use Yahoo, and 26% use MSN [81]. According to research by comScore Media Metrix in September 2004, 53 million adults trade instant messages and 24% of them exchange instant messages more frequently than emails [82]. Not only adults but also teenagers actively use instant messaging [83].

Xiao, Guo, and Tracey collect traces of AIM and MSN instant messaging sessions and analyze the data [84]. In terms of bandwidth, AIM and MSN do not require much bandwidth [84]. There are spikes of bandwidth up to 80 Kbps but these are due to one

[3] http://billionsconnected.com/blog/

or two file transfers and/or video/audio conversations. Most of the bandwidth usage is about 1 Kbps or less for instant messages. In terms of types of messages, there are fewer chat messages compared to hint and/or presence messages to the server that notify the server of users' information such as status. About 90% of the chat messages are less than 50 bytes, which fit in just one packet. The instant messaging users engaged in up to 12 to 15 conversations at a given time. About 95% of the users stayed logged on to instant messaging service for 10 hours or less.

Table 2.1 and Figure 2.1 summarize the network traffic characteristics of user activities discussed thus far. Figure 2.1 shows the relative relationship between delay tolerance and bandwidth usage of each activity. Delay sensitivity and bandwidth usage are generally inversely related. In other words, the higher bandwidth an activity uses, the higher tolerance it has for delay and vice versa. This becomes our key observation that drives the credit mechanism described in Chapter 4.

Table 2.1: User Activities and Network Characteristics

User Activity	Bandwidth	Delay Tolerance
Instant Messaging	low	low
Video games	low	low
VoIP	low	low
Web browsing	low	low
Audio streaming	low	high
E-mail download	low-high	high
Web browsing (AJAX)	medium-high	low
Video conferencing	high	low
File download (FTP)	high	high
File download (P2P)	high	high
Video streaming	high	high

2.2 Wireless Networks

Wireless networks have proliferated in the past decade and are now widely used not only in businesses and academic institutions but also in homes. While applications and protocols

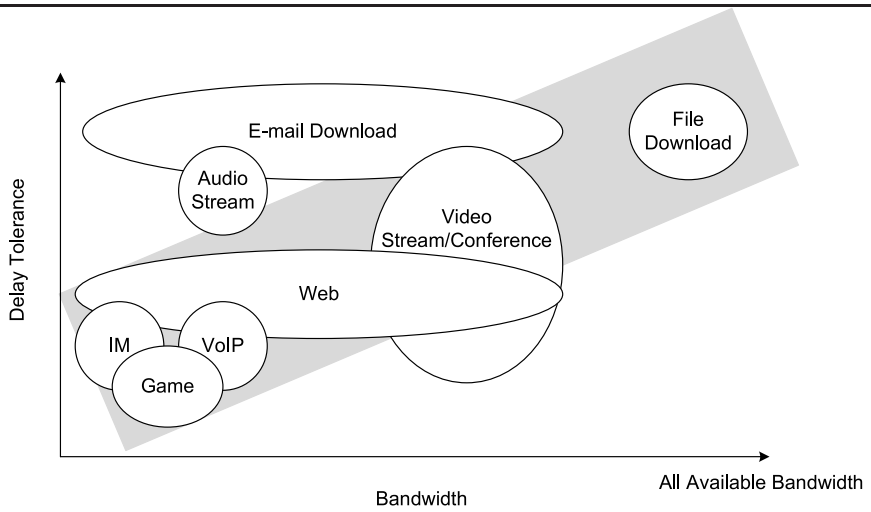


Figure 2.1: Characteristics of Network Applications

in wireless networks also function in wired networks, the characteristics of wireless networks impact their performance. This section reviews wireless networks and introduces Wireless Local Area Networks (WLAN) to illustrate characteristics of wireless networks.

2.2.1 Overview of Wireless Network

All wireless networks use a shared radio medium with potential high bit error rate caused by attenuation, interference, fading and collision. This section summarizes the characteristics of the wireless medium and categories of wireless networks.

- Shared Medium. The wireless medium requires broadcasting unlike in the wired media. All the wireless transmissions share the same medium, resulting in half-duplex communications. Collisions and interference can further degrade network performance. Moreover, the shared medium is regulated and the capacity cannot be increased by adding more media because the wireless is restricted to a limited available frequency band.

- Propagation. Attenuation, reflection, diffraction, and scattering effects can occur during wireless transmission. These effects cause multipath fading, resulting in time-varying channel conditions that affect the application performance perceived by users.

- Bursty channel errors. Wireless transmissions typically suffer higher bit error rate than wired transmissions. The bit error rate can be 10^{-3} or higher due to the attenuation, interference, and fading effects.

All these wireless characteristics may contribute to wireless network performance degradation. Therefore, most of the wireless network protocols implement ways to overcome these effects in wireless networks. Techniques such as Forward Error Correction (FEC), Automatic Repeat reQuest for retransmission (ARQ), and rate adaptation are discussed in Section 2.2.2.

There are five major types of wireless networks:

- Wireless Personal Area Network (WPAN). A WPAN is for networks that have a small range. Bluetooth [85] is an example of protocols used in WPAN. These are usually used with personal mobile devices to communicate with others. A Bluetooth headset is an example of a device in a WPAN.

- Wireless Local Area Network (WLAN). A WLAN is widely used in both businesses and homes because of ease of deployment. The most popular protocol used in a WLAN is IEEE 802.11 [86], reviewed in detail in Section 2.2.2. Relative to a WMAN and a WWAN, a WLAN has shorter range but higher capacities.

- Wireless Sensor Network (WSN). A WSN is composed of spatially distributed sensors that cooperatively monitor conditions such as temperature, luminance, sound and motion. Applications of WSNs includes agricultural and industrial monitoring. ZigBee [87] is one of the protocols used in WSNs. WSN protocols are designed to lower the power consumption.

- Wireless Metropolitan Area Network (WMAN). A WMAN is a wider area network that can stretch out to a metropolitan area, generally the size of a city. WiMax [88] is an example of such a protocol. Unlike cellular networks, WMAN cannot be accessed nationally but is only available in the city where the service is available.

- Wireless Wide Area Network (WWAN). A WWAN has the broadest coverage and widely deployed today via the cellular infrastructure for data transmission. 2.5G (Generation) services such as General Packet Radio Service (GPRS), Enhanced Data Rates for Global Evolution (EDGE) and the next-generation 3G services are examples of techniques used in WWANs.

Out of these wireless network techniques, WLAN is the most widely deployed wireless networks at home. Therefore, the research focuses only on WLANs.

2.2.2 IEEE 802.11 Wireless Local Area Network (WLAN)

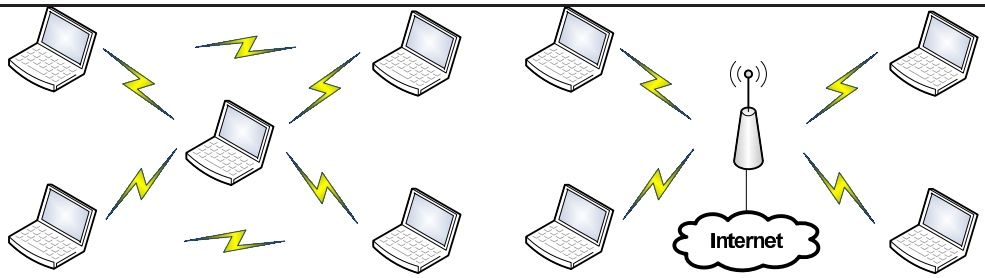
IEEE 802.11 is limited in scope to the Physical (PHY) layer and Medium Access Control (MAC) sublayer. The IEEE 802.11 MAC layer begins with IEEE 802.3 Ethernet standard, while the PHY layer supports a few variations, such as Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), Orthogonal Frequency Division Multiplexing (OFDM), and InfraRed (IR). The list of WLANs defined in the IEEE 802.11 standard [89] includes:

- 802.11 - The WLAN standard was originally 1 Mbps and 2 Mbps, using 2.4 GHz RF and IR (1997). All the other 802.11 standards listed below are Amendments to this standard, except for Recommended Practices 802.11F and 802.11T.
- 802.11a - 54 Mbps, 5 GHz standard (1999, shipping products in 2001)
- 802.11b - Enhancements to 802.11 to support 5.5 and 11 Mbps (1999)
- 802.11c - Bridge operation procedures, included in the IEEE 802.1D standard (2001)
- 802.11d - International (country-to-country) roaming extensions (2001)
- 802.11e - Enhancements: QoS, including packet bursting (2005)
- 802.11F - Inter-Access Point Protocol (2003, withdrawn February 2006)
- 802.11g - 54 Mbps, 2.4 GHz standard (backwards compatible with b) (2003)

- 802.11h - Spectrum Managed 802.11a (5 GHz) for European compatibility (2004)
- 802.11i - Enhanced security (2004)
- 802.11j - Extensions for Japan (2004)
- 802.11k - Radio resource measurement enhancements (2008)
- 802.11n - Higher throughput improvements using MIMO (multiple input, multiple output) antennas (2009)
- 802.11p - WAVE - Wireless Access for the Vehicular Environment (such as ambulances and passenger cars) (2010)
- 802.11r - Fast roaming Working "Task Group r" - (2008)
- 802.11s - Mesh Networking, Extended Service Set (ESS) (2010)
- 802.11T - Wireless Performance Prediction (WPP) - test methods and metrics recommendation (2008)
- 802.11u - Interworking with non-802 networks (for example, cellular) (2010)
- 802.11v - Wireless network management (2010)
- 802.11w - Protected Management Frames (2009)
- 802.11y - 3650-3700 MHz Operation in the U.S. (2008)
- 802.11z - Extensions to Direct Link Setup (DLS) (2007 - 2011)
- 802.11aa - Robust streaming of Audio Video Transport Streams (2008 - 2011)

IEEE 802.11e is discussed further in Section 3.1.

IEEE 802.11 offers two modes: infrastructure and ad-hoc. Figure 2.2 depicts the difference between the two modes. In ad-hoc mode, a group of wireless nodes form a decentralized network where each node forwards data to other nodes to enable data transmission and is



(a) Ad-hoc Mode (b) Infrastructure Mode

Figure 2.2: Ad-hoc and Infrastructure Mode

in charge of routing packets. This is useful for nodes that are in each other's transmission range with data that are just meant for those participating in the network. Ad-hoc mode is used in portable gaming consoles such as Sony PlayStation Portable (PSP) [90] and Nintendo DS [91]. Players on those consoles set up games and play with each other, forming separate ad-hoc networks for each game session.

Infrastructure mode is where all the wireless nodes must communicate with an access point (AP). This is the mode of the WLAN that is widely deployed in businesses and homes. The AP routes all the packets. Thus, the nodes do not communicate directly with each other without going through the AP first. The AP can also be connected to another network or the Internet. Wireless providers offer Internet access "hotspots" to customers at coffee shops, airports and other public areas. Use of wireless APs is an easy way also to connect devices to the Internet because it does not require any physical wiring.

There are three types of frames in IEEE 802.11: management frames, control frames, and data frames. Management frames enable nodes to establish and maintain communication with other nodes. There are many subtypes of management frames: authentication, deauthentication, association request, association response, reassociation request, reassociation response, disassociation, beacon, probe request, and probe response. Beacon frames are sent periodically by the AP to announce its presence and broadcast information such as Service Set IDentifier (SSID) to the nearby nodes. A node can send out a probe request frame to obtain information from other nodes. For example, a laptop can send out a probe request frame to locate the APs within its range. A probe response frame is sent out by

nodes that received a probe request frame. A node uses an association request frame to connect to an AP of its choosing. Then the AP can send back an association response frame to let the node know whether the request is granted or rejected. A reassociation request frame is used to associate to another AP with the same SSID but with a stronger signal. The AP with a stronger signal responds with a reassociation response frame to let the node know if the request is granted or rejected. Authentication and deauthentication frames are used for encryption. With an open system authentication, a node sends an authentication frame to the AP and the AP sends another authentication frame to either accept or reject the node. In a shared key authentication, a node sends an authentication frame and the AP responds with another authentication message containing a challenge text. Then the node has to encrypt it correctly with the shared key and send the encrypted text back to the AP. Depending on the correctness of this encrypted text, the AP sends an authentication frame back to either accept or reject the node. A deauthentication frame is used when a node decides to terminate secure communications.

Control frames help out with the data frame delivery between nodes. There are three subtypes of control frames: Request to Send (RTS), Clear to Send (CTS) and Acknowledgment (ACK). An ACK frame is used to acknowledge successful transmission of DATA frame. Data frames are literally frames with higher layer data. RTS and CTS frames are used in conjunction with each other. A node with data to transmit sends an RTS frame first to see if there is any collision. The destination node responds with a CTS frame to let the sender know that the channel is free. RTS/CTS frames are optional and are not required for data transmissions.

In IEEE 802.11, there exist two mechanisms to control access to the medium: Distributed Coordination Function (DCF) and Point Coordination Function (PCF). DCF is a random access scheme based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and it is supported by all IEEE 802.11 compatible devices. PCF is a centralized protocol that uses a point coordinator to determine which node has the right to

transmit. However, PCF is an optional component and not widely deployed[4].

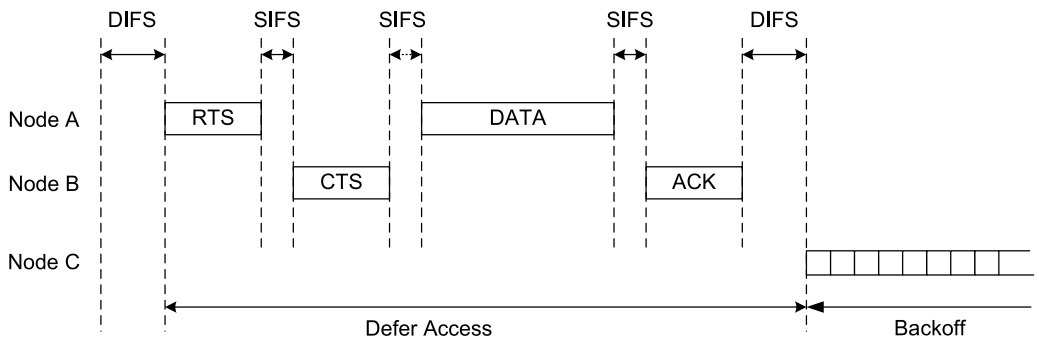


Figure 2.3: 802.11 Frame Transmission

Figure 2.3 depicts a typical transmission of an 802.11 frame with DCF, also showing the optional RTS/CTS transmission. When a node wants to send a frame, it starts to sense the channel for a Distributed Inter-Frame Space (DIFS) to see if the channel is idle. Then it sends out a RTS frame to see if it gets back a CTS frame. After Short Inter-Frame Space (SIFS), the CTS should be received. Reception of the CTS frame indicates that the channel is idle and there will not be any collisions from other stations transmitting. Then the node sends its DATA frame out and should receive back an ACK frame after a SIFS. The node must stay idle for at least DIFS after reception of the ACK frame before attempting to transmit another frame.

IEEE 802.11 supports different maximum number of transmission attempts of RTS and DATA frames based on their frame sizes. The IEEE 802.11 standard suggests that the number of transmission attempts for frames less than the RTS Threshold is seven while it is four for frames larger than the RTS Threshold. The RTS Threshold determines the use of the RTS/CTS mechanism. If the DATA frame is smaller than the RTS Threshold, the transmission occurs without the RTS/CTS exchange. Setting the RTS Threshold larger than the Maximum Transmission Unit (MTU) disables the RTS/CTS exchange for all transmissions. In this case, all the DATA frames are considered short frames and are transmitted at most four times.

[4]http://www.wi-fiplanet.com/tutorials/article.php/1548381/80211-Medium-Access-Methods.htm

IEEE 802.11 supports a multirate physical layer. For example, the Extended Rate PHY (ERP) of IEEE 802.11g supports data rates of 1, 2, 5.5, and 11 Mbps using DSSS modulation and 6, 9, 12, 18, 24, 36, 48 and 54 Mbps using OFDM modulation. Bit Error Rate (BER) and Signal-to-Noise Ratio (SNR) have different relationships based on different modulation schemes and data rates. Even with the same SNR, a lower BER can be achieved by switching the modulation scheme to a different rate, resulting in better wireless network performance. However, the rate adaptation schemes cause dynamic capacity changes in wireless networks and may impact the performance of applications running in this environment. The rate adaptation mechanisms can be based on either sender's inference or receiver's feedback of the current channel conditions.

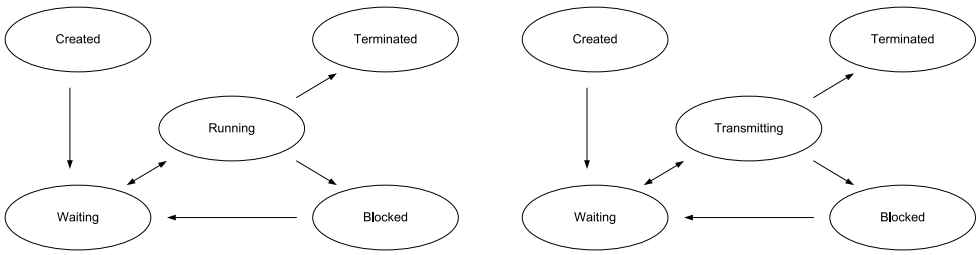
Most residential wireless APs are manufactured by companies such as Linksys [92], Netgear [93], and D-Link [94]. As of January 2009, Linksys, Netgear, and D-link offer both 802.11g and 802.11n access points. They also support 802.11b because 802.11g is backward compatible with 802.11b. 802.11e is specifically designed for QoS and will be discussed in further detail in Section 3.1.

As mentioned previously, wireless APs are used to set up 802.11 wireless networks in infrastructure mode. This is the most convenient way for home users to share their Internet access through one subscription. Figure 1.1 in Chapter 1 shows a typical setup at home. Home users subscribe to a local Internet Service Provider (ISP) to provide access through Cable, DSL, or FiberOptics. The ISPs provide an appropriate modem to provide more common connection to home computers through Ethernet. There are cases where this modem is a wireless AP as well. This modem connects to the ISP through the edge ISP gateway, which can route traffic to and from the Internet. The wireless APs direct traffic to and from the home devices.

2.3 Process Schedulers

Although this research focuses on packet queue management, process scheduling policies are relevant because there are certain similarities. Process scheduling policies are implemented

in the operating systems to manage many processes that are present in the system. Some are running, waiting, and blocking depending on their activity and interaction with the users. This has similarities to packet scheduling policies.



(a) Process States (b) Network Application States

Figure 2.4: Process States and Network Application States

Figure 2.4(a) depicts the states of processes and how a process moves from one state to another, under the assumption of a single processor system. A process requires a number of CPU cycles to complete its work. A process in the running state uses the CPU cycles. A process in the waiting state wants to use the CPU but another process is in the running state. A process in the blocked state does not want to use the CPU and is probably waiting for I/O or other activity to go to the waiting state. Figure 2.4(b) depicts the corresponding states of a network application and their packet states. A network application requires a number of packets to complete its work. Once these packets are transmitted from the application and enter a network packet queue, it can be either waiting or being transmitted. Network applications can also go into the blocked mode if it does not have any data to transmit.

There are many different types of process scheduling policies, but four basic forms are:

- First-Come-First-Serve (FCFS): This is the simplest type of process scheduler. The processes in the queue are served based on the order in which they entered the queue. It is not pre-emptive, and the processes in the queue have to wait until the process using the CPU is finished.

- Shortest Job First (SJF): SJF chooses a process that requires the least amount of

CPU time. However, in practice, it is difficult to estimate how much CPU time each process requires.

- Shortest Remaining Processing Time (SRPT): SRPT is a pre-emptive version of SJF and can remove the process using the CPU if the newly arrived process requires less CPU time.

- Round Robin (RR): Instead of giving the process all the time it needs to complete its job, RR gives processes CPU time in epochs. Processes can only use the CPU for a short amount of time and have to give up the CPU to another process.

- Priority: Instead of every process having equal priority, priorities can be pre-configured for different processes or can be dynamically altered based on behavior. Processes are selected to use the CPU based on their priority.

Traditional UNIX CPU scheduling policy involves a mechanism similar to credits. The traditional UNIX scheduler is a priority queue using round robin within each priority. The priorities of processes can change over time based on their behaviors. This nature of the queue is similar to credits because credit-based schedulers choose the process with the highest priority and the credits are incremented or decremented based on their behaviors. The priority queue of the traditional UNIX uses the following two formulas [95]:

$$CPU_j(i) = \frac{CPU_j(i-1)}{2} \tag{2.1}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + nice_j \tag{2.2}$$

where

- $CPU_j(i)$ = measure of processor utilization by process j through interval i

- $P_j(i)$ = priority of process j at beginning of interval i; lower values equal higher priorities

- $Base_j$ = base priority of process j

- $nice_j$ = user-controllable adjustment factor

The priorities of processes are recomputed every second along with a new scheduling decision. *Base* is used to separate processes into fixed bands of priority levels. *CPU* and *nice* are restricted to prevent a process from leaving its assigned priority band.

Time	Process A		Process B		Process C	
	Priority	CPU Count	Priority	CPU Count	Priority	CPU Count
0	60	0 1 . . 60	60	0	60	0
1	75	30	60	0 1 . . 60	60	0
2	67	15	75	30	60	0 1 . . 60
3	63	7 8 . . 67	67	15	75	30
4	76	33	63	7 8 . . 67	67	15
5	68	16	76	33	63	7

Figure 2.5: Example of a Traditional UNIX Process Scheduling

Figure 2.5 depicts how P is calculated based on CPU cycles used by 3 processes with the same *Base* of 60 and *nice* of 0. Shaded rectangles represent the process picked by the scheduler to use the CPU. Since the lower number has higher priority, the process with the lowest priority is selected by the scheduler. It is shown that P increases (lower priority)

as the process uses the CPU and lowered (higher priority) when it is not selected by the scheduler.

Process scheduling in Linux has evolved through many revisions. Up to Linux kernel 2.4.x, process scheduling involved a form of credits called a *counter*. A process counter is decremented when the timer interrupt is triggered, and the process gets suspended when its counter reaches 0. If there are no runnable processes (meaning that all backlogged processes have 0 for counter), the following equation is used to boost every process' counter:

$$counter = \frac{counter}{2} + priority \tag{2.3}$$

This scheduling policy naturally favors processes that do not require CPU cycles as often because the CPU heavy processes tend to drain their counters to 0, forcing the counter boosts. Those processes that do not need the CPU cycles get the same boost as all other processes even though they have not drained their counter.

The credit mechanism in Linux kernel 2.4.x is the main idea for this research as it favors the processes that do not use the CPU often. Similarly, the home wireless access point must give priority to those that do not use the wireless resources often. This is desirable based on the properties demonstrated in Figure 2.1.

As mentioned above, the credit mechanism for process scheduling is used in Linux kernel 2.4.x. In Linux kernel 2.6.x, the developers implemented a new process scheduler known as O(1) scheduler.

O(1) scheduler is the process scheduling policy used in Linux before the latest Completely Fair Scheduler (CFS). O(1) refers to the fact that it takes constant amount of time to schedule a process. It consists of multiple priority arrays and always picks the highest priority task on a system. If there are multiple tasks at the highest priority, round-robin is used. More specifically, it uses two priority arrays: the active and expired priority arrays. Picking the highest priority task, switching between tasks with round-robin, and making transitions between timeslice epochs[5] are all done in constant time.

[5] An epoch is the time between when all runnable tasks begin with a fresh timeslice and when all runnable

All processes in the O(1) scheduler have a static process priority called *nice* which ranges from -20 to 19. Higher nice value means lower priority. By default, all processes start with a static priority of 0 but these can be modified. O(1) scheduler favors I/O-bound processes and punishes CPU-bound processes by modifying tasks' static priorities. I/O-bound or interactive processes get priority boosts up to 5 while CPU-bound processes get priority penalties up to 5. These processes are assigned in priority arrays based on effective priorities that are calculated by mapping a process' sleep average onto the range 0-10. Then the timeslices are calculated by increasing the minimum timeslice by an amount mapping a process' static priority to the possible time slice range (difference between minimum and maximum timeslice).

Completely Fair Scheduler (CFS) is the latest addition to Linux as the new process scheduling policy. It is also known as the Rotating Staircase Scheduler. Unlike previous implementation of process schedulers, CFS is based on a red-black tree, which implements a *timeline* of future task execution. Moreover, CFS has a granularity of nanoseconds, the atomic unit of an individual process' share of the CPU.

Both O(1) and CFS employ structures and processing that may be a challenge for typical APs to support. Moreover, it is not clear that the added complexities have benefits over the more simple 2.4.x kernel scheduler.

2.4 Active Queue Management

Active queue management (AQM) is a technique to manage a network packet queue during congestion. Typically, a drop-tail queue drops incoming packets when the FIFO queue is at its limit. However, this causes bursty losses of packets and degrades network performance. Therefore, researchers came up with a variety of techniques to manage the queue before it reaches the limit. Because AQM techniques manipulate the queue before it reaches its limit, there is flexibility to either drop or mark incoming packets. Marking is a way to signal the sender to slow down without dropping packets. However, marking only works

tasks have used up their timeslices.

with flows with Explicit Congestion Notification (ECN) enabled.

The functions of AQM can be divided into: [96]

- Congestion Monitor. Responsible for congestion detection and estimation.
- Bandwidth Controller. Responsible for output bandwidth management.
- Congestion Controller. Responsible for computation and application of the congestion notification probability (CNP) to incoming traffic.
- Queue Controller. Responsible for merging buffer usage and packet scheduling.

The first task of AQM is efficient monitoring, detection, and estimation of congestion. There are different measurements to estimate and detect congestion. Queue length and traffic load are two high-level measures for congestion estimation. Queue length estimates can be instantaneous or averaged, and be compared to a threshold to detect congestion. Traffic load can be measured using various methods that rely on queue length or incoming rate, and compared with service rate to estimate congestion.

An AQM may have a bandwidth controller for QoS guarantees. The most common functionality of a bandwidth controller is to provide fairness protection for individual flows or groups of flows. This can be accomplished on a per-class or per-flow basis. The per-class method has less overhead compared to per-flow because a class is a group of flows. Per-flow information is much larger in aggregate than per-class information to maintain. Because of the great overhead in a per-flow method, a pseudo per-flow method can also be used to reduce the overhead required. Instead of keeping information on all flows, a pseudo per-flow method tracks only outstanding high bandwidth flows to minimize per-flow information. In addition to fairness protection, an AQM can also support priority forwarding and loss differentiation. Priority forwarding drops packets from lower priority flows before higher priority flows upon congestion. Loss differentiation specifies a predefined drop rate for each class.

The job of the congestion controller is to prevent or control network congestion by notifying traffic sources of the impending congestion early so that congestion responsive

sources such as TCP have opportunities to reduce their rates. Traditionally, packet loss is used as notification to the sources that there is congestion. However, a packet drop is costly enough to reduce network performance. Therefore, Explicit Congestion Notification (ECN) has been developed to mark packets for congestion notification instead of dropping. AQMs can either drop (implicit) or mark (explicit) packets using a congestion notification probability (CNP). CNP can be uniform, per-class, or per-flow.

The last component of an AQM is a queue controller. A queue controller is responsible for packet transmissions forwarded by the congestion controller or bandwidth controller. AQMs can utilize a single queue or multiple queues based on their goal. Each queue can either use First-In-First-Out (FIFO) discipline or other disciplines to serve their purpose.

One of the early AQM techniques is Random Early Detection (RED) [97], which keeps track of an exponentially averaged queue size for congestion monitoring. RED does not use a bandwidth controller but applies a uniformly increasing CNP based on the average queue length in its congestion controller. RED starts to drop packets when the average queue size exceeds minimum threshold. RED increases the CNP linearly from 0 to p between minimum and maximum threshold. A variation of RED called Random Early Marking (REM) [98] marks packets instead of dropping them for notification. It simply uses a single FIFO queue for its queue controller. Later versions of RED support a "gentle" QoS queue controller mechanism. These versions and other AQM techniques with QoS support will be reviewed further in more detail in Section 3.2

2.5 Quality Metrics

The diversity of QoS requirements for applications makes it difficult to use one metric to measure their quality. For example, packet loss causes significant degradation in the quality of VoIP applications while it has minimal effect on the quality of online games. This section discusses the quality metrics used to determine the quality of the following simulated applications: game, VoIP, video, Web, File Transfer Protocol (FTP), and Peer-to-Peer (P2P).

There is no standard way of measuring the quality of online games. Gamers often talk about the game performance with regards to "latency" and "frame rates" reported in their games. While these contribute to the overall quality of game, it does not capture all factors that contribute to the quality of game performance. Because the game application in simulation follows the traffic model for Halo 2, a popular First Person Shooter (FPS) game, the perceived quality of online games is calculated by the G-model Mean Opinion Score (MOS) [26], developed for Quake IV, another popular FPS game. The G-model MOS depends on two parameters: the average ping (round-trip time) and jitter. Although jitter usually refers to a variation in inter-arrival times between packets, the G-model MOS uses a different measurement for the average jitter. The average jitter for the G-model MOS is the difference of the average one-way delay of packets and the minimum one-way delay.

$$X = 0.104 \times ping + jitter \tag{2.4}$$

First, the network impairment (X) is given by Equation 2.4. The coefficients of the average ping and jitter indicate that the average jitter has greater impact on the network impairment. Authors found that packet loss hardly affects the perceived quality. The network impairment is mapped to the G-model MOS by Equation 2.5.

$$G(X) = -0.00000587 \times X^3 + 0.00139 \times X^2 - 0.114 \times X + 4.37 \tag{2.5}$$

The G-model MOS ranges from 0 to 4.37, where 5 is the best quality and 0 is the worst quality.

Unlike for online games, there is a widely accepted way of measuring the perceived quality of VoIP sessions called E-model [99]. While the quality of online games depends on the average round-trip time and jitter, the quality of VoIP sessions depends on the average delay and loss. The E-model R-Factor (R), based on degradation due to delay ($i(d)$) and loss ($i(l)$), is given by Equation 2.6.

$$R = 94 - i(d) - i(l) \tag{2.6}$$

The degradation due to delay (id) can be computed by Equation 2.7. As long as the one-way delay (d) is less than 177.3 ms, the degradation is minimal. Delays greater than 177.3 ms increases the degradation. id includes the degradation caused by packet jitter. One of the ways to mitigate the variability in inter-arrival times of packets is to place a buffer to hold packets before delivering them to the VoIP application. While it compensates for the variability, it increases the delay in delivering packets to the application. Therefore, the one-way delay (d) includes the buffer delay in addition to the network delay.

$$i(d) = \begin{cases} 0.024 \times d, & d \leq 177.3 \\ 0.024 \times d + 0.11 \times (d - 177.3), & d > 177.3 \end{cases} \tag{2.7}$$

The degradation due to loss can be calculated using Equation 2.8. There is no degradation in quality with the loss of 0. As the loss (l) increases, the degradation ($i(l)$) grows logarithmically.

$$i(l) = 30 \times log(1 + 15 \times l) \tag{2.8}$$

The E-model R-Factor is mapped to the E-model MOS with Equation 2.9. While the E-model R-Factor ranges from 0 to 94 where 94 is the best quality and 0 is the worst quality, the E-model MOS ranges from 1.0 to 4.5 where 4.5 is the best quality and 1.0 is the worst quality.

$$E(R) = 1.0 + 0.035 \times R + 0.000007 \times R \times (R - 60) \times (100 - R) \tag{2.9}$$

There are various ways to determine the perceived quality of video streaming. The video application in simulation is implemented to deliver frames over UDP following known traces of encoded videos. Under the assumption that the video application can allocate enough buffer to play the video smoothly, the playable frame rate is used to measure the

quality of video streaming. The known traces of encoded videos include three types of frames: I, P, and B. A frame is received successfully if all the packets of the frame are received or the Forward Error Correction (FEC) data can be used to repair the frame. The FEC mechanisms used in simulation are detailed in Section 6.1.3. A received I-frame is independently playable. A received P-frame is playable if and only if the previous I or P frame is playable. A received B-frame is playable if and only if the previous and next frames are playable. The playable frame rate can range from 0 to the encoded frame rate where 0 is the worst quality and the encoded frame rate is the best quality.

The quality of the Web browsing activity is measured in response time. Response time refers to how long it takes for a user to retrieve all the objects of a Website after entering the Website address or clicking on a link. The shorter the response time, the higher the quality of Web browsing.

The purpose of the FTP and Peer-to-Peer (P2P) applications is to download files to users' devices. The quality of these applications is measured in throughput because the throughput affects the time required to complete the file downloads. Higher throughput results in faster downloads and vice versa. Therefore, high throughput corresponds to high quality.

2.6 Summary

This chapter reviews fundamental techniques and terminologies referenced in the dissertation. Section 2.1 provides the network characteristics of user activities such as online games, audio conferences, video streaming, Web browsing, and file downloads. The investigation reveals that there is a directly proportional relationship between delay tolerance and bandwidth usage for most of the activities. Section 2.2 provides an overview of wireless networks. It focuses on the IEEE 802.11 Wireless Local Area Networks (WLANs), and describes the frame exchange defined by the IEEE 802.11 standard. Section 2.3 demonstrates the similarities between process and packet scheduling policies. It focuses on the Unix and Linux CPU schedulers, part of which is relevant to the CHAP approach in Chapter 4.

Section 2.4 provides an overview of Active Queue Management (AQM) techniques. Section 2.5 explains quality metrics used for user activities such as online games, Voice over IP (VoIP), video streaming, Web browsing and file downloads. Mean Opinion Scores (MOS's) based on G-model and E-model are used for online games and VoIP, respectively, while the playable frame rate and response time are used for video streaming and Web browsing, respectively. The quality of file downloads is determined by the application throughput.

Chapter 3

Related Work

This chapter reviews research work closely related to this dissertation. Section 3.1 reviews IEEE 802.11e and other wireless enhancements designed to improve QoS in home networks. Section 3.2 reviews AQM techniques developed to improve application QoS. Section 3.3 reviews traffic classification techniques to differentiate individual flows, and Section 3.4 reviews credit-based AQM techniques.

3.1 Wireless Home Networks and QoS

This section reviews 802.11e and other home network enhancement techniques to improve QoS in the wireless environment. All these techniques involve explicit classification of network traffic and require an external party to mark frames/packets in advance. Since one of the strengths of CHAP is automatic classification, these techniques are not compared to CHAP in terms of performance.

3.1.1 IEEE 802.11e

IEEE 802.11e is an amendment to IEEE 802.11 that focuses on improvement of QoS. Its aim is to support IntServ and DiffServ architectures.

QoS levels are added to 802.11e in order to distinguish different classes of traffic and access categories. Table 3.1 lists all the priority and access category levels with corre-

Table 3.1: Priority and Access Categories of IEEE 802.11e

Priority	Access Category	Designation
0	0	Best Effort
1	0	Best Effort
2	0	Best Effort
3	1	Video Probe
4	2	Video
5	2	Video
6	3	Voice
7	3	Voice

sponding types of service. Priorities from 3 to 7 are considered high priority classes while priorities from 0 to 2 are considered low priority classes. While 802.11 treats all traffic classes equally, 802.11e uses classification based on these priorities and uses a scheduler to resolve any virtual collisions due to frames from different classes.

To accommodate the newly added priority classes in 802.11, 802.11e introduces enhanced versions of Distributed Coordination Function (DCF) and Point Coordination Function (PCF) functions that are backward compatible. Enhanced DCF (EDCF) is the new version of DCF and uses different parameters for different traffic classes and access categories. It replaces DIFS with Arbitration Inter-Frame Space (AIFS), which is longer than DIFS. AIFS is shorter for audio and video traffic (access categories 1 and 2) where as AIFS is at least DIFS. Moreover, the Contention Window (CW) has different values for audio, video, and best effort traffic. CW_{min} and CW_{max} are smallest for audio and largest for best effort. Table 3.2 shows the corresponding values for CW_{min}, CW_{max}, and AIFS of different access categories.

Table 3.2: Typical QoS Parameters of 802.11e

AC	CW_{min}	CW_{max}	AIFS
0	CW_{min}	CW_{max}	2
1	CW_{min}	CW_{max}	1
2	$\frac{CW_{min}+1}{2} - 1$	CW_{max}	1
3	$\frac{CW_{min}+1}{4} - 1$	$\frac{CW_{max}+1}{2} - 1$	1

Hybrid Coordination Function (HCF), an enhanced version of PCF, is used on top of EDCF. Like PCF, HCF works in a Contention Period (CP) and a Contention-Free Period (CFP). Using HCF, the Hybrid Coordinator (HC) can allocate transmission opportunities (TXOP). During CP, a node can transmit if and only if the medium is available according to EDCF rules or the HC sends the node a QoS CF-Poll frame. The HC can send a QoS CF-Poll frame without back-off if it determines that the medium is idle for PIFS. Therefore, the HC is capable of allocating TXOP using its priority medium access during CP. During CFP, the HC determines the starting time and duration of TXOPs using QoS CF-Poll frames. In other words, the nodes do not attempt to transmit and access the medium unless TXOP is granted by the HC during the CFP. The CFP ends either when the HC sends out a CF-End frame or after the time specified in the beacon frame.

802.11e requires the specific traffic and access categories to be set to guarantee the QoS network traffic needs. It is possible that a greedy node sets all its traffic with the highest priority traffic class to dominate the medium access and also that all the nodes use the highest priority class in which the priorities are the same for everyone. Kyasanur and Vaidya suggest that the traffic parameters are set by the HC, instead of individual nodes, as a possible solution to this greedy problem [100].

If the traffic classes are set by the individual nodes, each application needs to indicate the class it needs to use and this traffic class will remain static. If the traffic classes are set by the HC, the HC needs to know the nature of different types of traffic to be able to classify them correctly. Once the class is determined, it is likely to remain static as well. As mentioned before, static priorities for a type of traffic can degrade performance due to the dynamic nature of the wireless conditions. Giving high priority to delay-sensitive traffic from a node with poor connectivity can degrade overall performance of the wireless network.

3.1.2 Home Network QoS Enhancement Research

Cuomo [101] proposes an architectural model for QoS support in home networks. Cuomo states that home networks are complex in the sense that many devices are connected over a variety of physical media: RF, infrared, and Ethernet, and different applications on different devices require a wide range of QoS. The proposed QoS solution consists of three parts [101]:

- mapping of WAN QoS parameters into the Home Area Network (HAN) parameters and vice versa;
- marking of packets entering the HAN, as a function of the selected QoS parameters;
- a scheduling mechanism in the Residential Gateway (RG) that differentiates flow performance in order to satisfy QoS requirements.

Cuomo assumes that QoS requirements of applications are marked in the WAN at layer 2 or 3 by the existing architecture. The proposed approach maps the QoS marks from the WAN to layer 3 marks in two steps. First, it does a Network Terminator adaptation, which is a WAN-LAN mapping. It maps the WAN QoS marks to IEEE 802.1p [102] priorities that range from 0 to 7, a layer 3 to layer 3 mapping. Lastly it does RG adaptation, which is a LAN-HAN mapping. Then it maps IEEE 802.1p priorities to IP DSCP (Differentiated Service Code Point) [103], a layer 2 to layer 3 mapping. After mapping, there are two proposed scheduling disciplines:

- PFIFO (Packet-limited FIFO), which uses three "bands" to enforce a strict priority queueing discipline. The packets are assigned to different bands based on the values in the Type of Service field in the IP headers.
- Hierarchical Token Bucket (HTB), which creates a number of classes and assigns priorities and service rates to each class. HTB divides the available bandwidth into multiple classes and specifies the average rate to be guaranteed and a maximum

rate for each class. Child classes can borrow bandwidth from parent classes if there exist unused resources. Packets are assigned to different classes based on their DSCP marks as well as their IP addresses, protocols, and ports.

Palazzi *et al.* propose a mechanism to tweak the wireless MAC layer and to exploit existing features of TCP [104]. Two parameters of interest to the authors in the IEEE 802.11 layer are the number of retransmissions and the buffer size. The authors state that these parameters were determined when TCP traffic was dominant and claim that these need to be changed to accommodate real-time application traffic, most of which is UDP. In addition, the authors propose to limit the sending window of TCP flows to prevent probing for more bandwidth. The maximum sending rate of a TCP flow is calculated by taking the difference of the capacity of the link and sum of UDP traffic bandwidth and then dividing the difference by the number of TCP flows. Palazzi *et al.* simulate their proposed approach using NS2 and demonstrate that the combination of tweaking IEEE 802.11 parameters and limiting the TCP advertised window helps to reduce one-way delay and jitter of real-time applications with a small degradation in TCP application performance.

Rubino, Varela, and Bonnin propose enhancements at three levels [105]:

- The Application level. Addition of Forward Error Correction to repair incorrectly received or lost packets instead of invoking retransmissions.
- The MAC level. Use of IEEE 802.11e for priority of classes and different MAC layer parameters.
- The IP level. Use of a Differentiated Service (DiffServ) architecture.

IEEE 802.11e was described early in this chapter and the DiffServ architecture will be explained in detail in Section 3.2. Application level enhancement is beyond the control of home network devices because the application developers must incorporate modifications into their programs. Rubino *et al.* demonstrate that the home network performance improves with the addition of enhancements at all three layers [105]. The performance metric

they developed is Pseudo-Subjective Quality Assessment (PSQA), which is computed in real time with high accuracy that resembles quality perceived by users.

All the approaches described in this section require explicit classification of traffic and pre-determined treatments for each class. Packets need to be marked for a specific class of traffic, which maps to a specific treatment. None of these techniques consider the effects of giving priority to nodes with poor connectivity, which may degrade overall home network performance.

3.2 Active Queue Management for QoS improvement

Section 2.4 discussed tasks of AQMs and this section reviews a subset of AQMs that support QoS. Like techniques discussed in the previous section, all these AQM techniques also require explicit marks from either edge routers or end hosts. Therefore, these will not be used for performance comparison with CHAP.

Quality of Service (QoS) has become an interesting research field because elasticity of application quality varies with regards to different performance metrics. Elasticity refers to how quality of application performance observed by users adjusts to available bandwidth and experienced delay. Different applications offer different elasticity for quality. To maximize application QoS for users, Integrated Services (IntServ) and Differentiated Services (DiffServ) have been proposed.

IntServ is composed of four components:

- Type of commitment: Services IntServ offers
- Packet scheduling: Methods IntServ uses to provide the services
- Service interface: Methods applications use to request the desired services
- Establishing the guarantee: Admission control for new applications

Type of commitment depends on the network characteristics of the applications. As reviewed in Section 2.1, some applications prefer lower delay over higher throughput while

other applications prefer higher throughput over lower delay. IntServ classifies traffic using two dimensions: *tolerance* and *rigidness*. The difference between tolerant and intolerant traffic is whether the applications can handle brief interruptions. The applications that cannot handle brief interruptions are intolerant. Rigidness is determined by how adaptive the applications are with regard to deadlines. An application is rigid if packets must meet a fixed deadline while an application is adaptive if it can deal with changes in network conditions. Although combinations of two dimensions provide four possibilities, most applications fit one of two combinations: intolerant and rigid, or tolerant and adaptive. Therefore, IntServ offers three classes of services to accommodate these two combinations plus a best effort service. *Guaranteed service* is for intolerant and rigid applications, and offers fixed guarantees as long as applications match traffic agreements. *Predictive service* is for tolerant and adaptive applications and takes steps to minimize performance degradation. *Best effort service* serves all traffic that does not subscribe to either of the former services.

The packet scheduling component is responsible for delivering the promises made in each class. A token bucket filter is used to characterize network traffic. Token buckets have two parameters: the rate r at which tokens are inserted into the bucket, and the depth b, which is the capacity of bucket. The bucket is constantly being filled at rate r and tokens are discarded if the bucket is full. Transmitting a packet of size P uses P tokens from the bucket. As long as there are enough tokens in the bucket, the packets can be transmitted at the maximum rate. If there are insufficient tokens in the bucket, the packets need to wait until enough tokens accumulate in the bucket. In the long run, the rate of traffic from the bucket is limited to rate r. In the short run, a burst of size b can be sent at the maximum rate. Therefore, the maximum amount of traffic in interval T can be bounded by $b + rT$. In addition to a token bucket filter, Weighted Fair Queueing (WFQ) is used to handle each traffic flow. Parekh [106] [107] proves that the worst case queueing delay is $\frac{b}{r}$ for any class of traffic in guaranteed service under the assumption that the incoming traffic rate is less than the outgoing link speed at any router.

For predictive service, IntServ needs to be able to isolate well-behaved traffic from misbehaving traffic and to mix different traffic in a way to maximize benefits to all traffic. WFQ used for guaranteed service can provide isolation but no sharing among traffic. FIFO can provide sharing but no isolation. However, FIFO can amplify jitter when there are multiple hops between source and destination. FIFO+ is an extension to FIFO to minimize this effect. FIFO+ measures average delay of traffic classes at a router. For each incoming packet, FIFO+ computes the difference between the queueing delay of the packet and the average delay of the traffic class to which the packet belongs. Then, FIFO+ inserts the packet into the queue based on the difference. Therefore, FIFO+ can reduce jitter significantly by providing tighter control of delay at each router.

IntServ uses WFQ and FIFO+ to serve all three service types. Guaranteed service flows get their own queue while predictive service and best effort flows aggregate into a single, separate queue in WFQ. This single, separate queue uses multiple FIFO+ queues separated by their priorities. Best effort traffic uses the lowest priority queue.

Applications in each service type uses different interfaces for specifying their needs. An application in guaranteed service specifies its rate to IntServ. a higher rate to overcome delay. An application in predictive service specifies delay D and loss rate L in addition to both rate r and bucket size b. Based on this information, IntServ assigns a priority class, and edge routers can either drop or tag packets to provide isolation between priority classes.

The DiffServ architecture, depicted in Figure 3.1, serves network traffic based on Service Level Agreements (SLA) between adjacent domains. Edge routers are responsible for traffic conditioning while core routers process packets based on packet marks and Per-Hop Behaviors (PHBs). This separation of pushing heavy tasks to edge routers and keeping core routers' tasks simple makes DiffServ much more scalable than IntServ.

Per-Hop Behavior specifies the behavior of individual routers and not services. It is possible to have more services than behaviors specified in the router. The behavior that services desire is marked inside the IP header using six bits of the TOS field. Currently,

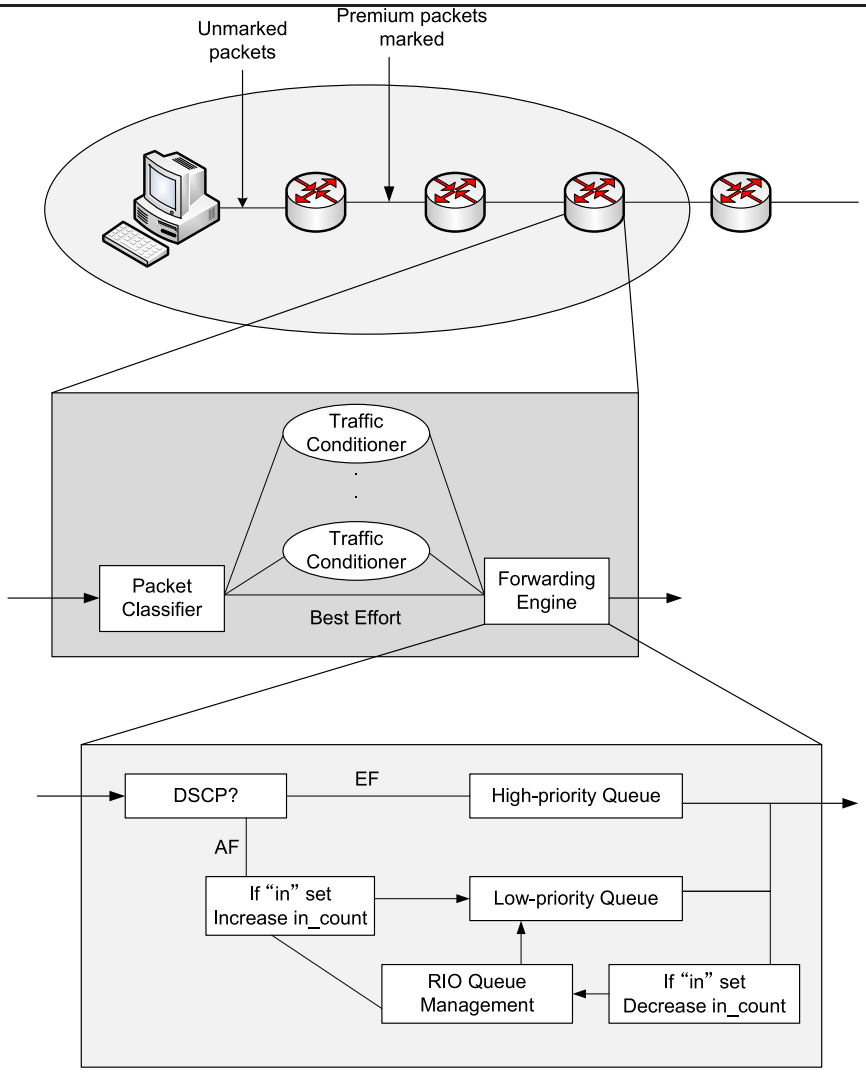


Figure 3.1: DiffServ Architecture

there are two types of PHBs defined: Expedited Forwarding (type P) and Assured Forwarding (type A). Expedited Forwarding is equivalent to having a virtual wire between end hosts. The source node requests a profile and the network commits to delivery as long as the node sends within its requested profile. The traffic is limited to rates between edge routers using a token bucket filter. Packets using Assured Forwarding get transmitted with minimal delay and loss up to the router's capacity. Unlike Expedited Forwarding, users and networks agree to some profile. Edge routers mark packets based on drop precedence values. Packets from network traffic behaving within the profile parameters are marked

with low drop precedence. All other packets are marked with high drop precedence. If a DiffServ node gets congested, packets marked with higher drop precedence are discarded to protect packets with lower drop precedence value. The drop mechanism is implemented with RIO (Red with In or Out) [108], which uses two probability functions for In-Profile and Out-of-Profile traffic. In addition, RIO keeps track of two exponentially averaged queue sizes: one of all packets, and the other of In-profile traffic packets only. The parameter min_{thresh} for Out-of-profile traffic is lower so that packets from Out-of-profile traffic are dropped before packets from In-profile traffic. As the average queue size increases, it will eventually cross min_{thresh} for In-profile traffic and start dropping packets from In-profile traffic as well.

Class-Based Queueing (CBQ) [109] and Class-Based Threshold (CBT) [110] are queue management strategies to guarantee bandwidth per traffic class by allocating fixed amounts of bandwidth to each traffic class. CBQ is a queue controller with multiple FIFO discipline queues. CBT is an extension of RED with a bandwidth controller with a "threshold" test. Unlike CBQ, CBT uses a single packet queue and allocates a pre-determined buffer size for each traffic class. The average buffer usage is monitored against the thresholds and CBQ starts dropping packets from classes whose average buffer usage exceeds the threshold.

Dynamic-CBT (D-CBT) and ChIPS [111] extend CBT by allocating proportional buffer space to the number of flows in each traffic class instead of using pre-determined values. First, D-CBT uses the mechanism of counting the number of active flows from Fair Random Early Drop (FRED) [112]. Then D-CBT uses the number of flows to compute the fair share of buffer space for each flow. D-CBT monitors the average buffer usage of each flow and compares it against its fair share of buffer space and drops the packets if the traffic uses more buffer space than its fair share. ChIPS (Cut-In Packet Scheduling) is an alternative packet scheduling for D-CBT to minimize multimedia jitter. ChIPS inserts the tagged UDP packets at the average queue length instead of at the end of the queue. Therefore, the tagged UDP flows tend to get queueing delays around the same average queue length, reducing jitter.

Alternate Best Effort (ABE) [113] identifies two classes of network traffic: *delay sensitive* (green) and *throughput sensitive* (blue). ABE's goal is to provide lower delay for green traffic while providing comparable throughput to blue traffic. Therefore, ABE forces higher loss rate to green traffic for the price of lower queueing delay. Green traffic is associated with a certain delay bound and packets from green traffic are dropped if they do not meet the delay bound inside the queue. AQM techniques like RED can be used on top of ABE to provide congestion monitoring.

Rate-Delay (RD) network services [114] provides two classes: an R (rate) service and a D (delay) service. The R service emphasizes high transmission rates while the D service supports low queueing delays. IP packets are marked using a bit in the Type-of-Service (ToS) field to indicate which service they desire. The unmarked packets correspond to the R service class. An RD router estimates the number of flows every T interval using the timestamp vector algorithm. Based on k (ratio of per-flow rates of R and D flows) and d (upper limit on queueing delay of D packets), the RD router adjusts the buffer allocated to the D service class. Because the buffer allocated to the D service class is smaller than the buffer allocated to the R service class, the flows in the D service class experience higher loss rates than the flows in the R service class.

Both ABE and RD require the traffic to be classified by an external entity or marked by the source for these mechanisms to work properly. In addition, neither ABE nor RD can distinguish the wireless conditions of the destination nodes of delay-sensitive traffic. Imposing a higher drop rate on delay-sensitive traffic when the destination has poor connectivity may result in application performance degradation.

RED-Worcester and RED-Boston are extensions of ARED [115] to support QoS for different classes of traffic [116]. Both RED-Worcester and RED-Boston require the applications to inject delay preference as hints in their packets. RED-Worcester uses these delay hints to calculate *target* based on capacity and packet size ($target = C\frac{D}{P}$ where C is capacity, D is delay hint and P is packet size). *target* is exponentially averaged with a weight of 0.02. If most of the packets that pass through RED-Worcester are marked

with low delay, average *target* decreases. If most of the packets are marked with high delay, average *target* increases. RED-Worcester provides queueing delay based on what the majority of packets require.

RED-Boston takes advantage of delay hints in a slightly different manner. RED-Boston provides different queueing delay for different classes of traffic instead of the same average queueing delay for all packets. The drop probability p is calculated following the RED mechanism. RED-Boston takes this probability p and scales it based on the delay hint. RED-Boston keeps track of the average delay instead of average *target*. This average delay is used against delay hints to scale the probability using the following formula: $p' = p\frac{d_{avg}}{D}$ where p is the normal probability calculated by RED, d_{avg} is the average delay and D is the delay hint. In other words, flows in a delay-sensitive traffic class have a higher drop probability than flows in a throughput-sensitive traffic class. In addition, RED-Boston sorts incoming packets by weights based on their delay hints if they are not dropped. The *weight* of a packet is calculated by adding the arrival time and the delay hint of the packet. This mechanism provides RED-Boston with automatic aging and protection from packet reordering.

The priority queue discipline uses strict priority of each class specified inside the queue. Packets are enqueued in the corresponding priority queues based on their tags. Packets from the highest priority queue are served first before packets from the next highest priority queue. Although this queueing discipline is simple and easy to implement, it is possible for packets from the lower priority queues to starve. The Probabilistic Priority Discipline queue attempts to overcome the starvation by applying specific priorities to each priority queue which each queue uses to determine if a packet from the queue is to be served.

Context-Aware Transport/Network Internet Protocol (CATNIP) [117] is a combination of protocol and queue management techniques to improve performance of network applications. The context awareness is added to a baseline Reno TCP configuration in the following forms:

- Rate-Based Pacing of the Last Window (RBPLW). A TCP source may choose to

spread out the transmissions of the last few packets over a short interval if it is aware of the number of packets remaining in its last window of data. This may reduce the risk of packet losses.

- Early Congestion Avoidance (ECA). A TCP source may choose to switch from the TCP slow-start algorithm to the TCP congestion avoidance algorithm if it is aware of the number of packets remaining. Linear growth of the congestion window may reduce the risk of packet losses with a small impact on transfer times.

- Selective Packet Marking (SPM). A TCP source may mark packets with "high" or "low" (default) priority if it is aware of how many packets there are in the data transfer. The congestion window is small (i.e., less than 4 packets) for data such as Web documents. The losses of first and last few packets impact the transfer time the most and therefore, the TCP source can mark these packets with "high" priority while marking the rest of the packets with "low" priority. These priorities refer to packet loss priority, not packet scheduling priority.

Wu *et al.* evaluate the performance of CATNIP with five queueing algorithms. Drop Tail and RED are CATNIP-unaware while CATNIP-Good, CATNIP-Bad and CATNIP-RED use information from CATNIP to influence their drop probabilities. CATNIP-Good and CATNIP-Bad are based on the Drop Tail queueing algorithm. No packets are dropped until the queue is filled, just like a Drop Tail queue. However, depending on the SPM of the packet, the incoming packet may or may not be dropped. If the incoming packet is marked with "low" priority, CATNIP-Good drops the packet normally. If the incoming packet is marked with "high" priority, CATNIP-Good drops a "low" priority packet in the queue to make room for the incoming "high" priority packet. If there is no "low" priority packet in the queue, the incoming packet is dropped. CATNIP-Bad follows the same algorithm as CATNIP-Good except it treats "low" and "high" priority packets in the opposite manner. In other words, it prioritizes "low" priority packets instead of "high" priority packets. CATNIP-RED uses RED's congestion detection mechanism and it drops

a "low" priority packet if there is an attempt to discard a packet from the RED algorithm.

All the techniques mentioned above have the same goal of prioritizing a certain class of traffic to improve quality of applications in that class. However, the common shortcomings of each technique is that the traffic needs to be tagged or marked so that the queue management technique can differentiate and treat the traffic accordingly.

3.3 Traffic Classification

This section reviews real-time traffic classification techniques. Traffic classification techniques are useful not only for trying to provide better QoS but also traffic shaping, as well as intrusion prevention and detection. Although one of the goals is to improve QoS, all these techniques must be tied to a treatment module to provide better performance. Since the purpose of CHAP is to improve overall quality of applications and not to classify network flows individually, these techniques are not used to compare the performance of CHAP.

3.3.1 Port-based Classification

The most common identification technique involves the analysis of the complete or partial packet header. The packet header includes information such as IP addresses, and the transport protocol and ports of two end hosts. Based on the header information, the application type and characteristics can be identified for a flow, especially when the flow is connected over a "well known port". Table 3.3 lists some "well known" transport protocols and ports used by several applications.

Table 3.3 only lists ports used by several common applications, where the official list of port assignment can be found at Internet Assigned Number Authority (IANA)[1]. Port-based approaches are the simplest among all traffic classification techniques and highly scalable since the port number from a single packet can be enough to identify the application. However, port-based classification can be inaccurate. Inaccuracy results from mis-matches

[1] http://www.iana.org/assignments/port-numbers

Table 3.3: Transport Protocols and Ports Used by "Well Known" Applications

Keyword	Port#	Protocol	Description
ECHO	7	TCP, UDP	Echo Protocol
FTP-Data	20	TCP	File Transfer [Default Data]
FTP	21	TCP	File Transfer [Control]
SSH	22	TCP, UDP	SSH Remote Login Protocol
Telnet	23	TCP, UDP	Telnet protocol - unencrypted text communications
SMTP	25	TCP	SMTP - used for e-mail routing between mail servers
Name	42	TCP, UDP	Host Name Server
DNS	53	TCP, UDP	Domain Name System
HTTP	80, 8080	TCP	HTTP - used for transferring web pages
POP3	110	TCP	POP3 - used for retrieving E-mails
IRC	194	TCP	Used for Internet Relay Chat
SMB	445	TCP	Used for Microsoft SMB file sharing
RTSP	554	TCP, UDP	Used for Real Time Streaming Protocol
MMS	1755	TCP, UDP	Used for Windows Media Services

between the server ports and the applications. For example, many Peer-to-Peer (P2P) applications allocate ports dynamically and these ports are not registered at IANA. Moreover, the inclusion of firewall capability in wireless access points forces applications to use well-known ports of other applications. Although port 80 is a well-known port for HTTP, other activities such as instant messaging and video streaming use port 80 as their back up because firewalls and security software often do not filter port 80 traffic. Port 80 can also be used to stream audio, play games, and download updates and patches. Users can use port 22 to open an SSH session to access a server remotely. At the same time, users can use SCP at port 22 to initiate a file transfer over an SSH session. The former is an highly interactive activity with low delay tolerance while the latter is a passive activity with high delay tolerance. Therefore, multiple applications using the same port makes it difficult to differentiate the QoS requirements of each application by port-based classification alone.

Although port-based classification has limitations described above, Maier *et al.* demonstrate that the port-based technique works quite well in home networks [73]. As modern wireless access points allow users to input specific ports for prioritization, it is possible

to configure access points manually to improve quality of applications. However, as new applications emerge, users need to keep track of ports assigned to those applications. One of CHAP's strengths is the removal of such need for manual configuration and the ability to adapt to emerging applications as long as the relationship depicted in Figure 2.1 holds.

3.3.2 Packet Payload-based Classification

To overcome the limitations of port-based techniques, alternative techniques to inspect the content of packet payloads has been proposed. These techniques rely on the assumption that packets in a given flow contain unique patterns or signatures to identify an application unambiguously. The payload-based approaches are similar to those widely used in security software that use virus signatures to detect the presence of malicious code in files.

Moore *et al.* propose a classification technique that inspects the payload of each packet and approaches 100% accuracy [118]. Their approach involves a nine-step process (Table 3.4) to gain sufficient confidence that the flow belongs to a specific application. Each flow is tested against the nine classification procedures sequentially. Once a procedure returns a positive output for a flow, the flow is verified by a *verification* module. If the *verification* module passes, the flow identification is complete. If not, a manual inspection is necessary to complete the flow identification.

Table 3.4: Procedures of Payload-based Classification

	Identification Method	Example
1	Port-Based classification (only)	-
2	Packet Header (including 1)	*simplex* flows
3	Single Packet Signature	Many Virus/Worm
4	Single Packet Protocol	IDENT
5	Signature on the first KBytes	P2P
6	First KByte Protocol	SMTP
7	Selected flow(s) Protocol	FTP
8	(All) Flow Protocol	VNC, CVS
9	Host History	Port-scanning

Although this approach can identify flows with near 100% accuracy, it is a computation-

intensive process. In the worst case, when the automatic classification fails, it involves nine sequential procedures, one verification process, and one manual inspection process. Due to its computational intensity, payload-based classification is difficult to perform in real-time and is often done offline. Moreover, payload approaches require prior knowledge of unique patterns or signatures of all the applications to identify flows correctly. Finding unique patterns and signatures of applications is a difficult task because of constantly evolving applications and lack of public documentation. Furthermore, encryption offered by applications such as VoIP and P2P makes it harder, if not impossible, to identify flows using payload-based approaches.

Meeting QoS requirements of applications often do not require a 100% accurate identification of each application. As long as applications are grouped into similar categories, treatments by categories are sufficient to provide enough network resources and assistance to improve their quality. For example, one of CHAP's strengths is the automatic placement of each flow in categories following the relationship depicted in Figure 2.1.

3.3.3 Behavior-based Approaches

Karagiannis *et al.* propose a classification technique called *BLINC* to identify applications by analyzing the transport layer behaviors of end hosts [119]. While both port-based and payload-based approaches use transport ports to map flows to applications, BLINC utilizes port numbers only as an index without any application information. The information gathered by packet headers is used to identify host behavior patterns at the transport layer on three levels of increasing detail: Social Level (hosts that it communicates with), Functional Level (server vs. clients or peer nodes) and Application Level (transport layer interactions between particular hosts on specific ports). BLINC analyzes the three-level information using a set of heuristic rules empirically derived by inspecting interactions present in various applications. These rules refine BLINC's classification results and are controlled by a set of operator-defined thresholds to balance between aggressive and conservative classification. The authors demonstrate BLINC's ability to classify 80-90% of real traffic samples with

more than 95% accuracy by comparing BLINC's classification results and payload-based classification results. Although BLINC can identify types of applications and has potential to identify unknown applications such as new P2P applications and malicious flows based on expected behavior, BLINC cannot handle encrypted traffic and short-lived flows.

3.3.4 Statistical Approaches

Because of the shortcomings of previously discussed approaches, researchers recently have focused their attention on machine learning and statistical approaches to differentiate applications. Claffy's investigations on the joint distribution of flow duration and number of packets demonstrated the differences between the distributions of some application protocols [120]. Following Claffy's research, other researchers have characterized and modeled particular applications [121, 122, 123]. Statistical approaches focus on application (payload) independent features such as packet length or inter-arrival times to classify flows [124]. A statistical classifier is trained on a representative set of flow instances where the network applications are known, allowing the classifier to determine the types of unknown flows. Thus, statistical approaches consist of selection of features and machine learning algorithms.

Features can be selected in five levels: Single Packet, Flow, Connection, Intra-flow, and Multi-flow levels. Single Packet level captures features such as mean packet length and various moments such as variance. These can be calculated directly from packet header information and are simple to compute. Flow level provides flow duration, mean data volume per flow, mean number of packets per flow, and variance of these metrics. Connection level offers behaviors associated with transport level connections, and information such as advertised window size and throughput distribution. Intra-flow provides statistics about the packets within a flow, such as inter-arrival times between packets and loss ratios. Multi-flow level can capture statistics across multiple flows or connections, providing valuable information on applications that use multiple flows for their communication.

A minimum set of features can be selected for an application to reduce learning and

classification times and to increase classification accuracy. Then machine learning algorithms can be used to compare network traffic to models created by the known sets of features for various applications. Statistical approaches offer higher accuracy than port-based approaches while providing much less computational complexity than payload-based approaches. However, measured features can be inconsistent in wireless networks. For example, the presence of management and control frames and retransmissions can alter packet inter-arrival information. To improve quality of applications, it is more important how to treat different network traffic rather than identifying specific applications.

3.4 Credit-based Active Queue Management

CHAP is not the first AQM to use credits for transmission scheduling. Credit-Based Fair Queueing (CBFQ) [125] and Wireless Credit-based Fair Queueing (WCFQ) [126] are queueing disciplines designed to ensure bandwidth fairness of flows using credits. Although there are many AQMs that seek bandwidth fairness for flows such as FRED [112], SCFQ [127], CSFQ [128], and DRR [129], CBFQ and WCFQ claim to ensure bandwidth fairness with delay bounds and the most simplicity. Although CBFQ and WCFQ are credit-based, their goal is to ensure bandwidth fairness of flows and not QoS. Moreover, the lack of details makes it difficult to implement and reproduce the results presented in the papers [125, 126]. Therefore, CBFQ and WCFQ are not used in a performance comparison with CHAP.

Figure 3.2 lists steps in the CBFQ algorithm in pseudo code where S_i is the bandwidth share of flow i, K_i is the credit of flow i and J is the total number of flows. Initially, every flow has no credits and gains credits when it is backlogged at the queue and does not get a turn to send its packets. CBFQ picks a flow from the backlogged flows by inspecting the credits. More specifically, it computes $\frac{L_i - K_i}{S_i}$ for all backlogged flows and sorts them in increasing order. In other words, CBFQ calculates the difference between the HOL packet size and the credit and scales them by the flow's bandwidth share. Therefore, the value of this computation increases with bigger HOL packet, lower credit, and/or lower bandwidth share. Since the priority is given to a flow with the lowest value, a flow with a small Head-

1 Initialization phase:

2 $\forall l, l \in 1, ..., J, SetK_l \leftarrow 0$;

3 Operation:

4 Let $F = j_1, ..., j_k$ such that queues $j_1, ..., j_k$ are currently backlogged;

5 Let L_{j_n} be the size of the HOL packet of queue $j_n \in F$;

6 sort the backlogged queues according to
$\frac{L_{j_1}-K_{j_1}}{S_{j_1}} \leq \frac{L_{j_2}-K_{j_2}}{S_{j_2}} \leq ... \leq \frac{L_{j_k}-K_{j_k}}{S_{j_k}}$

7 Transmit the HOL packet of queue j_1, and update the counters as follows:
$K_{j_n} \leftarrow K_{j_n} + \frac{L_{j_1}-K_{j_1}}{S_{j_1}} S_{j_n}, \ \forall j_n \in F \setminus j_1$
$K_{j_1} \leftarrow 0$

8 Goto 4:

Figure 3.2: CBFQ Algorithm

of-Line (HOL) packet, high credit and/or higher bandwidth share gets priority. Once the flow is selected, the HOL packet from that flow is served and the credits of all backlogged flows are updated following the formulas shown in Step 7. The flow CBFQ picks loses all its credit (reset to 0) while all the other backlogged flows are increased by $\frac{L_{j_1}-K_{j_1}}{S_{j_1}} S_{j_n}$. In other words, all the other flows that waited gain what the served flow lost, scaled by their own share of bandwidth.

Wireless Credit-based Fair Queueing (WCFQ) [126] is an extension of CBFQ to ensure bandwidth fairness in wireless environments. The WCFQ algorithm is similar to CBFQ but it includes the "cost" of a packet in the wireless network. The "cost" of a packet is estimated and is reflected in a packet selection process. Table 3.5 shows the difference in the flow selection.

Table 3.5: Flow selection criteria of CBFQ and WCFQ

CBFQ	$min_{i \in F} \frac{L_i - K_i}{s_i}$
WCFQ	$min_{i \in B(p)} \frac{L_i(p) - K_i(p) + U_i(p)}{\phi_i}$

p is the index of the packet in service, $B(p)$ is the set of backlogged flows, $U_i(p)$ is the

transmission cost of the packet and ϕ_i is the weight of flow i. As seen in Table 3.5, the only difference is the inclusion of $U_i(p)$. The WCFQ credit update algorithm is slightly different from CBFQ and is depicted in Figure 3.3.

1 for $(i = 1;\ i \leq N;\ i++)$

2 if $(i \in B(p+1)$ && $i \neq f_{p+1})$

3 $K_i(p+1) = K_i(p) + \max(\frac{L_{f_{p+1}}(p) - K_{f_{p+1}}(p)}{\phi_{p+1}}, 0)\phi_i$

4 elseif (not $(i \in B(p+1)))$

5 $K_i(p+1) = 0$

6 end for

7 if $(f_{p+1} \in B(p+1))$

8 $K_{f_{p+1}}(p+1) = \max(0, K_{f_{p+1}}(p) - L_{f_{p+1}}(p))$

9 else

10 $K_{f_{p+1}}(p+1) = 0$

Figure 3.3: WCFQ Algorithm

Steps 1 through 6 iterate through all active flows. Step 2 checks if flow i is backlogged after transmission of packet p and that flow i is not the flow from which packet $p+1$ is picked. With these conditions satisfied, the credit of flow i is increased either by 0 or by the amount similar to that in CBFQ scaled by its own weight in Step 3. If not, Step 4 checks if flow i is not backlogged after transmission of packet p. If it is not backlogged, it resets the credit of flow i to 0. Step 7 checks if the flow f_{p+1}, the flow from which a packet is served after packet p, is backlogged. If it is backlogged, it updates the credit to the maximum of 0 and the difference between the current credit and the size of HOL packet of flow f_{p+1}. If not, the credit of flow f_{p+1} is reset to 0. The "cost" function is used in the selection of the next flow but not in the calculation of the credits.

The "cost" function of flow i is expressed in the following equation:

$$U_i = -\beta log(1 - E_i) \tag{3.1}$$

where E_i is a uniform distribution in $[0, 1]$ and β is a configurable parameter. β ranges from 0 to ∞, where 0 represents perfect CBFQ fairness where wireless channel condition has no impact and ∞ represents best channel-condition scheduling. For simulations, E_i is expressed as a function of time t with the following equation:

$$E_i(t) = 0.5 + dcos(2\pi f_{i,t}(t) + \theta_i) + z_i(t) \tag{3.2}$$

where θ_i are independent and uniformly distributed in $[0, 2\pi]$ giving the channel conditions statistically independent phases, $f_{i,t}$ is a random process following a Gaussian moving average process, and $z_i(t)$ is the additive noise to model the effects of Rayleigh and Shadow fading with the conservative assumption of additive noise in the range of $[-w, w]$.

3.5 Summary

This chapter reviews research work closely related to the dissertation. Section 3.1 reviews techniques such as IEEE 802.11e to improve Quality of Service (QoS) in home networks. All the techniques discussed in the section require explicit classification of traffic and pre-determined scheduling of classes. Section 3.2 reviews Active Queue Management techniques used to improve QoS. All the approaches discussed in the section need packets to be tagged or marked to specify their quality requirements and also a pre-determined scheduling based on these tags or marks. Section 3.3 reviews traffic classification techniques to differentiate individual flows. Classification of individual flows is required to schedule packets according to their quality requirements. Port-based classification provides a simple method to identify flows based on transport ports. Packet payload-based classification offers a more accurate classification at the cost of complex computation. Behavior-based classification classifies flows based on the social, functional and application level based on a set of heuristic rules. Statistical approaches identify flows based on distributions of characteristics such as mean packet length, flow duration, and mean data volume. All these approaches require a manually configured scheduling policy to accommodate different quality requirements of

classified flows. Section 3.4 reviews credit-based AQM techniques such as Credit-Based Fair Queueing (CBFQ) and Wireless Credit-based Fair Queueing (WCFQ). CBFQ and WCFQ use credits to provide throughput fairness among flows. Their algorithms become the base for Credit-based Home Access Point (CHAP) explained in Chapter 4.

Chapter 4

Approach

4.1 Overview

This chapter proposes CHAP, an approach for improving overall quality of service for typical applications on a wireless network at home using a credit-based queue management system at the wireless access point. The approach focuses on using credits to classify flows implicitly, taking wireless condition of flows into account, to lower latency for delay-sensitive flows while maintaining reasonable throughput for delay-insensitive flows. The assumed conditions suitable for the proposed approach include:

- A focus on IEEE 802.11 networks. While 802.11 is the focus, CHAP can be applied to other types of wireless networks that may be deployed at home in the future, so the related issues are discussed in this dissertation.

- A focus on actively calculating credit for all flows that go through the wireless APs in a home wireless infrastructure network. Therefore, ad-hoc wireless networks are not studied in this research. This chapter discusses credit calculation for downstream traffic only.

- An assumption that the increase in latency for delay-sensitive flows results from packet queue build-up at the wireless AP. With home broadband access getting higher

bandwidths and wireless capacity lower than wired capacity historically, the queue is most likely to exist at the wireless AP.

- A cross-layer approach where the IP layer traffic queue requires information from the data link layer. The credit is calculated using the transmission time information relayed from the data link layer.

- Delay sensitivity of flows based on flow bandwidth usage. In other words, CHAP does implicit classification based on flow rates but also takes into account their wireless condition to dynamically adjust priorities.

The following methodologies are applied to investigate the proposed approach:

- Analytical Models. Mathematical modeling is used to study the effect of credit-based queue management on latency and determine guidelines for some CHAP parameters. (Chapter 5)

- Simulations. NS2 simulations are used to evaluate CHAP in home wireless networks and to validate the analytical model. (Chapter 6)

- Implementation. Real hardware implementation and measurements are used as proof of concept and to validate the simulations. (Chapter 7)

Analytical modeling can provide closed-form solutions that are easy to evaluate, but real systems usually have additional complexity and thus are hard to model precisely. Simulations can provide evaluations for the models and circumstances close to real systems with good repeatability and scalability. However, they may still not fully represent the complex network systems. Therefore, implementation and measurements are used to reinforce and evaluate the proposed approach in real home networks. Section 4.2 describes the approach for improving QoS in wireless home networks. Chapter 5 describes the mathematical model to analyze the latency characteristics using the credit-based queue management. The simulation and implementation methodologies are discussed in Chapters 6 and 7, respectively.

4.2 Algorithm

A flow is defined by a five-tuple: source address, source port, destination address, destination port, and protocol. Residential wireless access points keep track of all the flows that they serve because they need to know the network address translation (NAT) for proper routing of packets belonging to these flows. We propose that wireless access points track one additional piece of information - flow credits. The credit of a flow is an indication of the volume of information exchanged and also how costly it is to serve packets from the flow. In other words, high credits suggest that the flow is sending and/or receiving little information or it does not take much time to serve packets from this flow. Low credits suggest that the flow is sending and/or receiving a large amount of information or it takes a relatively long time to serve packets from this flow. We propose to give priority to packets with the highest credits at a given time.

A static credit system is equivalent to a system with explicit classification with priority. We make the credits dynamic by subtracting the cost of a packet upon dequeueing. The cost of a packet is measured by the amount of time it takes to transmit at the data link layer. In the case of a wireless data link layer, this includes the preamble, DATA/ACK frame, retransmissions, and timeouts. Even in the case of a lost frame, the cost includes all the attempts to transmit the frame. The longer it takes at the data link layer, the more credits a flow spends for having its packet transmitted. In addition, it is necessary to boost the credits to prevent a system to prevent all the credits from converging to negative infinity. The credit boost occurs when all the flows with backlogged packets have 0 or fewer credits. The system uses Equation 4.1 to boost credits for not only those flows with backlogged packets but also all active flows at that time:

$$\alpha_i' = \frac{\alpha_i}{2} + I \tag{4.1}$$

where α_i is the credit of flow i before the boost while α_i' is the credit of flow i after the boost. The parameter I is in units of time, consistent with the unit of credits (the value of

which is determined in Section 5.4). Due to the memory limitations of wireless APs, it is impossible to keep track of all flows forever. Therefore, the last time a packet was served from a particular flow and a timeout value are required to get rid of old flows that have not transmitted in a long time. The timeout value follows some default values such as 120 and 3600 seconds, used in current access point implementations.

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 return p
7 ... $\alpha_k = \alpha_k - cost$

enqueue(p)

1 $j =$ flow ID of p
2 add p to the end of the queue
3 if *the queue is full* then
4 find k such that $\alpha_k = min[\alpha_1, ..., \alpha_n]$ and $k \in$ set of backlogged flows
5 $p_{tmp} =$ the first packet from flow k in the queue
6 drop(p_{tmp})
7 end if

Figure 4.1: CHAP Algorithm for Downstream Traffic

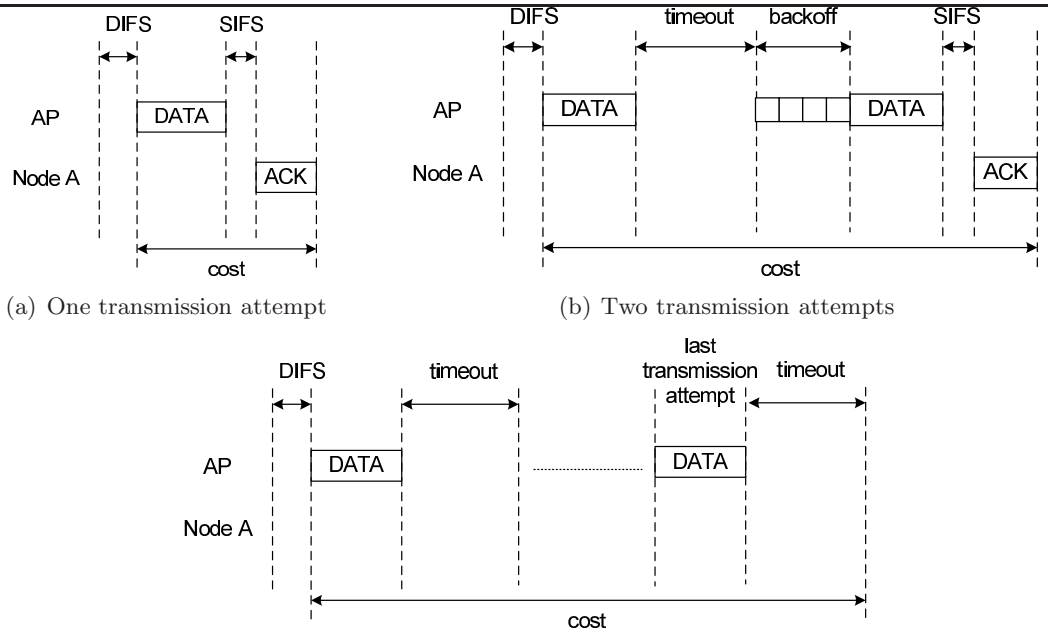
Figure 4.1 summarizes the dequeue and enqueue functions for downstream traffic in a wordy, pseudo code format. dequeue() returns the first packet that belongs to the flow with the most credits in the queue. Step 1 finds the flow with the most credits. If a credit boost is necessary in steps 2-4, CHAP boosts credits for all active flows. Step 5-6 dequeue the packet from the flow picked in Step 1. After the packet is transmitted, the wireless

data link layer decreases the credits by the corresponding channel usage time (Step 7).

enqueue(p) adds the incoming packet p to the queue. If the queue is full with the incoming packet, CHAP drops the first packet that belongs to the flow with the least credits in the queue. This mechanism ensures that CHAP can protect incoming packets from flows with more credits. Step 1 retrieves the flow ID of the incoming packet p. Step 2 adds the incoming packet to the queue. Step 3 checks if the queue is full. If the queue is full, CHAP proceeds to find a packet to drop from the backlogged flow with the fewest credits following Steps 4-6. Steps 3-7 ensure that there is always room for an incoming packet by limiting the maximum number of packets in the queue to $q_{size} - 1$.

Similar algorithms have been used in CBFQ [125] and WCFQ [126]. However, these algorithms involve more complex credit calculation and update methods. CBFQ, as detailed in Figure 3.2, keeps multiple queues of packets and involves sorting of the queues. WCFQ uses a slightly more complicated but similar algorithm and employs the credit update algorithm as detailed in Figure 3.3. Updating credits in WCFQ requires the calculation of the cost based on wireless channel scheduling. The CHAP algorithm explained above has been chosen for simplicity. CHAP takes advantage of the Network Address Translation (NAT) module to store per-flow information and augments it to store credits. It adopts the Linux credit update and boost methods and uses the channel usage time as cost for simpler computation complexity. While CBFQ and WCFQ algorithms allow only one packet per flow to be transmitted for more complete fairness among flows, CHAP allows a flow with the most credits to burst as long as it has the most credits compared to any other flows with backlogged packets. CHAP also provides an additional protection for flows with many credits by dropping packets from flows with few credits through the enqueue(p) function.

In 802.11, downstream and upstream traffic share the same medium. However, downstream traffic typically dominates in home networks [73]. Therefore, the CHAP algorithm focuses on enqueueing and dequeueing of packets of downstream traffic.



(a) One transmission attempt

(b) Two transmission attempts

(c) Timeout

Figure 4.2: Cost of a packet

4.3 Cost to Transmit a Packet

Figure 4.1 shows that a flow loses credits upon sending a packet. The cost of a transmitted packet in Step 7 of dequeue() is the corresponding channel usage time in the wireless medium. Because the AP maintains the credits of all flows, the AP updates the credits only upon dequeueing packets it transmits. Packets received at the AP from wireless stations do not affect the credits. The cost of a packet is calculated from the time the AP starts its first transmission attempt. Figure 4.2 depicts how the cost of a packet is calculated in three situations. Figure 4.2(a) shows the cost of a packet when the AP transmits the packet successfully in one transmission attempt. The cost of the packet is the sum of the transmission time of the DATA frame, a SIFS and the transmission time of the ACK frame. Figure 4.2(b) shows the cost of a packet when the AP transmits the packet successfully in two transmission attempts. In this case, the cost of the packet includes one additional transmission of the DATA frame, a timeout and a backoff period on top of the cost of a packet successfully transmitted in one attempt. Figure 4.2(c) shows the cost of

a packet when the AP fails to transmit the packet. Although the packet never reaches its destination, CHAP charges the time spent attempting to transmit the packet because the AP is unable to transmit any other packets while attempting to transmit this packet. The cost of a lost packet is the difference between the time of the first transmission attempt and the expiration time of the timeout of the last transmission attempt.

When CHAP selects a packet to dequeue, the cost of the packet is unknown because it is computed only after the wireless data link layer attempts its transmission. Therefore, it is possible for CHAP to select a packet from a flow with credits less than the cost. The boost mechanism in the algorithm ensures that the credits of active flows are positive once CHAP detects that it selects a packet from a flow with negative credits.

4.4 Example

To illustrate the mechanism further, consider an example of 3 flows. Let us assume that the ratio of bandwidth use for flows 1 through 3 is 2:3:5 and the cost of each frame is 5 for all flows. In addition, the increment value of I is 10 and every flow starts with 10 credits. If multiple backlogged flows have the same number of credits, the flow with lower index gets to send its packet. Also assume that the packets come as a burst of 10 packets total and the next batch of 10 packets come when the queue is empty. Table 4.1 shows how credits are decremented and incremented over time and which flow is chosen to send.

The first row shows every flow starting with 10 credits according to the assumption. The scheduler always picks the flow with the highest credits to send a packet. If we extend this over a longer period of time, the credits over time are shown in Figure 4.3 with a bar graph. Flow 3 reaches 0 the most, triggering the credit boost.

Table 4.2 shows queueing delays of each packet that entered the queue. With 10 packets entering every 10 time slots, FIFO provides 4.5 time slots of queueing delay on average for all packets. Packets from flow 1 experience queueing delays of 0 and 3 time slots, packets from flow 2 experience queueing delays of 1, 4, and 6 time slots and packets from flow 3 experience queueing delays of 2, 5, 7, 8, and 9 time slots. In other words, the average

Table 4.1: Example: Table of credits vs. time for flows 1, 2, and 3

Time Index	Flow 1	Flow 2	Flow 3	Packet from Flow	Note
0	10	10	10	-	Start
1	5	10	10	1	
2	5	5	10	2	
3	5	5	5	3	
4	0	5	5	1	
5	0	0	5	2	
6	0	0	0	3	
7	10	10	10	-	Boost
	10	5	10	2	
8	10	5	5	3	
9	10	5	0	3	
10	15	12.5	10	-	Boost
	15	12.5	5	3	

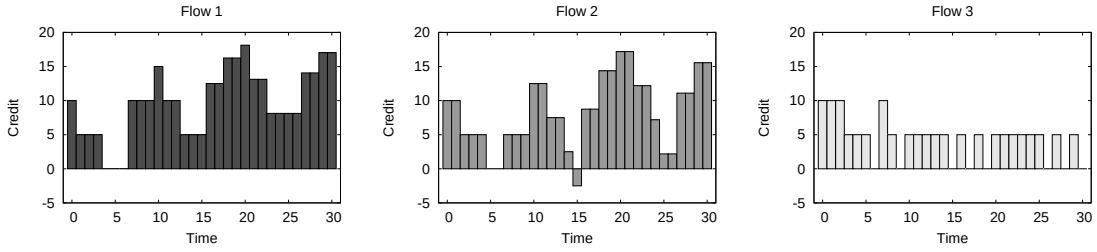


Figure 4.3: Example: Graph of credits vs. time of flows 1, 2 and 3

queueing delay for flow 1, 2, and 3 are 1.5, $\frac{11}{3}$, and $\frac{31}{5}$ time slots, respectively. The average queueing delays for flows 1 and 2 are lower than the average queueing delay of 4.5 time slots experienced using the FIFO discipline.

4.5 Summary

This chapter provides an overview of CHAP, an approach for improving overall quality of service for typical applications on a wireless network at home using a credit-based queue management system at the wireless access point. Section 4.1 provides the assumptions on which the CHAP algorithm is based and introduces the methodologies to investigate

Table 4.2: Example: Table of queueing delays of each packet

Flow ID	Packet ID	Queueing Delay
1	1	0
	2	3
2	3	1
	4	4
	5	6
3	6	2
	7	5
	8	7
	9	8
	10	9

CHAP. The CHAP algorithm, described in Section 4.2, details how CHAP classifies flows implicitly using credits, lowers latency for delay-sensitive flows, and maintains reasonable throughput for delay-insensitive flows while taking wireless conditions into account. Section 4.3 describes how CHAP measures the cost of a packet. The cost of a packet is the time difference between the first transmission attempt of the packet and the reception of the corresponding ACK frame for a successful transmission. In case of an unsuccessful transmission, the cost of a packet is the time difference between the first transmission attempt of the packet and the timeout of the last transmission attempt. The example provided in Section 4.4 demonstrates that CHAP provides shorter queueing delays for low-bandwidth flows than FIFO.

Chapter 5

Analytical Model

Chapter 5 explains the analytical model behind the credit-based queue management. It is important to analyze how rates of different flows drive their own credits and give each other dynamic priority. The flexibility of allowing limited burstiness and queueing delay from the credit-based algorithm are also important.

5.1 Credit Analysis

Assume the AP queueing system has reached steady state, meaning that all flows have reached their steady rates. Table 5.1 lists all the variables used in the analysis.

Table 5.1: Summary of all the variables

Variable	Description
C	Total outgoing rate (pkt/s)
i	Flow index
r_i	Outgoing rate of ith flow (pkt/s)
α_i	Credit of ith flow (s)
α_i^h	Upper bound of α_i (s)
α_i^l	Lower bound of α_i (s)
θ_i	Change in α_i over one second
I	Increment (s)

For the simplicity of analysis, we also assume that the cost to transmit all frames at the

data link layer is constant and is equal to $1/C$.

Equation 5.1 demonstrates that the sum of rates of all the flows equals the total outgoing rate C.

$$\sum_{i=1}^{n} r_i = C \tag{5.1}$$

Equation 4.1 shows the basic formula for boosting the credit of all flows when all the flows with backlogged packets have 0 or fewer credits. Since all n flows are active and backlogged, the credits of all flows get boosted with the equation once the flow with the highest credit draining rate reaches 0 or lower.

Since the cost in credits is the transmission time of each packet at the data link layer, the sum of changes in credits of all flows within a 1 second interval must equal 1 as shown in Equation 5.2. In other words, since each packet costs $1/C$ and there are C packets outgoing every second, the total change in credits from all flows must equal 1.

$$\sum_{i=1}^{n} \theta_i = 1 \tag{5.2}$$

Equation 5.3 shows the change in credits of the ith flow. The cost of transmission time of each packet is $1/C$ and the ith flow sends r_i packets per second.

$$\theta_i = r_i \frac{1}{C} \tag{5.3}$$

It is necessary to find out which flow will trigger the credit boost for all the flows. Let k be the index of a flow that uses the most credits within a one second interval:

$\theta_k = max[\theta_1, ..., \theta_n]$

This means that the kth flow will drain its credits to 0 before any other flows. Then, let t be the time it takes for the kth flow to drain its credits.

$$t = \frac{I}{\theta_k} = \frac{CI}{r_k} \tag{5.4}$$

After t seconds, all the flows will have drained their credits by ΔB_i, which is proportional to their change in credits, θ_i.

$$\Delta B_i = \theta_i t = \frac{r_i}{C}\frac{CI}{r_k} = \frac{r_i}{r_k} I \tag{5.5}$$

So far, we have ΔB_i for every flow after t seconds. The credit boost happens every t seconds when flow k drains its credits from I to 0. Therefore, every t seconds, every flow's credits are boosted using Equation 4.1. If we use every boost as an index, we can generalize the upper limit of the credit of a flow in steady state. Let us assume that $\alpha_i^h(y)$ is the upper limit of credit of the ith flow and it started with some arbitrary value x after the yth boost. In other words, $\alpha_i^h(0) = x$. After one t second interval, the credits of the ith flow are decremented by ΔB_i and the boost mechanism is triggered by the kth flow. Equation 5.6 shows the new value of the credit after the first boost. Equation 5.7 shows the credit after the second boost while Equation 5.8 shows the value of credit after the third boost.

$$\alpha_i^h(1) = \frac{x - \Delta B_i}{2} + I = \frac{x - \Delta B_i + 2I}{2} \tag{5.6}$$

$$\alpha_i^h(2) = \frac{\frac{x-\Delta B_i+2I}{2} - \Delta B_i}{2} + I = \frac{x - 3\Delta B_i + 6I}{4} \tag{5.7}$$

$$\alpha_i^h(3) = \frac{\frac{x-3\Delta B_i+6I}{4} - \Delta B_i}{2} + I = \frac{x - 7\Delta B_i + 14I}{8} \tag{5.8}$$

By generalizing the pattern in Equations 5.6 to 5.8 to an equation with an arbitrary number of boosts y, we get Equation 5.9:

$$\alpha_i^h(y) = \frac{x - (2^{y-1} - 1)\Delta B_i + (2^y - 2)I}{2^{y-1}} \tag{5.9}$$

Equation 5.10 is the result of taking y to infinity from Equation 5.9.

$$\alpha_i^h = 2I - \Delta B_i = 2I - \frac{r_i}{r_k} I \tag{5.10}$$

The lower limit is obtained by subtracting ΔB_i from the upper limit α_i^h.

$$\alpha_i^l = \alpha_i^h - \Delta B_i = 2I - 2\frac{r_i}{r_k} I \tag{5.11}$$

Now, we have the range of the credits for any given flow in steady state. Equation 5.12 combines Equation 5.10 and 5.11 to demonstrate the range using inequality:

$$2I - 2\frac{r_i}{r_k} I \leq \alpha_i \leq 2I - \frac{r_i}{r_k} I \tag{5.12}$$

Therefore, the average credit of the ith flow is the average value which is shown in Equation 5.13.

$$E(\alpha_i) = 2I - \frac{3}{2}\frac{r_i}{r_k} I \tag{5.13}$$

Assuming that the flows are sorted by their rates in increasing order, the average burst length allowed in terms of time is calculated by Equation 5.14. It is basically the difference of the average credits of flow i and flow $i+1$.

$$E(\alpha_i - \alpha_{i+1}) = \frac{3}{2}\frac{(r_{i+1} - r_i)}{r_k} I \tag{5.14}$$

5.2 Delay Analysis

Delay is one of the most important performance metrics relevant to QoS. One of our objectives is to reduce latency for delay-sensitive flows. In order to analyze the delay for all flows, we apply the delay analysis from the Probabilistic Priority Discipline Queue with the probabilities calculated from the credit-based queue management.

5.2.1 Probabilistic Priority Discipline

Strict priority discipline queues work on a simple principle that packets from a higher priority queue get serviced before packets from lower priority queues. It provides the proper quality of service as long as the high priority queues are not backlogged at all times. In such a case, the packets from lower priority queues will starve and never get service.

In order to mitigate this side effect of starving low priority queues, a probabilistic priority discipline is introduced [130]. Just like a strict priority discipline queue, probabilistic priority discipline queues have a certain number of discrete priority queues. The core difference is that a probabilistic priority discipline queue starts from the highest priority queue but does not always dequeue packets from that queue, instead having probabilities associated with each priority queue. On dequeue(), it starts with the highest priority queue that is not empty and serves a packet from that queue based on the probability associated with that queue.

Jiang *et al.* [130] analyze the delay from the probabilistic priority discipline queue in [131]. Figure 5.1 depicts the queueing diagram of a probabilistic priority discipline queue. The authors introduce two different approaches for estimating delay and we will introduce one approach in this section. For the simplicity of analysis, Jiang *et al.* choose two classes of traffic that are independent Poisson processes with rates λ_1 and λ_2. The total arrival rates of two classes are the sum of each class's arrival rate: $\lambda = \lambda_1 + \lambda_2$. Further, Jiang *et al.* assume that the service times of class i packets are independent, identically distributed,

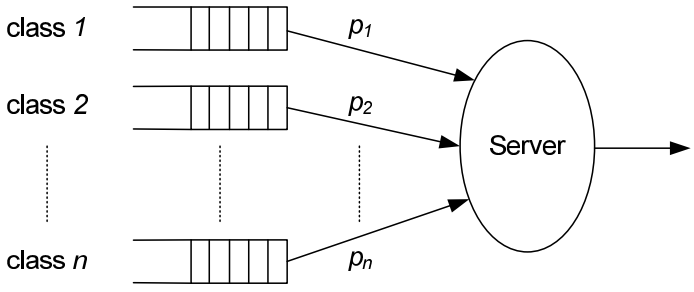


Figure 5.1: Probabilistic Priority Discipline Queue

stochastic variables that follow a general distribution with finite first and second moments s_i and $s_i^{(2)}$. Therefore, the queue for each class is essentially an M/G/1 queue. Table 5.2 lists other notations used in the analysis.

Table 5.2: Variables used in delay analysis

Variable	Description
$\overline{T}_i$	E[sojourn time for class i packets]
$\overline{W}_i$	E[waiting time in queue for class i packets]
$\overline{N}_i$	E[number of packets waiting in queue i]
$\overline{W}_0$	E[residual service time]
p_i	probability of service assigned to class i
ρ_i	$\lambda_i s_i$
ρ	$\sum_{i=1}^{n} \rho_i$
$\overline{i}$	class other than class i
q_i	probability that class i queue is nonempty
ω_i	probability that the head packet from class i is served when class $\overline{i}$ is nonempty

The sojourn time of a packet is its waiting time plus its service time, and the traffic intensity is less than unity to ensure stability. From this definition, we have:

$$\overline{T}_i = \overline{W}_i + s_i \tag{5.15}$$

From little's theorem, we have:

$$\overline{N}_i = \lambda_i \overline{W}_i \tag{5.16}$$

The work conservation law [132] states that if the scheduler is work-conserving, the sum of the average waiting times, weighted by their share of the network utilization, is independent of the service discipline. Since we are using M/G/1 queues here:

$$\sum_{i=1}^{2} \rho_i \overline{W}_i = \frac{\rho \overline{W}_0}{1-\rho}, \qquad \text{for} \rho < 1 \tag{5.17}$$

where

$\overline{W}_0 = \sum_{i=1}^{2} \rho_i \frac{s_i^{(2)}}{2s_i}$.

As shown in Table 5.2, $\overline{W}_0$ is the average residual service time which is the average remaining service time for the packet found in service by a newly arriving packet.

Because Jiang *et al.* consider two classes ($i = 1, 2$), $i = 1, 2$, $\overline{i} = 2, 1$ respectively. From the definition of ω_i, we have:

$$\omega_1 = p_1, \quad \omega_2 = 1 - p_1, \quad \text{and} \quad \omega_i + \omega_{\overline{i}} = 1 \tag{5.18}$$

Jiang *et al.* use a decomposition approach to analyze delay. For any newly arriving packet of class i, the mean delay is composed of three components: the average residual service time, the delay in its own queue and the delay waiting for the other queues. Therefore, the decomposition can be expressed as:

$$\overline{W}_i = \overline{W}_0 + s_i \overline{N}_i + s_{\overline{i}} n_{\overline{i}}, \quad \text{for } i = 1, 2 \tag{5.19}$$

where $n_{\overline{i}}$ is the average number of packets from class $\overline{i}$ served before the newly arrived packet.

In order to calculate q_i, the probability that queue i is non-empty, Jiang *et al.* examine an arbitrarily long time interval τ under the assumption that $\rho < 1$ for system stability. The number of newly arrived packets in the time interval for class i is close to $\lambda_i \tau$. In addition, the portion of the time interval that the system is busy serving packets from class i is equal to $q_i[\omega_i q_{\overline{i}} + (1 - q_{\overline{i}})]$. $\omega_i q_{\overline{i}}$ is the probability that class i is picked while queue $\overline{i}$ is nonempty and $(1 - q_{\overline{i}})$ is the probability that queue $\overline{i}$ is empty. The sum of these two terms is the probability that queue i is picked. Multiplying this sum with q_i gives the overall probability that queue i is picked and non-empty. In order to convert this probability to the number of packets, we can multiply by $\frac{\tau}{s_i}$, which is the maximum number of packets from class i that can be served in the time interval:

$$\lambda_i \tau = \frac{\tau}{s_i} q_i [\omega_i q_{\overline{i}} + (1 - q_{\overline{i}})] \tag{5.20}$$

As we take τ to infinity, we get:

$$\rho_i = q_i[\omega_i q_{\overline{i}} + (1 - q_{\overline{i}})] \tag{5.21}$$

Since there are only two flows, Equation 5.21 can be expressed in two equations:

$$\rho_1 = q_1(1 - \omega_2 q_2) \tag{5.22}$$

$$\rho_2 = q_2(1 - \omega_1 q_1) \tag{5.23}$$

Solving Equations 5.22 and 5.23 for q_1 and q_2 results in the following equations.

$$q_1 = \frac{[1 + \omega_1\rho_1 - \omega_2\rho_2] - \sqrt{(1 + \omega_1\rho_1 - \omega_2\rho_2)^2 - 4\omega_1\rho_1}}{2\omega_1} \tag{5.24}$$

$$q_2 = \frac{[1 + \omega_2\rho_2 - \omega_1\rho_1] - \sqrt{(1 + \omega_2\rho_2 - \omega_1\rho_1)^2 - 4\omega_2\rho_2}}{2\omega_2} \tag{5.25}$$

The missing piece from the decomposition approach is $n_{\overline{i}}$, the number of packets of class $\overline{i}$ that are served before the newly arrived packet. First, the newly arrived packet has to wait for $\overline{N}_i$ packets from its own queue i before it reaches the head of the queue. Let β_i be the probability that the packet that has reached the head of the queue and contends with packets from queues $\overline{i}$. The packet from queue i is served with probability 1 if queue $\overline{i}$ is empty and with probability ω_i otherwise. Therefore:

$$\beta_i = (1 - q_{\overline{i}}) + \omega_i q_{\overline{i}} \tag{5.26}$$

Define a random variable X, for the number of class $\overline{i}$ packets served before the packet at the head of queue i is. Then X is distributed with the following probability function:

$$P(X = l, l \geq 0) = (1 - \beta_i)^l \beta_i \tag{5.27}$$

The average number of packets from queue $\overline{i}$ served before the packet at the head of queue i can be estimated by:

$$X \approx \sum_{l=0}^{\infty} lP(X = l) = \frac{1 - \beta_i}{\beta_i} \tag{5.28}$$

Therefore, $n_{\overline{i}}$ is approximately:

$$n_{\overline{i}} = (\overline{N}_i + 1)X \approx (\overline{N}_i + 1)\frac{1 - \beta_i}{\beta_i} \tag{5.29}$$

Substituting all the pieces from the above analysis into Equation 5.19, the delay is as shown in the following equation:

$$\overline{W}_i = \frac{\overline{W}_0 + s_{\overline{i}}\frac{1-\beta_i}{\beta_i}}{1 - \rho_i - \lambda_i s_{\overline{i}}\frac{1-\beta_i}{\beta_i}} \tag{5.30}$$

5.2.2 Credit-based Probability

Section 5.2.1 analyzes the delay on a Probabilistic Priority Discipline Queue. In order to apply this analysis to the delay of credit-based queue management, it is necessary to derive the probabilities of each priority queue. For credit-based queue management, we can assume that each priority class belongs to a flow. The priorities are determined by the steady state credits. As shown in Section 5.1, each flow has its own range of credits in steady state (Equation 5.12). In credit-based queue management, the flow with the highest credit receives the highest priority. In other words, as long as the credits of a flow are higher than any other flows' at a given time, its packet gets served. However, the ranges of credits do overlap and there are times when a flow with higher average credit can have lower credit than a flow with lower average credit. Therefore, calculating the probability that a flow with higher average credit has higher credit than a flow with lower average credit allows us to apply it to the Probabilistic Priority Discipline Queue analysis.

Figure 5.2 shows flows i through $i + 3$ having their credit ranges overlap. We assume that the probability is uniform within the range of credits for all the flows for simplicity

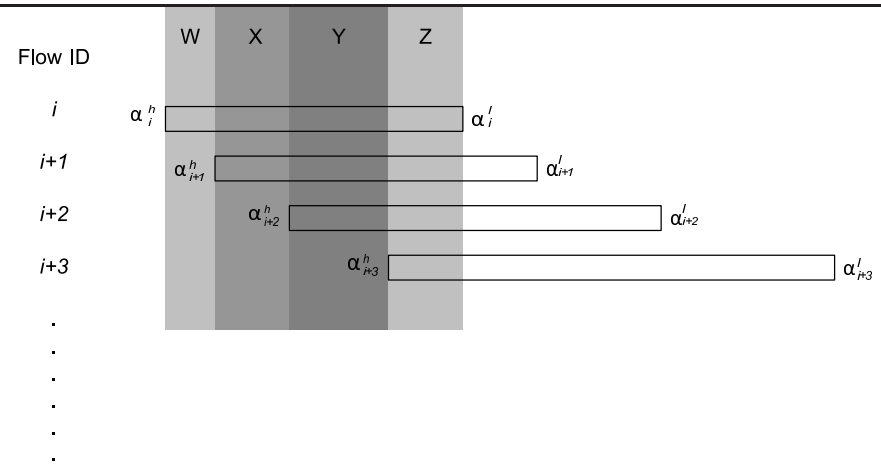


Figure 5.2: Example of overlapping four ranges of credits

of the analysis. Let us use $\Delta\alpha_i = \alpha_i^h - \alpha_i^l$ for simplifying the equations. We also sort the flows by their rates in increasing order. In other words, $r_1 \leq r_2 \leq ... \leq r_n$.

First, let us consider the simplest case where the ith flow's credit does not overlap with any other flows. In that case, $p_i = 1$ because there is zero probability that the ith flow's credit is ever less than that for any flows with lower priority.

Let us assume that we have ith and $i+1$th flow and they do overlap. Then we have to calculate the probability from two regions: W and XYZ. In region W, we only need to consider the uniform probability that the credit is in region W. It is simply the range of region W over the entire range of credit of the ith flow. Let us call that probability p_w:

$$p_w = \frac{\alpha_i^h - \alpha_{i+1}^h}{\Delta\alpha_i} \tag{5.31}$$

For region XYZ, let us look at Figure 5.3 that depicts flows i and $i+1$ from Figure 5.2. We need the conditional probability that the credit of the ith flow is in region XYZ and also greater than the credit of the $i+1$th flow. For any given credit of the ith flow within region XYZ, we can calculate the probability that the credit of the $i+1$th flow is less than that. Since the range is continuous, we construct an integral to calculate the sum of the probabilities. Let us call this probability p_{xyz}:

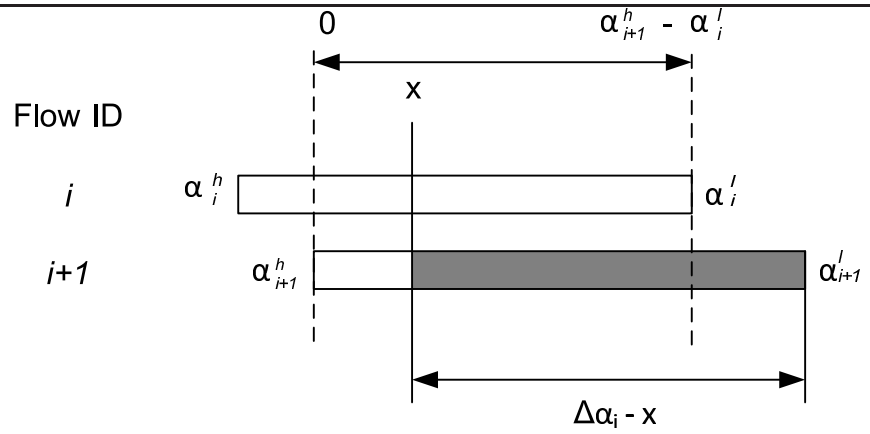


Figure 5.3: Conditional probability for overlapping regions between flows i and $i+1$

$$p_{xyz} = \int_0^{\alpha_{i+1}^h - \alpha_i^l} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x}{\Delta\alpha_{i+1}} dx \tag{5.32}$$

Therefore, p_i in the case there are two flows with overlapping ranges of credits is the sum of p_w and p_{xyz} as shown in Equation 5.33.

$$p_i = \frac{\alpha_i^h - \alpha_{i+1}^h}{\Delta\alpha_i} + \int_0^{\alpha_{i+1}^h - \alpha_i^l} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x}{\Delta\alpha_{i+1}} dx \tag{5.33}$$

Likewise, we can calculate p_i when there are 3 flows with overlapping ranges of credits. With reference to Figure 5.2, it would involve the ith, $i+1$th and $i+2$th flow and 3 distinct regions: W, X and YZ. p_i is shown as below in Equation 5.34.

$$\begin{aligned} p_i = & \frac{\alpha_i^h - \alpha_{i+1}^h}{\Delta\alpha_i} + \int_0^{\alpha_{i+1}^h - \alpha_{i+2}^h} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x}{\Delta\alpha_{i+1}} dx \\ + & \int_0^{\alpha_{i+2}^h - \alpha_i^l} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x - (\alpha_{i+1}^h - \alpha_{i+2}^h)}{\Delta\alpha_{i+1}} \frac{\Delta\alpha_{i+2} - x}{\Delta\alpha_{i+2}} dx \end{aligned} \tag{5.34}$$

Similarly, we can calculate p_i for 4 flows with overlapping ranges of credits.

$$\begin{aligned} p_i = & \frac{\alpha_i^h - \alpha_{i+1}^h}{\Delta\alpha_i} + \int_0^{\alpha_{i+1}^h - \alpha_{i+2}^h} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x}{\Delta\alpha_{i+1}} dx \\ + & \int_0^{\alpha_{i+2}^h - \alpha_{i+3}^h} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x - (\alpha_{i+1}^h - \alpha_{i+2}^h)}{\Delta\alpha_{i+1}} \frac{\Delta\alpha_{i+2} - x}{\Delta\alpha_{i+2}} dx \\ + & \int_0^{\alpha_{i+3}^h - \alpha_i^l} \frac{1}{\Delta\alpha_i} \frac{\Delta\alpha_{i+1} - x - (\alpha_{i+1}^h - \alpha_{i+3}^h)}{\Delta\alpha_{i+1}} \frac{\Delta\alpha_{i+2} - x - (\alpha_{i+2}^h - \alpha_{i+3}^h)}{\Delta\alpha_{i+2}} \frac{\Delta\alpha_{i+3} - x}{\Delta\alpha_{i+3}} dx \end{aligned} \tag{5.35}$$

From Equations 5.33, 5.34, and 5.35, we can see the pattern and generalize it to an arbitrary number of overlapping flows. Let us use m for the number of flows with overlapping ranges of credits. Then p_i for m flows with overlapping ranges of credits is shown in Equation 5.36 when there are m overlapping flows:

$$\begin{aligned} p_i = & \frac{\alpha_i^h - \alpha_{i+1}^h}{\Delta\alpha_i} \\ + & \sum_{w=1}^{m-2} \int_0^{\alpha_{i+w}^h - \alpha_{i+w+1}^h} \frac{\Delta\alpha_{i+w}-x}{\prod_{y=0}^{w}\Delta\alpha_{i+y}} \prod_{z=1}^{w-1}(\Delta\alpha_{i+z} - x - (\alpha_{i+z}^h - \alpha_{i+w}^h))dx \\ + & \int_0^{\alpha_{i+m-1}^h - \alpha_i^l} \frac{\Delta\alpha_{i+m-1}-x}{\prod_{y=0}^{m-1}\Delta\alpha_{i+y}} \prod_{z=1}^{m-2}(\Delta\alpha_{i+z} - x - (\alpha_{i+z}^h - \alpha_{i+m-1}^h))dx \end{aligned} \tag{5.36}$$

5.2.3 Application of Probability Priority Discipline Queue on CHAP

Section 5.2 calculated p_i for any flow in a steady state that has m flows with higher rates whose ranges of credits overlap. Since credit-based queue management introduces dynamic priority based on flows' credits and each flow has its own priority queue, the probability p_i is applied to each flow i.

For simplicity of analysis, a two flow case is considered, where one flow is delay-sensitive with low throughput and the other is delay-insensitive. Let the delay-sensitive flow be flow 1 and the other be flow 2 because lower numbers indicate higher priority. Following the delay analysis in [131], p_1 is necessary to calculate the expected delays for both flows. Since there are only two flows, Equation 5.33 can be used to calculate p_1. The queueing delays are analyzed, varying ρ (0.2, 0.5, and 0.8) and the ratios of rates of the two flows ($\frac{r_1}{r_2} = \kappa$).

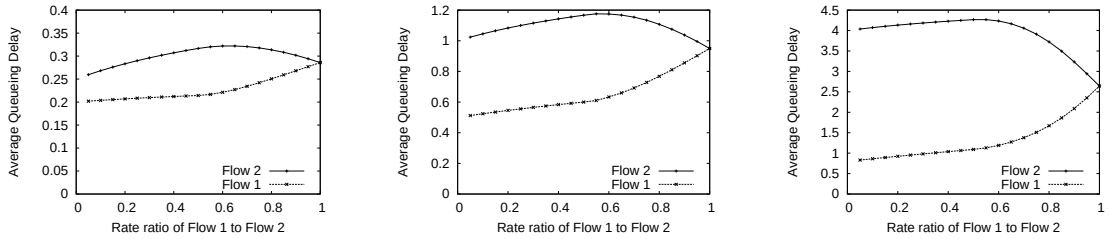
Using Approach 1 explained in Section 5.2.1, we require the following parameters: $\overline{W}_0, s_i, s_i^{(2)}, \beta_i, \rho_i, \lambda_i$. We assume s_i and $s_i^{(2)}$ to be 1 for the simplicity of the analysis. Then, κ relates to the ratio between λ_1 and λ_2. Since we assume $\rho = 0.5$, $0 < \lambda_1 \leq 0.25$ and $0.25 \leq \lambda_2 < 0.5$ for $i = 1, 2$ depending on κ.

The range of credits is necessary to compute p_1. Equation 5.12 shows that Flow 2 ranges from 0 to I while Flow 1 ranges from $2I(1-\kappa)$ to $I(2-\kappa)$. Now, these ranges will only overlap if and only if $2I(1-\kappa) < I$. Therefore, there is an overlap of the range if and only if $\kappa > 0.5$. This means that Flow 2 has to be less than twice as fast as Flow 1 to result

in their ranges to overlap. Otherwise, there is no overlap and this forces the system to act as a strict priority queue. The cases with overlap and no overlap makes us work with two different values of p_1. If there is no overlap, $p_1 = 1$ because it is a strict priority system. If there is an overlap, Equation 5.33 can be used to calculate the appropriate value using κ. Equation 5.37 shows the result of applying κ in Equation 5.33 $\forall \kappa > 0.5$.

$$p_1 = \frac{-4\kappa^2 + 6\kappa - 1}{2\kappa} \tag{5.37}$$

Once p_1 is known, ω_1 and ω_2 can be computed using Equation 5.18. However, ω_2 needs to be non-zero to compute q_i. Therefore, p_1 is capped at 0.9999 for the rest of the analysis. Values for q_i are obtained using Equations 5.24 and 5.25. Once ω_i, q_i, and β_i based on p_1 are computed, the queueing delays can be estimated based on Equation 5.30.



(a) $\rho = 0.2$ (b) $\rho = 0.5$ (c) $\rho = 0.8$

Figure 5.4: Change in delay relative to κ

Figure 5.4 depicts the delay estimates for both flows with a variation of κ and ρ. The horizontal axis shows the rate ratio between Flow 1 and Flow 2. It ranges between 0 and 1 where Flow 1 and Flow 2 have the same rate at 1. The vertical axis shows the average queueing delay of packets. All three values of ρ demonstrates the same trend of the average queueing delay experienced by Flow 1 and Flow 2. When $\kappa < 0.5$, credit-based queue management behaves like a strict priority queue. Delays from both flows are increasing because as κ increases, there are more packets from Flow 1 to be served before switching to packets from Flow 2. As κ increases beyond 0.5, there is a possibility that Flow 1's credit is lower than Flow 2's. It can be observed that this probability is insignificant up to $\kappa = 0.55$. Then, the delay for Flow 2 starts decreasing because credit-based queue

management starts serving only some of the packets from Flow 1 instead of all of them before getting to packets from Flow 2. As κ increases, both curves eventually converge to the same point at $\kappa = 1$ because they have the same rates. At this point, credit-based queue management is serving packets from both flows following a round-robin mechanism. Figure 5.4 demonstrates that credit-based queue management does provide lower delay to delay-sensitive flows without any prior configuration.

5.3 Wireless Considerations

All the analysis done up to this point has been based on the fact that it costs a constant amount of time to transmit a packet at the data link layer, whether it is wired or wireless. In a wired environment, this assumption is closer to being true. However, in a wireless environment, we need to relax this assumption due to the fact that there are many factors affecting the transmission of a data frame over IEEE 802.11.

Relaxation of this assumption changes the rate at which each flow drains its credits. In other words, the flow rate is not proportional to the rate at which it drains its credit. Even with low bandwidth, a flow can drain its credit faster than a high bandwidth flow depending upon the cost to send packets. Therefore, the interval used in credit analysis in Section 5.1 will be calculated differently by a flow that drains its credits fastest.

The difference in delay analysis is that the service time for every packet is not equal but depends on the wireless condition. Fortunately, the model already incorporates differences in service times. For our analysis in Section 5.2.3, we used the same value for all s_i, but we can use different values to observe the differences in the transmission delay affected by the packet size and wireless connectivity. This implies that the flows with weak wireless connectivity are punished due to the fact that their frames take much longer to be transmitted. For overall QoS improvement, it is recommended to give relatively lower priority to those flows with weak connectivity because these flows tend to spend most of their time on losses and retransmissions. If the majority of the wireless time is spent on retransmissions, it brings down the overall throughput of the wireless network.

Let us walk through an example with flows 1, 2, and 3, where the rate ratio among the flow rates is still 2:3:5 respectively, but the cost of frames sent from each flow is now 15, 3, and 5 respectively. Therefore, the idea is that flow 1 should not get as much priority compared to when it has good wireless connectivity. We follow the same set of assumptions as discussed in the example in Section 4.2.

Flow 1 does not get priority any more since it costs too much to send the packet. While Table 4.1 showed that flow 1 was able to send its 2 packets by time index 4, Table 5.3 shows that flow 1 sends its 2nd packet at time index 8. In this example, flow 1 triggers the credit boost at time index 7 but flow 3 also triggers the credit boost at time 10. This is due to the fact that flow 3 has more packets to send compared to flow 1 and it can also drain its credits. Figure 5.5 depicts credits of flows 1, 2, and 3 over time. Flow 1 sometimes sits just below 0 because it does not have any packets to send at that time or some other backlogged flow has positive credit. Both flow 1 and 3 are triggering the credit boosts.

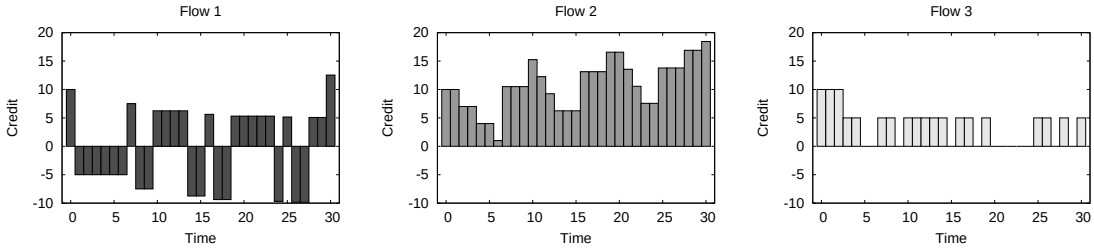


Figure 5.5: Wireless Example: Graph of credits vs. time of flows 1, 2 and 3

Table 5.4 shows queueing delays of each packet that entered the queue. With 10 packets entering every 10 time slots, FIFO provides 4.5 time slots of queueing delay on average to all the packets. Packets from flow 1 experience queueing delays of 0 and 7 time slots, packets from flow 2 experience queueing delays of 1, 3, and 5 time slots and packets from flow 3 experience queueing delays of 2, 4, 6, 8, and 9 time slots. In other words, average queueing delay for flow 1, 2, and 3, are 3.5, 3, and $\frac{29}{5}$ time slots, respectively. Unlike the previous example where the cost of each packet was the same for all flows, flow 1 gets higher queueing delay than flow 2. Because it costs more to serve packets from flow 1, flow

Table 5.3: Wireless Example: Table of credits vs. time of flows 1, 2 and 3

Time Index	Flow 1	Flow 2	Flow 3	Packet from Flow	Note
0	10	10	10	-	Start
1	-5	10	10	1	
2	-5	7	10	2	
3	-5	7	5	3	
4	-5	4	5	2	
5	-5	4	0	3	
6	-5	1	0	2	
7	7.5	10.5	10	-	Boost
	7.5	10.5	5	3	
8	-7.5	10.5	5	1	
9	-7.5	10.5	0	3	
10	6.25	15.25	10	-	Boost
	6.25	15.25	5	3	

2 gets higher priority than flow 1.

Table 5.4: Example: Table of queueing delays of each packet

Flow ID	Packet ID	Queueing Delay
1	1	0
	2	7
2	3	1
	4	3
	5	5
3	6	2
	7	4
	8	6
	9	8
	10	9

5.4 Bursts and Effect on CHAP Parameter I

It is important for any queue disciplines to prevent starvation of any particular flow. PPDQ is an attempt to make a strict priority discipline queue to prevent starvation. Under an extreme case, CHAP can starve a flow from having its packets served. This case occurs

when a new flow or a flow that has been inactive keeps entering the system before the packets from a flow in service are exhausted. However, the case of an infinite number of such flows entering the system serially is unlikely. This section discusses the derivation of a reasonable limit of such systems occurring at the same time.

The extreme case of the situation described above happens under the following conditions: 1) A delay-sensitive flow reaches just above zero credit; 2) n flows that have been inactive and have $2I$ credits start bursting at the same time.

Figure 5.6 depicts the credits of such flows over time for two different cases. In both cases, there is one delay-sensitive flow. This flow reaches near-zero credits at the end of region A. For the first case, a burst from one flow enters at the beginning of region B. The burst is enough to last over $2I$ and continues until the end of region B, following the red dotted line. Therefore, the delay-sensitive flow must wait $2I$ before its packets get service. However, there can be bursts from more than one flow. The second case demonstrates bursts from two flows at the same time, starting at the beginning of region B and following the dotted line, reaching 0 at the end of Region C. Because there are two flows bursting together, their packets get served in a round-robin fashion until they exhaust both of their credits. Because there are two flows, it takes twice as long as the first case to spend all the credits. The delay-sensitive flow gets its packets served after $4I$ at the end of region C. Therefore, we can conclude that with n flows bursting at the same time, the worst case waiting time for the delay-sensitive flow is $2nI$.

Although these cases can be extended to include n simultaneous bursts, we would like to set n to a reasonable limit. For possible candidates for bursty flows, we examine Web traffic and video traffic. Although Web traffic does not usually change state, it tends to be "bursty" in the sense that it starts a new flow, downloads a set of pages and objects and disappears. Video flows transmit Groups of Pictures (GoPs), which usually include one I frame and a bunch of P and B frames. Because I frames are much larger than P or B frames, video flows are bursty whenever I frames are transmitted. We investigate the time a flow takes to download these bursts of information under the maximum application

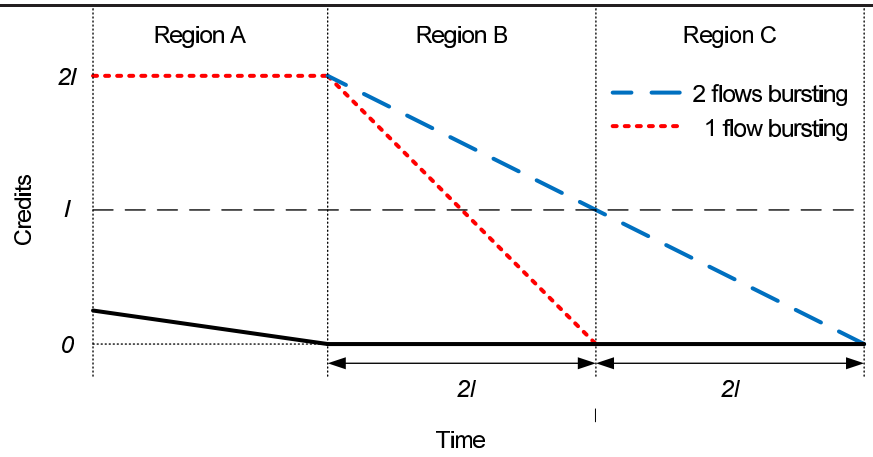


Figure 5.6: Bursts and Effects on Credits

throughput and compare it to mean idle time before the next burst.

Mean HTML container page size, mean embedded object size, mean number of embedded objects, and mean reading time are specified by [133]. Mean HTML container page size is 11,872 bytes, mean embedded object size is 12,460 bytes, mean number of embedded objects is 5, and mean reading time is 39.70 seconds. Using these mean values, we can calculate the total activity time. The total number of bytes to be retrieved is the sum of the HTML container page and embedded objects, which is 74,172 bytes. Assuming the maximum application throughput is about 27 Mbps in an IEEE 802.11g network, the time it takes to receive 74,172 bytes is $\frac{74172*8}{27*10^6} = 0.022$ seconds. In addition, there are two round trip times involved: the three-way handshake and the request transmission and data reception. Maier *et al.* find that there are three modes of round-trip times for TCP from residential networks in Europe: 13 ms, 100 ms and 160 ms. These round-trip times correspond to destinations in Europe, Eastern United States, and Western United States, respectively. We choose 100 ms for subsequent analysis. Therefore, the total activity time is $0.022 + (0.100 * 2) = 0.222$ seconds. On average, a flow will burst 222 ms out of $39.700 + 0.222$ seconds. This produces the activity fraction of $\frac{0.144}{39.922} = 0.0036$, meaning that there is 0.36% chance that one flow will burst at any given time.

We examined the video traces from [13] and [14] to find out the maximum frame sizes of each trace. The maximum frame sizes from Indiana Jones, Die Hard 1, Matrix 1 and

Lord of the Rings 1 are 50427, 37359, 63092, and 55971 bytes respectively. Therefore, we use the largest, 63092 bytes from Matrix 1, for our calculation as the worst case. Each of these movies was encoded using G16B3, meaning that a GoP consists of 16 frames with 3 B frames between I/P frames. Since Matrix 1 was encoded at 30 frames per second, it introduces an I frame every 533 ms. It takes 18.69 ms to transmit 63092 bytes over a wireless link with about 27 Mbps of application-level throughput. Therefore, the activity fraction is $\frac{0.01869}{0.533} = 0.0351$.

The probability that one flow bursts is p, where p is the activity fraction calculated above. Once a flow has burst, another flow can burst anytime between when the first burst starts and ends. Let $p(n)$ be the probability that n flows continue to starve another flow as demonstrated in Figure 5.6.

$$p(2) = p \int_0^p 1dx = p^2 \tag{5.38}$$

$$p(3) = p \int_0^p (\int_0^{p+x} 1dy)dx = 2p^3 \tag{5.39}$$

Equation 5.38 shows the probability of two flows delaying another flow based on the activity ratio p. Once a flow is bursting, the second flow can burst anywhere between 0 and p scaling the time period to 1 to match the activity ratio. Assuming that there is uniform probability that the second flow starts within the time period of 1, the probability that it will start between 0 and p, so that the second flow's burst duration overlaps with the first, is calculated by $\int_0^p dx$. Equation 5.39 shows the probability of three flows delaying another flow based on the activity ratio p.

Based on activity ratios of 0.0036 and 0.0351 from Web browsing and video streaming respectively, the corresponding probabilities can be calculated using the equations above. The probability of one flow bursting is 0.0036 and 0.0351 for Web browsing and video streaming, respectively. The probabilities of bursts are $0.0036^2 = 0.00001296$ and $0.0351^2 = 0.00123201$ for two flows and $2 \times 0.0036^3 = 9.33 \times 10^{-8}$ and $2 \times 0.0351^3 = 8.65 \times 10^{-5}$.

Therefore, we bound the maximum number of simultaneous bursts to 2 flows, which limits the waiting time for the delay sensitive flow to $4I$. As the human response time is in the order of 100 ms, we would like $4I$ to be less than 100 ms. This provides the upper bound for I as 25 ms.

For a lower bound, the parameter I should not be set too small because CHAP will not be able to distinguish one flow from another. Ideally, we would like CHAP to be effective for a reasonable burst length. We discussed the mean burst of Web flows and the maximum burst of video flows above. Assuming full packets, bursts of Web and video flows consist of 50 and 42 packets, respectively. Therefore, we would like I to be big enough to handle at least 50 packets. The time it takes to transmit 50 full packets can be calculated by dividing the total number of bytes of 50 packets with the maximum application throughput.

$$50 \text{ packets} \times 1500 \text{ bytes/packet} \times 8 \text{ bits/byte} / 27000000 \text{ Mbps} = 0.02222 \text{ s} \tag{5.40}$$

Considering that Web flows are usually new and assigned the default credit of I, I has to be at least 22.22 ms to support the average burst of Web flows. Equation 5.41 shows the lower and upper bounds for I. We set the parameter I to 25 ms.

$$22.22 \leq I \leq 25.00 \tag{5.41}$$

5.5 Summary

This chapter provides the analytical model of CHAP. Section 5.1 explores the steady state credit ranges for flows with different rates. The derived credit ranges demonstrate that they map the relationship between bandwidth and delay tolerance shown in Figure 2.1 to priorities as intended. Section 5.2 adapts the delay analysis of a Probabilistic Priority Discipline Queue (PPDQ) to derive the average queueing delays of flows with CHAP. The corresponding service probabilities are derived from the credit ranges from Section 5.1

and applied to the PPDQ model. The analytical model shows that the average queueing delay for a low-bandwidth flow is lower than a high-bandwidth flow and that the average queueing delays converge to the same point as the rates of two flows get closer to each other. Section 5.3 discusses how the model is affected by relaxing the assumption that the service rate is constant. An example demonstrates how the behavior of credits differs from the example presented in Section 4.2. Section 5.4 explores factors influencing the CHAP parameter I, which is based on the average nature of bursts. The recommended value of I is 25 ms.

Chapter 6

Simulation

This chapter discusses the validation and performance evaluation methods for CHAP. Section 6.1 describes the implementation details of simulation activities. Section 6.2 describes experimental strategies to validate NS2 simulations. Section 6.3 presents simulations to validate the mathematical model derived in Chapter 5. Section 6.4 covers the NS2 shadowing model for radio propagation and demonstrates the distribution of successful and unsuccessful frame transmissions in NS2. Sections 6.6-6.13 describe various combinations of simulation topologies and network traffic mixes to evaluate CHAP against FIFO and Strict Priority Queue (SPQ) using metrics explained in Section 2.5. FIFO, the default queue management mechanism in wireless access points, provides the baseline performance while SPQ provides the best performance, given an explicit, static classification of flows and corresponding priorities.

6.1 User Activities in NS2

This section describes the implementation details of the simulated user activities. Online games, Voice over IP (VoIP), video streaming, Web browsing, and file downloads are discussed in the following subsections.

6.1.1 Online Games

There are many genres of games such as First Person Shooter (FPS), Real Time Strategy (RTS), and Massively Multiplayer Online (MMO). FPS games are generally more fast-paced and interactive compared to other genres. The online game traffic is simulated with the built-in UDP packet generators following the traffic model for Halo 2, a popular FPS [22]. The game server sends a 72-byte payload to the game client every 40 ms while the game client sends a 44-byte payload to the game server every 40 ms.

6.1.2 Voice over IP

There are many protocols that Voice-over-IP (VoIP) applications use to allow users around the world to talk to one another using audio. Although each protocol has different packet sizes and intervals to transmit and receive audio data between users, the overall bandwidth of VoIP protocols are similar from one protocol to another. Therefore, G.711 is selected to simulate the VoIP network traffic in NS2. The built-in UDP traffic generator sends a constant rate stream of 200-byte packets between two nodes every 20 ms.

6.1.3 Video Streaming

There are many applications and protocols for video streaming along with a variety of methods to encode videos. We implement a video application to send video frames over a UDP connection. The sizes of frames are extracted from pre-encoded video traces. The simulated video application can also modify the frame sizes with three settings of Forward-Error-Correction (FEC): no FEC, small FEC, and large FEC [134]. FEC introduces redundancy to frames so that frames with lost packets can be repaired. Assume that a frame is composed of n packets and there are k redundant packets for repair. A total of $n + k$ packets are transmitted for the frame, tolerating loss of up to k packets. The original frame can be reconstructed with loss of up to any k packets. "No FEC" does not add any FEC bytes. Any loss of packets in a frame results in a frame loss. "Small FEC" adds one full packet (1500 bytes) to all the I-frames. Therefore, all the I-frames can be reconstructed

with up to one packet loss. "Large FEC" adds 15% of the bytes in a frame to all types of frames.

The video application uses the Indiana Jones I movie trace encoded in a single layer Common Intermediate Format (CIF) from Arizona State University [13, 14]. The movie has a resolution of 352 × 288 pixels with 30 frames per second. The length of Group of Pictures (GoP) is 16 frames with 3 B-frames between each pair of I/P frames. The quantization scales used for I, P, and B frames are 10, 10, and 12, respectively. Only the second layer with the full 30 frames per second is used by the video application to send the movie.

6.1.4 Web Browsing

There are many analyses of Web traffic through core routers, but these do not describe the Web traffic in the user's perspective. Lee and Gupta model the Web traffic based on user behavior [133]. Table 6.1 lists the HTTP model and its parameters. Following this model, the built-in TCP data generator is used to model the Web traffic. Most of the random number generators are already built in NS2 but the missing Gamma random number generator was written in TCL.

Table 6.1: HTTP Model Parameters

Parameters	Mean	Std. Dev.	Best Fit (Parameters)
HTML Object Size	11872 B (Max 2 MB)	38306	Truncated Lognormal (μ 7.90272, σ 1.7643)
Embedded Object Size	12460 B (Max 6 MB)	116050	Truncated Lognormal (μ 7.51384, σ 2.17454)
Number of Embedded Objects	5.07 (Max 300)	-	Gamma (κ 0.141385, θ 40.3257)
Reading Time	39.70s (Max 10,000s)	324.92	Lognormal (μ -0.495204, σ 2.7731)
Request Size	318.59 B	179.46	Uniform (350 B)

The simulated Web application uses a built-in TCP traffic generator to connect to the Web server. The Web application transmits a request to the Web server. Upon receiving

the request, the Web server responds with one HTML object. After receiving the HTML object, the Web application computes the number of embedded objects and the size of each embedded object. If there is at least one embedded object, the Web application transmits another request to the Web server. Finally, the Web server responds with an object whose size is the sum of all the embedded objects. The sequence of Web requests and responses assumes that 1) all the objects in a Web page are on one server; and 2) the Web application and server use HTTP/1.1 for a persistent connection and pipelining.

6.1.5 Downloading Files

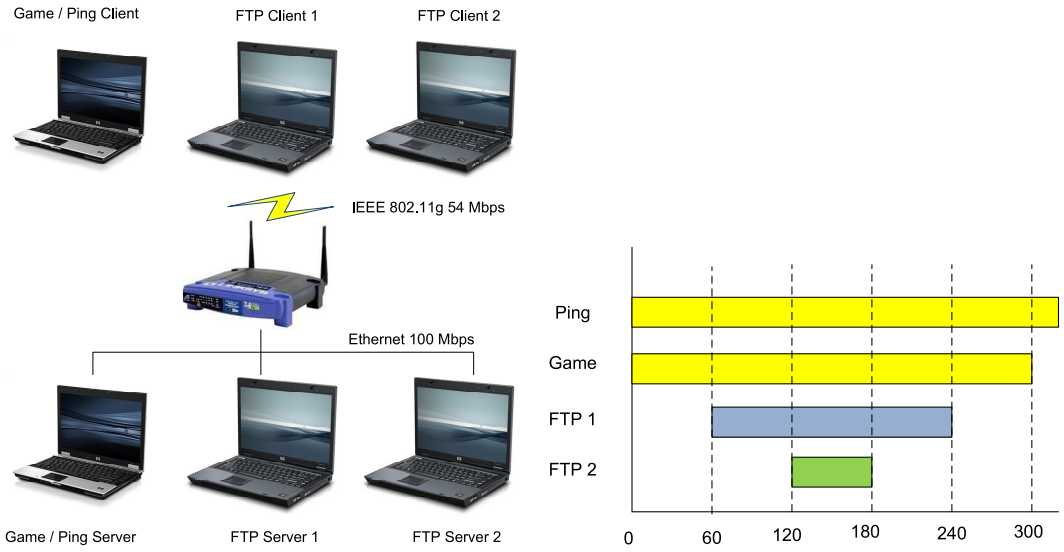
We simulate two types of file downloads: FTP and P2P. The FTP traffic is generated using the FTP application in NS2 with TCP. The TCP congestion control mechanisms used in modern operating systems vary from one operating system to another. Microsoft Windows supports TCP Compound [18] while Linux 2.6.x kernel supports TCP CUBIC [17]. While Windows supports TCP Compound, it is not enabled by default, and the default TCP congestion control mechanism for Windows is unknown. Because the Linux 2.6.x kernel uses TCP CUBIC by default, we use TCP CUBIC with the built-in FTP application.

Unlike traditional file transfer applications, peer-to-peer (P2P) applications connect to "peers" to distribute files. Not only do P2P applications download files, they also provide files or parts of files to other peers. Therefore, it is a combination of both downstream and upstream activity. As discussed earlier in the dissertation (Section 4.2), we concentrate only on the downstream traffic part of P2P applications. Because P2P applications are able to share their own files with other peers, they use multiple flows to download a single file. Basher *et al.* examine P2P flow characteristics in a campus environment [**?**]. The results show that Gnutella uses only one flow to download a file while BitTorrent uses more flows. 50% of BitTorrent clients use fewer than 20 concurrent flows. Maier *et al.* examine residential DSL characteristics in Europe [73]. Their study of concurrent flows of BitTorrent and eDonkey clients shows that eDonkey clients use about 10 concurrent flows while BitTorrent clients use 10 to 15 concurrent flows. Therefore, we use 10 concurrent

built-in FTP applications sending data from distinct sources to a single destination.

6.2 Simulation Validation

This section describes experimental strategies to validate NS2 simulations. It is important to validate simulation results with experimental results to reinforce the validity of simulation results presented throughout the rest of this chapter.



(a) Equipment Setup (b) Flow Schedule

Figure 6.1: Simulation Validation Setup

Figure 6.1 depicts the experiment setup and flow schedule. Six laptops are connected in an internal network using a wireless access point. Three laptops are connected over the 100 Mbps Ethernet connections while the other three laptops are connected over the IEEE 802.11g 54 Mbps connections. Table 6.2 lists all the hardware details of each device used in the experiment. The game/ping server and client run Microsoft Windows XP SP3 while the FTP servers and clients run Debian Linux (Lenny). Linux is chosen for the FTP servers and clients to ensure they use TCP CUBIC for data transfers. The game application is simulated using *MGEN*[1] to send UDP packets at a fixed interval following the average packet size and interval from the Halo 2 traffic model [22]. The server and

[1]http://cs.itd.nrl.navy.mil/work/mgen/

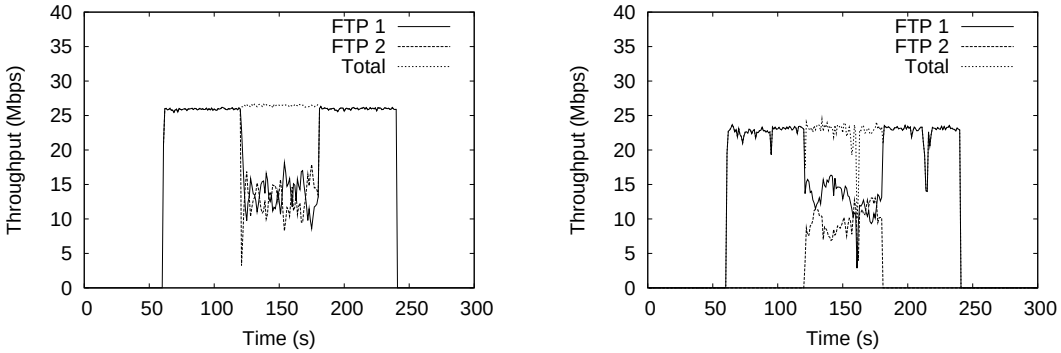
client send 72-byte and 44-byte packets, respectively, every 40 ms. The time differences in the machines makes it difficult to measure one-way delays. *UDP Ping*[2] is used to measure the round-trip delay from the client to the server and later halved as an estimated measure of one-way delay. The FTP application is written as a simple TCP server and client to produce per-packet logs for analysis. Since the FTP application is run on Linux, the TCP congestion control mechanism used is CUBIC. Most of the network traffic is downstream from the servers to the clients and the only upstream traffic consists of TCP ACKs of the FTP application and the game client UDP packets. In the experiment, the emulated game traffic runs for 5 minutes starting at 0 seconds and ending at 300 seconds. UDP Ping runs for the full duration of the experiment. Two bulk file downloads run for 3 and 1 minutes each, starting at 60 and 120 seconds and ending at 240 and 180 seconds, respectively. The same topology and flow schedule are set up in NS2 without the hardware details. RTS/CTS exchange is disabled for IEEE 802.11g in both simulation and experiment.

Table 6.2: Simulation Validation Equipment Specification

Role	Hardware and Software
Game/Ping Server/Client	HP Elitebook 6930p Intel Core2Duo T7300 @ 2.00 GHz 2 GB of RAM Intel Wireless Wifi 3945ABG Windows Vista SP1 (32bit)
FTP Servers/Clients	HP Compaq 6710b Intel Core2Duo P8600 @ 2.40 GHz 2 GB of RAM Intel Wireless Wifi 4965AGN Debian Linux 5.0 Kernel 2.6.26 (32bit)
Wireless Access Point	Linksys WRT54GL Rev. 1.1 Official Firmware 4.30.12

Figure 6.2 depicts the throughputs of each application run in both simulation and experiment. Throughputs in simulation and experiment are measured by counting the number of bytes received at the destination node and machine, respectively. The total

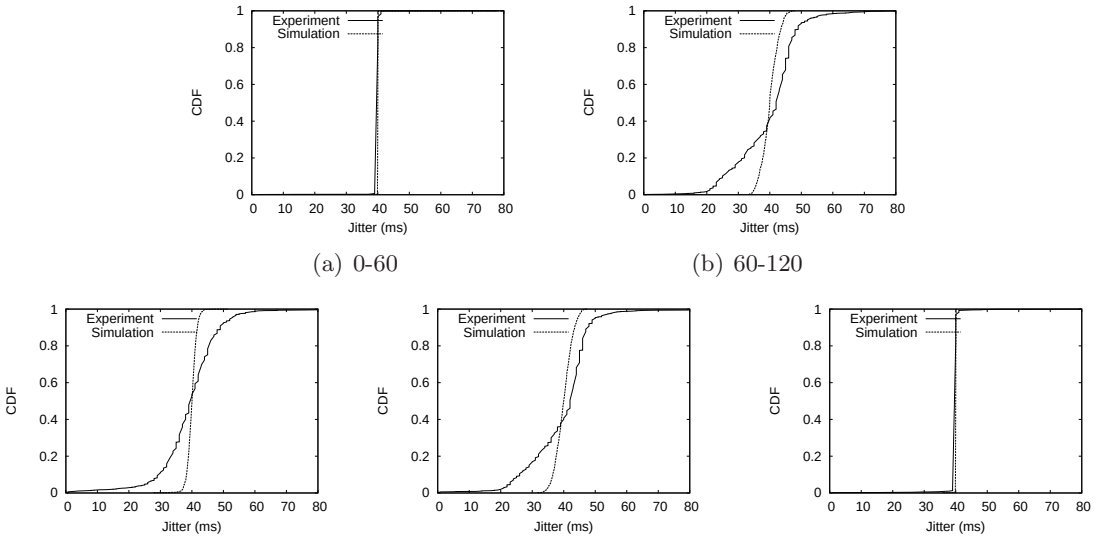
[2]http://perform.wpi.edu/tools/



(a) Throughput (Simulation) (b) Throughput (Experiment)

Figure 6.2: Simulation vs. Experiment: Throughput

throughput in the simulation is about 27 Mbps while the total throughput in the experiment is about 23 Mbps. The first bulk file download is enough to saturate the wireless bandwidth by itself. When the second bulk file download starts, they share the bandwidths equally with some noise around their fair share in both simulation and experiment.



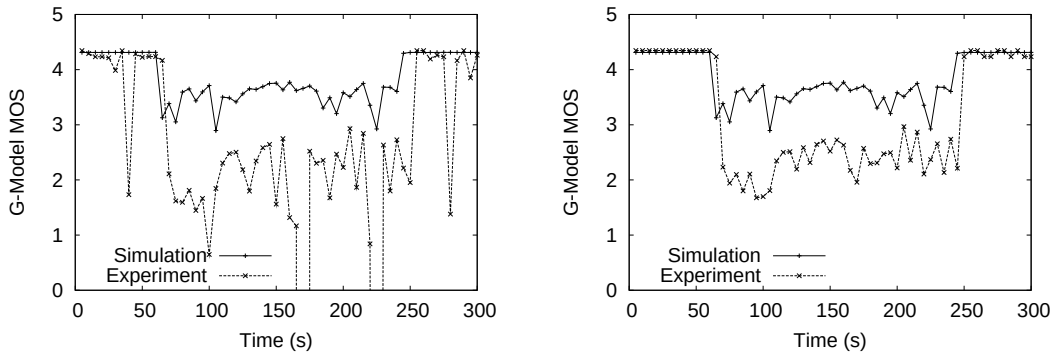
(a) 0-60 (b) 60-120

(c) 120-180 (d) 180-240 (e) 240-300

Figure 6.3: Simulation vs. Experiment: Inter-arrival Times of Game Packets

Figure 6.3 depicts the inter-arrival times of game packets at the game client in five 1-minute intervals. The expected inter-arrival time of game packets is 40 ms because the game server sends its packets every 40 ms. In the first and last minutes, the distributions

in simulation and experiment are almost identical. However, in the middle 3 minutes, the distributions differ from the simulation to the experiment. The simulation results show a tighter range of inter-arrival times than the experiment.



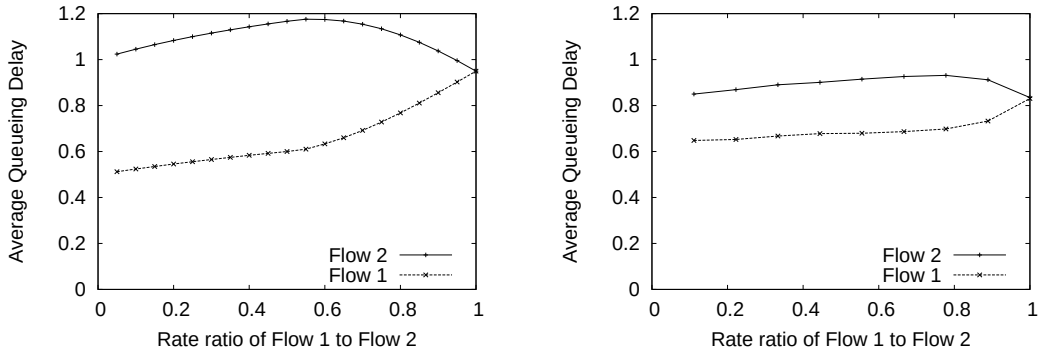
(a) All Data (b) Filtered Data

Figure 6.4: Simulation vs. Experiment: G-model MOS

Figure 6.4(a) depicts the G-model MOS calculated every 5 seconds throughout the simulation and experiment runs. The simulation and experiment show the same trend over the 5 minutes where G-model MOS drops when the first bulk file download starts and stays low until it ends after 3 minutes. The G-model MOS shows more variance in the experiment than in the simulation. The G-model MOS in the experiment is lower than the G-model MOS in the simulation in general due to the greater variation in inter-arrival times of game packets. We believe that the noise in the experiment is caused by different TCP implementations and/or factors that are not simulated in NS2 such as system level overheads and interference from cordless phones and other wireless networks. Further investigation is necessary to pinpoint the exact causes of the differences and is outside the scope of this work. In order to compensate for the noise, the top and bottom 2.5% of jitter data within each one minute intervals are filtered as outliers. The filtered data produce a much cleaner G-model MOS line, shown in Figure 6.4(b).

6.3 Model Validation

An analytical model for average queueing delay for CHAP is derived in Chapter 5. An NS2 simulation is run to validate this model, which uses the delay analysis of a Probabilistic Priority Discipline Queue (PPDQ) along with derived service probabilities based on CHAP credits. The analysis of a PPDQ assumes M/G/1 queues for each priority, where the inter-arrival times of each incoming traffic flow follow an exponential distribution. Therefore, the NS2 Telnet application, which utilizes an exponential distribution to compute inter-packet times, is used on top of a UDP protocol to generate packets. Wireless nodes are situated 1m away from the wireless access point and connected over an IEEE 802.11g 54 Mbps network with RTS/CTS disabled. 100 Mbps Ethernet connections with 1 ms latency connect the wireless AP to a switch and the switch to two wired nodes. Each wired node generates UDP packets using the built-in Telnet application. The network traffic with a slower rate is "Flow 1" while the one with a faster rate is "Flow 2". Flow 1 is varied such that the ratio of rates of Flow 1 to Flow 2 increases in steps of $\frac{1}{9}$ while keeping the total rate at 13.5 Mbps. 13.5 Mbps is chosen to match $\rho = 0.5$ in the analysis because the IEEE 802.11g network can accommodate the throughput of about 27 Mbps in NS2. The average rates of flows are configured by calculating the average inter-packet times for each Telnet application in NS2.



(a) Analytical Model (b) Simulation

Figure 6.5: Mean Queueing Delay from Analytical Model and Simulation

Figure 6.5 depicts the average queueing delay from the analytical model and the sim-

ulations. Figure 6.5(a) shows the average queueing delay from the derived model in Chapter 5. From the simulation result, the average queueing delay is calculated for every packet sent from Flows 1 and 2. Then, it is normalized relative to the transmission time of a packet with 1500 byte payload over the wireless network, which is 306 μs ($= 50 + \frac{(1500+34)\times 8\times 10^6}{54\times 10^6} + 10 + \frac{14\times 8\times 10^6}{6\times 10^6}$)[3]. Figure 6.5(b) shows the normalized average queueing delay from the simulation. Both graphs show similar trend as the rate of Flow 1 gets closer to the rate of Flow 2. When the rates of Flow 1 and Flow 2 are equal, they get about the same average queueing delay in both the analytical model and simulation.

6.4 Shadowing Model and Transmission Statistics over Distance

NS2 supports three types of radio propagation models: free space, two-ray ground reflection and shadowing. The free space model and two-ray ground reflection models provide a deterministic signal degradation over distance. Therefore, they both represent wireless communication in a perfect, open circle. In the real world, many factors contribute to wireless signal degradation and introduce a random nature to the signal strength at the receiver.

The shadowing model is a more general model that consists of two parts: path loss and variation. The path loss model predicts $P_r(d)$, the mean received power at distance d and uses $P_r(d_0)$, a close-in distance d_0 as a reference.

$$\frac{P_r(d_0)}{P_r(d)} = \left(\frac{d}{d_0}\right)^{\beta} \tag{6.1}$$

Equation 6.1 demonstrates the relative relationship between $P_r(d)$ and $P_r(d_0)$ where β is the path loss exponent. Equation 6.1 is equivalent to the free space model when $\beta = 2$. The greater the path loss exponent is, the more obstructions there are, resulting in faster decrease in averaged received power over distance. Equation 6.1 can be transformed to

[3]DATA and ACK frames are transmitted at 54 Mbps and 6 Mbps, respectively.

Equation 6.2 because the path loss is measured in dB.

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta log\left(\frac{d}{d_0}\right) \tag{6.2}$$

Variation is added to Equation 6.2 with a log-normal random variable. X_{dB} is a Gaussian random variable with zero mean and standard deviation σ_{dB}, the shadowing deviation. Equation 6.3 represents the overall shadowing model.

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta log\left(\frac{d}{d_0}\right) + X_{dB} \tag{6.3}$$

Table 6.3: Some typical values of path loss exponent β

Environment	β
Outdoor - Free space	2
Outdoor - Shadowed urban area	2.7 to 5
Indoor - Line-of-sight	1.6 to 1.8
Indoor - Obstructed	4 to 6

Table 6.4: Some typical values of shadowing deviation σ_{db}

Environment	σ_{dB} (dB)
Outdoor	4 to 12
Office, hard partition	7
Office, soft partition	9.6
Factory, line-of-sight	3 to 6
Factory, obstructed	6.8

The NS2 manual[4] lists typical values for path loss exponent and shadowing deviation. Table 6.3 depicts some typical values of path loss exponent β while Table 6.4 depicts some typical values for shadowing deviation.

Figure 6.6 depicts the loss statistics from sending a UDP stream of packets at a constant rate over a IEEE 802.11g infrastructure network with one wireless node. RTS/CTS is

[4]http://www.isi.edu/nsnam/ns/ns-documentation.html

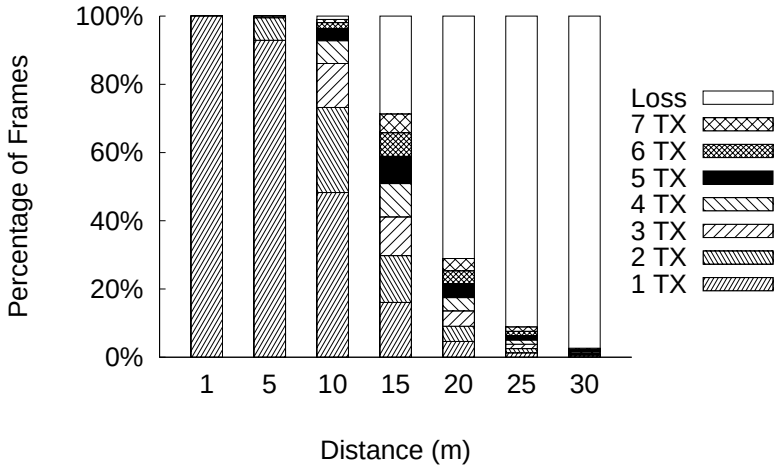
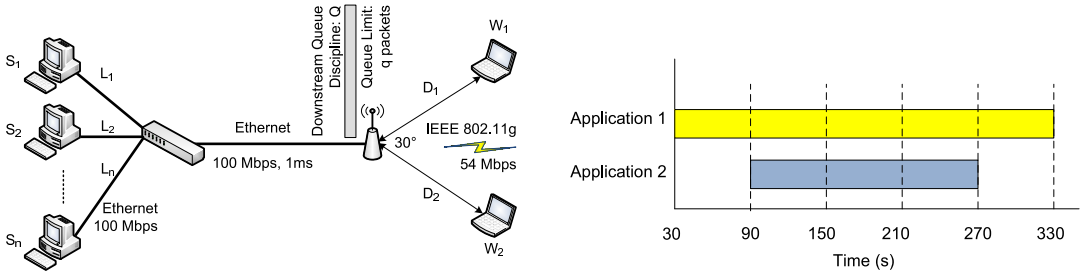


Figure 6.6: Transmission Statistics over Distances

disabled and the shadowing model is used for the radio propagation. The wireless node is located at distances ranging from 1m to 30m. All frame transmissions are successful in one transmission at 1m. At 5m, 92.9% and 6.5% of frames are successful in one transmission and two transmissions, respectively. At 10m, the AP transmitted 48.3%, 24.9%, and 12.9% of frames in one, two, and three transmissions respectively, while 1.0% of frames is lost after 7 transmissions. At 30m, 97.3% of frames is lost after 7 transmissions.

6.5 Simulation Topology



(a) Simulation Topology

(b) Flow Schedule

Figure 6.7: Simulation Setup

Figure 6.7 depicts the simulation topology and flow schedule for all the simulation runs. There are two wireless nodes connected to the access point (AP) with distances D_1 and

D_2 respectively. The wireless nodes and AP are connected over a single channel, IEEE 802.11g infrastructure network at 54 Mbps. RTS/CTS frame exchange is disabled. The AP is configured to send beacon frames every 100 ms. The radio propagation model for wireless transmission is the shadowing model described in Section 6.4. The parameters for the shadowing model are 3.0 and 7.0 for the path loss exponent and standard deviation, respectively. The path loss exponent of 3.0 corresponds to an obstructed indoor environment while standard deviation of 7.0 corresponds to an office with hard partitions. The rate adaptation is not enabled due to lack of NS2 support for the simulations. The AP is connected to a switch over a symmetric 100 Mbps Ethernet line with 1 ms latency. The switch is connected to multiple wired nodes over 100 Mbps Ethernet lines with varying latencies of L_1 through L_n. The AP uses different queueing disciplines for the downstream traffic coming from the wired nodes S_1 through S_n and going to the wireless nodes W_1 and W_2. The queueing disciplines used are First-In-First-Out (FIFO), Strict Priority Queue (SPQ), and CHAP. The queue size limit is configured to q packets. Unless specified, the queue size at the AP is set to 35 packets [135], the latencies to the wired nodes are 1 ms, and the distances between the AP and the wireless nodes are 1m. CHAP uses 25 ms for the parameter I, and SPQ is configured to prioritize Application 1 over Application 2.

There are two types of network traffic in the simulation topology. Each wireless node runs one application. Wireless node W_1 runs Application 1, which starts at 30 seconds after the simulation starts and ends at 330 seconds for the total duration of 5 minutes. Wireless node W_2 runs Application 2, which starts at 90 seconds and ends at 270 seconds for the total duration of 3 minutes. Two applications run concurrently between 90 seconds and 270 seconds. All the measurements and quality metrics between 90 and 270 seconds are used for analysis.

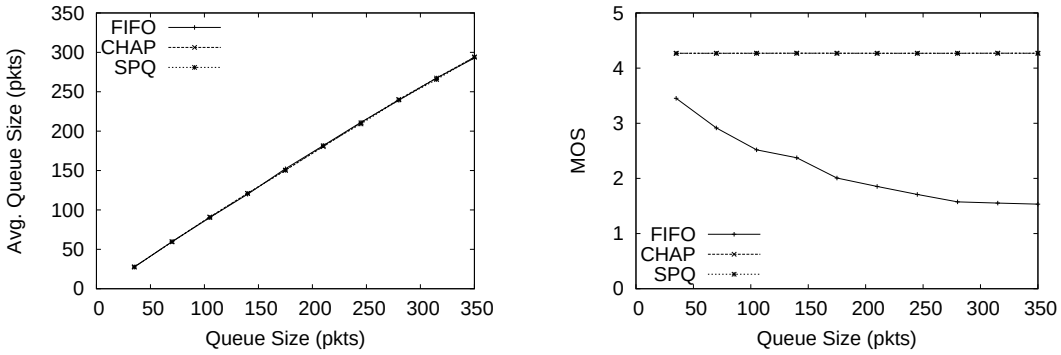
6.6 Queue Size

This section describes a series of simulation runs with varying queue sizes. Table 6.5 lists all the parameters of the simulation runs. Residential and commercial APs have different

queue sizes, and this set of simulation results demonstrates how different queue sizes affect the application quality under FIFO, CHAP, and SPQ. The queue size ranges from 35 to 350 packets in increments of 35 packets.

Table 6.5: Parameters for Queue Size Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS), VoIP, Video, Web
S_2, W_2	FTP
L_1	1 ms
L_2	1 ms
D_1	1m
D_2	1m
q	35, 70, 105,...,350 packets
Q	FIFO, CHAP, SPQ

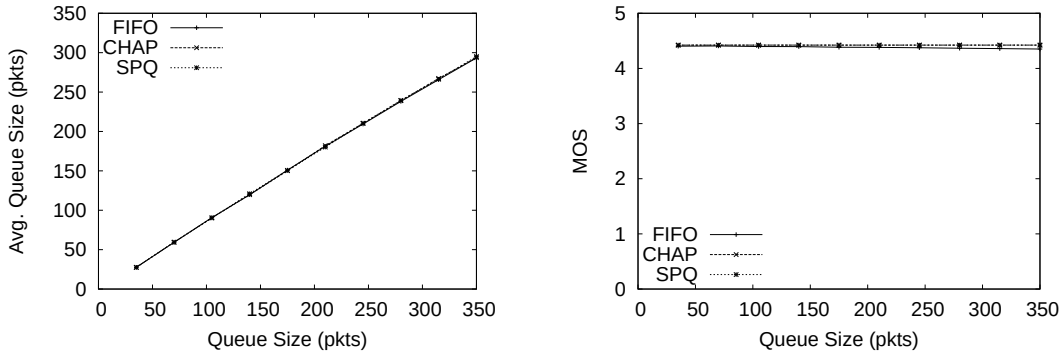


(a) Avg. Queue Size (b) G-model MOS

Figure 6.8: Improvement of Game vs. Queue Size

Figure 6.8 depicts the average queue size at the wireless access point and the G-model MOS of the game application. The average queue size demonstrates that the queue is almost always full as the queue size increases for all three queueing disciplines in the presence of the game application. The G-model MOS of the game application with the FIFO queue shows degradation in quality from 3.5 to 1.5 as the queue size increases. The increase in the queueing delay contributes to the degradation in quality of the game application. On the contrary, the game application performance stays constant at 4.27

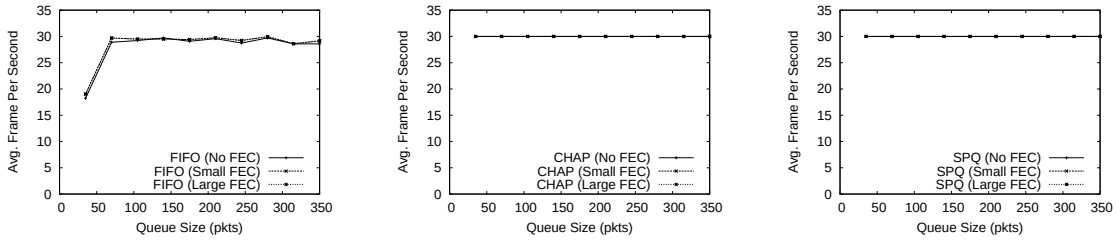
with both CHAP and SPQ because CHAP and SPQ prioritize the game traffic regardless of the queue size.



(a) Avg. Queue Size (b) E-model MOS

Figure 6.9: Improvement of VoIP vs. Queue Size

Figure 6.9 depicts the average queue size at the wireless access point and the E-model MOS of the VoIP application. The average queue size demonstrates that the queue is almost always full as the queue size increases for all three queueing disciplines in the presence of the VoIP application. The E-model MOS of the VoIP application with the FIFO queue shows little degradation in quality as the queue size increases. The only factor that contributes to the degradation in the E-model MOS in the simulation scenario is packet loss. The VoIP application performance stays constant at 4.3 with both CHAP and SPQ because CHAP and SPQ prioritize the VoIP traffic regardless of the queue size.

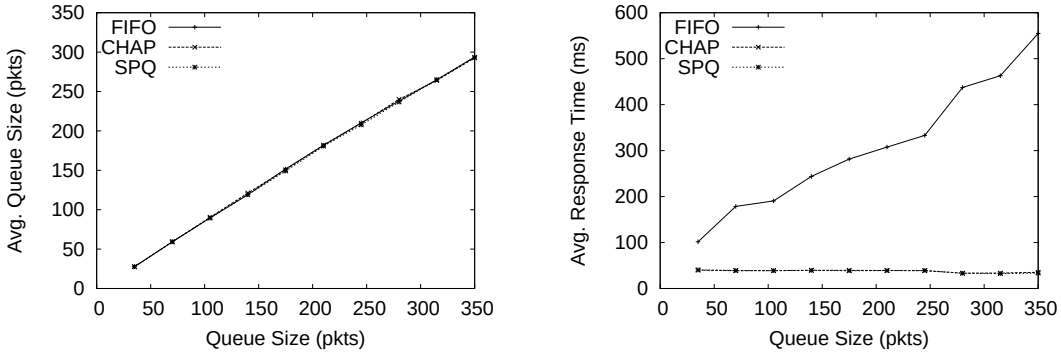


(a) Frame Rate (FIFO) (b) Frame Rate (CHAP) (c) Frame Rate (SPQ)

Figure 6.10: Improvement of Video vs. Queue Size

Figure 6.10 depicts the average frame rate of the video streaming application. The video frame rate with FIFO increases as the queue size increases as the probability of

losing frames is lower with a larger buffer. The use of small and large FEC helps the video frame rate for FIFO but only by about 2 frames. CHAP and SPQ demonstrate a constant rate of 30 frames per second regardless of the amount of FEC. Since the benefits of FEC are marginal, no FEC is used for subsequent video results, unless otherwise stated.



(a) Avg. Queue Size (b) Response Time

Figure 6.11: Improvement of Web vs. Queue Size

Figure 6.11 depicts the average queue size at the wireless access point and the average response time of the Web application. The average queue size demonstrates that the queue is almost always full as the queue size increases for all three queueing disciplines in presence of the Web application. The average response time of the Web application with the FIFO queue increases as the queue size increases. This is due to the increase in the queueing delay as the queue size increases. On the contrary, the Web application responds consistently faster with both CHAP and SPQ because CHAP and SPQ prioritize the Web traffic regardless of the queue size.

6.7 Distances

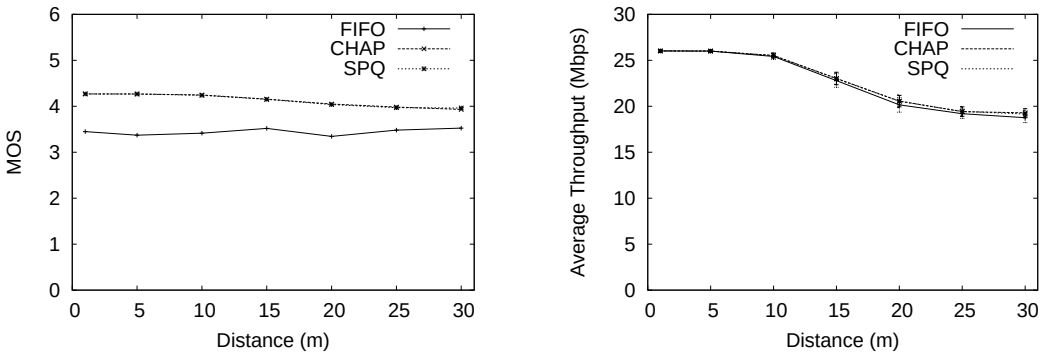
This section describes a series of simulation runs with varying distances. Physical distances between the AP and the wireless nodes affect the nodes' wireless connectivity. Even one node with poor connectivity can cause performance degradation of the entire wireless network. This set of simulation results demonstrates how varying distances of different nodes affect the application quality under FIFO, CHAP, and SPQ.

6.7.1 Distance of the Application Node under Test

This section discusses how the application quality is affected by varying distances of the node running the application under test. The FTP application node is placed at 1m away from the AP. Table 6.6 lists all the parameters of the simulation runs.

Table 6.6: Parameters for Application Node Distance Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS), VoIP, Video, Web
S_2, W_2	FTP
L_1	1 ms
L_2	1 ms
D_1	1,5,10,15,20,25,30m
D_2	1m
q	35 packets
Q	FIFO, CHAP, SPQ



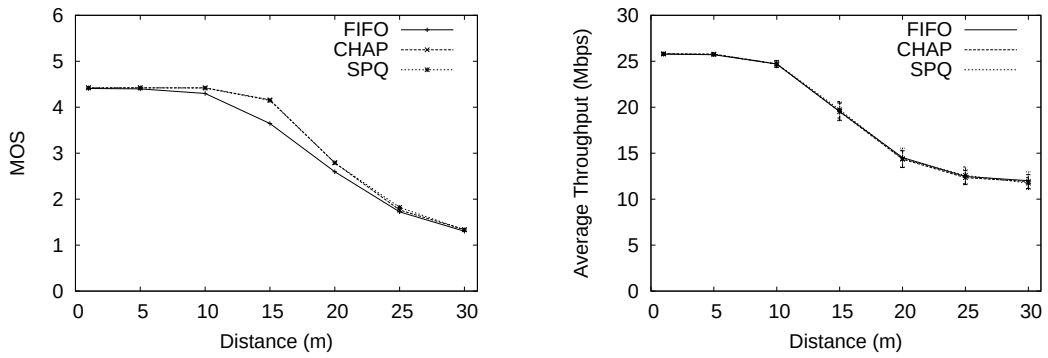
(a) G-model MOS

(b) FTP Throughput

Figure 6.12: Application Distance Case (Game)

Figure 6.12 depicts how the G-model MOS of the game application and the throughput of the FTP application are affected by the distance of the game application node. The G-model MOS of the game application stays relatively constant at 3.5, 4.27, and 4.27 for FIFO, CHAP, and SPQ respectively. Although the game application experiences higher packet loss rate, the G-model assumes that the game application is capable of handling losses. The FTP throughput decreases as the game application node moves farther away

from the AP because the game packets take a longer transmission time due to the signal attenuation and retransmissions. The FTP performance degradation is about the same for FIFO, CHAP and SPQ.



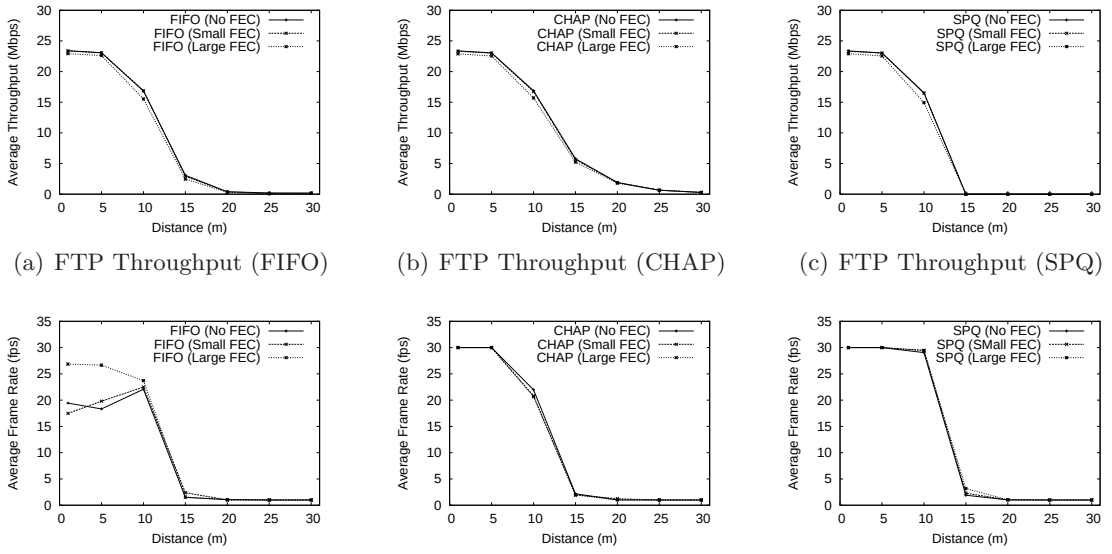
(a) E-model MOS

(b) FTP Throughput

Figure 6.13: Application Distance Case (VoIP)

Figure 6.13 depicts how the E-model MOS of the VoIP application and the throughput of the FTP application are affected by the distance of the VoIP application node. The E-model MOS of the VoIP application degrades as the VoIP application node moves farther away from the AP. Increasing packet loss rate due to increases in the distance contributes to the degradation of the VoIP application quality. Like in the game application scenario, the FTP throughput decreases as the VoIP application node moves farther away from the AP because the VoIP packets take longer transmission times due to the signal attenuation and retransmissions. Although the FTP performance degradation is similar across FIFO, CHAP, and SPQ, it is worse compared to the game scenario because the VoIP application sends and receives packets at twice the rate of the game application.

Figure 6.14 depicts how the FTP throughputs and the frame rate of the video application are affected by the distance of the video application node. The total throughput of the wireless network degrades similarly for FIFO and SPQ. FIFO, CHAP, and SPQ degrade the FTP throughput differently. The FTP throughput under FIFO reaches 0 Mbps at 20m while the FTP throughput under SPQ reaches 0 Mbps at 15m. CHAP degrades the FTP throughput of the wireless network more gracefully compared to FIFO and SPQ because



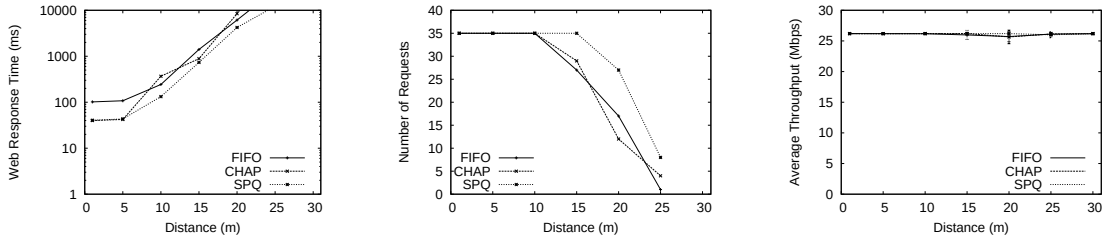
(a) FTP Throughput (FIFO) (b) FTP Throughput (CHAP) (c) FTP Throughput (SPQ)

(d) Frames per Second (FIFO) (e) Frames per Second (CHAP) (f) Frames per Second (SPQ)

Figure 6.14: Application Distance Case (Video)

CHAP adjusts the priority of the video traffic as the video application node moves farther away from the access point. Increases in the transmission time for video packets increase the cost in credits, pulling the video application's priority down. Therefore, the FTP application achieves higher throughput at distances greater than 10m with CHAP compared to FIFO and SPQ. Increases in distance of the video application also affect the degradation of the video application. The video is no longer viewable at distances greater than 15m with the achievable rate of about 2 frames per second. Therefore, it is reasonable to stop prioritizing the video traffic at such distances. Under FIFO, large FEC helps with losses when the video application is close to the AP. As the video application node is further away from the AP, the benefit of FEC decreases due to high loss rates. FEC does not help improve frame rates under CHAP and SPQ.

Figure 6.15 depicts how the distance of the Web application affects the average response time, the number of completed requests, and the throughput of the FTP application. The average response time increases as the distance of the Web application increases. SPQ provides the shortest average response time as it is manually configured to always prioritize the Web traffic regardless of the Web application node's distance to the access



(a) Response Time

(b) Number of Completed Requests

(c) FTP Throughput

Figure 6.15: Application Distance Case (Web)

point. Because of differences in the number of completed requests under FIFO, CHAP, and SPQ, the average response time is calculated with only the completed requests that exist under all three scenarios.

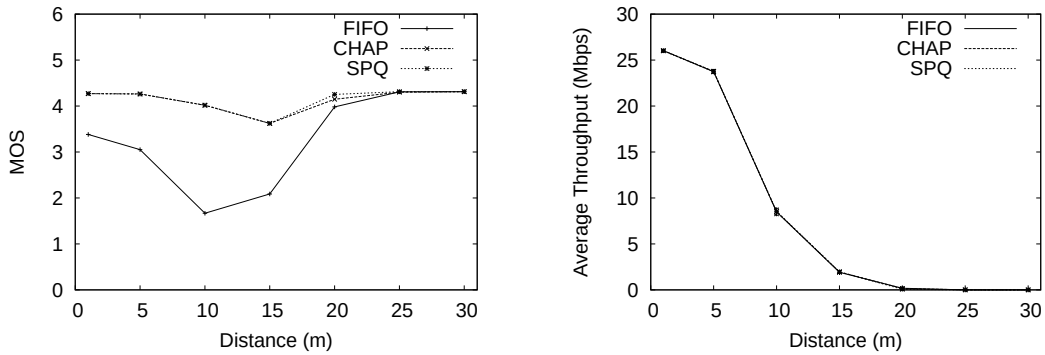
6.7.2 Distance of the FTP Node

This section discusses how the application quality is affected by varying distances of the FTP application node. The node running the application under test is placed at 1m away from the AP. Table 6.7 lists all the parameters of the simulation runs.

Table 6.7: Parameters for FTP Node Distance Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS), VoIP, Video, Web
S_2, W_2	FTP
L_1	1 ms
L_2	1 ms
D_1	1m
D_2	1,5,10,15,20,25,30m
q	35 packets
Q	FIFO, CHAP, SPQ

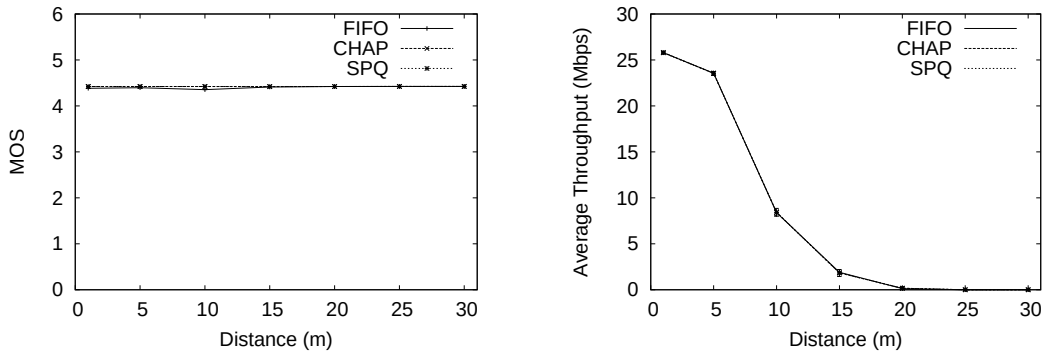
Figure 6.16 depicts how the G-model MOS of the game application and the throughput of the FTP application are affected by the distance of the FTP application node. The G-model MOS of the game application decreases up to a certain distance and then increases for FIFO, CHAP, and SPQ. The game application experiences longer queueing delays and



(a) G-model MOS (b) FTP Throughput

Figure 6.16: FTP Distance Case (Game)

higher variation in jitter. The G-model MOS for FIFO, CHAP, and SPQ increases to the maximum quality past 25 m because the FTP application is hardly transmitting any packets. The G-model MOS for CHAP and SPQ are relatively unaffected compared to FIFO over all distances. The FTP throughput decreases as the FTP application node moves farther away from the AP because the FTP packets take longer transmission time due to the signal attenuation and retransmissions. The FTP performance degradation is about the same for FIFO, CHAP, and SPQ.

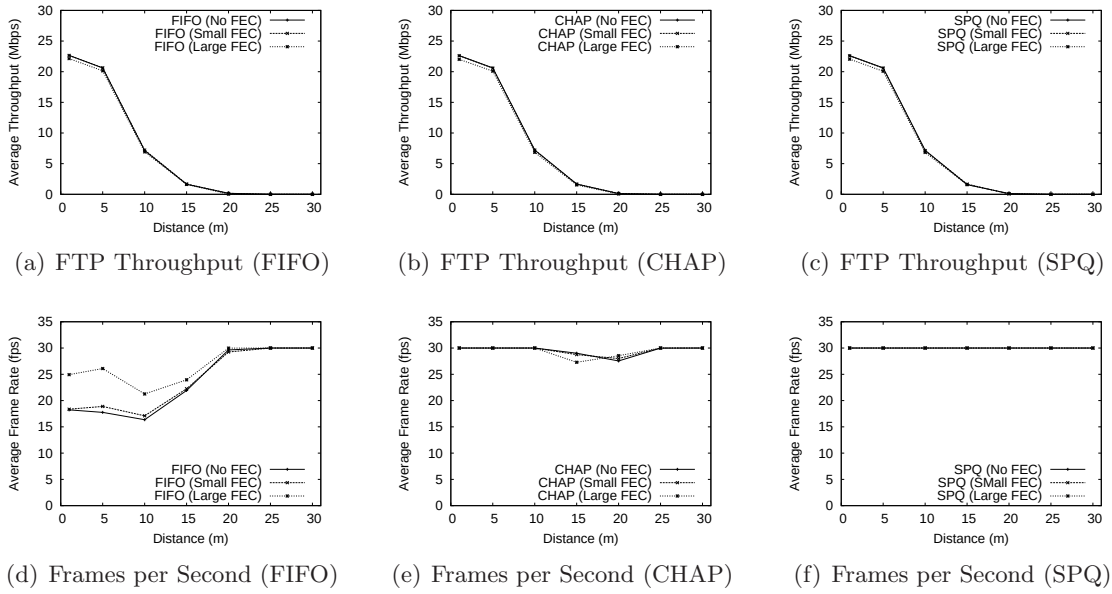


(a) E-model MOS (b) FTP Throughput

Figure 6.17: FTP Distance Case (VoIP)

Figure 6.17 depicts how the E-model MOS of the VoIP application and the throughput of the FTP application are affected by the distance of the FTP application node. Unlike the G-model MOS, the E-model MOS is resilient to delays up to 177.3 ms while it is prone

to packet losses. Because the delays do not exceed 177.3 ms and there is no VoIP packet loss, the E-model MOS of the VoIP application stays constant at all distances for FIFO, CHAP and SPQ. The FTP throughput decreases as the FTP application node moves farther away from the AP because the FTP packets take longer transmission time due to the signal attenuation and retransmissions. The FTP performance degradation is about the same for FIFO, CHAP, and SPQ.

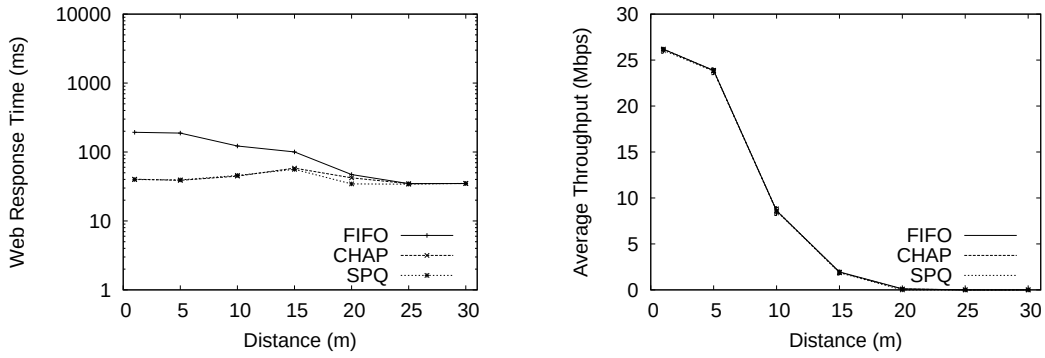


(a) FTP Throughput (FIFO) (b) FTP Throughput (CHAP) (c) FTP Throughput (SPQ)

(d) Frames per Second (FIFO) (e) Frames per Second (CHAP) (f) Frames per Second (SPQ)

Figure 6.18: FTP Distance Case (Video)

Figure 6.18 depicts how the FTP throughputs and the frame rate of the video application are affected by the distance of the FTP application node. The FTP throughput decreases as the FTP application node is further away from the AP. Viewable frame rate starts low under FIFO when the FTP application node is close to the AP. The frame rate increases under FIFO as the FTP application moves away, allowing the video application enough bandwidth to send all its frames without losses. CHAP and SPQ are able to provide close to 30 frames per second regardless of the distance of the FTP application node. Under FIFO, large FEC helps with losses when the video application is close to the AP. As the video application node is further away from the AP, the benefit of FEC decreases due to high loss rates. FEC does not help improve frame rates under CHAP and SPQ.



(a) Response Time

(b) FTP Throughput

Figure 6.19: FTP Distance Case (Web)

Figure 6.19 depicts how the distance of the FTP application affects the average response time and the throughput of the FTP application. The FTP throughput decreases as the FTP application node is further away from the AP. The average response time for FIFO decreases as the distance of the FTP application increases. CHAP and SPQ provide average response times lower than FIFO. SPQ provides the shortest average response time as it is manually configured to always prioritize the Web traffic regardless of the FTP application node's distance to the access point.

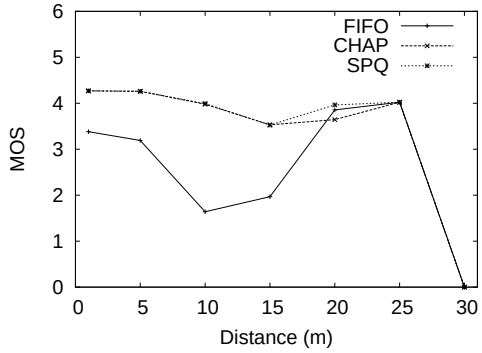
6.7.3 Distance of Both Nodes

This section discusses how the application quality is affected by varying distances of both nodes. In each simulation run, the nodes running the application under test and the FTP application are placed at the same distances, ranging from 1m to 30m. Table 6.8 lists all the parameters of the simulation runs.

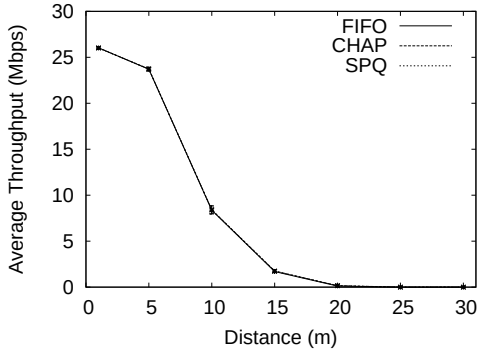
Figure 6.20 depicts how the G-model MOS of the game application and the throughput of the FTP application are affected by the distance of both nodes. The G-model MOS of the game application degrades up to 10m for FIFO while it stays relatively constant between 3.5 and 4.3 for CHAP and the G-model MOS for FIFO increases back to around 4 past 20m. Although the game application experiences higher packet loss rates at greater distances, the G-model assumes that the game application is capable of handling losses.

Table 6.8: Parameters for Both Node Distance Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS), VoIP, Video, Web
S_2, W_2	FTP
L_1	1 ms
L_2	1 ms
D_1	1,5,10,15,20,25,30m
D_2	1,5,10,15,20,25,30m
q	35 packets
Q	FIFO, CHAP, SPQ



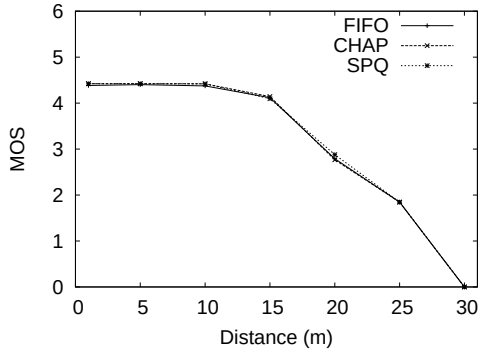
(a) G-model MOS



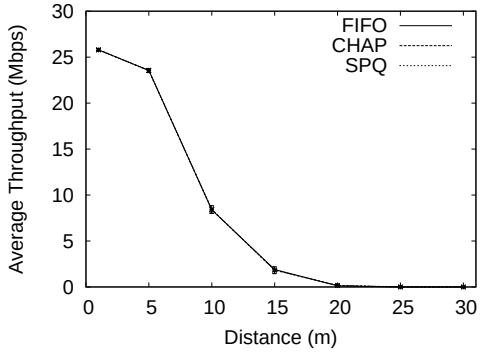
(b) FTP Throughput

Figure 6.20: Both Distance Case (Game)

The FTP throughput decreases as the FTP application node moves farther away from the AP and behaves the same for FIFO, CHAP, and SPQ.



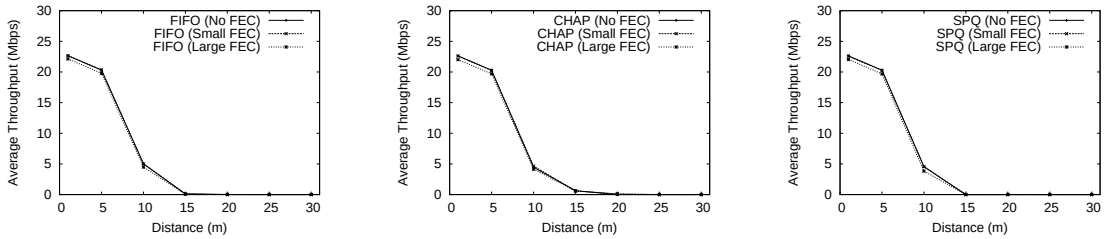
(a) E-model MOS



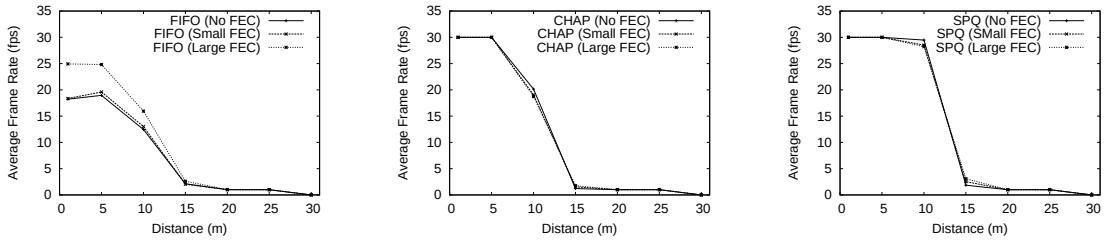
(b) FTP Throughput

Figure 6.21: Both Distance Case (VoIP)

Figure 6.21 depicts how the E-model MOS of the VoIP application and the throughput of the FTP application are affected by the distance of both nodes. The E-model MOS of the VoIP application stays constant at around 4.3 until 10m and starts to degrade, eventually reaching 0 at 30m, similar for FIFO, CHAP, and SPQ. Higher packet loss rates at higher distances cause the E-model MOS degradation over distance. The FTP throughput decreases as the FTP application node moves farther away from the AP and behaves the same for FIFO, CHAP, and SPQ.



(a) FTP Throughput (FIFO) (b) FTP Throughput (CHAP) (c) FTP Throughput (SPQ)

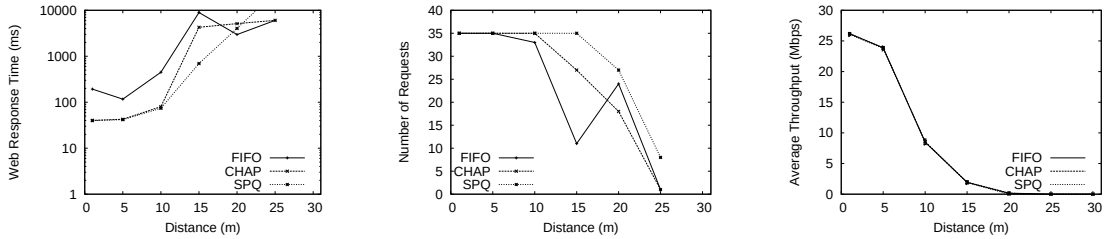


(d) Frames per Second (FIFO) (e) Frames per Second (CHAP) (f) Frames per Second (SPQ)

Figure 6.22: Both Distance Case (Video)

Figure 6.22 depicts how the FTP throughputs and the frame rate of the video application are affected by the distance of both nodes. The FTP throughput decreases as the FTP application node is further away from the AP. Frame rates decrease as the video application node is further away from the AP. Under FIFO, large FEC helps with losses when the video application is close to the AP. As the video application node is further away from the AP, the benefit of FEC decreases due to high loss rates. FEC does not help improve frame rates under CHAP and SPQ.

Figure 6.23 depicts how the Web response time of the Web application and the throughput of the FTP application are affected by the distance of both nodes. FTP throughputs



(a) Response Time

(b) Number of Completed Requests

(c) FTP Throughput

Figure 6.23: Both Distance Case (Web)

behave the same for FIFO, CHAP, and SPQ. The throughputs eventually reach 0 Mbps as the FTP application node is further away from the AP. The Web application is able to handle more requests under SPQ because SPQ prioritizes Web traffic regardless of the wireless condition. CHAP provides similar performance to SPQ until 10m. Because of differences in the number of completed requests under FIFO, CHAP, and SPQ, the average response time is calculated with only the completed requests that exist under all three scenarios.

6.8 Multiple Applications

This section describes a series of simulation runs with multiple activities taking place concurrently. This set of simulation results demonstrates how the concurrency affects the quality of applications under FIFO, CHAP, and SPQ.

For this scenario, SPQ is configured to treat network traffic in two classes: delay-sensitive and delay-insensitive applications. The game, VoIP, video and Web applications fall under the delay-sensitive application class, while the FTP application falls under the delay-insensitive application class. Separate simulations are run with three different FEC settings for the video application.

Table 6.10 shows the summary of application performance from the three simulation runs. The performance results for the game, VoIP, and Web applications are from the simulation run with no FEC. The G-model MOS for the game application with CHAP

Table 6.9: Parameters for Multiple Applications Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS)
S_2, W_2	VoIP
S_3, W_3	Video (No FEC, Small FEC, Large FEC)
S_4, W_4	Web
S_5, W_5	FTP
L_{1-5}	1 ms
D_{1-5}	1m
q	35 packets
Q	FIFO, CHAP, SPQ

Table 6.10: Summary of Performance Metrics

App. (Unit)	FIFO	CHAP	SPQ	% Impr
Game (MOS)	3.57	4.27	4.25	+20%
VoIP (MOS)	4.31	4.42	4.42	+3%
Video - No FEC (fps)	15.58	30.00	30.00	+93%
Video - Small FEC (fps)	17.44	30.00	30.00	+72%
Video - Large FEC (fps)	23.16	30.00	30.00	+30%
Web (ms)	113.39	56.07	41.85	+51%
FTP (Mbps)	22.95	22.91	22.91	0%

shows an improvement of 20% over FIFO from 3.57 to 4.27. The E-model MOS for the VoIP application with CHAP shows an improvement of 3% over FIFO from 4.31 to 4.42. The response time for the Web application with CHAP shows an improvement of 51% over FIFO from 113 ms to 56 ms. The frame rate of video streaming with CHAP increases to 30 frames per second regardless of the use of FEC. The improvement over FIFO is 93%, 72%, and 30% from 15.58, 17.44, and 23.16 frames per second with no FEC, small FEC, and large FEC, respectively. While CHAP provides such improvement for these delay-sensitive applications over FIFO, the FTP throughput remains unchanged with 22.95 Mbps and 22.91 Mbps for FIFO and CHAP respectively. Again, CHAP's performance for these delay-sensitive applications is close to SPQ's. The game, VoIP, and video applications perform the same under CHAP and SPQ. The response time for the Web application with

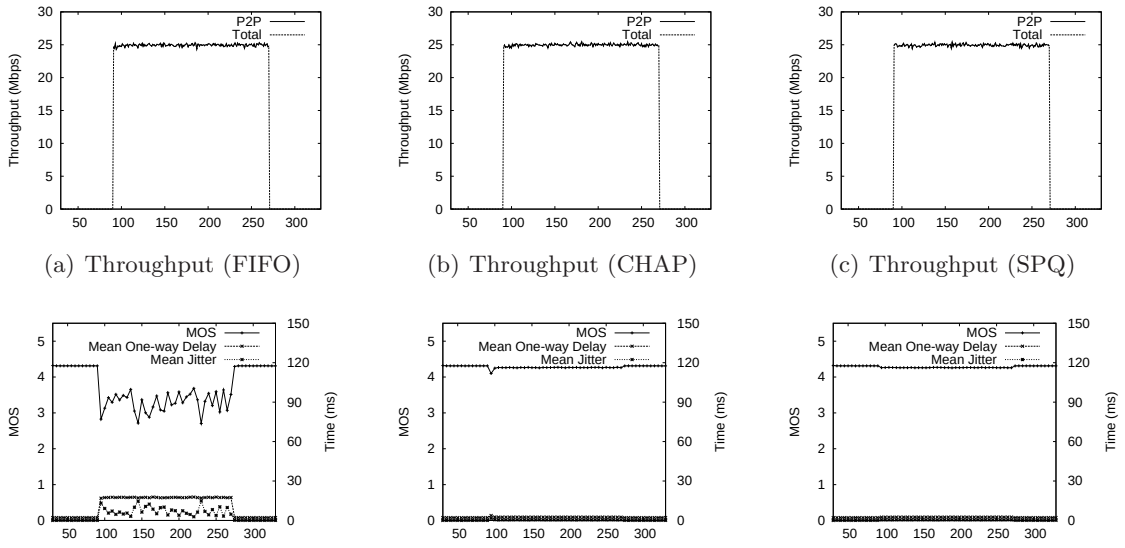
CHAP is not as low as with SPQ.

6.9 Peer-to-Peer Application

This section describes a series of simulation runs with a Peer-to-Peer (P2P) application as the background traffic. Simulation results show the effects on the quality of delay-sensitive applications in the presence of a P2P application, which uses multiple concurrent TCP flows to download files. Table 6.11 lists all the parameters of the simulation runs.

Table 6.11: Parameters for Peer-to-Peer Scenario

Parameter	Setting/Value
S_1, W_1	Game (FPS), VoIP, Video, Web
S_{2-11}, W_2	P2P
L_1	1 ms
L_2	1 ms
D_1	1m
D_2	1m
q	35 packets
Q	FIFO, CHAP, SPQ

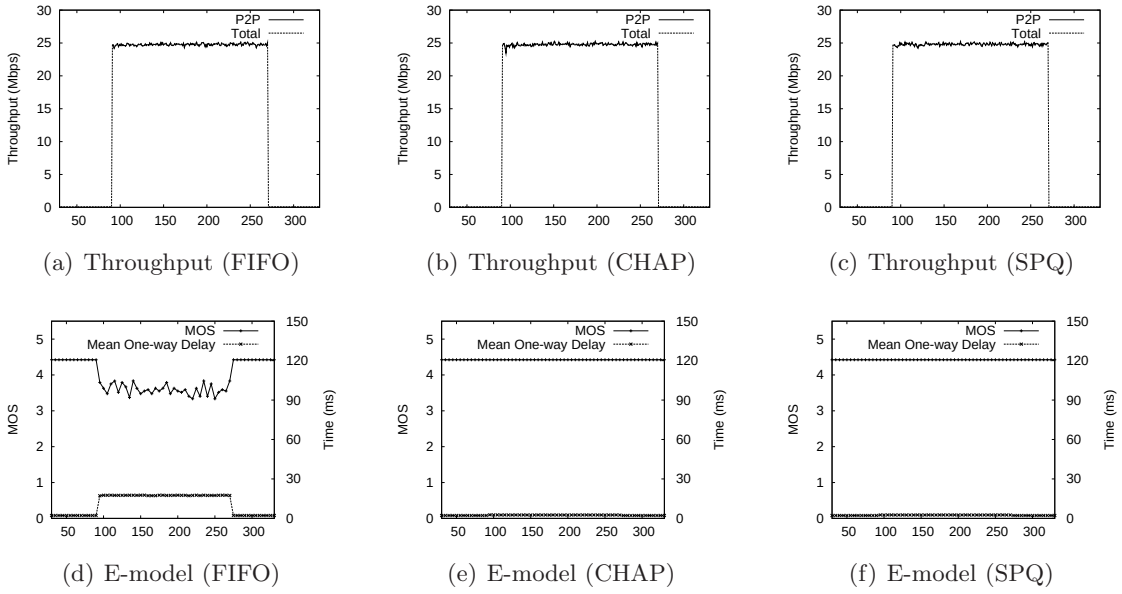


(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) G-model (FIFO) (e) G-model (CHAP) (f) G-model (SPQ)

Figure 6.24: P2P Throughput and Game Quality

Figure 6.24 depicts the G-model MOS of the game application and the P2P application throughput over time. The P2P application throughputs are the same for FIFO, CHAP, and SPQ. The game application throughput is excluded in the figure because the game application uses only 14.4 Kbps. The G-model MOS drops to around 3 when the P2P application is active in the background between 90 and 270 seconds for 3 minutes. The G-model MOS stays almost constant at around 4.3 for both CHAP and SPQ. CHAP is able to keep the G-model MOS up because the throughput of the game application is still less than an individual flow of the P2P application.

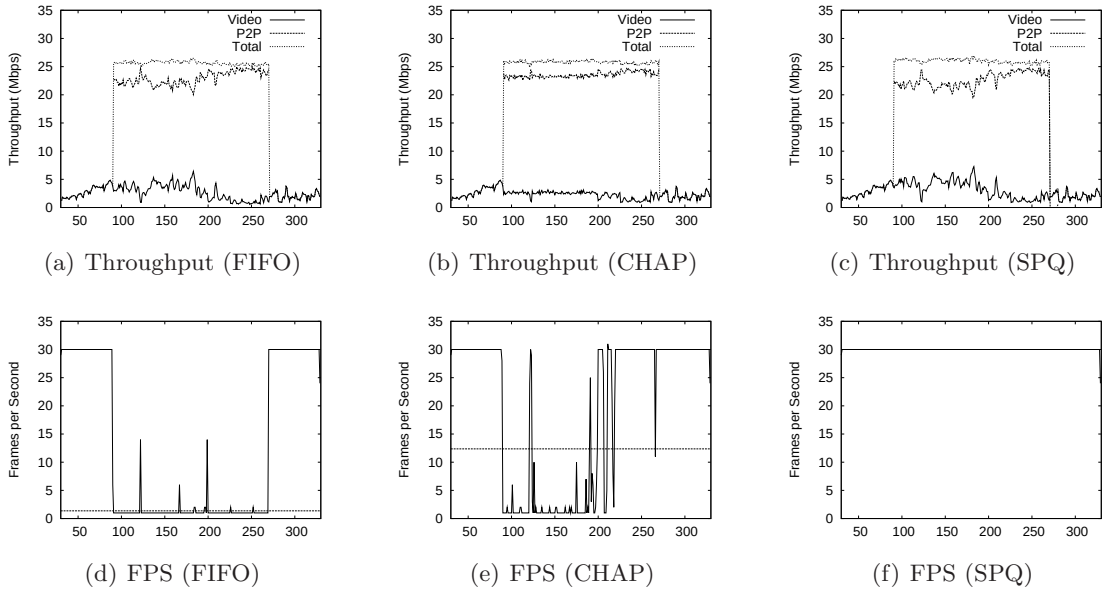


(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) E-model (FIFO) (e) E-model (CHAP) (f) E-model (SPQ)

Figure 6.25: P2P Throughput and VoIP Quality

Figure 6.25 depicts the E-model MOS of the VoIP application and the P2P application throughput over time. The P2P application throughputs are the same for FIFO, CHAP, and SPQ. The VoIP application throughput is excluded in the figure because the game application uses only 80 Kbps. The E-model MOS drops to around 3.5 when the P2P application is active in the background between 90 and 270 seconds for 3 minutes. The E-model MOS stays almost constant at around 4.3 for both CHAP and SPQ. CHAP is able to keep the E-model MOS up because the throughput of the VoIP application is still less than an individual flow of the P2P application.



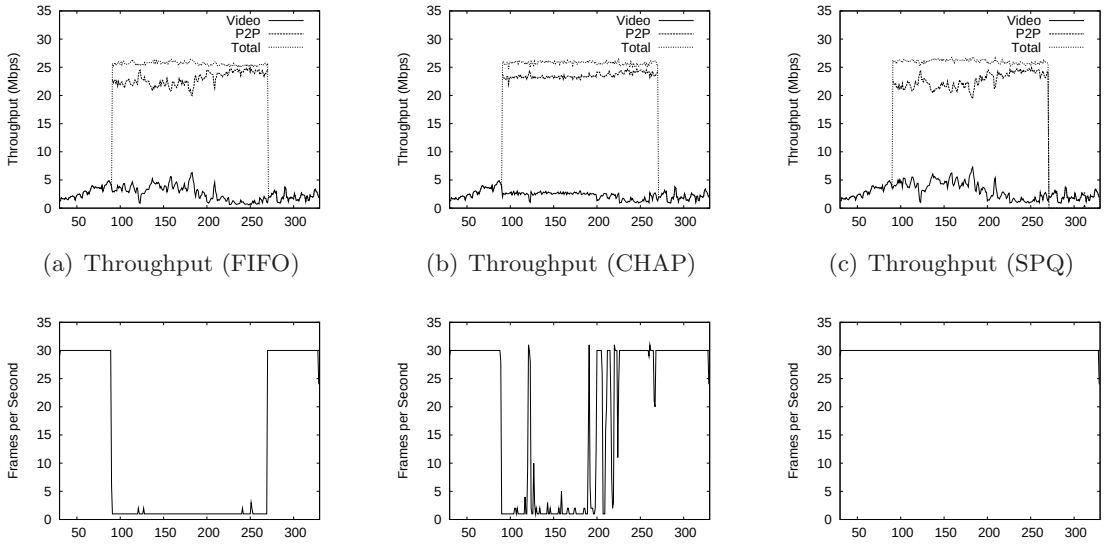
(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) FPS (FIFO) (e) FPS (CHAP) (f) FPS (SPQ)

Figure 6.26: Throughputs and Video Quality with No FEC

Figure 6.26 depicts the video frame rate of the video application without FEC and the throughputs of both applications over time. The total throughputs are the same for FIFO, CHAP, and SPQ, while the individual application throughputs are different. The video application throughput under SPQ is the encoded video throughput because SPQ prioritizes the video traffic regardless of the presence of the P2P application. The video application throughput under FIFO follows the same trend as the video application throughput under SPQ. The video application throughput under CHAP only follows the same trend when the input video throughput is less than 2.33 Mbps because 2.33 Mbps is the fair share of a flow when there are 11 flows present. Therefore, CHAP keeps the video application throughput at around 2.33 Mbps until 200 seconds while the input video throughput is mostly greater than 2.33 Mbps. However, the video frame rate does not follow the throughput. The video application under FIFO is able to play only about 1 frame per second in the presence of the P2P application while it can play 30 frames per second under SPQ. CHAP is able to play 12 frames per second on average in the presence of the P2P application. When the input video throughput is greater than the fair share of 2.33 Mbps, it also drops to around 1 frame per second like FIFO but provides higher

frame rate once it drops below 2.33 Mbps.

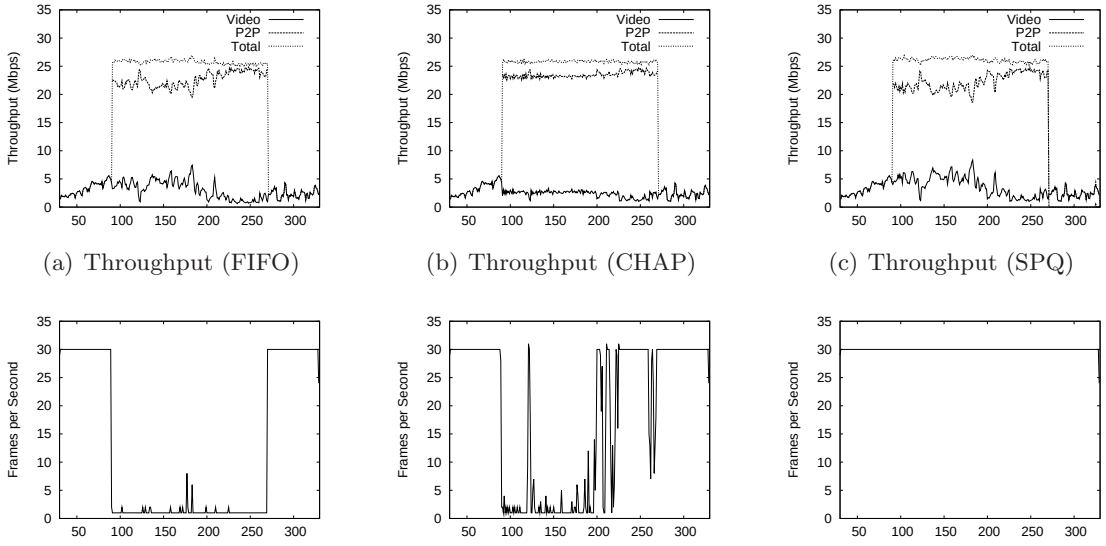


(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) FPS (FIFO) (e) FPS (CHAP) (f) FPS (SPQ)

Figure 6.27: Throughputs and Video Quality with Small FEC

Figure 6.27 depicts the video frame rate of the video application with Small FEC and the throughputs of both applications over time. The throughputs and frame rates look similar to those shown in Figure 6.26.

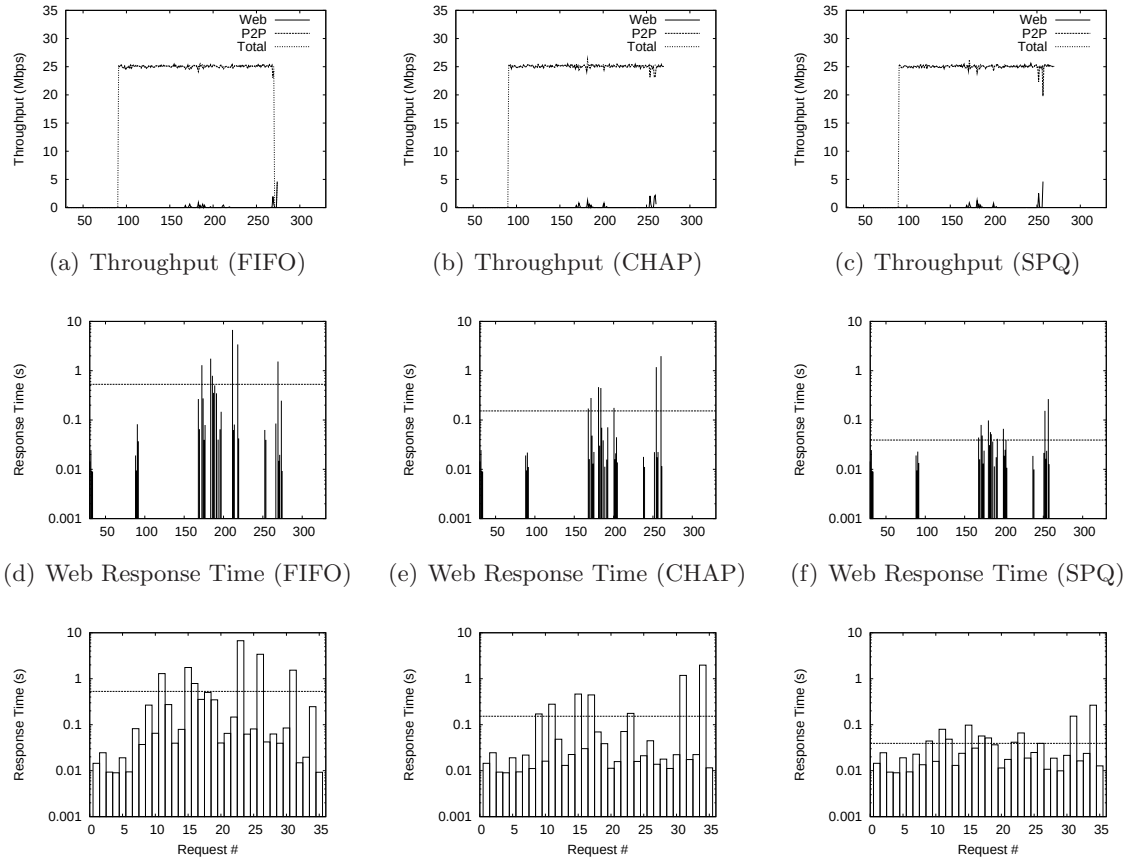


(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) FPS (FIFO) (e) FPS (CHAP) (f) FPS (SPQ)

Figure 6.28: Throughputs and Video Quality with Large FEC

Figure 6.28 depicts the video frame rate of the video application with Large FEC and the throughputs of both applications over time. The throughputs and frame rates look similar to those shown in Figure 6.26.



(a) Throughput (FIFO) (b) Throughput (CHAP) (c) Throughput (SPQ)

(d) Web Response Time (FIFO) (e) Web Response Time (CHAP) (f) Web Response Time (SPQ)

(g) Web Response Time (FIFO) (h) Web Response Time (CHAP) (i) Web Response Time (SPQ)

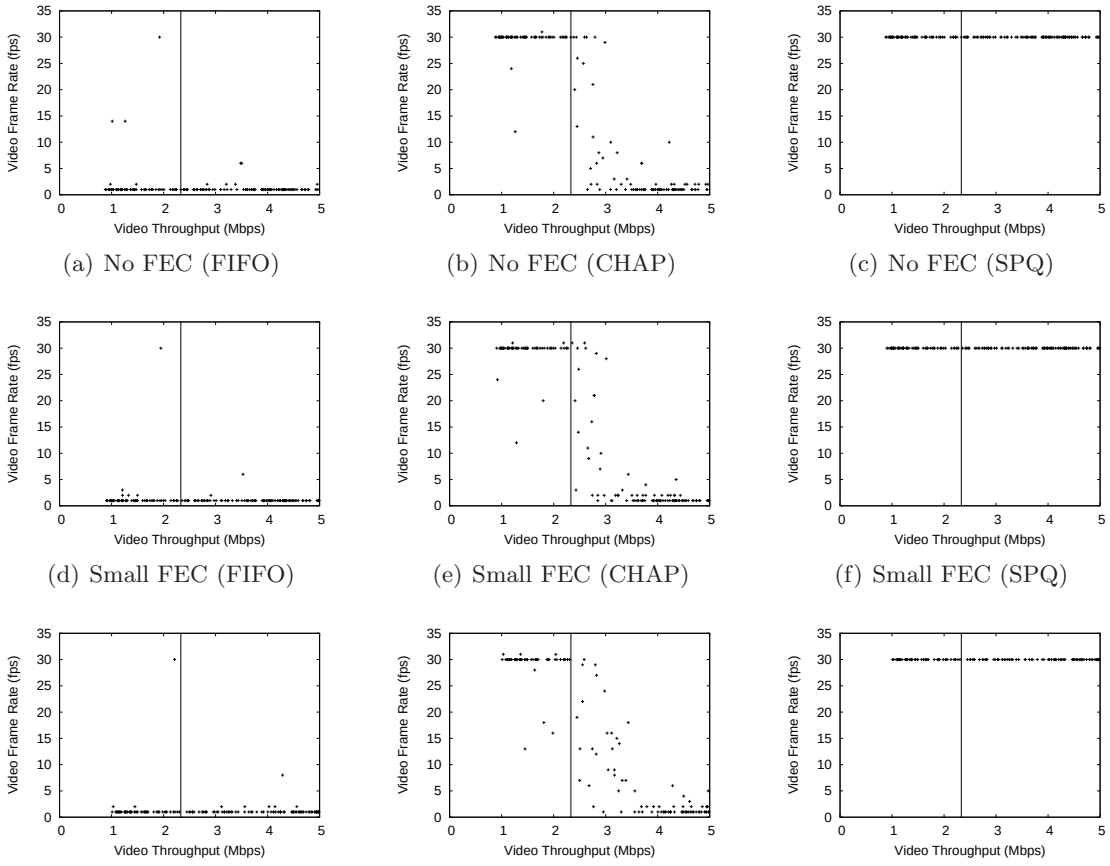
Figure 6.29: Throughputs and Web Quality

Figure 6.29 depicts the response time of the Web application with the P2P application in the background and the throughputs of both applications over time. The total throughputs are the same for FIFO, CHAP, and SPQ. SPQ provides the shortest average response time of 39 ms while FIFO provides the longest average response time of 530 ms. CHAP provides the average response time of 153 ms, closer to SPQ than FIFO. CHAP is able to provide benefit to Web application because Web flows are generally short-lived and tend to use little bandwidth. Figures 6.29 (d)-(f) show the response time of each Web request against the simulation time on the horizontal axis while Figures 6.29 (g)-(i) show the response time

of each Web request by its request number.

6.10 Edge Behavior

CHAP's performance improvement for delay-sensitive applications relies heavily on the relationship depicted in Figure 2.1. The simulation results from Section 6.9 demonstrate that the video application performance varies as the input video throughput varies across the fair share of the bandwidth. In this section, we explore such edge conditions of CHAP and show how it performs compared to FIFO and SPQ with different visualizations of the same simulation results from Section 6.9.



(a) No FEC (FIFO) (b) No FEC (CHAP) (c) No FEC (SPQ)

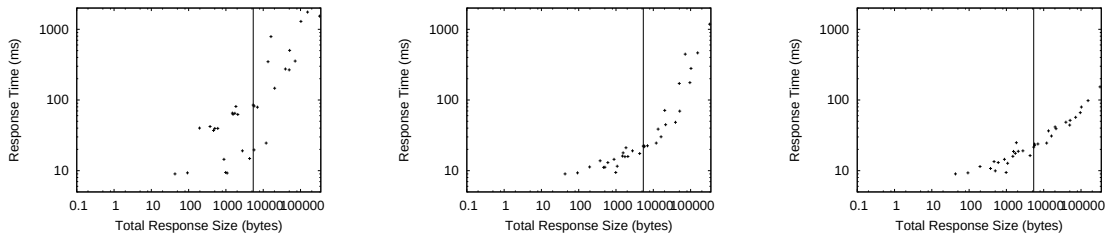
(d) Small FEC (FIFO) (e) Small FEC (CHAP) (f) Small FEC (SPQ)

(g) Large FEC (FIFO) (h) Large FEC (CHAP) (i) Large FEC (SPQ)

Figure 6.30: Edge Condition (Video)

Figure 6.30 depicts the scatter plots of video frame rates and corresponding video

throughputs with the system sampled every second in each simulation run. The vertical line represents 2.33 Mbps, the fair share of bandwidth among 11 flows. Regardless of the FEC scheme, the scatter plots for FIFO show that the video frame rate suffers and stays at around 1 frame per second. Similarly, SPQ provides for a full 30 frames per second regardless of the FEC scheme and video throughput. CHAP is able to provide close to 30 frames per second when the video throughput is below 2.33 Mbps, as expected. However, as the video throughput exceeds 2.33 Mbps, the video frame rate starts to degrade. The additional FEC in the video stream helps to mitigate the frame rate degradation as the video throughput crosses the edge.



(a) Web Response Time (FIFO) (b) Web Response Time (CHAP) (c) Web Response Time (SPQ)

Figure 6.31: Edge Condition (Web)

Figure 6.31 depicts the scatter plots of response times of Web requests and corresponding total response size of Web objects. The vertical lines show the median of the total response sizes. SPQ shows the shortest possible response times because SPQ prioritizes Web traffic regardless of the presence of the P2P traffic. FIFO offers the worst response time. CHAP is able to offer response times close to SPQ below the median total response size. Because CHAP allows a short burst determined by the parameter I and switches to round-robin for the remainder of the data, CHAP provides response times greater than SPQ but still less than FIFO as the total response size increases.

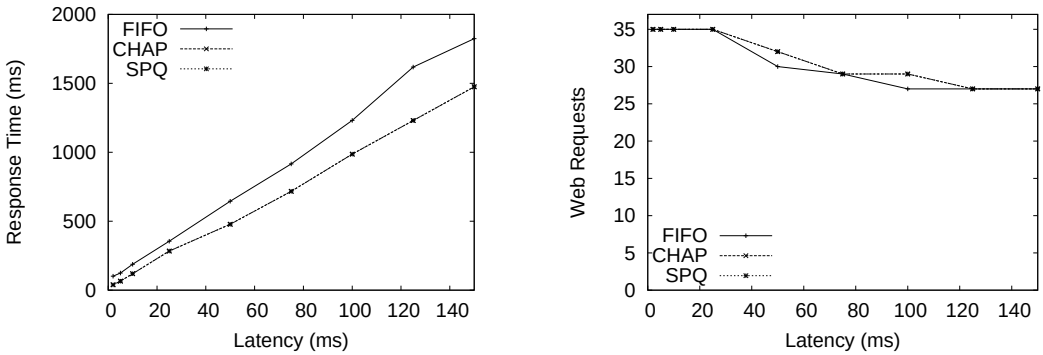
6.11 Latency and Web Application Performance

This section describes a series of simulation runs with varying network latencies in the presence of the FTP application in the background. End hosts involved in users' activities

are placed all around the world and their end-to-end latencies vary from one to another. Maier *et al.* show a wide range of round-trip times of flows from residential networks with modes of 13 ms, 100 ms, and 160 ms [73]. This set of simulation results demonstrates how different network latencies affect application quality under FIFO, CHAP, and SPQ.

Table 6.12: Parameters for Latency Scenario

Parameter	Setting/Value
S_1, W_1	Web
S_2, W_2	FTP
L_1	1, 4, 9, 24, 49, 74, 99, 124, 149 ms
D_1	1m
q	35 packets
Q	FIFO, CHAP, SPQ



(a) Web Response Time

(b) Number of Completed Requests

Figure 6.32: Web Performance over Different Latencies

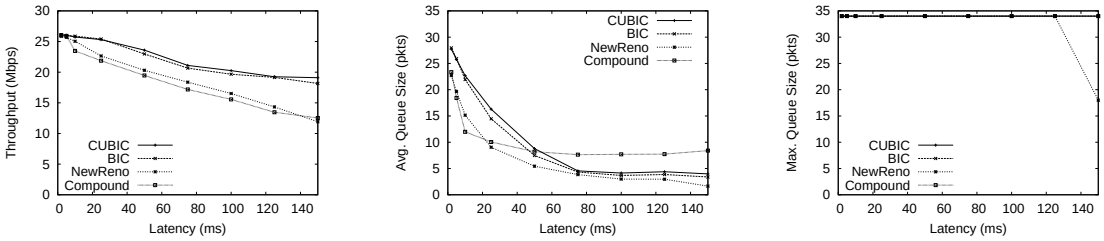
Figure 6.32 depicts the average response time of Web requests as the latency increases between the Web server and client. FIFO consistently provides longer response time compared to CHAP and SPQ. The difference in response time between FIFO and CHAP/SPQ increases as the latency increases. The total number of requests fulfilled decreases as the latency increases for FIFO, CHAP, and SPQ. As it takes longer to retrieve the Web pages, the user is able to browse fewer pages per given amount of time.

6.12 TCP Congestion Control and Performance

Modern operating systems use different TCP congestion control mechanisms. NewReno [15] is the latest modification to TCP's Fast Recovery Algorithm. Microsoft Windows supports TCP Compound [18] but it is not enabled by default. Windows 2000 follows RFC 2581 [136] for TCP congestion control[5]. Linux uses CUBIC [17], which is an extension of BIC [16]. This section shows how TCP NewReno, BIC, CUBIC, and Compound congestion control mechanisms behave differently in terms of queue sizes and throughputs with respect to latencies. Table 6.13 lists all the parameters of the simulation runs.

Table 6.13: Parameters for TCP Congestion Control and Latency Scenario

Parameter	Setting/Value
S_1, W_1	FTP
L_1	1, 4, 9, 24, 49, 74, 99, 124, 149 ms
D_1	1m
q	35 packets
Q	FIFO



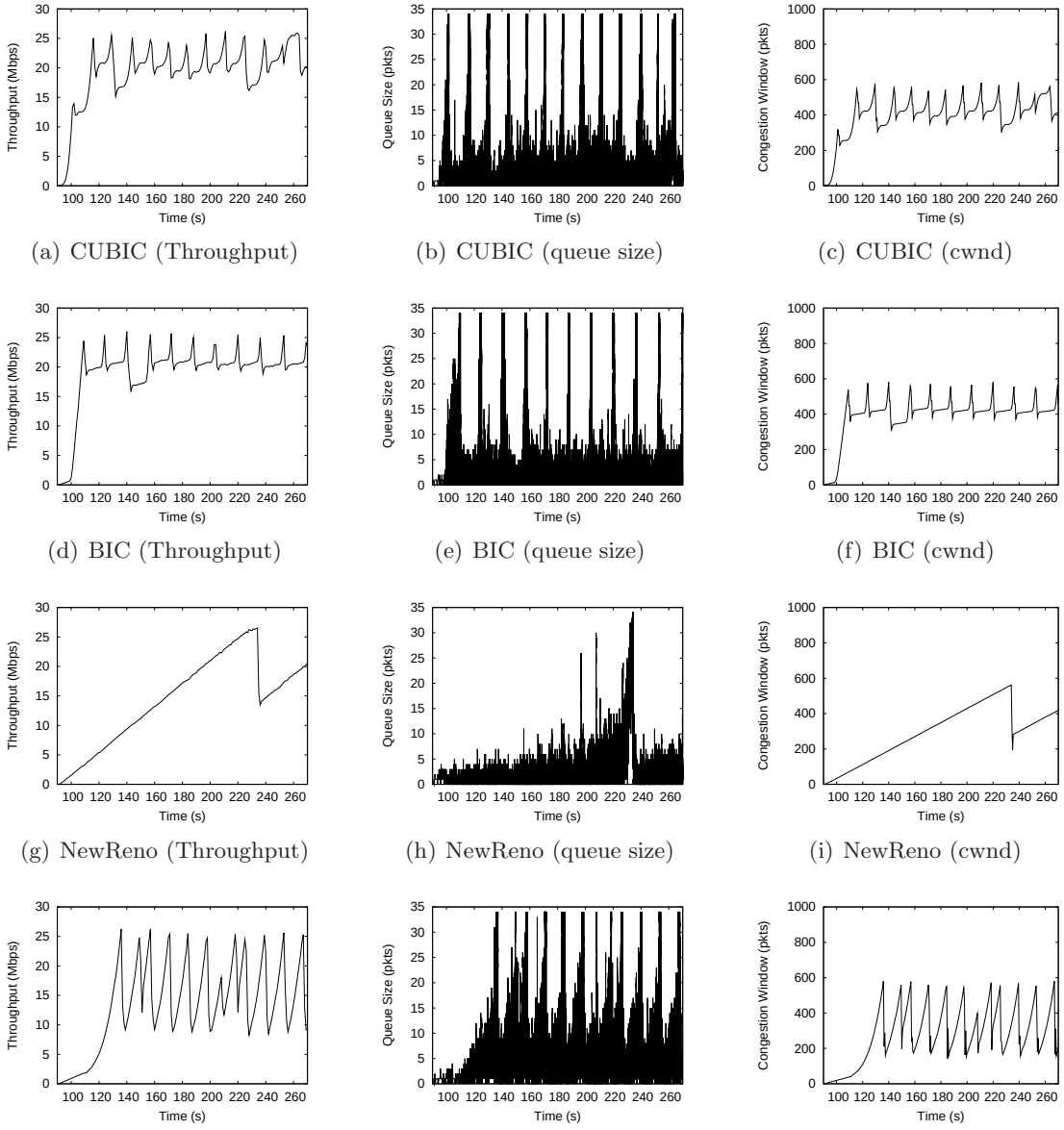
(a) Throughput (b) Average Queue Size (c) Maximum Queue Size

Figure 6.33: TCP Performance and Impact on Queue Size over Different Latencies

Figure 6.33 summarizes the throughput, average queue size and maximum queue size across NewReno, BIC, CUBIC, and Compound. BIC and CUBIC are more resilient against changes in latencies compared to NewReno and Compound. The FTP throughputs using BIC and CUBIC drop the throughput from 26.1 Mbps and 26.0 Mbps to 19.1 Mbps and 18.2 Mbps, respectively, as the latency increases from 2 ms to 150 ms. Increases in the

[5] http://technet.microsoft.com/en-us/library/bb726981.aspx

latency decrease the FTP throughputs using NewReno and Compound from 25.8 Mbps and 26.1 Mbps to 11.9 Mbps and 12.5 Mbps, respectively. Average queue sizes decrease for NewReno, BIC, CUBIC, and Compound as the latency increases. NewReno, BIC, CUBIC, and Compound are able to fill up the queue as the queue sizes reach the maximum. NewReno does not fill up the queue to the limit when the latency is 150 ms because it is not able to increase *cwnd* high enough to do so within the simulation duration.



(a) CUBIC (Throughput) (b) CUBIC (queue size) (c) CUBIC (cwnd)

(d) BIC (Throughput) (e) BIC (queue size) (f) BIC (cwnd)

(g) NewReno (Throughput) (h) NewReno (queue size) (i) NewReno (cwnd)

(j) Compound (Throughput) (k) Compound (queue size) (l) Compound (cwnd)

Figure 6.34: Queue Size over Time (L_1 = 125ms) (q = 35 packets)

Figure 6.34 depicts the throughput, instantaneous queue size, and *cwnd* of NewReno, BIC, CUBIC, and Compound for the latency of 125 ms and the queue size of 35 packets. The queue size is not large enough to stabilize the FTP throughput. The throughputs using NewReno, BIC, and CUBIC range from 15 Mbps to 25 Mbps, while the throughput using Compound ranges from 10 Mbps to 25 Mbps. The *cwnd* of each TCP variant corresponds to its FTP throughput. The instantaneous queue size remains below 10 packets most of the time for NewReno, BIC, CUBIC, and Compound.

Figure 6.35 depicts the throughput, instantaneous queue size, and *cwnd* of NewReno, BIC, CUBIC, and Compound for the latency of 125 ms and the queue size of 350 packets. In contrast to the queue size of 35 packets, the throughput stabilizes at around 27 Mbps once it is reached. The *cwnd* of each TCP variant also corresponds to its FTP throughput. The instantaneous queue size remains high for BIC and CUBIC at around 250 packets, while the instantaneous queue size remains below 50 packets for Compound.

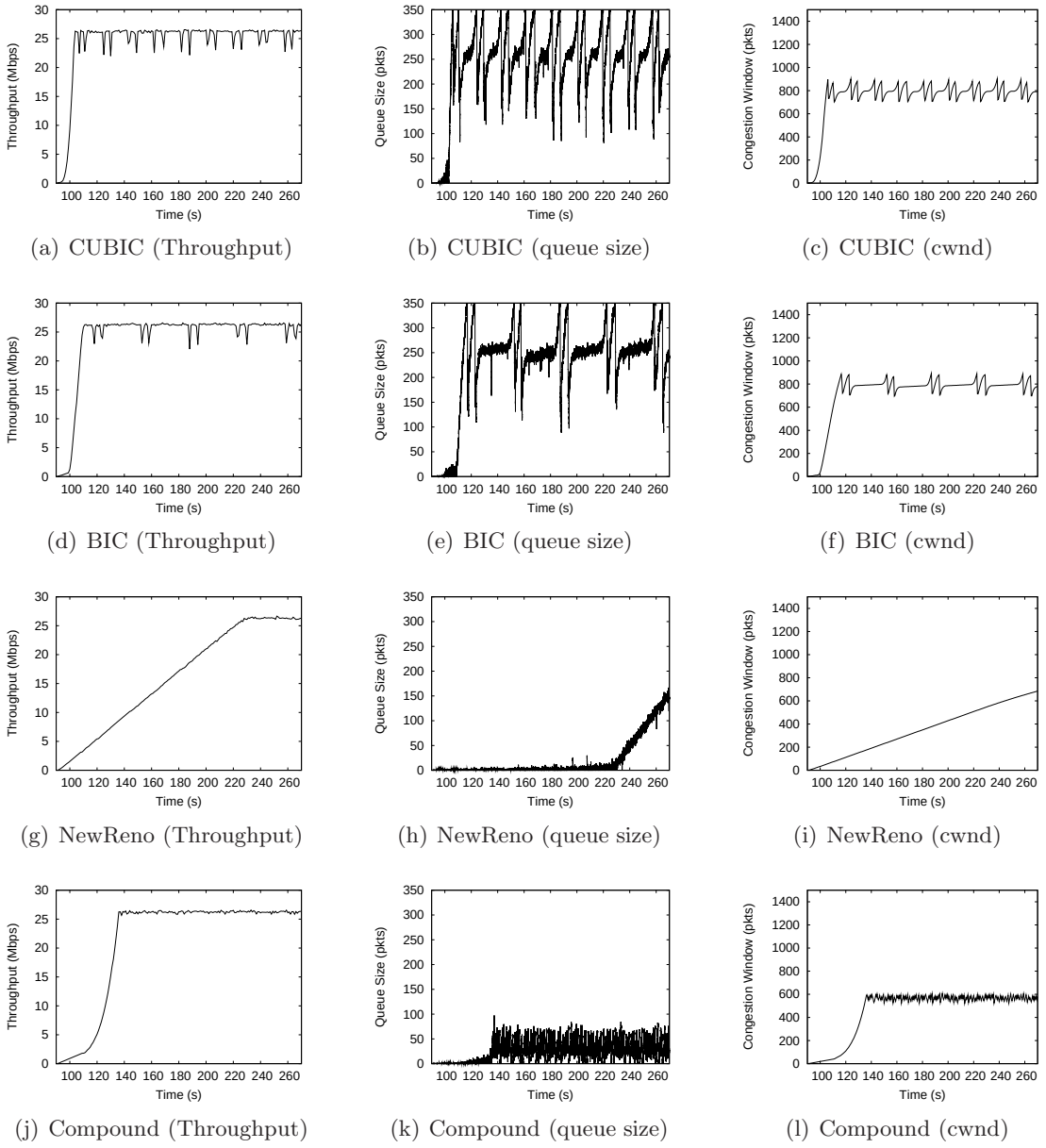
6.13 TCP Fairness

This section explores the TCP throughput fairness between FIFO and CHAP. Table 6.14 lists the simulation parameters for the latency scenario. SPQ is omitted from this scenario because the two FTP applications are in the same class and SPQ treats them exactly the same way as the FIFO queue does within a class.

Table 6.14: Parameters for Latency Scenario

Parameter	Setting/Value
S_1, W_1	FTP 1
S_2, W_2	FTP 2
L_1	1, 4, 9, 24, 49, 74, 99, 124, 149 ms
D_1	1m
q	35 packets
Q	FIFO, CHAP

Figure 6.36 depicts TCP throughput fairness and Jain's Fairness Index between FIFO



(a) CUBIC (Throughput) (b) CUBIC (queue size) (c) CUBIC (cwnd)

(d) BIC (Throughput) (e) BIC (queue size) (f) BIC (cwnd)

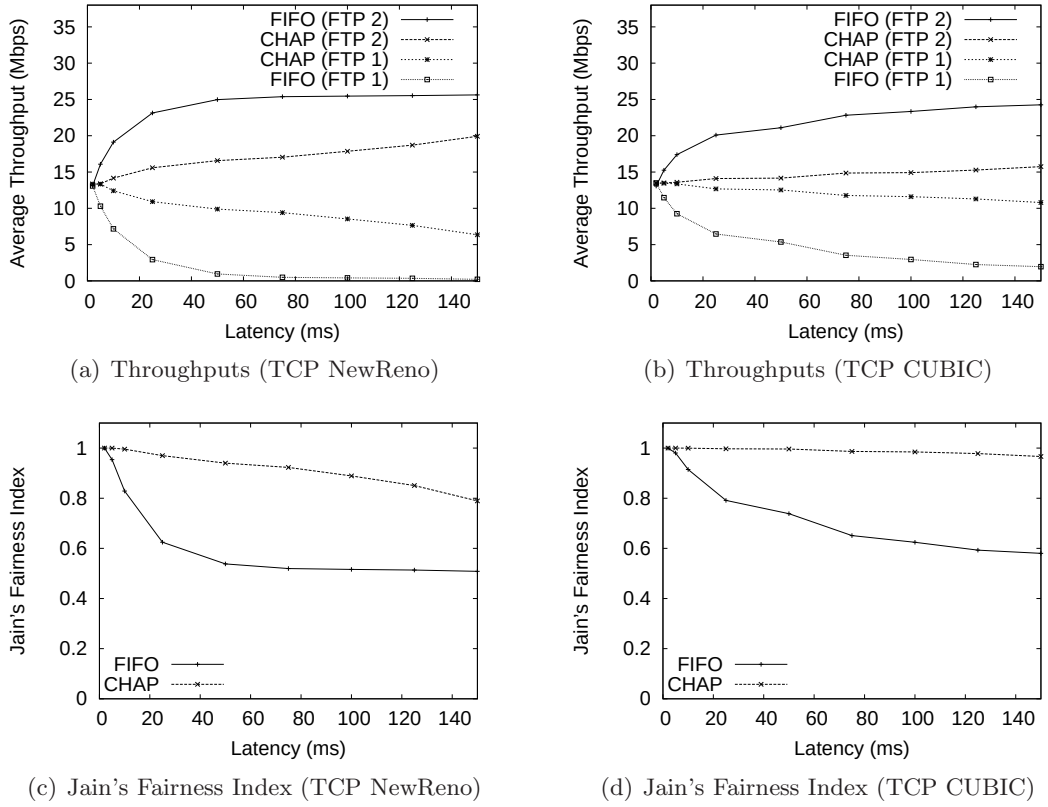
(g) NewReno (Throughput) (h) NewReno (queue size) (i) NewReno (cwnd)

(j) Compound (Throughput) (k) Compound (queue size) (l) Compound (cwnd)

Figure 6.35: Queue Size over Time (L_1 = 125ms) (q = 350 packets)

and CHAP using TCP NewReno and CUBIC while varying the latency of one FTP application node (FTP 1) and keeping the latency of the other FTP application node (FTP 2) constant. Jain's Fairness Index is calculated using the following equation:

$$fairness = \frac{(\sum x_i)^2}{n \times \sum x_i^2} \tag{6.4}$$



(a) Throughputs (TCP NewReno)

(b) Throughputs (TCP CUBIC)

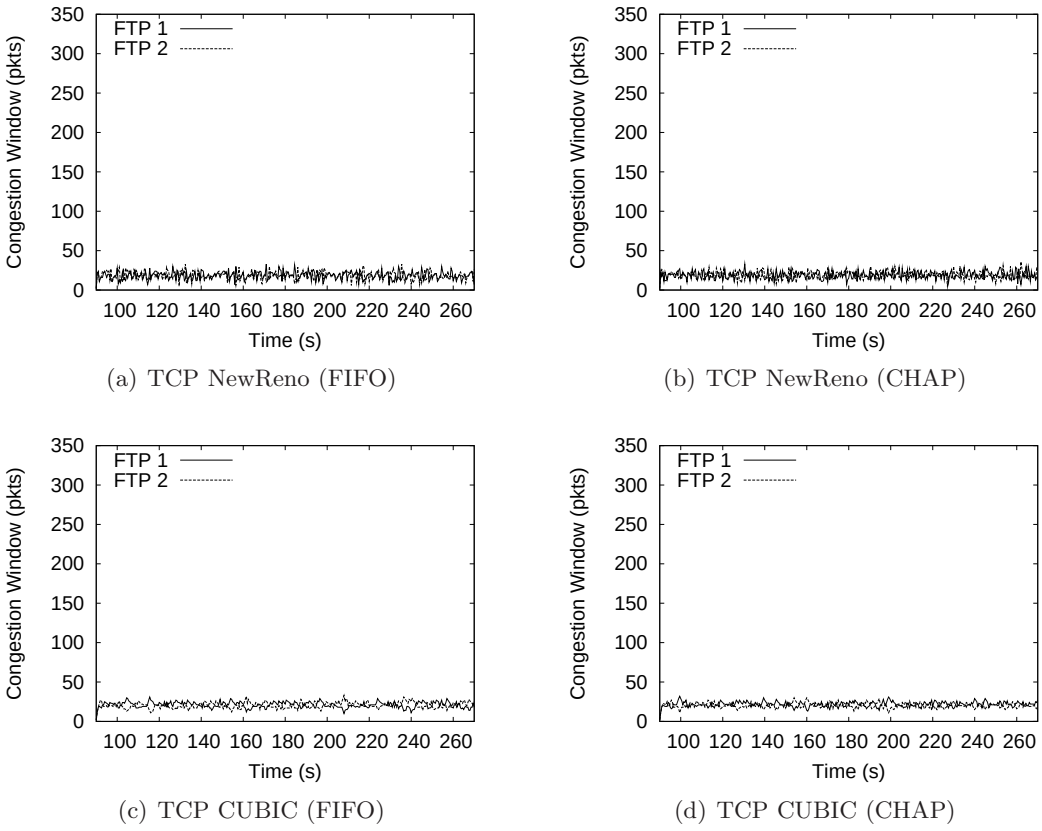
(c) Jain's Fairness Index (TCP NewReno)

(d) Jain's Fairness Index (TCP CUBIC)

Figure 6.36: TCP Fairness over Distances

The closer Jain's Fairness Index is to 1, the fairer are the FTP throughputs. When both FTP application nodes are at the same latency from the servers, they achieve about the same average throughput for both NewReno and CUBIC. Jain's Fairness Index for NewReno and CUBIC under FIFO and CHAP at 2 ms latency for both FTP applications are both 1.000. However, as the latency increases for one of the FTP application nodes (FTP 1), its throughput under FIFO decreases quickly to 0 for NewReno while the other FTP application takes the remaining bandwidth. Jain's Fairness Index decreases from 1.000 at 2 ms to 0.509 at 150 ms. CHAP is able to mitigate the unfairness for NewReno while providing similar total throughput of the system. Jain's Fairness Index decreases from 1.000 at 2 ms to 0.789 at 150 ms under CHAP. CUBIC provides better fairness than NewReno under both FIFO and CHAP. Jain's Fairness Index of FTP throughputs with Cubic under FIFO at 150 ms is 0.580. Jain's Fairness Index decreases only to 0.966 under

CHAP. CHAP is able to preserve the fairness feature of credit-based algorithms.



(a) TCP NewReno (FIFO)

(b) TCP NewReno (CHAP)

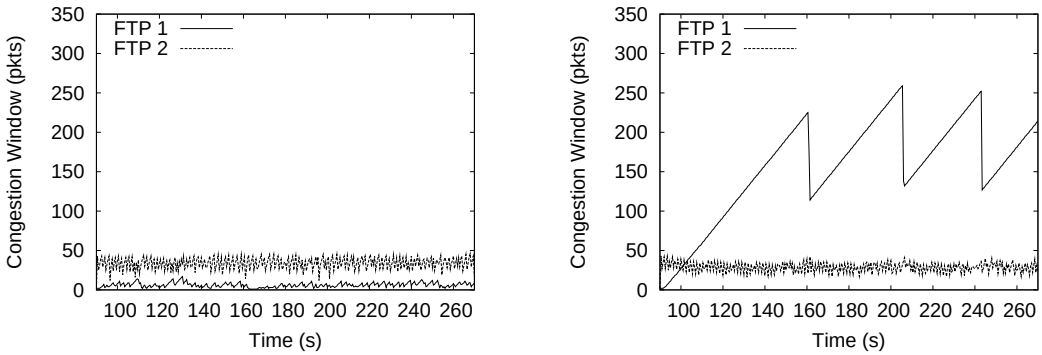
(c) TCP CUBIC (FIFO)

(d) TCP CUBIC (CHAP)

Figure 6.37: Congestion Window ($L_1 = 2$ ms)

Figure 6.37 depicts the *cwnd* of TCP used by both FTP applications when the latencies between the FTP server and client for both applications are 2 ms. Both NewReno and CUBIC have the same range of *cwnd* under FIFO and CHAP.

Figure 6.38 depicts the *cwnd* of TCP used by both FTP applications when the latency between the FTP server and client for FTP 1 is increased to 150 ms. Under FIFO, FTP 1's NewReno is unable to advance its *cwnd* for more throughput due to its increased latency. The use of CUBIC for FTP 1 under FIFO helps to advance its *cwnd* but introduces wider variation compared to FTP 2. CHAP helps FTP 1 to advance its window in the case of both NewReno and Cubic, improving fairness between the two FTP application throughputs.



(a) TCP NewReno (FIFO)

(b) TCP NewReno (CHAP)

(c) TCP CUBIC (FIFO)

(d) TCP CUBIC (CHAP)

Figure 6.38: Congestion Window (L_1 = 150 ms)

6.14 Summary

This chapter provides the validation and performance evaluation of CHAP using a series of NS2 simulations.

Section 6.2 describes an experiment with hardware conducted to validate NS2 simulations. Although the NS2 simulation results do not align with the experimental results exactly, they demonstrate similar behaviors such as the degradation in the G-model MOS in the presence of a concurrent FTP download. Section 6.3 demonstrates the validation of the model derived in Section 5.2. The average queueing delays observed by the analytical model and simulation results show similar trends such as the increasing queueing delay with the increasing rate of the low-bandwidth flow and the same queueing delay when two flows have the same rate.

Section 6.6 describes the performance evaluation of CHAP with varying queue sizes. The benefits of CHAP to applications whose quality is affected by delay and jitter, such as online games and Web browsing, increases as the queue size increases.

Section 6.7 discusses the performance evaluation of CHAP with varying distances of wireless nodes. CHAP provides higher application quality of the delay-sensitive applications while providing similar throughput for the delay-insensitive application in the background. Section 6.7.1 describes simulation scenarios where the node running the delay-sensitive application is further away from the AP. In the case of the unresponsive video streaming application, CHAP is able to mitigate the degradation of the FTP throughput compared to FIFO and SPQ as the video client is further away from the AP. Section 6.7.2 describes simulation scenarios where the node running the delay-insensitive application moves further away from the AP. The delay-insensitive application degrades the quality of the delay-sensitive applications under FIFO while it is in range of the AP to achieve a reasonable throughput. CHAP is able to provide higher quality consistently regardless of the distance of the delay-insensitive application. Section 6.7.3 describes simulation scenarios where both nodes are further away from the AP. The quality of the delay-sensitive and delay-insensitive applications degrade as they are further away from the AP. CHAP provides higher or similar application quality compared to FIFO at all distances.

Section 6.8 shows the benefits of CHAP while running multiple applications concurrently. CHAP provides higher application quality for all the delay-insensitive applications while maintaining the FTP throughput similar to FIFO and SPQ.

Section 6.9 demonstrates CHAP behavior with a P2P application. CHAP's behavior is similar to other results except for the video application. Because the game, VoIP and Web applications receive less than the fair share of bandwidth, CHAP prioritizes these applications over the individual flows of the P2P application. The range of throughputs of the video application overlaps with the fair share of bandwidth. Therefore, P2P flows are sometimes prioritized over the video application. Such edge conditions are explored in more detail in Section 6.10. As the throughput of the delay-sensitive application crosses

the fair share of bandwidth in the network, CHAP's benefits declines gracefully.

Section 6.11 explores different latencies for Web browsing. CHAP benefits the quality of Web browsing as the network latency increases between the Web server and client.

Section 6.12 examines different TCP variants as there is a variety of TCP congestion control mechanisms deployed in modern operating systems. BIC, CUBIC, NewReno and Compound demonstrate different behaviors with respect to the AP queue size and their congestion window and throughput. Section 6.13 explores the fairness of TCP NewReno and CUBIC under FIFO and CHAP. Inheriting the fairness feature of credit-based mechanisms, CHAP provides much fairer throughput to two FTP flows using NewReno and CUBIC.

Chapter 7

Implementation

This chapter discusses the implementation of CHAP and subsequent performance evaluation of the implementation. Section 7.1 describes a Linux Queue Discipline (Qdisc) implementation of CHAP. Section 7.2 describes a set of controlled experiments to validate the implementation. Section 7.3 describes a case study conducted at home to evaluate the performance of network applications using the CHAP implementation.

7.1 Linux Qdisc

Linux allows users to connect multiple network interfaces to create a bridge. Each network interface can be associated with a Qdisc using the 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 Stochastic Fair Queue (SFQ) Qdisc as the base. CHAP hashes each incoming packet based on flow information determined by the source address, destination address, and protocol type, and keeps track of their corresponding credits.

Because of difficulties with the existing Linux code base[1] for the host AP and madWifi driver in an IEEE 802.11 infrastructure network, we emulated IEEE 802.11g frame transmission over a 100 Mbps Ethernet connection. The FIFO and CHAP Qdiscs are developed

[1]Kernel panics and frequent disassociation of wireless hosts.

using a kernel timer interrupt to schedule packet departures. The IEEE 802.11 delay of each packet is calculated by:

$$d = DIFS + \frac{p_{data} + p_{header}}{R_{data}} + SIFS + \frac{p_{ack}}{R_{basic}} \tag{7.1}$$

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_{trans} is 177 μs when p_{data} is 0 bytes and a p_{size} of 1500 bytes results in a d_{trans} of 399 μs.

Because IEEE 802.11g is emulated over a wired connection, frame error rate is implemented for every wireless frame transmission. It uses an Independent and Identically-Distributed (IID) random variable to introduce errors. Therefore, the probability of a successful transmission in n transmissions can be calculated with the frame error rate of p.

$$p_s(n) = (p + (1-p)p)^{n-1}(1-p)^2 \tag{7.2}$$

The maximum number of retransmissions is set to 4 to allow 5 transmission attempts total. Figure 7.1 depicts how the success probabilities change with respect to different frame error rates. The 6th transmission attempt in the figure refers to losses.

Each unsuccessful transmission results in a random exponential back-off, which chooses a random number of slots to wait before attempting to transmit again. IEEE 802.11 has minimum and maximum contention windows (W_{min} and W_{max}, respectively) which are set to 32 and 1024. Each unsuccessful transmission doubles the contention window size while restricting it to W_{max} at the largest. The back-off after n-th unsuccessful transmission is calculated by Equation 7.3.

$$b(n) = \begin{cases} random(W_{min} \times 2^{n-1}) \times slot, & W_{min} \times 2^{n-1} \leq W_{max} \\ random(W_{max}) \times slot, & W_{min} \times 2^{n-1} > W_{max} \end{cases} \tag{7.3}$$

Therefore, the total *cost* of frames transmitted successfully at n-th transmission can be

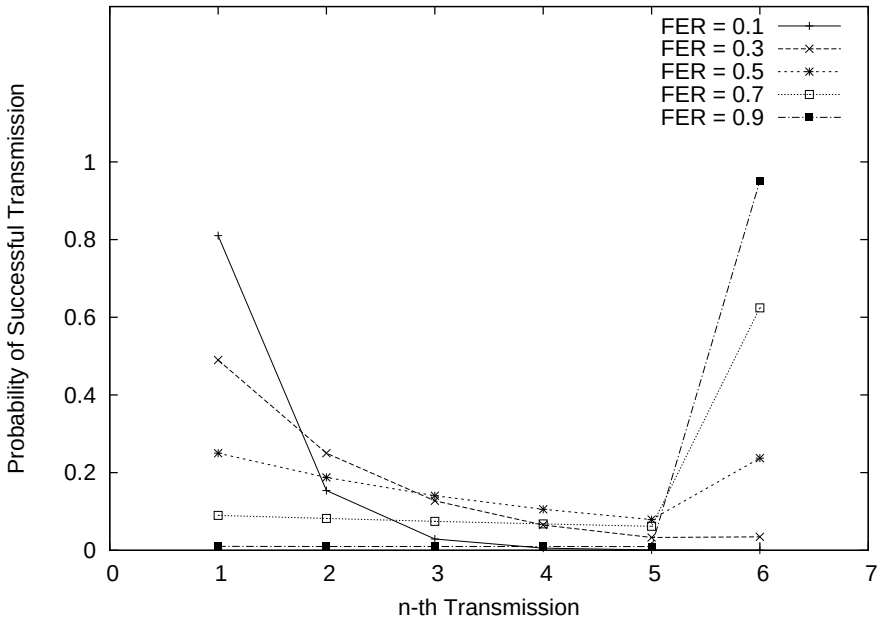


Figure 7.1: Probability of Successful Transmissions

calculated by Equation 7.4.

$$cost(n) = n \times d + \sum_{i=1}^{n-1} b(i) \tag{7.4}$$

7.2 Validation

This section describes the method for validating the CHAP implementation, along with the results. This validation seeks to ascertain the accuracy of the implementation and also to compare the distances in the simulation and general frame error rates in the implementation.

7.2.1 Validation Setup

Figure 7.2 depicts the experimental setup for the validation of the Linux implementation of CHAP. All the devices in the setup are connected over 100 Mbps Ethernet connections. The Linux bridge running the Linux implementation of CHAP is a IBM ThinkPad T60 with a Core2Duo processor and 2 GB of RAM while the four computers running the game and

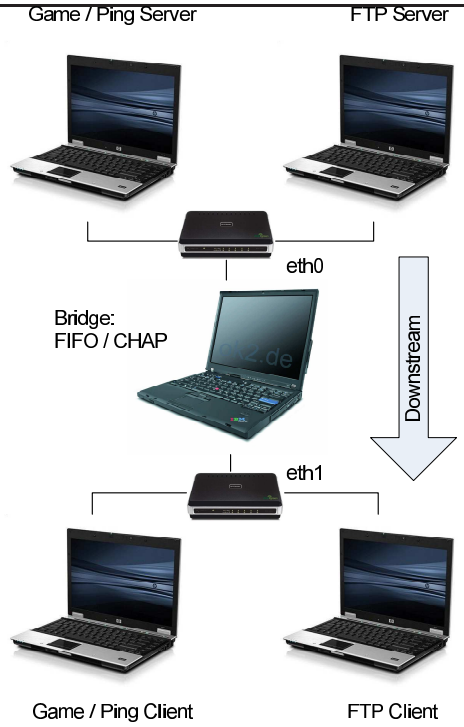


Figure 7.2: Controlled Experiment Topology

FTP applications are HP Elitebook 6930p's with a Core2Duo processor and 2 GB of RAM each. The left pair of laptops are running the game and ping applications on Windows XP SP3 while the right pair of laptops are running the FTP application on Debian Linux (Lenny). The Linux bridge is set to use either FIFO or CHAP Qdisc for the *eth1* interface with the queue size of 350 packets on Debian Linux. Each scenario tests the FIFO and CHAP queues. The CHAP queue uses 25 ms for the parameter I. The game application starts first and the FTP application starts 60 seconds later. The FTP application stops after 3 minutes while the game application continues for another minute before stopping.

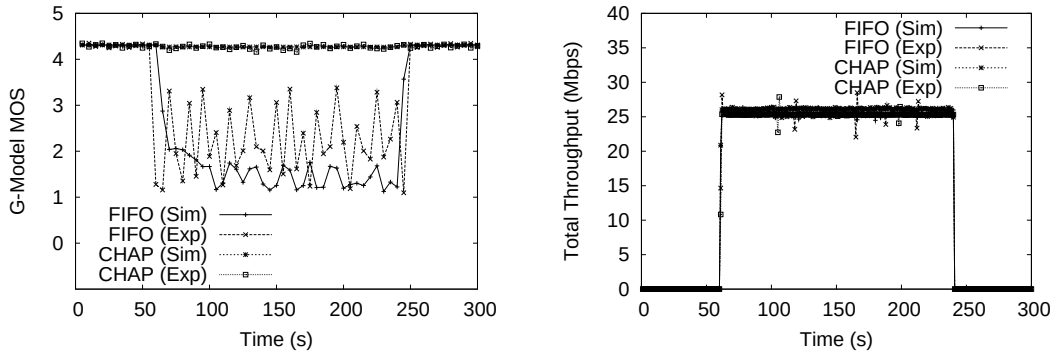
The game application is simulated using *MGEN*[2] to send UDP packets at a fixed interval following the average packet size and interval from the Halo 2 traffic model [22]. The server and client send 72-byte and 44-byte packets, respectively, every 40 ms. The time differences in the machines makes it difficult to measure one-way delays. *UDP Ping*[3]

[2]http://cs.itd.nrl.navy.mil/work/mgen/
[3]http://perform.wpi.edu/tools/

is used to measure the round-trip delay from the client to the server and later halved as an estimated measure of one-way delay. The FTP application is written as a simple TCP server and client to produce per-packet logs for analysis. Since the FTP application is run on Linux, the TCP congestion control mechanism used is CUBIC. Most of the network traffic is downstream from the servers to the clients and the only upstream traffic consists of TCP ACKs of the FTP application and the game client UDP packets.

7.2.2 Without Frame Errors

This section describes an experiment without induced frame errors to validate the quality of applications against simulation results. Figure 7.3 depicts the G-model MOS and total throughputs of both simulation and experiment results. The G-model MOS for FIFO in the experiment is slightly lower than the G-model MOS for FIFO in the simulation. In addition, the G-model MOS in the experiment is almost cyclical. Depending on the oscillation of the queue size, the G-model's jitter parameter is small or large in each 5 second interval, causing the G-model MOS to jump up and down. The G-model MOS's for CHAP in both simulation and experiment look identical, staying at around 4.2 for both. Total throughputs in both simulation and experiment look the same for FIFO and CHAP.

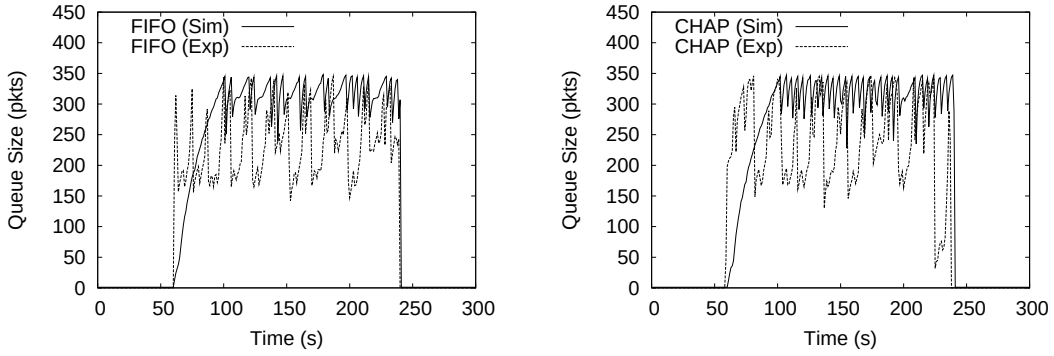


(a) G-model MOS

(b) Total Throughput

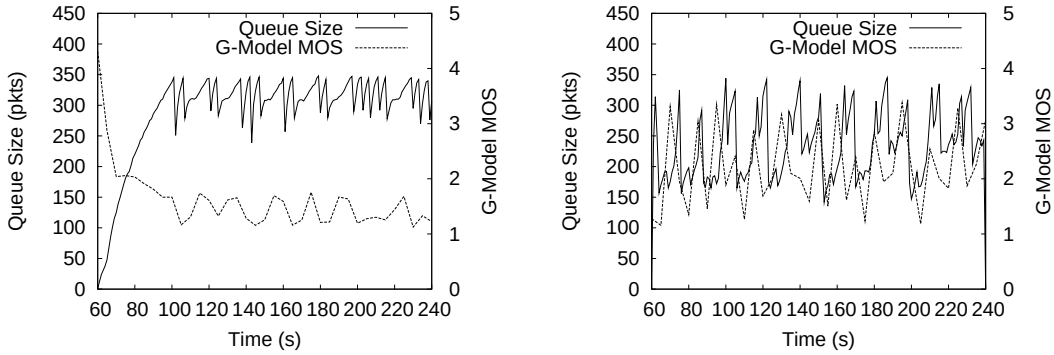
Figure 7.3: Controlled Experiment (Validation)

Figure 7.4 depicts the queue size sampled every second in both simulation and experiment for FIFO and CHAP. Solid lines in both graphs show the queue size in simulation.



(a) FIFO (b) CHAP

Figure 7.4: Controlled Experiment (Validation) - Queue Size

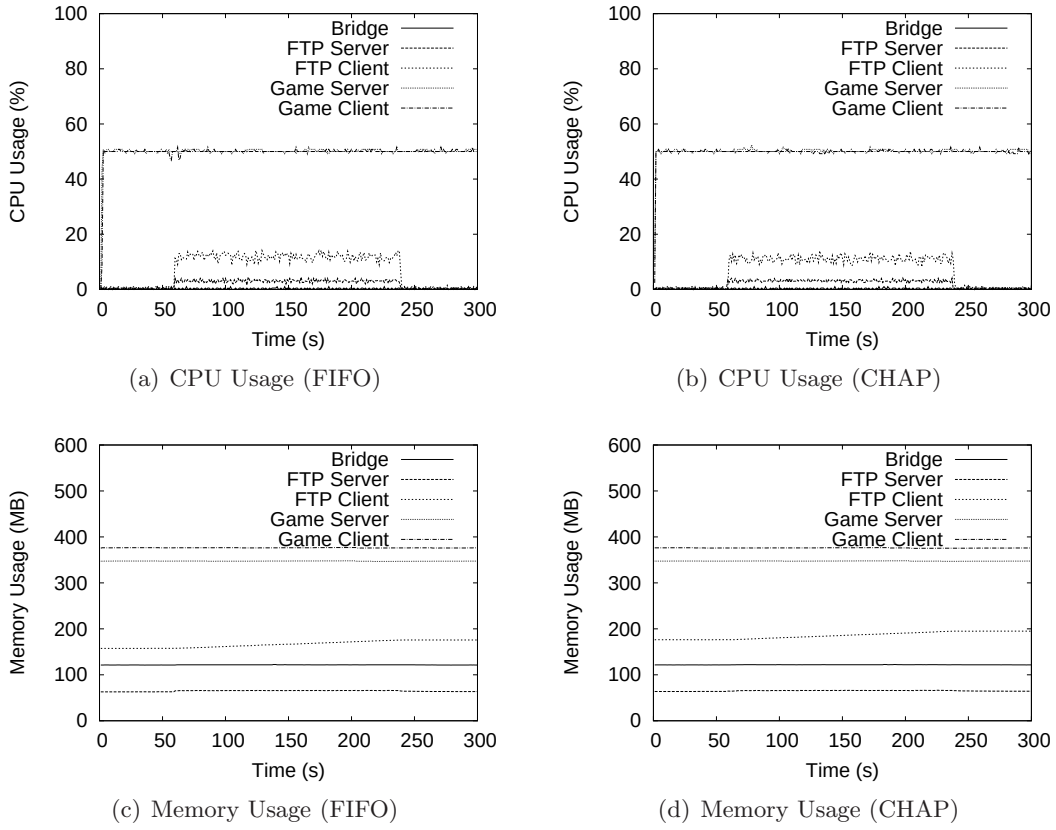


(a) Simulation (b) Experiment

Figure 7.5: Controlled Experiment (Validation) - G-model and Queue Size (FIFO)

In the simulation, it takes longer to reach the queue limit of 350 packets than in the experiment but it stays above 250 packets consistently. The queue size in the experiment reaches over 300 packets quickly but it has a wider range of variation between 150 and 350 packets. Figure 7.5 depicts the correlation between the variation in the queue size and G-model MOS. The G-model MOS relies on one-way delay and jitter and the variation in the queue size over time affects the jitter. Simulation results show that the G-model MOS is low due to high queue size, which contributes to longer one-way delay. The variation in the queue size is almost consistently between 250 and 350 packets. The G-model MOS does not vary much as the delay and jitter are relatively constant. The experiment result shows lower queue size and higher variation in queue size. Lower queue size and low variation in queue size in some periods contribute to high G-model MOS. When there is high variation

in queue size, G-model MOS drops to the 1-2 range.



(a) CPU Usage (FIFO)

(b) CPU Usage (CHAP)

(c) Memory Usage (FIFO)

(d) Memory Usage (CHAP)

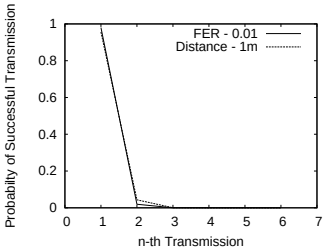
Figure 7.6: Controlled Experiment (Validation) - CPU and Memory Usage

Figure 7.6 depicts the CPU and memory usage of all the systems in the experiment over time. CPU and memory usage look about the same for both FIFO and CHAP at the Linux bridge. It suggests that CHAP does not add much more overhead than FIFO. Both the game server and client use 50% of the CPU when transmitting and receiving packets because MGEN uses busy waiting. The CPU usage is higher for the FTP server than the FTP client at around 12-13% because it uses a busy waiting while loop to send packets as fast as it can.

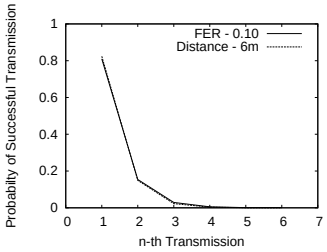
7.2.3 With Frame Errors

As explained in Section 7.1, emulated frame error rates can be forced at the Linux bridge. Frame error rates are induced for the FTP traffic to emulate the FTP client machine

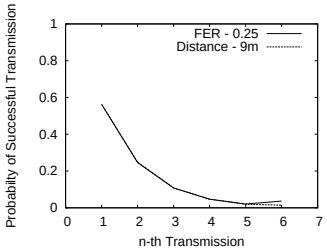
moving further away from the AP. Validation of such frame error rates requires matching the frame error rates and simulated distances in NS2. Frame error rates of 0.01, 0.10, 0.25, 0.50 and 0.75 are run in experiments. Figure 7.7 depicts the IID probability curve generated following Equation 7.2 along with the measured fraction of successful frame transmissions in NS2 simulations. Since both the simulation and experiment are set to try up to 4 retransmissions (total of 5 transmission attempts) the 6th transmission in the figure refers to losses of frames. Simulations were run with distances ranging from 1m up to 20m every 1m. The closest matches are found visually and shown in Figure 7.7. Frame error rate of 0.01, 0.10, 0.25, 0.50, and 0.75 match the simulated distances of 1m, 6m, 9m, 13m and 19m, respectively.



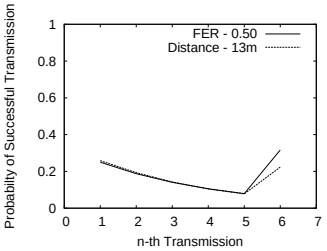
(a) FER 0.01 and Distance 1m



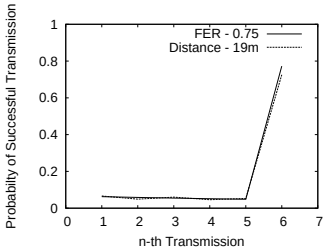
(b) FER 0.10 and Distance 6m



(c) FER 0.25 and Distance 9m



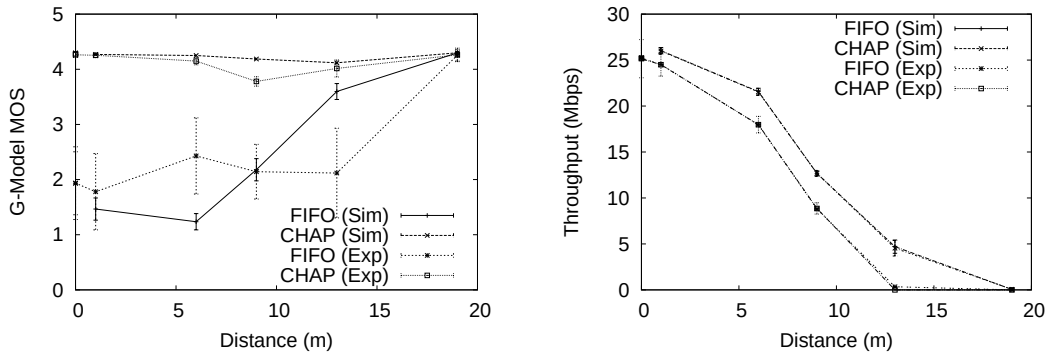
(d) FER 0.50 and Distance 13m



(e) FER 0.75 and Distance 19m

Figure 7.7: Matching Frame Error Rates and Simulation Distances

Frame error rates are induced for the FTP traffic to emulate the FTP client machine moving further away from the AP because the G-model MOS is resilient to packet loss. Figure 7.8 depicts G-model MOS's and total throughputs for the simulations and experiments. The G-model MOS from the experiments was calculated with the lowest 2.5% of jitter values dropped to account for outliers. As explained in Section 6.2, the outliers may be the result of unsimulated factors such as system level overheads and differences



(a) G-model MOS (b) Total Throughput

Figure 7.8: Controlled Experiment (Distance Error Emulation)

between TCP implementations. Further investigation is necessary to pinpoint the exact causes of differences, and it is discussed in Section 8.2. Simulation and experiment results demonstrate the same trend although they have slight differences in values. As the FTP client moves away from the AP, the G-model MOS for FIFO stays around 2 up to about 10m for both simulation and experiment. Then, the G-model MOS increases towards 4.27 as the FTP client is further away. For CHAP, the G-model MOS in simulations stays around 4.27 while the G-model MOS in experiments ranges between 3.8 and 4.2. The throughputs between FIFO and CHAP are almost identical in the simulations and experiments. The throughputs in the simulations and experiments also demonstrate the same decreasing trend as the FTP client is further away, eventually reaching 0 Mbps at 19m. We believe that the differences are due to the TCP Cubic implementations and the timer granularity of the Linux implementation whose timer intervals are $\frac{1}{4096} = 0.000244$ (244 μs). As explained in Section 7.1, the lowest timer with small packets can be close to 177 μs, which the Linux system is not capable of handling with the given timer intervals.

7.3 Case Study

This section describes a case study conducted in a home network showing the performance evaluation of Web browsing and online games.

7.3.1 Case Study Setup

Figure 7.9 depicts the experiment setup used to represent a home network. The Internet is connected by a 7 Mbps downstream, 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. An additional wireless-emulation delay of 2 ms is added to the emulated delay calculated by Equation 7.1 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. No frame error rates are induced to emulate wireless clients close to the AP. One computer runs one of the two delay-sensitive applications: *Firefox* for Web browsing and *Quake IV* for an online game. The other computer runs Firefox for bulk file download of the Debian Linux DVD ISO. Each scenario 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.

7.3.2 Web Browsing

Four popular news Web sites – Google News, Yahoo, Wall Street Journal (WSJ), and CNN – were used for the Web browsing tests. Table 7.1 indicates the composition of each Web site in terms of its total size in bytes and the number of text and non-text objects. Text objects are HTML pages while non-text objects include images, flash objects, etc. CNN is

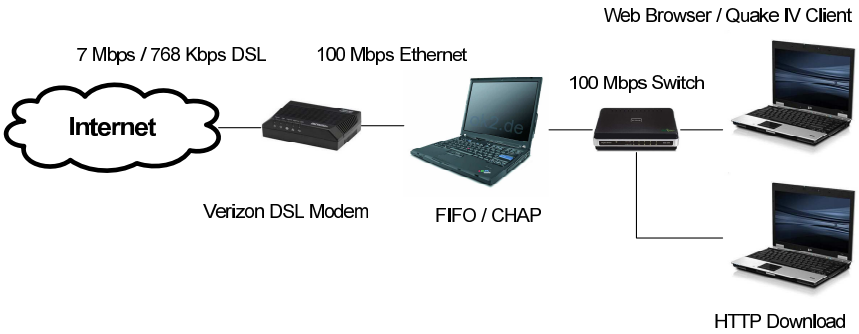
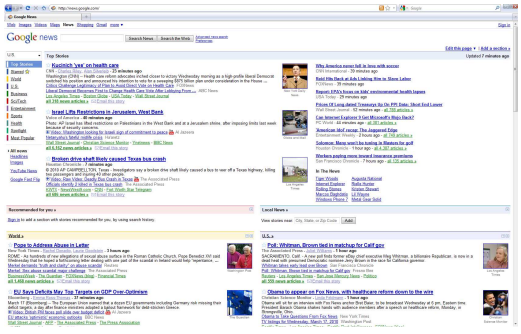
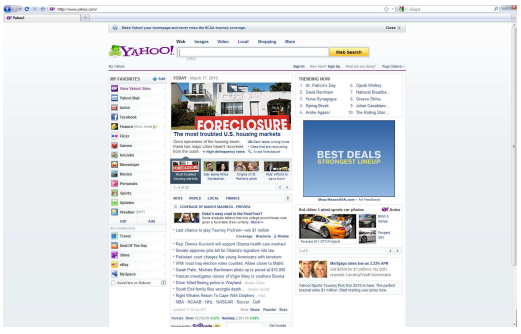


Figure 7.9: Case Study Experiment Topology

the heaviest in terms of the total size while WSJ has the most non-text objects. Google News is the lightest with respect to the size and the total number of objects. The 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.



(a) Google News



(b) Yahoo

(c) Wall Street Journal

(d) CNN

Figure 7.10: Sample Websites

Figure 7.11 depicts the average Web response times for five visits to each of the four Web sites. Error bars represent the standard deviation of the 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 0.525s and 0.514s, Yahoo in 1.43s and 1.45s, Wall Street Journal in 3.22s and 3.36s, and CNN in 3.54s and 3.40s using FIFO and CHAP

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

Table 7.1: Website Composition Data

Website	# of Text Objs	# of Non-Text Objs	Total # of Objs
Google News	2	4	6
Yahoo	6	48	54
WSJ	19	132	151
CNN	33	103	136

Website	Size of Text Objs (KB)	Size of Non-Text Objs (KB)	Total Size (KB)
Google News	146	9	155
Yahoo	45	277	322
WSJ	97	652	749
CNN	190	952	1142

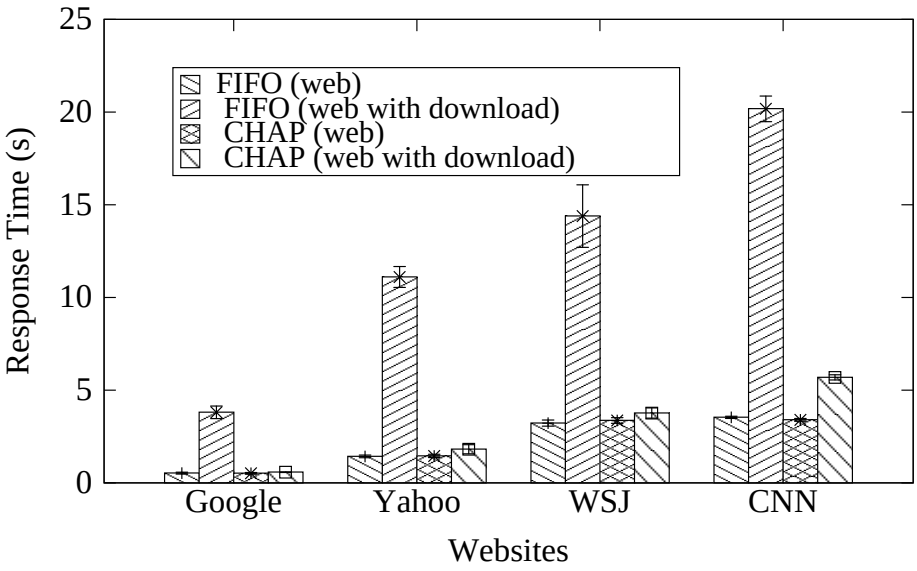


Figure 7.11: Web Response Time

respectively. When loading Web sites while downloading the Linux DVD ISO, the Web response times for FIFO increased by 300% to 700%. Google News finished loading in 3.81s (+625%), Yahoo in 11.11s (+678%), WSJ in 14.40s (+347%), and CNN in 20.18s (+470%) 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 0.574s (+12%), Yahoo in 1.81s (+25%), WSJ in 3.77s (+12%), and CNN in 5.69s (+67%). Videos of loading CNN have been recorded to illustrate the differences in

browsing performance under these conditions.[5]

7.3.3 Quake IV

Quake IV is used to evaluate performance for online games. Quake IV is a first person shooter game, representing 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 [31]. 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 average ping times to each server. The IP addresses returned by qstat are mapped to longitudes and latitudes according to GeoIP[7]. Figure 7.12 depicts the geographical locations of active Quake IV servers around the world in March 2010, with the white circle indicating Worcester, MA, USA.

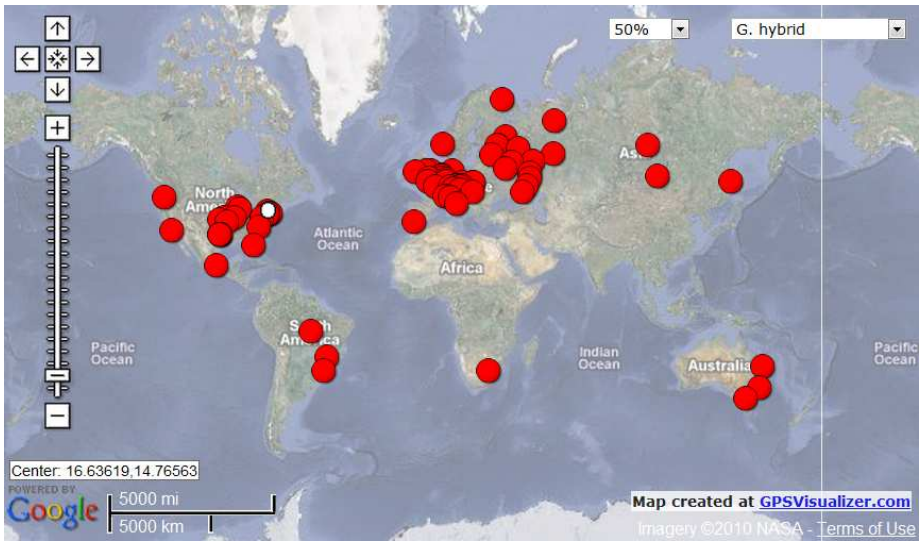
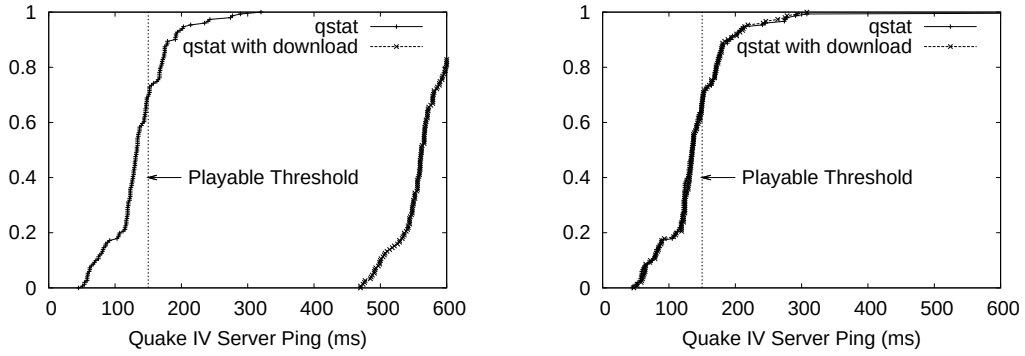


Figure 7.12: Quake IV Servers around the World

Figure 7.13 depicts 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

[5]http://perform.wpi.edu/chap/
[6]http://www.qstat.org/
[7]http://www.maxmind.com/app/ip-location



(a) FIFO (b) CHAP

Figure 7.13: CDF of Quake IV Server Pings

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. Videos have been recorded to illustrate the differences in game performance under these conditions.[8]

7.4 Summary

This chapter provides the implementation of CHAP and performance evaluation of the implementation. Section 7.1 describes the implementation details of CHAP using a queueing discipline (Qdisc) in Linux. Section 7.2 offers a series of controlled experiments to validate the implementation against the equivalent simulation results. Similarly to Section 6.2, the simulation and experiment results show a similar trend such as the degradation in the G-model MOS in the presence of a concurrent FTP download, even if they do not align exactly

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

with each other. Section 7.3 demonstrates how CHAP benefits delay-sensitive applications in a real home network. Web browsing and an online game (Quake IV) are tested with a concurrent HTTP download. CHAP provides a significant improvement in the quality of Web browsing and Quake IV over FIFO while providing a comparable throughput for the concurrent HTTP download.

Chapter 8

Conclusions and Future Work

The dissertation provides a queue management solution, called Credit-based Home Access Point (CHAP), which can be deployed in wireless access points for home networks to: 1) minimize network delays to support quality of service (QoS) requirements of delay-sensitive applications such as Web, VoIP, steaming media, and network games; 2) adjust to dynamic wireless conditions to provide better overall quality of applications; and 3) provide an automatic classification of network activities without any manual configuration from users. CHAP has been implemented as a queue controller in NS2 simulator and as a queueing discipline (Qdisc) in Linux for performance evaluation. Section 8.1 summarizes this dissertation research and Section 8.2 lists possible future work.

8.1 Summary

The core of the home network is the wireless access point that connects all the devices at home through a single, shared Internet connection. Credit-based Home Access Point (CHAP) can be implemented in the existing wireless access points to minimize network delays for delay-sensitive applications and maintain overall application quality in dynamic wireless conditions.

CHAP is an active queue management approach designed to minimize queueing delay through efficient per-flow management. CHAP detects bandwidth and wireless channel

usage costs for each flow and manages credits accordingly. Unlike other QoS mechanisms, CHAP provides a single parameter that is pre-configured for practical deployment and removes the need for home users to configure their access points for quality improvement. Unlike other credit-based queue management mechanisms such as CBFQ [125] and WCFQ [126], CHAP allows bursts of packets from flows as long as they have more credits.

CHAP meets a few requirements for "smart middle" devices recommended by Calvert *et al.* in their paper [10]. The single, pre-configured parameter satisfies the self-configuring requirement by relieving home users of the burden of AP configuration. CHAP is an active queue management approach that does not require any modifications to end hosts, preserving the compatibility with existing external TCP/IP-based applications. Moreover, CHAP achieves application independence by relying on the relationship between the bandwidth and delay tolerance of user activities instead of the properties, patterns, and/or signatures of specific applications.

Through a simulation study using NS2, this dissertation demonstrates that CHAP can efficiently support a wide range of traffic conditions, providing high overall application quality. CHAP can provide application quality that is significantly improved over FIFO and similar to SPQ, but without the explicit classification required of SPQ. In addition, this dissertation shows that the benefits of CHAP increases with the queue size of the wireless access point. Edge behavior of CHAP with a P2P application in the background shows that CHAP slowly degrades performance of delay-sensitive applications crossing the edge, where the throughput of a delay-sensitive application becomes smaller than the throughput of a delay-insensitive application. As the network latency between end hosts increased, TCP applications benefit more with CHAP.

Distances of wireless nodes to the AP in the simulation runs affect the wireless connectivity of the nodes to the AP. CHAP improves the quality of the delay-sensitive application close to SPQ regardless of its wireless connectivity. In the video scenario, SPQ prioritizes the video traffic, sacrificing the FTP throughput even when the viewable frame rate of the video is near 0. In the same case, CHAP reduces the priority of the video traffic

due to its frames experiencing longer transmission times. Therefore, the FTP application achieves higher throughput under CHAP compared to FIFO and SPQ. In the cases where the wireless connectivity of the FTP application node varies, CHAP consistently provides quality higher than FIFO and similar to SPQ for the delay-sensitive applications. The FTP throughput degradation due to varying wireless connectivity is the same for FIFO, CHAP, and SPQ. In the scenarios where the wireless connectivity of both nodes degrade together, CHAP generally provides quality higher than FIFO and similar to SPQ for both the delay-sensitive application and FTP application.

The exploration of different TCP variants, network latencies, and queue sizes demonstrates that each factor has a significant impact on application performance. Unfortunately, none of these are fixed on the Internet. Different operating systems and versions default to different TCP congestion control mechanisms. Manufacturers impose a range of queue sizes in their wireless APs. Network latencies depend on where home users access the Internet and where the systems their applications communicate with are. When examined independently, TCP CUBIC seems to provide the best throughput regardless of queue sizes and network latencies. Large queue sizes help all tested TCP variants in terms of their throughput. Long and diverse network latencies result in a degradation and unfairness in TCP throughputs.

Finally, this dissertation evaluates CHAP using a Linux implementation. The Linux implementation connects end hosts over an emulated IEEE 802.11g WLAN with and without induced error rates on top of wired 100 Mbps Ethernet connections. Validation of the implementation shows that CHAP does not use more system resources than FIFO with respect to CPU and memory. Case studies of Web browsing and online games in a home network demonstrate significant decrease in Web page loading times and game server ping times compared to FIFO in the presence of bulk data download.

8.2 Future Work

Section 6.2 and Section 7.2 demonstrate the differences between simulation and experiment results. Although it is likely that system level overheads, wireless interference and different TCP implementation contribute to the differences, further investigation into the exact causes can help refine the simulator to mimic the real world more closely.

This dissertation evaluates CHAP with the parameter I set to 25 ms. Large values for I may accommodate longer bursts from flows but they make CHAP behave more like SPQ. Small values make it difficult for CHAP to distinguish one flow from another with respect to their credits. Section 5.4 presents an analytical justification for this value, but it will be beneficial to conduct a sensitivity analysis of I on CHAP performance.

Although this dissertation evaluates CHAP with an emulated IEEE 802.11 data link layer, CHAP needs to be implemented to support a real IEEE 802.11 infrastructure network. In Linux, it may be as easy as setting it up as a wireless access point using *hostAP*[1] software. As explained in the dissertation, constant kernel panic and disassociation of wireless nodes impeded such implementation. In addition, there are many commercial access points that support custom Linux firmware[2].

Depending on the size of the residence, more than one access point (AP) may be required to provide wireless connectivity all around the residence. Network configurations with multiple APs may or may not satisfy the assumption that each AP is the bottleneck to home networks. For example, the Internet connection may be the bottleneck for a house with a 100 Mbps Internet connection with two 802.11g (54 Mbps) APs that are equally active. In such cases, CHAP needs to be aware of other APs to keep providing higher application quality.

This dissertation does not consider upstream traffic as network traffic in home networks is generally asymmetric. However, upstream traffic shares the wireless medium with downstream traffic. Applications, such as P2P file sharing and Skype, with both downstream

[1] http://hostap.epitest.fi/
[2] http://www.dd-wrt.com/

and upstream network traffic have become popular. Therefore, it may be beneficial for CHAP to consider upstream traffic as much as downstream traffic in its mechanism to provide high application performance.

The focus of CHAP in this dissertation is to improve home network application performance. Wireless networks are deployed not only at homes but also in corporations and educational institutions. Although the network traffic composition may be different, CHAP may also be beneficial to these networks. Network configurations in these networks may differ from residential networks, placing bottlenecks at different points in the network topology. It will be necessary to analyze the types of common activities in such networks to see if CHAP can improve quality of such activities in those networks.

Although this dissertation covers CHAP tied to IEEE 802.11 networks, the mechanism does not rely on anything specific to IEEE 802.11. The cost for each packet for a flow is merely the transmission time for the packet, and it can be obtained from any other wireless data link layer. Therefore, it is possible to apply the general CHAP algorithm in other types of wireless local area networks. Furthermore, it can be extended to wireless personal area networks (WPANs), metropolitan area networks (WMANs), and even wide area networks (WWANs).

The video application implemented in NS2 simulates an unresponsive UDP video streaming that uses encoded video traces [13, 14]. These traces of videos encoded using scalable video coding (SVC) include multiple layers at different frame rates. Enhancements such as sending frames over TCP and switching between layers according to available bandwidth will help the video application to generate more realistic video streaming network traffic.

Appendix A

List of Acronyms

ABE Alternate Best Effort

ACK ACKnowledgement

AIFS Arbitration Inter-Frame Space

AJAX Asynchronous Javascript and XML

AP Access Point

AQM Active Queue Management

ARED Adaptive Random Early Detection

BBS Bulletin Board Service

BER Bit Error Rate

BIC Binary Increase Congestion control

BLINC BLINd Classification

CATNIP Context-Aware Transport/Network Internet Protocol

CBFQ Credit-Based Fair Queueing

CBQ Class Based Queue

CBT Class Based Threshold

CF Contention Free

CFP Contention Free Period

CFS Completely Fair Scheduler

CHAP Credit-based Home Access Point

ChIPS Cut-In Packet Scheduling

CNP Congestion Notification Probability

CP Contention Period

CPU Central Processing Unit

CSFQ Core Stateless Fair Queueing

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

CTS Clear-to-Send

CUBIC Enhanced Binary Increase Congestion control

CVS Concurrent Versions System

CW Contention Window

D-CBT Dynamic Class Based Threshold

DCF Distributed Coordination Function

DiffServ Differentiated Service

DIFS Distributed Inter-Frame Space

DLS Direct Link Setup

DRR Deficit Round Robin

DSCP Differentiated Service Code Point

DSL Digital Subscriber Line

DSSS Direct Sequence Spread Spectrum

ECA Early Congestion Avoidance

ECN Explicit Congestion Notification

ERP Extended Rate Physical layer

FCFS First-Come-First-Serve

FEC Forward Error Correction

FHSS Frequency Hopping Spread Spectrum

FIFO First-In-First-Out

FiOS Fiber Optic Service

FPS First Person Shooter

FRED Fair Random Early Detection

FTP File Transfer Protocol

GoP Group of Pictures

HAN Home Area Network

HC Hybrid Coordinator

HCF Hybrid Coordination Function

HD High Definition

HOL Head-of-Line

HTB Hierarchical Token Bucket

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IANA Internet Assigned Number Authority

IMAP Internet Message Access Protocol

IntServ Integrated Service

IP Internet Protocol

IR InfraRed

ISP Internet Service Provider

MAC Medium Access Control

MIMO Multiple Input Multiple Output

MMO Massive Multiplayer Online

MOS Mean Opinion Score

MSN Microsoft Network

MTU Maximum Transmission Unit

NAT Network Address Translation

NS2 Network Simulator

OFDM Orthogonal Frequency Division Multiplexing

P2P Peer-to-Peer

PCF Point Coordination Function

PDA Personal Digital Assistant

PFIFO Packet First-In-First-Out

PHB Per-Hop Behavior

PIFS Point coordination Inter-Frame Space

POP Post Office Protocol

PPDQ Probabilistic Priority Discipline Queue

PSP PlayStation Portable

PSQA Pseudo-Subjective Quality Assessment

Qdisc Queueing Discipline

QoS Quality of Service

RBPLW Rate-Based Pacing of the Last Window

RD Rate-Delay

RED Random Early Detection

RG Residential Gateway

RIO RED with In or Out

RR Round Robin

RTP Real-time Transport Protocol

RTS Real Time Strategy

RTS Request-to-Send

RTSP Real-Time Streaming Protocol

SCFQ Self-Clocked Fair Queueing

SCP Secure Copy

SIFS Short Inter-Frame Space

SJF Shortest Job First

SLA Service Level Agreement

SMB Server Message Block

SMTP Simple Mail Transfer Protocol

SNR Signal to Noise

SPM Selective Packet Marking

SPQ Strict Priority Queue

SSH Secure Shell

SSID Service Set IDentifier

TC Traffic Control

TCP Transmission Control Protocol

TOS Type of Service

TXOP Transmission Opportunity

UDP User Datagram Protocol

VNC Virtual Network Computing

VoIP Voice over IP

WAVE Wireless Access for the Vehicular Environment

WCFQ Wireless Credit-based Fair Queueing

WFQ Weighted Fair Queueing

WLAN Wireless Local Area Network

WMAN Wireless Metropolitan Area Network

WoW World of Warcraft

WPAN Wireless Personal Area Network

WPP Wireless Performance Prediction

WSM Windows Streaming Media

WSN Wireless Sensor Network

WWAN Wireless Wide Area Network

WWW World Wide Web

Appendix B

Supplementary Figures

Table B.1: Multiple Application Scenario - Summary of Performance Metrics (No FEC)

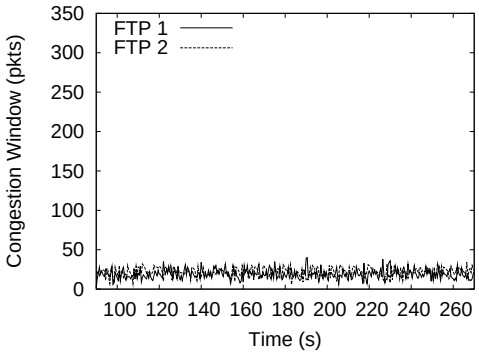
App. (Unit)	DropTail	CHAP	SPQ	% Impr
Game (MOS)	3.57	4.27	4.25	+20%
VoIP (MOS)	4.31	4.42	4.42	+3%
Video (fps)	15.58	30.00	30.00	+93%
Web (ms)	113.39	56.07	41.85	+51%
FTP (Mbps)	22.95	22.91	22.91	0%

Table B.2: Multiple Application Scenario - Summary of Performance Metrics (Small FEC)

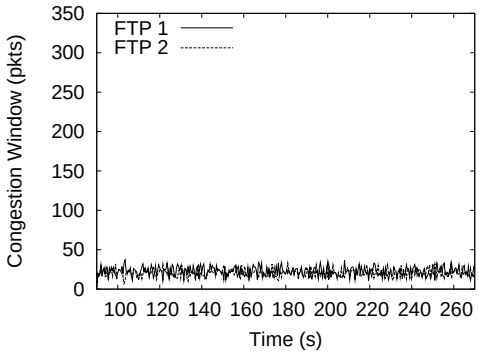
App. (Unit)	DropTail	CHAP	SPQ	% Impr
Game (MOS)	3.56	4.27	4.25	+20%
VoIP (MOS)	4.31	4.42	4.42	+3%
Video (fps)	17.44	30.00	30.00	+72%
Web (ms)	100.60	53.94	41.41	+46%
FTP (Mbps)	22.94	22.89	22.89	0%

Table B.3: Multiple Application Scenario - Summary of Performance Metrics (Large FEC)

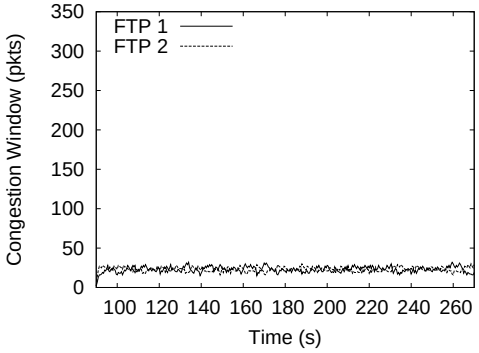
App. (Unit)	DropTail	CHAP	SPQ	% Impr
Game (MOS)	3.61	4.27	4.24	+18%
VoIP (MOS)	4.30	4.42	4.42	+3%
Video (fps)	23.16	30.00	30.00	+30%
Web (ms)	124.08	58.09	42.73	+53%
FTP (Mbps)	22.57	22.51	22.51	0%



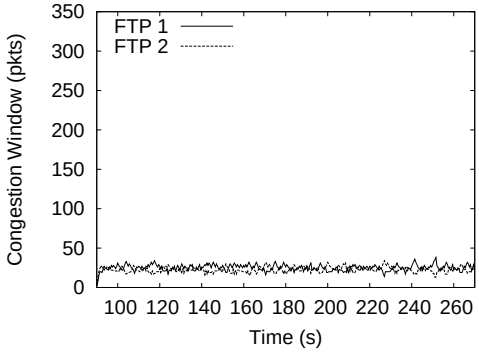
(a) TCP NewReno (DropTail)



(b) TCP NewReno (CHAP)

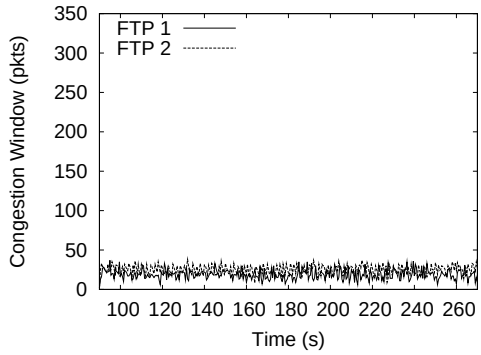


(c) TCP Cubic (DropTail)

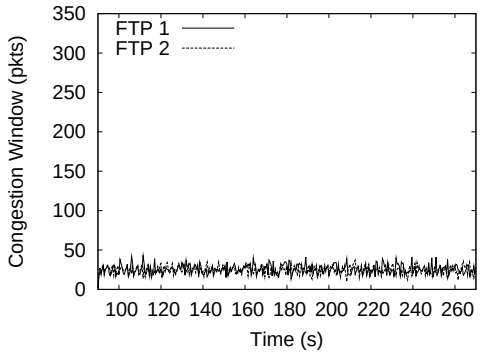


(d) TCP Cubic (CHAP)

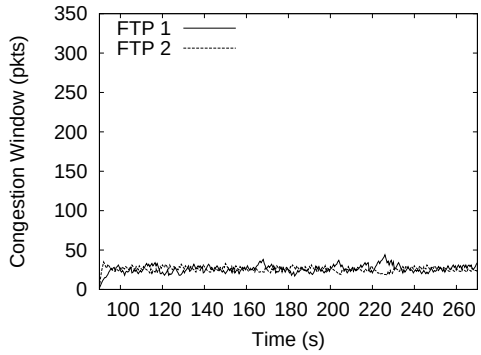
Figure B.1: Congestion Window (10ms)



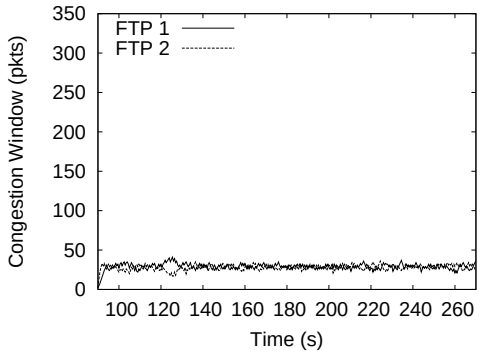
(a) TCP NewReno (DropTail)



(b) TCP NewReno (CHAP)

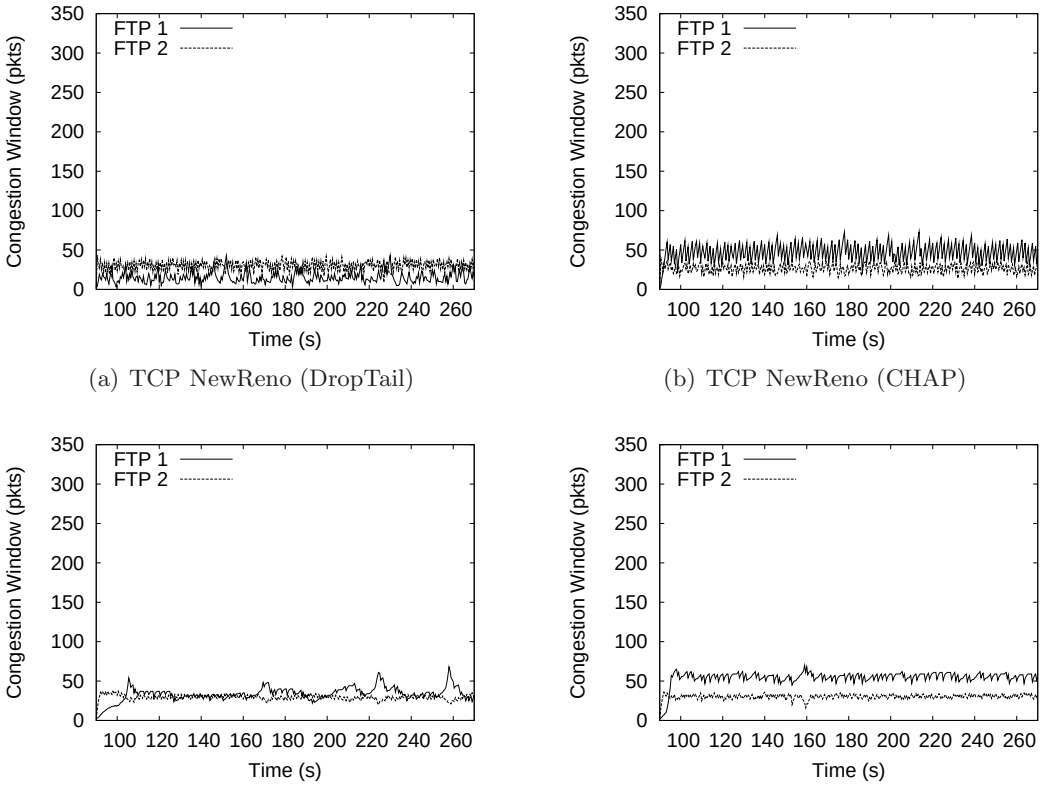


(c) TCP Cubic (DropTail)



(d) TCP Cubic (CHAP)

Figure B.2: Congestion Window (20ms)



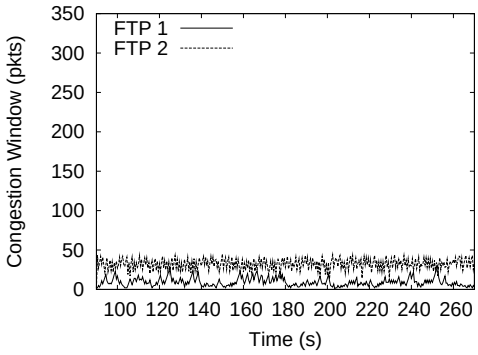
(a) TCP NewReno (DropTail)

(b) TCP NewReno (CHAP)

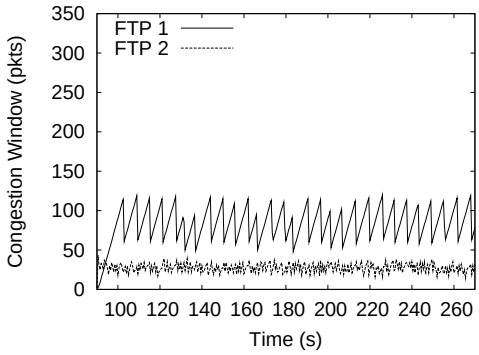
(c) TCP Cubic (DropTail)

(d) TCP Cubic (CHAP)

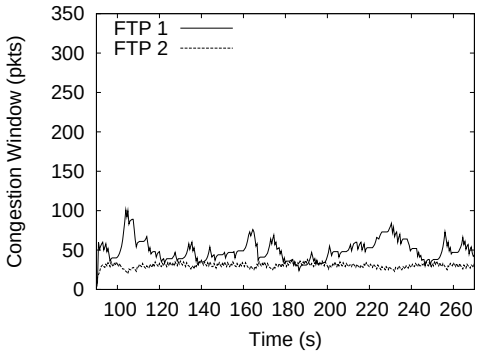
Figure B.3: Congestion Window (50ms)



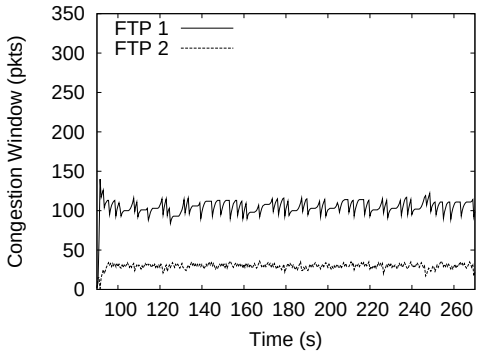
(a) TCP NewReno (DropTail)



(b) TCP NewReno (CHAP)

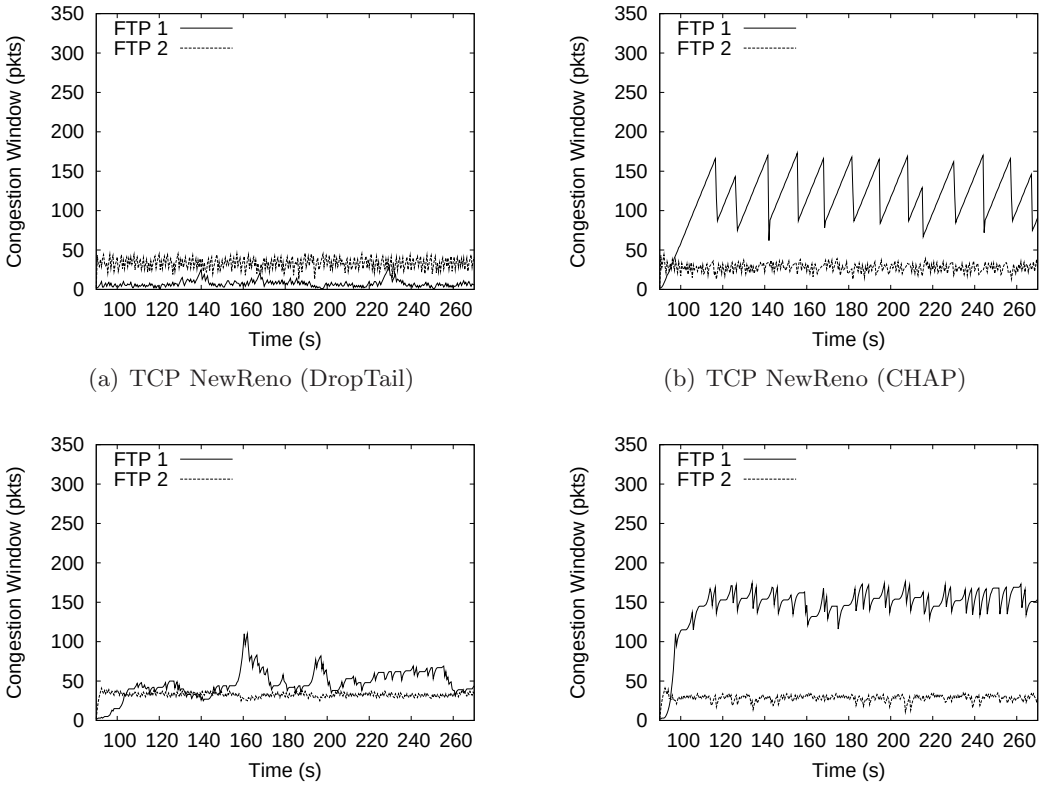


(c) TCP Cubic (DropTail)



(d) TCP Cubic (CHAP)

Figure B.4: Congestion Window (100ms)



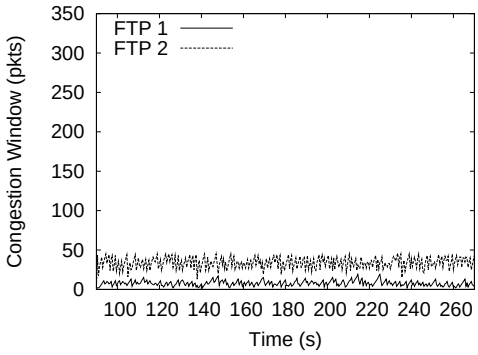
(a) TCP NewReno (DropTail)

(b) TCP NewReno (CHAP)

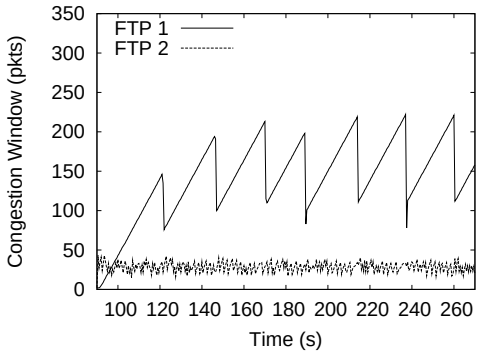
(c) TCP Cubic (DropTail)

(d) TCP Cubic (CHAP)

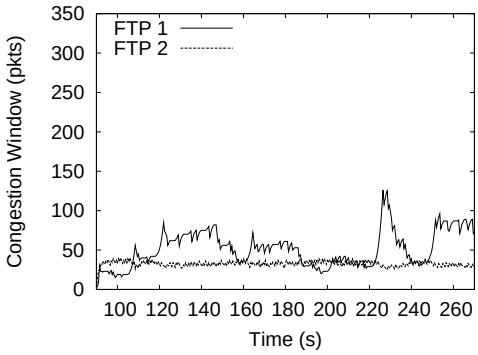
Figure B.5: Congestion Window (150ms)



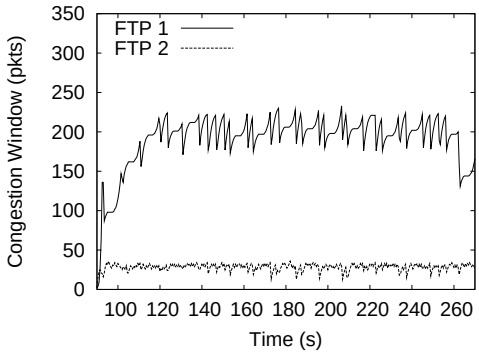
(a) TCP NewReno (DropTail)



(b) TCP NewReno (CHAP)

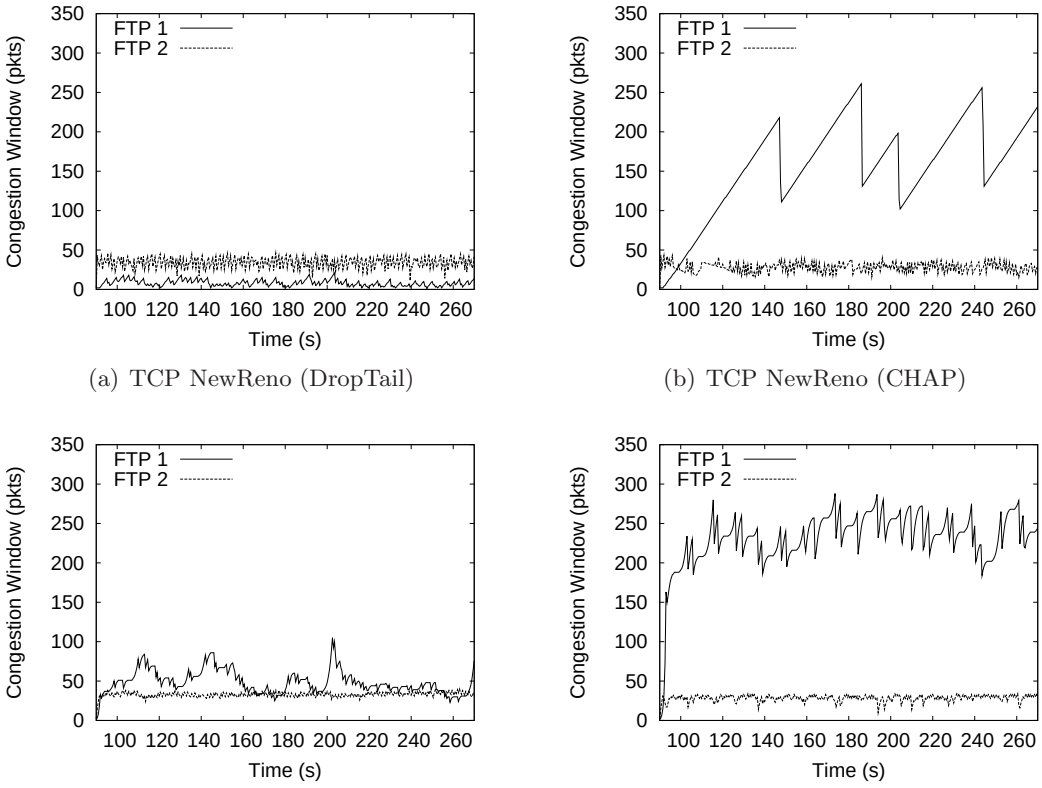


(c) TCP Cubic (DropTail)



(d) TCP Cubic (CHAP)

Figure B.6: Congestion Window (200ms)



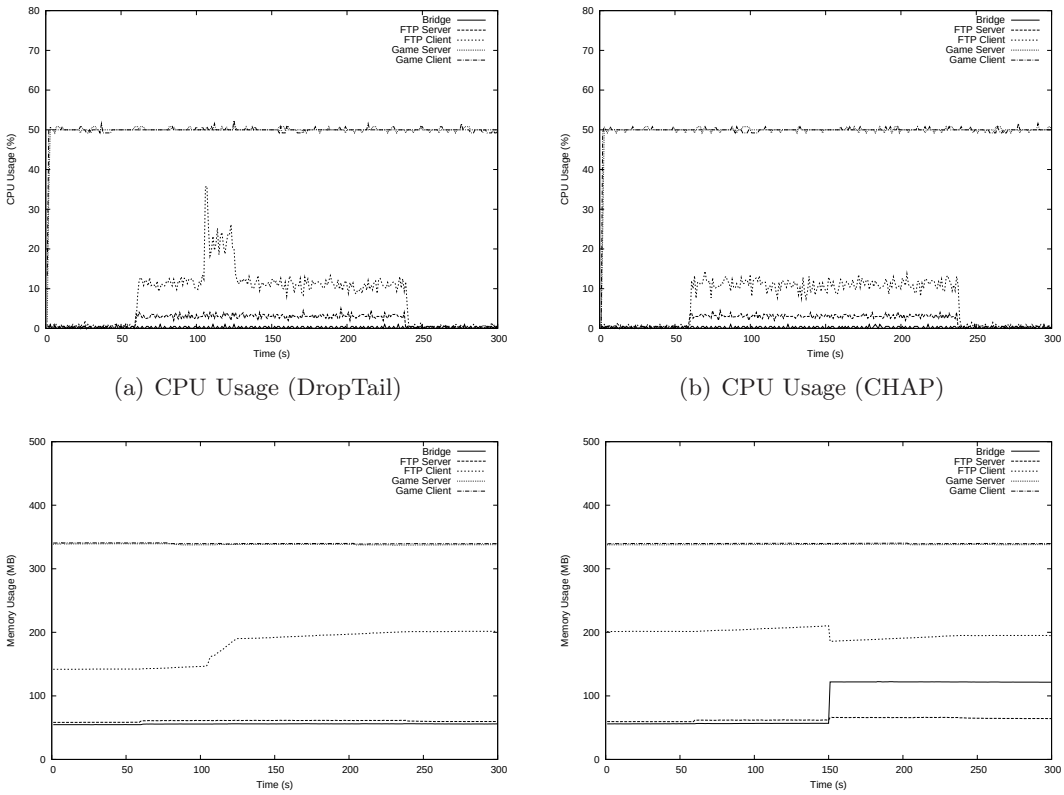
(a) TCP NewReno (DropTail)

(b) TCP NewReno (CHAP)

(c) TCP Cubic (DropTail)

(d) TCP Cubic (CHAP)

Figure B.7: Congestion Window (250ms)



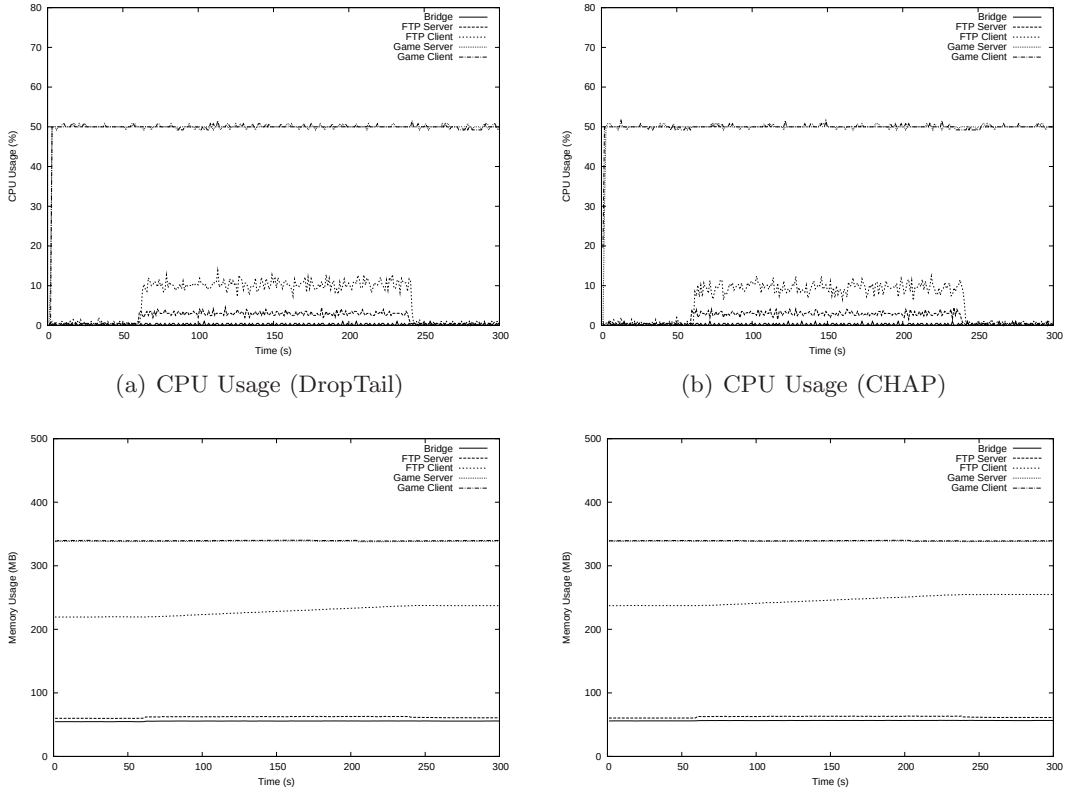
(a) CPU Usage (DropTail)

(b) CPU Usage (CHAP)

(c) Memory Usage (DropTail)

(d) Memory Usage (CHAP)

Figure B.8: Controlled Experiment (FER = 0.0001) - CPU and Memory Usage



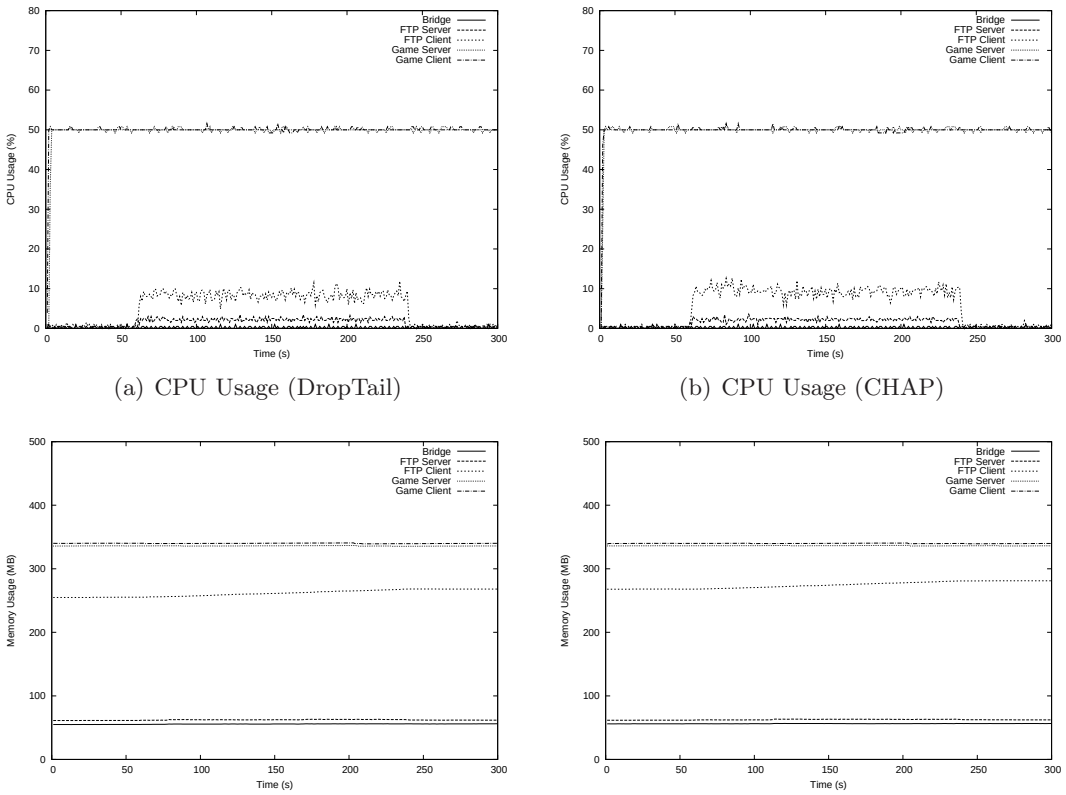
(a) CPU Usage (DropTail)

(b) CPU Usage (CHAP)

(c) Memory Usage (DropTail)

(d) Memory Usage (CHAP)

Figure B.9: Controlled Experiment (FER = 0.01) - CPU and Memory Usage



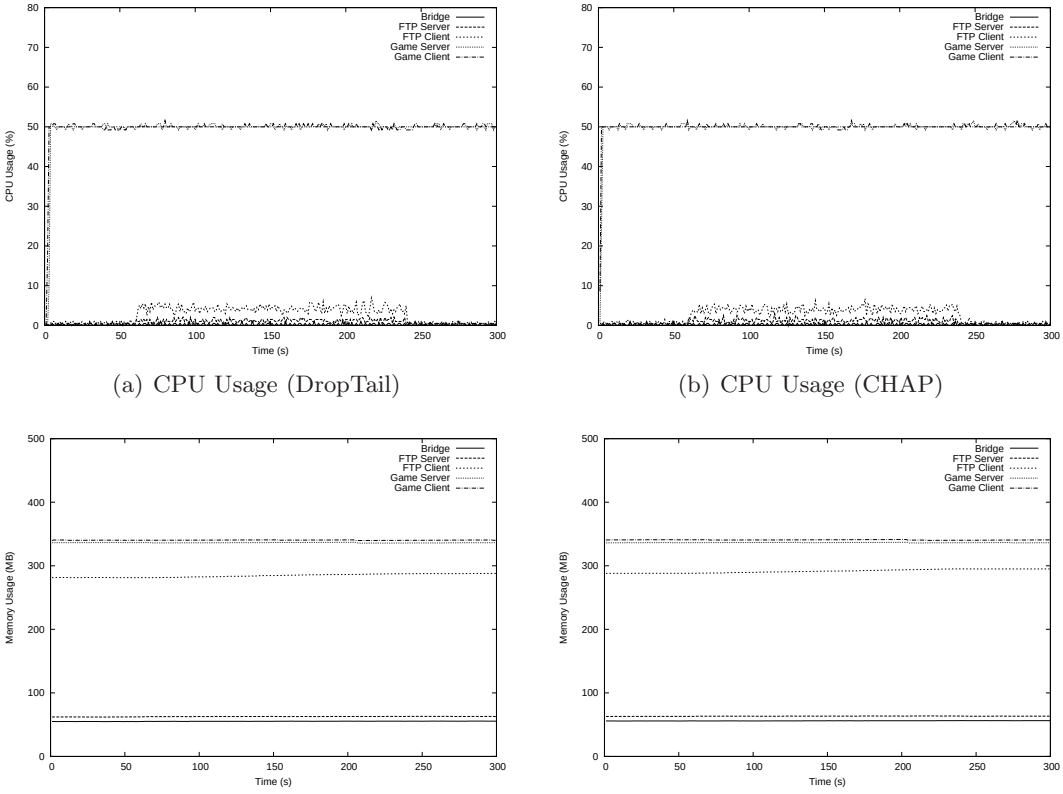
(a) CPU Usage (DropTail)

(b) CPU Usage (CHAP)

(c) Memory Usage (DropTail)

(d) Memory Usage (CHAP)

Figure B.10: Controlled Experiment (FER = 0.10) - CPU and Memory Usage



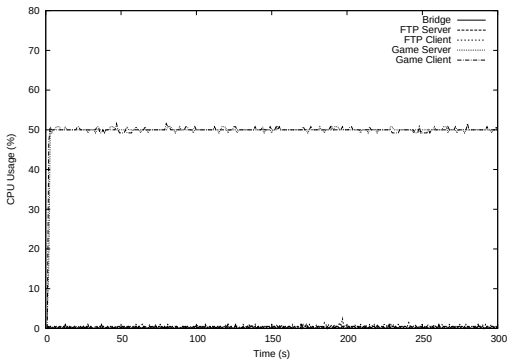
(a) CPU Usage (DropTail)

(b) CPU Usage (CHAP)

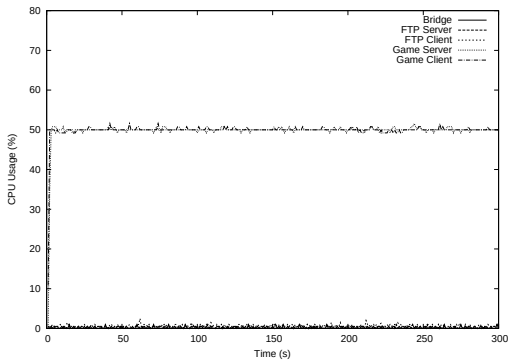
(c) Memory Usage (DropTail)

(d) Memory Usage (CHAP)

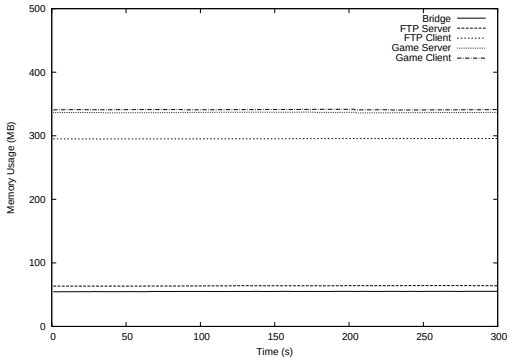
Figure B.11: Controlled Experiment (FER = 0.25) - CPU and Memory Usage



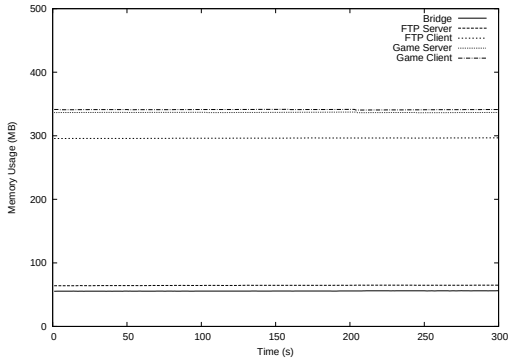
(a) CPU Usage (DropTail)



(b) CPU Usage (CHAP)

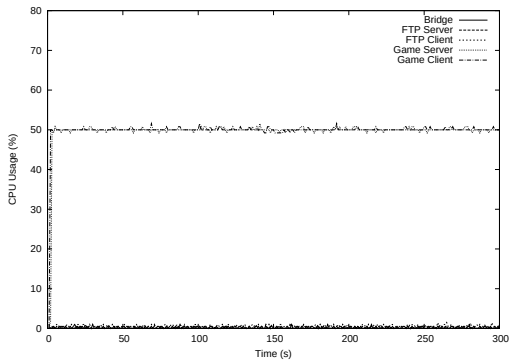


(c) Memory Usage (DropTail)

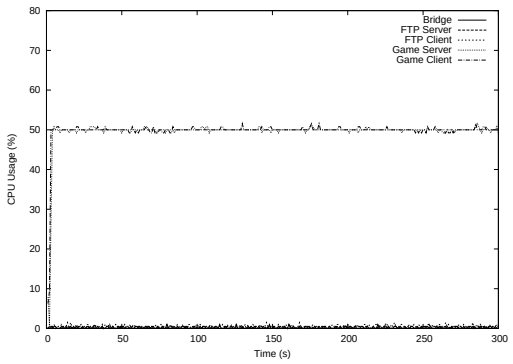


(d) Memory Usage (CHAP)

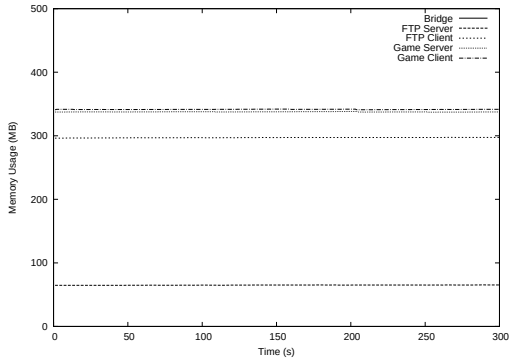
Figure B.12: Controlled Experiment (FER = 0.50) - CPU and Memory Usage



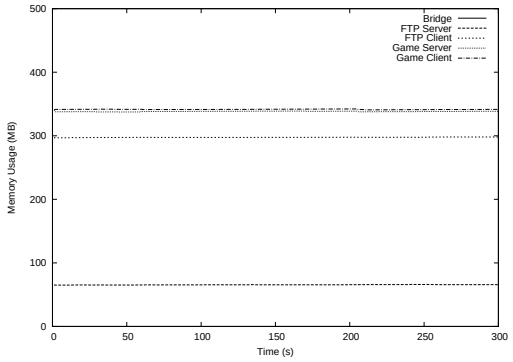
(a) CPU Usage (DropTail)



(b) CPU Usage (CHAP)



(c) Memory Usage (DropTail)



(d) Memory Usage (CHAP)

Figure B.13: Controlled Experiment (FER = 0.75) - CPU and Memory Usage

Bibliography

[1] Wikipedia, "IEEE 802.11n." Online at http://en.wikipedia.org/wiki/IEEE_802.11n. 1

[2] Linksys by Cisco, "Simultaneous Dual-N Band Wireless Router (WRT610N)." Online at http://www.linksysbycisco.com/US/en/products/WRT610N. 1

[3] Verizon, "Verzion FiOS Internet Plans." Online at http://www22.verizon.com/Residential/FiOSInternet/Plans/Plans.htm. 2

[4] AllBusiness, "HKBN launches world's first 1 Gbps residential broadband services.." Online at http://www.allbusiness.com/media-telecommunications/data-transmission-broadband/7322542-1.html. 3

[5] Converge, "Hong Kong Broadband Launches 1 Gbps Home Service for US$215/month.." Online at http://www.convergedigest.com/Bandwidth/newnetworksarticle.asp?ID=14545. 3

[6] Japan Today, "KDDI to launch 1Gbps fiber-optic service in Oct." Online at http://www.japantoday.com/category/technology/view/kddi-to-launch-1gbps-fiber-optic-service-in-oct. 3

[7] ElectronicDesign, "Hanaro 100 Mbps Broadband Thrives In Korea.." Online at http://electronicdesign.com/Articles/Index.cfm?AD=1&ArticleID=17215. 3

[8] BroadbandReports, "Amsterdam Tests Residential 1Gbps Fiber.." Online at http://www.broadbandreports.com/shownews/Amsterdam-Tests-Residential-1Gbps-Fiber-97642. 3

[9] TheLocal, "Sigbritt, 75, has world's fastest broadband.." Online at http://www.thelocal.se/7869/20070712/. 3

[10] K. L. Calvert, W. K. Edwards, and R. E. Grinter, "Moving Toward the Middle: The Case Against the End-to-End Argument in Home Networking," in *Proceedings of the Sixth ACM Conference on Hot Topics in Networks (HotNets-VI)*, 2007. 3, 156

[11] J. Cao, Y. Wu, and C. Williamson, "A Station-Based Adaptation Algorithm to Improve Robustness of IEEE 802.11," in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, 2006. 5, 6

[12] U. of California Berkeley, "The Network Simulator - ns-2." Online at http://www.isi.edu/nsnam/ns/. 7

[13] P. Seeling, M. Reisslein, and B. Kulapala, "Network Performance Evaluation with Frame Size and Quality Traces of Single-Layer and Two-Layer Video: A Tutorial," *IEEE Communications Surveys and Tutorials*, vol. 6, no. 3, pp. 58–78, 2004. 9, 90, 97, 159

[14] G. V. der Auwera, P. T. David, and M. Reisslein, "Traffic and Quality Characterization of Single-Layer Video Streams Encoded with H.264/MPEG-4 Advanced Video Coding Standard and Scalable Video Coding Extension," in *IEEE Transactions on Broadcasting*, vol. 54, pp. 698–718, September 2008. 9, 90, 97, 159

[15] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," 2004. 9, 130

[16] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast, Long Distance Networks," in *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM)*, March 2004. 9, 130

[17] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42. 9, 98, 130

[18] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks," in *Proceedings of the 25th Conference of the IEEE Communications Society (INFOCOM)*, April 2006. 9, 98, 130

[19] Sony, "Playstation Network." Online at http://www.us.playstation.com/PSN. 12

[20] Microsoft, "Xbox LIVE." Online at http://www.xbox.com/en-US/live/. 12

[21] Bungie, "Halo 2." Online at http://www.bungie.net/Projects/Halo2/default.aspx. 12

[22] S. Zander and G. Armitage, "A traffic model for the xbox game halo 2," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2005. 12, 96, 99, 142

[23] Blizzard Entertainment, "Starcraft." Online at http://www.blizzard.com/us/starcraft/. 13

[24] A. Dainotti, A. Pescape, and G. Ventre, "A packet-level traffic model of starcraft," in *Proceedings of the 2005 Second International Workshop on Hot Tipics in Peer-to-Peer Systems (HOT-P2P '05)*, 2005. 13

[25] Blizzard Entertainment, "Battle.Net." Online at http://www.battle.net/. 13

[26] R. A. Bangun, E. Dutkiewicz, and G. J. Anido, "An Analysis of Multi-Player Network Games Traffic," in *Proceedings of the 9th ACM international conference on Multimedia*, (Ottawa, Canada), 2001. 13, 35

[27] Blizzard Entertainment, "World of Warcraft." Online at http://www.worldofwarcraft.com/. 13

[28] MMOGCHART.COM, "MMOG Subscriptions Market Share - April 2008." Online at http://www.mmogchart.com/Chart7.html. 13

[29] MMOGCHART.COM, "MMOGCHART.COM." Online at http://www.mmogchart.com/. 13

[30] P. Svoboda, W. Karner, and M. Rupp, "Traffic analysis and modeling for world of warcraft," in *Proceedings of the IEEE International Conference on communications*, 2007. 13

[31] 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. 13, 151

[32] Skype Limited, "skype." Online at http://www.skype.com/. 14

[33] S. Sharafeddine, A. Riedl, J. Glasmann, and J. Totzke, "On traffic characteristics and bandwidth requiremetns of voice over ip applications," in *Proceedings of the 8th IEEE International Symposium on Computers and Communication*, 2003. 14

[34] "ITU-T Recommendation G.108: Application of the E-Model: A Planning Guide," September 1999. 14

[35] "Implementation Architecture Specification for the Premium IP Service, 2002. Gn1(Gant) deliverable d9.7-addendum 1." Online at http://archive.dante.net/geant/public_deliverables/GEA-02-079v2.pdf. 14

[36] MacInTouch, "MacInTouch Special Report: Streaming Media Market Report - 2001." Online at http://www.macintouch.com/stream2001.html. 15

[37] MacInTouch, "MacInTouch: The original mac news and information site since 1994." Online at http://www.macintouch.com/. 15

[38] Real, "Real Player." Online at http://www.real.com/. 15

[39] Microsoft, "Windows Media." Online at http://www.microsoft.com/windows/WindowsMedia. 15

[40] Apple, "QuickTime." Online at http://quicktime.apple.com/. 15

[41] BNET Business Network, "Media Player Format Share for 2006 Confirms Windows Media Remains Dominant with A 50.8% Share of Video Streams Served, Followed By Flash At 21.9% - 'CDN Growth and market Share Shifts: 2002-2006'." Online at http://findarticles.com/p/articles/mi_m0EIN/is_2006/_Dec_18/ai_n16912185/. 15

[42] Research and Markets, "Research and Markets." Online at http://www.researchandmarkets.com/. 15

[43] Adobe, "Flash." Online at http://www.adobe.com/products/flashplayer/. 15

[44] M. Li, M. Claypool, and R. Kinicki, "MediaPlayer versus RealPlayer - A Comparison of Network Turbulence," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, (Marseille, France), November 2002. 15

[45] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," 1998. 15

[46] Sling Media, "SlingBox." Online at http://www.slingmedia.com/. 15

[47] Sony, "LocationFree." Online at http://www.sonystyle.com/. 15

[48] Orb Networks Inc., "Orb." Online at http://www.orb.com/. 16

[49] W3C, "W3C HTML." Online at http://www.w3.org/html/. 16

[50] NCSA, "Mosaic Browser." Online at http://www.ncsa.uiuc.edu/Projects/mosaic.html. 16

[51] Microsoft, "Internet Explorer." Online at http://www.microsoft.com/ie/. 16

[52] mozilla, "Firefox Browser." Online at http://www.mozilla.com/en-US/firefox/. 16

[53] Google, "Chrome Browser." Online at http://www.google.com/chrome/. 16

[54] Apple, "Safari." Online at http://www.apple.com/safari/. 16

[55] Opera Software, "Opera Browser." Online at http://www.opera.com/. 16

[56] tcpdump.org, "tcpdump/libpcap." Online at http://www.tcpdump.org/. 16

[57] J. Charzinski, "Measured HTTP Performance and Fun Factors," in *Proceedings of ITC*, (Salvador, BA, Brasil), Dec. 2001. 16

[58] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann, "The New Web: Characterizing AJAX Traffic," in *Proceedings of the 2008 Passive and Active Measurement Conference*, (Cleveland, OH, USA), April 2008. 16, 17

[59] Google, "Google Maps." Online at http://maps.google.com/. 16

[60] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User-Perceived Quality into Web Server Design," in *Computer Networks*, (Amsterdam, Netherlands), 1999. 17

[61] Microsoft, "HotMail." Online at http://www.hotmail.com/. 17

[62] Google, "GMail." Online at http://www.gmail.com/. 17

[63] Yahoo!, "Yahoo! Mail." Online at http://mail.yahoo.com/. 17

[64] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," Second Quarter 2005. 17

[65] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system." Online at http://freenetproject.org/. 17

[66] Wikipedia, "Gnutella." Online at http://en.wikipedia.org/wiki/Gnutella. 17

[67] Kazaa, "Kazaa media desktop." Online at http://www.kazaa.com/. 17

[68] BitTorrent Inc., "BitTorrent." Online at http://www.bittorrent.com/. 17

[69] Overnet, "The overnet file-sharing network." Online at http://www.overnet.com/. 17

[70] eDonkey2000, "The eDonkey2000 file-sharing network." Online at http://www.edonkey2000.com/. 17

[71] N. Basher, A. Mahanti, A. Mahanti, C. Williamson, and M. Arlitt, "A Comparative Analysis of Web and Peer-to-Peer Traffic," in *Proceedings of WWW2008*, (Beijing, China), pp. 287–296, April 2008. 18

[72] N. Basher, A. Mahanti, A. Mahanti, C. Williamson, and M. Arlitt, "A comparative analysis of web and peer-to-peer traffic," in *Proceeding of the 17th international conference on World Wide Web*, 2008. 18

[73] 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. 18, 53, 67, 98, 129

[74] ICQ, "icq." Online at http://www.icq.com/. 18

[75] Microsoft, "Windows Live Messenger." Online at http://messenger.msn.com/. 18

[76] American Online, "AOL Instant Messenger." Online at http://www.aim.com/. 18

[77] Yahoo, "Yahoo! Messenger." Online at http://messenger.yahoo.com. 18

[78] Jabber Software Foundation, "Jabber." Online at http://www.jabber.org/. 18

[79] Google, "Google Talk." Online at http://www.google.com/talk/. 18

[80] qq, "QQ." Online at http://qq.co.za/. 18

[81] Billions Connected, "Global Instant Messaging Market Share - Open Data." Online at http://billionsconnected.com/blog/2008/08/global-im-market-share-im-usage/. 18

[82] Pew
Internet & American Life Project, "How Americans use instant messaging." Online at http://www.pewinternet.org//media/Files/Reports/2004/PIP_Instantmessage_Report.pdf.pdf. 18

[83] R. E. Grinter and L. Palen, "Instant messaging in teen life," in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002. 18

[84] Z. Xiao, L. Guo, and J. Tracey, "Understanding instant messaging traffic characteristics," in *Proceedings of the 27th International Conference on Distributed Computing Systems*, 2007. 18

[85] Bluetooth SIG, Inc., "Bluetooth." Online at http://www.bluetooth.org/. 21

[86] I. C. S. L. M. S. Committee, "IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications." Standard, Aug. 1999. 21

[87] ZigBee Alliance, "ZigBee." Online at http://www.zigbee.org/. 21

[88] Wikipedia, "WiMax." Online at http://en.wikipedia.org/wiki/Wimax. 21

[89] Wikipedia, "IEEE 802.11." Online at http://en.wikipedia.org/wiki/802.11. 22

[90] Sony, "PlayStation Portable (PSP)." Online at http://www.us.playstation.com/PSP. 24

[91] Nintendo, "Nintendo DS." Online at http://www.nintendo.com/ds. 24

[92] "Linksys by Cisco." Online at http://www.linksysbycisco.com/. 27

[93] "NETGEAR." Online at http://www.netgear.com/. 27

[94] "D-Link." Online at http://www.dlink.com/. 27

[95] W. Stallings, *Operating Systems: Internals and Design Principles*. Pearson, Prentice Hall, sixth ed., 2009. 29

[96] J. W. Chung, *Congestion Control for Streaming Media*. PhD thesis, Computer Science Dept. at Worcester Polytechnic Institute, 2005. 33

[97] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, Aug. 1993. 34

[98] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: active queue management," *IEEE Network*, vol. 15, pp. 48–53, May 2001. 34

[99] L. Carvalho, E. Mota, R. Aguiar, A. F. Lima, J. N. de Souza, and A. Barreto, "An E-Model Implementation for Speech Quality Evaluation in VoIP Systems," in *Proceedings of the 10th IEEE Symposium on Computers and Communications*, (Washington, DC, USA), 2005. 35

[100] P. Kyasanur and N. Vaidya, "Detection and Handling of MAC Layer misbehavior in Wireless Networks," in *Proceedings of the 2003 International Conference on Dependable Systems and networks*, pp. 173–182, June 2003. 41

[101] F. Cuomo, "An Architectural Model to Provide QoS in a Home Network and its Evaluation in a Real Testbed," *Journal of Networks*, pp. 44–53, June 2008. 42

[102] Wikipedia, "IEEE 802.1p." Online at http://en.wikipedia.org/wiki/IEEE_802.1p. 42

[103] K. Kilkki, "Differentiated Services for the Internet," in *Macmillan Technology Series, Technical Publishing*, (Indianapolis, IN, USA), June 1999. 42

[104] C. E. Palazzi, G. Pau, M. Roccetti, S. Ferretti, and M. Gerla, "Wireless home entertainment center: reducing last hop delays for real-time applications," in *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, (New York, NY, USA), 2006. 43

[105] G. Rubino, M. Varela, and J.-M. Bonnin, "Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric," *Oxford Computer Journal*, vol. 49, no. 2, pp. 137–155, 2006. 43

[106] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks; the single-node case," *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 3, pp. 344–357, 1993. 45

[107] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks; the multiple node case," *IEEE/ACM Transactions on Networking (TON)*, vol. 2, no. 2, pp. 130–150, 1994. 45

[108] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Service," *IEEE/ACM Transactions on Networking*, Aug. 1998. 48

[109] S. Floyd and V. Jacobson, "Link-sharing and Recourse Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, 1995. 48

[110] M. Parris, K. Jeffay, and F. D. Smith, "Lightweight Active Router-Queue Management for Multimedia Networking," *Multimedia Computing and Networking, SPIE Proceedings Series*, vol. 3020, January 1999. 48

[111] J. Chung and M. Claypool, "Dynamic-CBT and ChIPS - Router Support for Improved Multimedia Performance on the Internet," in *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, (Chapel Hill, NC, USA), June 2000. 48

[112] D. Lin and R. Morris, "Dynamics of Random Early Detection," in *Proceedings of ACM SIGCOMM Conference*, (Cannes, France), Sept. 1997. 48, 57

[113] P. Hurley, M. Kara, J. L. Boudec, and P. Thiran, "ABE: Providing a Low Delay within Best Effort," *IEEE Network Magazine*, May/June 2001. 49

[114] M. Podlesny and S. Gorinsky, "RD Network Services: Differentiation through Performance Incentives," in *Proceedings of ACM SIGCOMM 2008*, 2008. 49

[115] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," tech. rep., 2001. 49

[116] V. Phirke, M. Claypool, and R. Kinicki, "Traffic Sensitive Active Queue Management for Improved Multimedia Streaming," in *Proceedings of the International Workshop on QoS in Multiservice IP Networks (QoS-IP)*, (Milano, Italy), February 2003. 49

[117] Q. Wu and C. Williamson, "Context-aware TCP/IP," in *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modling of computer systems*, (Marina Del Rey, CA, USA), June 2002. 50

[118] A. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," in *Proceedings of the 6th Passive and Active Measurement Workshop (PAM)*, March/April 2005. 54

[119] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pp. 229–240, 2005. 55

[120] K. C. Claffy, *Internet Traffic Characterization*. PhD thesis, Computer Science Department at University of California at San Diego, 1997. 56

[121] B. Krishnamurthy and J. Rexford, "Web Protocol and Practice." Chapter 10: Web Workload Characterization, Addison-Wesley, 2001. 56

[122] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming Media Workload," in *Proceedings of USNIX Symposium on Internet Technologies and Systems*, (San Francisco, CA), March 2001. 56

[123] P. Barford, J. Kline, D. Plonka, and A. Ron, " A Signal Analysis of Network Traffic Anomalies," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW)*, (Marseille, France), Novemeber 2002. 56

[124] N. Williams, S. Zander, and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification," *SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006. 56

[125] B. Bensaou, D. H. K. Tsang, and K. T. Chan, "Credit-based fair queueing (CBFQ): a simple service-scheduling algorithm for packet-switched networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, pp. 591–604, October 2001. 57, 67, 156

[126] Y. Liu, S. Gruhl, and E. W. Knightly, "WCFQ: an opportunistic wireless scheduler with statistical fairness bounds," *IEEE Transactions onWireless Communications*, vol. 2, pp. 1017–1028, September 2003. 57, 58, 67, 156

[127] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of IEEE INFOCOM '94*, (Toronto, Ont., Canada), 1994. 57

[128] I. Stoica, S. Shenker, and H. Zhang, "*Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proceedings of ACM SIGCOMM Conference*, (Vancouver, British Columbia, Canada), pp. 118 – 130, Sept. 1998. 57

[129] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings of ACM SIGCOMM Conference*, (Boston, MA, USA), pp. 231 – 243, Sept. 1995. 57

[130] Y. Jiang, C.-K. Tham, and C.-C. Ko, "A probabilistic priority scheduling discipline for multi-service networks," in *IEEE ISCC*, 2001. 77

[131] Y. Jiang, C.-K. Tham, and C.-C. Ko, "Delay analysis of a probabilistic priority discipline," Nov./Dec. 2007. 77, 84

[132] L. Kleinrock, *Queueing Systems*, vol. 2. Wiley-Interscience, first ed., 1976. 78

[133] J. J. Lee and M. Gupta, "White Paper: A New Traffic Model for Current User Web Browsing Behavior," 2007. Online at http://blogs.intel.com/research/HTTPr.pdf. 90, 97

[134] H. Wu, M. Claypool, and R. Kinicki, "Guidelines for Selecting Practical MPEG Group of Pictures," in *Proceedings of IASTED International Conference on Internet and Multimedia Systems and Applications (EuroIMSA)*, (Innsbruck, Austria), February 2006. 96

[135] F. Li, M. Li, R. Lu, H. Wu, M. Claypool, and R. Kinicki, "Tools and Techniques for Measurement of IEEE 802.11 Wireless Networks," in *Proceedings of the Second International Workshop on Wireless Network Measurement (WiNMee)*, (Boston, MA, USA), 2006. 107

[136] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control," 1999. 130