Treatment-Based Classification in Residential Wireless Access Points

by

Feng Li

A Dissertation
Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

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

in
Computer Science

by

May 2014

APPROVED:

Professor Mark Claypool Advisor	Professor Robert Kinicki Co-advisor
Professor Craig Wills Committee Member Head of Department	Doctor Timothy Strayer External Committee Member BBN Technologies

Abstract

IEEE 802.11 wireless access points (APs) act as the central communication hub inside homes, connecting all networked devices to the Internet. Home users run a variety of network applications with diverse Quality-of-Service requirements (QoS) through their APs. However, wireless APs are often the bottleneck in residential networks as broadband connection speeds keep increasing. Because of the lack of QoS support and complicated configuration procedures in most off-the-shelf APs, users can experience QoS degradation with their wireless networks, especially when multiple applications are running concurrently.

This dissertation presents CATNAP, *Classification And Treatment iN an AP*, to provide better QoS support for various applications over residential wireless networks, especially timely delivery for real-time applications and high throughput for download-based applications. CATNAP consists of three major components: supporting functions, classifiers, and treatment modules. The supporting functions collect necessary flow level statistics and feed it into the CATNAP classifiers. Then, the CATNAP classifiers categorize flows along three-dimensions: response-based/non-response-based, interactive/non-interactive, and greedy/non-greedy. Each CATNAP traffic category can be directly mapped to one of the following treatments: push/delay, limited advertised window size/drop, and reserve bandwidth. Based on the classification results, the CATNAP treatment module automatically applies the treatment policy to provide better QoS support.

CATNAP is implemented with the NS network simulator, and evaluated against DropTail and Strict Priority Queue (SPQ) under various network and traffic conditions. In most simulation cases, CATNAP provides better QoS supports than DropTail: it lowers queuing delay for multimedia applications such as VoIP, games and video, fairly treats FTP flows with various round trip times, and is even functional when misbehaving UDP traffic is present. Unlike current QoS methods, CATNAP is a plug-and-play solution, automatically classifying and treating flows without any user configuration, or any modification to end hosts or applications.

Acknowledgments

I cannot express enough thanks to my committee for their continued support and encouragement: Prof. Mark Claypool, Prof. Robert Kinicki, Prof. Craig Wills, and Dr. Timothy Strayer. Especially, I would like to express my deep gratitude to my advisors, Prof. Robert Kinicki and Prof. Mark Claypool, for their advise and encouragement throughout my graduate studies. They inspired my interests in network research and guided me to move the right direction. Without their consistent supports, the dissertation could never be completed. I offer my sincere appreciation for the learning opportunities provided by them.

My deep thanks also go to the members in my committee, Prof. Craig Wills and Dr. Timothy Strayer from BBN Network Technologies, who provided valuable feedback and suggestions to my comprehensive exam, my dissertation proposal and dissertation drafts.

I would also like to thank all my friends, Mingzhe Li, Mingrui Wei, Jae Chung, Huahui Wu, Hao Shang and Songxiang Gu, They provide many supports and inspiring discussions related to this research. Thanks to my parents and my parents in law, for their love and encouragement during my graduate studies in WPI. To Jayden - thank you for allowing me time away from you to research and write.

Finally, to my caring, loving, and supportive wife, Xiaoying Chen: my deepest gratitude. Your encouragement when the times got rough are much appreciated and duly noted. It was a great comfort and relief to know that you were willing to provide management of our household activities and spend countless times to take care of Jayden in past two years while I completed my work. My heartfelt thanks.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The decrease in price of IEEE 802.11 wireless network devices and the increase in wireless link capacities have significantly propelled the deployment of IEEE 802.11 wireless networks in residential places. The promise of up to 72 Mbps capacity[1] from a wireless AP means that users now expect to see applications such as streaming video and on-line games that have tight Quality of Service (QoS) requirements running seamlessly over residential wireless networks. However, since most QoS traffic shaping technologies are primarily designed for wired networks, even applications provided with QoS support may perform poorly when running in a residence with a last mile wireless network. Meanwhile, with the growth in adoption of broadband access technologies such as Asymmetric Digital Subscriber Loop (ADSL) and Cable Internet, multiple network applications are able to run concurrently over residential networks. Thus, multiple computers in the same residence are competing for network resources such as wireless medium access and up-link or down-link Internet bandwidth.

As Figure 1.1 shows, the wireless access point (AP) serves as a central communication point for all computation and entertainment devices acting as a gateway between client devices inside the house and the outside world (Internet). Residential class wireless APs are integrated with a fast Ethernet switch to provide wired access as well. Thus, most residential-based wireless APs perform the function of a router enhanced with additional features such as firewalls and name services.

[1] IEEE 802.11n can achieve 72 Mbps on a single 20 MHz channel with one antenna [15, 16].

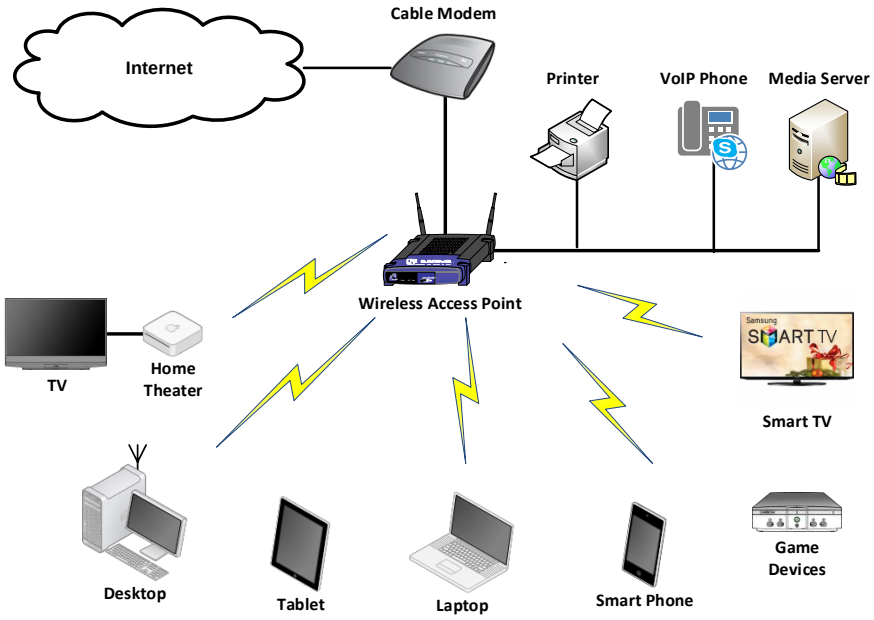


Figure 1.1: Typical Residential Wireless Network

Because of the rapid increase of the broadband connection capacity to homes, the bottleneck is often the wireless link.

Unfortunately, implementations of wireless APs do not differentiate among the various traffic types over residential networks, treating applications with different quality of service (QoS) requirements the same. Contention for wireless resources can degrade the performance for one or more wireless applications. Residential wireless users often experience degraded QoS for on-line games, real-time videos, or Voice over IP (VoIP) when another application is simultaneously downloading a file through the same AP [17–19].

Although various mechanisms are proposed to provide better QoS support, such as IEEE 802.11e, Integrated Services (IntServ), or Differentiated Services (DiffServ), QoS is yet to be widely deployed. One of the key factors holding back broad deployment of QoS is the lack of suitable classification methodologies to map different application flow types into different QoS classes. Because network devices have limited ability to detect what kinds of applications are running over the networks, they seldom have a chance to shape traffic to provide better QoS support. In the past

decade, port-based traffic identification approaches, which identify an application by inspecting transport layer port numbers, have been effective and widely used. However, recent studies show that this kind of classification approach is error-prone [7, 20–22] because the applications and users often intentionally avoid using well-known or consistent ports. To achieve higher accuracy, packet payloads can be inspected [7]. However, this approach is not practical for real-time classification because of its high computational complexity and ineffectiveness for encrypted packet payloads.

Recently, researchers have proposed several statistics-based traffic classification methods using machine learning techniques [23–28]. These approaches assume that applications typically transfer data in a repeatable manner, and that this pattern can be used as a means of identification [28]. To find these patterns, flow statistics (such as mean packet length, flow length and packet inter-arrival time) are needed. These flow features can be collected from TCP/IP headers, thus avoiding the use of packet payload information. However, these approaches are designed for wired environments, and most do not consider wireless medium characteristics such as retransmission or rate adaptation which can impact flow patterns such as packet inter-arrival timing. Also, typically the main objective of these statistical approaches is to identify the application type without improving QoS support over networks.

Last but not least, another challenge is from the wireless APs themselves. Wireless APs in residential networks are low end devices with limited CPU and memory, making it difficult to deploy classification methods and traffic shaping schemes with high computational complexity. For example, the 5th generation Airport Extreme IEEE 802.11n wireless router from Apple Inc, which debuted in 2013, does not provide QoS support because Apple believes it is difficult for home users to correctly configure QoS settings [29]. Even the Cisco Wireless LAN controller 5500 series, a dominant product for the U.S. enterprise class wireless device market, does not support QoS in a convenient way: it needs 14 steps to add per-user capacity limitations. Therefore, most of residential wiress APs do not provide QoS support, or only support QoS with complicated manual configuration.

In summary, although neither traffic classification nor traffic treatment is a novel technique, no effort has been made to combine these two techniques together onto residential wireless APs to automatically improve QoS support for applications over residential wireless networks. This dissertation presents a traffic classification technique which identifies and classifies applications based on their wireless network traffic characteristics, and enhances the wireless AP with new

traffic treatment techniques to shape the traffic by mitigating the effects of wireless media [30].

1.2 Bottleneck on Residential Wireless Networks

There exists a long debate which device is the bottleneck on residential wireless networks: "last-mile" connection or wireless AP. In the last decade, the IEEE 802.11 (a/b/g/n) wireless protocol provides up to 260 Mbps link speeds. However, in a well controlled environment, IEEE 802.11 protocols never reach their claimed link speed. For example, IEEE 802.11g only provides 19 Mbps throughput, much smaller than the maximum link speed of 54 Mbps [15].

Under real wireless environments, wireless devices seldom reach their claimed link speed because of the shared medium nature of wireless. Specifically, the following could degrade the performance of IEEE 802.11 devices over residential places:

1. Neighbors' APs can act as hidden terminals, especially in a thickly settled urban area. Both the home wireless measurement study conducted by a group of researchers from University of Wisconsin [31] and our home wireless measurement study (Chapter 4) observed more than 10 neighbor APs in an urban area. The high density of neighbor APs can degrade the performance of wireless APs.

2. IEEE 802.11 b/g can degrade the performance of IEEE 802.11n wireless devices. IEEE 802.11 provides non-HT (High Throughput) mode and HT mixed mode to provide backward compatibility to IEEE 802.11 b/g devices. Non-HT mode makes IEEE 802.11n devices act as IEEE 802.11 b/g devices, and there are no performance gains from IEEE 802.11n technology. The HT mixed mode requires a device to transmit legacy formats of CTS-to-self or RTS/CTS frames. Although these CTS/RTS frames are short, these small frames still degrade the performance of IEEE 802.11n devices. However, due to cost concerns, IEEE 802.11 b/g devices are still produced, for example, for disposable wireless sensors since the lower data rate can reach a greater distance.

3. Non IEEE 802.11 devices, e.g., microwave ovens and bluetooth devices, can interference with IEEE 802.11 b/g/n devices especially on the 2.4 GHz band.

4. The 5 GHz band does not work well for residential networks. Although the 5 GHz band provides more available channels, it has more trouble penetrating solid objects because of its

relatively small wave length. Thus, the emerging IEEE 802.11ac standard may not work well in an indoor environment since it is on the 5 GHz band.

For the above reasons, IEEE 802.11 a/b/g/n devices never reach their claimed link speeds for indoor environments.

Table 1.1 lists the top 10 countries with the highest ISP connection speeds [1]. It shows that the top 4 countries have more than 50 Mbps peak connections to the home, which is much faster than the achievable throughput of IEEE 802.11g. IEEE 802.11 b/g and IEEE 802.11n in HT mixed mode would be the bottleneck devices for these homes. Three states: Massachusetts, New Jersey and Maryland provide 50 Mbps peak connections to residences [1], and IEEE 802.11 b/g would be the bottleneck in these three states.

Table 1.1: Top 10 countries with Peak Connection Speeds [1]

Country/Region	Mbps
1 Hongkong	65.4
2 South Korea	63.6
3 Japan	52.0
4 Singapore	50.1
5 Israel	47.7
6 Romania	45.5
7 Latvia	43.1
8 Taiwan	42.7
9 Netherlands	39.6
10 Belgium	38.5
.....	....
16 United States	36.3

Because IEEE 802.11 a/b/g/n APs could not reach their theoretical throughput as noted previously, IEEE 802.11 is the bottleneck for many residential networks in the United States.

1.3 The Dissertation

The goal of the dissertation is to improve QoS support for various applications over residential wireless networks by implementing treatment-based classification methods and corresponding treatments on a home AP.

Figure 1.2 shows the architecture of the Classification And Treatment On an Access Point (CATNAP), which automatically differentiates the traffic of flows into eight categories, and treats

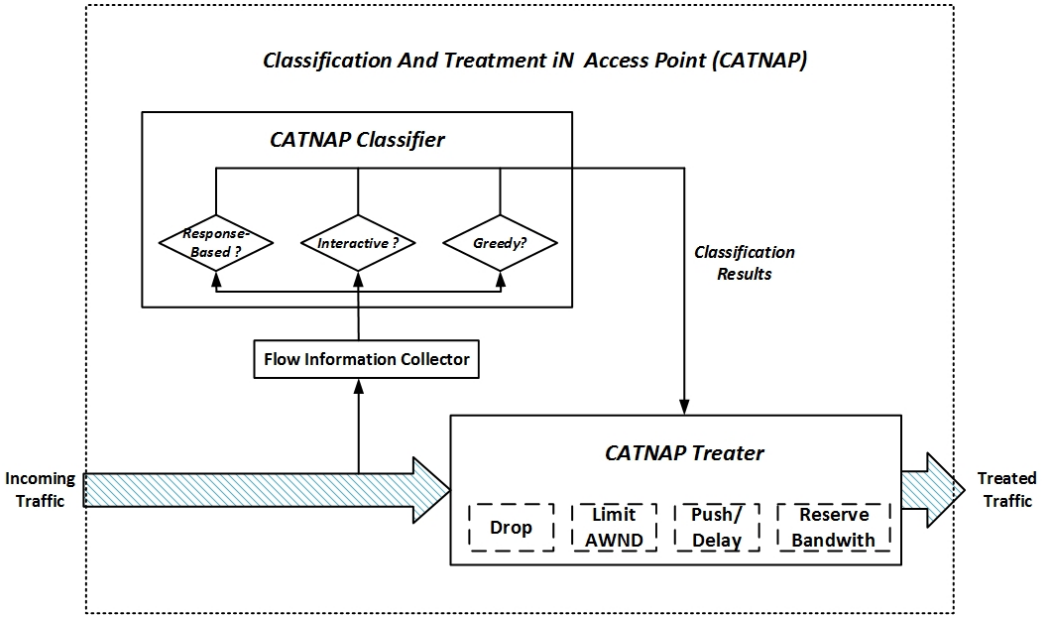


Figure 1.2: Architecture of CATNAP

the traffic to improve their QoS over residential wireless links. This study assumes the last mile wireless link is the bottleneck connection and that the AP serves as an edge router.

Due to the closed nature of off-the-shelf residential wireless APs, the study builds CATNAP with the NS-2 network simulator and validates it with various traffic loads and network configurations. CATNAP consists of two major components: the classifier and the treater. Because wireless APs are devices with limited-powered processor and memory, the CATNAP flow treater uses four non CPU-intensive operations: *push, delay, drop* and *limit* to help flows meet their QoS requirements. For instance, CATNAP attempts to help interactive and real-time applications by pushing them in front of the AP queue without significantly deceasing the throughput of bulk-download applications by explicitly reducing their TCP advertised window instead of dropping packets.

The CATNAP classifier first places the flows into different categories in real-time. Unlike other statistics-based classification techniques, the CATNAP classifier does not associate a flow with a particular application but instead places it into one of the treatment categories according to the nature of the traffic. Therefore, it is not necessary to train the traffic classifier with historical data. Meanwhile, current widely used classification techniques such as port-based or payload-based classifiers exhibit a number of shortfalls such as unable to detect flows running on Web ports. The CATNAP classifier attempts to identify traffic based on payload-independent features, including

reverse traffic and packet length.

In addition to the classification module, the CATNAP treater applies one or more treatments to the classified flows. The goals of the flow treater are to provide better QoS for real-time and interactive applications (lower delay) when the wireless link is stressed by greedy bulk-downloading applications. The flow treater takes advantage of the CATNAP classification results, and treats the flow according to their nature.

To evaluate CATNAP, we analyze application behaviors from real network traces as well as through a wireless simulator, and generalize flow behavior patterns for several representative applications such as online gaming and video streaming running over residential wireless networks. Our treatment-based classification criteria categorizes flows with similar features and QoS requirements into the same treatment categories. We evaluate our classifier with simulated application flows under various wireless network conditions.

Moreover, we conduct a series of measurement studies to investigate the current status of home wireless environment, especially to get the performance of video, game and VoIP applications under normal network activities. In addition, through the home wireless measurement study, we are able to validate our simulation setup with real home wireless environments and provide a set of realistic simulation settings for future home wireless network simulations.

1.4 Contributions

The main contributions of this dissertation are to design a set of novel treatment-based classification methods and corresponding treatments to automatically improve the Quality of Service (QoS) of various applications running over residential networks, and verify it through carefully designed simulations. The specific contributions of this dissertation include:

- Design CATNAP which automatically improves QoS support for various applications running on home wireless networks. without any modifications on applications or network protocols. Different from IEEE 802.11e based approaches, which requires applications set the Type of Service (TOS) bits, CATNAP provides automatically flow level classifications and maps the classified flows with corresponding treatment methods. Different from other statistics based flow classification methods, the CATNAP classifiers do not need the training data set. Thus, CATNAP has the potential to detect and treat any unknown type flows in future, if the new

flows have similar characteristics or QoS requirement with the current applications.

- Verify the CATNAP by thorough simulation studies. CATNAP has been evaluated under various network and wireless conditions with a carefully designed simulation setup for home wireless environment. Moreover, we carefully study the behaviors of representative applications found in public available traces, and enhance the NS-2 simulator with realistic traffic generator to provide "close to real" applications. The simulation results are a good starting point for future researchers who want to build a more accurate application behavior model over residential wireless networks.

- Improvement of wireless AP QoS by making a "smarter" AP. QoS packet scheduling algorithms are usually designed for core router devices, and most of them assumes that applications correctly set the Type of Service (TOS) bits. The CATNAP approach attempts to improve the QoS of various applications running over residential wireless networks without any changes in end hosts, applications, or network protocols. Thus, by enhancing the wireless AP, CATNAP is a promising approach to improve QoS without any user interventions or configurations.

- Design and conducted a home wireless measurement study in New England area. The home wireless study was the first attempt to study provides a complete snapshot of home wireless network usage from physical to application layer. Through this wireless measurement study, we have better understanding of the usage of home wireless networks, and developed a set of wireless measurement tools which can be used for future wireless measurement study. Moreover, the knowledge and pitfalls gained in this study could help future researchers to design home wireless measurement study.

1.5 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 classification and QoS support over wireless links; Chapter 4 presents results of the residential wireless network measurement case study to understand the current usage of home wireless networks as well as validate the physical layer and MAC layer settings for our NS-2 simulation; Chapter 6 compares

the simulation results between CATNAP and its competitors: DropTail and Strict Priority Queue (SPQ); and Chapter 7 and Chapter 8 summarize the future work and conclusion respectively.

Chapter 2

Background

This chapter reviews the fundamental techniques and terminologies that are referred to in the dissertation. Section 2.1 overviews QoS requirements for various applications running over residential networks. Section 2.2 reviews wireless network techniques, such as wireless medium characteristics, IEEE 802.11, and wireless access points (APs). Section 2.3 surveys several potential technologies which could provide QoS of various applications over residential wireless networks, including IEEE 802.11e, IP QoS and others.

2.1 QoS Requirements of Applications

The growth in the popularity and capacity of the Internet has led to an increasing diverse set of application with varying network behaviors and requirements running over residential networks. In order to design a *smart* AP to provide better concurrent QoS supports for residential applications, application Quality of Service (QoS) requirements must be ascertained, particularly along the key network metrics of *delay*, *loss* and *throughput*.

2.1.1 Network Layer Quality Metrics

Although the ultimate QoS consideration is "user-perceived quality", for applications running over networks, it is necessary to study the network-based application quality metrics. Knowing the network service required for a particular application can help to avoid both over- and under-allocation of network resources. For applications running over networks, three IP layer quality metrics, *delay,*

loss and *throughput* are widely used:

- **Delay** is the time it takes for a packet of data to get from one designated point to another. For delay sensitive applications like VoIP, the delay should be low (less than 150 ms). Large delay values decrease the quality of such applications [32,33]. The dominant factor that affects delay on the Internet is buffering inside the network nodes along the flow path, such as at routers and switches. For IEEE 802.11 infrastructure networks, the dominant contributor to delay is the buffering inside the wireless AP. In our previous study [34], while the one-way delay observed between a wireless station and an AP is less than 10 ms, we have observed more than 200 ms delay for a queue capacity of 50 packets to 300 packets [35]. Related to delay, *jitter*, the variation in the inter-arrival time of adjacent packets [13], also can be used to measure application performance.

- **Loss** is the ratio of lost IP packets to those generated on the source node. Packets can be lost for several reasons in a wireless network in addition to congestion in a node along the communication path. For instance, collisions, interference, noise, fading, or location-based errors in a communication link [13] can also cause packet loss. However, link losses in IEEE 802.11 are often hidden by the MAC layer's automatic retransmission mechanism. Thus, the final IP layer packet loss fraction might be low, despite high wireless frame error rates.

- **Throughput** is the amount of digital data per time unit that is delivered over a physical or logical link. However, throughput is a metric limited by every component along the path from the source node to the destination node, including all hardware and software. Thus, *maximum throughput* and *achievable throughput* describe two different characteristics of throughput:

  - *Maximum Throughput (T_{max})* is the highest transfer rate than can be successfully performed between two end hosts if they are connected with each other directly [36]. This characteristic is software independent and describes the ceiling of throughput for a given network configuration.

  - *Achievable Throughput (A_t)* is the throughput between two end points under a complete given set of conditions, such as transport protocol used, end host hardware, round-trip time, operating system, tuning method and parameters, etc. [36,37]. This characteristic represents the performance that an application in this specific setting might achieve.

Throughput Sensitivity

For bulk download applications like FTP, achievable throughput tries to be as close as possible to the maximum throughput, because of the "best-effort" nature of TCP/IP protocols. For the other applications such as IP phone applications (VoIP applications), their achievable throughput (64 Kbps) is typically limited by the application itself. Moreover, for streaming video applications, their achievable throughput is limited by the maximum throughput or the video encoded bit rate, whatever is less.

Thus, we define the ratio of achievable throughput and maximum throughput as the *greedy indicator* G to describe the bandwidth requirement nature for applications. For a given application A, its greedy indicator G is defined as:

$$G(A) = \frac{A_t}{T_{max}} = c, 0 \leq c \leq 1.$$

where T_{max} is maximum throughput over a link. The greedy indicator of a given application is a constant c between 0 and 1.

If the bandwidth increases to infinity, the achievable throughput (A_t) of some applications such as FTP also increases to infinity. But for other applications such as video streaming, their achievable throughput is limited by the application's parameters such as video encoding bit rates. Thus, when the maximum throughput is increased to infinity, the greedy indicator would be:

$$\lim_{T_{max}\to\infty} G(A) = \lim_{T_{max}\to\infty} \frac{A_t}{T_{max}} = \begin{cases} 1 & : \quad A \text{ is a bulk download application.} \\ 0 & : \quad A\text{'s transmission rate is limited by itself.} \end{cases}$$

where T_{max} is maximum throughput over a link. As the above equation shows, when $T_{max} \to \infty$, the greedy indicator of bulk download applications like FTP equals to 1, and for applications like streaming or gaming, their greedy indicator is 0.

Here, based on the greedy indicator G, we define the *throughput sensitivity* for applications. It describes the application behavior as the maximum throughput changes. *Throughput tolerant* applications have a greedy indicator typically a constant c where $0 \leq c < 1$ for a given T_{max}. It means that its achievable throughput is limited by the application itself even when the maximum throughput is infinity. For example, typical videos from youtube.com only consume a modest amount of bandwidth typically (less than 792 Kbps). Compared to the maximum throughput

on IEEE 802.11g network (22 Mbps), the greedy indicator (G) for a streaming application from youtube.com is a small fraction. On the other hand, for *throughput sensitive applications*, the achievable throughput grows as the maximum throughput increases, and its greedy indicator (G) would be 1 even when maximum throughput T_{max} increases.

2.1.2 Application QoS Requirements

Based on the three major performance metrics: *delay, loss* and *throughput*, applications can be grouped into eight different groups with distinct QoS requirements. As Figure 2.1 shows, applications can be divided as delay sensitive or tolerant, loss sensitive or tolerant, throughput sensitive or tolerant. Since each metric is independent of the others, it results in eight different categories for applications.

- *Throughput tolerant, packet loss sensitive*, and *delay tolerant.* Emails traveling over the Internet can be put into this category. The response-time requirement for an Email can be on the order of tens of minutes. Traditional Email applications can tolerate the end-to-end delay and only consume a small fraction of bandwidth unless containing a large attachment. Similarly, reading USENET News can be also placed into this category.

- *Throughput tolerant, packet loss sensitive*, and *delay sensitive.* Traditional Web traffic can be classified into this category. Compared to File transfer (FTP) and Email, Web browsing is sensitive to HTTP server response time, which is affected by both the end-to-end delay and packet loss rate on the network [14]. Moreover, telnet (ssh and remote login), DNS query, on-line gaming and on-line chatting applications can also be placed into this category.

- *Throughput sensitive, packet loss tolerant*, and *delay tolerant.* Only some special purpose applications can be put into this category, for example, a Video Crawler [38], a robotic program retrieving video information from the Web, consuming a large fraction of bandwidth, but tolerates packet loss and delay. Also, some malicious applications[1] including viruses can be categorized into this category. However, most applications running over residential networks would not be in this category.

- *Throughput sensitive, packet loss tolerant*, and *delay sensitive.* High-definition (HD) video streaming can be put into this category. Unlike most video clips stored on line, studio quality

[1] Some programs like SYN Flooding attack virus.

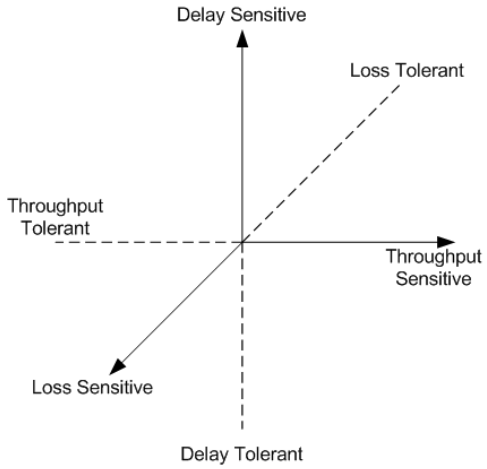


Figure 2.1: The QoS Spectrum of Applications

videos are encoded at about 36 Mbps, and HDTV quality videos stream at about 25-34 Mbps. Therefore, HD video streaming requires high throughput, and low delay (jitter) but can tolerate some packet loss.

- *Throughput sensitive, packet loss sensitive*, and *delay tolerant.* File transfer (FTP) and Peer-to-Peer (P2P) file sharing programs are placed in this category. FTP can consume a large portion of bandwidth, but the response time requirement of FTP is on the order of minutes while P2P file downloading programs often run in background over several nights.

- *Throughput sensitive, packet loss sensitive*, and *delay sensitive.* Some interactive video games (First Person Shooter) and interactive streaming programs are placed in this category. This kind of application can consume a large fraction of bandwidth and is less tolerant of packet loss and delay since the data flow contains loss sensitive control information along with video data.

- *Throughput tolerant, packet loss tolerant*, and *delay sensitive.* Typical online streaming traffic can be put in this category. Nearly 70% of videos are targeted for broadband with encoded bit rates between 56 Kbps and 768 Kbps [38]. Thus, these streaming video clips do not require high bandwidth. Meanwhile, these applications can tolerate some kind of data loss, since a little data loss can be repaired without noticeably degrading the user perceived quality [39]. However, their performance is sensitive to delay changes (jitter) [2] which can trigger

[2]Playout buffers can significantly reduce effects of jitters.

re-buffering events [37].

- *Throughput tolerant, packet tolerant*, and *delay tolerant.* This kind of applications can tolerate loss, delay and insufficient bandwidth. It is difficult to find such application instances running over residential networks.

The method to categorize applications by the three QoS metrics is a complete classification in that any flow over the Internet can be classified into one of the eight categories. However, a single kind of application might have different QoS requirements depending upon its usage and be mapped into different categories. For example, the non-interactive streaming application can be mapped into two different categories: "throughput tolerant, packet loss tolerant, and delay sensitive" for most Web based streaming applications; "throughput sensitive, packet loss tolerant, and delay sensitive" for HD video streaming applications. They might even use the same streaming software, but the flows carrying video content have different QoS requirements.

Table 2.1 [14] summaries QoS characteristics for common applications running over residential networks. We also list *duration* and *volume* as an application characteristic in addition to the three QoS metric we discussed above. *Duration* is not a QoS related-metric in the general sense, but indicates the time window for treating an application. Short-lived flows have less possibility to be treated than long-lived flows. Meanwhile, the volume of a flow could also be used to characterize applications. Interactive Web traffic and FTP are both "best effort" applications, but the volume of their flows is different. Web traffic is small, whereas the FTP flows have large volume of data [40]. From the view of treatment, during congestion, reducing the date volume of flows with large data volume might be more effective. Table 2.1 also lists the "traffic ratio" for major kinds of applications which is the data volume over the total captured volume. Because there is no publicly available traffic analysis for residential wireless networks, we use traffic analysis results from [41], which is an eleven-week trace collected over a campus wireless network in 2001. The authors in [41] identify the traffic type based on protocol and port used. Therefore, some traffic like file sharing over HTTP protocol is also classified as HTTP traffic. Moreover, in [41], there are 5.7 GB (2.8%) traffic from IM software which has similar QoS requirements as Telnet applications. Yet, around 5.3 GB (2.4%) traffic is classified as unknown since neither the source port nor the destination port matches on the authors' port list.

Table 2.1: Internet Applications: QoS Requirements and Characteristics.

Internet Applications	Sensitivity to			Characteristics		Traffic Ratio
	Delay	Loss	Tput	Duration	Volume	Bytes, Ratio
FTP (TCP)	low	low	high	long	large	3.6 GB, 1.6%
Email (TCP)	low	low	low	short	small	2.9 GB, 1.3%
Telnet (TCP)	high	high	low	long	small	N/A
HTTP (TCP)	low	high	med	short/med	small/med	116 GB, 53%
VOD (UDP)	low	med	high	long	large	N/A
A/V Phone (UDP)	high	med	high	long	small/med	N/A
Video Game (UDP)	high	med	high	long	small	N/A
DNS (UDP)	high	high	low	very short	tiny	N/A

2.2 IEEE 802.11 Wireless Networks

Because of its mobility and flexibility, wireless networks have been widely deployed with fixed infrastructure networks in residential places. Most protocols and applications developed for wired networks have been transferred to wireless networks directly. However, some characteristics of wireless media that differ from wired media might impact the performance of applications over wireless. This section reviews characteristics of wireless media, the IEEE 802.11 Wireless Local Area Networks (WLANs) standard, and additional challenges for wireless QoS.

2.2.1 Characteristics of Wireless Media

The important characteristics of wireless radio medium that differ from wired networks are:

1. **Shared Medium.** Unlike the wired medium, the broadcast nature of wireless means that all transmissions share the same medium at half-duplex. Furthermore, the shared medium causes collisions and interference that additionally degrade network performance. Finally, with wireless, the network is restricted to a limited available bandwidth for operation and cannot accommodate more capacity by adding new frequency range or duplicating the medium as in wired networks.

2. **Propagation.** Wireless transmissions experience attenuation, reflection, diffraction and scattering effects, and cause multi-path fading which results in radio signals reaching the destination by two or more paths. The effects of multi-path include constructive and destructive interference, which produce time varying channel conditions, such as varying received signal power, and eventually cause propagation loss. Especially in residential environments, indoor

propagation is dependent on the building layout, the construction material used and the building type, making it more difficult to model propagation and predict loss [42, 43].

3. **Bursty channel errors.** Wireless channels are error prone with a higher bit error rate (BER) of 10^{-3} or more. Such high bit error rates can be caused by the multi-path, transmission interference, signal fading and other issues such as microwave ovens or thunderstorms.

4. **Location dependent carrier sensing.** In wireless LANs as well as other kinds of wireless links, wireless performance is significantly dependent on the location of the wireless stations relative to its neighboring wireless nodes. For example, hidden terminals and exposed terminals [4] could considerably degrade wireless performance. A hidden terminal is a wireless station which is within the range of the intended destination but out of the range of the sender. Thus, collisions may happen at the destination if the sender and the hidden terminal transmit at the same time because they can not detect each other. On the other hand, an exposed terminal is one wireless station that is within the range of the sender but out of interference range of the destination. The sender may mistakenly back-off when the exposed terminal is transmitting even if that transmission will not collide with the sender's transmission. Either hidden terminal and exposed terminal problems cause performance degradations. This location dependent issue might be worse in crowded residential places.

2.2.2 IEEE 802.11 Overview

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 Hopped Spread Spectrum (FHSS), Orthogonal Frequency Division Multiplexing (OFDM) and Infrared (IR). The IEEE 802.11 Standard [2, 3] defines a family of Wireless Local Area Networks (WLANs), including 802.11a, 802.11b, 802.11g, and others [2, 3]. A brief comparison of the main standards is given in Table 2.2. Note that all the standards use the same MAC layer specification, but with different physical layer specifications.

IEEE 802.11 standards [2] support both an infrastructure network topology or ad-hoc network topology. In an infrastructure network, there is a fixed infrastructure that supports communication between mobile stations and fixed stations via an Access Point (AP). Conversely, in an ad-hoc

Table 2.2: IEEE 802.11a, b, and g WLAN Standards [2–4]

	802.11 list	802.11a	802.11b	802.11g
Frequency	2.4 GHz	5 GHz	2.4 GHz	2.4 GHz
Date Rate(s) (in Mbps)	1, 2	5, 9, 12, 18, 24, 36, 48, 54	1, 2, 5.5, 11	6,9, 12, 15 24, 36, 48, 54
Modulation	FHSS, DSSS	OFDM	DSSS	OFDM
Advertised Range	300 ft	225 ft	300 ft	300 ft
Encryption type	40 bit RC4	40 bit, 104 bit RC4	40 bit, 104 bit RC4	40 bit, 104 bit RC4
Network Support	Ethernet (IEEE 802.3)	Ethernet (IEEE 802.3)	Ethernet (IEEE 802.3)	Ethernet (IEEE 802.3)

network, there is no fixed infrastructure. The mobile stations communicate directly with each other without the use of an AP. Recently, IEEE 802.11 mesh networks appear in several communities [44, 45], which is a hybrid ad-hoc/infrastructure structure. However, this study focuses on IEEE 802.11 infrastructure networks with a single AP. Ad-hoc networking, mesh network or infrastructure network with multiple APs typically is not part of a residence network, thus they are out of the scope of this dissertation and will not be covered in this review.

The main purpose of this study is to classify the traffic over residential wireless networks into different categories according to their QoS requirements with same traffic shaping techniques. In particular, the packet inter-arrival time for different flows impacts traffic detection and treatment selection. Therefore, we focus on the IEEE 802.11 techniques which directly affect packet inter-arrival times. These techniques include IEEE 802.11 Distributed Coordination Function (DCF), frame retransmission and multi-rate physical layer.

IEEE 802.11 Distributed Coordination Function (DCF)

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 [3]. PCF is a centralized MAC 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 [3, 46]. Therefore, in this dissertation, we limit our study to DCF only.

CSMA/CA Mechanism

DCF defines two techniques for frame transmission: the default two-way handshake, referred to as basic access, and an optional four-way handshake.

Figure 2.2 depicts the medium access procedure for a wireless station (WS) in a wireless infrastructure network that wants to transfer a frame with the basic access mechanism. A station that wants to access the channel performs carrier sense on the medium before initiating the transmission of a new frame. If the channel is idle for a distributed inter-frame space (DIFS), which is a time period a station has to wait before sensing the medium again, the frame is transmitted. The receiver of the data (AP in this case) waits a Short Inter Frame Spacing (SIFS), which is a shorter time period than a DIFS, and replies with an acknowledgment (ACK). When a station has access to the channel, all other stations defer access.

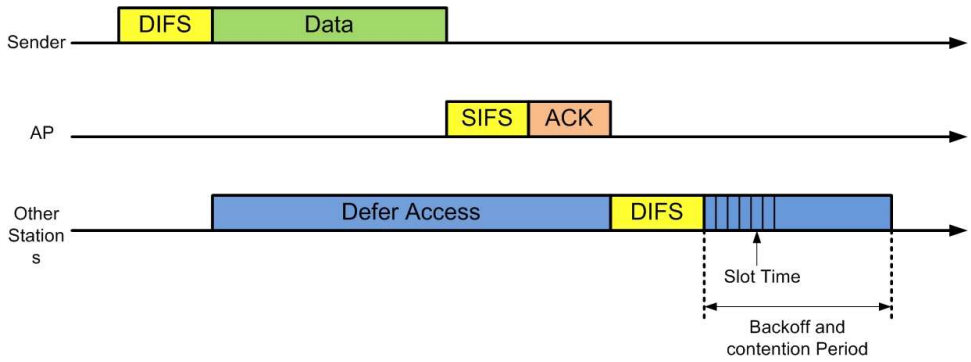


Figure 2.2: Medium Access Control Procedure Using DCF Basic Access Mechanism.

Otherwise, if the sender senses the channel is busy, it has to wait a DIFS and enter the contention period. In this contention phase, each station waits a random number of back off time slots (9 usec) before the sender repeats the carrier sense. If the channel is idle for a DIFS after its back off time period, the sender repeats the procedure described above and transmits the data frame after a DIFS if the channel is still idle.

The back-off time period is calculated by an exponential back-off timer. The timer is decremented only when the medium is idle and it is frozen when the medium is sensed busy. The slot size of the backoff timer is denoted by the time needed by any node to detect the transmission of a packet by any other node, typically 9 micro seconds in IEEE 802.11g. At each frame transmission, the back-off time is uniformly chosen in the range $(0, CW - 1)$, where the CW is called the *contention window* (CW). For each packet queued for transmission, the contention window CW

starts from an initial value CW_{min} that doubles after each unsuccessful frame transmission, up to a maximum of CW_{max}. The contention window remains at CW_{max} for all remaining attempts. In addition, to avoid channel capture, a node must wait for a random back-off time between two consecutive frame transmissions even if the medium is sensed idle in the DIFS time [2].

The four-way handshake mechanism is used to mitigate the hidden terminal problem [3, 4]. The four-way handshake mechanism involves the usage of the request-to-send (RTS) and clear-to-send (CTS) control frames prior to the transmission of the actual data frame. For most of the residential wireless devices, RTS/CTS is turned off by default [35], as RTS/CTS schemes can significantly lower system throughput [3, 47]. Thus, we do not investigate the four-way hand shake mechanism in this dissertation[3].

MAC Layer Retransmission

The IEEE 802.11 DCF MAC layer retransmits DATA frames and RTS frames for a number of times based on the frame size. The IEEE 802.11 standard suggests that the transmission attempts for a frame with a size less than the RTS Threshold is seven, and for the frame with a size larger than the RTS Threshold is four. The RTS Threshold parameter is also used as an indicator of the usage of RTS/CTS mechanism. If the DATA frame is smaller than the RTS Threshold, the frame is considered as a short frame, and can be transmitted without the RTS/CTS exchanges. Moreover, if a station has an RTS Threshold value greater than the Maximum Transmission Unit (MTU), the RTS/CTS mechanism is simply disabled. Then all DATA frames are retransmitted following the short frame retry limit. In residential wireless networks, the maximum frame size is typically limited by the MTU size of 1500 bytes/packet, which is smaller than the default RTS threshold (2312 bytes). Therefore, generally, over a residential network, data frames are handled as short frames with up to seven retransmissions. Several researchers have tried to dynamically tune the retransmission threshold to provide better QoS support over wireless LANs [48, 49].

IEEE 802.11 Multi-rate Physical Layer

The IEEE 802.11 medium access protocols provide support for multi-rate physical layer modulations. For example, the Extended Rate PHY (ERP) of IEEE 802.11g supports payload data rates of 1, 2, 5.5, and 11 Mbit/s using DSSS modulations to be compatible with IEEE 802.11b devices, and

[3]The detailed transmission procedure for RTS/CTS could be found in [2, 3, 47].

additional payload data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbit/s using OFDM modulation.

The rate adaptation mechanisms are based on either the sender's inference or receiver's feedback of the current channel conditions. The adaptation schemes can be either SNR-based (few implementations), statistics-based, such as number of retransmissions, packet error rate (PER) or throughput based [50]. For instance, the scheme designed in [51] uses the statistical information on the sender retries and the scheme in [52] uses the SNR data feed back from the receiver.

For a given SNR, the modulation scheme with a higher data rate has a higher BER. By adapting the data rate with different modulation schemes under various wireless network conditions, a low BER, and therefore better performance, can be achieved. However, wireless rate adaptation results in dynamic link capacity changes in wireless networks. Such changes might impact the performance of rate-based applications, such as streaming and VoIP applications over wireless networks.

IEEE 802.11n

IEEE 802.11n builds on previous 802.11 standards by adding multiple-input multiple-out (MIMO) and 40 MHz channels to PHY layer, and enhancing the MAC layer with frame aggregation [53]. IEEE 802.11n becomes one implementation of IEEE 802.11ac [54]. IEEE 802.11n can achieve 600 Mbps data rate only with the maximum of four spatial streams using one 40 MHz-wide channel. IEEE 802.11n can only achieve 72 Mbps data rate on a single 20 MHz channel with one antenna and 400 ns guard interval; if there is no interference with other nearby Bluetooth, microwave or WiFi devices by using 40 MHz mode.

Figure 2.3 [10] shows the wireless channels in 2.4 GHz band. IEEE 802.11n only can have three non-overlapping 20 MHz channels same as IEEE 801.11g. Thus, it is impossible to guarantee OFDM operation thus affecting the number of possible overlapping channels and limiting the achievable throughput of IEEE 802.11n devices, especially when IEEE 802.11 b/g devices present. Zubow *et al.* shows that 802.11 is an inappropriate protocol for multi-channel MAC/routing protocols based on multi-radio systems [55] where an explicit MAC layer link-scheduling is a more promising approach to improve the performance of IEEE 802.11 network.

2.2.3 Wireless Access Points

In an IEEE 802.11 infrastructure network, a wireless access point (AP) is a device that connects wireless communication devices together to form a wireless network. The AP usually connects to

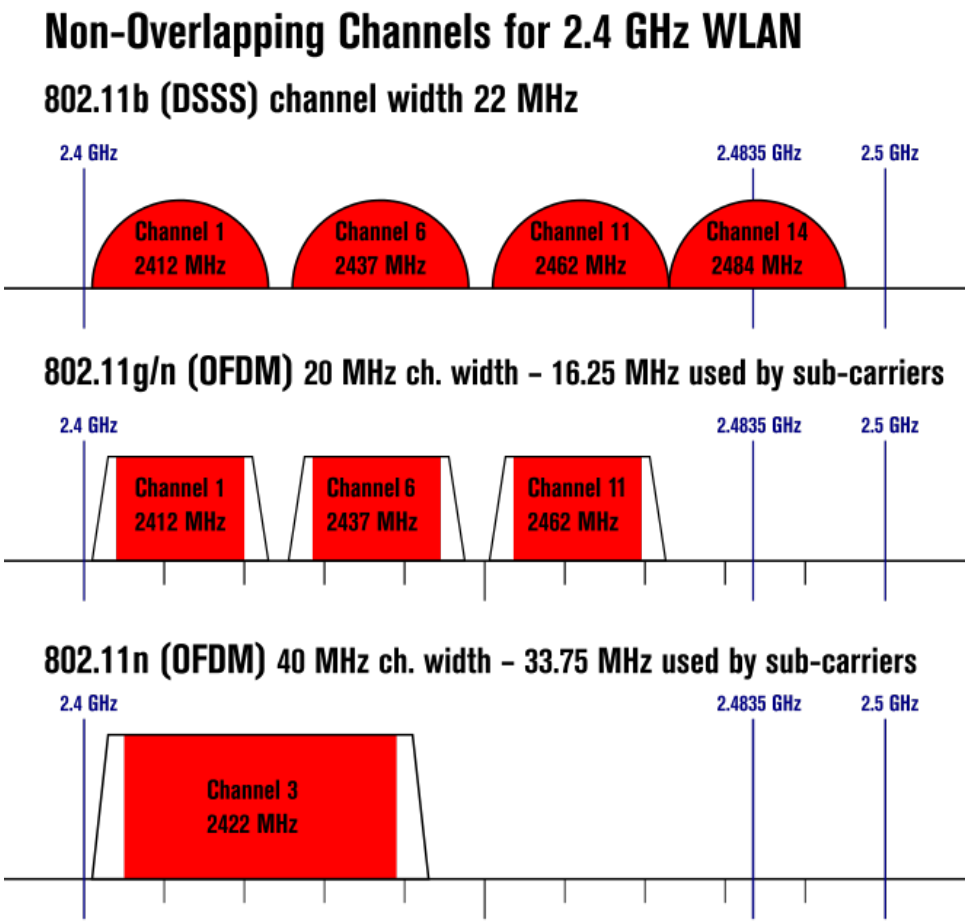


Figure 2.3: IEEE 802.11 Channels on 2.4 GHz Band [10].

a wired network and can relay data between wireless devices and wired devices. Several APs can link together to form a larger network that allows roaming. However, in this study, we focus on residential wireless networks which typically contain only one AP.

Over residential IEEE 802.11 wireless networks, there exist various wireless devices that provide wireless coverage in addition to the wireless stations. The off-the-shelf wireless APs play multiple roles in residential networks. Their functions can be categorized according to the OSI model:

- **MAC Layer Function**

  One of the MAC layer function of an AP is to form a bridge between the wireless client devices and the hard-wired network. An AP encapsulates IEEE 802.3 frames from wired link into the IEEE 802.11 frame format and sends them over a wireless link and vice versa. Moreover,

an AP provides medium access control function and other management functions (such as beacon frames) for wireless clients associated with it.

- **IP Layer Function**

  Most residential wireless APs in market are wireless routers. They are wireless APs enhanced with the functions of an IP router and an Ethernet switch. These wireless routers provide DHCP services, network address translation (NAT) and firewall services for the clients connected with it as well as extend its connectivity to wired devices inside the home. All wireless routers can be configured to work as traditional wireless APs by disabling the IP layer functions. However, in this study, we interchangeably use the term wireless AP and wireless router.

- **Application Layer Function**

  In reality, most residential wireless routers and access points act as *residential gateways* which are used to connect home networks with the Internet. These wireless residential gateways often combine the functions of an IP router, firewall, multi-port Ethernet switch and IEEE 802.11 access point, and even integrate with a DSL or Cable Modem in addition to providing Web interface for maintenance. These wireless gateways also act as wireless routers/access points.

It is widely accepted that the bottleneck of the connection is generally located at the edge routers, especially when the last hop is a wireless link. Because of the rapid increase of the bandwidth delivered to residence places, we can still assume the bottleneck exists at the wireless AP. Indeed, when several applications share the same wireless link, it might be the case that the AP receives packets at higher rates than its forwarding rate. This can be caused by several circumstances such as a wireless link not being full-duplex as most wired links.

Table 2.3: Components Used in Several Top Selling Residential Class Wireless APs.

Model	Processor Speed	RAM	Flash Memory
Belkin F5D7130 v1000	Broadcom 4702 @ 125MHz	8 MB	4 MB
LinkSys WRT-54GS, v1.0	Broadcom 4712 @ 200 MHz	32 MB	8 MB
LinkSys WRT-54G, v2.0	Broadcom 5352 @ 200 MHz	16 MB	4 MB
Dell TrueMobile	Broadcom 4710 @ 125MHz	16 MB	4 MB
Dlink DI-524UP, vA.2	RealTek RTL8650B @ 200Mhz	16 MB	4 MB
Microsoft MN-700	Broadcom 4710 @ 125MHz	16 MB	4 MB
Netgear WGR614, v3.0	Atheros 2312 @ 180MHz	16 MB	2 MB

Table 2.3 [56] lists the processors[4] and memory installed for several top selling residential wireless access points. These wireless devices are run embedded Linux on low end processors (most are 32-bit MIPS DSPs) with 8 MB to 32 MB DRAM memory, 8 MB to 16 MB flash memory, and corresponding wireless and Ethernet modules. Figure 2.4 illustrates the software architecture of the PRISM Dual Band Access Point Developer Kit (ISL39300A) [11]. The dark grey blocks in Figure 2.4 represent the non-GNU software which means generally publicly unavailable, and the light grey blocks are open source modules. Based on Figure 2.4, instrumenting the transmit queue inside the Linux kernel may be the best way to implement hardware or driver independent QoS support in an AP.

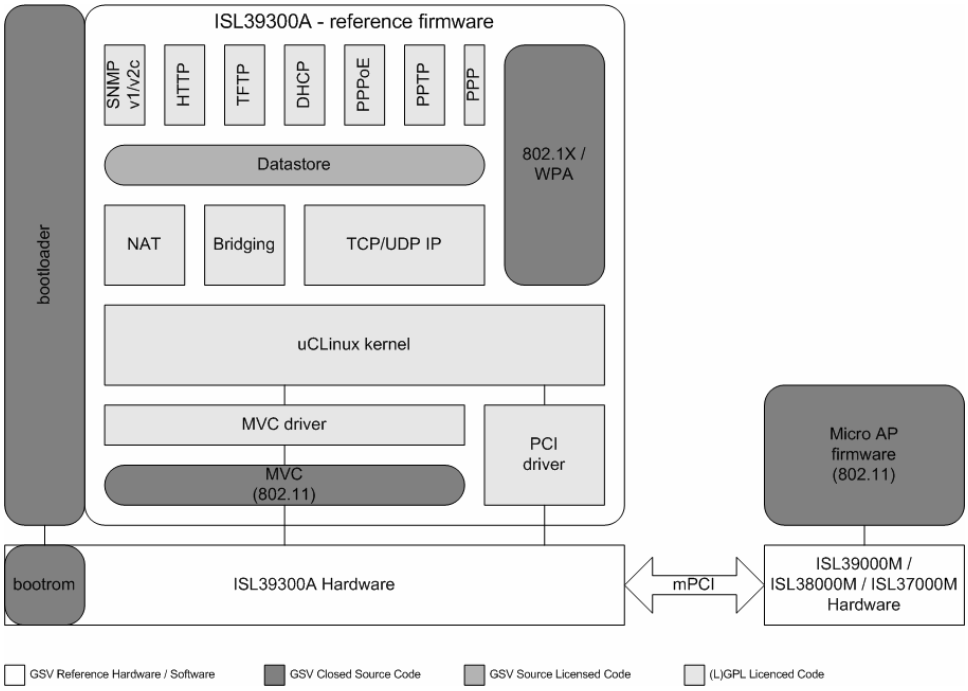


Figure 2.4: Software Architecture for Prism Dual-Band Access Point Developer Kit [11].

2.3 Possible Treatments

In recent years, network researchers have proposed several schemes to provide better QoS support for applications running over wireless networks. These schemes can work on various parts of the

[4] According to OpenWrt [56] the Linksys WRT54G series use several different processors, all of them 32-bit MIPS architecture processors manufactured by Broadcom.

network architecture. Focus has been placed on the data link/MAC layer, IP layer, and application layer. This section contains a brief review of these approaches.

2.3.1 IEEE 802.11e

IEEE 802.11 Task Group E currently defines enhancements to the 802.11 MAC layer mechanisms described in Section 2.2, called IEEE 802.11e [12,57–59] with an Enhanced Distributed Coordination Function (EDCF) by introducing a new coordination function: the Hybrid Coordination Function (HCF). Within the HCF, there are two methods of channel access, similar to those defined in the legacy 802.11 MAC: HCF Controlled Channel Access (HCCA) and Enhanced Distributed Channel Access (EDCA). Wi-Fi Multimedia (WMM) [6] certified APs must be enabled for EDCA, but other enhancements of the 802.11e amendment are optional and HCCA is not mandatorily supported by 802.11e APs.

Both EDCA and HCCA define Access Classes or Category (AC) as a set of access parameters, such as CW_{min}, CW_{Max}, AIFS, and TXOP limit. Traffic that falls in the same access category is effectively given identical priority with respect to access to the medium. A Traffic Category (TC) is a application category that is not related to access to the medium. Therefore, IEEE 802.11e wireless stations can support up to eight access categories, one for each TC. However, Ni *et al.* propose to use fewer ACs than TCs to reduce the MAC layer overhead based on the observation that usually eight kinds of applications do not transmit frames simultaneously [5, 58]. Table 2.4 depicts the mapping between Traffic Categories [12] and Access Categories [5, 6, 58].

Table 2.4: Mapping between Traffic Categories (TCs) and Access Categories (ACs) in [5, 6].

TC	Designation	802.11e AC	Service Type
2	Not defined	AC_BK(0)	Best Effort
1	Back ground (BK)	AC_BK(0)	Best Effort
0	Best Effort (BE)	AC_BK(0)	Best Effort
3	Excellent Effort (BE)	AC_BE(1)	Video Probe
4	Controlled Load (CL)	AC_VI(2)	Video
5	VI (video $\leq$ 100ms delay and jitter)	AC_VI(2)	Video
6	VO (video $\leq$ 100ms delay and jitter)	AC_VO(3)	Video/Voice
7	Network Control	AC_VO(3)	Voice

A single IEEE 802.11e station may implement up to eight transmission queues [12], one for each AC or TC, in order to support IEEE 802.11e QoS parameters that determine their priorities. Figure 2.5 compares the queue implementation architectures between the 802.11 and the 802.11e

standard. In the 802.11 queue implementation (on the left part of Figure 2.5), the device driver only contains one queue which holds various kinds of traffic. In 802.11e devices, as the right part of Figure 2.5 shows, each AC (TC) queue works as an independent DCF station and uses its own backoff parameters. Table 2.5 shows the EDCF parameters used by WMM [6], an industrial standard enhanced with several IEEE 802.11e features. Since the counters of two or more parallel TCs in a single station reach zero at the same time, a scheduler inside the station is needed to avoid virtual collision. Also, the scheduler grants the TXOP to the AC (TC) with the highest priority, as illustrated in Figure 2.5. Note, there is then still a possibility that the transmitted frame collides at the wireless medium with a frame transmitted by other stations.

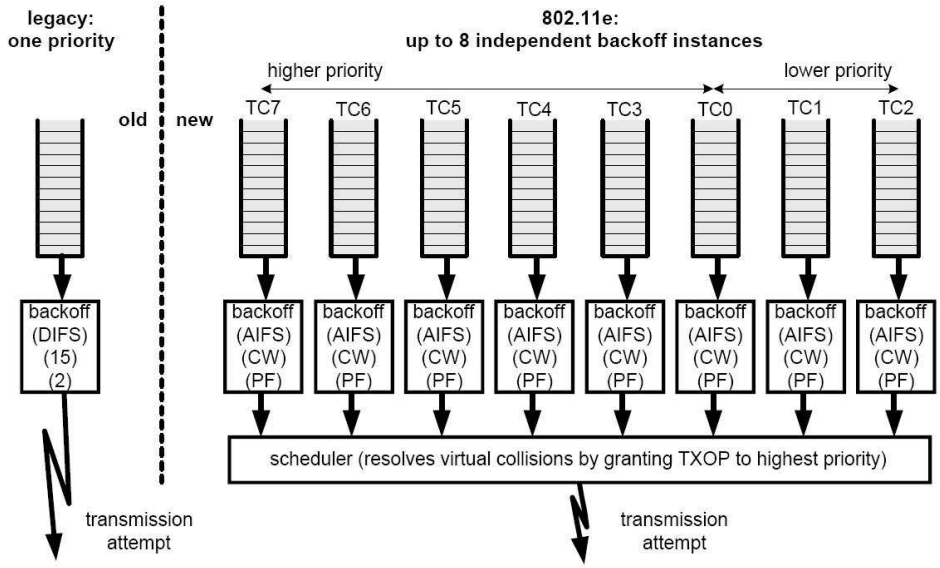


Figure 2.5: Virtual Backoff of Eight Traffic Categories [12]

Table 2.5: WMM Default Parameters for Four Access Categories(AC) [6].

AC	CW_{min} (slots)	CW_{max} (Slots)		AIFS (slots)		TXOP
		STA	AP	STA	AP	
AC_BK	15	1023	1023	7	7	0
AC_BE	15	1023	63	3	3	0
AC_VI	7	15	15	2	1	3.008 ms
AC_VO	3	7	7	2	1	1.504 ms

Limitation of IEEE 802.11e

The scientific community has designed IEEE 802.11e to assign different priorities to traffic traveling through the AP in order to favor real-time applications over download-based applications. This protocol requires the sender to mark each packet with a priority level that can then be used to differentiate among flows. Thus, IEEE 802.11e faces serious obstacles for deployment because current applications do not have this functionality. In future, even if all new applications are enhanced with the ability to mark packets, designers may choose different priority levels for the same applications or not have priority for old applications. Moreover, few the off-the-shelf IEEE 802.11e devices are available, products only providing enhanced subsets of the IEEE 802.11e features, such as WMM [6]. Implementation differences might hinder researchers in providing a general solution to improve QoS support over wireless LAN by tuning the IEEE 802.11e parameters.

2.3.2 QoS Enhancement at the Network Layer

There are many QoS schemes implemented at the network layer. One major advantage is that network layer schemes are common to all architectures regardless of the transmission media involved. This subsection reviews Integrated Services (InteServ) and Differentiate Services (DiffServ) mechanisms in the network layer.

Integrated Services (InteServ)

RFC 1633 [60] describes the Integrated Service (InteServ or IS) architecture, which supports applications requiring resource reservation and prioritization of individual flows to provide QoS. The IS framework consists of four blocks: *Packet scheduler*, *Admission control*, *Classifier*, and *Reservation setup* [61].

- **Packet Scheduler** schedules the transmission of packets with various types of queues. These queues typically exist at the transmit driver in an operating system and correspond to the link level. Different implementations can have different packet scheduling algorithms.
- **Classifier** categorizes each incoming packet into a particular service class so as to be scheduled appropriately for transmission by the packet scheduler. The packets are classified based on certain information that is present in the IP packet header. All packets belonging to the same class are treated similarly.

- **Admission Control** implements the decision algorithm that a router or host uses to determine whether a new flow can be granted the requested QoS without impairing earlier guarantees. The decision to accept or reject a reservation request is based on the QoS and the available QoS on the outgoing link.
- **Resource Reservation Protocol (RSVP)** [62] is responsible for creating the flow specific information in the end hosts and the routers along the path from the source to the destination.

The Integrated Service architecture (InteServ) provides defined services to traffic flows with certain QoS commitments using RSPV signaling. For example, when a station wants to transfer data, it starts by asking the control mechanism at the resource sharing node for a reservation [13]. The request contains a message-signaling packet specifying characteristics of the sender's data, and a resource specification message-signaling packet, which is used for resource reservation. If the reservation is accepted by the node, there is a one-way resource reservation agreement and the station starts to transmit packets to the router. Arriving packets are classified into classes and treated dependent on packet type and the agreement in RSVP [13, 62]. Classified packets are then scheduled and transferred to the next node or its destination.

InteServ provides two services: guaranteed and controlled-load service:

- The guaranteed service is designed for applications which require certain minimum bandwidth and maximum delay. Both traffic specification message and resource reservation message signaling packets are used to provide service that guarantees both delay and bandwidth. A token bucket method is used to reshape the traffic flow into the traffic specification [63].
- The controlled-load service is intended for traffic flows that can tolerate a certain amount of loss and delay. The applications may assume that only few, if any, packets are lost or exceed minimum transmission delay. Only a traffic specification-signaling message is used for flow specification. Admission control is used to assure quality [63].

Unfortunately, the InteServ model is not readily implementable nor scalable [61]. The InteServ model needs to maintain state information for each flow, which is a burden for backbone routers where more than thousands of flows are passing simultaneously. On the other hand, one important assumption made in the development of the IS model is that the network resources can be explicitly controlled [61]. However, controlling the network resources is difficult for residential APs since home users cannot influence nodes outside their residence.

Differentiated Services

Differentiated Service (DiffServ) provides QoS in a network to preferentially identified classes of traffic flows without maintaining per flow status or per hop signaling [61]. Diffserv setup is static at network nodes, which builds domains for a long term [13]. When traffic flows enter a boundary node, they are classified into aggregate behavior classes that designate treatment of packets in the flow dependent on the service level agreement.

There are two essential components in DiffServ mode for an end-to-end implementation: *Marking TOS bits* and *Per Hop Behavior* [13, 61]:

Marking TOS bits Figure 2.6 illustrates the content of Internet Protocol (IP) packet headers which are used by switches and routers in a network to direct packets and also by stations to set values [4]. In particular, Type of Service (TOS) bits are defined to provide QoS support.

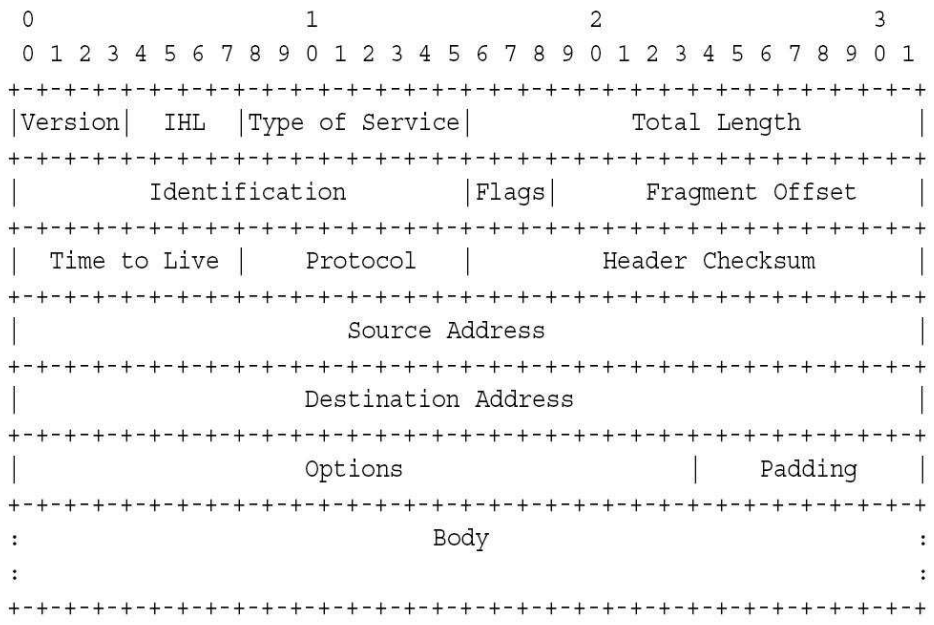


Figure 2.6: Internet Protocol Header [13].

The TOS byte format is shown in Figure 2.7 [13]. The TOS field is eight bits long. The two right most bits (bit 6 and bit 7) are unused. The other six bits are called Differentiated Service Code Point (DSCP) [61], which are used to describe the packet type and possible treatments. The DSCP contains two groups, bits 0 to 2 are used for IP precedence and bits 3 to 5 describes the

DiffServ class selector code points which means the treatment of the packets [13].

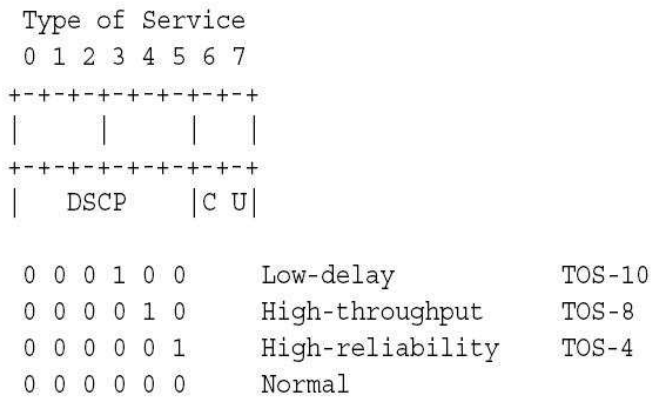


Figure 2.7: TOS Bits [13].

In residential wireless networks, the AP acts as an edge router which connects to up-link devices, wired and wireless residential network. Similar to IEEE 802.11e, applications currently running over residential networks have no ability to mark the TOS octet. Moreover, these TOS bits are not used by most of routers [61]. The wireless APs, therefore, can not simply utilize the TOS routing algorithms to provide QoS support over residential wireless links.

Per Hop Behavior This component refers to the service given for packets dependent on the DSCP code [13]. When packets arrive at an edge router of a Diffserv domain, they are classified into classes based on the DSCP code. Meanwhile, the DSCP code specifies what type of service and treatment the packet shall have in its Per Hop Behavior (PHB). The PHB supports two types of services: *Expedited Forwarding* and *Assured Forwarding*.

- *Expedited Forwarding (EF)* service assures that packets will be forwarded from the node with low delay, low jitter and low loss possibility to its destination. Figure 2.7 shows four examples of DSCP codes with marked EF bits [13]. For applications requiring minimum delay such as Telnet and SSH, the "low-delay" in EF field is marked. Applications like FTP with requirements of fast service for its packets through the network, marks them for "high-throughput". While packets needing guaranteed delivery through the network are marked with "high-reliability". The default is unmarked EF filed with DSCP code "000 000".

- *Assured Forwarding (AF)* offers special service better than best effort, but lower than that in EF. It defines four classes for aggregate traffic behavior. Each class has a buffer space and bandwidth specified in the network interface. For each class there are three drop-precedence values which are used to drop packets when there is congestion [64]. Thus, AF services provide 12 combinations (four classes and three drop precedences) for a packet. When there is congestion in a network, packets with high drop precedence are dropped first [64].

One notable network layer QoS DiffServ architecture is SWAN [65] in wireless ad-hoc networks. SWAN combines several service differentiation features in order to provide service guarantees without requiring special QoS services in the MAC layer. The first feature of SWAN is using an admission control system to deny flows requesting real-time service if the network is not able to meet their requested requirements. SWAN also requires nodes to shape best-effort traffic to ensure no interference with real-time traffic. With these two features, the wireless ad-hoc network can make sure it is never too congested to satisfy its service guarantees. Moreover, SWAN utilizes the congestion notification field (ECN) in the IP header to inform nodes about congestion levels in the network. With the information marked in the ECN field, the nodes can make decisions about admission and traffic shaping.

2.3.3 Treatments in Application Layer

In addition to QoS enhancements at the MAC layer and Network layer, several real-time applications introduce application layer treatments to provide better quality over error prone wireless channels. For example, Skype, a popular VoIP protocol over P2P networks, introduces forward error correction (FEC) packets when the Skype client experiences high packet loss rate. Windows media streaming server can stream a video clip with lower encoding bit rates in order to reduce the stress upon a poor quality wireless link. However, changing such application layer treatments require modification of the applications running the end hosts. Thus, application layer treatments are beyond the scope of this dissertation.

Chapter 3

Related Work

This chapter reviews previous research related to the work in this dissertation. Two corresponding research areas are covered: traffic classification techniques in Section 3.1 and traffic shaping techniques in Section 3.2.

3.1 Traffic Classification

The rapid evolution of the Internet in the last decade has been characterized by dramatic changes to the way users behave, interact and utilize the network. Academic researchers and network operators design and deploy new traffic measurement and classification techniques in response to these new changes. In fact, accurately identifying and categorizing network traffic according to its application type is at the core of many fundamental network research, operation and maintenance activities, such as better QoS support, traffic shaping, intrusion prevention and detection.

It is, therefore, not surprising that researchers and network operators are interested in identifying application types by monitoring network flows. A network flow is defined as *a series of IP packets with the same transport protocol (TCP or UDP), exchanged between two hosts, identified by the five-tuple* $(IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, Protocol)$*, with flow termination determined by a preset timeout or by distinct flow termination semantics* [23]. Each network application generates at least one flow, and often, one application can generated multiple concurrent flows. For example, P2P file downloading often generates dozens of TCP flows. The key part of traffic classification or traffic categorization is to associate flows to a particular application or a group of applications. Network

operators seek the ability to identify applications or hosts which generate a large number of flows or a large volume of traffic.

However, traffic classification is not a trivial undertaking. The rapid emergence of novel applications such as network games, P2P applications and video conferences results in a significant volume of data over the Internet. Consequently, monitoring the traffic generated from such applications becomes difficult. And the increasing presence of malicious traffic requires the traffic classification techniques to detect the denial of services (DoS) attacks, virus, worms etc. Unfortunately, the widespread use of encryption over IEEE 802.11 wireless links makes the measurement, analysis and classification of Internet traffic a more challenging task, even in a well-controlled laboratory environment.

Meanwhile, network operators realized that supporting different levels of QoS for different applications requires association of the traffic with the different applications [22]. One issue behind the slow deployment of QoS support is not the lack of the interest or need, but rather, the absence of effective classification techniques. For that reason, the research community has attempted to design novel algorithms capable of overcoming the limitations of port-based classification techniques, which are ineffective for modern network based applications [20, 22, 66].

Over residential networks, more and more end systems are connected to edge routers by wireless LANs using the IEEE 802.11 standard due its convenience and promised high link capacities. Wei *et al.* [67] use the *packet pair* techniques [68] to identify link connection types for the last hop connection. Their result implies that the same application might behave differently over wireless links than over Ethernet LANs, especially for the packet inter-arrival timing which is widely used as a discriminator for several classification techniques [69]. The application behavior information is vital not just for the research community to design new schemes to provide better QoS supports, but also in the design of future networks, e.g. in planning the capacity required for each kind of traffic, or balancing trade-offs of traffic shaping techniques between cost and performance.

The following section reviews several cutting-edge research achievements in the area of network controlled traffic classification with the emphasis on real-time classification algorithms over residential networks. The reminder of this section is organized as follows: Section 3.1.1 overviews the port-based classification techniques. Section 3.1.2 presents the payload based classification model, and Section 3.1.3 depicts the novel behavior-based approaches. Section 3.1.4 reviews several statistics based approaches. At last, Section 3.1.5 discusses several issues related to classification over

IEEE 802.11 wireless networks.

3.1.1 Port-Based Classification

As mentioned earlier in this section, the major task of flow classification is to identify which application or which kind of application generated such flows. The information used to perform traffic classification can be gathered from the packet header, the payload or the inter-packet timing. But the most common identification technique is based on the analysis of the complete or partial packet header. Based on the information extracted from the packet header, the application type and characteristics can be identified, especially when the application is running on a well-known port. Table 3.1 lists some well known transport ports used by several sample applications.

Table 3.1: TCP or UDP Port Numbers Used by Several 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

Table 3.1 only lists ports used by several common applications, where the information of official port assignment can be found at Internet Assigned Number Authority (IANA) [70]. Generally, the TCP and UDP ports are divided into three ranges: the Well Known Ports (0-1023), the Registered Ports (1024-49151) and the Dynamic and/or Private Ports(49152-65535). For example, a typical TCP client addresses its initial SYN packet to the well known server port of a particular application, where the server is listening while the source port number is dynamically chosen by the client. The UDP connection uses mechanisms similar to TCP but without the connection semantics. All future packets in a TCP or UDP session use the same pair of ports to identify the client and server side of the session. Consequently, in principle, the TCP and UDP server port number can be used

to identify the higher layer application type by simply identifying the server port and mapping it to an application using the registered port list published by IANA (Internet Assigned Numbers Authority) [70].

Port-based classification method can be deployed on IP routers in conjunction with QoS routing in order to direct application flows to capable links. Several classifiers, such as Coralreef[1] based on these kinds of mechanisms are introduced and evaluated in [71], [72], and [73].

Port-based application approaches are the simplest among all traffic classification methods, and highly scalable since the port number from a single packet is enough to identify the application. However, in recent years, researchers have recognized that port-based classification is inaccurate [20, 22]. The main limitation of port-based approaches is that the mapping between the server ports and the applications is not always well defined. For example, many Peer-to-Peer (P2P) applications dynamically use ports which are not registered in IANA. Moreover, the wide deployment of firewalls over residential networks forces applications to run on the well-known ports of other applications. For example, port 80 is used by a variety of non-Web applications, such as Web-based Internet Chat (Web MSN) or video streaming, because the firewalls and other network security software often does not filter port-80 traffic. Last but not least, emerging P2P applications and malicious software (Trojans or DoS attackers) intentionally use well-known ports and generate a high volume of traffic which should not be correlated to applications running on those ports. In the meantime, the implementation of IP over HTTP allows non-web applications to tunnel through TCP port 80 [22].

Table 3.2 enumerates the common applications running on TCP port 80. From that table, applications with different QoS requirements can run on the same port. For example, *scp*, a file transfer protocol, runs on the same port as *ssh* does (TCP port 22). A user can launch a file transfer session by launching *scp* command in an *ssh* session. Thus, a good classification algorithm that

[1] http://www.caida.org/tools/measurement/coralreef/

Table 3.2: Several Applications Running on TCP Port 80.

HTTP for Web traffic.
Streaming for You-Tube or similar video clips.
Streaming for on-line radio services.
Gaming on for World of War Craft.
Downloading for zipped USENET files.
Downloading for Microsoft online updates and patches.
Using software like Google Earth professional or other similar ‘thin’ client software.

needs to differentiate should be able to separate these different applications from the traffic running on same port, and also be able to identify different use cases for the same application [22].

Because of the limitations of port-based classification methods, more and more researchers are looking for other alternative methods which can accurately differentiate the different applications running on the same ports or identify new applications running on unknown ports. However, many network traffic analysis tools [74, 75] still integrate port-based approach because of its simplicity.

3.1.2 Packet Payload Based Classification

To overcome the limitations with port-based approaches, new alternative techniques have emerged by inspecting the content of packet payloads. Payload based methods are based on the assumption that packets in a given flow contain some pattern or signatures that can unambiguously identify a kind of application. This payload based approach is similar to the method that has been widely used in security software which uses the virus signatures (unique bit sequences) to detect the presence of malicious code in files.

Moore *et al.* [7] propose a classification method which can approach 100% accuracy by inspecting the payload of each packet. Their approach is an iterative procedure whose objective is to gain sufficient confidence that the packet is generated by a specific application. As Table 3.3 shows, their approach consists of nine distinct procedures; each procedure tests a particular property of the target flow to obtain evidences of the identity of the causal application.

Table 3.3: Methods of Content Based Flow Classification [7].

	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

As shown in Figure 3.1, each flow is tested against the nine classification methods sequentially. When any of the nine methods gives positive output for the flow, the flow is verified by a *verification* module. The whole classification process terminates if the flow passes the verification module, otherwise it is examined by a labor intensive manual interactive process.

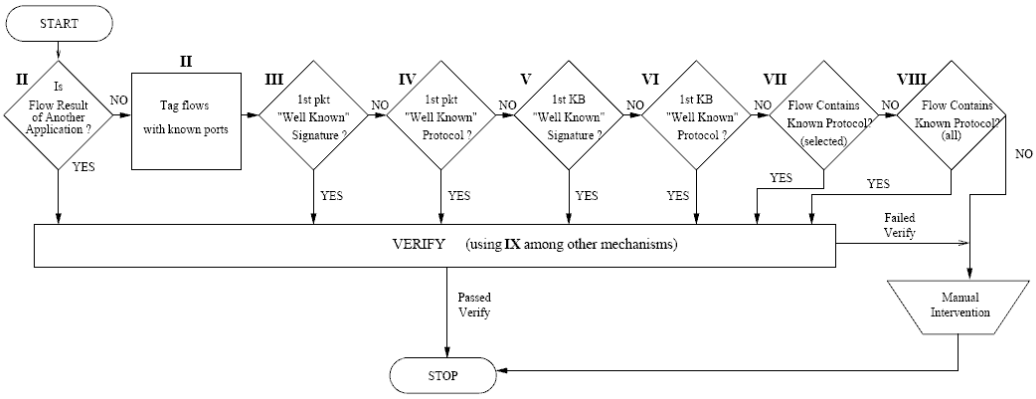


Figure 3.1: Content Based Classification Procedure [7]

The advantage of the classification approach used in [7] is that their approach can identify a flow with near 100% accuracy. However, it is a computation intensive process. In the worst case, when automatic classification fails, the flow is tested against nine sequential classification criteria, one verification process and one manual intervention verification process. The European Networking Tester Center (EANTC) [76] resulted in 99% of detection and accuracy for popular P2P traffic with OpenDPI tools[2].

Therefore, generally, payload based approaches are computation intensive and not typically done in real-time, especially, over high capacity links. It is critical to develop pattern matching algorithms with tight CPU and memory limitations. Another limitation of payload based approaches is that they require the precise prior knowledge on the application protocol specifications and their packet format a head of time. This means that payload based approaches can not automatically detect new applications. To make matters worse, closed protocols (e.g., Skype) do not provide reliable information to produce signatures for all variants of the same application. For example, the P2P protocol (Peercast for instance) is not publicly available and is constantly evolving. Even though some open source P2P protocols like Gnutella provide development documentation[3], they are incomplete. Even worse, various implementations of the Gnutella client may not exactly follow the specifications. Also, the increasing wide usage of encryption techniques, such as for Skype, makes it harder for payload based approaches since the signatures are protected by encryption at the end hosts.

[2]http://www.OpenDPI.org. It is open source version of ipoque's DPI engine. However, this tool is no longer public available in 2014.

[3]http://www.gnutella.com/

However, the payload based approaches rely on inspecting of real packet payload, it is not possible to deceive the payload based approaches by using non-standard ports. Thus, the payload based approach can be used to establish "ground truth" on the existing datasets [77].

In brief, payload based approaches [7, 76, 78] are computation intensive, less accurate on novel applications, and ineffective on encrypted packets. However, payload based approaches can be the most accurate approaches among all classification methods since they get close to the application. Therefore, they can be used to verify the accuracy of other kinds of classification tools.

3.1.3 Behavior-Based Approaches

Unique user behaviors can be used to identify the presence of a particular user on wireless LANs [79]. Unique behavior of wireless card drivers can be used to detect what kind of wireless card driver is used [80]. Similarly, Karagiannis *et al.* [20] proposed a novel classification approach: *BLINC* by analyzing the transport layer behaviors of hosts. Unlike other classification methods, BLINC utilizes the port numbers only as an index without any application related information. The information gathered from the packet header 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) [20, 81]. The three level approach of examining traffic flow is probably the most important contribution of the authors.

Based on the three level information gathered from the packet header, BLINC uses a set of heuristic rules empirically derived through inspection of interactions present in various applications to refine its classification results. These rules are controlled by a set of operator defined thresholds to achieve the desired balance between aggressive and conservative classification [20]. Generally, BLINC reports aggregate per class statistics such as the total number of packets, flow and bytes in addition to producing a list of all flows (five tuple) tagged with the corresponding application for every given time interval.

BLINC can accurately classify $80\% - 90\%$ of real traffic samples with more than 95% accuracy. The authors of BLINC also developed a payload-based classification method to evaluate the accuracy of BLINC. Moreover, behavior-based approach has potential abilities to identify unknown applications, such as new P2P applications and malicious flows, which emerge as deviations from expected behavior. Xie *et al.* developed a behavior-based classification approach to detect the Web

traffic under the hood of HTTP traffic [82], and Yang *et al.* identify P2P live streaming traffic by analyzing the behavioral characteristics [83]. As we mentioned in previous sections, detecting new applications is a challenging task for port- or payload-based applications.

On the other hand, BLINC also has several limitations:

- *Cannot identify specific application sub-type.* BLINC can identify the type of an application but may not be able to identify distinct applications. However, for network management, the finer classification may not be needed. For example, different instances of the same application may have the same QoS requirements.

- *Cannot handle encrypted transport layer headers.* BLINC is based on the relationship contained in the fields of the packet header. BLINC only has the ability to characterize encrypted transport layer payload. This limitation is probably true for most classification methods.

- *Limitation on points of observation.* BLINC is evaluated with traces collected at the edge of the network although intuitively it should not be affected by the monitoring points. However, when BLINC is deployed on the backbone networks, the larger variety of individual user behaviors might be harder to discern.

- *Potential for real-time classification.* BLINC appears sufficiently efficient to allow for a real time implementation. It can process 34 hour trace file in less than 8 hours with flow tables updated over 5 minutes intervals. For network management or security, it is close to a real-time classification. However, for traffic treatment, it cannot handle short-lived traffic.

The motivation of the behavior based approach is different from our treatment classification approach. Our treatment based-approach focuses on placing flows into different categories which can be treated with similar methods. BLINC emphasizes differentiating application types through server behaviors. Therefore, behavior-based classification is not suitable for our treatment based classification requirements.

3.1.4 Statistical Approaches

Because of the shortfalls of the header-based and payload-based classification methods, recently researchers have used machine learning (ML) [84] and statistical approaches to differentiate applications. Claffy [85] investigated the joint distribution of flow duration and number of packets, and

observed the differences between the distributions of some application protocols, although overlaps clearly exist between some applications. After that, a wealth of other researchers [86–88] characterized and modeled the workloads for particular applications. Therefore, based on better understanding of applications behavior, researchers [22, 24, 89, 90] propose statistical methods to infer the application types based on gathered flow statistics.

Compared to the limitation of port-based and payload-based approaches, statistics-based approaches provide a promising alternative method in classifying flows based on application (payload) independent features such as packet length or inter-arrival times [89, 91, 92]. Each traffic flow is characterized by the same set of features but with different feature values [89]. A statistics-based classifier is built by training on a representative set of flow instances where the network applications are known, and a classifier is built to determine the type of unknown flows. Thus, the statistics-based approaches consist of two major steps: features selection and machine learning algorithms used in the classifier.

Statistical Flow Features

Moore and Zuev list 248 flow features which can be used to differentiate application types in [69]. These features are not only derived from the transport layer header (especial the TCP header), but also from packet length and inter-arrival time which are independent from packet payload content. Moore *et al.* [24] utilize a correlation based feature selection method to identify "stronger" features, and find only a small subset of the total 248 features are needed to have accurate classification results.

Roughan *et al.* summarized the candidate features used in [22] into five levels:

1. **Single Packet-Level** features such as mean packet length and various moments such as variance, root mean square size (RMS), etc., are gathered directly from packet header information and are simple to compute. Also, these features are independent of the notion of flows, connection or other higher level aggregations [22]. Another advantage of single packet-level features is that packet sampling has limited impact on these statistics. Other level features can be derived from simple packet data such as time series [22].

2. **Flow-Level** statistics are summary statistics at the granularity of work flows. The five tuple (source IP, source port, destination IP, destination port, and protocol) is widely used to

identify a flow. Flow level features generally include mean flow duration, mean data volume per flow, mean number of packets per flow, and variance of these metrics. These statistics can be obtained using flow-level data collected at routers, e.g. Cisco NetFlow [93]. The limitation of flow level statistics is that the flow collection might aggregate packets that belong to multiple application-level connections into a single flow, which might distort the flow-level features [22].

3. **Connection-Level** statistics are required to trace the behaviors associated with transport level connections such as TCP connections. For example, a typical TCP connection starts and ends with well-defined handshakes for a client to a server. To collect the connection level statistics, the classification algorithm needs to track down the connection state changes. Generally, the connection level includes the advertised windows size and throughput distribution. The connection level data generally provides better quality data than flow level information, but requires additional overhead, and can also be impacted by sampling or asymmetric routing at the collection point.

4. **Intra-flow/connection** features are statistics about the packets within each flow, e.g., the inter-arrival times between packets in a flow. This information requires data being collected at the packet level, but grouped into flows. In addition to inter-arrival packet times, intra-flow/connection features include loss ratios, one-way delay, etc..

5. **Multi-flow:** Some characteristics can only be captured by considering statistics across multiple flows or connections. For example, RTSP [94] supports two connections between the same set of end systems, one connection is used as data connection, and the other concurrently exchanges intermittent control information. Multi-flow features are more complex to capture than flow- or connection-level data.

Table 3.4 [22] summarizes the types of features, and the measurements used to collect them.

Note, Roughan *et al.* [22] define an inter-arrival variability metric and use it as a feature to differentiate bulk downloading data from streaming data. Inter-arrival variability is found to be insensitive to whether or not ACKs are included. Roughan *et al.* find that packet length is another distinguishing feature that can differentiate the bulk transfer and streaming flows, although this feature alone does not provide enough information to separate bulk-data transfer from streaming data.

Table 3.4: Features' Availability for Different Measurement Tools.

Data Source	Features				
	Packet Level	Flow Level	Connection Level	Intra Flow	Multi Flow
Packet trace	YES	YES	YES	YES	YES
Sampled packets	YES	Biased	NO	Biased	Biased
Flow data	Some	YES	NO	NO	YES
Server logs	Some	Some	Some	NO	Some

The representative quality of the selected feature set greatly influences the effectiveness of a statistics-based classification algorithm [91]. Cutting the number of candidate features reduces learning and classification times as well as increases classification accuracy by removing irrelevant or redundant features. The feature selection algorithms can be categorized into a filter or wrapper [95]. In this study, our candidate features can be selected by Correlation-based Feature Selection (CFS) [95, 96]. Other feature selection algorithms can be found in [96].

Correlation-Based Feature Selection (CFS)

The Correlation-Based Feature Selection (CFS) algorithm examines the usefulness of individual features along with the inter-correlation levels among candidate features with heuristic evaluation. High scores are assigned to subsets containing features which are highly correlated with the class but with low inter-correlation with each other [91]. Conditional entropy is used to measure the correlation between each feature and the class. $H(X)$ denotes the entropy of a candidate feature X, and $H(X|Y)$ is the entropy of a feature X given the occurrence of another candidate feature Y. The correlation $C(X|Y)$ between features X and Y is calculated with:

$$C(X|Y) = \frac{H(X) - H(X|Y)}{H(Y)}$$

Williams *et al.* calculate the goodness of a subset of features with the formula [91]:

$$G_{subset} = \frac{k\bar{r}_{ci}}{\sqrt{k + k(k-1)\bar{r}_{ii}}}$$

where G_{subset} is the goodness of a subset, k is the number of features in the subset, $\bar{r}_{ci}$ is mean feature correlation with the class and $\bar{r}_{ii}$ is the mean feature correlation.

Machine Learning Algorithms

The goal of a statistics-based traffic classifier is to form a model based on training data, and find a mapping between the training data and the new data. Machine learning techniques are typically used to find such mapping. Such learning methods are referred as supervised learning methods [23,91]. McGregor *et al.* [25] use the Expectation Maximization (EM) algorithm to cluster flows described by features such as packet length, inter-arrival time and flow duration. With the EM algorithm, traffic can be correctly classified into generic groups such as *bulk-transfer*. Armitage *et al.* [26] also propose an approach for differentiating network applications based on "greedy forward feature search" and the EM algorithm. Their result shows that a variety of applications can be separated into an arbitrary number of clusters. Dunnigan and Ostrouchove [89,97] apply principal component analysis (PCA) to detect malicious intrusion. Their key findings include that network flows show consistent statistical patterns and can be detected when running on either default or non-default ports. Roughan *et al.* use nearest neighbor (NN) and linear discriminate analysis (LDA) to map different applications to four different broad QoS classes: *Interactive, Bulk data transfer, Streaming* and *Transactional*.

Williams *et al.* present a preliminary result of comparing the performance of five supervised machine learning algorithms in [89,91]. Table 3.5 lists ML algorithms widely used by statistics-based traffic classification methods. In our study, Naive Bayesian and Nearest Neighbor (NN) will be used as benchmarks to compare the performance of our treatment-based classification method. Thus, we will briefly review these two methods next, and other algorithms can be found in reference listed in Table 3.5.

Table 3.5: Machine Learning Algorithms Used in Traffic Classification.

Name	Used In
C4.5 Decision Tree	N. Williams [89,91], G. Armitage [98]
Naive Bayes	N. Williams [89,91], Zhang [90].
Nearest Neighbor	M. Roughan *et al.* [22], N. Williams [89,91], Zhang [99]
Multilayer Perception Network	N. Williams [89,91], Zhang [90]
Sequential Minimal Optimization	N. Williams [89,91], Yu [100]
Bayesian Networks	N. Williams [89,91], G. Armitage [98]
Clustering	J. Erman *et al.* [23], Yu [100].
Expectation Maximization	A. McGregor *et al.* [25], G. Armitage *et al.* [26] Ameel [101]
Linear Discriminate Analysis	M. Roughan *et al.* [22], Ameel *et al.* [101]

Naive Bayesian

The Naive-Bayesian classification method is based on the Bayesian theorem, which derives conditional probability for the relations between feature values and the class by analyzing the relationship between each feature and the class for each instance. In a Naive Bayesian classifier, X denotes a vector of instances where each instance is described by k features $X_1, \cdots, X_k$, and C represents a class of an instances, for a given instance x and a given class c. During the training, the probability of each class, called the prior probability $P(C = c)$, is computed as its occurrence frequency in the training data set. Also, the Naive Bayesian classifier computes the probability for the instance x given c. Note, the Naive Bayesian classifier holds the assumption that the features are independent so that the probability becomes the products of the probabilities of each single feature. Yet, the Naive Bayesian algorithm can achieve good results even when that assumption does not hold [90, 91].

After the training process, the probability of a flow instance x belonging to a class c is computed with the Bayes formula:

$$P(C = c|X = x) = \frac{P(C = c)\prod_{\nu_i=1}^{k} P(X_i = x_i|C = c)}{P(X = x)}$$

The above formula combines the prior probability and the probability from each features density function. Note, the dominant is invariant across classes and only necessary when used as a normalizing constant. Thus, it can be computed as the sum of all joint probabilities of the enumerator [91]:

$$P(X = x) = \sum_{i=1}^{k} P(C_j)P(X = x|C_j).$$

Nearest Neighbor

One simple method of classification is Nearest Neighbor (NN). When a new instance is presented to the model established in the training phase, the algorithm assigns the new instance to the class of its nearest neighbor from the training data set. The distance metric between two instances can be calculated by the following formula derived by [102]:

$$D(x, y) = \sqrt{\sum_{i=1}^{n} f(x_i, y_j)}$$

where $D(x, y)$ denotes the distance between two instances x and y. If the ith feature is a numeric value feature, where $f(x_i, y_i) = (x_i - y_i)^2$ (Euclidean Distance), for the nominal feature, $f(x_i, y_i) = 1$ if x_i matches with y_i, otherwise, $f(x_i, y_i) = 0$ [91].

For instance, to classify a new flow x, the classifier computes the distance D between x and each instance in the training data set. If there were 200 training instances, x needs to be evaluated against each of them. The instance in the training data set which returns with the smallest D is consider as the nearest neighbor of x, and flow x is assigned with the same label as y. Roughan *et al.* find that NN methods work well on low dimensional features (n is small) , but is less effective on high-dimensional data [22, 90].

In general, statistics-based approaches can provide results with higher accuracy than port-based approaches, with much less computational complexity than payload-based approaches. Therefore, recent years have seen an increased interests in statistics based approaches with machine learning knowledge [23–26]. The relatively lower computational complexity makes statistics-based approach a candidate for a real-time traffic classification algorithm, because port-based approaches cannot provide accurate classification results over modern networks and other approaches cannot provide real-time information. Most of statistic based methods are fast enough for real-time classification of a large number of concurrent flows (at least several thousands per second) [90, 92]. On the other hand, the objective of current classification approaches is to identify applications without necessarily improving QoS of application over residential wireless networks. For the purpose of providing better QoS support for residential wireless networks, a classifier needs to group the traffic into categories which can be treated similarly.

3.1.5 Challenges for Traffic Classification over Wireless LANs

Traffic classification over IEEE 802.11 wireless LANs have more challenges than over wired Ethernet. Over encrypted wireless LANs, the payloads of IEEE 802.11 frames are protected by various encryption schemes (WEP, WPA, or WPA-2), making it impractical to gather information from IP headers. Thus, statistics-based traffic classification approaches are more important tools if frames cannot be decrypted.

An important assumption behind statistics-based traffic classification is that applications have consistent characteristics, so researchers can obtain statistics information (signatures) for that application. However, this assumption might not hold over wireless LANs since, as discussed in

Section 2.2, MAC layer retransmission and rate adaptation changes flow characteristics such as packet inter-arrival timing. Wei *et al.* [103] are able to differentiate wireless traffic from the traffic over Ethernet by studying inter-arrival times between ACK-pairs. In earlier studies [67], they identify different network types based on the estimation of entropy of the inter-arrival time for packet-pairs. Baiamonte [104] uses a similar approach to passively detect wireless hosts remotely. If a host sends the same constant bit rate flow through the wired and wireless link, it is expected to see more variance in the packet inter-arrival time from wireless transmissions [104]. The statistics-based traffic classification methods developed over a wired link, especially the ones using packet inter-arrival time might not be as efficient as expected over wireless LANs.

Moreover, the management and control traffic (beacon frames, RTS or CTS frames) over IEEE 802.11 LANs also changes packet inter-arrival timing information. Management and control traffic acts as competing traffic [37] to regular data frames and introduces more variance into the packet timing information. In our previous study, we find WBest [37], a bandwidth test tool that uses packet timing information, to be less accurate in an uncontrolled environment partly because of management and control traffic. This management and control traffic is dependent on the implementation of the device driver, and can be passively identified by analyzing the timing of probing frames [80]. Thus, the implementation of various device drivers introduces more unpredictability to the inter-packet timings. Furthermore, the network congestion inside wireless APs and multiplexing with other traffic may distort the timing signature of a given application [104].

Finally, the time varying wireless channel conditions make it more difficult to capture the traffic signature for various applications. The location dependent noise and unpredictable human movements in residential places vary the propagation over the wireless channel. The frames may then not be correctly decoded at the receiving wireless stations with packet loss triggering retransmission or rate adaption. All these unpredictable events introduce variable latency into packet delivery. This unpredictability is exacerbated when many wireless stations co-exist in a condensed residential place because the collisions are more likely to cause extra back-offs and retransmissions.

In brief, statistics-based approaches, especially for inter-packet time are less efficient over wireless networks than over wired networks because of the unpredictable wireless channel conditions and the impact of wireless MAC layer mechanism [104–107].

3.2 Improve Application QoS over Wireless Link

3.2.1 Overview

IP architectures designed to improve the QoS given to various flows are based on the design of packet buffering and scheduling on the IP layer with few assumptions on link layer. However, as discussed in Section 2.2 and Section 2.3, if link layer behavior is not at least predictable under known load conditions, it is not possible to accurately predict QoS support.

Compared to wired environments, the following mechanisms used by wireless packet transmission have direct impact on QoS. First, wireless link layers do not easily fit to either the point-to-point model or the broadcast model. While the medium is inherently a broadcast environment, the scope of a 'line' can be nebulous as there is no physical wire to connect. Thus, sophisticated association control and medium access control mechanisms are required to define the scope of a wireless link. Today, most residential based wireless networks are infrastructure networks (IEEE 802.11 WLAN) where a single AP provides access to a fixed network for multiple attached wireless devices. As we discussed in Section 2.2, in an IEEE 802.11 WLAN, access to the shared medium is based on contention. Therefore, the parameters of the DCF function would impact the QoS over wireless networks.

Secondly, wireless link layers typically deploy different modulation techniques, encoding schemes, and retransmission techniques (ARQ) in order to increase the reliability of transmission over the noisy wireless link. The wireless link adaptation can dynamically choose between different modulation and encoding schemes based on measurements of the prevailing radio environment, trading bandwidth for delay and more robust transmission, and vice versa. Therefore, the delay and bandwidth available for the network layer and network layer flows change over time.

Moreover, the link-layer ARQ retransmission mechanism can introduce unpredictable latency for individually transmitted packets (frames). Although retransmissions are bounded by some counter (maximum number of retransmissions), the resulting delay spread can make it difficult for network layer QoS mechanism to control delay.

Finally, the mobility of wireless stations introduces more problems. In mobile wireless environments, the setup and release of dynamic links and link layer QoS negotiation and re-negotiation impact link layer performance. Also, a small movement of a wireless station may change the receiving signal strength, probably triggering transmission rate changes and unpredictably impact IP

layer performance.

3.2.2 Classifier and Treatment Scheduling Interactions

Operating systems implement a simple data interface between the network and link layer. Each network device is attached with a single input queue [35, 108–111], with a limited amount of buffer space to store the packet (frames) to be transmitted over the link. As Figure 3.2 shows, the link device assumes that the transmitted packet (frames) are in FIFO order and arrive at the network layer in the same order.

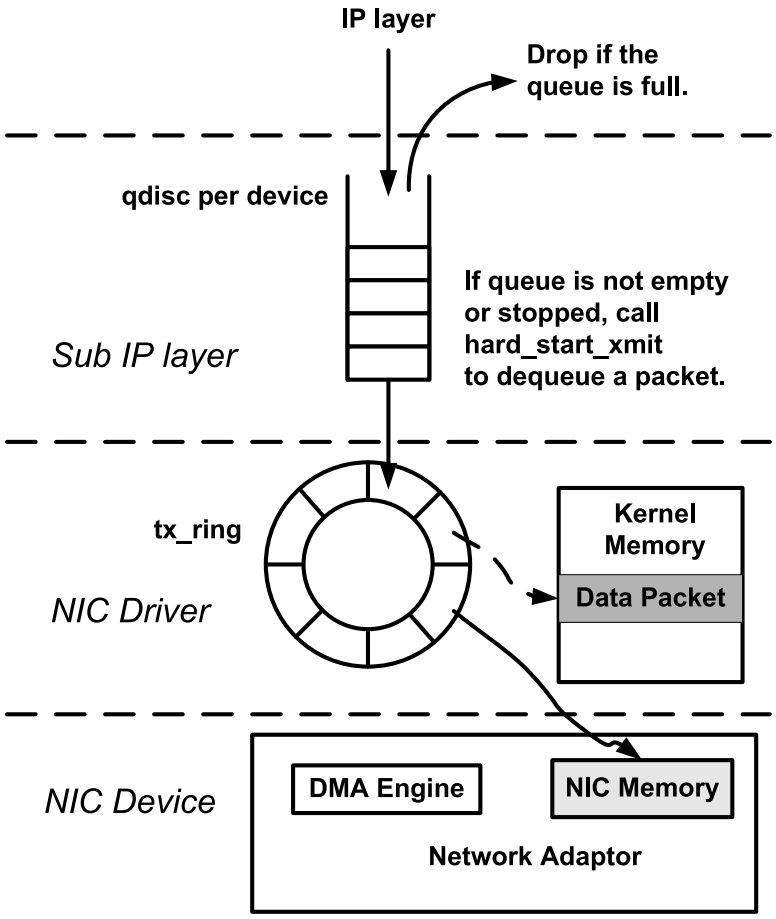


Figure 3.2: Transmission of a Packet in a Host AP, Showing a Link Layer Queue (Driver Queue) and IP Layer Queue.

However, for wireless environments, especially when serving several concurrent flows with different QoS requirements, the wireless medium can not be abstracted simply as a single channel because:

1. The wireless channel may suffer from a high error rate. While the link layer error correction mechanisms (e.g. retransmission) are often able to recover most errors. However, the errors are dependent on the location of wireless client and often bursty. Therefore, flows sent to a particular wireless station may fill up the buffer space in the FIFO queue attached to the link interface, resulting in "head of line" blocking as all other flows have to wait until these packets are successfully delivered [17].

2. There may be several APs working on the same channel. In this case, multiple independent scheduled channels can result in the the head of line blocking again.

3. Flows with different reliability and delay requirements should be treated differently in the link layer retransmission scheme. If a loss sensitive flow without any delay requirement occupies the transmission queue just before a delay sensitive flow, the delay sensitive flow may suffer, especially when the loss sensitive flow has delivery problems.

Figure 3.3 [109,112,113] illustrates four possible approaches to provide different QoS levels with IP layer queues.

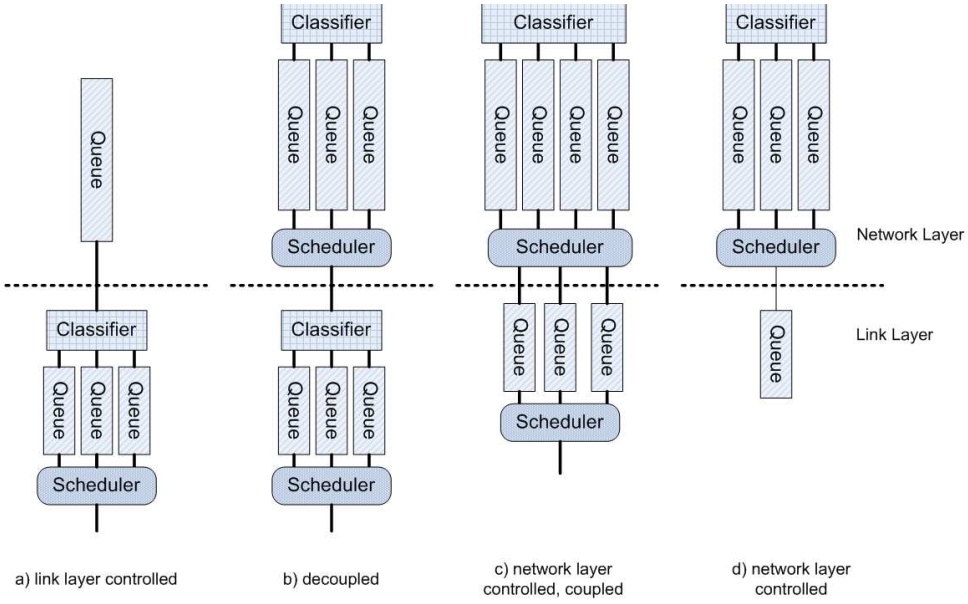


Figure 3.3: Scheduling Models Choices for Wireless Link.

Figure 3.3 (d) shows the *network controlled approach* following the traditional model. In this model, the flow classification and scheduling is performed entirely at the network layer and the

link layer only provides a single FIFO queue. This model works adequately over wired links which seldom encounter link layer distortions. However, over wireless links with high latency and bursty error rates, it may encounter the head-of-line blocking.

The *link layer controlled model* is depicted in Figure 3.3 (a). This model is adopted by IEEE 802.11e [57]. In this model, the link layer handles both classification and scheduling activities. It avoids the head-of-line blocking problem at the link layer, but it has several major drawbacks. To classify the packet (frame) at the link layer, the link layer needs to access IP packets, violating the layered model. Furthermore, the link layer classifier could not access the transport layer header or application layer header if the IP packet payload is protected by encryption.

Even if the link classifier can perform classification for other flows, it might still have head of line blocking problem, but in the IP layer queue [112]. When a single flow or several flows are transmitted at a high data rate, the link layer queue will become full, and finally the packets from these flows will fill up the FIFO queue inside the IP layer. Therefore, the packets belonging to these high volume flows may block the delivery of packets arriving later in the IP layer queue and belonging to higher priority and delay sensitive flows. It results, in turn, that the active queue management (AQM) which provides congestion control may perform inefficiently since the link layer queue is filled up before AQM senses the IP queue build up. In addition, the network layer scheduler needs to implement classification and treatment to support other link layers. Thus, the link layer controlled model might result in function duplications.

The *decoupled approach* in Figure 3.3 (b) combines the *network controlled approach* and *link layer controlled approach* by performing flow classification and scheduling independently at each layer. However, it is inefficient and may have conflicting performance since it has independent scheduling at both layers.

The *network layer controlled, coupled approach* is shown in Figure 3.3 (c) [112, 114]. In this model, flow classification is performed at the IP layer and the IP layer scheduler works together with an appropriately designed middle layer which forwards packets to the correct link layer queue with corresponding link QoS context for IP flows. This model enables efficient use of link resources under upper layer (IP) layer control. Meanwhile, the network layer scheduler should be informed of link layer condition changes which require cross-layer information sharing.

The first three scheduling models all require IEEE 802.11e device implementations. Although, *network layer controlled coupled approach* is an attractive choice, it still needs to implement one ad-

ditional middle layer in addition to a corresponding link layer. Unfortunately, academic researchers have limited access to implementations in the wireless card driver. Therefore, from the view point of implementation, we focus on the *network controlled model* in Figure 3.3 (d), since it can be done with current hardware, and its result can be used for the *network controlled coupled approaches* in the future.

To avoid the head-of-line blocking problem in the link layer queue, we assume that there is only a small transmit buffer inside the wireless card driver and wireless device. In reality, the transmit buffer size inside the device driver is implementation dependent. However, in our previous study, the capacity of transmission queue inside the device driver is less than 40 packets while the default IP layer transmission queue capacity inside Linux kernel layer are much larger[4]. Thus, the dominant queuing delay is in the IP layer when an AP is congested [35].

Traffic Control inside Linux

Traffic control inside the Linux Kernel (2.4.20 or newer) is called Traffic Control (TC) and is included in the network layer. The TC module consists of packet-based queues, classification and scheduling modules, and is designed to provide management of the outgoing traffic. Each network interface on a wireless AP is assigned one queue discipline (qdisc) to control outgoing traffic. A qdisc is called a classful queue if it has classes of queues within it [13]. Each traffic type can be queued into a classless queue under the control of a classful qdisc. The default classless queues defined in TC include:

- **First In First Out (FIFO)** is the most well-known classless queue. Packets are dequeued in the order as arrived. It is the default qdisc for IP queues inside Linux.
- **Token Bucket Filter (TBF)** is a controlling mechanism using tokens and a leaky bucket. Tokens are dropped into a bucket and leak from a controlled hole of the bucket. For each leaking token, a certain amount of data is dequeued. This dequeue method is controlled and can be used to shape packet flows.
- **Stochastic Fair Queue (SFQ)** is a fair distribution of services in the queue for all flows [13]. Packets arriving to the queue are sorted by flows into FIFO queues and these queues are dequeued in a round-robin fashion.

[4]The default transmission queue capacity in IP layer is 100 packets for 2.4 kernel and 1000 for 2.6 kernel.

However, the classification module inside Linux is based on the TOS octet inside the IP header and the traffic control rules are specified by the user. Therefore, the classification module alone inside generic Linux can not provide the result we need.

3.2.3 Methods to Improve IP QoS

This section reviews several techniques to handle IP layer queues and improve multimedia performance on the Internet. There have been two major approaches to handle IP queues rather than the traditional drop tail FIFO queuing schemes. The first approach uses packet or link scheduling on multiple logical or physical queues to explicitly reserve and allocate bandwidth to each class of traffic, where a class can be a single flow or a group of similar flows. This is the basic idea of various fair queuing (FQ) disciplines such as DRR [115], and class-based queuing (CBQ) [116] algorithms. When coupled with admission control, these mechanisms provide a solution to congestion as well as offer potential performance guarantees for the real-time traffic [33,117]. Yu *et al.* [33] improve VoIP applications by implementing a classful qdisc with two FIFO queues: a real-time (RT) queue used for real-time traffic such as VoIP flows, the other a non real-time (NRT) queue used by other kinds of traffic. However, explicit resource reservation approaches change the "best effort" nature of the current Internet, and it changes the fairness definition. Thus, deploying this mechanism probably changes the network management and billing practices. In addition, the algorithmic complexity and state requirements of its scheduling make its deployment difficult [118].

The second kind of approach, called Active Queue Management (AQM), uses advanced packet queuing disciplines other than traditional FIFO drop tail queuing on an outbound queue of a router to actively prevent outbound buffer overflow and control queuing delay with the help of cooperative traffic sources. When notified of network congestion, cooperative traffic sources like TCP flows recognize packet loss as an indicator of network congestion, and its back off algorithm reduces transmission load when congestion is detected [119].

Figure 3.4 lists the AQM taxonomy used in this dissertation, with a more the complete version summarized by Chung [14]. Generally, the task of AQM is divided into four parts: a *Congestion Monitor* which detects and estimates congestion; a *Bandwidth Controller* that manages the output bandwidth; a *Congestion Controller* that calculates and applies the congestion notification probability (CNP) to incoming traffic and a *Queue Controller* which manages buffer usage and packet scheduling schemes. Because treatments used by our proposed treatment based classification are

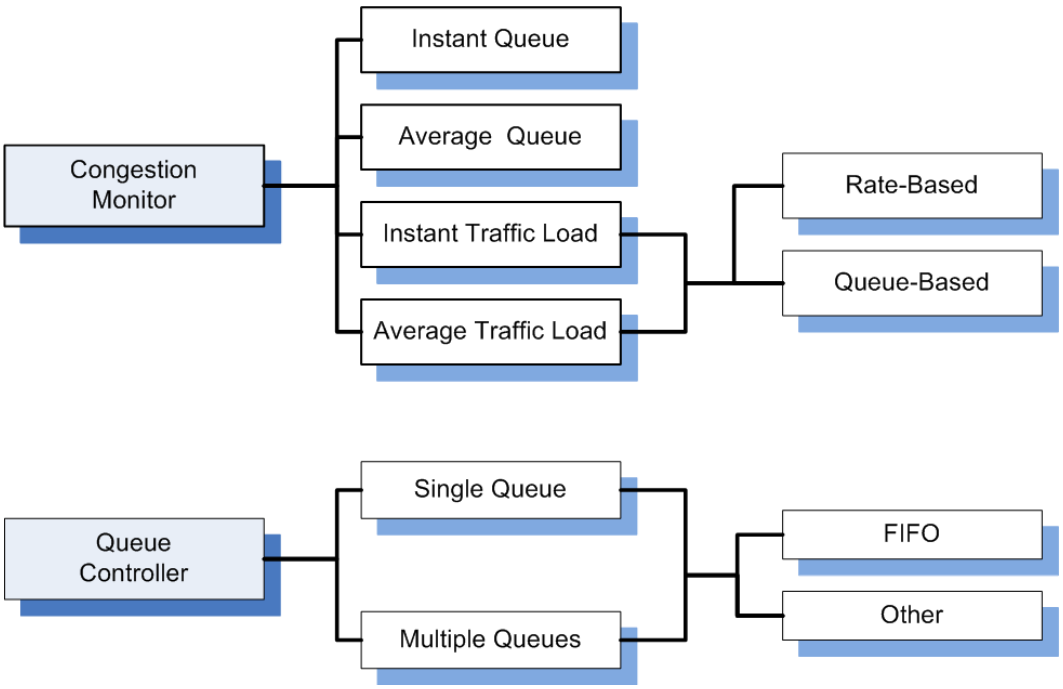


Figure 3.4: AQM Taxonomy [14]

to be deployed over wireless APs with limited resources, our treatments focus on the *Congestion Monitor* and the *Queue Controller*.

Congestion Monitor

The first task of an AQM is to efficiently monitor, detect and estimate congestion [14]. This estimation is used by the Bandwidth Controller to manage bandwidth, or by the Congestion Controller to compute congestion notification probability (CNP).

AQM congestion detection and estimation mechanisms can be classified by the monitoring policy using either *queue length* as a measure of congestion, or the incoming *traffic load* as a measure of congestion. In each case, either the *instant* or *average* measure of the quantity can be used.

By using queue statistics, AQM mechanisms detect impending congestion by monitoring instant or average queue length against a pre-decided queue threshold. The Congestion Monitor can also measure the incoming traffic load and detect congestion when the measured load is larger than preset target load threshold which is typically set to near 1. That means the congestion occurs when estimated incoming traffic load is greater than the outgoing link capacity. Generally, either an instant queue or average measure of the traffic load can be used in the load-based policies, whereas

an average load can be defined as the average of instant loads over a certain period [14].

The traffic load based congestion estimation policies can be further classified by the monitoring method (traffic rate or queue length) to estimate the traffic load [14]. Traffic load is intuitively measured in terms of incoming traffic rate over service rate. Moreover, the traffic load can also be estimated in terms of queue length differences over a measurement interval. Both rate-based and queue-based load estimation methods have advantages and disadvantages. Rate-based methods have a little more overhead than queue-based methods since rate-based methods need to collect every incoming packet length while queue-based methods can sample the queue size every measurement interval. However, rate-based congestion estimation methods can detect impending congestion before the queue starts to grow, allowing the Bandwidth Controller or Congestion Controller to more efficiently respond to imminent congestion. This advantage of the rate-based congestion estimation is attractive for wireless links. For instance, sending large packets at relative low sending rate to a wireless station with weak reception signal strength also causes the congestion at the wireless AP even though the instant queue length might be low for a short time interval, but per packet queuing delay is probably high.

One important issue in load estimation is to determine an effective measurement interval. A poorly chosen measurement interval may have negative effects on the stability of the feedback control system, link utilization and queuing delay and cause unwanted buffer overflow [14]. For example, an insufficiently small interval can lead to the Congestion Controller over-reacting to network congestion while an exceedingly large interval can make the Congestion Controller less responsive. Similarly, when an averaging technique is used for congestion estimation, the averaging factor affects the responsiveness and performance of the controller.

Queue Controller

AQM queue controllers utilize advanced packet queuing disciplines other than traditional FIFO drop tail queuing on the egress queue of a wireless AP to actively handle or avoid congestion with the help of cooperative traffic sources [117]. The back-off algorithm in TCP flows reduces the transmission rate when there is packet loss as an indicator of congestion. Random Early Detection (RED) [120] is one of the earliest and most well-known AQM methods, which prevents congestion by monitoring egress buffers to detect impending congestion and randomly selects and notifies senders of network congestion to reduce their transmission rate. However, RED fairly handles congestion

for TCP flows, but non-TCP flows which are unresponsive or have greedier flow control mechanisms than TCP consume more of the share of the bandwidth than do TCP flows [117]. In the worst case, unresponsive UDP flows monopolize the output bandwidth while TCP flows are forced to transmit at their lowest rates.

To address the unfairness between unresponsive UDP flows and well-behaved TCP flows, several algorithms have been proposed to penalize the unresponsive or misbehaving flows and protect well-behaved TCP flows, such as Fair Random Early Drop (FRED) [121]. FRED adds per-active-flow accounting to RED, and isolates each flow from the effects from others. However, FRED has a potential problems that its TCP favored per-flow punishment could unnecessarily discourage flow controlled interactive multimedia flows [117, 121–123].

Jeffay *et al.* [118] propose an active queue management scheme called Class-Based Threshold (CBT), which releases UDP flows from strict per flow punishment while providing protection for TCP flows by introducing a simple class-based static bandwidth reservation mechanism to RED. CBT classifies the flows into three categories: tagged UDP (multimedia UDP), untagged UDP and TCP. It utilizes class thresholds that determine ratios between the number of queue elements that each category may use during congestion [117]. However, the static resource reservation mechanism of CBT could result in poor performance when traffic mixes change rapidly.

The Traffic Sensitive QoS controller (TSQ) [124, 125] provides Quality of Service when used in conjunction with most existing AQM mechanisms. TSQ finds the trade-off between delay-sensitive applications and throughput-sensitive applications by providing a low queuing delay without significantly reducing the link utilization. TSQ also permits the packet from delay sensitive flows using "cut-in-line" mechanisms. Meanwhile, in order to avoid penalizing throughput-sensitive applications, during congestion, TSQ adapts the drop probability to provide higher throughput for delay-sensitive applications [125].

Generally, most AQM techniques try to control the queue size by dropping packets with a uniform drop probability. The Queue Controller can support diverse per-flow QoS requirements using a packet scheduling scheme other than FIFO. To provide better QoS, the Queue Controller can consider the delay requirement of each flow as well as the QoS information gathered from the traffic sources. However, the QoS packet scheduling may need to address issues such as starvation that can affect the throughput of window-based traffic such as TCP flows.

AQM Enhancement over Wireless Link

Palazzi *et al.* study the interference between MAC layer protocol and Transport layer protocol over wireless links [105], and propose a solution to reduce last hop delays for real time applications by tuning the parameters of the IEEE 802.11 MAC protocol [126]. They assume that UDP based real-time applications are resilient to packet loss while sensitive to packet delivery. Thus, it is preferable to drop a packet rather than to waste time on retransmissions. Meanwhile, they try to find an optimal compromise between the needs of TCP-based and real-time applications by adjusting the MAC layer buffer size in the AP. They argue that a large buffer helps to maintain larger sending rates and diminishes the impact of busty traffic during a long time period while a large buffer may jeopardize time sensitive applications because of the longer queuing time when the buffer is filled up. Li *et al* improve the video delivery quality over IEEE 802.11 networks by combining AQM with IEEE 802.11e [127]. Note, our previous study [35] has similar conclusion that larger IP layer buffer size might be harmful to time-sensitive applications.

When congestion occurs in APs, dropping a small fraction of UDP packets alone will not help recovery from congestion. Palazzi *et al.* proposes an approach called *Smart Access Point with Limited Advertised Window* (SAP-LAW) [126]. SAP-LAW improves the overall performance of wireless APs by limiting the maximum sending rate of TCP connections. It maintains the sending rate of TCP flows high enough to efficiently utilizing the available bandwidth while, at the same time, avoids excessively using buffers by limiting growth. In a real scenario, SAP-LAW needs to compute the appropriate upper bound for the sending window size and find a way to change the advertised window size.

The main goal of SAP-LAW is to achieve full utilization of the available bandwidth with no queuing delays [126]. To avoid queuing delays, SAP-LAW does not allow the aggregate bandwidth consumed by TCP flows to exceed the total link capacity of the bottleneck link reduced by the real-time traffic traveling over UDP. Thus, the maximum sending rate for each TCP flow (R_{tcp_max}) at time t is presented by :

$$R_{tcp\_max}(t) = \frac{C - T_{udp}(t)}{\#TCPflows(t)}$$

where $T_{udp}(t)$ represents the amount of bandwidth used by UDP-based applications at time t, $\#TCPflows(t)$ corresponds to the number of TCP flows at time t, and C depicts the current link capacity.

Computing the most appropriate value for the advertising windows size requires a comprehensive knowledge about all the flows that are passing through the bottleneck link. Thus, SAP-LAW selects the wireless AP as the most suitable place to apply the above formula because the AP is able to retrieve all necessary information from the flow passing through it. The AP can gather the information such as the connection type and link speed by querying the status of the network interface, also, it can sniff the wireless channel or inspect the packets passing through to infer the number of active TCP connections and the aggregate number of UDP flows. The AP hence can easily compute the ceiling of the TCP sending rate and modify the advertised window size in the corresponding ACKs.

However, SAP-LAW could not automatically detect real-time traffic since it assumes that real-time traffic is sent over UDP. However, due to the wide deployment of firewalls, a large fraction of real-time applications such as video streaming are delivered by TCP [128]. Thus, to take advantage of SAP-LAW, it needs to be enhanced with an automatic traffic classification module. Otherwise the real-time applications traveling over TCP would be assigned more bandwidth than they can consume while other TCP connections are limited by under-rated advertised window size.

Andrew *et al.* propose CLAMP, a receiver side rate control algorithm in [129, 130]. CLAMP attempts to allocate bandwidth to users when the user's channel condition is good, but with fairness constraints. CLAMP deploys a pair of agents on the wireless AP and wireless client. The agent on the wireless AP sends the queue length information to the client agent by modifying the packet header or answering query messages from the client agent. With the queue information, the client agent computes and sets appropriate TCP advertised window size in TCP-ACK packets in order to reduce the sender's congestion window size. Compared to SAP-LAW, the major advantage of CLAMP is to deploy agents on both wireless stations and wireless APs. Thus, for particular wireless devices like game consoles CLAMP faces difficulties in deployment. In this study, we will focus on the methods to enhance APs without any protocol modification on the client or server. However, the ideas behind the CLAMP algorithm are useful in implementing our rate control algorithms.

3.3 Measurement Tools

This research includes a wireless measurement study to provide a snapshot of the current home wireless usage. Several measurement tools have been developed and used in our previous study [35]

and [131], and a couple of them are developed for this study.

Wireless Sniffer

While sniffers have been widely used to monitor network traffic at the data-link layer and above, most commercial wireless sniffers are costly and are not a flexible open source solution. However, wireless sniffers are the only tool which does not interfere with the hosts under test and does not require access to wireless devices themselves. Especially, wireless sniffers can be used to measure black-box devices such as hand-held smart phones and wireless APs. However, due to the cost of commercial class wireless sniffers, this study chooses to convert a Linux based laptop as a wireless sniffer. We chooses to use a Dell laptop with Prism GT IEEE 802.11b/g card as an open source wireless sniffer. The step by step guide can be found in [132].

However, wireless sniffers including commercial class ones lack the ability to decrypt the frames protected by WPA2-EAP or WPA-PSK in realtime. Thus, the wireless APs monitored by wireless sniffers are required to choose WEP or unencrypted, otherwise wireless sniffers only can record the information of MAC layer headers.

Wireless Scanner

The wireless sniffers only can monitor one of eleven IEEE 802.11b/g channels at same time. Thus, wireless sniffers are not able to capture the beacon frame sent by APs working on different channels. Therefore, we develop a small program, wireless scanner (wlscan) to scan all eleven channels and detect all possible APs even including the APs which disables beacon frames.

The wireless scanner (wlscan) configures the wireless card to transmit probing frames on each of IEEE 802.11 channels, and collects all probe-response frames replied by every visible neighbor APs. Wlscan then retrieves AP related information such as received signal strength (RSSI), encryption scheme used, working channel and BSSID etc, from the collected pro-response frames. Because wlscan forces the wireless device driver to work in probing mode, it disables the normal send and receive functions of wireless cards. However, the probing process only takes 15 seconds, and wlscan resets wireless cards and resumes their normal activities. In this study, wlscan scans for neighbor APs every half hour for 48 hours. Moreover, we carefully choose the grace period between the runtime of wlscan and other active measurement tools to allow the network to fully recover from the probing status.

WRAPI

WRAPI is a software library that allows user space applications to gather information from IEEE 802 network cards [5]. In our previous study [35], we enhanced the WRAPI to support Windows XP/SP2 and Vista, and developed a tool with same name to collect IEEE 802 statistics from wireless device drivers. The WRAPI tool records the wireless layer information such as RSSI of the associated AP, which is collected directly from the IEEE 802.11 device driver. Different from the wireless sniffer, WRAPI reports the wireless link quality of the active measurement client directly, while the wireless sniffer observes the wireless traffic from a different observation point. Compared to the wlscan tool, the WRAPI tool is able to run when the wireless card is configured in their normal mode. But the WRAPI tool is only able to report the RSSI of the associated APs, and it could not detect the existence of neighbor APs.

UDP Ping Tool

Our previous studies [35, 133] found several limitations of the Ping tool packaged with Windows XP,

1. many ISPs and WPI network administrators block ICMP traffic based on security concerns.
2. the Ping program could not precisely control the sending rate greater than 2 packets per seconds.
3. ICMP packets are handled differently than regular IP based packets in most network devices.

Thus, this study reuses the UDP ping tool developed in [35] to measure the round trip time (RTT). The UDP ping tool sends and receives a 1500 byte long UDP packets every 500 ms and reports the round trip time just as regular ping does. The UDP ping tool provides a larger time out threshold up to 5 seconds for each packet, a more actual measurement result with a finer clock granularity, and an ability to transmit packets at a more accurate rate. The UDP ping can easily go through the firewall of the WPI campus network.

VoIP and Game Application Emulator

VoIP and game applications are two common kinds of realtime applications running on residential networks. Both of them have similar characteristics, such as low bandwidth consumption etc. In

[5] http://sysnet.ucsd.edu/pawn/wrapi/

this study, we planned to evaluate the performance of them over several volunteers' places. In stead of using any commercial software, this study uses a small tool to generate synthetic VoIP and game flows. The key part of the VoIP and game application emulator is based on the implementation of MGEN 4.0 [6], an open source tool set to generate real-time traffic patterns. The VoIP and game emulator simply configures the MGEN API and drives the generated loading patterns over the course of the time.

The duplex VoIP voice traffic is emulated by sending a series of 200 bytes long UDP packets in both upstream and downstream. The UDP packet size and interval are selected based on the $G.711$ codec. Upon connection, both the server and client send 200-byte long packets (including UDP and IP layer header) every 20 ms to each other. A single VoIP session lasts 40 seconds. The performance of the VoIP application is measured using the Mean Opinion Score (MOS) and R-factor produced by the E-Model [134].

The synthetic game flow emulates the traffic of Halo 2 [135], where the game server sends UDP packets with 72 bytes payload to the client and the client sends UDP packets with 44 bytes payload every 40 ms. A single game session lasts 40 seconds. Game performance is measured by the MOS produced by the G-model based on Quake IV game performance [136].

MediaTracker

A customized version of Windows Media Player, *MediaTracker* [7] developed with the Windows Media Software Development Kit (SDK), starts to measure video performance over residential wireless network. MediaTracker provides the basic ability to provide buffering usage and playout frame rate statistics information when playing back Windows Streaming Media Clips. In this measurement study, Windows Steaming Media (WSM) Server (v 9.0) streams a high motion, 640 × 360 pixel, 24 frames per second, 120 second long video clip to MediaTracker running on the active measurement client. In order to evaluate the media scaling behavior during our experiments, two video clips with same content are used in this study. One clip is encoded with multiple bit rate levels with top rate of 1128 Kbps, and the other is encoded with a fixed rate at 1128 Kbps. Both video clips stream with TCP protocol to go through the widely deployed firewalls over residential networks.

In addition to the above tools, this study uses FTP/SFTP tools to measure the uplink and

[6] http://www.nrl.navy.mil/itd/ncs/products/mgen
[7] http://perform.wpi.edu/real-tracer/download/MediaTracker/

downlink capacity respectively. Each ftp session lasts 40 seconds and the average application layer throughput is calculated based on the actual transferred file size.

3.4 Summary

This chapter summarizes research work related to this study: classification technologies, QoS treatment technologies, and measurement tools used in our home wireless measurement study. We group the flow classification methods into four categories: port-based, payload-based, statistics-based and behavior-based. There are trade-offs between each category of classification method, but none of them can be easily mapped into a QoS treatment method. Similarly, QoS treatment methods are complicated. Some treatment methods are based on the assumption that the flow has been pre-classified - for example, TOS bits have been set correctly; other treatment methods are complicated to configure by a home user; and some treatment policies, e.g., dropping, may not suitable for TCP-based flows. Thus, this study provides a bridge between flow classification and treatment by automatically classifying flows in a wireless AP and selecting appropriate treatments to improve QoS for applications running over home wireless networks.

Chapter 4

Home Wireless Measurement

A typical home wireless network consists of different kinds of IEEE 802.11 enabled devices including laptops, smartphones, game consoles, printers and other entertainment devices. The main objective of the home wireless measurement study is to gather first-hand wireless traces from current residential networks, measure the performance of different kinds of applications, and have quantitative observations of realtime application running over residential wireless networks. We conducted a home wireless measurement study at six volunteers' home around New England area in Spring 2009. Because of resource limitations, this wireless measurement study only provides a snapshot instead of a generic picture of residential wireless network usage in year 2009.

An objective is to evaluate the performance of several applications including VoIP, game, video streaming and file transfer over residential wireless networks. In particular, this study wants to answer questions such as:

1. What kind of wireless frames are transmitted?
2. Do the neighbors' APs effect the performance of wireless networks ?
3. What kind of applications are running on wireless links ?
4. How does a real time application perform over home wireless links ?

Additionally, the measurement study results also help tune simulation parameters of the CATNAP implementation as well as validate the wireless model of NS-2.3.

This chapter is organized as : Section 4.1 describes the experimental design and set up; Section 4.2 shows the one slice of our measurement study results from physical layer to application

layer as a case study; and Section 4.3 validates our NS-2 simulation setup with our measurement result.

4.1 Experiment Design

Figure 4.1 depicts the generic topology setup for the wireless measurements conducted. As the Internet connection speed in Massachusetts increases from 5.7 Mbps (2009 Q4) to 9.1 Mbps (2012 Q3) [137, 138], IEEE 802.11g wireless APs with 54 Mbps link capacities are typically not the bottleneck over residential networks. Therefore, we place an internal application server behind users' broadband ingress devices to emulate the future Internet server with fast link connections. In addition to the internal server, the measurement study set up an external server on the WPI campus network to emulate an application server over the current Internet. This measurement setup allowed us to control the application servers and the measurement client, although this study does not have access to any network devices along the network path between servers and wireless clients.

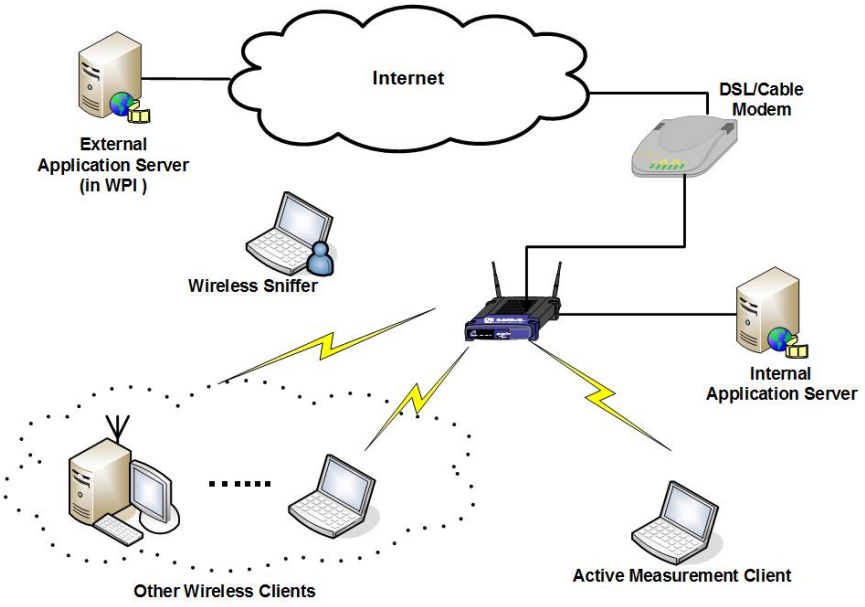


Figure 4.1: Residential Measurement Study Setup

As Figure 4.1 shows, the internal application server, the active measurement client and the wireless sniffer sit inside the volunteer's home. The measurement study consists of two parts: active measurement and wireless sniffer measurement. Even though time sensitive multimedia

applications have become popular, researchers only identified a small fraction of home network traffic as time sensitive [139]. HTTP still dominates traffic over residential networks [139]. Therefore, the measurement study introduces active measurement, which intentionally injects a set of applications: video streaming, FTP, and emulated VoIP/game applications to ensure that the same type of time sensitive applications are running over the volunteer's home network. In this way, the expectation is to observe performance degradation of the injected applications when they are contending with the users' normal network traffic.

The wireless sniffer shown in Figure 4.1 collects the wireless layer traffic from the volunteers' wireless network. Because this study only has access to the application servers and the active measurement client, the wireless sniffer is the only way to provide information of network activity of volunteers' wireless APs and their wireless clients.

4.1.1 Active Measurement

The active measurement suite runs between the "active" measurement client and two Windows 2003 servers: internal and external servers. An IBM ThinkPad $T41$ laptop with Windows XP/SP2 equipped with a Netgear WAG-311 IEEE 802.11g card serves as the active measurement client. A P4 Windows Server 2003/SP2 desktop is placed in our volunteer's home network, acting as the internal server. Another Windows 2003 Server with Intel Core2 CPU is deployed on the WPI campus network, acting as our external server. Both the internal and external server provide FTP, SSH, Windows Media, emulated VoIP and game services to the active measurement client.

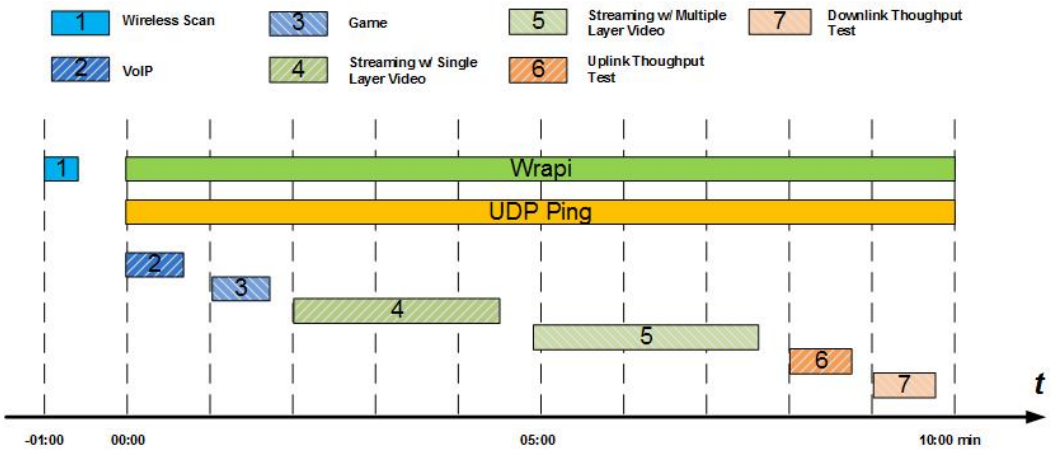


Figure 4.2: Time Execution of Programs Used in Active Measurement Study

Figure 4.2 diagrams the execution plan of applications in each active measurement suite running on our wireless active measurement client. As Figure 4.2 shows, the wireless measurement client launches FTP, streaming, emulated VoIP and game applications sequentially. In addition to the wireless sniffer, several measurement tools (described in Section 3.3): a wireless scanner (wlscan), a wireless link monitor program (WRAPI), and a RTT measurement tool (UDP ping) also runs from the active measurement client. The whole active measurement suite runs for 10 minutes. The active measurement suite runs four times every hour: two with the internal server and two with external server. A five minute grace period is introduced between each run, and a script collects and backs up log files on the active measurement client during the grace period.

The active measurement suite is designed to minimize its intrusiveness. Except for the two 40 second long "bandwidth eager" FTP sessions at the end of each run, other applications are non-greedy. Among the injected real applications, windows video streaming consumes the most bandwidth, as much as 1 Mbps, and the rest VoIP/Game flows only consumes an insignificant amount of bandwidth. During our measurement test period, five of the six volunteer families did not notice the presence of the active measurement traffic. Thus, the active measurement client just acts as one additional wireless client in the volunteer's home network, and launches the pre-known applications periodically. The wireless sniffer treats the active measurement same as the volunteer's wireless client.

4.1.2 Wireless Sniffer Measurement

A Dell laptop with a Prism GT IEEE 802.11b/g card acts as a wireless sniffer in this study. It monitors the wireless activities over the volunteer's home network. Since the location of the sniffer significantly influences the sniffer result [140], the sniffer is placed no more than six feet from the volunteer's AP to capture traffic from/to the AP as much as possible. The side effect of placing the sniffer close to the AP rather than to the active measurement client is that the RSSI observed by the sniffer is different from the RSSI observed by the active measurement client. As compensation, *WRAPI* running on the active measurement client provides the signal strength information when analyzing the active measurement results.

With the respect to the privacy of our volunteers, the sniffer only stores the first 250 bytes when the captured wireless frame length is greater than 250 bytes. The first 250 bytes include the complete information of wireless layer header, IP layer header and Transport layer header which is

adequate to identify flows and application type through a port-based classification approach. The study was run between 8 : 00 AM every Tuesday and 8 : 00AM every Thursday during March 17th and May 20th, 2009.

4.2 Home Wireless Measurement Results

Table 4.1 lists the selected seven volunteer residences in this study. However, the data sets gathered from the last residence are not complete, because the owner uses an IEEE 802.11n AP, which transmits and receives frames on two individual channels. The wireless sniffer used in this study lacks the ability to concurrently monitor traffic on two separate channels. Therefore, our analysis focuses on the data set gathered from the first six residences.

Table 4.1: Residential Places Selected

DataSet	Description	Access Point			Clients	
		Model	Channel	Security	Wired	Wireless
Glaston	Retired couples in a single family house.	LinkSys WRT54GL	6	None	2	2-3
Worcester1	A graduate student in a residence tower with more than 100 units.	LinkSys WRT54GL	6	None	1	1
Worcester2	A graduate student in a house in a thickly settled area.	LinkSys WRT54G-TM	6	128-WEP	1	2
Worcester3	A couple with a one year old baby, and a senior in a two family house.	DLink WBR-1310	11	128-WEP	1	2
Worcester4	A graduate student in a three family house.	Berlkin F5D7234	1	64-WEP	1	2
Waltham	A software engineer in an apartment	LinkSys WRT54GL	8	128-WEP	0	1
Worcester5	A young couple on the third floor in 9 unit apartment.	Trendent 54N	3	128-WEP /WPA-PSK	1	2-3

Table 4.2 summaries experimental information from the active measurement study and the volume of data collected by the wireless sniffer. The original plan was to find two locations with different signal strengths at each of volunteer's residence. However, while conducting the experiments, it was difficult to find a location experiencing low RSSI in four out of the six residences. Consequently, the wireless active measurement client was not moved to a bad location in four residential places on the second day. The active measurement runs 48 hours and only generates a small size (less than 10 MBytes) application level trace file. Therefore, only active measurement traces

collected over the first 24 hours at each location are processed.

Moreover, some unexpected issues arose when the active measurement suite was running at several residences. For example, because the AP in the residence of Glaston renews the DHCP address automatically for the internal server, the active measurement client lost its connection to the internal server. Consequently, the active measurement with the internal server only ran for 6-8 hours. The WPI network administrator shut down the port used for the UDP ping tool when the wireless measurement study was running at the residence of Waltham and Worcester1. Because the measurement client is behind the firewall of participating residences, it is impossible for us to fix it remotely without the owner's cooperation. Thus, some of the active measurement results are incomplete.

Meanwhile, because the wireless card on the wireless sniffer is configured in monitor mode, the sniffer does not have a network connection and it is not re-configurable once it starts. Due to the lack of network connectivity, the sniffer could neither synchronize its clock with a NTP server nor take any command remotely. Thus, the wireless sniffer starts collecting the wireless trace when it is deployed and stops collecting when the active measurement study finished. As Table 4.2 shows, the duration of the wireless sniffer trace is longer than the duration of the active measurement trace.

Table 4.2: Data Sets Collected during Home Wireless Measurement Study

Dataset	Date	Active Measurement		Wireless Sniffer	
		Internal	External	Size	Duration
Glaston	04/14-04/15	6-8 hrs	Complete	5.75GB	89hrs
Waltham	03/17-03/18	Complete	ping failed	10.17GB	85hrs
Worcester1	05/12-05/13	Complete	ping failed	9.69GB	55hrs
Worcester2	04/21-04/22	Complete	Complete	23.12GB	75hrs
Worcester3	05/18-05/20	Complete	Complete	12.85GB	97hrs
Worcester4	03/30-04/01	Complete(33hrs)	Complete(33hrs)	3.68GB	16hrs

Because each active measurement study runs for 48 hours Tuesdays and Wednesdays, there is not much traffic during the daytime, late night and early morning. Unlike enterprise or campus networks, the most active time period of the residential networks is during the early evening. After parsing all wireless sniffer traces collected, no traffic except the injected active measurement traffic is found during the daytime, late night or early morning. Therefore, it is redundant to show the analyzed results of all wireless traces.

Table 4.3 summarizes the results, tools and involved datasets. The following part of this section is organized follows: Section 4.2.2 and Section 4.2.3 only show the results from a one hour long

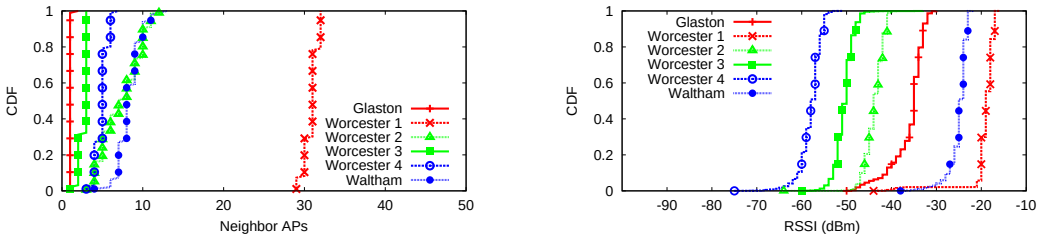
evening wireless trace and a one hour long early morning trace from gathered wireless sniffer traces as a case study. Section 4.2.1 and Section 4.2.4 shows the results from our active measurement experiments from all six residences.

Table 4.3: Summary of Measurement Results

Layer	Measurement	Tools	Data Set
Physical	Active	wlscan	All six locations
Wireless	Sniffer	sniffer	Worcester2
Network	Sniffer	Sniffer	Worcester2 excludes active measurement traffic
Transport			
Application	Active	FTP, UDP ping, WRAPI, VoIP/UDP, Game/UDP, MediaTracker	All six locations

4.2.1 Physical Layer Result

Two small tools, the wireless scanner (wlscan) and WRAPI [35], are developed to collect wireless physical layer information. Different from the wireless sniffer, these two small tools are running upon the wireless active measurement client, and report the physical layer information from the view of the wireless client. Figure 4.3(a) shows the cumulative distribution function (CDF) of the observed APs reported by the wireless scanner (wlscan) running on the first day of each experiment, while Figure 4.3(b) depicts the CDF of RSSI of the volunteer's AP observed by WRAPI.



(a) Number of Observed APs

(b) RSSI of Associated AP

Figure 4.3: Wireless Environment of Volunteers' Residences

Figure 4.3(a) shows that the wireless scanner observes more than one AP in five of six residences. At the residence in Glaston, the wireless scanner sees only one AP most of the time. As expected, several APs are visible to the active measurement client at the other five residences in the thickly settled suburbs around Boston or Worcester. For example, Figure 4.3(a) indicates that the residence

Worcester1 sitting in a residential tower in Worcester downtown observes more than 30 APs in its neighborhood. At such a location with high-density APs, the wireless frames would experience high error rate if the user picks a channel which a large number of APs use.

Figure 4.3(b) summarizes the RSSI measured by WRAPI at the six residences. Our previous studies [17, 34, 133] demonstrate the wireless received signal strength indicator (RSSI) is an important metric to predict whether video streaming applications are experiencing good quality. In [34] and [133], reception locations with RSSI between -70 dBm and -30 dBm are considered locations with good wireless conditions, where usually video streaming applications perform well. Figure 4.3(b) shows that the RSSI at all six locations is in "good" range and it predicts that video streaming applications will perform well under such signal strengths. The results in Section 4.2.4 confirm this prediction.

4.2.2 Wireless Layer Result

The wireless sniffer is the only tool used in this measurement study to monitor the volunteer's wireless activity. However, there is no active network activity during the daytime, deep night or early morning. During these time periods, the only traffic captured on the volunteer's network is injected by the active measurement client. Therefore, this study simply divides captured wireless sniffer traces into two categories:

- *"busy" traces:* gathered during the early evenings between 1900PM and 2100PM, when most likely family members are running network applications; or
- *"quiet" traces:* gathered during the early morning, daytime and night, when the volunteer family members are seldom online.

After parsing all 60+ GB of sniffer traces, the quiet traces gathered from all six volunteers' families are virtually similar: they only contain the traffic injected by the active measurement client. Meanwhile, the busy traces contain the traffic generated by the volunteers as well as the traffic injected by the active measurement client. However, the dominant traffic from the volunteers are HTTP-based traffic. Because only six families were involved in this study, one cannot draw statistics significance from these results. The wireless, IP and Transport layer analysis is based on two one-hour sniffer traces collected at the Worcester2 residence as a case study: one gathered between 19:00PM-20:00PM, April 21st, 2009 as the representative of "busy" traces, and the one

collected between 05:00-06:00AM April 22nd, 2009 as the representative of the "quiet" hour traces.

Table 4.4: Different Wireless Frames Captured during a Busy and a Quiet Hours

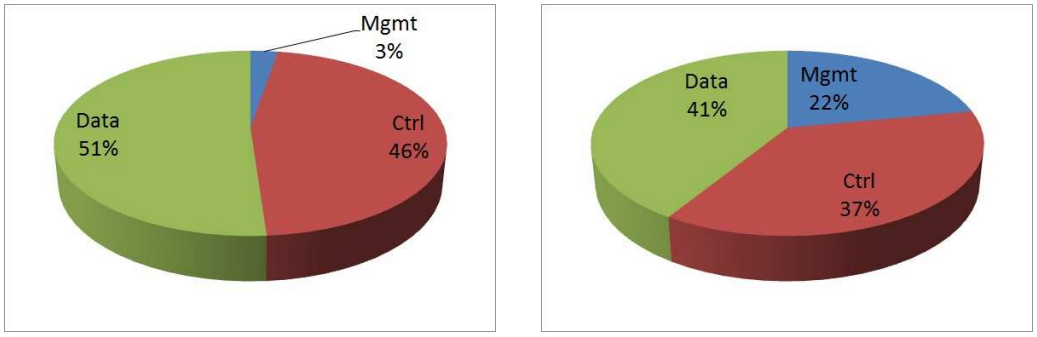
Frame Type		*Busy*				*Quiet*			
		Count		Volume		Count		Volume	
		Num.	Perc.	bytes	Perc.	Num.	Perc.	bytes	Perc.
Mgmt	Probe Req.	12.2 kilo	0.3	0.8 MB	0.1	1.3 kilo	0.2	85.7 KB	< 0.1
	Probe Resp.	35.5 kilo	0.9	3.5 MB	0.2	37.6 kilo	5.7	3.2 MB	1.0
	Beacon	61.0 kilo	1.5	6.3 MB	0.4	107.2kilo	16.1	10.0 MB	3.8
	Authen.	2	0	79	0	-	-	-	-
Ctrl	Reserved	1	0	18	0	1	0	18	0.0
	CTS	13.5 kilo	0.3	0.2 MB	< 0.1	1.0 kilo	0.6	14.2 KB	< 0.1
	ACK	1.9 mill.	45.9	26.2 MB	1.8	0.2 mill	36.7	3.4 MB	1.1
Data	Data	2.1 mill.	51.7	1.4 GB	97.5	0.3 mill	41.1	0.3 GB	94.5
	NULL Data	1.9 kilo	< 0.1	53.5 KB	0.0	25	0.0	700	0.000
	QoS Data	2	0	205	0	-	-	-	-

Table 4.4 compares different types of wireless frames captured during the representative busy and quiet hours at the residence Worcester2. The data frames (type 32) and ACK frames (type 29) are the two major dominant frame types during both busy and quiet hours. Note, the data frames and ACK frames shown in the quiet hour are traffic injected by the active measurement client. The wireless layer analysis does not exclude the active measurement traffic because the active measurement client can be treated as another normal wireless user on the volunteer's network.

In additional to data and ACK frames, 0.3% frames from the busy hour trace are CTS frames while no corresponding RTS frames are captured. The 802.11g standard provides CTS-to-Self signaling protection mechanism to manage communication in a hybrid 802.11 b/g environment [141]. Although the volunteer's AP at Worcester2 disables RTS/CTS, an IEEE 802.11b device associated with a neighbor's AP still sends CTS frames. Since both APs are sitting on Channel #6, the wireless sniffer captures the CTS frames from the neighbor's AP due to the shared media nature of wireless networks. However, the CTS-to-Self scheme introduces more overhead and lowers the overall throughput of IEEE 802.11g APs [141].

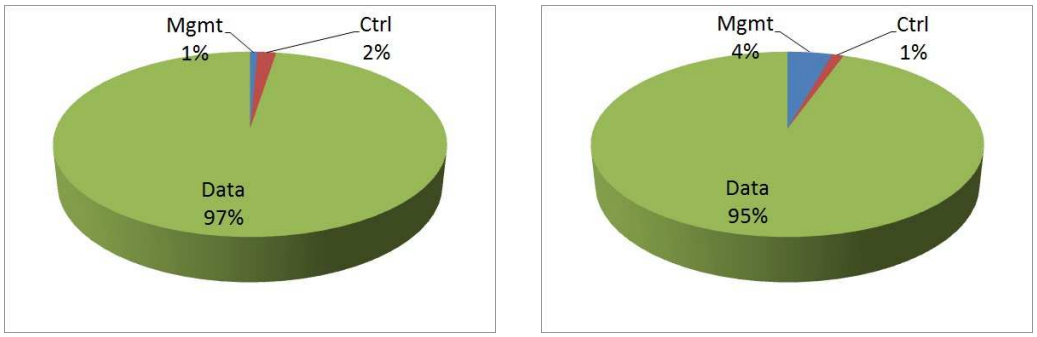
Furthermore, Table 4.4 shows that the wireless sniffer collects more beacon frames during the early morning (the "quiet" hour) than the early evening (the "busy" hour). According to IEEE 802.11 standard, an AP transmits a beacon frame every 100 ms by default. It is expected that the sniffer captures the same amount of beacon frames during the same time period because an AP is always on and stationary. As Abib *et al.* noticed, human activity has effects on IEEE 802.11

wireless channel conditions [142]. One possible explanation is that wireless channel conditions are much better during the early morning than the early evening because there are almost no human or computer activities around 5AM.



(a) a "busy" hour (b) a "quiet" hour

Figure 4.4: Frame Count by Types

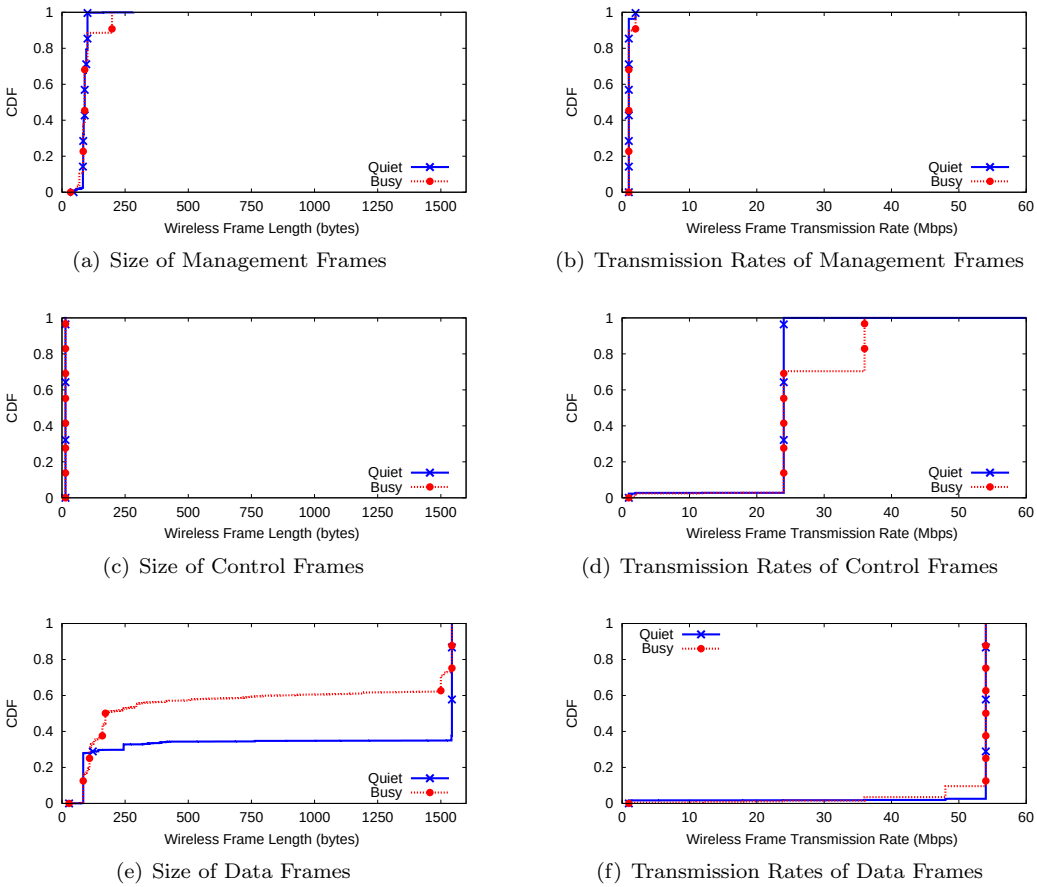


(a) a "busy" hour (b) a "quiet" hour

Figure 4.5: Frame Volume by Types

Table 4.4 gives the volume of each type of frame in addition to their count, and Figure 4.4 and Figure 4.5 visualize the content of Table 4.4. Most of the management or control frames are small, but data frames produce 95% or more of the traffic volume. However, the small management or control frames still potentially reduce the overall throughput of wireless link because the per frame physical layer header introduces relatively more overhead and reduces the channel utilization [140].

Figure 4.6 (a)-(e) compares the cumulative distribution functions (CDF) of frame size and transmission rate for management, control and data frames captured in the busy and quiet hours. The majority of the management frames and control frames are smaller than 250 bytes. But the length of data frames are in bi-modal distribution, 20% to 50% frames are smaller than 250 bytes



(a) Size of Management Frames

(b) Transmission Rates of Management Frames

(c) Size of Control Frames

(d) Transmission Rates of Control Frames

(e) Size of Data Frames

(f) Transmission Rates of Data Frames

Figure 4.6: Size and Transmission Rate Comparison between Different Types of Wireless Frames

and the rest of them are larger than 1000 bytes. The small data frames could be control packets from upper layers such as TCP ACK, and TCP control packets which are transmitted as data frames in the wireless layer; or they could be application packets (VoIP or game packets).

Meanwhile, the majority of data frames are transmitted at high speed as fast as 54 Mbps because the users' wireless clients are usually placed in the same room or the room next to the AP. However, management frames, such as probing response frames sent by APs, travel at a lower speed to reach a large range. These low speed and retransmit-able management frames lower the wireless throughput when a wireless client sends probing requests to an AP at a distance.

Wireless Data Frames

Although control frames and management frames can have significant impact on the performance of residential wireless APs, data frames are more interesting to study. Tuning the wireless param-

eters on control frames and management frames for residential APs potentially can improve the performance of home wireless networks, but this measurement study focuses on the understanding the behavior of data frames.

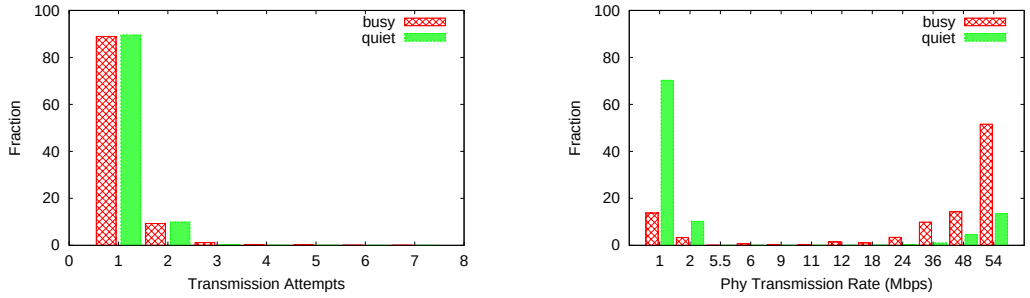
The technical report from Broadcom Inc [141] shows that wireless IEEE 802.11g cards are able to support 54 Mbps transmission rates within 75 feet. Considering the size of a typical house in North American is only 2000 square feet, most wireless clients will be within 75 feet of the wireless AP sitting in the middle of the house. Additionally, wireless signals can easily penetrate materials such as drywall, plywood, wood and glass, which are common construction material in North American houses. Thus, it is expected to observe a large fraction of data frame transmitted at rates of 54 Mbps in the six volunteers' small houses.

Table 4.5: Data Frame Transmission Rate Comparison

Transmission Rate(Mbps)	*Busy*				*Quiet*			
	Count		Volume		Count		Volume	
	count	frac.	bytes	frac	count	frac.	bytes	frac
1	5,589	0.3	1.0 MB	0.1	1,900	0.7	640.3KB	0.2
2	570	0.0	0.1 MB	0.0	-	-	-	-
5.5	138	0.0	97.3 KB	0.0	87	0.0	7.8 KB	0.0
6	1,283	0.1	0.5 MB	0.0	3	0.0	252	0.0
9	687	0.0	0.6 MB	0.0	-	-	-	-
11	1,188	0.1	1.0 MB	0.1	-	-	-	-
12	3,139	0.2	1.6 MB	0.1	13	0.0	1.2 KB	0.0
18	3,056	0.1	2.2 MB	0.4	40	0.0	6.7 KB	0.0
24	7,569	0.4	4.4 MB	0.3	101	0.0	27.2 KB	0.0
36	52,869	2.5	38.1 MB	2.4	367	0.1	241.9 KB	0.1
48	155,677	7.5	102.3 MB	7.4	1,865	0.7	587.6 KB	0.2
54	1.8 mill.	88.9	1.2 GB	89.4	269,376	98.4	287.8 MB	99.5

Table 4.5 lists the physical transmission rates of data frames gathered at Worcester2, during the busy hour and quiet hour as a case study. The physical transmission rate is reported by the Prism2 wireless card device driver on our wireless sniffer, and the actual data transfer rate is lower than the physical transmission rate. Table 4.5 shows than more than 88% of data frames are sent at 54Mbps, which is the maximum transmission of IEEE 802.11g during either busy or quiet hour. Even during the busy hour, 19:00-20:00PM on April 21st 2009, when wireless frames have high collision probability, only 12% of data frames are sent at rates lower than 54 Mbps.

Figure 4.7 depicts transmission attempts of data frames gathered at the Worcester2 location during the busy and quiet hours. The logical link control layer (LLC) of IEEE 802.11 introduces



(a) Data Frames (b) Retransmitted Data Frames

Figure 4.7: Data Frames and Retransmitted Data Frames

the retransmission mechanism in order to provide the reliable communication over an error prone wireless channel. However, the IEEE 802.11 retransmission and the exponential backoff algorithm actually introduces more overhead and lowers the medium utilization [58]. Figure 4.7(a) shows that about 90% of the data frames arrive at their recipient with their first attempt, and less than 1% of the data frames attempt more than 4 times. Figure 4.7(b) focuses on the physical transmission rate of retransmitted data frames which have been attempted transmission more than once. During the busy hour, more than 50% retransmitted data frames are sent at 54 Mbps and 13% retransmitted data frames are sent at the lowest 1 Mbps. However, during the quiet hour, nearly 70% of the retransmitted data frames are sent at 1 Mbps.

4.2.3 IP and Transport Layer Results

The initial objective of analysis of the IP and transport layer information is to understand the flow level behavior of network applications running at home. However, we lack control over the volunteer's AP. This study only can retrieve the IP and transport layer information from the IP and TCP/UDP headers encapsulated by the wireless frames from the wireless sniffer traces. The wireless sniffer is not able to capture all airborne wireless frames. Shulman *et al.* report that wireless sniffers inevitably miss at least 5% of wireless frames [140]. Recovering the IP and transport layer information from frames captured by a wireless sniffer is not a reliable approach for flow level analysis. To avoid the disadvantage of the sniffer, several researchers choose an indirect way to get the flow level information of residential wireless networks by analyzing traces gathered from the ISP's edge routers [106, 143, 144]. But this study has no access to any ISP devices and the traces from the ISP's edge router do not contain any wireless level information.

Since the flow level information retrieved from a wireless sniffer trace may not be reliable, this study only presents the IP and Transport layer results from the busy hours at Worcester2 location as a case study. In order to study "native" flows generated by our volunteers, the flow level analysis excludes the traffic injected by the active measurement client. The quiet hour is at the early morning between 05:00-06:00AM, and there is no active volunteers' network activity after excluding our active measurement traffic. Therefore, our case study focuses on the traffic gathered between 19:00PM-20:00PM, April 21st, 2009 at the Worcester2 location.

As discussed in Section 3.1, no reliable flow level classifier is available to analyze the wireless traces captured during this measurement study. For example, statistics based approaches usually need training datasets, but our home wireless measurement does not have enough historical traces to train the classifiers. Therefore, we choose port-based traffic classification methods, which may be somewhat inaccurate. To respect the volunteer's privacy, the wireless sniffer truncates any packets larger than 250 bytes and stores the first 250 bytes of each wireless frame. This information is enough to recover the IP and Transport layer information but may not be enough to recover application layer information.

Table 4.6 lists the native flows detected in the "Worcester2" site, during 19:00-20:00PM, April 21st, 2009, the busy hour. As expected, the port-based classifier marks a large fraction of flows as "unknown". However, the port-based approach identifies another large fraction of flows as some applications which should not appear on residential networks. But after carefully inspecting the flows, we find that these flows are mistakenly classified by the port-based classifier because the source port or destination port of these flows accidentally matches some port registered with IANA. Therefore, this study groups these flows as "misc". Except for these two categories, the dominating flows captured over residential networks are Web flows and DNS flows. However, some interactive flows could be covered under the big umbrella of Web, e.g. browser-based games and Web-based online chats. Unfortunately, the port based approach cannot identify them separately.

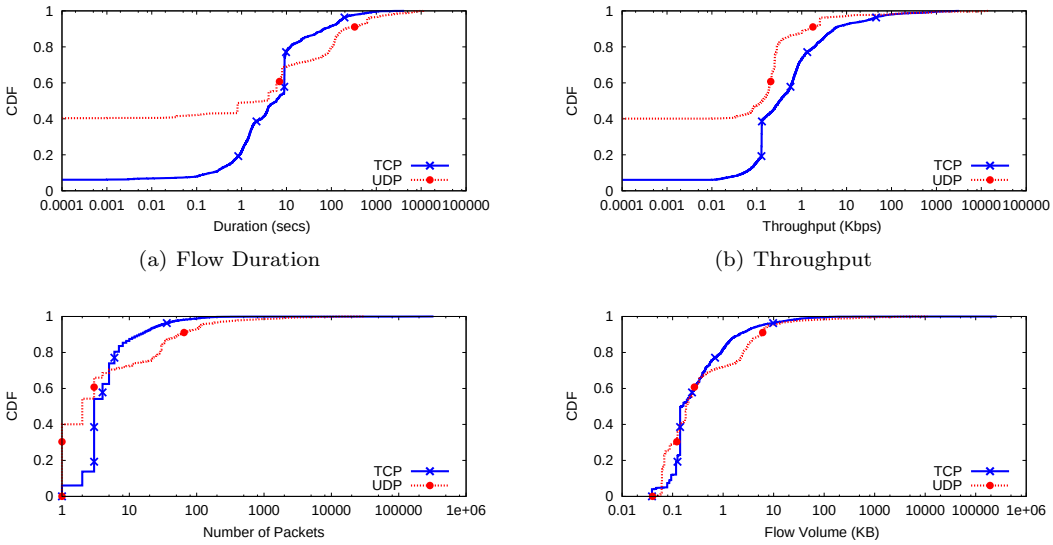
Figure 4.8(a) summarizes the duration of native flows detected during the busy hour trace. We use the terms, *dragonflies* and *tortoises* defined by Brownlee *et al.* [145] to describe the lifetime nature of IP flows: dragonflies are the flows lasting less than 2 seconds and tortoises are flows lasting longer than 15 minutes. As Figure 4.8 shows, 40-50% of flows captured over residential networks are dragonflies.

Figure 4.8(b) shows that the throughput of most TCP and UDP flows is lower than the link

Table 4.6: Native Flows Captured over Busy and Quiet Hours

	Upstream Flows			Downstream Flows			Total
	TCP	UDP	Subtotal	TCP	UDP	Subtotal	
Web (port 80,8080,443)	703	0	703	708	0	708	1411
DNS (port 53)	0	616	616	0	638	638	1254
acmsoda (port 4939)	107	0	107	218	0	218	325
unknown	155	383	538	217	512	729	1267
misc	644	134	778	678	180	858	1636
Total	1609	1133	2742	1821	1330	3151	5893

capacity of an IEEE 802.11g network. This is because the gateways in the volunteer's residential network are bottleneck devices, and most home users only surf Web pages. Figure 4.8(c) depicts the number of packets detected in each flow.



(a) Flow Duration

(b) Throughput

(c) Number of Packets in Flow

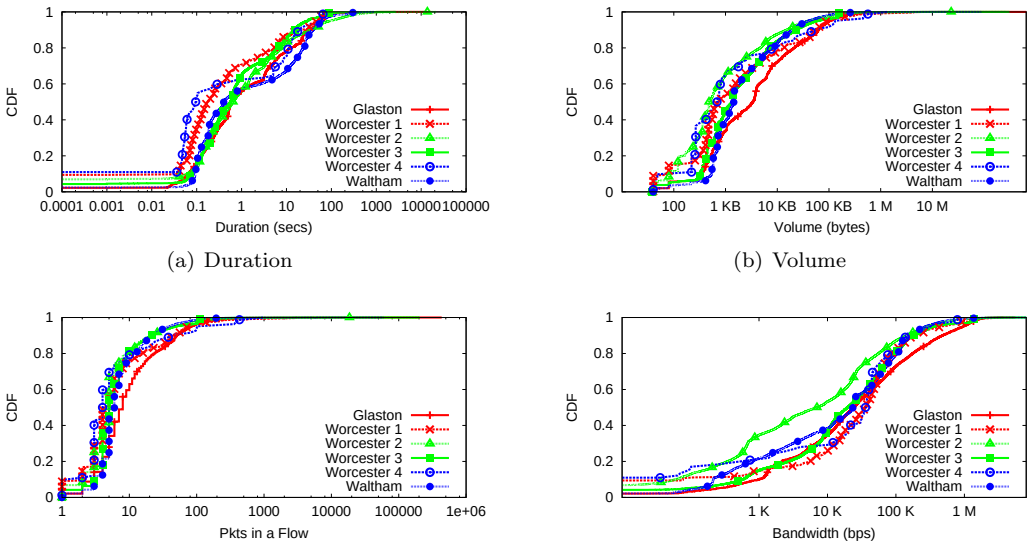
(d) Volume of Flow

Figure 4.8: IP and Transport Layer Results of Native Traffic

Web flows

Web flows are the dominant flows detected in the busy hour at Worcester2 location. Although the port-based classifier does not work well for applications running on unpublished ports, it is a reliable approach on applications running on well-known ports, e.g. Web applications, DNS applications. Thus, we analyze all Web flows detected from the trace gathered during 19:00-20:00PM on the first day at all six locations. The Web port used are Port 80, 443, 8080 and 8081 [70]. Figure 4.9

summarizes the duration, throughput, packets in flow and flow volumes for the Web flows detected.



(a) Duration (b) Volume

(c) Number of Packets (d) Bandwidth

Figure 4.9: Web Flows

Figure 4.9 shows that high similarity exists in the Web flows captured at all 6 volunteers' locations. The majority of Web flows are mice flows with small volume as well as dragonflies with short duration. 75% of the Web flows last less than 10 seconds, and 80% of them are smaller than 10 KB in total.

4.2.4 Application Layer

This section presents the results of our active measurement study. The main purpose of our active measurement study is to provide performance for game, VoIP and video streaming applications over residential wireless network. Different from Section 4.2.2 and Section 4.2.3 which analyze our sniffer traces, this section uses the results from application logs gathered from our active measurement client.

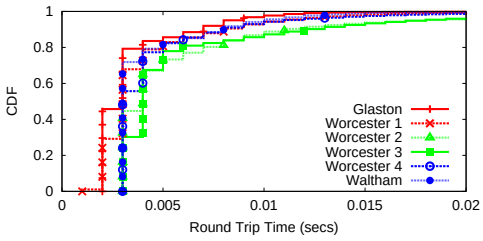
However, the active measurement test suite experienced several unexpected technical and non-technical issues. In Glaston, because the owners' AP reset its DHCP service 8 hours after our active measurement starts, our active client was unable to detect the internal server's IP address changes and the active measurement in Glaston includes only 7 hours of data collection with our internal server[1]. Moreover, a WPI network administrator shutdown our UDP ping port due to security

[1]The wireless scanner (wlscan) and WRAPI collect 48 hour data because they do not need to connect to the

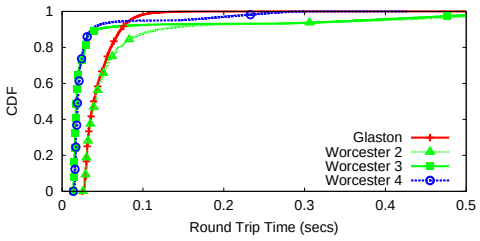
concerns when we ran the tests from Worcester1 and Waltham locations. Thus, the datasets of Worcester1 and Waltham do not contain external ping results which makes its more difficult to understand the performance of VoIP and Game application performance with our external server at these two locations.

Round Trip Time

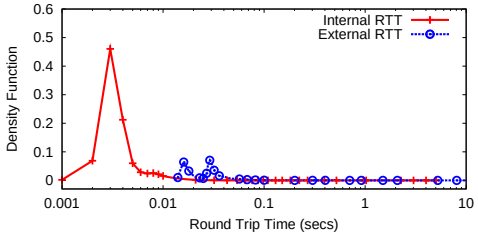
Round trip time (RTT) is one of the most important metrics to evaluate the performance of network applications. However, ICMP ping is not able to go through the firewall of the WPI campus network. Furthermore, in our previous studies [17, 34, 133], ICMP ping program packaged with Windows XP did not precisely control the packet sending rate. Thus, this study utilizes *UDP Ping*, a UDP based ping tool to measure RTT [35] between our wireless active measurement client and the Internal/External servers respectively.



(a) Internal Server



(b) External Server



(c) RTTs Measured

Figure 4.10: Round Trip Time Measured

Figure 4.10(a) depicts the CDF of RTTs between our active measurement client and the internal server which is behind the volunteer's gateway. Note, due to the system clock granularity, our UDP based ping tool only provides the degree of accuracy of 1 ms. Figure 4.10(a) clearly shows 80% of RTTs between our active measurement client and internal server are under 5 ms. Figure 4.10(c)

internal server.

shows the density function of RTTs: the internal RTTs follow a positive skewed normal distribution with the mode at 3 ms and the mean at 6 ms.

Figure 4.10(b) shows the CDF of RTTs between the active client and the external server sitting on the WPI campus network. Figure 4.10(b) illustrates that the majority of external RTTs are under 100 ms, while Figure 4.10(c) shows that the external RTTs fit a bi-modal distribution, with one peak at 16 ms and the other at 28 ms. Our external RTTs are consistent with the result reported in [139], where Maier *et al.* reported that RTTs over Germany residential network fit a bi-modal distribution, with one peak at 7 ms and the other at 45ms.

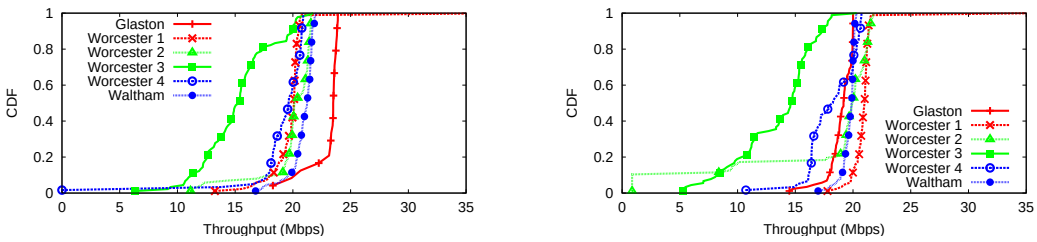
It is quite surprising that Figure 4.10(c) shows 4% of internal RTTs exceed the first peak of external RTTs, and a tiny fraction (0.07%) of local RTTs are larger than 1 second. Typically, large RTTs indicate large queuing delays. Since there is only the wireless AP between the measurement client and our internal server, it is possible that the AP queue fills up occasionally. However, because the measurement study ran with the owner's off-the-shelf AP at all locations, we are unable to determine the AP queue length to further investigate this issue. Meanwhile, the above mentioned neighbor interference which acts like hidden terminals would yield multiple MAC layer retries, and depending on the timeouts this neighborhood interference could also increase the RTTs. Last but not least, the UDP ping tool might occasionally mismatch the sending and received ACK sequence numbers and these outlier ping times could also skew the RTTs.

File Transfers

End-to-end throughput is another important metric representing the network quality. The active measurement suite measures application layer throughput between the active measurement client and the internal/external server with two FTP sessions in different directions: one downstream and one upstream.

Figure 4.11 shows the CDF of average downlink and uplink throughput between the active measurement client and the internal server, where the volunteer's AP is the only device between them. Figure 4.11(a) shows that most downlink FTP sessions reach 18-22 Mbps throughput at five of six measurement locations. Because an IEEE 802.11b based network printer is on the Worcester3 network, the Worcester3 AP involves the CTS-to-self mechanism to provide backward capability for the IEEE 802.11b printer. Thus, the maximum throughput of the FTP application measured at Worcester3 is only around 15.0 Mbps when the IEEE 802.11b printer is turned on. This is consistent

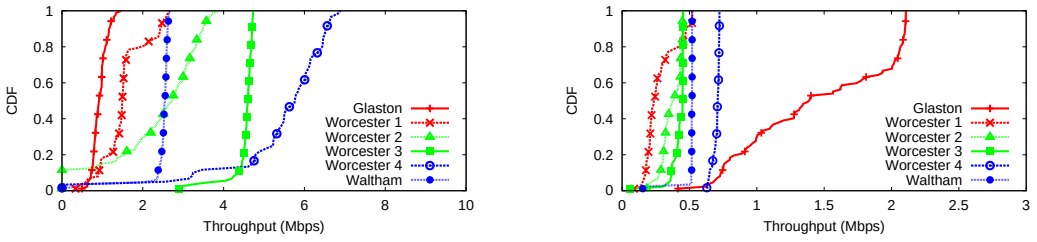
with the maximum throughput (14.7 Mbps) reported by Broadcom in an IEEE 802.11b/g mixed environment [141]. As Figure 4.11(b) shows, the uplink throughput to the internal server is similar to the downlink throughput counterpart, and the downlink and uplink throughput look symmetric inside IEEE 802.11 residential networks.



(a) FTP Download Throughput with Internal Server (b) FTP Upload Throughput with Internal Server

Figure 4.11: Throughput Measurement with Internal Server

The result of throughput measured with our external server is complicated. First of all, compared with the claimed link capacity of IEEE 802.11g AP, the users' DSL modems or cable modems are the bottleneck devices. Second, the uplink and downlink capacities of DSL or cable modems are not symmetric. For example, Worcester2, Worcester3 and Worcester4 share the same ISP, and the ISP provides 6-8 Mbps downlink capacity and 1-2 Mbps uplink capacity. Finally, we have little knowledge on the brand and model of the volunteers' cable modems, even when the users share the same ISP, their cable modems are from different vendors. Figure 4.12 shows the downlink and uplink throughput measured at the six places. Generally, the external links have higher downlink capacity than the uplink capacity. "Glaston" is the only location that has symmetric link capacity because it may have a different ISP than the other residences[2].



(a) FTP Download Throughput with External Server (b) FTP Upload Throughput with External Server

Figure 4.12: Throughput Measurement with External Server

[2]Glaston is the only residential place around Hartford, CT; the rest of them are in Worcester and Waltham, MA

VoIP Applications

The performance of the VoIP application is one of the benchmarks to evaluate ability of the network to support for multimedia applications. We use the MOS values for each talk spurt [146] as the VoIP application quality indicator, and the 10th percentile as the end-of-call quality for each VoIP session. How to apply E-Model based on the one way latency from the VoIP server to the VoIP client is a challenging task. Because of the lack of an accurate system clock, although the wireless client and servers synchronize their system clocks with the WPI NTP server every hour, there still is a 1 or 2 second clock skew between them. Thus, this study estimates the one way latency based on the RTT measured at the active measurement client, and computes the R-Value and MOS value with the estimated RTT.

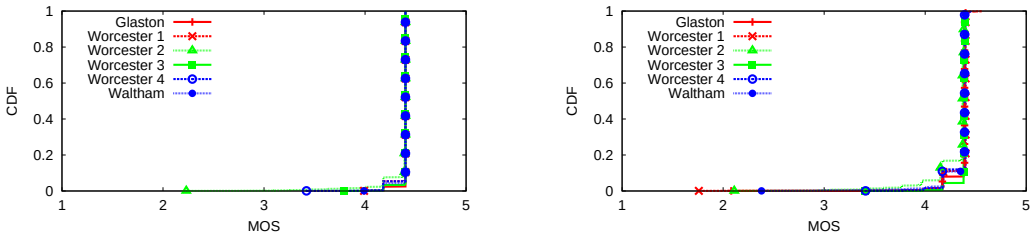
Figure 4.13(a) shows that the MOS value of 95% or more VoIP calls with our internal server is as high as 4.34 and only a couple of VoIP calls experience lower voice quality, their MOS value less than 4. The quality of 90% of the external VoIP calls are with good quality and their MOS value is also as high as 4.34 also. Note, the measurement study suffers ping failure in Worcester1 and Waltham locations because WPI network administrator shutdown the ports used by our UDP pings due to security concerns. Therefore, the average RTT of the remaining four residences is used for these two locations when calculating their MOS values.

The reason that almost all VoIPs have with good quality is because VoIP applications have low bandwidth consumption, and their qualities are vulnerable to large delays. More than 173 ms[3] one way latency significantly degrades the quality of VoIP calls. However, this study observes some occasionally more than 1 second RTTs between the internal server and the active measurement client, the median RTT between our internal server and client is under 5 ms, which is far below 173 ms.

Moreover, after inspecting VoIP log file carefully, no bursty packet loss happened during the measurement study. One of the possible reasons for no VoIP packet loss might be the majority of traffic detected over the volunteer's place is HTTP based Web flows, which are typically mice and dragonflies: small volume and short duration. It is difficult to saturate a 8 Mbps DSL modem with TCP flows [139] only and even more difficult to saturate IEEE 802.11g AP with Web flows only. Thus, with the assumption of no AP queue loss, there is a small probability that a 200 byte long VoIP frame fails to reach its destination after seven sequential transmission attempts. Therefore,

[3] 173 ms is the threshold in the Heavyside function used to compute R-factor and MOS value [146].

it is low probability that the quality of VoIP applications degrade due to bursty packet loss.



(a) Internal Server (short RTT)

(b) External Server (long RTT)

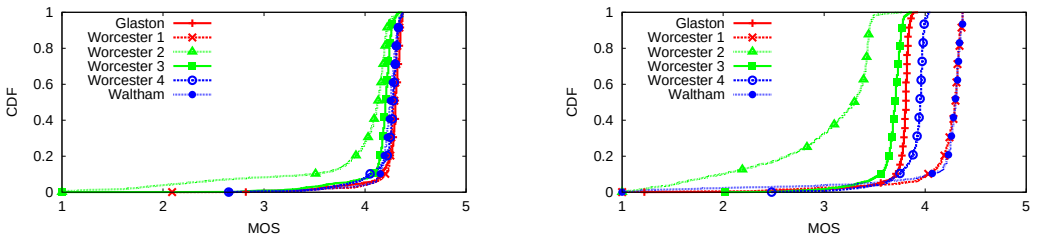
Figure 4.13: Quality of VoIP Applications

FPS Game Applications

Similar to VoIP applications, games, especially First Person Shooter (FPS) games, are another benchmark application to evaluate the performance of multimedia applications over networks. Different from the VoIP applications, the FPS games are more vulnerable to large network delays [136]. Our home measurement study emulates the Quake IV game with constant rate UDP flows, and calculates the MOS value for each game session [136] as their quality metrics. Due to the lack of a high accuracy NTP server, this study uses the same approach to compute the MOS value for game flows as it does with VoIP applications: estimating the one-way latency based on the RTTs reported by the UDP Ping tool.

Figure 4.14(a) shows the MOS values of emulated game applications running with the internal server at six residential places. Most of emulated game applications experience high QoS, and their MOS value are greater than 4.2. However, unlike the VoIP applications, game applications are more sensitive to large network latencies, and approximately 10% of the game applications experience some performance degradation. The MOS value of 20% game sessions at the Worcester2 location even drops under 4 and enters the "fair" range. The reason is that the owner attempts to copy a DVD image from one wireless laptop to a desktop through Windows SMB service. This large file transfer saturates the wireless link and builds up the queue inside the wireless AP. As a result, the game applications are severely punished by the large queuing delay. The VoIP application also experiences quality degradation and packet loss, but the quality of VoIPs does not degrade as much as game applications.

Figure 4.14(b) shows the MOS values of game applications with our external server at WPI.



(a) Internal Server (short RTT)

(b) External Server (long RTT)

Figure 4.14: Quality of Game Applications

Compared with the game quality with the internal server, the quality with the external server drops into the "fair" zone at all six locations, because one way latency between the external and our wireless client is larger than the latency between our internal server and client. However, the performance of games degrades more than VoIP, although the game flow consumes only 14.4 Kbps, which is much lower than G.711 based VoIP flows. Because of the delay sensitive nature of FPS games, it is more suitable than VoIP applications to serve as a benchmark program to measure residential wireless network performance.

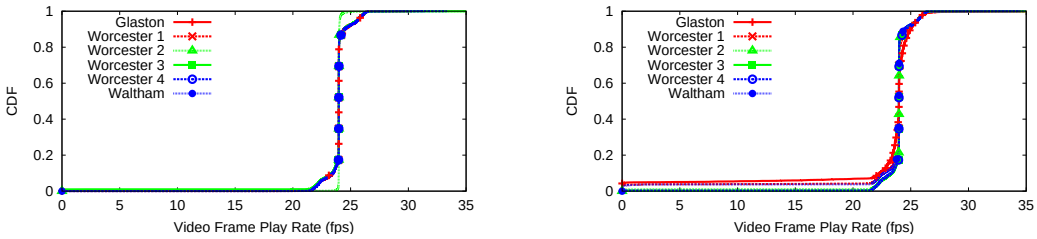
Video Applications

The active measurement suite also includes video streaming applications to evaluate their performance over residential wireless networks. Video streaming applications traditionally use UDP as a transport layer protocol. However, more and more streaming video applications choose TCP to penetrate widely deployed firewalls, and some Web based streaming applications even choose HTTP as their application layer protocol instead of traditional streaming protocol such as RTSP. This study chooses TCP based video streaming to go through the users' firewalls.

Similar to our previous video performance measurement studies [34,35], this study uses Windows Media Maker to encode two HD video clips with the same content and resolutions, one multiple layer video and the other single layer video. The highest video encoding bit rate is 1128 Kbps which is the same as the single layer video encoding rate. The purpose of using a multiple layered video clip is to take the advantage of modern media scaling and media thinning techniques [37], which are widely used by media servers.

Figure 4.15 depicts the performance of multilayer video streaming from the internal and external server respectively. The multiple layer video performs well with the internal server. Generally, the

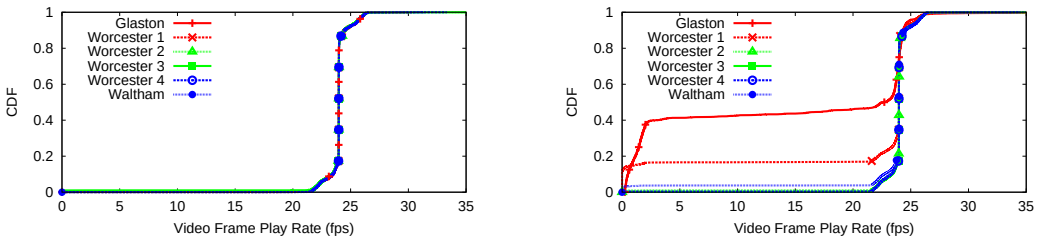
playout frame rate reaches its original encoding rate of 24 frames per second (fps), and much higher than the "good" quality threshold, 15 fps. In total, only one or two multiple layer video clips fail to play out. The multiple layer video from our external server perform a little worse than the clips from our internal server. However, less than 5% of the video clips fail to play out and 90% of the multiple video clips receive high quality.



(a) Internal Server (b) External Server

Figure 4.15: Quality of the Multiple Layer Video Clip Streaming

Compared with multiple layered video, the performance of single layer video from the internal server is almost the same as multiple layer video, but the performance of single layer video with the external server degrades. 40% of the single layer video could not stream at Glaston, and 15% of the single layer video fails at the location of Worcester1. However, the FTP throughput results in Section 4.2.4 show that the external link capacity of Glaston is approximately 2 Mbps. Thus, it is not surprising to observe video performance degradation when streaming 1128 Kbps video with a shared 2000 Kbps link. The single layer video performance degradation at the location of Worcester1 is a little bit complicated: the volunteer at Worcester1 used a wired connected desktop during the measurement period and the video performance might be degraded by "unseen" wired traffic which might cause congestion at the volunteer's gateway.



(a) Internal Server (b) External Server

Figure 4.16: Quality of the Single Layer Video Clip Streaming

However, with current residential wireless network setup, video streaming applications generally perform well, which is consistent with our previous measurement results over campus networks [17, 34, 133]. However, multiple layer video more likely performs better than single layer video on the residential networks which have smaller "last mile" ingress link capacities.

4.3 Validation NS-2 Simulator with Home Measurement Results

4.3.1 Case 1: Congested Wireless Link

One of the main objectives of the residential wireless measurement study is to find when the wireless AP is saturated by normal network usage, which would help us to design NS-2 simulations and more realistically simulate a residential wireless environment. However, most of the network activities observed in this measurement study were Web-based traffic. The sniffer traces captured in Waltham contain a long World of Warcraft game session, confirmed with the volunteer. It is surprising that no P2P (eDonkey or Bit Torrent) traffic was identified in any sniffer traces. Most of the traffic identified in the sniffer traces is not heavy enough to saturate IEEE 802.11g links. Therefore, most of the active measurement results with the internal application server do not show any performance degradation.

Moreover, because this measurement study was conducted in six volunteers' home, friends of graduate students, the selected residential places are smaller than typical American houses where the largest place in our study is under 1800 square feet. Consequently, it is difficult to find a location with "poor" wireless signal reception where the RSSI is below -75 dBm [133]. Additionally, due to the small size of the volunteers' places, only a small fraction of wireless data frames is transmitted at the rates lower than 54Mbps, and rate adaptation is rarely observed.

To avoid disturbing the volunteer's normal network activity, the active measurement test suite only injects small amounts of traffic during a short time period. It might cause congestion on the users' gateway, but it hardly congests the residential wireless APs. However, in order not to inconvenience the volunteers, this study does not inject large volume traffic to build up the queue inside AP. This made the task more difficult.

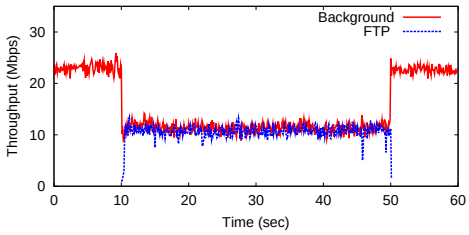
Fortunately, at least three congestion instances, one at Worcester2, one at Worcester3 and

one at Worcester4, are identified, where the volunteer's network activity might saturate their own wireless link. However, there is an IEEE 802.11b based wireless printer sitting at Worcester3, and the Worecester3 AP invokes CTS-to-self mechanism to provide backward capability for the IEEE 802.11b device. The overall throughput of Worcester3 is only 14.7 Mbps, which is at least 5 Mbps less than the theoretical throughput in 802.11g only environment. Meanwhile, since we have no plan to simulate an IEEE 802.11b/g mixed environment, and we did not consider the Worcester3 instances. Worecester2 is an IEEE 802.11g only environment, although its neighbor has one 802.11b device sitting on the same channel. The main issue is that the Worcester2 AP is not stable under heavy TCP traffic, and the downlink throughput dips approximately every 10 seconds or so. Therefore, this study did not analyze the instance found at Worcester2 because we could not fully understand what causes the periodic throughput dips.

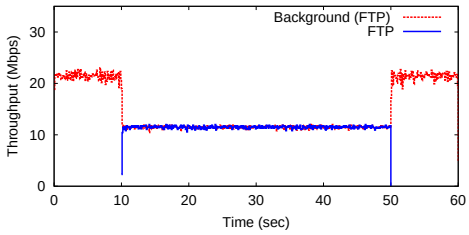
Thus, the instance of Worcester4 is the only evidence which demonstrates that normal user's network activity can saturate the residential wireless link and degrade the performance of applications running on the active measurement client. A two minute long, high volume TCP flow is identified between 15:00 PM and 16:00 PM EDT, March 31, 2009, which was concurrently running with several of the active measurement applications. Confirmed with the volunteer, the "background" TCP traffic contains a 100 second long file transfer session between one wireless client and a wireless server using Windows Samba (SMB) service, and a couple of very short and insignificant Web flows.

Figure 4.17(a) shows a 60 second snapshot of the IP throughput traffic when the FTP downloading application from the active measurement suite is concurrently running with the SMB file transfer traffic. The 40 second long FTP flow is injected by the active measurement client. Figure 4.17(a) shows that after the FTP flow starts, the two file transfer flows share the bandwidth and both of them reach 10-11 Mbps throughput. Note, because of the lack of control over the volunteer's laptop or desktop, the IP layer throughput of background traffic is calculated based on the information retrieved from the wireless sniffer traces. The sniffer was two or three feet away from the wireless AP, and it was not at the same location as the volunteer's station. Thus, even within such short distance, the wireless sniffer might still miss 5% of the frames transmitted by the wireless AP [140].

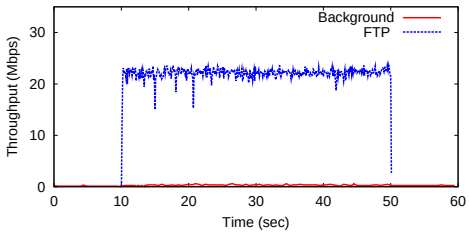
Figure 4.17(b) shows that the NS-2 simulator is able to simulate the above congestion instance. A simulated FTP flow named "background FTP" starts at the 0th second, and another FTP flow



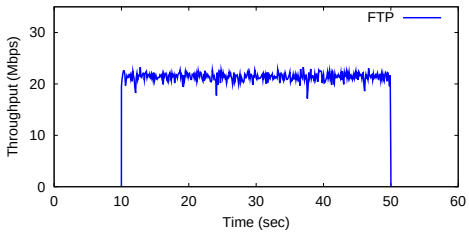
(a) FTP Downloading w/ Background Traffic



(b) Simulated FTP Downloading w/ Background Traffic



(c) FTP Downloading w/ Insignificant Background Traffic



(d) Simulated FTP Downloading w/o Background Traffic

Figure 4.17: FTP Downloading Flows with/without Background Traffic (Simulation)

joins at the 10th second. The simulation result matches with the above congestion evidence found in the wireless sniffer traces. Therefore, by carefully tuning the simulation parameters, the NS-2 simulator is able to simulate wireless network activity close to reality. Table 4.7 summarizes the parameters used in the NS-2 simulator validation. Note, Chapter 6 utilizes the same physical and MAC layer settings while several IP and transport layer settings are tuned to provide a better performance comparison between CATNAP and DropTail AP queue controllers.

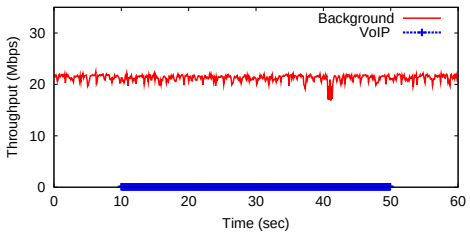
Table 4.7: Simulation Parameters Used for Validation

Layer	Parameter	Value	Notes
Physical	User-AP Distance	5 m	Wireless & AP and clients in same room
MAC	Wired Link	100 Mbps	Common Ethernet link Speed.
	Wireless Link Capacity	54 Mbps	Max. data rate for 802.11g.
	Retry Threshold	6	Default for short frames w/o RTS.
	Shadowing Deviation	7	Indoor environment, medium interference.
	Beacon Interval	100 ms	Default for most APs.
	Rate Adaptation	off	Default module for NS-2.33.
IP	RTT	3 ms	Mode of RTTs measured w/ our internal server.
	AP Queue Capacity	150 packets	Maximum queue capacity for home APs [35]
	MTU	1500 bytes	Typical setting w/o jumbo frames
Transport	TCP Version	NewReno	NS-2 fulltcp supports
	TCP CWND Max	40 packets	NS-2 default.
	TCP Nagle Algorithm	Disable	NS-2 default.

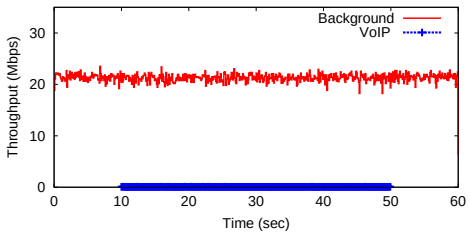
In order to have a complete comparison, Figure 4.17(c) shows that the FTP throughput without the heavy SMB traffic. The sniffer traffic trace is gathered at the same residence during 09:00-09:15 AM on March 31st, 2009. Because all residents go to work, there is no competing or contending traffic. The FTP flow reaches its expected maximum throughput around 22 Mbps, which is same as the FTP maximum throughput measured on the WPI campus wireless network [17]. Note, the tiny spike of the background traffic observed in Figure 4.17(c) is injected by the UDP ping tool which monitors the round trip time between the internal server and the active measurement client. Similar to other occasional IP layer traffic, the tiny spike can be ignored. Figure 4.17(d) shows the simulation result of a single FTP flow running alone on the NS-2 simulation testbed. Without any competing or contending traffic, the simulated FTP flow reaches its maximum throughput as 22 Mbps, which is consistent with the measurement result shown in Figure 4.17(c).

4.3.2 Case 2: VoIP under High Volume Background Traffic

In addition to validating the simulation setup with two non-interactive FTP flows, this study also compares the performance of simulated VoIP flows with VoIP flows identified in the wireless sniffer traces. Figure 4.18(a) depicts the measured throughput results of VoIP and background traffic including the SMB flow, which is from the same sniffer trace used in the previous section. Figure 4.18 shows that the throughput of simulated traffic is similar to the measurement result. Figure 4.18(b) shows the throughput results of VoIP and background flows simulated with the NS-2 simulator. Note, the background traffic in Figure 4.18(a) is the data flow from SMB windows file sharing, and the background traffic in Figure 4.18(b) is the FTP flow generated by NS-2.33 simulator.



(a) VoIP with Heavy Background Traffic

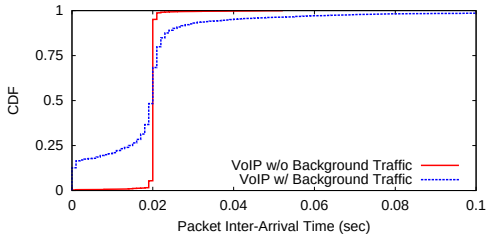


(b) Simulated VoIP with Heavy Background Traffic

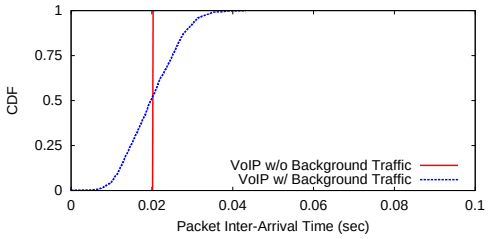
Figure 4.18: Throughput Comparison

Figure 4.19(a) and Figure 4.19(b) compares the CDF of packet inter-arrival time between the

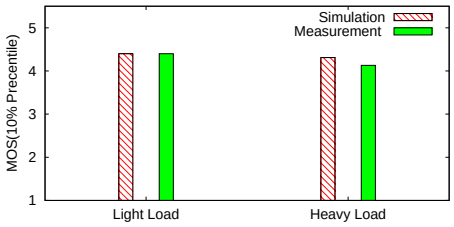
simulated and measured VoIP flows. Because of the clock skew problem mentioned earlier, it is not suitable to compare the skewed one way delay of measurement result with simulation results which do not have the clock drifting problem. Thus, Figure 4.19(a) chooses the packet inter-arrival time instead of one way delay to avoid the clock skew problem. Figure 4.19(a) and Figure 4.19(b) also compare the CDF packet inter-arrival time of simulated and measured VoIP flows without high volume background flows. When no high volume background flows, 99% of packet inter-arrival time is 20 ms which is the packet sending interval specified by G.711 standard. There is almost no jitter in both the simulated and measured VoIP flows when the wireless link is not saturated.



(a) Packet Inter-arrival Time of VoIP Flows



(b) Packet Inter-arrival Time of Simulated VoIP Flows



(c) MOS Values of VoIP Flows

Figure 4.19: VoIP Quality

When VoIP flows are running with the stress of high volume background traffic, the CDF of packet inter-arrival time shown Figure 4.19(a) is similar to what is shown in Figure 4.19(b). But, around 20% of packet inter-arrival times is 0 ms, which means 20% of the actual VoIP packets arrive back-to-back while simulated VoIP flows do not have any back-to-back packets. This phenomenon could be explained as side effects from the Windows device driver implementation. Due to the small length of VoIP packets, the device driver is able to buffer several small packets and pass them together to application layer within one interrupt. This kind of implementation generally saves CPU time on I/O related operations, but it potentially impairs the performance of real-time applications. As Figure 4.19(c) shows, however, because the packet inter-arrival time of the measured VoIP is

similar to the simulated VoIP flow, the qualities (MOS value) of simulated and measured VoIPs are almost same.

Section 4.3.1 and Section 4.3.2 demonstrate that NS-2 simulator is able to simulate residential wireless network accurately by carefully designing the simulation testbed and tuning the simulation parameters.

4.4 Summary

With the spread of high speed broadband network activity to the home, the interest in home networks has increased in the last decade. Numerous researchers have measured residential network performance [106, 107, 139, 143, 144, 147–149]. However, most previous studies have focused on the measuring Internet access connections to the home [107, 139], which are often large scale by analyzing traces gathered from ISP's gateways or servers. Other studies have focused on management and trouble shooting techniques by measuring the performance and utilization of home networks [148,149]. Although the performance of home wireless networks has been received substantial attention in the past few years, few studies have been done with residential wireless networks. Especially when we designed our home measurement study in 2008, there were not any home wireless measurement study results public available. The closest research work to our goal is the research done by Papagiannaki *et al.* [147] and Dicioccio [106]. However, neither of them provides a picture of home wireless networks to cover a complete network stack, from physical layer to application layer as does our measurement study.

Through this home wireless measurement study, we find the answers for several questions which we could not find answers from other researchers, such as:

1. **What the current home wireless network looks like?** Because this study only involves six volunteer families, and most of them are in their early 30's, we could only provide a snapshot of current usage of home wireless network. The home wireless network are different from enterprise or campus network: most of six volunteers are only active during early evenings. Majority of identified flows are Web based.

2. **What kind wireless frames are traveling in air?** During the early evening, the most active time of week day, most of wireless frames are data frames, and control frames. However, during the quiet time such as early morning, the majority frames are control and management

frames.

3. **Do the neighbors' APs effect the performance of my wireless network?** The short answer is *yes*. The neighbor's AP or other wireless device such as wireless printer, might have influences on your wireless network. However, it might be difficult to quantify their influences.

4. **What kind of applications are running on wireless link?** We observe Web, game (WoW) traffic, DNS, and several video sessions.

5. **How does a real time application perform over home wireless link?** Generally speaking, the VoIP and game application in our active measurement suite perform well.

In addition, the home wireless measurement results help us to design the NS-2 simulation testbed and choose simulation parameters to realistically simulate home wireless networks. As Section 4.3 shows, by carefully tuning the simulation parameters, NS-2 is able to simulate traffic behaviors seen on residential wireless network fairly well.

At the end of this chapter, we conclude some high level experimental lessons and suggests future research for home wireless measurement study,

1. **Measurement point must be inside home.** The absence of home network measurement data, especially for home wireless network data, is due to the difficulties of gathering home network data on a large scale. Because the majority of home networks are behind firewalls and network address translators, any observation point outside the residence places can not measure the characteristics of home networks themselves. The traffic traces gathered on the ISP's gateways can provide statistics on connection links, but can not provide information inside home. Without deploying physical devices (e.g., a sniffer) inside home, there is no way to provide the MAC layer information for home wireless networks.

2. **Host AP needed.** Sundaresan *et al.* select the gateway as their observation point for home networks [144], yet they could not provide any wireless MAC information for home wireless networks. Most off-the-shelf wireless APs provides four or more wired ports, and they act as gateways in most volunteers' home seen in this study. However, because the residential class APs usually do not provide packet capture functions like high-end enterprise level APs, simply using residential wireless APs as observation points can not provide any MAC layer information. Host AP [35], a Linux based wireless AP, might be a good choice for residential

wireless network studies. By modifying the Linux driver, the host AP could record every IEEE 802.11 frames sent and received. Meanwhile, because a host AP can also record the traffic on its wired port, it can solve the problem of monitoring the wired link. Last but not least, the host AP avoids the encryption schemes (WPA-2) widely used in APs since inside the device driver, all MAC frames are plain text. A group of researchers from University of Wisconsin [31] deployed 30 OpenWrt-based WiFi APs, which is similar to host APs, with enhanced measurement and analysis software into volunteers' home, and studied the home wireless experience through their unique APs.

3. **Wireless sniffer is not entirely effective.** In our measurement study, our wireless sniffer still misses up to 5% of the frames even when the wireless sniffer sits just 1 or 2 feet away from the home AP. Though Schuman *et al.* [140] provide a way to estimate the uncaptured wireless frames, their method could not provide a valid "ground truth" for wireless traffic. Moreover, IEEE 802.11n has become a dominant product recently. Because an IEEE 802.11n AP uses two dynamic channels for transmission and reception, it is more difficult to use a sniffer to monitor home wireless usage. Thus, a host AP or an AP with a "dump" function is a better choice for home wireless measurement study than a commercial wireless sniffer.

4. **Respect users' privacy.** Unlike measurement studies in well controlled environments, or measurement studies for a campus wireless network [17, 34, 133], the volunteers involved in these study all expressed privacy concerns in some degree. A couple of volunteers actually intentionally avoided using their wireless network during this study, although we explained that the wireless sniffer only collected packet headers and not packet payloads. Because of privacy concerns, users may change their use of their wireless network; for example, we were unable to identify a large volume of P2P traffic, perhaps because users stopped using P2P downloading through wireless links after they knew the existences of the wireless sniffer.

In addition to the above lessons and conclusions, Sundaresan *et al.* also list three lessons: *"One measurement does not fit all"*, *"One ISP does not fit all"*, and *"Home network equipment matters"* in [144], which our measurement experiences confirm. Due to the nature of a home wireless measurement study, it is difficult to determine which set of homes is representative. Actually, finding a large number of volunteers itself is a challenging problem in itself. Thus, the combination of a large scale ISP based measurement study and a wireless measurement study in a small representative

group might be a feasible solution.

Chapter 5

CATNAP

This chapter describes our approach to improve the QoS of applications running over residential wireless networks by utilizing a treatment-based traffic classification technique. CATNAP focuses on IEEE 802.11 infrastructure networks in residential places. It infers application types by observing the characteristics of IP flows, and applies the treatment policies inside the IP and the transport layer. The major advantage of working inside the network layer is to make CATNAP independent from wireless network interface hardware and device driver implementations. Moreover, the classification techniques applied to the approach include statistics-based classification techniques. As discussed in Section 3.1, compared with other classification techniques such as port or payload-based approaches, statistics-based classification techniques can provide the fast convergence times and low computation power necessary for real-time processing for an AP. Finally, security and privacy related issues are assumed to not be an issue for this study, because home users have full access to any traffic content on their own networks. Thus, the CATNAP classifier has full access to IP and transport layer headers. On the other hand, malicious applications are assumed to be blocked by users' firewalls and are not treated as normal applications. Thus, we consider that viruses or any other malicious applications are out of the scope of this study.

The CATNAP approach contains three main components:

1. Classification Modules: a set of classifiers with high accuracy and low convergence time for applications running over residential wireless networks.

2. Treatment Modules: shape traffic passing through an AP in order to provide better QoS

support.

3. Utility Functions: functions such as downlink effective capacity estimator and round trip time estimator which provide necessary information for CATNAP treatment and classification modules.

The rest of this chapter is organized as: Section 5.1 starts with a discussion of the possible traffic treatments which could be deployed on resource-limited wireless access points (APs); Section 5.2 describes three utility functions: the downlink effective capacity estimator, round trip time (RTT) estimator, and per-flow information tracker, which CATNAP depends on; Section 5.3 gives a set of novel treatment-based traffic classification criteria in CATNAP; Section 5.4 presents implementation details on the CATNAP treatment modules; and Section 5.5 summarizes the content of this chapter.

5.1 Possible Treatments

Unlike backbone routers or commercial class wireless APs, residential APs are equipped with less powerful processors. As mentioned in Section 2.2.3, the processor inside a Link-Sys WRT54G wireless router is based on Broad-Com BCM5352 chip [150]. It contains a 200-MHz MIPS32 CPU core with only a 16 KB instruction cache and 8 KB data cache, plus a SDRAM controller which supports only up to 256 MB memory. Enterprise class wireless APs also use the low-end processors but they are more powerful than the ones in residential APs. For instance, the IBM PowerPC 405 200MHz chip [151] in Cisco Aironet 1240 series APs, has its highest performance at 1.52 DMIPS/MHz with a 4 GB data and instruction address space[1]. On such under powered processors, especially for residential class APs, it is difficult to implement complicated traffic shaping and traffic classification schemes.

Roughan *et al.* describe a classification framework to differentiate applications into several categories based on statistics-based application signatures [22]. Claypool *et al.* propose a classification scheme to differentiate applications into different QoS groups based on the *nature of traffic* [30]. Inspired by this QoS related classification work, we implement a treatment-based classification method which allows APs to treat equally application flows that have similar QoS requirements providing better QoS support over the wireless links.

[1]The Cisco Aironet 1242 A/G AP is only equipped with 16 MB memory plus 8 MB flash memory.

Our previous study [35] discovered sub-IP layer transmission queues inside wireless APs. For residential wireless APs, these queues are packet based with maximum capacities ranging from 50 to 120 packets[2]. Based on this knowledge, we propose five traffic treatment [30] schemes which can be implemented with these sub-IP layer queues inside the AP. These five types of treatments are:

1. **Drop Packets.** For some flows, packets can be dropped if the AP is congested as these applications are more tolerant to lost data. For example, when the transmission queue inside a wireless AP nears capacity, a small fraction of packets from a streaming flow over UDP can be dropped without serious degradation to quality.

2. **Limit Transmission Rate.** When congestion occurs, an AP can limit the advertised window size carried by ACK packets to force the TCP sender to lower its transmission rate. This treatment method can avoid the costly retransmission of TCP packets over a long RTT connection.

3. **Push or Delay Packets.** Packets in interactive flows can be pushed or placed at the head of the transmission queue to reduce latency. For example, packets from VoIP applications can be put in the front of the queue when congestion occurs. Packets from non-interactive flows can be delayed slightly without impairing the applications' performance. For instance, P2P file downloading can take hours. Thus, it is possible to delay P2P packets and give higher priority to the packets of time sensitive applications, such as VoIP or games.

4. **Reserve Bandwidth.** The bandwidth usage of some applications is limited by their implementations, for example the video codec usually decides the bandwidth usage of video streaming flows. Therefore, we can reserve a fraction bandwidth for these flows. Moreover, we also can allocate and reserve bandwidth to help TCP flows in their slow start phase.

Based on the queue implementation inside the Linux kernel [108], these five operations do not require CPU intensive operations, only needing several pointer operations and some additional statistical information for each flow. Thus, these operations can be done within the packet-based queue inside APs without significantly changing the IEEE 802.11 MAC layer protocol stack or current architecture of wireless APs. Moreover, unlike backbone routers, there are only dozens of flows concurrently passing through an AP in most residences. In fact, most wireless access

[2]The queue capacity inside commercial APs is from 50 to 350 packets [35].

devices only support hundreds of concurrent connections. Even the RTL 8186, a commercial class wireless gateway, can only support 1024 concurrent connections as well as a maximum of 64 client associations [152]. Accordingly, we assume that the wireless AP has enough resources (processor power and memory space) to perform these operations and maintain flow state information.

5.2 CATNAP Supporting Functions

This section presents several functions on which CATNAP depends. CATNAP includes a flow-based classifier and a treatment module, and it needs to maintain flow level information for traffic passing through an AP. Typical wireless APs only forward packets from wired link to wireless link and vice versa, and would not maintain per-flow level statistics information. Moreover, the CATNAP treatment functions require some system level information such as how many concurrent flows are currently active at the AP in addition to per-flow level information. Therefore, a set of utility functions provide support for the functionality of CATNAP. These supporting functions include: a system and flow information tracker, a downlink capacity estimator, and a round trip time (RTT) estimator. Although these supporting functions provide the foundation of CATNAP, they can be replaced with third party algorithms with better implementations in future.

5.2.1 Statistics Information Tracker

Residential wireless APs usually act as IP routers in addition to bridges between wired and wireless networks. A wireless AP can access the MAC layer, IP layer and Transport layer, but normally most residential wireless APs do not provide flow level information aggregation. Table 5.1 summarizes the set of functions to retrieve information from packet headers of IP packets and functions to update the IP headers used by CATNAP.

This study enhances the wireless AP NS-2.33 implementations to keep per-flow statistics and flow classification results. Table 5.2 lists the per-flow information entries and their descriptions tracked by CATNAP. The per-flow information can be categorized into three categories,

1. Flow information: flow id i, start timestamp t_{start}, last active timestamp t_{last}, round trip time RTT, flow ingress rate r, and EWMA packet Length $\overline{L}$.

2. CATNAP classification result flags: $isResponse$, $isInteractive$, $isGreedy$, $isActive$, and $isEnd$.

Table 5.1: Functions to Manipulate IP Packets

Name	Description
get_pkt_length(packet p)	get packet length including IP headers.
get_src_ip(packet p)	get sender's IP address.
get_dst_ip(packet p)	get receiver's IP address.
get_src_port(packet p)	get source port number.
get_dst_port(packet p)	get destination port number.
get_protocol(packet p)	get transport layer protocol.
get_awnd_size(packet p)	get advertised window size carried by TCP packet.
isTCPCtrl(packet p)	returns **true** for TCP control only packets, otherwise returns **false**.
isTCPACK(packet p)	returns **true** when TCP ACK flag is set, otherwise returns **false**.
isTCPSYN(packet p)	returns **true** when p is TCP SYN packet, otherwise returns **false**.
set_awnd_size(packet p, long $newsize$)	set advertised window size ($newsize$) for TCP.
update_crc(packet p)	recalculate TCP and IP CRC for TCP.

3. CATNAP treatment parameters: Advertised Window Size $AWND$ and Drop Probability d.

The per-flow record is implemented with a hash map to achieve linear searching time, where the flow id i is used as the hash key, and the pointer to the corresponding flow entry is stored as a hash value. Meanwhile, CATNAP provides a set of getter and setter APIs to maintain the flow records. Several of the flow level statistics such as EWMA packet length are updated every packet while others are calculated every small time period (as an epoch), for example flow rate (r) and advertised window size ($AWND$). The main purpose to introduce epoch is to reduce the computational overhead for rate related metrics, such as flow ingress rate and link capacity estimation, Section 5.2.2 and Section 5.4.2 have a discussion on the epoch value chosen.

In addition to the per-flow information maintained by the CATNAP, Table 5.3 summarizes the system-wide information the CATNAP needed. The most important system metric is the downlink capacity (C) which is calculated by Algorithm 1 in addition to the counters associated with different type of flows.

5.2.2 Link Capacity Estimator

One key function of CATNAP is to perform rate control for flows passing through the AP. CATNAP needs to estimate the downlink capacity used to calculate dropping probability for non-response-based flows and the advertised window size for response-based flows. CATNAP includes a light weight bandwidth estimator to provide realtime downlink capacity for classification and treatment modules. However, most bandwidth estimation algorithms are designed for end-to-end link path estimation [37] and computational model-based estimation approaches are generally either com-

Table 5.2: Flow Information Stored by CATNAP

Symbol	Description
i	flow id, a hash value of <srcip, srcport,dstip, dstport, proto>.
$isExist$	flag indicates a flow whose information has been stored by CATNAP.
$isEnd$	flag indicates a terminated flow; *true* for an alive flow, *false* for a terminated flow.
$isActive$	flag indicates active flow; *true* for an active flow, *false* for an inactive flow.
$isResponse$	flag indicates response-based flow; *true* for a response-based flow, *false* for a non-response-based flow.
$isInteractive$	flag indicates interactive flow; *true* for an interactive flow, *false* for a non-interactive flow.
$isGreedy$	flag indicates greedy flow; *true* for a greedy flow, *false* for a non-greedy flow.
$isEmpty$	flag indicates that some packets from flow i are enqueued.
$isNewBorn$	flag indicates a new detected flow; *true* for a new detected flow, *false* for a flow has been classified as Response/NonResponse-Based.
RTT	RTT measured by RTT estimator.
t_{start}	timestamp when the first packet of this flow received.
t_{latest}	timestamp when the latest packet received.
l	queuing delay when TCP SYN is enqueued.
r	Flow ingress rate in the most recent epoch.
L	Total length of packets received in the most recent epoch.
	$\sum_{j=1}^{n} length(p_j$ for n packets received in the most recent epoch.
$\overline{L}$	EWMA packet length.
$AWND$	Advertised window size (TCP based flow only).
d	Packet dropping probability. (UDP based flow only.)

Table 5.3: System Information

Symbol	Description
C	Downlink Capacity.
R_{nint}	Fairshare rate for non-interactive flows.
R_{greedy}	Fairshare rate for greedy flows.
S_b	Total queue size in bytes.
S_p	Total queue size in packets.
n	Total number of packets sent in last epoch.
N_{act}	Total number of active flows.
N_{nact}	Total number of inactive flows.
N_{int}	Total number of interactive flows.
N_{nint}	Total number of non-interactive flows.
N_{greedy}	Total number of greedy flows.
$N_{ngreedy}$	Total number of non-greedy flows.

putationally intensive or lack accuracy. Therefore, we propose a light weight downlink capacity estimator (Algorithm 1), which can be easily implemented on resource limited home wireless APs.

The core idea of Algorithm 1 is to measure the actual sending time (Δt) for each IP packet inside the IP layer dequeue function. For example, at time t_s, the AP queue controller removes the IP

packet p and passes it to the underlying wireless driver. At time t_e, after finishing the transmission of p and receiving the corresponding IEEE 802.11 ACK frame, the wireless device driver returns the control back to the AP queue controller through a callback function, and the AP queue controller retrieves the next IP packet if the queue is non-empty. The actual sending time (Δt) is time elapsed between when the packet dequeued from IP queue and when the control returned to IP layer queue controller from the callback function.

The instantaneous link capacity C' can be estimated by the actual sending rate of the packet p:

$$C' = \frac{get\_pkt\_length(p)}{\Delta t} = \frac{get\_pkt\_length(p)}{t_e - t_s} \tag{5.1}$$

where $get\_pkt\_length(p)$ returns the size of the packet p and Δt is the actual sending time which is the time difference between when packet p is dequeued from IP queue at the time t_s and when the device driver call back function returns to the IP queue controller at t_e for packet p. C' should be close to the theoretical IP layer throughput over wireless link because once the wireless device driver receives packet p, the device driver transmits the packet. The actual sending time (Δt) includes the overhead from transmitting an IP packet over the wireless link, such as possible retransmission overhead, rate adaptation overhead, exponential back off and even overhead from beacon frames. From the view of recipient and sender, it takes Δt to deliver the packet p from the wireless AP to its destination.

In order to assimilate overhead introduced by occasional system events, CATNAP calculates the average effective capacity C_{epoch} at the end of each epoch time period as Line 3 in Algorithm 1 shows, where $\sum_{k=1}^{n} length(p_k)$ is the summation of the length of all n packets sent in the epoch, and the $\sum_{k=1}^{n} \Delta t_k$ is the summation of actual sending time of all IP packets sent in the epoch. Although we calculate the downlink capacity C_{epoch} every epoch in our first implementation, C_{epoch} still oscillates over time with a large ω. Thus, Line 4 of Algorithm 1 introduces an exponential weighted moving average (EWMA) to reduce the variance of the calculated downlink capacity.

The initial value of effective capacity (C_0) is set as 25 Mbps which is the maximum value we observed in our previous measurement study [17] as well as in our initial simulation studies. If there is no IP packet transmitted during an epoch time period, the estimator simply assumes that the wireless condition has not changed in the last epoch time period and carries over the effective capacity (C) from the previous epoch.

Algorithm 1 Estimate Effective Link Capacity

At the end of each epoch,

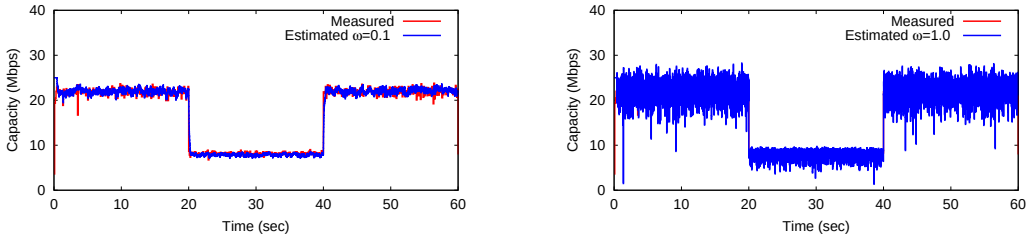
1: $n \leftarrow get\_total\_pkt\_sent()$ ▷ get total number of packet sent in this epoch.
2: **if** $n > 0$ **then**
3: $C_{epoch} \leftarrow \frac{\sum_{k=1}^{n} get\_pkt\_length(p_k)}{\sum_{k=1}^{n} \Delta t_k}$ ▷ compute instantaneous capacity
4: $C \leftarrow (1-\omega) \times get\_link\_capacity() + \omega \times C_{epoch}$
5: **else**
6: $C \leftarrow get\_link\_capacity()$ ▷ No packet, so use previous estimate.
7: **end if**
8: $set\_link\_capacity(C)$ ▷ Save link capacity into CATNAP system variable table.

Parameter Selection for Capacity Estimator

Our initial implementation shows that the selection of EWMA weight (ω) for effective capacity and the length of epoch time have significant effect on the performance of Algorithm 1. The epoch selection, however, not only impacts the effective capacity estimation but also affects the calculation of the CATNAP treatment policy maker. Therefore, Section 5.4.2 focuses on the selection of epoch time and this section focuses on the selection of EWMA weight (ω) used by our effective capacity estimator, Algorithm 1.

In order to assist the selection process of EWMA weight (ω), we designed a simulation setup to evaluate our capacity estimator. An FTP flow starts transmitting 1500 byte packets from a wired node to a wireless client when the simulation starts. The wireless link speed is reduced from 54 Mbps to 11 Mbps at the 20th second and is reset at the 40th second. In this way, two dramatic link speed change events are introduced to assist the selection of EWMA weight (ω) of Algorithm 1 under various wireless link capacities. An alternative way to change the wireless condition is to move simulated wireless node away from the AP, but mobility of wireless nodes would introduce more uncontrolled variables to our simulation and increases the complexity of simulation analysis.

Figure 5.1 visualizes the effects from EWMA weight (ω) on the effective capacity estimation results by comparing the results of Algorithm 1 with $\omega = 0.1$ and $\omega = 1.0$, respectively. Note, Figure 5.1(b) actually depicts the instantaneous capacity (C_{epoch}) where the weight (ω) is set as 1.0. Clearly, the effective capcity reported with $\omega = 1.0$ is oscillating and has more variance than the effective capacity reported with smaller ω as 0.1. Except for the intentionally introduced link capacity changes at 20th and 40th second, there is not any other sudden wireless condition changes such as node mobility. Thus, the link capacity (C) should not have high variance except for the



(a) Capacity Estimator with $\omega = 0.1$ (b) Capacity Estimator with $\omega = 1.0$

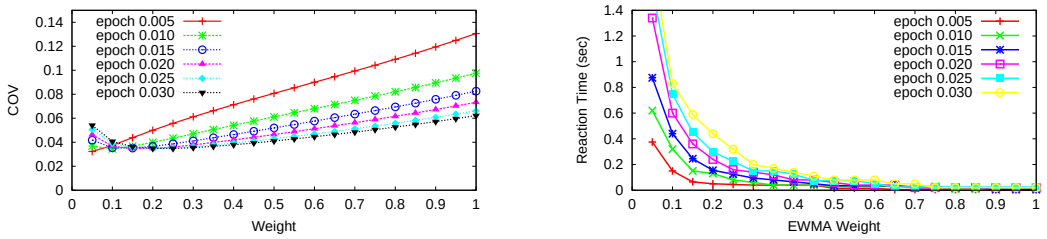
Figure 5.1: Comparison of Capacity Estimators with $\omega = 0.1$ and $\omega = 1.0$

time period around the 20th and 40th second when resetting wireless link speed.

This study chooses *stability* and *small reaction time* as two major selection criteria to evaluate the performance of Algorithm 1. Stability means that the capacity estimator should provide a stable estimation, not oscillating all the time, otherwise the fluctuating estimated capacity would introduce more difficulties for our treatment module implementation. Meanwhile, when the link condition dramatically changes, the estimator should be able to detect the wireless changes in a short time period for the CATNAP AP to adapt its treatment policies promptly.

Because there is no commonly accepted metric for stability of estimated wireless link capacity, this study chooses the Coefficient of Variation (CoV) of estimated capacity as an indicator of stability. Figure 5.2(a) shows the CoV of estimated capacity under different EWMA weights (ω) and epoch time periods between 5 ms and 30 ms. Although Section 5.4.2 mainly discusses the selection of epoch time, we still choose several possible epoch values between 5 ms to 30 ms to help understand the behavior of our effective link capacity estimator under the combination of different EWMA weights and epoch time period. Each curve in Figure 5.2(a) represents the Coefficient of Variation (CoV) of estimated link capacity with various EWMA weights with a pre-selected epoch value in range of 10 ms and 30 ms. Figure 5.2(a) shows a knee point exists between $\omega = 0.1$ and $\omega = 0.2$ for all curves. The CoV of estimated link capacity increases when ω is greater than 0.3. Thus, under the given epoch time period values, the EMWA weight (ω) should be between 0.1 and 0.2 to have a stable link capacity estimation result.

On the other hand, Figure 5.2(b) depict the reaction time under the various combination of the EWMA weights (ω) and epoch time periods. The reaction time is the difference between the moment when the estimated capacity is below or above a preset threshold and the moment when the link speed is set to a new value. Gretarsson *et al.* found that the maximum IP layer throughput over



(a) Stability (CoV)

(b) Reaction Time

Figure 5.2: Capacity Estimator with Different ω

802.11b network is only up to 9 Mbps while the maximum IP layer throughput over 802.11g is above 20 Mbps [17]. Therefore, as the physical link capacity reduces from 54 Mbps to 11 Mbps, this study measures the reaction time as the time difference between when the CATNAP effective capacity estimator first reports the effective capacity below 9 Mbps, and the 20th second. Similarly, the reaction time is measured as the time difference between the CATNAP effective capacity estimator first reports the effective capacity above 20 Mbps and the 40th second.

Similar to the curves of stability shown in Figure 5.2(a), each curve in Figure 5.2(b) shows the reaction times of Algorithm 1 with different EWMA weights and one fixed epoch time. Figure 5.2(b) shows that the reaction time of Algorithm 1 are not sensitive to the selection of epoch time period between 5 ms and 30 ms because all curves are in similar shape. Meanwhile, given a particular epoch time, e.g. 10 ms, a larger EWMA weight such as 0.6 results in a quick reaction time while a smaller EWMA weight gives a slower reaction time.

Figure 5.2(a) and Figure 5.2(b) show that:

1. a smaller EWMA weight (ω) usually provides a stable effective capacity estimator with smaller CoV.

2. a larger EWMA weight (ω) generally results in a faster reaction time, and it allows Algorithm 1 to detect link speed change promptly.

3. neither reaction time and stability of Algorithm 1 is sensitive to the epoch time period in a given range between 10 ms and 30 ms.

Thus, this study selects the EWMA weight as 0.1 for Algorithm 1 to balance reaction time and stability. The relatively small EWMA weight as 0.1 can provide a stable effective capacity estimation which helps the implementation of the CATNAP treatment module, as well as have a

reasonable reaction time to capture the link speed variations. The reasonable reaction time actually helps Algorithm 1 to be able to assimilate occasional wireless condition changes and avoid reporting oscillating link capacity.

5.2.3 RTT Estimator

Round Trip Time (RTT) is another important network metric which can significantly effect the performance of applications. This section explains how CATNAP estimates RTT for response-based flows (TCP flows). However, although the CATNAP treatment modules do not need the RTT for non-response-based flows (UDP flows), the RTT estimator sets the RTT for non-response-based flows to the average RTT for Internet flows reported in [153] and [139].

The key part of the RTT estimator (Algorithm 2) derives from the well-known TCP SYN, SYN-ACK, ACK three-way handshaking approach [106,107,139,143]. CATNAP measures the time difference between when it detects the first TCP SYN segment from the TCP initiator to the TCP responder and when it detects the first TCP ACK segment from the TCP initiator.

However, CATNAP treats the first TCP SYN packet as a packet from an non-interactive and greedy flow because CATNAP can not decide flow characteristics through a single TCP SYN packet. CATNAP would not transmit the SYN packet until it sends all enqueued packets, and the estimated RTT through the TCP SYN-ACK method includes the possible queuing delay. Therefore, CATNAP introduces a small adjustment step to subtract the queuing delay from estimated RTT in Algorithm 2. After detecting the first TCP SYN packet of a new flow, CATNAP records the SYN packet arrival time ($t_{syn,i}$) as well as retrieves the current queue length (S_b) in bytes and the current effective capacity (C) estimated in Algorithm 1. As Line 16 in Algorithm 2 shows, CATNAP calculates the estimated queuing delay (l_i) for the SYN packet for flow i as:

$$l_i = \frac{S_b}{C} = \frac{get\_queue\_length\_in\_bytes()}{get\_link\_capacity()} \tag{5.2}$$

where i is the flow id for the SYN packet, C as the estimated capacity and S_b is current queue size in bytes when the SYN packet arrives.

For non-response-based (UDP) flows, there is no reliable way to estimate RTT because of their connectionless nature. However, our current treatment module implementation does require the RTT of non-response-based (UDP) flows when making treatment decisions. Therefore, we simply

Algorithm 2 Estimate RTT

```
1: for each arrival packet p do
2:     i ← get_flow_id(p)
3:     if isExists(i) = false then                          ▷ Detects a new flow i
4:         create_entry(i)
5:         set_newborn(i, true)
6:     end if
7:     if get_rtt(i) > 0 then
8:         return                                           ▷ The RTT of flow i has been measured.
9:     end if

10:    if isResponse(i) = false then                        ▷ Non-response based flows usually are UDP flows.
11:        RTT_i ← RTT_udp ← 200 ms
12:        set_rtt(i, RTT_i)
13:    else
14:        if isTCPSYN(p) = true then                       ▷ first TCP SYN packet detected.
15:            t_syn ← now()
16:            l_i ← get_queue_length_in_bytes() / get_link_capacity()
17:            set_timestamp_syn(i, t_syn)
18:            set_queue_latency(i, l_i)
19:        else if isTCPACK(p) = true then                  ▷ first TCP ACK packet detected.
20:            t_ack ← now()
21:            RTT_i ← t_ack − get_timestamp_syn(i) − get_queue_latency(i)
22:            set_rtt(i, RTT_i)
23:        end if
24:    end if
25:    return
26: end for
```

use a fixed value 200 ms as the default RTT for non-response-based flows (RTT_{udp}) in order to maintain completeness in Algorithm 2. The 200 ms RTT is measured by Maier *et al.* in Germany residential places in [139]. Coincidentally, PingER [153] also reports the RTT measured between *EDU.SLAC.STANFORD.N3* and worldwide sites with mean as 237 ms and median RTT as 210 ms. In the future, if any new treatment method requires the RTT from non-response-based flows, we may estimate the RTT for non-response-based flows by measuring the RTT of response flows between the same pair of nodes.[3]

[3]For example, RTSP, a video streaming protocol, always uses response-based (TCP) flows to carry control messages and uses response-base (TCP) or non-response-based (UDP) to deliver the video content. In this case, we can use the RTT of control flows to estimate the RTT of non-response-based (UDP) flows.

5.3 Treatment-Based Classification

This section describes the treatment-based traffic classification method in detail. Unlike other classification methods, the objective of CATNAP is not to identify which particular application generates a particular flow, but to develop a better and more effective matching strategy between the resultant flow signatures and the potential treatments that need to be applied to wireless traffic flows over residential networks. Thus, the traffic characterization in CATNAP is based on three treatment based classifiers in terms of the nature of the traffic. In addition to three treatment based classifiers, we also introduce an active/inactive classifier to differentiate the inactive flows from active flows to improve the link utilization.

5.3.1 Identify Response-Based/Non-Response-Based Traffic

Wireless APs act as central communication points in most residential networks. Because all traffic passes through the wireless AP with a single Internet connection, we can assume that asymmetric routing does not exist in residential networks. Thus, wireless APs can capture all residential traffic and observe whether a flow is uni-directional or bi-directional.

An IP flow can be identified by the five-tuple:

$$(IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, Protocol).$$

where port numbers are only used to identify a single flow and are not used to determine application type. A *Reverse flow* is defined as:

Definition 1 (Reverse Flow) *Flow* $A : (S, X, D, Y, P)$ *is the reverse flow of Flow* $B : (D, Y, S, X, P)$.

Based on the definition of reverse flow, we describe a *response-based flow* as a flow where packets flowing in one direction directly or indirectly (e.g., delayed TCP ACKs) cause packets to flow in the reverse direction. In a TCP connection, the flow (data flow) from the active peer to the passive peer is response-based. Meanwhile, the other flow (ACK flow) from passive peer to active peer is also a response flow of the data flow.

Response-based flows over UDP are difficult to differentiate. For transaction-based applications, such as DNS, the request sent by the client is a response-based flow. However, when considering a

VoIP application over UDP, a duplex VoIP connection working over the same two UDP ports has two UDP streams carrying voice data. While both UDP streams are reverse streams of each other, they are not response-based flows, because neither application directly or indirectly sends packets in direct response to other packets. If one flow terminates, it would not immediately cause the other one to fail or block. Fortunately, these duplex VoIP flows can often be identified by inspecting other features, such as a unique packet size or its information inside packet headers.

```
Packet 64746
Internet Protocol, Src: 130.215.29.93 (130.215.29.93),
Dst: 130.215.29.29 (130.215.29.29)
Transmission Control Protocol, Src Port: http (80),
Dst Port: 1573 (1573), Seq: 58654752, Ack: 1706, Len: 1460
%Hypertext Transfer Protocol

Packet 64747 (ACK Packet)
Internet Protocol, Src: 130.215.29.29 (130.215.29.29),
Dst: 130.215.29.93 (130.215.29.93)
Transmission Control Protocol, Src Port: 1573 (1573),
Dst Port: http (80), Seq: 1706, Ack: 58656212, Len: 0
```

Figure 5.3: A Sample of IP and Transport Layer Packet Header.

Figure 5.3 illustrates the decoded IP and the transport layer packet headers for one TCP packet and one ACK packet from the corresponding ACK flow. Packet 64746 is from a TCP data flow, which is identified by the five tuple: (130.215.29.93, 80, 130.215.29.29, 1573, TCP). Packet 64747 is captured from the corresponding ACK flow and can be identified as (130.215.29.29, 1573, 130.215.29.93, 80, TCP). Based on the definition of reverse flow and response-based flows, they are both response-based flows. To identify the reverse flow and response-based flow is thus fairly easy on a residential wireless AP because there is symmetric routing and packets in both directions pass through the same AP.

Differentiating response-based flows is important for the selection of treatment methods. If one packet is intentionally dropped from a response-based flow, it triggers upper layer retransmissions. Thus, simply dropping a packet from a response-based flow may not alleviate congestion inside an AP effectively. On the contrary, dropping from response-based flows might aggravate congestion by introducing more retransmissions. For instance, dropping packets from a DNS flow will cause a timeout and retransmission in the DNS client. The DNS client resends the request while the end user experiences a long response time.

Algorithm 3 describes the response/non-response-based flow classifier. The current CATNAP implementation simply classifies the UDP flows as non-response based while TCP flows as response based flows because DNS is the only UDP-based response-based application found over residential network. However, DNS flows must be on a well known DNS port, and CATNAP can easily identify DNS flows by checking its source or destination port number. By inspecting the first packet's IP header, Algorithm 3 is able to determine the response/non-response-based nature of a flow. TCP and UDP are dominant transport layer protocols over current residential networks, but in the future some new transport layer protocols might appear over residential networks. Thus, our current response/non-response-based flow classifier simply classifies the flow with other transport layer protocol as response-based, and CATNAP would treat them as response-based flows. Algorithm 3 might be enhanced to distinguish the response-based UDP flows after having more thorough understanding of UDP based multimedia application behavior.

Algorithm 3 CATNAP Response/Non-Response-Based Flow Classifier

```
1:  for each arrival packet p do
2:      i ← get_flow_id(p)
3:      if isNewBorn(i) = true then
4:          if is_UDP_header(p) = true then
5:              if is_DNS(p) = true then                ▷ Classify DNS flow as a special case.
6:                  set_response_based(i, true)
7:              else
8:                  set_response_based(i, false)
9:              end if
10:         else
11:             set_response_based(i, true)
12:         end if
13:         set_newborn(i, false)
14:     end if
15: end for
```

5.3.2 Identify Interactive/Non-interactive Traffic

Karagiannis [20] and Roughan [22] identify applications based on average packet length. The CATNAP interactive/non-interactive flow classifier is similar to these classification methods based on the average packet length. However, CATNAP examines the ratio of full to non-full packets, where bulk-data transfers are likely dominated by full packets and interactive applications tend to send smaller, non-full packets.

The maximum packet length could be limited either by the network MTU or by the application

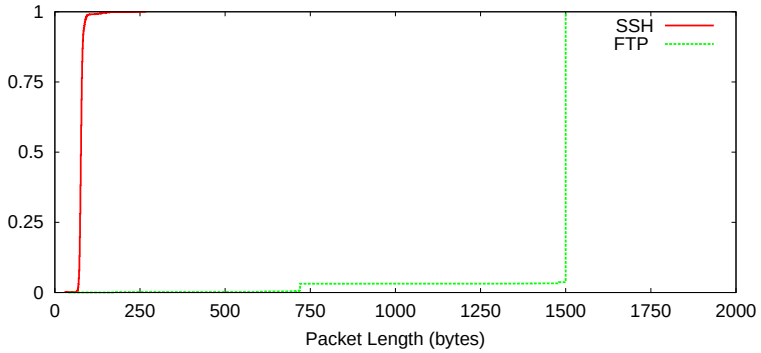


Figure 5.4: CDF of Packet Length for a VoIP and an FTP Application.

Table 5.4: Statistical Information of Packet Length for Several Application Instances.

Application	Maximum(bytes)	Avg (bytes)	Std Dev (bytes)	CoV
Web Browsing	1500	883.10	676.91	0.76
FTP data	1500	1459.56	190.19	0.13
Windows Media Streaming	1417	1413.43	32.64	0.02
RealMedia Audio	1500	1392.32	332.25	0.24
FPS Game	248	83.86	28.76	0.34
VoIP (Vonage)	200	200.00	0	0
VoIP (Yahoo Messenger)	100	75.58	4.15	0.06
SSH	1300	276.59	319.45	1.15

itself. Figure 5.4 compares the difference in packet length density between a VoIP session and an FTP file downloading session (data flow). As Figure 5.4 shows, the maximum packet length [4] observed in a VoIP flow generated by a LinkSys VoIP PAP2T adapter is only 200 bytes, much less than the typical MTU of 1500 bytes. The length of most data packets in the FTP data flow is 1500 bytes, which is the typical MTU setting over an Ethernet. Thus, the packet length and its various moments can be used as a feature to classify application type [20, 22].

To create the packet length signature for each flow, we choose statistics on packet length which can be updated in real-time with low computational complexity. The moving average, weighted moving average, and windowed average (used in [154]) are good candidates for packet length signature, and more complicated statistics such as quantile can also be calculated in real-time by approximation algorithms [155]. Table 5.4 lists the packet length signatures calculated with an average for several kinds of application instances gathered during the early stage of this study. The packet length in several applications is limited by the application or device involved. The VoIP

[4] Packet length in this preliminary study includes the IP header and its IP payload length.

voice stream generated by the Linksys VoIP adapter consists of packets with the same length (200 bytes in Table 5.4). These packets can be taken as "non-full" packets. In addition, interactive programs generate flows with high variance in packet length, for example, SSH and Web browsing. On the other hand, some flows such as an FTP data contain large packets with low variance. The maximum packet length in FTP flow is probably limited by the MTU settings. Thus, the FTP data packets are "full". Therefore, different applications can be identified by these statistical signatures based on their packet length.

The main purpose of introducing "full" or "non-full" packets is to identify time sensitive interactive applications. The flow which contains a large fraction of "non-full" packets is most likely to be generated by interactive applications, and needs to be treated in a timely fashion when congestion occurs at the AP. Based on the packet length classification results of our preliminary experiments, the CATNAP interactive/non-interactive classifier chooses the exponentially weighted moving average EWMA) of packet length as the interactive/non-interactive signature of a flow. For instance, when the packet p of flow i arrives, the CATNAP interactive/non-interactive classifier updates the packet length signature for flow i:

$$\overline{L} \leftarrow \overline{L} \times (1 - \alpha) + \alpha \times get\_pkt\_length(p) \tag{5.3}$$

where α is a constant used to calculate EWMA packet length and $\overline{L}$ is the EWMA length.

The main functionality of pure TCP control packets such as SYN, FIN, SYN-ACK, and ACK (not including piggyback ACK) are:

1. establish, manage and terminate a TCP connection.

2. provide flow control and avoid possible congestion.

3. provide reliable data transmission.

Since pure TCP control packets only provide TCP connection mechanism and do not carry any information from the application layer, calculating the EWMA packet length with pure TCP control packets can not help to infer the interactive/non-interactive nature of applications. Thus, as Line 3 - 5 of Algorithm 4 shows, the CATNAP interactive/non-interactive classifier excludes pure TCP control packets when calculating the EWMA packet length. However, the CATNAP interactive/non-interactive classifier calculates the EWMA packet length with every packet from

UDP flows because the UDP flows do not have flow control or connection control mechanism at the transport layer and generally there are no UDP packets without any payload.

Algorithm 4 describes the CATNAP interactive/non-interactive classifier, and Table 5.5 lists the values of constants and thresholds used in Algorithm 4.

Algorithm 4 CATNAP Interactive/non-interactive Flow Classifier

```
1:  for each arrival packet p do
2:      i ← get_flow_id(p)
3:      if isTCPCtrl(p) = true then
4:          return                                   ▷ Skip pure TCP control packets
5:      end if
6:      if get_ewma_pkt_length(i) = 0 then    ▷ Initial interactive/non-interactive classification.
7:          if get_packet_length(p) > T_init then
8:              set_interactive(i, false)
9:          else
10:             set_interactive(i, true)
11:         end if
12:         set_ewma_pkt_length(i, get_pkt_length(p))
13:         return
14:     end if

15:     if isInterative(i) = true then    ▷ Choose EWMA weight α based on current flow type.
16:         α ← α_int
17:     else
18:         α ← α_nint
19:     end if

20:     L̄ ← (1 − α) × L̄ + α × get_pkt_length(p)
21:     set_ewma_pkt_length(i, L̄)                    ▷ Update EWMA length for flow i

22:     if L̄_i > T_nint then
23:         set_interactive(i, false)                          ▷ Non-interactive flow.
24:     else if L̄_i < T_int then
25:         set_interactive(i, true)                               ▷ Interactive flow.
26:     else          ▷ Don't change the interactive/non-interactive attribute of flow i.

27:     end if
28:     return
29: end for
```

As Line 6 to 14 of Algorithm 4 shows, the CATNAP interactive/non-interactive classifier initially classify the flow based on its first non TCP-control-only packet. If the length of the first non TCP-control-only packet is larger than the initial interactive/non-interactive packet length threshold (T_{init}), flow i is classified as an non-interactive flow; otherwise it is classified as a interactive flow. Figure 5.5 depicts the initial classification process of the CATNAP interactive/non-interactive

Table 5.5: Constants in Algorithm 4

Symbol	Description	Value
T_{init}	Packet Length Threshold for initial interactive/non-interactive classification	750
T_{int}	Packet Length Threshold for interactive flows	500 bytes
T_{nint}	Packet Length Threshold for non-interactive flows	1000 bytes
α_{int}	EWMA weight for interactive flows	0.6
α_{nint}	EWMA weight for non-interactive flows	0.05

classifier, when the packet length (l_1) of the first non-pure control packet from a given flow is greater than the initial interactive/non-interactive packet length threshold (T_{init}), CATNAP categorizes the flow as an non-interactive flow. On the contrary, CATNAP classifies a flow as an interactive flow when the packet length of its first non-pure control packet is smaller than T_{init}. In this manner, CATNAP is able to have initial interactive/non-interactive classification result on the first non-pure control packet from each flow.

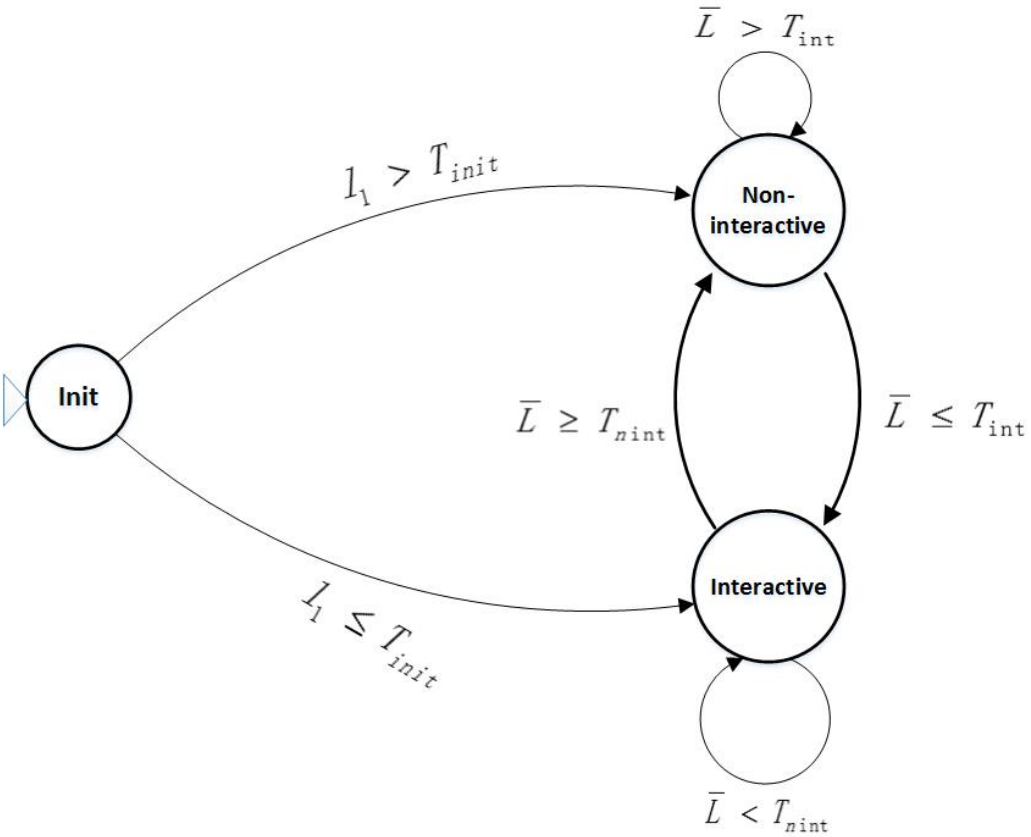


Figure 5.5: State Diagram for Interactive/non-Interactive

Unlike the response/non-response based classification, a flow can switch from interactive state to non-interactive state when the EWMA packet length ($\overline{L}$) is greater than a given threshold, and vice versa. However, our preliminary study shows that the classification results oscillate when there is only a single fixed classification threshold. Therefore, the CATNAP interactive/non-interactive classifier chooses two set thresholds and EWMA weights for interactive and non-interactive flow respectively. As Figure 5.5 shows, flow i leaves the interactive state and enters non-interactive state only when its EWMA length $\overline{L}$ is greater than the non-interactive threshold (T_{nint}). Similarly, CATNAP re-classifies flow i from non-interactive to interactive only when its EWMA length $\overline{L}$ is smaller than the interactive threshold (T_{int}).

In addition to stability, the interactive/non-interactive classifier also needs to detect when the interactive/non-interactive flow nature changes. CATNAP uses two different weights to calculate the EWMA packet length in order to meet the requirements of stability and low reaction time of the interactive/non-interactive classifier. For non-interactive flows, we choose a small EWMA weight (α_{nint}) to assimilate the effects from occasional small packets in non-interactive flows. We use a large EWMA weight (α_{int}) to calculate the EWMA packet length for interactive flows in order to detect the flow nature promptly.

Parameter Selection

As Figure 5.5 shows, Algorithm 4 utilize two sets of thresholds and EWMA weights for interactive/non-interactive flows respectively to provide:

1. stable classification results, which implies the EWMA weight should be *small.*
2. short reaction time when the link quality changes, which implies the EWMA weight should be *large.*

This study utilizes a special SSH flow with flow state changes to help choose the two sets of thresholds and EWMA weights for Algorithm 4. As Figure 5.6 shows, a SSH flow switches from interactive to non-interactive at the 40th second when the user starts to *cat* a large log file; and the same SSH flow returns to interactive at the 60th second after the *cat* command finishes. Each red + dot in Figure 5.6 indicates the IP packet length of each packet from the SSH downstream. The SSH flow consists of "non-full" packets as an interactive flow except between the 40th and 60th seconds when it acts as an non-interactive flow. Note, Figure 5.6 also shows the EWMA packet

length calculated by Algorithm 4 with parameters listed in Table 5.5, which are parameters used on our simulation study.

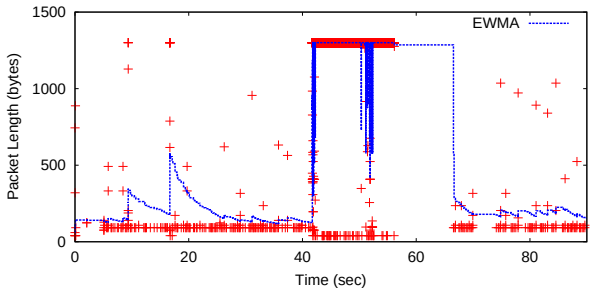
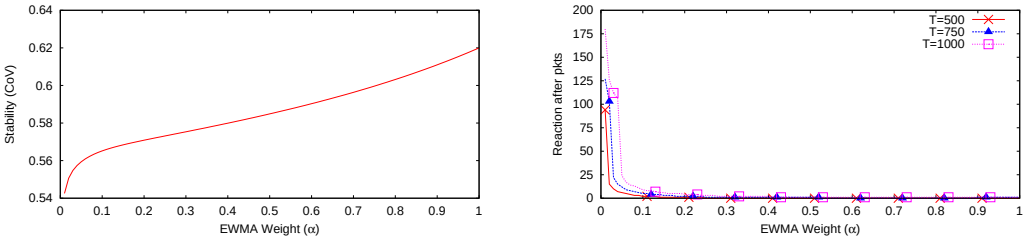


Figure 5.6: An SSH Flow Switching between Interactive and Non-interactive

To facilitate the selection of parameters and reduce the number of variables, this study first determines the EWMA weight (α) for Algorithm 4. Intuitively, Algorithm 4 with a larger EWMA weight (α) gives more stable results. However, there is no commonly accepted metric to evaluate the stability of EWMA packet length. This study chooses the coefficient of variance (CoV) of the EWMA packet length as the indicator of stability for Algorithm 4. Figure 5.7(a) shows the CoV of EWMA packet length calculated by Algorithm 4 with EWMA weight (α) between 0.01 and 1 for the whole SSH trace. Figure 5.7(a) shows that the CoV of EWMA packet length of the whole SSH trace increases as the EWMA weight (α) increases. But there exists a knee when α is around 0.05.



(a) Stability (b) Reaction

Figure 5.7: Interactive/non-interactive Classifier Parameter Selection

The reaction time is another metric to evaluate the performance of Algorithm 4. However, because Algorithm 4 is a packet based approach not a time-based approach, a temporal metric may not effectively evaluate its reaction speed. Thus, we choose *the number of packets received by CATNAP* after the flow changes its interactive/non-interactive nature until Algorithm 4 detects the change. Algorithm 4 chooses the initial interactive/non-interactive packet length threshold (T_{init}) as 750 bytes which is half of the default MTU setting as 1500 bytes. To simplify the problem, we

choose a single fix threshold,where $T_{int} = T_{nint}$ when evaluating the reaction speed of Algorithm 4. The three curves in Figure 5.7(b) show the reaction speed under three different thresholds (T): 500 bytes, 750 bytes, and 1000 bytes. The reaction speed increases as the EWMA weight (α) increases. When α is greater than 0.6, CATNAP only needs a couple more packets before it is able to detect the flow nature changes. As Figure 5.7(b) shows, all three curves are in same shape, each of them has a knee point when EWMA weight (α) is in the range of 0.01 and 0.1. Moreover, Figure 5.7(b) tells that the selection of the interactive/non-interactive threshold (T_{int} and T_{nint}) is not sensitive in the range between 500 bytes and 1000 bytes.

CATNAP *pushes* interactive flows by dequeing the packet from interactive flows first before any packet of non-interactive flows. If CATNAP chooses a large EWMA weight (α), a couple of small packets from an non-interactive flow may mislead CATNAP to classify it as an interactive flow mistakenly. Therefore, mistakenly classifying an non-interactive flow as interactive flow potentially harms other flows. On the contrary, mistakenly classifying an interactive flow as non-interactive would not harm other flows. Thus, Algorithm 4 chooses a two threshold and two EWMA weight scheme to balance the stability and reaction time. For interactive flows, Algorithm 4 calculates packet length with a large EWMA weight to have quick reaction when they switch to non-interactive. For non-interactive flows, CATNAP calculates packet length with a small EWMA weight to avoid mistakenly classifying them as interactive by only several small packets among a stream of large packets. Thus, in combination of Figure 5.7(a) and Figure 5.7(b), CATNAP choose $\alpha_{nint} = 0.05$ for non-interactive flows, and $\alpha_{int} = 0.6$ for interactive flows. Meanwhile, CATNAP chooses $T_{nint} =$ 1000 bytes as "high water" marker for flows switching from interactive to non-interactive, and $T_{int} = 500$ bytes as the "low water" marker for flows switching from non-interactive to interactive. In this way, CATNAP is able to assimilate occasional outlier packets to have a stable classification result.

This section describes how CATNAP classifies interactive/non-interactive flows by using packet length information. In the future, the interactive/non-interactive classifier can be improved by combining with other classification techniques. For example, TCP based interactive applications usually set the PUSH flag at the TCP headers. However, the implementation of this study only differentiates interactive and non-interactive flows simply by the packet length.

5.3.3 Identify Greedy/Non-Greedy Traffic

There exist two kinds of applications from the perspective of their bandwidth usage behavior, *greedy* applications and *non-greedy* applications. Greedy applications such as file downloading (FTP) and P2P file sharing, always try to use as much as bandwidth as the transport layer allows, while non-greedy applications only consume a certain amount of bandwidth even when more bandwidth is available. The bandwidth usage of these non-greedy applications is usually constrained by application implementation (e.g., codec used) or the content to deliver. For example, DivX MPEG-4/CIF format videos with high dynamic content usually are encoded at 80 - 2000 Kbps [156], which is much less than the IEEE 802.11 link capacity. Thus, CATNAP tries to differentiate applications into *greedy* and *non-greedy* categories based on the application behavior of bandwidth usage. Over residential wireless network, typical non-greedy applications are multimedia applications, such as video streaming, VoIP, and game applications.

Unlike interactive/non-interactive flows, greedy/non-greedy flows is relative to the current link capacity. There is no constant threshold to decide the greedy/non-greedy nature of flows. CATNAP classifies a flow as *greedy* when its bandwidth usage is greater than the fair share rate. In reality, greedy flows tend to be non-interactive flows. Therefore, this study introduces the non-interactive fair share rate (R_{nint}) to help identify the greedy flows from non-greedy flows. Meanwhile, CATNAP uses the fair share rate (R_{greedy}) as the upper limitation of bandwidth of greedy flows. Section 5.4.4 presents how the CATNAP treatment module limits the greedy flow bandwidth usage under the fair share rate (R_{greedy}) for greedy flows.

The CATNAP greedy/non-greedy classifier uses the fair share rate (R_{nint}) of non-interactive flows as a threshold to differentiate greedy/non-greedy flows:

$$R_{nint} \leftarrow \frac{C - \sum r_{int}}{N_{nint}} \leftarrow \frac{C - \sum r_i}{count\_noninteractive\_flows()}, \text{where } isInteractive(i) = \textbf{true} \qquad (5.4)$$

where C is estimated downlink capacity calculated by Algorithm 1, N_{nint} is the number of non-interactive flows, and $\sum r_{int}$ is the summation of bandwidth usage of all interactive flows. Algorithm 5 presents the implementation details of Eq 5.4.

When calculating the fair share rate R_{nint}, Algorithm 5 excludes the bandwidth consumed by interactive flows when reserving the bandwidth for non-interactive flows. Therefore, malicious applications might take advantage of the CATNAP implementation by sending numerous small

Algorithm 5 Calculate Fair Share Rate for Non-interactive Flows

At the end of each epoch period,
$t \leftarrow now()$
$C \leftarrow get\_link\_capacity()$
$R_{int} \leftarrow N_{nint} \leftarrow 0$ ▷ Reset counters.
for each flow i **do**
  if $isActive(i) =$ **true then** ▷ Skip inactive flows.
    if $isInteractive(i) =$ **true then**
      $R_{int} \leftarrow R_{int} + get\_flow\_rate(i)$
    else ▷ non interactive flows
      $N_{nint} \leftarrow N_{nint} + 1$
    end if
  end if
end for
$R_{nint} \leftarrow (C - R_{int})/N_{nint}$
$set\_fairshare\_noninteractive(R_{nint})$

packets to exhaust all available bandwidth because the CATNAP interactive/non-interactive classifier mistakenly categorizes them as *interactive* flows. However, we could not find any application which fit into this category in reality. In the future, if there are any greedy and interactive flows, we may use other techniques to filter out these malicious applications and apply special treatment on them[5].

Algorithm 6 CATNAP Greedy/Non-greedy Flow Classifier

At the end of each epoch period,
for each flow i **do**
  if $isActive(i) =$ **true then** ▷ Skip inactive flows.
    $r_i \leftarrow get\_flow\_rate(i)$
    $T_{greedy} \leftarrow \frac{R_{nint}}{2} \leftarrow \frac{get\_fairshare\_noninteractive()}{2}$
    if $r_i < T_{greedy}$ **then**
      $set\_greedy(i,$ **false**$)$
    else
      $set\_greedy(i,$ **true**$)$
    end if
  end if
end for

Algorithm 6 shows the CATNAP greedy/non-greedy classifier. With the assumption that the bandwidth usage of a flow would not double during the next epoch period, we classify the flows whose current bandwidth usage is lower than half of the fair share rate as non-greedy flows, and the rest as greedy flows. Note, in order to protect TCP flows during slow start, new start/restart TCP flows are classified as non-greedy until their bandwidth usage is greater than the greedy/non-

[5]In theory, transmitting small packets would significantly lower the overall throughput of a wireless AP.

greedy threshold, which is half of the fair share rate. Meanwhile, CATNAP tests all active flows' bandwidth usage with the greedy or non-greedy threshold, including both interactive and non-interactive flows. In this way, CATNAP is able to detect and treat any interactive flows acting as greedy flows consuming too much bandwidth.

5.3.4 Identify Active/Inactive Traffic

In addition to above three treatment-based classifiers, this study also implements an active/inactive flow classifier to differentiate inactive flows from active flows. An inactive flow is a flow which has not transmitted any packets in a given time period. By identifying the inactive flows, CATNAP is able to temporarily loan the link capacity assigned to inactive flows to other active flows and improve the overall link capacity utilization. In addition to detecting inactive flows, CATNAP also needs to detect terminated flows. However, currently there exists no widely accepted method to determine when is the end of a flow [93]. NetFlow [93], a widely used commercial flow analyzer, allows users to set the *inactive flow threshold* between 1 to 60 seconds. If no packet arrives during the inactive flow threshold, Netflow marks the corresponding flow as inactive, and if no packet arrives after more than 60 seconds, NetFlow considers the corresponding flow as terminated. CATNAP is able to determine a TCP flow is terminated when detecting a FIN/RST packets, while it is difficult to detect the end of a UDP based flow without inspecting the payload of every packet.

Similar to NetFlow, CATNAP also maintains the per-flow information listed in Table 5.2 to classify active and inactive flows. The CATNAP treatment module uses the active/inactive classification result such as number of inactive flows to calculate fair share rate for greedy flows. Thus, a large flow inactive threshold such as 60 seconds would reduce the link utilization because CATNAP would mark the corresponding flow as active even if it has not received any packets in the past 60 seconds. Therefore, we introduce an active/inactive classifier as Algorithm 7 to promptly distinguish the active, inactive and terminated flows.

Instead of using a fixed threshold to classify the active/inactive flows, Algorithm 7 calculates active/inactive threshold T_{act} for each flow i. The active/inactive threshold T_{act} for flow i is calculated as $c \times RTT_i$, where c is a positive constant, and RTT_i is the round trip time for flow i. If no packet from the given flow arrives in the last T_{act} time period, the flow is marked as inactive by setting the *IsActive* flag as *false*. In this study, we assume that an active sender always has some content to send in one RTT, otherwise the flow does not act as an active sender. Therefore,

Algorithm 7 CATNAP active/in-active flow classifier

At the end of each epoch period,
1: $t \leftarrow now()$
2: **for** each flow i **do**
3: $\Delta t \leftarrow t - get\_last\_active\_tm(i)$
4: $T_{act} \leftarrow c \times get\_rtt(i)$
5: **if** $(\Delta t > T_{end})$ **then**
6: $set\_end(i, \textbf{true})$
7: **else if** $(\Delta t > T_{act})$ **then**
8: $set\_active(i, \textbf{false})$
9: **else**
10: $set\_active(i, \textbf{true})$
11: **end if**
12: **end for**

CATNAP is able to loan the bandwidth reserved for inactive flows and give it to active flows. In this study, active/inactive threshold T_{act} for flow i is chosen as one and half of flow i's RTT (c is set to 1.5).

Table 5.6: Constants/Variables Used in Algorithm 7

Symbol	Description	Value
T_{end}	Threshold to determine terminated flows.	60 secs
c	Constant to calculate active/inactive Threshold	0.5
T_{act}	Threshold to determine active/inactive flows	-

5.3.5 Summary

Table 5.7: Treatment Classification for Typical Applications.

Classification			Representative Application
Response-based	Interactive	Greedy	
false	true	false	Game w/UDP VoIP w/UDP, Streaming w/UDP
false	false	true	NFS w/UDP
false	false	true/false	High Quality Video Streaming w/UDP
true	true	false	Instant Messenger, SSH/Telnet, Game w/TCP, Web, DNS VoIP w/ TCP, Streaming w/TCP
true	false	true	Email, FTP, P2P
true	false	true/false	Streaming High Quality Video w/TCP

Table 5.7 lists classification categories in CATNAP with typical representative applications running over residential networks. The CATNAP classifier divides the applications into eight different

categories, where applications in the same category can be treated with the same treatment methods because they have the similar QoS requirement.

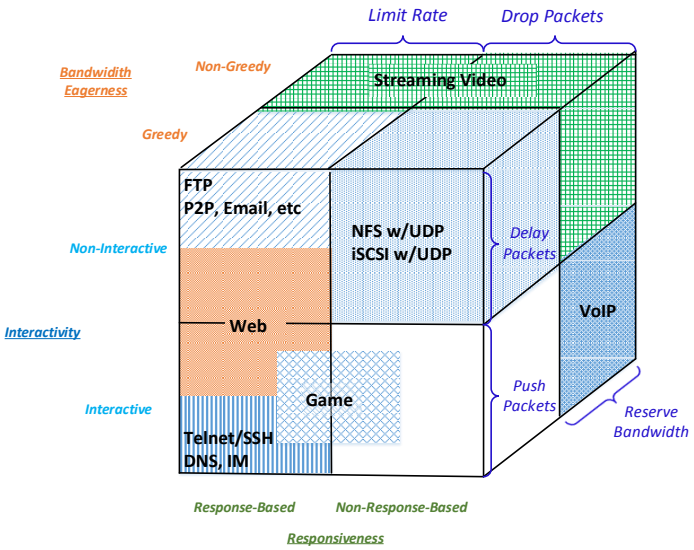


Figure 5.8: Treatment-Based Traffic Classification.

Figure 5.8 visualizes the mappings between applications and potential treatments. The three classifiers form axes in a classification space for both applications and treatments. Focusing on representative applications, which are overlaid as areas with different file patterns on the face of the cube, most instances of an application are expected to consistently be classified in one category. For example, interactive applications, such as Telnet or DNS, exhibit response-based, non-full packet traffic that is sent as it is available. Instances of other applications, such as Web or Games, might span multiple categories in the classification cube.

Rather than this 'multiple classification' being a problem, our CATNAP classifier indicates that not all instances of an application should be treated in the same manner. For instance, downloading game maps and playing games are two different use cases with different QoS requirements, and they should be handled differently when congestion occurs. The key point in showing the application classification in Figure 5.8 is to visualize the expected range of instances of an application instead

of associating flows with one application.

Response-based or non-response based flows can be identified by matching its reverse traffic except for several special cases such as duplex VoIP applications over UDP. Interactive or non-interactive can be decided after inspecting the packet length of several packets in a short time period. Before distinguished whether a flow is interactive or non-interactive, the flow is marked as non-interactive flow by default. Similarly, for non-greedy or greedy, it may take several epoch periods before classifying a flow as greedy. When a new flow is detected, CATNAP marks it as a default flow which is response-based, non-interactive, greedy until the CATNAP classifier can gather enough information to decide its type.

5.4 Treatments on Classified Flows

Section 5.1 briefly discusses the possible treatments to be deployed onto a resource limited wireless AP. This section describes the implementation details of CATNAP treatment modules: Section 5.4.1 describes the CATNAP per-flow queue controller; Section 5.4.2 presents the treatment policy maker which decides how to treat each active flow; Section 5.4.3 describes how to implement push and delay treatment within the IP layer dequeue function; and Section 5.4.4 shows how to handle flows which consume too much bandwidth.

5.4.1 Queue Implementation

CATNAP implements the four treatment methods: *push*, *delay*, *drop*, and *limit advertised window size* by modifying the default queue discipline inside a wireless AP. Several types of queues can be used to implement the four treatment policies.

1. **An individual queue for each flow:**

   The AP maintains one queue for each flow and uses a common scheduler to decide the dequeue procedure and dropping or rate control policy.

2. **An individual queue for each category:**

   Instead of maintaining a separate queue for each flow, the AP maintains a separate queue for each of the eight categories.

3. **An individual queue based on treatment operations:**

   Two of the four treatments: *push* and *delay*, can simply have a separate queue. Packets enqueued in the push queue have higher priority than packets in the delay queue. However, the operations *limit advertised windows* and *drop* require additional implementation overhead. For instance, flows in the same drop queue may have different drop probabilities.

4. **An individual queue for each client:**

   The AP maintains a separate queue for each wireless client.

5. **A single queue with complex queue discipline:**

   The AP still uses one single queue to buffer the packets from all flows, but it provides an enhanced enqueue and dequeue policy to provide the treatment operations.

A separate queue for each traffic category is a method widely used in traffic shaping. For example, IEEE 802.11e uses a similar approach. However, in the CATNAP classification, a flow may change its category as its traffic nature changes, thus, requiring a move of a batch of packets from one queue to another. Another disadvantage of using a separate queue for each traffic category is that this method requires a complicated scheduler when multiple queues are moved from one category to another category at the same time because the scheduler needs to maintain the inter-flow packet order as well as intra-flow packet order. In this case, CATNAP needs a complicated/efficient dequeue and enque method to preserve the ordering of packets of each flow. Similar to a separate queue for each traffic category, the individual queue for each treatment operation and individual queue for each wireless client face similar problems.

Using a single queue for several flows avoids moving packets between queues. However, it is still complicated from the point of view of implementation, requiring a set of enqueue and dequeue policies to place packets in front of other packets and to prevent starvation. Moreover, although a single queue can avoid the movement between different queues when the classification result of a flow changes, it eventually has to be able to switch the position of a batch of enqueued packets inside the single queue.

Based on implementation considerations, a separate queue for each flow is the best candidate for implementing our treatment policies. The obvious advantage of a separate queue for each flow is to avoid copying packets in or across queues and easily maintain the packet ordering inside each flow when applying push and delay operations. With this method, each flow can be easily associated

with its classification result and corresponding treatment. When the classification result changes as the flow nature changes, a separate queue can adapt without moving the enqueued packets from one class-based queue as required by other methods.

However, the per flow queue method will need a complicated scheduler to dequeue the packets and a complicated queue monitor[6] to detect congestion because the number of concurrent flows could be large and varying over time. In our preliminary implementation, a congestion detection module based on [14] controls the CATNAP treatment module, and the treatment module only activates when congestion occurs. However, the congestion detection module triggers unnecessary oscillation of the overall throughput of the CATNAP AP especially when FTP flows with long RTT are present. In order to avoid the oscillation introduced by the congestion monitor, we decide to enable the treatment module all the time instead of only enabling when congestion occurs because our treatment methods are also not computational intensive, introducing little CPU overhead with light traffic load.

5.4.2 Treatment Policy Maker

When greedy or non-interactive flows exceed the fair share rate allocated to them, CATNAP needs to intentionally drop some packets from non-response-based flows or reduce the advertised window size for response-based flows in order to avoid possible congestion inside the AP. Therefore, this study implements a treatment policy maker to decide how to apply treatment on flows to control their bandwidth usage. Figure 5.9 shows how CATNAP decides the dropping probability (d) for non-response-based flows, and the advertised window size ($AWND$) for response-based flows. Note, because the DNS flows are response-based UDP and there is no $AWND$ field in UDP packet headers, CATNAP simply bypasses the step of setting $AWND$ for DNS flows. However, DNS flows are usually classified as interactive flows and CATNAP still *pushes* the DNS flows. Moreover, at the time of this research, no interactive flows behaved in a greedy manner. Therefore, CATNAP treats interactive flows only based on their response-based/non-response-based nature and do not differentiate greedy/non-greedy among interactive flows. If any future applications act as interactive and greedy at same time, CATNAP is able to apply *drop* and *limit advertised window size* treatment policies similar to greedy and non-interactive flows.

[6] A virtual queue can be introduced to monitor the total buffer usage for congestion detection purposes.

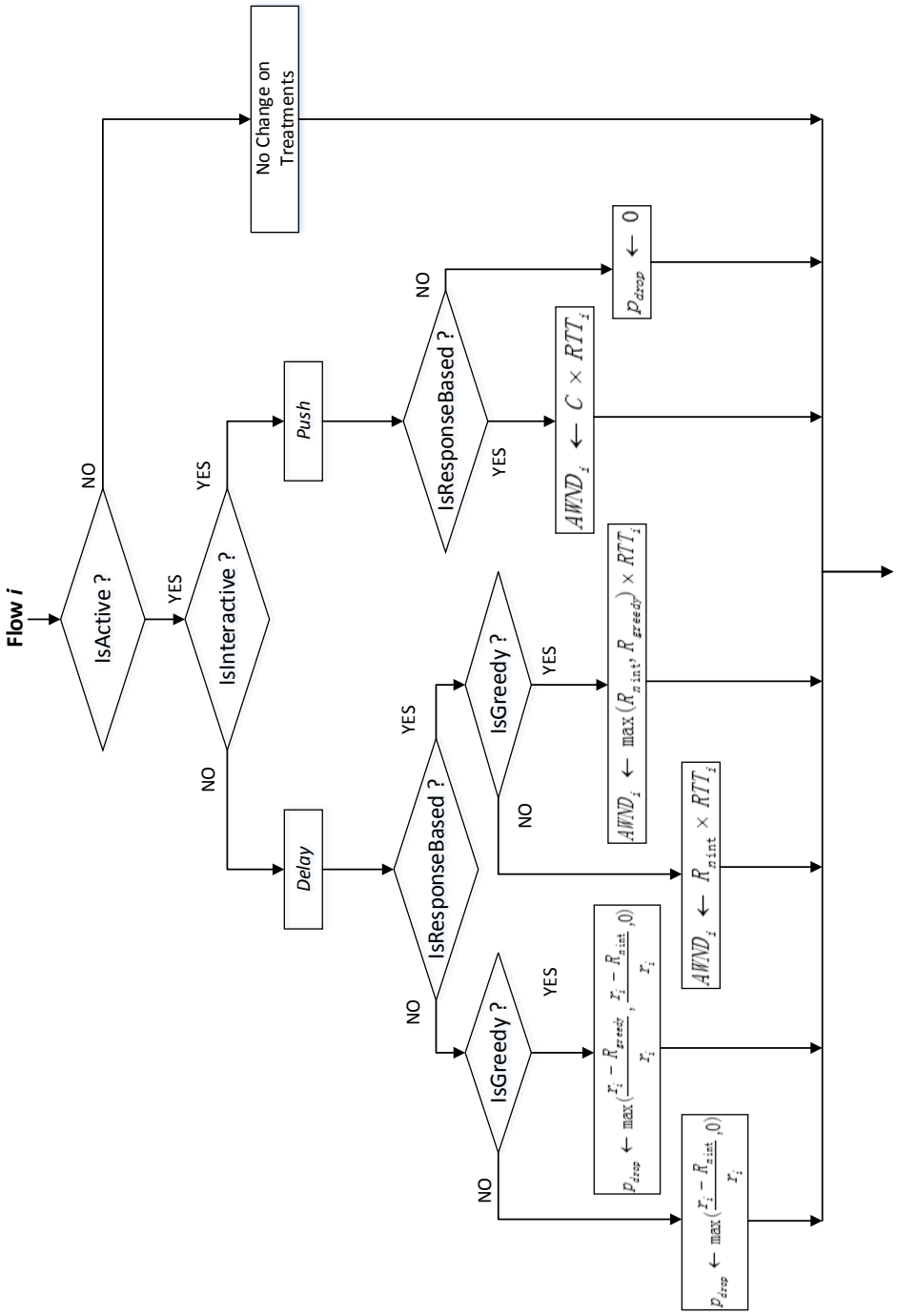


Figure 5.9: Flow Chart of Treatment Policy Maker

The treatment policy maker in Figure 5.9 decides how CATNAP treats flows. However, CATNAP implements actual treatments by enhancing the wireless AP queue controllers: the push or delay treatment is implemented inside the down-link enqueue function, the drop treatment is implemented inside the down-link dequeue function, and the limited advertised windows function is implemented inside the uplink enqueue function.

Algorithm 8 shows the implementation details of our CATNAP treatment policy maker where d_i is drop probability for a non-response-based flow i, and $AWND_i$ is the advertised window size for response-based flow i. Note, in most cases, the fair share rate of non-interactive flows R_{nint} is greater than the fair share rate of greedy flows R_{greedy}. But when interactive and greedy flows exist, the R_{greedy} might be larger than R_{nint}. Although no flows acting as interactive and greedy are observed in real network traces, CATNAP uses the larger of R_{greedy} and R_{nint} when calculating the advertised window size or drop probability for non-greedy flows. In this way, CATNAP protects the non-greedy flows or any newly started TCP flows. Additionally, the CATNAP treatment policy maker would not make any new decisions on the inactive flows which do not transmit any packets in a given time period. If any packet from inactive flows arrives in the next epoch period, the CATNAP treatment module applies the same treatment policy as during the last known active epoch period.

Epoch Selection

As mentioned earlier, the interactive/non-interactive classifier and response/non-response classifier are packet based, and CATNAP checks every packet header and updates the classification result. However, the active/inactive classifier, the greedy/non-greedy classifier, the treatment policy maker and the downlink capacity estimator are epoch based: they are triggered by a soft timer interrupt every epoch.

This study introduces the epoch period because the four modules require global flow information such as the number of active flows in the current system and the traffic volume of every active flows passing through the CATNAP AP. However, a true real-time method would be a CPU intensive per packet based approach, and its time complexity is $O(N^2)$ where N is number of packets passing through the AP. The packet-based approach would introduce unnecessary computational overhead especially when the CATNAP AP is under the pressure of high volume FTP flows. Therefore, CATNAP chooses the epoch-based method: for every epoch, the CATNAP module updates per

Algorithm 8 Determine Treatments for Downlink Flows

At the end of each epoch period,
$C \leftarrow get\_link\_capacity()$
for each flow i **do**
  if $isActive(i) =$ **false then**
    continue ▷ Skip inactive flows
  end if
  $RTT_i \leftarrow get\_rtt(i)$
  if $isInteractive(i) =$ **true then**
    if $isResponse(i) =$ **true then**
      $AWND_i \leftarrow C \times RTT_i$
      $set\_AWND(i, AWND_i)$
    else
      $d_i \leftarrow 0$
      $set\_drop\_probability(i, d_i)$
    end if
  else ▷ $isInteractive(i) =$ **false**
    if $isResponse(i) =$ **true then**
      if $isGreedy(i) =$ **true then**
        $AWND_i \leftarrow R_{greedy} \times RTT_i$
      else
        $AWND_i \leftarrow max(R_{greedy}, R_{nint}) \times RTT_i$
      end if
      $set\_AWND(i, AWND_i)$
    else ▷ $isResponse(i) =$ **false**
      $r_i \leftarrow get\_flow\_rate(i)$
      if $isGreedy(i) =$ **true then**
        $d_i \leftarrow max(\frac{r_i - R_{greedy}}{r_i}, \frac{r_i - R_{nint}}{r_i}, 0)$
      else
        $d_i \leftarrow max(\frac{r_i - R_{nint}}{r_i}, 0)$
      end if
      $set\_drop\_probability(i, d_i)$
    end if
  end if
end for

flow statistics, recalculates the fair share rate for greedy flows, and detects possible inactive flows. Determining the value of epoch time is another challenge of this study. A small epoch value may place overwhelming computation load without significantly improving performance and even worse a small epoch value may cause the CATNAP treatment module to overtreat flows. On the contrary, a large epoch time could not accurately capture the fluctuation of traffic, and CATNAP may not treat bursty greedy flows, for example FTP flows, with long RTTs.

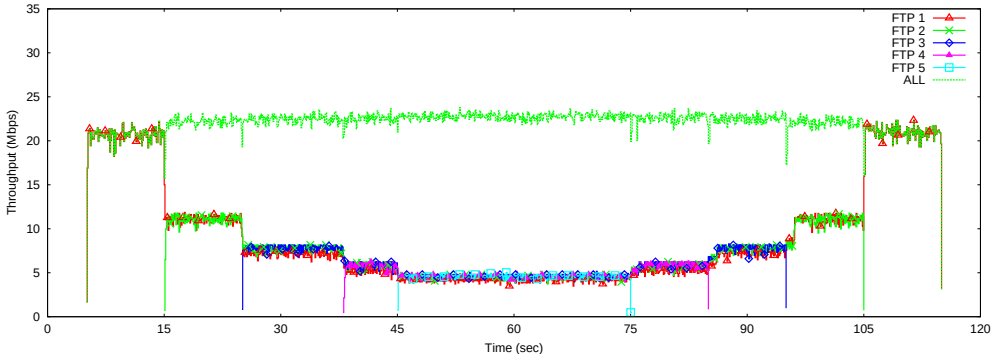


Figure 5.10: Simulation to Selection Epoch

Figure 5.10 depicts a special simulation test bed to help determine an appropriate epoch value. To fully saturate the IEEE 802.11g link, this study injects five concurrent FTP flows into the simulation testbed. Each of them has homogeneous settings: RTT is set as 20 ms and the maximum of CWND is set as 50 packets by NS-2.33 default. In order to demonstrate that the CATNAP module treats each FTP flows fairly, a new FTP flow is launched every 10 second interval since the fifth second. From the 75th second when the fifth FTP flow has been running for 30 seconds, the flows leave every 10 seconds in the reverse order of their arrival sequence. The mean cumulative throughput for the whole system is calculated between the 50th second and the 70th second to avoid the initial state of this experiment. Moreover, to reduce the number of variables in the simulation setup, we replace the CATNAP downlink capacity estimator with a static estimator which returns a preset static link capacity (C). In our previous measurement studies, we learned that the maximum effective capacity of an IEEE 802.11g link is between 20-26 Mbps. Thus, we repeat our simulation with gradually increased link capacity from 20 to 30 Mbps in order to find out a priori the available capacity for our simulation set up when the cumulative throughout of five FTP flows reaches its peak.

Meanwhile, the epoch value varies between 0.001 seconds to 0.2 seconds to evaluate the perfor-

mance of Algorithm 8 under different epoch values. This simulation setup uses FTP flows with the same settings with RTT of 0.02 seconds. Thus, we choose the epoch value as 0.001 second as 1/20 of RTT, 0.002 second as 1/10 of RTT, 0.01 second as a half of RTT, 0.02 second as 1 RTT, 0.1 second as 5 times of RTT, and 0.2 seconds as 10 times of RTT. The simulation repeats under various combination of epoch values and link capacities (C), and calculates the average overall throughput between the 50th and the 70th seconds shown in Table 5.8.

Moreover, to study the AP queue dropping behavior, the same traffic load is applied onto a DropTail AP with the queue size of 100 packets (the queue size CATNAP used in this simulation). Table 5.9 lists the total number of packet dropped over the whole simulation time.

Table 5.8: Average Downlink Throughput (Mbps) with Various Epochs and Link Capacities

	CATNAP Epoch (sec)					
Capacity	0.001	0.002	0.01	0.02	0.1	0.2
C = 20 Mbps	18.40 ± 0.32	18.51 ± 0.41	18.92 ± 0.67	18.93 ± 0.65	18.08 ± 0.64	21.43 ± 2.38
C = 22 Mbps	20.88 ± 0.26	20.42 ± 0.21	20.89 ± 0.21	20.90 ± 0.22	20.85 ± 0.27	20.86 ± 0.24
C = 23 Mbps	21.65 ± 0.41	21.50 ± 1.22	22.10 ± 0.24	22.09 ± 0.27	22.07 ± 0.25	21.26 ± 0.29
C = 24 Mbps	22.66 ± 0.44	22.70 ± 0.38	22.72 ± 0.45	22.23 ± 0.47	22.60 ± 0.37	22.74 ± 0.52
C = 25 Mbps	22.71 ± 0.49	22.71 ± 0.40	22.73 ± 0.38	22.80 ± 0.38	22.80 ± 0.42	22.42 ± 1.98
C = 26 Mbps	22.50 ± 0.45	22.53 ± 1.76	22.48 ± 0.45	22.39 ± 0.77	22.24 ± 0.46	22.13 ± 0.83
C = 28 Mbps	21.47 ± 0.51	22.12 ± 0.80	21.96 ± 1.65	22.02 ± 1.19	21.88 ± 1.68	22.19 ± 0.99
C = 30 Mbps	21.40 ± 0.47	21.62 ± 0.39	21.90 ± 1.40	22.10 ± 0.76	22.10 ± 0.76	21.82 ± 1.63
DropTail						
21.08 ± 3.76						

Table 5.9: Queue Drop Events in with Various Epochs and Link Capacities

	CATNAP Epoch (sec)						DropTail
Capacity	0.001	0.002	0.01	0.02	0.1	0.2	
C = 20 Mbps	0	0	0	0	0	163	**4802**
C = 22 Mbps	0	0	0	0	0	176	
C = 23 Mbps	0	6	4	4	0	167	
C = 24 Mbps	4	4	0	27	54	134	
C = 25 Mbps	13	8	1	6	12	163	
C = 26 Mbps	18	11	110	135	96	313	
C = 28 Mbps	51	42	330	301	507	101	
C = 30 Mbps	64	59	573	310	309	528	

Table 5.8 shows that an inflexion point exists around the 24 Mbps with all possible epoch values. The overall throughput does not increase even with a link capacity greater than 25 Mbps. In some cases when the dummy link capacity estimator over estimated the link capacity by giving a link capacity more than 25 Mbps, the overall throughput is even lower because the five FTP flows over saturate the link. Table 5.9 also shows the number of packet drop events increase when the estimator mistakenly over-estimates the link capacity. Based on the information shown in Table 5.8,

the achievable throughput over IEEE 802.11g is between 21 to 22 Mbps, which matches the results from our previous measurement study [17]. Meanwhile, Table 5.8 helps us choose the initial value of effective link capacity (C_0) used in Algorithm 1 in Section 5.2.2.

Table 5.8 and Table 5.9 show that the simulations with smaller epoch times of 0.001 and 0.002 seconds achieve similar performance as an the epoch time of 0.01 seconds. CATNAP with *smaller* epoch time (0.002 and 0.002 seconds) has better performance than CATNAP with larger epoch time when the effective link capacity estimator over-estimates the link capacity. A large epoch time of 0.2 seconds, as large as 10 times the RTT, does accurately capture the fluctuation of TCP flows, and the AP queue still drops packets even when underestimated available capacity (C=20 Mbps). How to more accurately estimate the RTT at middle point is beyond the scope of this study. Because we assume that all wireless nodes are immobile, the RTT between source nodes and destination nodes does not change much. CATNAP with 0.01 or 0.02 epoch time introduces less computational overhead and has similar performance as CATNAP with small epoch values when the effective link capacity estimator does not over-estimate the link capacity.

In addition to the CATNAP treatment modules, our link capacity estimator described in Algorithm 1 is also an epoch based approach. Theoretically, Algorithm 8 is able to choose a different epoch value than Algorithm 1. However, both Algorithm 8 and Algorithm 1 result in a similar epoch time period: 10 ms. Thus, this study selects the epoch time as 0.01 seconds.

5.4.3 Push or Delay

The *push* and *delay* operations can be easily implemented with a priority queue. Because the CATNAP classifier categorizes flows into two categories: *interactive* and *non-interactive*, CATNAP pushes interactive flows while delaying non-interactive flows. Therefore, we implement a dequeue function with a two-band priority, which always dequeues the packet from interactive flows before dequeuing any packet from non-interactive flows. In this way, CATNAP always pushes interactive flows in front of non-interactive flows. When CATNAP detects more than one flow with the same priority, it acts as a FIFO queue controller, and dequeues the packet with the lowest enqueue time stamp.

Algorithm 9 describes implementation of the CATNAP dequeue function to support push and delay treatments. In Algorithm 9, the function $get\_front\_pkt\_tm(i)$ returns the enqueue timestamp of the current head packet in the head of the queue of flow i.

Algorithm 9 Dequeue Function to Support Push and Delay

```
    At the end of each epoch period,
 1: flag_nint ← true                          ▷ Default to send non-interactive flows.
 2: t_min ← ∞
 3: next ← 0                                  ▷ the id of next flow to dequeue.
 4: for each flow i do
 5:     if isEmpty(i) = false then
 6:         t_front ← get_front_pkt_tm(i)
 7:         if isInteractive(i) = true then
 8:             if flag_nint = true then     ▷ flow i becomes interactive.
 9:                 flag_nint ← false
10:                 t_min ← t_front
11:                 next ← i
12:             else if t_min > t_front then ▷ sending packets from interactive flows as FIFO
13:                 t_min ← t_front
14:                 next ← i
15:             end if
16:         else                             ▷ flow i is non-interactive flow.
17:             if flag_nint = true and t_min > t_front then
18:                 t_min ← t_front
19:                 next ← i
20:             end if
21:         end if
22:     end if
23: end for
24: if next > 0 then
25:     p ← dequeue(next)
26: else
27:     p ← null
28: end if
29: return p
```

The preliminary CATNAP implementation chooses a weighted round robin (WRR) dequeue function to avoid possible starvation of non-interactive flows. However, during the simulation, the CATNAP queue controller never uses WRR dequeue function even with 10 pairs of current VoIP flows as interactive flows and 2 FTP flows as non interactive flows, because the interactive flows usually consumes much smaller bandwidth with small packets than non-interactive flows such as FTP flows. Thus, CATNAP simply choose the two-band priority queuing scheme to implement the push and delay operations.

5.4.4 Drop or Limit Advertised Window Size

In addition to *push* and *delay* operations, the CATNAP treatment module also implements *drop* and *limit advertised window size* operations to control the bandwidth usage of non-interactive flows,

especially non-interactive and greedy flows, to protect the interactive flows. The drop operation for non-response-based flows is implemented in the enqueue function of the CATNAP downlink controller, while the limit advertised window size is implemented in the uplink enqueue controller.

However, either drop operation or limit advertised window size operation requires CATNAP to decide how much bandwidth needs to be reduced for a given flow when its bandwidth usage exceeds the fair share rate for non-interactive flows (R_{nint}) calculated in Algorithm 5, or the fair share rate for greedy flows (R_{greedy}) calculated in Algorithm 10.

Fair Share Rate for Greedy Flows

Although the fair share rate of non-interactive flows (R_{nint}) provided by Algorithm 5 works well with the CATNAP greedy/non-greedy classifier, it is conservative because the link capacity reserved for the non-greedy and non-interactive flows is wasted and not used by any other flows. Thus, in order to improve the overall link capacity utilization, we introduce Algorithm 10 to calculate the fair share rate of greedy flows (R_{greedy}). As Equation 5.5 shows, the fair share rate of greedy flows (R_{greedy}) is calculated as

$$R_{greedy} \leftarrow \frac{C - \sum r_{interactive} - \sum r_{non-greedy}}{N_{greedy}} \tag{5.5}$$

where C is effective downlink capacity provided by Algorithm 1, $\sum r_{interactive}$ is the bandwidth usage of all interactive flows, $\sum r_{non-greedy}$ is the bandwidth usage of all non-greedy and non-interactive flows, and N_{greedy} is the count of greedy flows. Algorithm 10 describes the implementation detail of Equation 5.5.

Downlink Enqueue Controller

The downlink enqueue function implements the drop treatment of non-response-based flows. Algorithm 11 describes the dropping treatment. The drop treatment only applies to non-response-based and non-interactive flows because no interactive greedy flow has been observed in real residential networks. Note, the current CATNAP implementation still *helps* DNS flows by pushing DNS packets because DNS flows are classified as interactive flows as they usually consists of packets under 500 bytes.

Algorithm 10 Calculate Fair Share Rate for Greedy Flows

At the end of each epoch period,
$C \leftarrow get\_link\_capacity()$
$N_{greedy} \leftarrow 0$
$R_{int} \leftarrow 0$
$R_{ngreedy} \leftarrow 0$
for each flow i **do**
  if $isActive(F_i) =$ **false then**
    continue
  else if $isInteractive(i) =$ **true then**
    $R_{int} = R_{int} + get\_rate(i)$
  else if $isGreedy(i) =$ **true then**
    $N_{greedy} \leftarrow N_{greedy} + 1$
  else ▷ $isGreedy(i) =$ **false**
    $R_{ngreedy} \leftarrow R_{ngreedy} + get\_rate(i)$
  end if
end for
if $N_{greedy} > 0$ **then**
  $R_{greedy} \leftarrow \frac{C - R_{int} - R_{ngreedy}}{N_{greedy}}$
else
  $R_{greedy} \leftarrow C$
end if
$set\_fairshare\_greedy(R_{greedy})$
return

Algorithm 11 Enque Downlink Packet (drop operation)

for every downlink packet p **do**
  $i \leftarrow get\_flow\_id(p)$
  if $isResponse(i) =$ **false and isInteractive(i) = false then**
    ▷ Treat non-interactive, non-response (UDP) flow
    $d_i \leftarrow get\_drop\_probability(i)$
    if $uniform(0, 1) \leq d_i$ **then**
      drop(p)
      return
    end if
  end if

  if $enque(p) = false$ **then** ▷ Downlink queue is full.
    drop(p)
  end if
  return
end for

Uplink Queue Controller

The limit advertised window size treatment for response-based TCP flows is implemented by enhancing the uplink enqueue function. For each arriving ACK packet, the enqueue function retrieves its advertised window size ($AWND_{orig}$) from its TCP header. Because the flow is unidirectional,

the advertised window size is used to control the sending rate of the reverse flow of the ACK flow as Line 3-4 shows. The advertised window size of the downlink stream flow i is calculated in Algorithm 8. As Line 6-9 in Algorithm 12 shows, CATNAP updates the TCP header with the smaller of $AWND_{orig}$ and $AWND_i$ in packet p, and the corresponding TCP and IP layer CRC. In this way, CATNAP explicitly informs the TCP sender to reduce its transmission rate by reducing the advertised window size, and the TCP sender reduces its sending rate if the advertised window size is smaller than the sender's window size.

Algorithm 12 Enque Uplink Packet

```
1: for every uplink packet p do
2:     if isTCPACK(p) = true then
3:         i ← get_reverse_flow_id(p)
4:         AWND_orig ← get_awnd_size(p)
5:                                  ▷ AWND_i advertised window size is calculated in Algorithm 8
6:         if AWND_i < AWND_orig then
7:             set_awnd_size(p, AWND_i)
8:             update_crc(p)                          ▷ Update IP and Transport Layer CRC
9:         end if
10:    end if
11:    return
12: end for
```

5.5 Summary

This chapter describes the design and implementation of the CATNAP classification and treatment methods. Table 5.10 summaries the key variables and their initial values used in CATNAP.

Table 5.10: Constants and Thresholds Used in CATNAP

Layer	Parameter	Value	Notes
Classifier	T_{init}	750 Bytes	Initial threshold for inter/non-interactive flows
	T_{nint}	1000 Bytes	Threshold for non-interactive flows
	T_{int}	500 Bytes	Threshold for interactive flows
	α_{nint}	0.60	EWMA packet length weight α for non-interactive flows
	α_{int}	0.05	EWMA packet length weight α for interactive flows
	T_{end}	60 sec.	Threshold used to detect end of flows
	c	1.5	Flow i would be considered as inactive if no packet received after $c_{\times} RTT_i$
Treatment	*epoch*	10 ms	
	$AWND_{init}$	1000 packets	Init advertised window size for unclassified/interactive TCP flows
Supporting func.	ω	0.1	EWMA weight used to estimate effective downlink capacity
	C_0	25 Mbps	Initial valued for effective downlink capacity
	RTT_{udp}	200 ms [139]	Threshold used to detect inactive UDP flows

Additionally, Figure 5.11 summaries all implemented modules and the dependencies between them. The CATNAP modules can be grouped into three functional categories: supporting functions, classifiers, and treatment modules. The supporting modules are not part of the CATNAP architecture itself, they can be replaced with the function integrated within current AP or the third party algorithms. However, when this research was conducted, no AP was able to provide these functions. The treatment methods are integrated with the uplink and downlink queue controllers.

Meanwhile, the CATNAP modules can be categorized into *packet-based* or *epoch based* from the implementation perspective. The packet-based modules updates their information based every arriving packets, and the epoch-based modules updates their information every epoch time. The modules implemented as epoch based could be implemented as packet based modules, but it may introduce more computational complexity and unnecessary overhead.

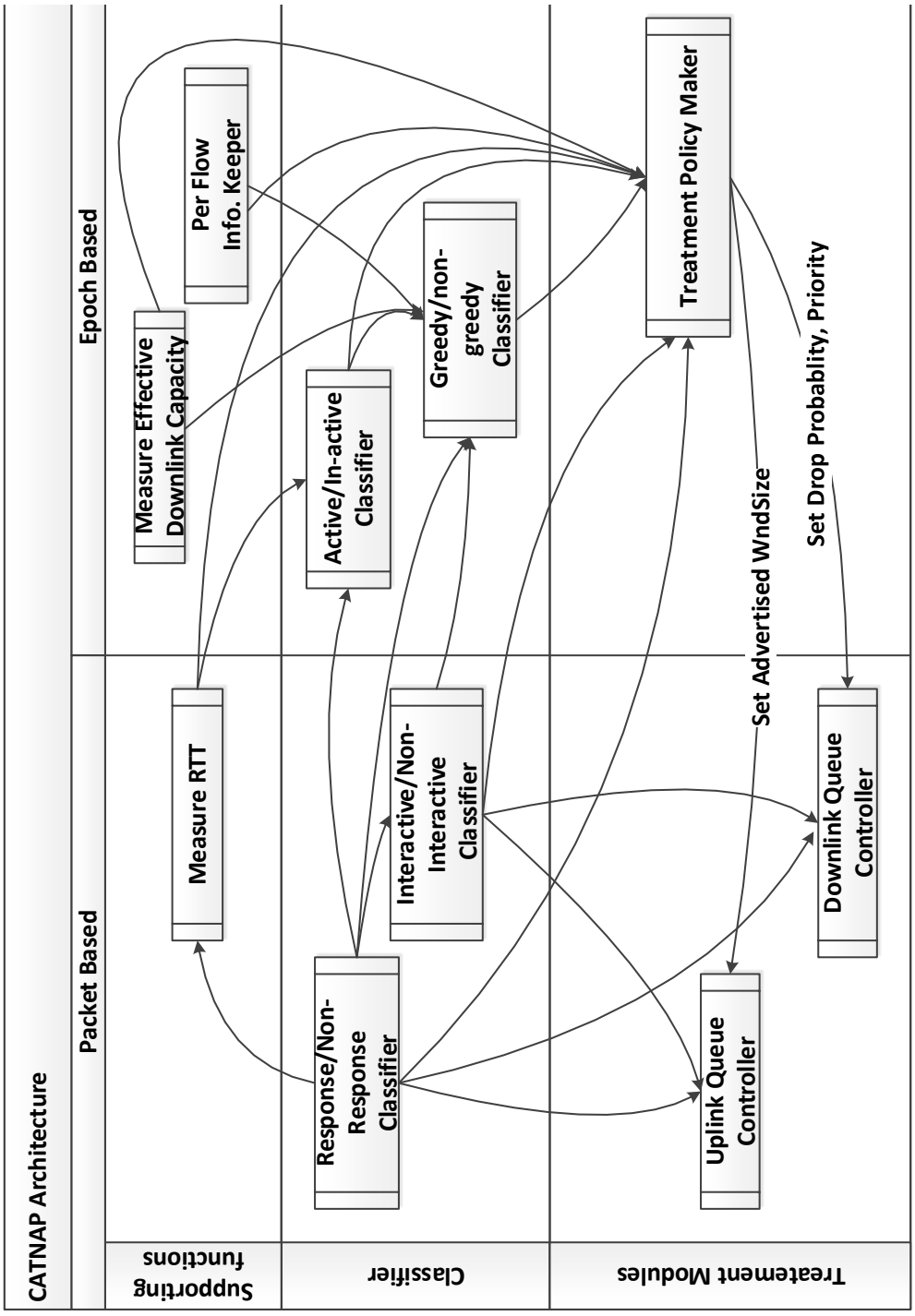


Figure 5.11: Dependencies between CATNAP Modules

Chapter 6

Simulation

This chapter describes simulation results for CATNAP implemented with NS-2.33 [157, 158]. This chapter is organized as following: Section 6.1 presents the classification simulation; Section 6.2 describes the NS-2 simulation setup; Section 6.3 compares simulation results between CATNAP, DropTail and Strict Priority Queue (SPQ); Section 6.4 further compares the performance between CATNAP and DropTail under various configurations, such as different latency, different queue capacity and multiple foreground applications.

6.1 Classification

As described in Section 5.3, the CATNAP classifier consists of four parts: response-based/non-response-based classifier, interactive/non-interactive classifier, greedy/non-greedy classifier, and active/inactive classifier.

The response-based/non-response-based classifier is straightforward because CATNAP only needs to examine one packet from a given flow to determine its response or non-response nature. The greedy/non-greedy classifier requires runtime information such as the downlink capacity and the statistical information of concurrent flows. Without this runtime information, it is almost meaningless to run the greedy/non-greedy classifier to decide the greedy/non-greedy nature of a given flow. Thus, the greedy/non-greedy classifier is not suitable to validate with offline traces. Moreover, the main purpose to introduce the active/inactive classifier is to improve the downlink capacity utilization by allowing inactive flows to surrender their bandwidth share to active flows.

The active/inactive classification result would change the treatment methods of flows. Therefore, this section focuses on the evaluation of the CATNAP interactive/non-interactive classifier implemented with Perl and NS-2.33 against offline traces.

6.1.1 Dataset Used

Real network traces are essential to verify the accuracy of the CATNAP interactive/non-interactive classifier. These real network traces have to be pre-classified or provide corresponding application layer information, which helps to determine the interactive/non-interactive nature for a given flow. Though numerous IP level packet traces are available over the Internet, most of them only keep the IP header information and erase the application layer information due to privacy concerns. Moreover, most public available traces are gathered from core routers that may contain enterprise application traffic in addition to residential traffic. These enterprise applications may have different flow characteristics and QoS requirements than residential applications.

Even the same kind of applications might behave differently under different scenarios, and their QoS requirements might be different also. For example, a SSH user can start an interactive SSH session by editing a file on a remote host, but another user can copy a huge binary file with a SSH session. In this case, the ordinary port-based classifiers would classify these two different SSH sessions as the same kind of application because they are using the same port, but the CATNAP interactive/non-interactive classifier classifies them into two different categories since their interactive/non-interactive nature are different. Therefore, verifying the CATNAP interactive/non-interactive classifier requires traces with pre-classified traffic.

In this study, the traces used to verify the CATNAP interactive/non-interactive classifier need to meet the following requirements:

1. cover all CATNAP traffic categories;
2. have results from a known classification method;
3. provide application level information to understand their QoS requirements.

A group of Italian researchers published a set of packet level traces [159] along with their traffic statistic and analysis tool: *Tstat*. Their traces contain Skype (VoIP) flows with different audio codec, MSN (instant message) traces, and FastWeb IP-TV (video) traces. Kinicki and Claypool gather packet level traces for *Second Life* [160], a virtual world with 3D graphics players interacting

with each other through avatars. The Second Life traces use TCP to handle the login server and other utilities, while the communication between the game client and server is based on UDP.

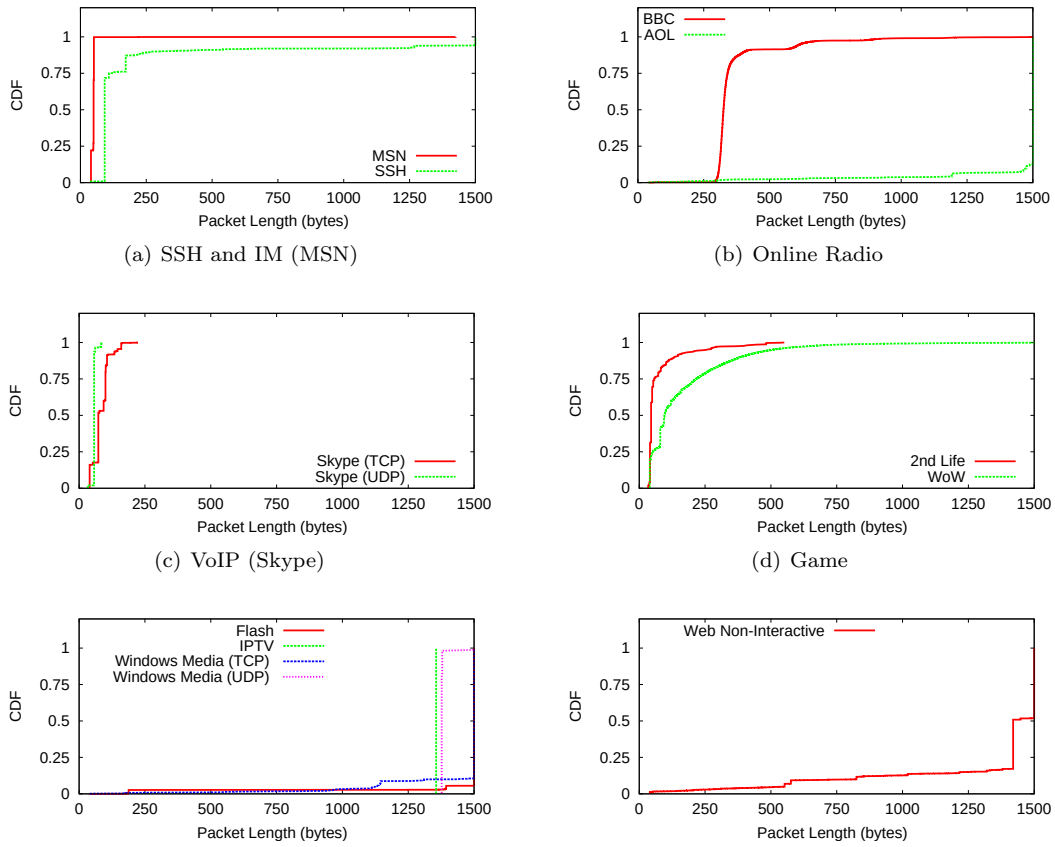
In addition to these traces provided by other researchers, this study setup a testbed to gather more traces to cover the rest of CATNAP traffic categories. The testbed included a Dell P4 server running with Windows 2003 media center placed inside the WPI campus network, a Dell P4 desktop running with Windows XP/SP2 stationed in a residence near WPI. The client and server were connected through the WPI campus network and a cable network. Windows Media Video traces were generated from the Windows 2003 server to the XP client. The Dell P4 desktop collected traffic of World of Warcraft, Online Radio, Flash Video and SSH.

Web traces used in this study are gathered by a group of Italian network researchers [161] in 2004. The trace files are collected from a 200 Mbps link connecting the University of Napoli "Federico II" network to the rest of the world. The Web traces only contain TCP port 80 traffic between clients inside the University of Napoli and the outside world. Because of privacy concerns, the trace files only contains TCP and IP headers with anonymous IP addresses.

Table 6.1 summarizes traces used to validate the CATNAP interactive/non-interactive classifier. Figure 6.1 presents the CDF of packet length of flows identified from the traces. To have a clear presentation, Figure 6.1 groups flows into six categories according to their application types, e.g., the online radio flows include both BBC and AOL online radio broadcasting.

Table 6.1: Trace Used to Validate CATNAP Interactive/Non-interactive Classifier

APPs	Type	Source	Approx. Gathered	Notes
IPTV	Video	[159]	2008	UDP-based HD video.
Skype	VoIP	[159]	2008	Skype audio trace w/ various audio codec.
MSN	IM	[159]	2008	MSN messenger, online chatting.
SecondLife	Game	[160]	2007	UDP-based MMO game
flash video	TCP	WPI	2009	TCP-based flash video clips provided by Adobe.
SSH	TCP	WPI	2009	A graduate student daily SSH login traces (reading email with Pine, and list/read/write plain text files.
Windows Media Streaming	TCP & UDP	WPI	2009	Multi-layer video clips encoded with Windows Media 2003 server, the highest video encoding rate is 1192 Kbps.
War Of Warcraft	TCP	WPI	2009	A level 80+ Korean player played 10 sessions with a Korean server.
BBC audio	TCP	WPI	2009	bbc online live news radio broadcasting with BBC real player IE plugin.
NPR, AOL	TCP	WPI	2009	AOL and NPR online live news radio broadcasting with player downloaded from the NPR website.
Web	HTTP	[161]	2004	Traffic on Port 80 generated inside network of University of Napoli, Italy.



(a) SSH and IM (MSN)

(b) Online Radio

(c) VoIP (Skype)

(d) Game

(e) Video

(f) Web

Figure 6.1: CDF of Packet Length of Different Applications

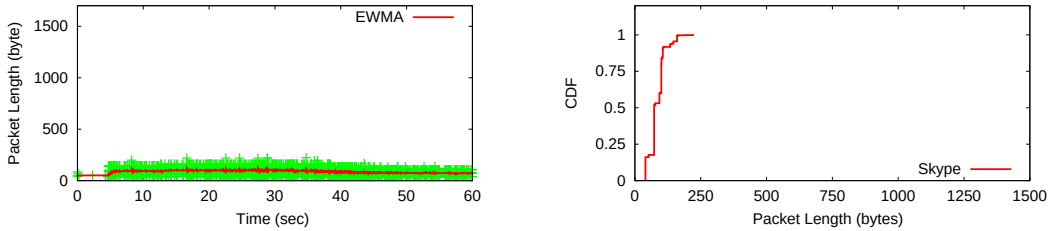
6.1.2 Offline Interactive/Non-interactive Classifier Result

Figure 6.2 to Figure 6.7 depict the sample results of our Perl-based CATNAP interactive/non-interactive classifier. To provide a clear explanation, we randomly select one flow from each of the six categories and group figures together:

1. figure (a) not only depicts the EWMA packet length calculated by Algorithm 4 but also shows the packet length for each packet as a "plus" sign.

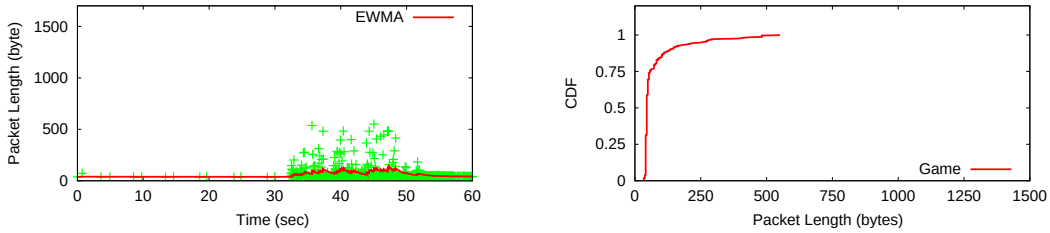
2. figure (b) shows the CDF of packet length from the selected flow.

Figure 6.2 presents the sample result for VoIP applications, and Figure 6.3 shows the sample result for game applications. The CATNAP interactive/non-interactive classifier classifies both VoIP and game flows as interactive, because their EWMA packet length is much smaller than the interactive to non-interactive threshold (T_{nint}) of 1000 bytes.



(a) EWMA (b) CDF of Packet Length

Figure 6.2: VoIP (Skype w/TCP) CATNAP Interactive/Non-interactive Classification Result

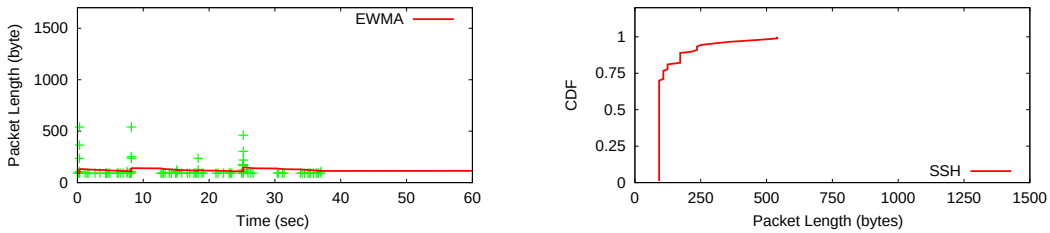


(a) EWMA (b) CDF of Packet Length

Figure 6.3: Game (2nd life) CATNAP Interactive/Non-interactive Classification Result

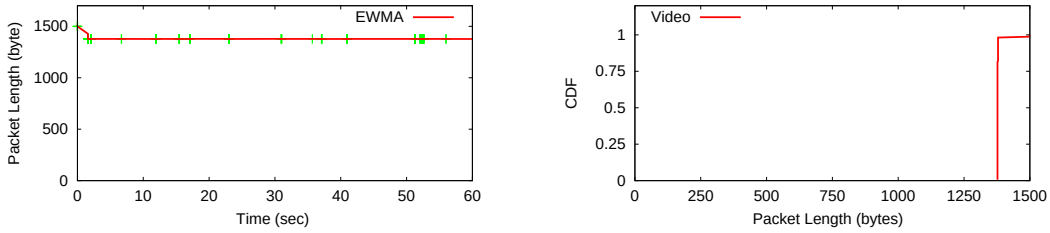
Figure 6.4 shows a sample SSH stream. Different from VoIP or game flows, SSH flows might be interactive or non-interactive or both. The user behavior might effect the interactive/non-interactive nature of SSH flows. CATNAP classifies the sample SSH flow as interactive because the majority of its packets are small.

Figure 6.5 shows one sample of UDP-based IPTV flows gathered in [159]. The IPTV flow gathered in [159] carries HD video, and the size of video packets are greater than the non-interactive packet threshold (T_{nint}). Thus, CATNAP classifies the IPTV flows as non-interactive.



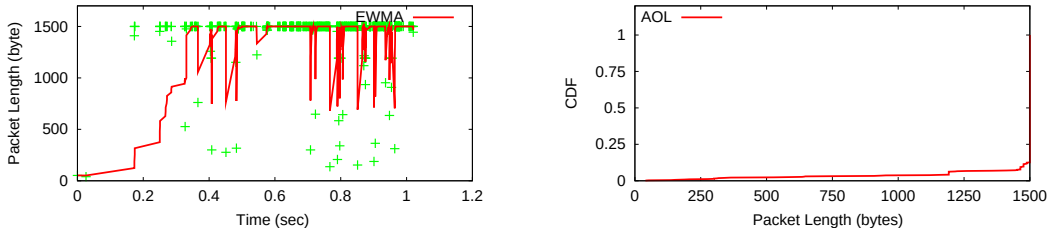
(a) EWMA (b) CDF of Packet Length

Figure 6.4: SSH (interactive) CATNAP Interactive/Non-interactive Classification Result



(a) EWMA (b) CDF of Packet Length

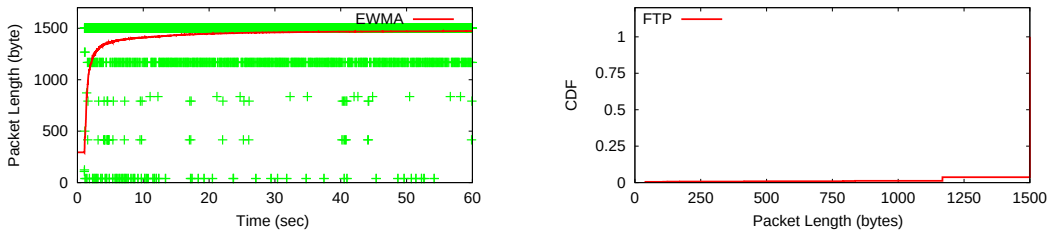
Figure 6.5: Video Streaming (Windows Media) with UDP CATNAP Interactive/Non-interactive Classification Result



(a) EWMA (b) CDF of Packet Length

Figure 6.6: Online Radio (AOL) CATNAP Interactive/Non-interactive Classification Result

Figure 6.6 shows one sample of AOL online radio flows. Some online radio applications still choose traditional streaming protocols such as RTSP, and their flows consist of small packets similar to VoIP flows. But some the online radio applications such as the one shown in Figure 6.6 choose to use buffering technology: they download and buffer audio clips as temporary files, and play media back later from local file caches. Thus, the online radio flow shown in Figure 6.5 only lasts for one second, and some of its audio packets are larger than 1250 bytes. Thus, CATNAP classifies it as non-interactive.



(a) EWMA (b) CDF of Packet Length

Figure 6.7: File Downloading (FTP) CATNAP Interactive/Non-interactive Classification Result

Figure 6.7 shows a sample of FTP/SFTP flows. As expected, the majority of the FTP/SFTP packets are large, and CATNAP classifies it as non-interactive.

6.1.3 NS-2 Implementation of CATNAP Classifier

This section compares the CATNAP interactive/non-interactive classification results between the NS-2 and Perl implementation. We implement all four CATNAP classifiers in both NS-2 and Perl, but the greedy/non-greedy classifier is not suitable to be verified with offline traces because its classifier requires bandwidth estimation results. The NS-2 implementation of the response-based/non-response-based and active/in-active classifiers is the same as the Perl offline implementation, and we did not see any difference between their results.

The main difference between NS-2 and Perl implementation is that the Perl based CATNAP classifier directly takes offline trace files in Wireshark/PCap format as input, which is the same as the file format used by the wireless sniffer in our home wireless measurement study. However, due to the limitation of NS-2 simulator, the Wireshark/PCap trace file needs to be converted to NS-2 trace format before it can be fed into the NS-2 CATNAP classifier. Due to the implementation limitation of NS-2 TCP layer, the NS-2 TCP agent combines small packets into several big packets where the maximum packet length is limited by the MTU settings if the NS-2 TCP layer receives a large amount of small packets from the application layer in a short time period. Thus, the actual packet size transmitted does not exactly match the trace file fed into the NS-2 simulator. Thus, this Section needs to confirm that the NS-2 implementation would not skew the interactive and non-interactive classification results.

To compare Perl and NS-2 classification results, this study chooses a special SSH trace collected from a controlled wireless testbed. The reason to choose SSH flows is that SSH flows typically act as interactive flows but they can act as non-interactive flows when transferring a large file through a SSH connection. Therefore, a special SSH flow is generated:

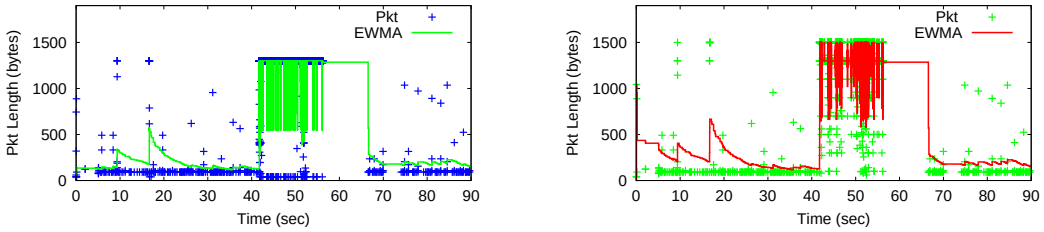
1. after the user logged into the SSH server, he traversed through several directories with shell commands;

2. around the 40th second after he logged in, the user opened a 20MB plain text file, and "cat"ed its contents to the screen;

3. around the 55th second, the whole file content has been displayed; the user resumed the

directory operations until disconnected at 65th second.

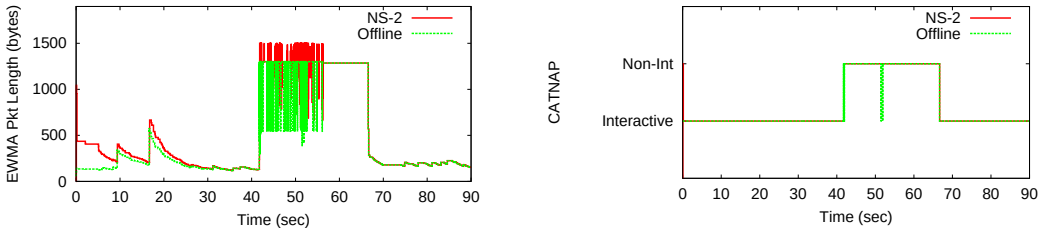
Clearly, the SSH flow contains two flow nature transitions: at the 40th second, it changed from interactive to non-interactive while it changed from non-interactive to interactive at the 55th second. The offline and NS-2 classifier should be able to detect the two flow state transitions.



(a) Offline (b) NS-2

Figure 6.8: EMWA Calculated by Perl (offline) and NS-2 Implementation

Figure 6.8 compares the EWMA packet lengths calculated by the Perl based CATNAP offline classifier and the CATNAP NS-2 implementation respectively. The green "plus" symbols represent packet length for each packet in the SSH flow, and the red line presents the EMWA length calculated in the CATNAP interactive/non-interactive classifier (Algorithm 4).



(a) EWMA (b) CATNAP

Figure 6.9: Interative/Non-Interactive Classification Results

Figure 6.9(a) compares the EWMA packet length calculated by the NS-2 implementation and Perl based implementation which is marked as "offline". Figure 6.9(b) shows that both Perl and NS-2 implementation yield the same interactive/non-interactive classification result.

6.1.4 Offline Interactive/Non-Interactive Classification Results

The CATNAP interactive/non-interactive classifier is a packet-based approach. It infers the interactiveness of a flow by calculating the EWMA packet length when each packet arrives. A flow may

change from interactive to non-interactive if its packet length nature changes, e.g. the SSH flow shown in the previous section. Thus, we use two tables to summarize the CATNAP interactive/non-interactive classification results for traces listed in Table 6.1:

1. Table 6.2 summarizes the flows which do not switch their interactiveness state during their life cycle,

2. Table 6.3 summarizes the flows which involve interactive state changes.

Table 6.2: Interactive and Non-Interactive Classification Result (Flows without State Changes)

Apps	# of flows	Interactive		Non-Interactive		SubTotal	
		Count	%	Count	%	Count	%
msn	100	90	90.0	0	0	90	90
skype(audio)	24	24	100.0	0	0	24	100.0
ssh	138	137	99.3	0	0	137	99.3
second-life	115	101	87.8	0	0	101	87.8
WoW	10	10	100.0	0	0	10	100.0
iptv	125	1	0.8	122	97.6	123	98.4
flash-video	10	0	0	0	0	0	0.0
wmtcp(video)	10	0	0.0	0	0.0	0	0.0
wmudp(video)	10	0	0.0	10	100.0	10	100.0
BBC(audio)	22	10	45.5	0	0.0	10	45.5
AOL,NPR(audio)	12	1	8.3	0	0.0	1	8.3
Web	317	41	12.9	31	9.8	72	22.6

Table 6.3: Interactive and Non-Interactive Classification Result (Flows with State Changes)

Apps	# of flows	Interactive		Non-Interactive		SubTotal		As One Flow	
		Count	%	Count	%	Count	%	%	%
msn	100	9	9.0	1	1.0	10	10.0	83.6	16.4
skype(audio)	0	0	0.0	0	0	0	0	-	-
ssh	138	1	0.7	0	0	1	0.7	97.7	2.3
second-life	115	14	12.2	0	0	14	12.2	96.3	3.7
WoW	10	0	0.0	0	0	0	0	-	-
iptv	125	0	0.0	2	1.6	2	1.6	26.9	73.1
flash-video	0	0	0	0	0	0	0	-	-
wmtcp(video)	10	0	0.0	10	100.0	10	100.0	1.6	98.4
wmudp(video)	10	0	0.0	0	0.0	0	0.0	-	-
BBC(audio)	22	2	9.1	10	45.4	0	54.5	45.1	54.9
AOL,NPR(audio)	12	2	16.7	9	75.0	11	91.7	-	-
Web	317	63	19.9	182	57.4	245	77.2	11.4	88.6

As Table 6.2 shows, 90 of the 100 MSN flows are consistently classified as interactive without any flow state changes. The remaining 10 MSN flows involve CATNAP classification result changes: some packets in the flow are classified as interactive while some packets are classified as non-

interactive. However, if we treat the 10 MSN flow as one flow, 83.6% packets of the 10 flows are classified as interactive, as the rightmost column shown in Table 6.3.

VoIP applications such as Skype are all classified as interactive because the bandwidth usage of Skype is limited by the audio codec used. World of Warcraft flows are all classified as interactive, while the 87.8% of Second Life flows are classified as interactive flows. The 14 of 115 Second Life flows which involves flow state changes are more likely be interactive also. 96.3% of packets are classified as interactive if we consider the 14 flows with state changes as one flow.

The behaviors of online radio player such as BBC-audio, AOL-audio and NPR-audio are different from VoIP audio application such as Skype. After manually inspecting the online radio traces, we notice that AOL and NPR audio players are flash based and the player buffers audio clips through HTTP protocol just as a regular FTP downloading flow. Thus, although the audio traffic normal consists of small packets, flash based players behave like a non-interactive FTP downloading application.

Moreover, Web flows listed in Table 6.2 and Table 6.3 diversify in interactive and non-interactive categories. Web traffic is the most popular and the most interesting traffic among residential network. Different types of applications can be hidden under the umbrella of Web. Browser based game, flash video player as browser plug-in, and regular Web content are all delivered on the well-known Web port 80, 443 and 8080 in order to go through widely deployed firewalls. Thus, as expected, Web traffic are classified either interactive or non-interactive. However, if we consider all Web flows as one flow, 88.6% of packets are classified as non-interactive.

6.2 Traffic Generation in NS-2

This section describes the traffic generation in the NS-2 simulator: how to generate game, VoIP, video and Web flows to evaluate the performance of CATNAP.

6.2.1 Online Games

Among many genres of games widely played by residential users, such as First Person Shooter (FPS), Real Time Strategy (RTS) and Massively Multi player Online (MMO), FPS games are generally more fast paced and require more frequent user interactive actions than other genres. This study utilizes an NS-2 traffic generator for Quake 4, a popular FPS game with an Auto

Regressive Moving Average (ARMA) model [162, 163]. Cricenti *et al.* apply ARMA (1,1) model to capture the correlated nature of Quake 4 traffic through a combination of an auto regressive (AR) component and a moving average (MA) component. The parameter (1,1) describes the number of terms in the AR and MA components respectively. Cricenti *et al.* demonstrate that the ARMA(1,1) process models the server to client packet size distribution of Quake 4 better than the simpler AR(1) process [162].

The simulated game server sends a packet with average payload size of 69.5 bytes, excluding IP, Transport and Application layer headers, every 50 ms. The game client sends a payload with an average size of 64.5 bytes, excluding all headers, to the game server every 10.75 ms. However, UDP game packets are 8 bytes smaller than TCP game packets because of the size differences between UDP and TCP headers.

6.2.2 Voice over IP (VoIP)

VoIP applications send small packets carrying audio information during small time intervals. Although VoIP applications are built over a variety of VoIP protocols, the overall bandwidth usage of VoIP application is small, usually less than 80 Kbps [164]. This study implements a VoIP traffic generator to simulate VoIP flows with G.711 codec, which is one of the first audio codecs used by VoIP protocols and still widely supported by various VoIP applications such as Skype[1] and Yahoo Talk [165]. The VoIP traffic generator is able to simulate either TCP-based or UDP-based VoIP flows. Based on the G.711 specification, the VoIP traffic generator transmits a constant rate stream consisting of packets with 172 byte transport layer payload between two nodes every 20 ms.

6.2.3 Streaming Video

There are many applications and protocols for video streaming along with a variety of video codec. This study utilizes the NS-2 video application plug in developed by researchers at Arizona State University [166, 167]. The video frames used by the video generator are extracted from a pre-encoded video traces, *Indiana Jones I: Raiders of the Lost Ark*, and encoded in single layer Common Intermediate Format (CIF) at 30 frame per second (fps) with a resolution of 352×288 pixels. The Group of Pictures (GoP) consists of 16 frames with 3 B-frames between each pair of I/P frames.

[1] G.711 as the first codec implemented by Skype team http://devforum.skype.com/t5/Audio-Video/Forcing-G-711/td-p/34.

The quantization scales used for I, P, and B frames are 10, 10, and 12, respectively. Note, the simulated video flows can be transmitted with either TCP or UDP protocol.

6.2.4 Web Browsing

The characteristics of Web traffic largely depend on the content of Web page. However, the duration of a Web session is short, usually ending in a couple of seconds. Therefore, treating Web traffic is challenging because Web flows might terminate before any treatment can take effect. Consequently, CATNAP may not be able to improve the performance of Web applications even when it can correctly and promptly classify Web flows.

Although Web traffic analysis has been done with core routers by numerous researchers, little describes Web traffic from the residential end-user's perspective. This study utilizes a Web traffic model based on a study conducted by a group of researchers from IBM [9] and enhanced the Web traffic model with a NS-2 Gamma random number generator [8]. Table 6.4 summarizes the Web model and its parameters used in this study.

Table 6.4: Simulation Parameters for Web Flows [8, 9]

Parameters	Mean	Std. Dev.	Best Fit (Parameters)
HTML Object size	11872(Max 2MB)	38036	Truncated Lognormal (μ 7.90272, δ 1.7643)
Embedded Object size	12460(Max 6MB)	116050	Truncated Lognormal (μ 7.51384, δ 2.17454)
Number of Embedded Objects	5.07 (Max 300)		Gamma (κ 0.141385, θ 40.3257)
Parsing Time	3.12 (median 0.30 (Max 300 sec.)	14.21	Truncated Lognormal (μ $-$ 1.24892, δ 2.08427)
Embedded Object Inter-Arrival Time	0.83	8.4	Weibull (α 0.2089, β 0.376)
Reading Time	39.70	8.4	Truncated Lognormal (μ $-$ 0.495204, δ 2.7731)
Request Size	318.59	179.46	Uniform(350 Bytes)

The sequence of the simulated Web session assumes that all the objects in the requested Web page are placed on the same Web server; and the Web client and server simulate $HTTP1.1$ for a persistent connection and pipelining. The Web client starts with a built-in TCP traffic generator to connect to the Web server, and sends a request to the Web server after establishing a connection. Upon receiving the request from the Web client, the server responds with one HTML object. After

receiving the HTML object, the Web client calculates the number of embedded objects and the size of each embedded object, and sends the corresponding requests to the Web server. If the requested Web page contains at least one embedded object, the Web client transmits another request for each embedded object to the Web server. Finally, the Web server responds to the request with the simulated objects.

6.2.5 File Downloading

In this simulation study, file downloading traffic is generated by the built-in FTP application in NS-2. However, the TCP's advertised window is not implemented in either the default TCP or FULLTCP headers because the NS-2 simulator assumes that the receiving buffer size on TCP sender or receiver is unlimited. Thus, this study enhances the built-in TCP and FULLTCP agent in NS-2 simulator by adding the advertised window data structure into the TCP header structure, and modifying the corresponding code at the sender and the receiver sides.

Because the NS-2 TCP implementation does not have complete TCP headers and most of the TCP header flags are missing. Both the CATNAP classifier and the round trip time (RTT) estimator depend on the TCP headers and flags. Thus, this study implements the advertised window mechanism based on the NS-2 FULLTCP header. Unlike the TCP NS-2 implementation, most popular TCP congestion control mechanisms, such as TCP CUBIC or TCP Compound, are not supported by the NS-2 FULLTCP agents. We are only able to simulate TCP flows with TCP NewReno, Reno or Tahoe, and this study chooses TCP NewReno as its default TCP flavor.

P2P downloads are only a collection of multiple FTP downloads [8], and the downstream behavior of P2P application will be very similar to a file downloading flow. Thus, CATNAP would classify P2P flows as non-interactive, response-based, and greedy as it classifies FTP flows. Therefore, this simulation traffic does not include P2P applications.

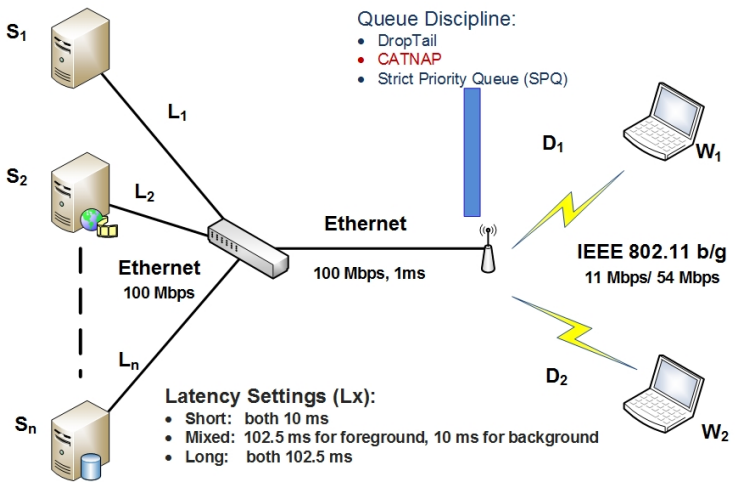
In typical residential networks, there are few non-response-based, non-interactive and greedy flows because firewalls classify these flows as malicious traffic and reject them[2]. One possible candidate of this kind of applications is NFS with UDP over a local area network. NFS clients and servers can communicate to each other through either UDP or TCP. When an NFS file is transmitted over UDP, the NFS application layer does the file integrity check and retransmits the

[2]In enterprise level Data center or other reliable Ethernet environment, iSCSI with UDP flows are non-response-based,non-interactive and greedy flows. Some Fiber Channel over Ethernet (FCoE) flows can be classified and treated as non-response-based, non-interactive and greedy.

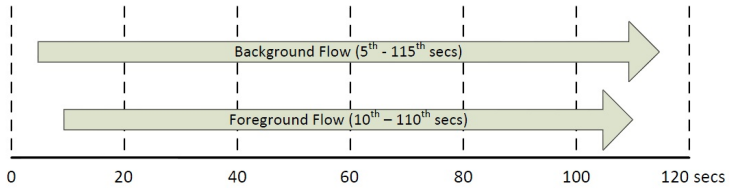
missing part.

NFS file transfer is simulated over UDP with a high volume UDP flow. A high volume UDP flow of 32Mbps, higher than the maximum effective capacity of IEEE 802.11g link, is used to stress the wireless link and our CATNAP queue controller. For evaluation purposes, these high volume UDP flows can be taken as misbehaved flows to stress wireless links.

6.2.6 NS-2 Simulation Setup



(a) Simulation Setup



(b) Flow Timeline

Figure 6.10: Simulation Setup

Figure 6.10(a) illustrates the topology of our simulation test bed, and Figure 6.10(b) shows the flow schedule plan for all core simulation runs. Two wireless nodes, W_1 and W_2, are connected to the access point (AP) with distance D_1 and D_2 respectively. The wireless nodes and AP are connected with each other through a single channel IEEE 802.11 infrastructure network at two different link speeds: 54 Mbps (IEEE 802.11g) or 11 Mbps (IEEE 802.11b). An IEEE 802.11b network is used to simulate 802.11 nodes with poor signal strength because the nodes with weak reception tend to roll back to the 802.11b rate. However, rate adaptation is turned off for this study

due to the lack of support in the NS-2 simulator.

Just as in most residential APs, this simulation study set most of the wireless layer parameters as their default values: RTS/CTS is disabled for all wireless nodes, MAC layer retransmissions are set to 4, and the wireless AP is configured to transmit a beacon frame every 100 ms.

The NS-2.33 simulator supports three different radio propagation models: free space, two-ray ground reflection and shadowing [157]. However, both the free space and two-ray ground models represent wireless communication channels in a open flat space. The shadowing model is a realistic and widely used signal fading model for indoor wireless environments. Based on the NS-2 user manual [157], parameters of shadowing model are selected to represent a typical home environment partitioned into several rooms. Specifically, the study chooses the path loss exponent of the shadowing model as 4.0, and the shadowing deviation as 7.0. The path loss exponent of 4.0 corresponds to an obstructed indoor environment while standard deviation of 7.0 corresponds to an office with hard partitions [157] [3]

As Figure 6.10(a) shows, the wireless AP is connected to the gateway with a duplex 100 Mbps Ethernet link with 1 ms latency, which are typical settings of most residential networks. The gateway is connected with multiple wired servers over symmetric 100 Mbps links with various latencies of L_1 to L_n.

The simulated AP is implemented with different queuing disciplines: DropTail, Strict Priority Queue (SPQ), and CATNAP, for the downstream traffic coming from the wired nodes S_1 through S_n to the wireless nodes W_1 and W_2. The queue capacity limit is configured to q packets. We set the CATNAP queue capacity as 1000 packets because CATNAP tries to treat flows rather than simply drop packets. The queue capacity limit for DropTail and SPQ is decided by the product of bandwidth and round-trip time (RTT). Table 6.5 shows the queue capacity limits for DropTail and SPQ under different combinations of RTT and link capacities.

Table 6.5: Queue Capacity of DropTail and SPQ

Link Speed	54 Mbps	11 Mbps
$max(L_1, L_2) = 10ms$	90 pkts	20 pkts
$max(L_1, L_2) = 102ms$	900 pkts	200 pkts

The distances ($D1$ and $D2$) between the AP and wireless nodes are set to 5 meters to simulate the wireless node in a different room than the AP. Unless specified, the parameters used by CATNAP

[3] There is no clear documents on which parameter should be used for residential places, thus we choose the one most close to dry wall.

are listed in Table 5.10.

Two types of network traffic are simulated in the test bed illustrated in Figure 6.10(a) (a). Each wireless node serves as a destination for one application: wireless node W_1 runs with Foreground Application (Flow 1), which starts at the 10th second and ends at the 110th second for a total duration of 100 seconds; wireless node W_2 is the destination node of background application (Flow 2), which starts at the 5th second and terminates at the 115th second with the total duration of 110 seconds. The background and foreground application run concurrently between the 10th second and 110th seconds. However, only the measurements and quality metrics between the 15th and 105th seconds with 90 second simulation duration are used for analysis in order to exclude the initial and final states when the foreground flow joins and departs. Note, the SPQ queue controller always prioritizes the foreground flow over the background flow even when they are same kind of traffic.

Two applications, FTP or NFS with UDP, run as background flows to stress the wireless link. CATNAP classifies both FTP and NFS with UDP as non-interactive and greedy. While CATNAP classifies FTP flows as response-based, and NFS with UDP flows as non-response-based. Therefore, CATNAP applies different treatments to constrain their rates when congestion occurs: reducing advertised window sizes for FTP flows and dropping packets from NFS over UDP flows.

Table 6.6: Applications Used for Simulation

Background Flow	Foreground Flow			
	Apps	Type		
FTP	FTP	response-based	non-interactive	greedy
	NFS w/UDP	non-response-based	non-interactive	greedy
	Game w/UDP	non-response-based	interactive	non-greedy
	Game w/TCP	response-based	interactive	non-greedy
	VoIP w/UDP	non-response-based	interactive	non-greedy
	VoIP w/TCP	response-based	interactive	non-greedy
	Video w/UDP	non-response-based	non-interactive	non-greedy
	Video w/TCP	response-based	non-interactive	non-greedy
	Web	response-based	both	both
NFS w/UDP	FTP	response-based	non-interactive	greedy
	Game w/UDP	non-response-based	interactive	non-greedy
	Game w/TCP	response-based	interactive	non-greedy
	VoIP w/UDP	non-response-based	interactive	non-greedy
	VoIP w/TCP	response-based	interactive	non-greedy
	Video w/UDP	non-response-based	non-interactive	non-greedy
	Video w/TCP	response-based	non-interactive	non-greedy

As Table 6.6 shows, foreground flows consist of applications from each category of the CATNAP

architecture. Note, Web flows may be either interactive or non-interactive depending upon the contents of the Web pages, such as number of objects and sizes of objects.

In addition to the wireless latency, the latency between the wireless nodes and the application server also have significant effects on the application performance. This simulation study chooses three latency configurations based on the measurement study conducted by Maier *et al.* in Germany [139], in which they report the round-trip times (RTTs) of flows over residential networks are with two modes of 20 ms and 205 ms. Thus, this study chooses three latency configurations:

1. *short*: both flows with 10 ms one-way latency;
2. *long*: both flows with 102 ms one-way latency;
3. *mixed*: the foreground flow with 102 ms one-way latency, and the background flow with 10 ms one-waylatency.

Note, because it is difficult to stress IEEE 802.11g link with a background flow with 102ms one-way latency, this study eliminates the other mixed case where the background flow is with 102 ms latency and the foreground flow is with 10 ms one-way latency.

6.3 Core Results

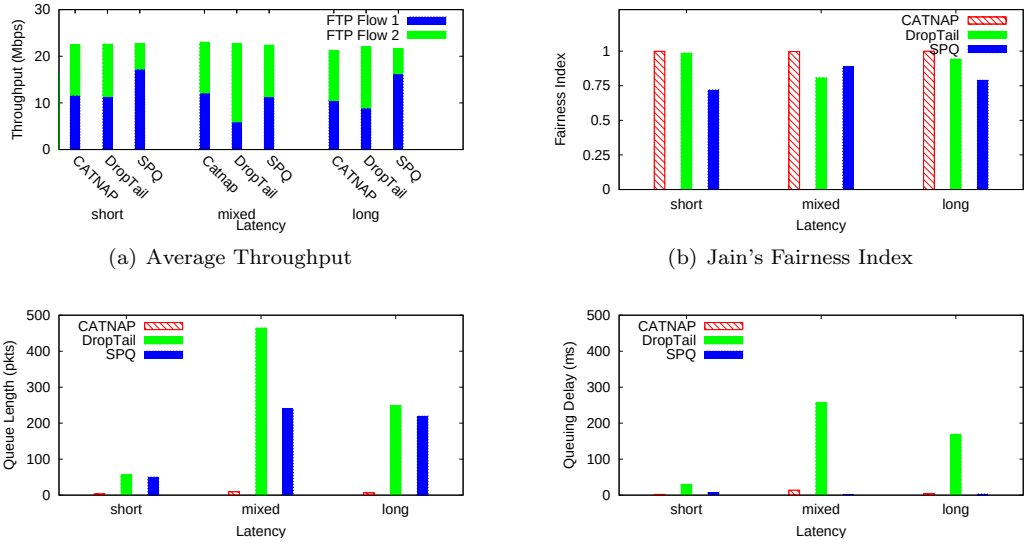
This section summarizes the core simulation results which compare the performance between CATNAP, DropTail and SFQ queues. To reduce the uncontrolled variables in simulations, the *core* simulation study only involves two concurrent flows, one is the foreground flow and the other is the background flow. As Table 6.6 shows, FTP and UDP based NFS flows act as the background flow to stress the wireless link, while VoIP, game, and video applications with tight QoS requirements act as foreground flows.

6.3.1 File Downloading Applications as Foreground Applications

CATNAP classifies FTP flows as response-based, non-interactive, and greedy. This study generally uses FTP flows as background flows to stress the wireless link. This section, however, evaluates the performance of the three queue controllers when two FTP flows are concurrently passing through them. Figure 6.11 and Figure 6.12 summarize the results with two current FTP flows, one as

foreground (FTP Flow 1) and the other as background (FTP Flow 2), under IEEE 802.11g and IEEE 802.11b link respectively.

Figure 6.11 (a) shows the cumulative downstream throughput of two FTPs over IEEE 802.11g link. The two stackable bars represent the average throughput for each flow. All three queue controllers give similar average throughput around 20-21Mbps, which is approximately the effective TCP throughput over an IEEE 802.11g network. However, the throughput of CATNAP is slightly lower than DropTail or SPQ because the CATNAP bandwidth estimation algorithm is conservative in order to lower queuing delay for interactive applications by strictly controlling the queue length.



(a) Average Throughput

(b) Jain's Fairness Index

(c) Queue Length

(d) Queuing Delay of Foreground FTP

Figure 6.11: Two FTP Flows over IEEE 802.11g Link

The stackable bar graph in Figure 6.11 (a) also discloses that SPQ and DropTail do not treat FTP flows fairly, especially under the mixed latency configuration. SPQ is expected to be unfair because it always prioritizes the foreground flow even when the foreground flow is the same type as the background flow. Therefore, we exclude the analysis of SPQ controller in the discussion of fairness. The DropTail queue controller, however, tends to prioritize the FTP flow with shorter RTT over the FTP flow with longer RTT.

Figure 6.11 (b) presents the "unfairness" by using Jain's Fairness Index, which is calculated by the following equation:

$$fairness\_index = \frac{(\sum x_i)^2}{n \times \sum x_i^2} \tag{6.1}$$

where x_i is the throughput of i_{th} flow and n stands for the number of flows.

The closer Jain's Fairness Index to 1, the fairer the queue controller is to the flows. When both FTP nodes have the same latency from the servers, they achieve about the same average throughput for DropTail and CATNAP. Jain's Fairness Index for FTP flows under DropTail and CATNAP for both FTP are close to 1. However, for the mixed configuration, when the foreground FTP flow has 102 ms one-way latency and the background flow has 10 ms one-way latency, CATNAP treats them more fairly than DropTail because DropTail drops Jain's Fairness Index to 0.8 but CATNAP maintains Jain's Fairness Index as high as 0.98.

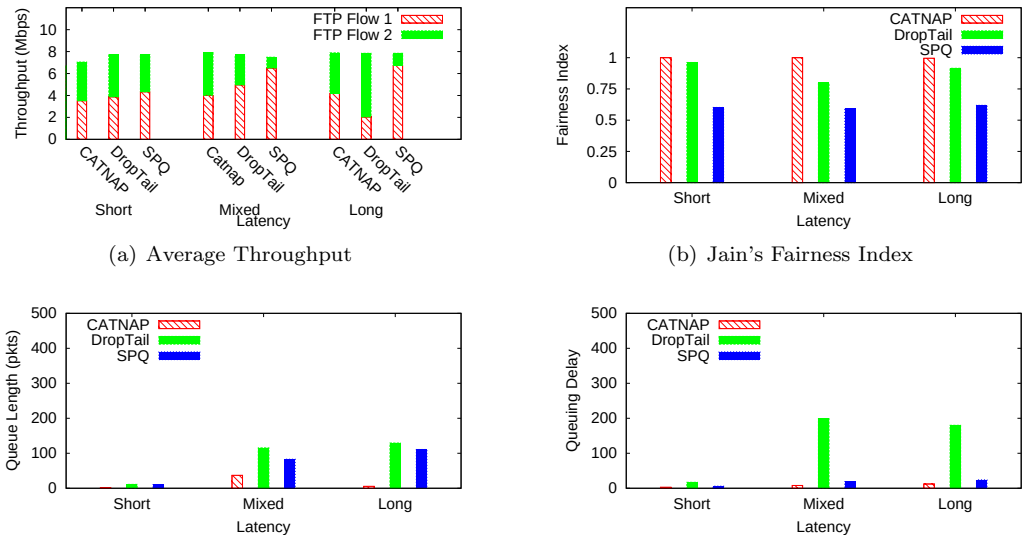
Figure 6.11 (c) shows the internal queue buffer usage of each queue controller. CATNAP more effectively controls queue length than DropTail or SPQ. Because the queue capacity of SPQ and DropTail (listed in Table 6.5), is carefully selected to avoid any queue drops inside the AP due to the limitation of queue size, DropTail and SPQ maximize the throughput of TCP flows passing through them. However, as a trade-off, a large queue capacity significantly increases the queuing delay, and eventually impairs the performance of interactive applications.

Figure 6.11 (d) shows the measured queuing delay for foreground FTP flows. The SPQ controller always prioritizes the foreground flow and there is almost no queuing delay for the foreground FTP flow. Thus, it provides a "floor" for the queuing delay imposed by any queue controller. As Figure 6.11 (d) indicates, the queuing delay introduced by CATNAP is similar to SPQ, while DropTail imposes more than 150 ms queuing delay because it fills up its queuing buffer.

Figure 6.12 shows the result of all three queue controllers with an IEEE 802.11b link. Although the link capacity is reduced, the result with IEEE 802.11b link is very similar to the result shown in Figure 6.11, illustrating CATNAP has similar performance with smaller link capacities except that CATNAP outperforms the two queue controller in terms of fairness.

UDP based NFS vs. FTP

Figure 6.13 illustrates the results when a foreground UDP based NFS flow concurrently runs against a background FTP flow. Different from the FTP flows, CATNAP classifies UDP based NFS flows as non-response-based, non-interactive, and greedy. Because UDP does not have retransmissions as does TCP, this study assumes that the NFS application layer does file completeness checking and error handling. When the CATNAP AP detects that the ingress bandwidth is greater than the effective wireless downlink capacity, the CATNAP treatment module starts to drop packets from



(a) Average Throughput

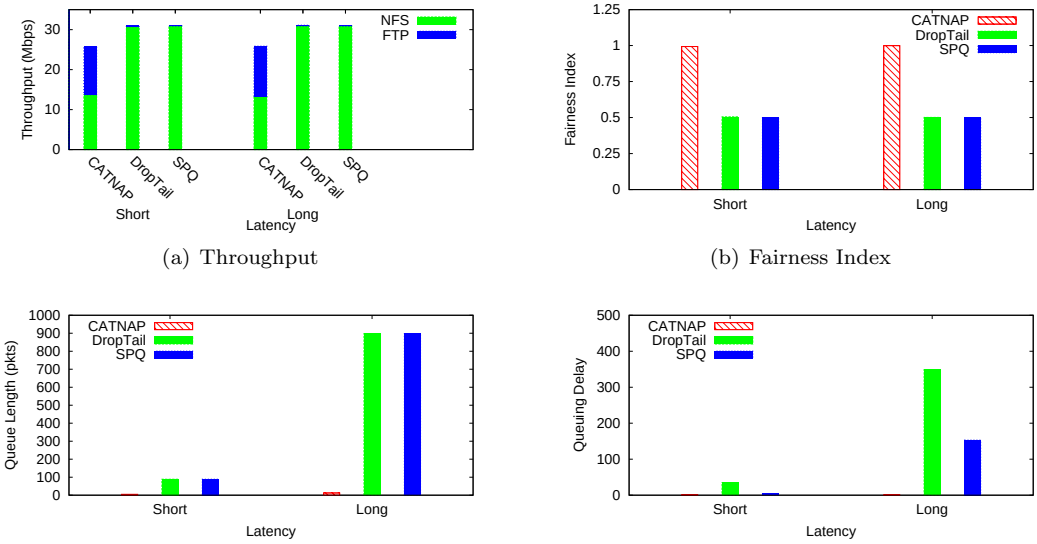
(b) Jain's Fairness Index

(c) Queue Length

(d) Queuing Delay of Foreground FTP

Figure 6.12: Two FTP Flows over IEEE 802.11b Link

non-response-based, non-interactive, and greedy flows to avoid congestion if there exists any. In the meantime, if any response-based, non-interactive, and greedy flow such as an FTP flow exists, the CATNAP treatment module informs the FTP sender to slow down by reducing the advertised window size carried by TCP ACK packets.



(a) Throughput

(b) Fairness Index

(c) Queue Length

(d) Queuing Delay of Foreground NFS

Figure 6.13: NFS w/UDP and FTP flows over IEEE 802.11g Link

Figure 6.13(a) compares the cumulative throughput of the three different APs with IEEE 802.11g links. Over IEEE 802.11g links, the saturate rate of UDP based flows is around 30-31 Mbps, about 6 Mbps higher than the saturate rate of TCP based flows. However, when UDP based flow and TCP based flows are concurrently running, the throughput of the "polite" TCP flow is greatly reduced and the TCP connection may ever be terminated. Figure 6.13(a) clearly shows that under such condition, the FTP flow hardly survives with either DropTail or SPQ. Figure 6.13(b) shows the unfairness of DropTail and SPQ under such configuration. However, on the contrary, CATNAP is able to fairly treat the UDP based NFS flow and TCP based FTP flows when they co-exist. Additionally, Figure 6.13(a) and Figure 6.13(b) exhibit the potential risk of a static priority queue, such as SPQ or port based priority scheme, where an incorrect priority setting can be harmful.

The cumulative throughput of CATNAP is only 26 Mbps, lower than throughput of DropTail or SPQ whose throughput is dominated by the UDP foreground flow. When TCP and UDP flows co-exist, the possible range of saturated throughput of IEEE 802.11g should be between 23 Mbps (TCP only) and 31 Mbps (UDP only). However, there is no theoretical study to show the saturated throughput when heavy volume UDP and TCP flows are currently running[4]. More accurately estimating the downlink capacity under different traffic loads is addressed in Section 6.4.3.

Figure 6.14 shows the results when a UDP based NFS flow and a FTP flow are concurrently running with an IEEE 802.11b link, which are very similar to the results over an IEEE 802.11g link.

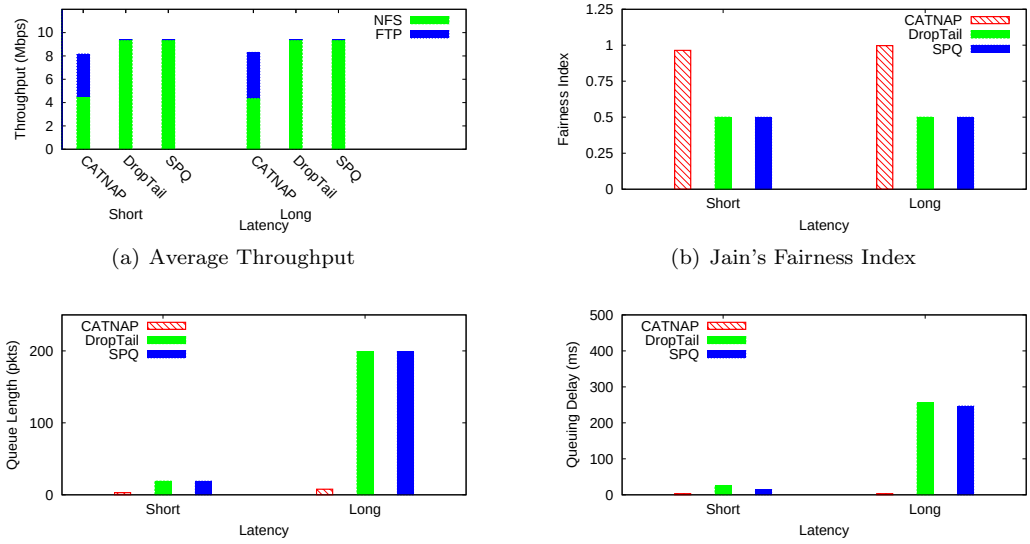
6.3.2 VoIP as Foreground Application

UDP based VoIP vs. FTP

Figure 6.15 summarizes simulation results where a foreground UDP based VoIP flow is running with a FTP background flow over IEEE 802.11g links. CATNAP categorizes UDP based VoIP flows as non-response-based, interactive, and non-greedy, and classifies FTP flows as response-based, non-interactive, and greedy.

Figure 6.15(a) shows that all three AP provide similar cumulative throughput because the VoIP flow only consumes up to 80 Kbps bandwidth which is insignificant to the bandwidth taken by the FTP background flow. Moreover, Figure 6.15(a) indicates that it is difficult for a single background

[4]Experimental study may not work because TCP flows hardly survive under the pressure of high volume UDP flows.



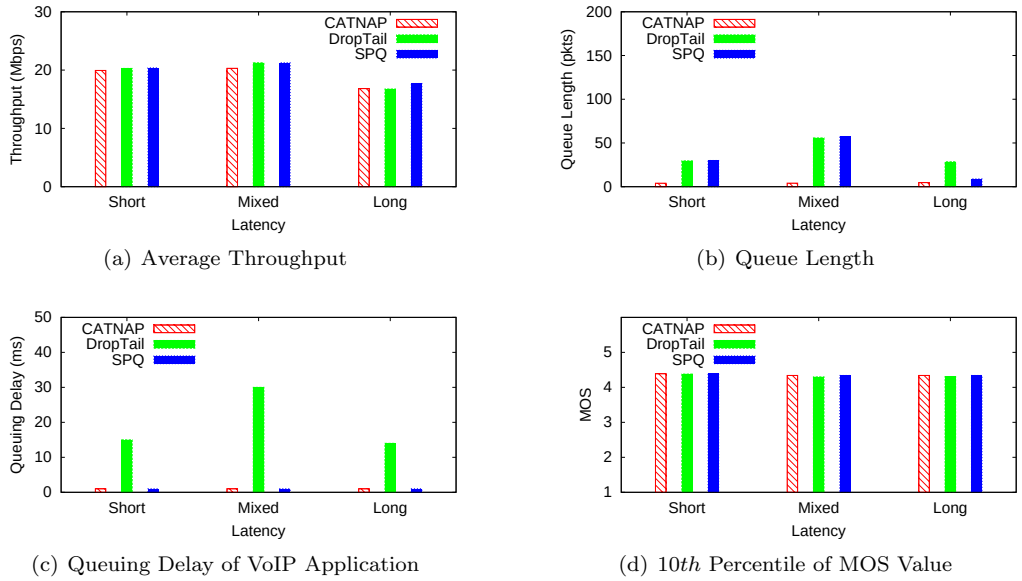
(a) Average Throughput

(b) Jain's Fairness Index

(c) Queue Length

(d) Queuing Delay of Foreground NFS

Figure 6.14: NFS w/UDP and FTP flows over IEEE 802.11b Link



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

(d) $10th$ Percentile of MOS Value

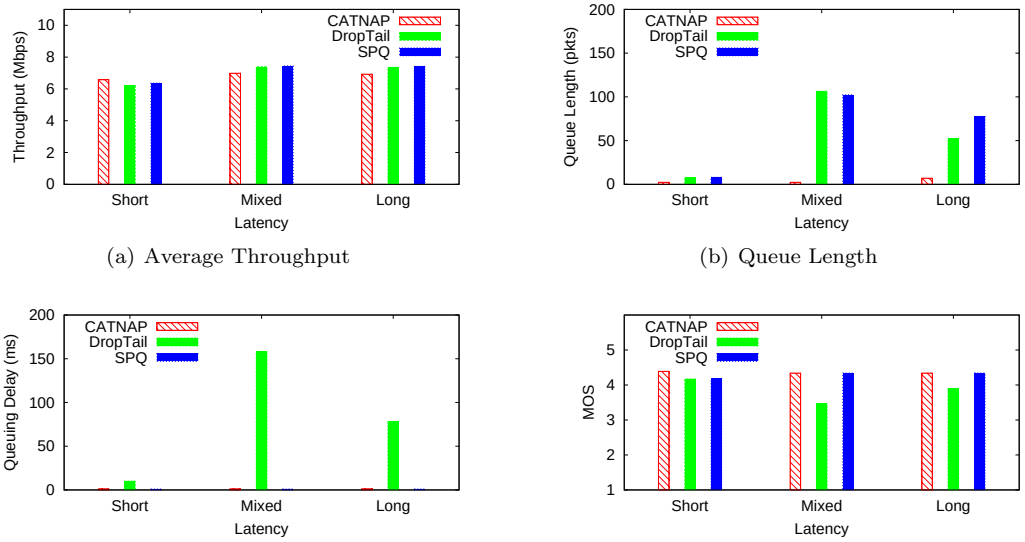
Figure 6.15: VoIP (w/UDP) and FTP over IEEE 802.11g Link

FTP flow with a long latency (102ms) to saturate IEEE 802.11g links. Under a long latency configuration, the TCP sender more likely reaches its retransmission timeout (RTO) threshold and triggers retransmissions due to any occasional RTT variances. Thus, the overall throughout with long latency configuration is only around 17 Mbps and less than throughput (20-21 Mbps) observed

in the other two configurations where the FTP flow is running with a short latency (10 ms).

Figure 6.15(b) shows the internal buffer usage for each queue controller. The wireless link is not completely saturated because no queue buffer is fully utilized in any of three APs while the queue length of DropTail is between 30 to 50 packets, much larger than the queue length (less than 10 packets) in CATNAP.

Because the wireless link is not fully saturated in all three latency configurations, as Figure 6.15(c) shows, the maximum average queuing delay observed in this simulation is around 30 ms, which is not large enough to degrade the quality of the VoIP applications. The VoIP quality indicator, MOS, is calculated every talkspurk (1.0 seconds). The $10th$ percentile of all MOS values for one run is used as indicator of overall quality shown in Figure 6.15(d). The VoIP qualities under all configurations are similar, though the quality with DropTail queue degrades to 4 but quality with CATNAP or SPQ stays with 4.39.



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

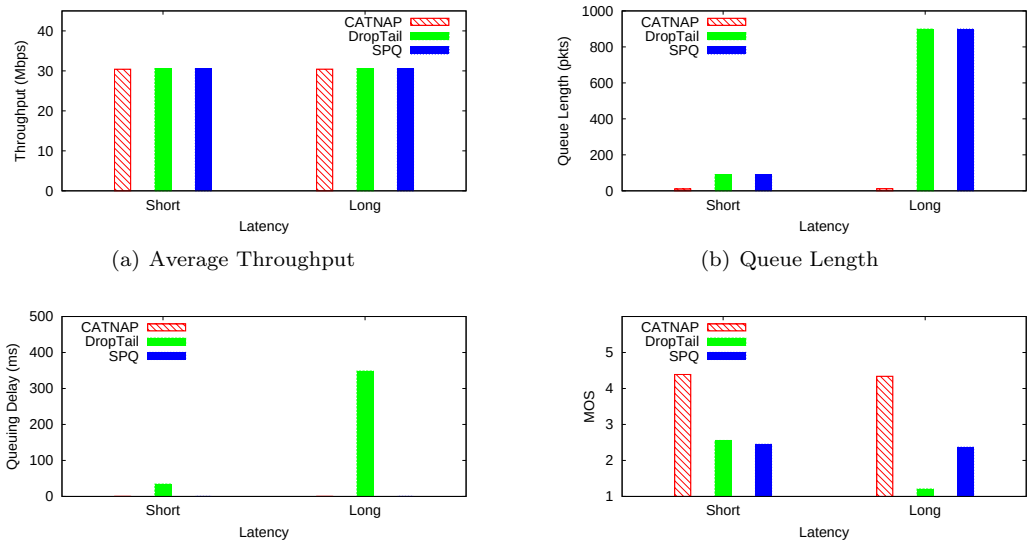
(d) $10th$ Percentile of MOS Value

Figure 6.16: VoIP (w/UDP) and FTP over IEEE 802.11b Link

Figure 6.16 shows the results when the foreground VoIP and background FTP flows are concurrently running over IEEE 802.11b links. Unlike IEEE 802.11g links, the effective downlink capacity of IEEE 802.11b links is only 6-9 Mbps. Because the TCP sender can adjust its sending speed by observing RTT and counting sequence number of ACKs, a single FTP flow with a large RTT still hardly saturates IEEE 802.11b link. However, as Figure 6.16 (b) shows, the background FTP

background flow is able to fill a 50-100 packet queue inside the DropTail AP or the SPQ AP under the long and mixed configurations. In consideration of the low link speed of IEEE 802.11b links, 50-100 enqueued packets are still able to dramatically increase queuing delay and impair the VoIP quality. The VoIP quality with the DropTail AP degrades below 4.0 and enters the "fair" zone. On the contrary, the VoIP flow over CATNAP has similar quality as on SPQ: the VoIP with these two APs maintains the high VoIP quality as high as 4.34, which is the ceiling of MOS value for G.711 codec.

UDP based VoIP vs. NFS w/UDP



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

(d) 10*th* Percentile of MOS Value

Figure 6.17: VoIP (w/UDP) and NFS (w/UDP) over IEEE 802.11g Link

Figure 6.17 shows the results when the foreground UDP based VoIP flow and the background UDP based NFS flow are concurrently running with IEEE 802.11g links. CATNAP classifies the UDP based NFS flow as non-response-based,non-interactive, and greedy. Because of the lack of rate control mechanisms and connectionless, the UDP based NSF flow is able to easily saturate wireless link, and triggers congestion in APs.

To saturate IEEE 802.11g wireless links, this study intentionally configures the UDP sender to send packets at 32 Mbps which is a little bit higher than the effective throughput of UDP flows as 30 Mbps over IEEE 802.11g links. Figure 6.17(a) clearly shows that the cumulative throughput of all

three APs reaches 30 Mbps. Note, we do not show the result with the mixed configuration, because the UDP throughput is not related to latency settings and the result with mixed configuration is almost same as the result with long configuration.

Figure 6.17(b) illustrates internal buffer usage of three APs. DropTail and SPQ almost reach their queue capacity limits, and 90+% of their buffer spaces are filled up for the long latency configuration. With a such large queue size, DropTail imposes high delay for foreground VoIP flows (Figure 6.17(c)) and dramatically degrades the VoIP quality.

The large queue length inside the SPQ also impairs the quality of the foreground VoIP application. Because SPQ runs out of room in its queue buffer, the foreground VoIP flow experiences around 7-8% packet loss, and the VoIP quality is significantly degraded by such a high loss. On contrary, the VoIP quality with CATNAP still remains 4.38 because CATNAP AP intentionally drops packets from the greedy NFS flow to avoid congestion and protects the interactive VoIP flow by giving it higher priority.

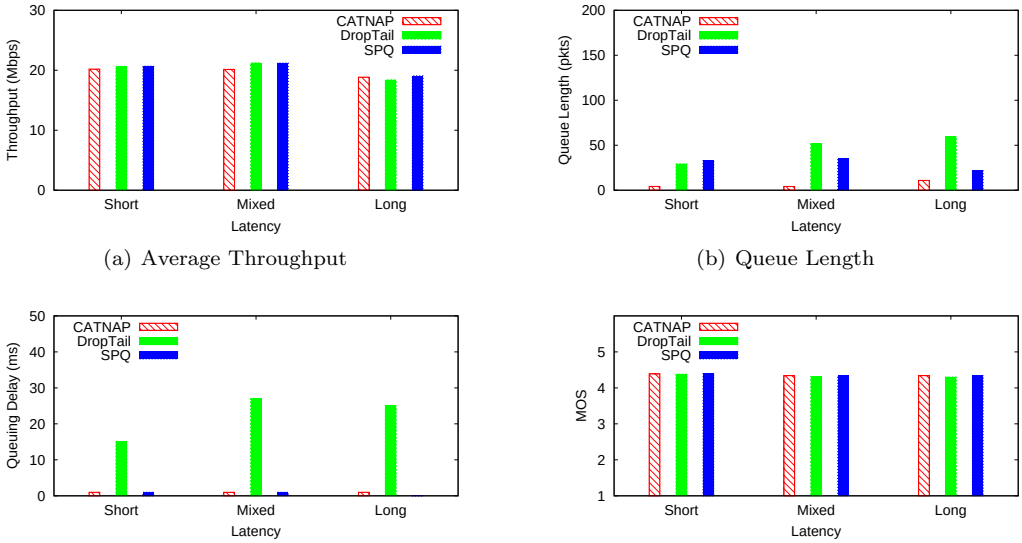
TCP-based VoIP vs. FTP

TCP-based VoIP applications have become more and more popular over residential networks because TCP-based applications can easily go through widely deployed firewalls. CATNAP classifies TCP-based VoIP flows as responses-based, interactive and non-greedy. Even when the congestion occurs, CATNAP still protects the TCP-based VoIP flows without dropping packets or reducing their transmission rate by shrinking their advertised window size.

However, CATNAP is not able to improve the quality of VoIP application much with all three latency configurations. Figure 6.18(d) shows that there is not much difference between CATNAP, DropTail and SPQ under all three configurations. Figure 6.19(d) indicates that the CATNAP improves the quality of VoIP flows better than DropTail over the IEEE 802.11b links. In general, the result of TCP-based VoIP flows is similar to the result of UDP-based VoIP flows.

Because the TCP protocol retransmits dropped packets, the loss ratio of TCP VoIP is calculated based on the number of TCP packets received during a one-second talk spurt.[5] In reality, retransmission of VoIP packets is also not helpful to improve VoIP quality if the VoIP recipient might run out its playback buffer before the retransmitted packets arrive. Retransmission of VoIP packets may not be able to improve the VoIP quality at all.

[5]The VoIP recipient knows that the sender sends 50 packets per second.



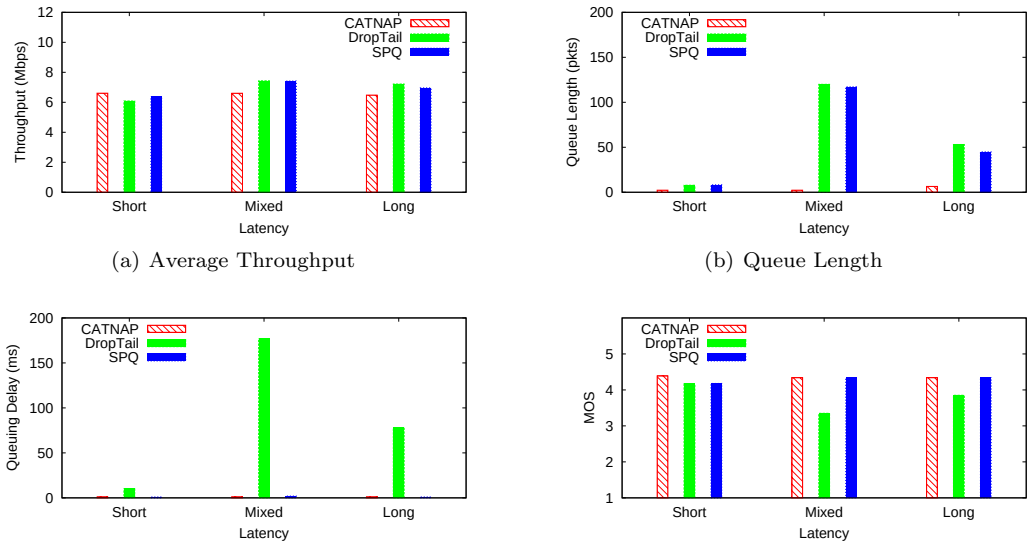
(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

(d) 10*th* Percentile of MOS Value

Figure 6.18: VoIP (w/TCP) and FTP over IEEE 802.11g Link



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

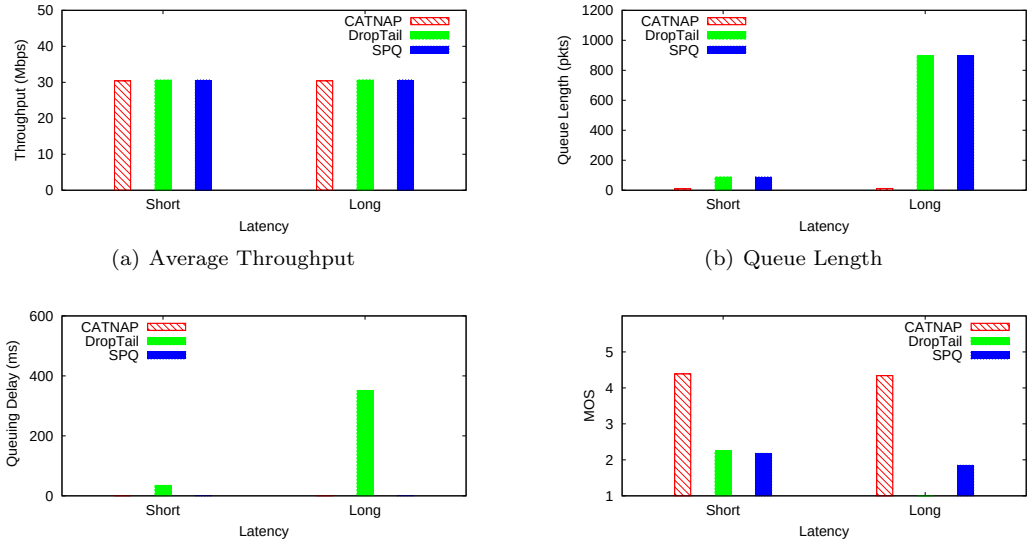
(d) 10*th* Percentile of MOS Value

Figure 6.19: VoIP (w/TCP) and FTP over IEEE 802.11b Link

When the UDP-based NFS flow acts as the background flow, CATNAP actually improves the quality of TCP-based VoIP flows. Figure 6.20(d) shows that CATNAP improves the VoIP quality with both configurations.[6] As expected, all three wireless AP provide similar cumulative throughput

[6]The result with IEEE 802.11b network is in Appendix Table A.2.

as much as 30 Mbps. However, the high volume UDP flow consumes almost all of the queue space inside DropTail and SPQ. Consequently, DropTail or SPQ drops a large fraction (7-10%) of TCP-based VoIP packets and significantly impairs the VoIP quality. The quality of the VoIP flow with DropTail or SPQ drops as low as 2.3. On the contrary, CATNAP actively drops 7% of the packets from greedy NFS flows, and maintains a high MOS (4.39) for TCP-based VoIP flows.



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of VoIP Application

(d) $10th$ Percentile of MOS Value

Figure 6.20: VoIP (w/TCP) and NFS (w/UDP) over IEEE 802.11g Link

6.3.3 Game as Foreground Application

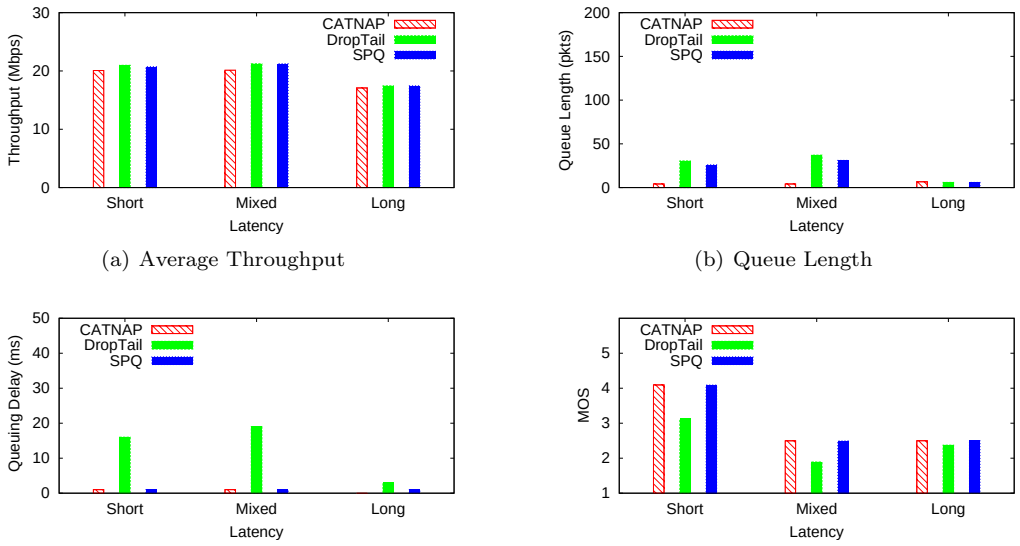
Game applications are another kind of popular multimedia application running over home networks. Similar to VoIP flows, game flows usually consist of small packets, and CATNAP classifies them as interactive, non-greedy flows also. Different from VoIP applications, the quality of game applications are more sensitive to delay and jitters [162].

This study chooses the $10th$ percentile of the G-Model MOS value as the end of session quality metric. In reality, when an FPS game player experiences a couple seconds of frozen actions, he/she may disconnect and rejoin another server. Thus, this study chooses the lowest game quality as the quality indicator for a whole game session.

UDP-based Game vs. FTP

Figure 6.21 summaries simulation results where a UDP-based game flow acts as the foreground flow and an FTP flow runs as the background flow. Similar to the simulation when a VoIP flow running foreground, the cumulative throughput of three APs are also close to each other. Because the background FTP flow with long latency settings is unable to saturate IEEE 802.11g links, the cumulative throughput of the all three APs with long latency configuration are only around 17-18 Mbps. As the queue size of the three APs shown in Figure 6.21(b), DropTail and SPQ builds up a 30-40 packet queue under the short and mixed configurations. Moreover, Figure 6.21(c) indicates that DropTail imposes 20 ms delay on the game flow under the short and mixed configurations. Because game flows are less tolerant to delay changes, the small queuing delay introduced by DropTail impairs the quality of the game flow as seen in Figure 6.21(d).

Except for the short latency configuration, the game quality of all three APs are below 3 because the G-Model is more sensitive to long one way latency. With a large latency (105 ms), the maximum MOS is only 2.50 or so. Thus, neither CATNAP nor SPQ can improve the quality of game flows with such long one-way latency. Because game flows are very sensitive to latency changes, Section 6.4.1 discusses the simulations with game flows under a variety of latency settings.



(a) Average Throughput

(b) Queue Length

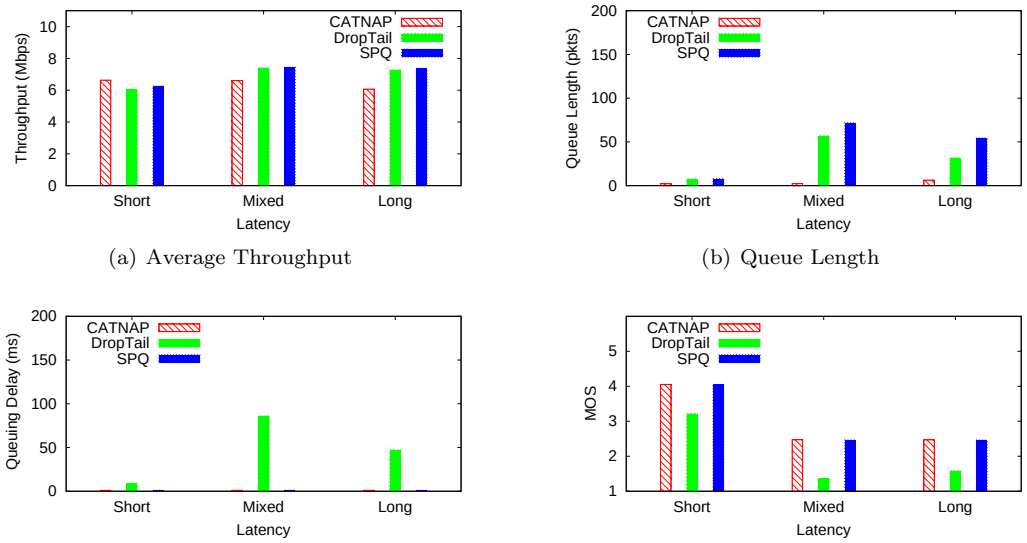
(c) Queuing Delay of Game Application

(d) 10th Percentile of MOS Value

Figure 6.21: Game (w/UDP) and FTP over IEEE 802.11g Link

Figure 6.22 shows the results where the UDP-based foreground flow and the FTP background

flow concurrently running on IEEE 802.11b links. The quality of the game flow is similar to the ones running on IEEE802.11g links except the cumulative throughput is between 7-8 Mbps because the IEEE 802.11b links provides slower link speed as 11 Mbps.



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

(d) $10th$ Percentile of MOS Value

Figure 6.22: Game (w/UDP) and FTP over IEEE 802.11b

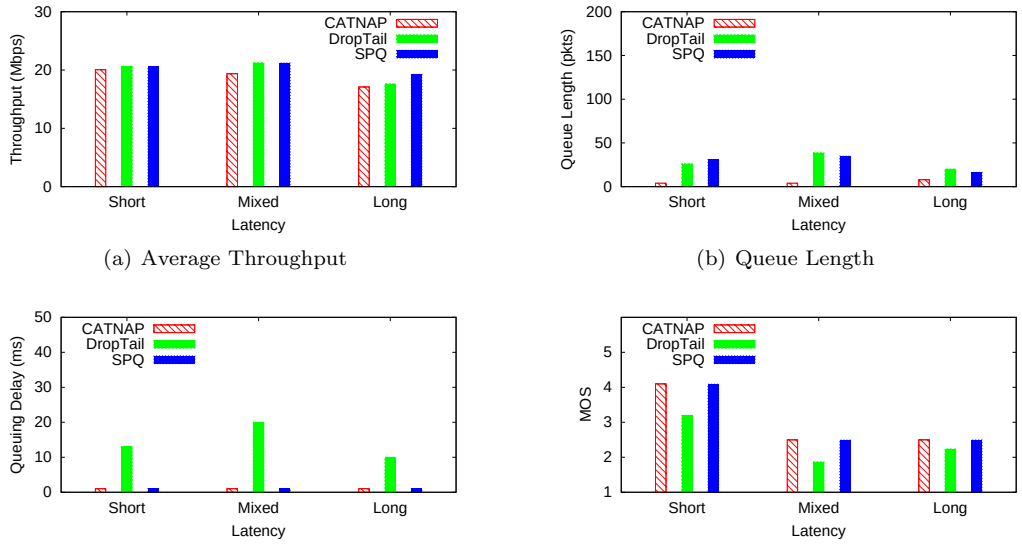
TCP-based Game vs. FTP

Figure 6.23 and Figure 6.24 show the simulation results when the TCP-based foreground game flow concurrently runs with the FTP background flow on IEEE 802.11g and IEEE 802.11b links respectively. Different from the UDP-based game flows, CATNAP classifies the TCP-based game flows as response-based. CATNAP still prioritizes TCP-based game flows over FTP flows as it does for UDP based game flows.

Game vs. UDP-based NFS

As discussed in previous sections, UDP-based NFS flows severely degrade the quality of other flows concurrently running. Game flows are unable to survive from the pressure of greedy UDP-based NFS flows on the DropTail and SPQ APs.

Because the UDP protocol does not have any rate control mechanism, a high volume UDP flow can easily fill up the queue in the DropTail or SPQ APs. As Figure 6.25(b) shows, the simulated



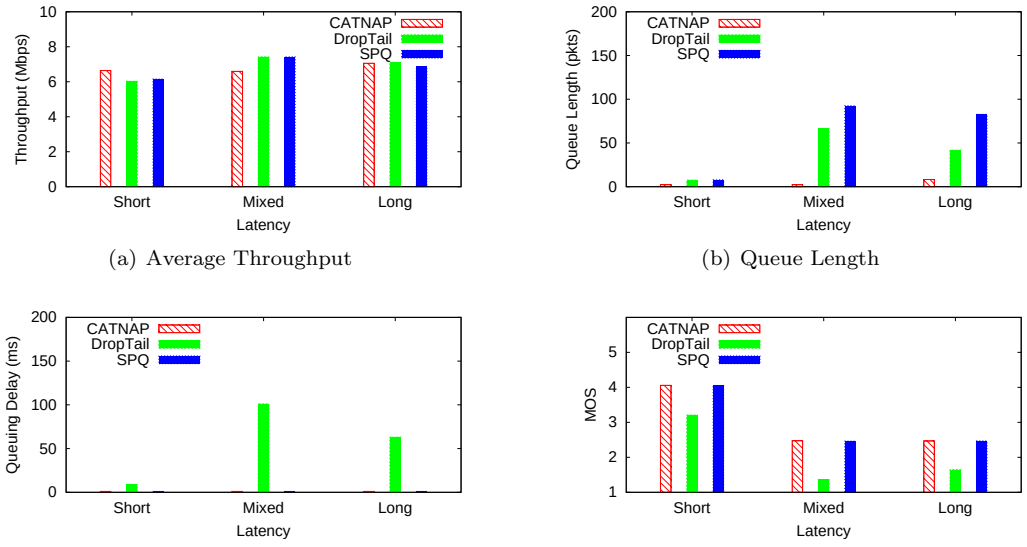
(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

(d) 10*th* Percentile of MOS Value

Figure 6.23: Game (w/TCP) and FTP over IEEE 802.11g



(a) Average Throughput

(b) Queue Length

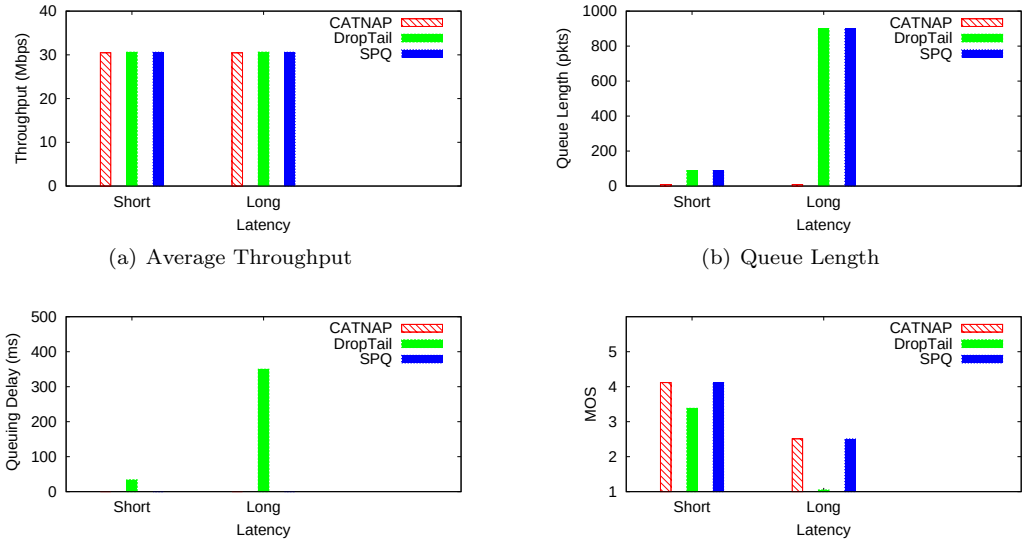
(c) Queuing Delay of Game Application

(d) 10*th* Percentile of MOS Value

Figure 6.24: Game (w/TCP) and FTP over IEEE 802.11b Link

NFS flow fills up the queue in both DropTail and SPQ. Figure 6.25(c) shows than DropTail imposes average 320 ms queuing delay to the foreground game flow. With a such large queuing delay, the quality of the game flow drops below 3.2 on DropTail even with the short latency configuration in Figure 6.25(d). However, CATNAP favorites the game flows in all configurations: CATNAP

provides the same quality for the game flow as SPQ, even with IEEE 802.11b links (shown in Figure 6.26).



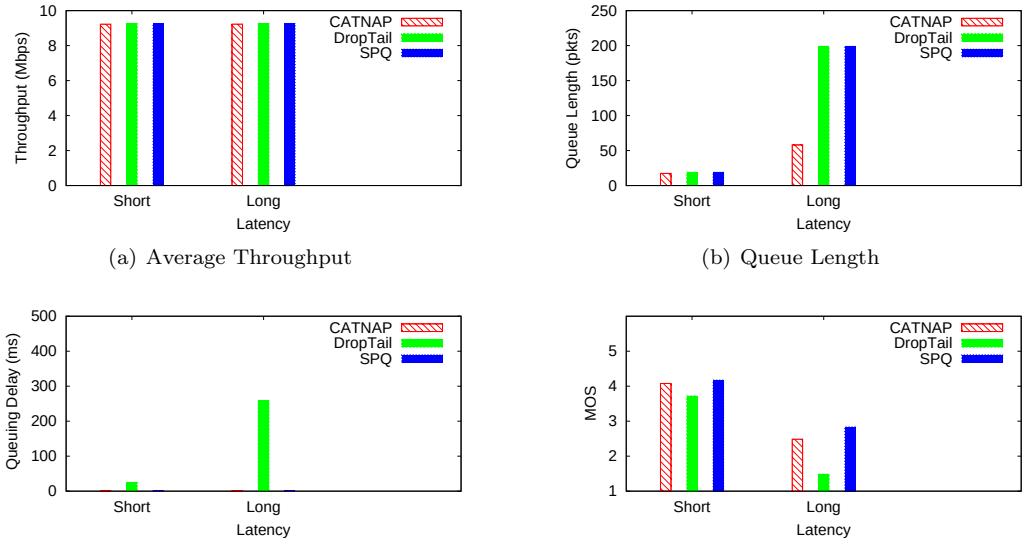
(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

(d) $10th$ Percentile of MOS Value

Figure 6.25: Game (w/UDP) and NFS (w/UDP) over IEEE 802.11g Link



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

(d) $10th$ Percentile of MOS Value

Figure 6.26: Game (w/UDP) and NFS (w/UDP) over IEEE 802.11b Link

6.3.4 Video Applications as Foreground Applications

CATNAP classifies video streaming flows as non-interactive and non-greedy flows. Different from VoIP or games flows, video streaming flows generally consists of "large" packets. However, video flows are less sensitive to large delay than VoIP and game flows because most video players are able to buffer 3-5 seconds video ahead when they play back. Meanwhile, with the help of widely deployed video repair techniques, e.g. FEC, video applications can tolerate 1% to 3% packets loss without degrading video quality [39].

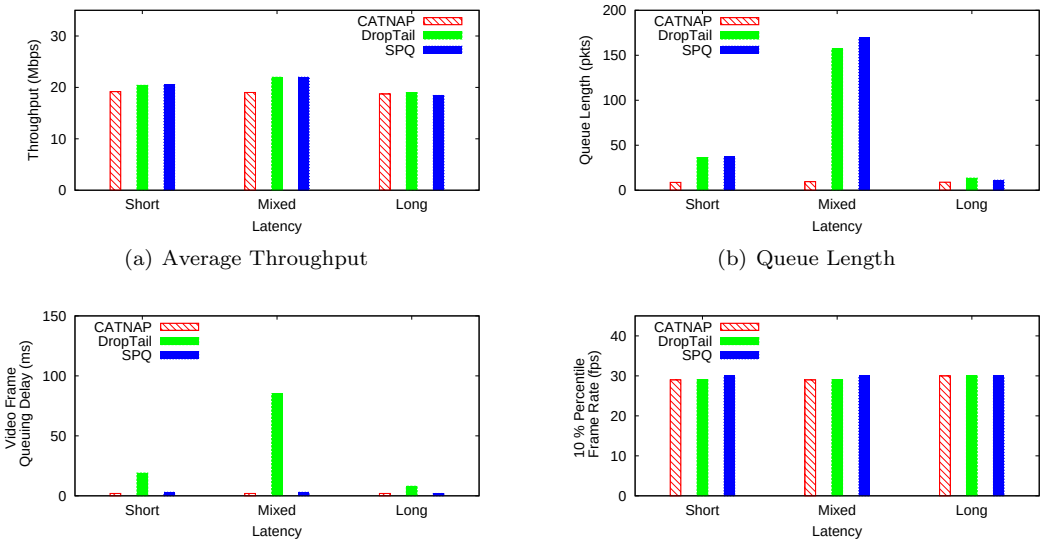
On the other hand, because the video encoding rate limits the actual network bandwidth demands of video applications, video applications are less greedy than the file downloading applications. Based on the above characteristics, CATNAP treats video applications differently than VoIP or FTP: protecting video flows without any dropping or rate limitation until its bandwidth goes over its fair share rate.

UDP-based Video Application vs. FTP

Figure 6.27 summarizes the results when the UDP-based foreground video flow is concurrently running with the background FTP flow with IEEE 802.11g links. Shown in Figure 6.27(a), CATNAP provides only 19 Mbps cumulative downlink throughput in mixed and short latency configurations, which is 1 Mbps less than DropTail or SPQ. The reason is that CATNAP does not fully take the advantage of the large queue buffer size. For example, Figure 6.27(b) shows that CATNAP still maintains a much smaller queue than DropTail or SPQ in the "mixed" configuration where DropTail or SPQ has 150 or more enqueued packets. With such a large queue, DropTail imposes 70 ms delay for the video flow. The video flow running over the SPQ AP does not degrade its quality because SPQ prioritizes the video flow.

This study chooses the $10th$ percentile of playout frame rate as the metric of video quality, which is shown in Figure 6.27(d). Although the video flows running on DropTail suffers from increased queuing delay, their quality do not degrade because the video player's playout buffer assimilates the increased delay. Figure 6.27(d) shows that all video flows experience the similar qualities with all three APs.

Figure 6.28 shows the results when video and FTP concurrently run over an IEEE 802.11b link. The cumulative throughput are decreased to 7-8 Mbps as the link capacity is reduced to 11 Mbps. Meanwhile, DropTail uses 150-200 out of 900 queuing spaces, and introduces more than 200 ms



(a) Average Throughput

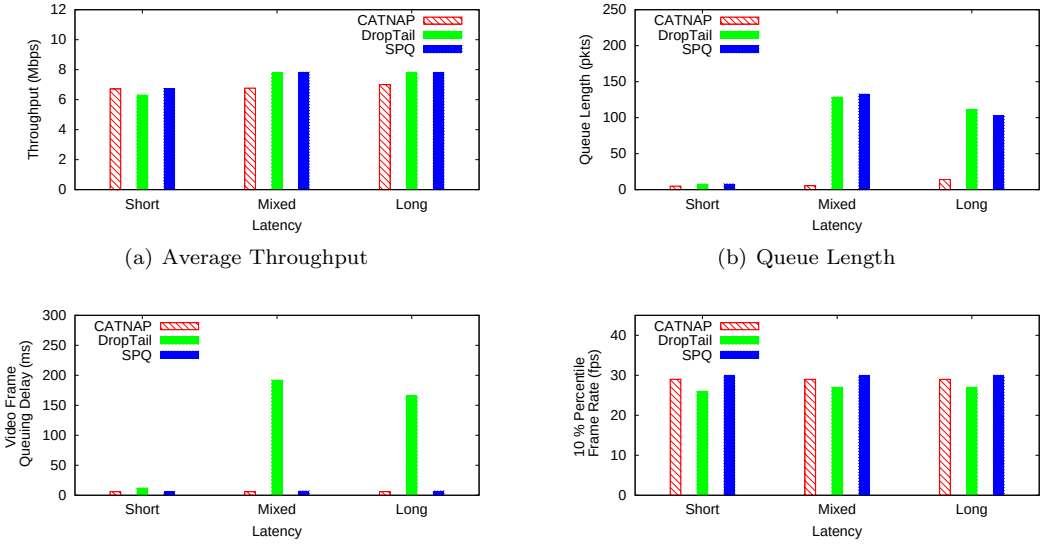
(b) Queue Length

(c) Queuing Delay of Video Application

(d) 10*th* Percentile of Video Frame Rate

Figure 6.27: Video (w/UDP) and FTP over IEEE 802.11g Link

queuing delay. Therefore, the 10*th* percentile video frame rate for the DropTail AP degrades to 25-26 fps, while CATNAP and SPQ maintain 29 fps. Although video performance with the DropTail is still in the range of good quality, considering its large queuing delay, other video applications which are sensitive to delay, such as a video conference, might be significantly impaired.



(a) Average Throughput

(b) Queue Length

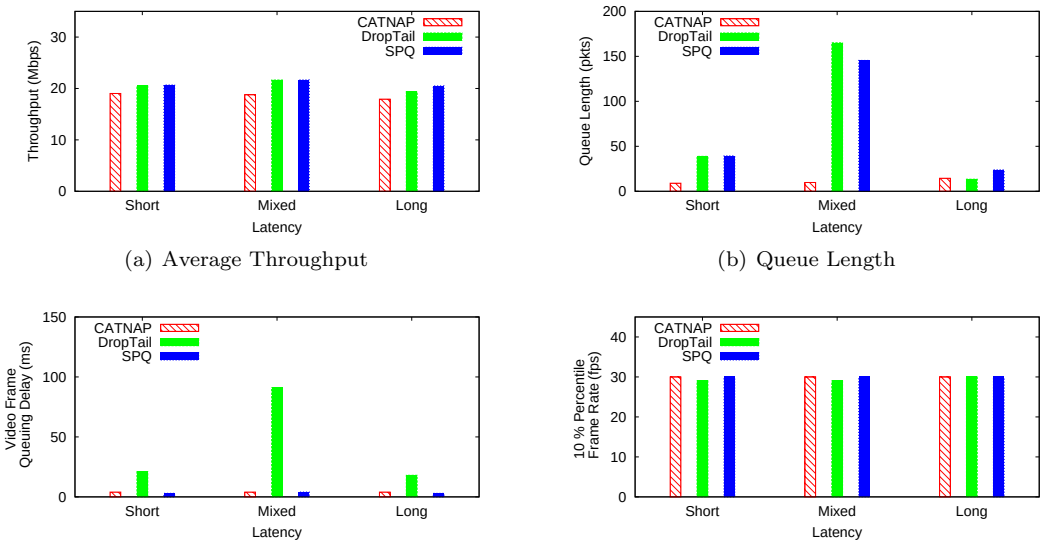
(c) Queuing Delay of Video Application

(d) 10*th* Percentile of Video Frame Rate

Figure 6.28: Video (w/UDP) and FTP over IEEE 802.11b

TCP-based Video Application vs. FTP

Figure 6.29 and Figure 6.30 show the results when TCP-based foreground video flows and FTP background flows are concurrently running over IEEE 802.11g and IEEE 802.11b networks respectively. With FTP as a background flow, there is only trivial performance differences between the TCP-based and the UDP-based video flows.



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Video Application

(d) $10th$ Percentile of Video Frame Rate

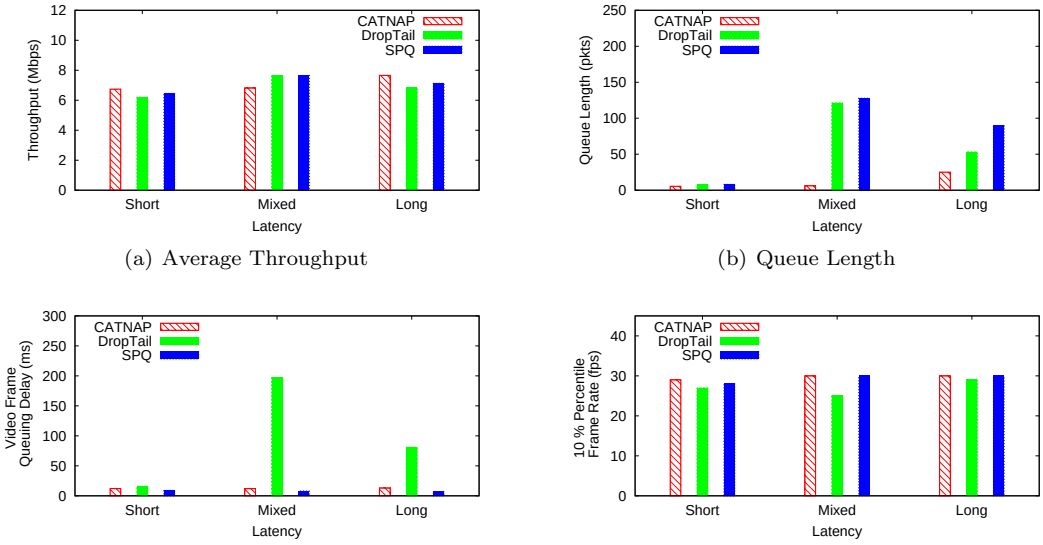
Figure 6.29: Video (w/TCP) and FTP over IEEE 802.11g

Video Application vs. UDP-based NFS

Figure 6.31 elucidates the results when the UDP-based foreground video flows and the UDP-based NFS background flow are concurrently running over IEEE 802.11g links. Similar to other simulation cases when the UDP-based NFS flow acts as a background flow, DropTail and SPQ build up large queues and almost exhaust their queuing spaces with the "long" latency configuration in Figure 6.31(b). Shown in Figure 6.31(c) A such large queue inside the DropTail introduces more than 300 ms delay for the video flow.

Although SPQ priorities the video flow and it does not introduce a large queuing delay for the video flow, SPQ has to drop 7-8% video packets because it run out of buffer spaces. Thus, as Figure 6.31(d) shows, both SPQ and DropTail degrade the video frame rate to 25 bps.

On the contrary, the video flow running with CATNAP survives from the pressure of the greedy



(a) Average Throughput

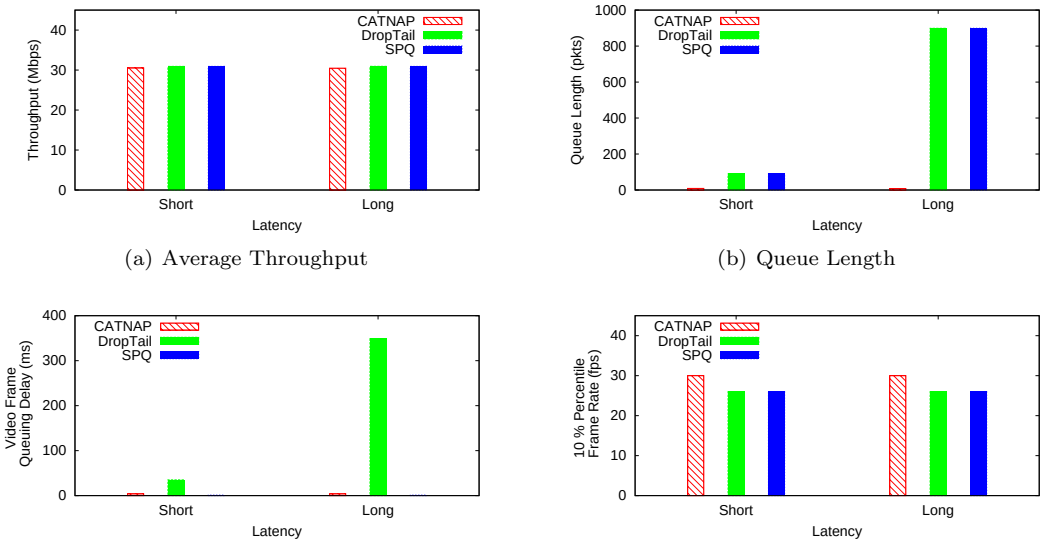
(b) Queue Length

(c) Queuing Delay of Video Application

(d) 10*th* Percentile of Video Frame Rate

Figure 6.30: Video (w/TCP) and FTP over IEEE 802.11b

NFS flow. Its playout frame rate remains at 28-30 fps and experiences almost no queuing delay. Thus, CATNAP performs better than the other two APs when wireless link are under the pressure of greedy UDP flows.



(a) Average Throughput

(b) Queue Length

(c) Queuing Delay of Video Application

(d) 10*th* Percentile of Video Frame Rate

Figure 6.31: Video (w/UDP) and NFS (w/UDP) over IEEE 802.11g

6.3.5 Web Applications as Foreground Applications

CATNAP might classify Web flows as either non-interactive or interactive depending upon the nature of Web flows themselves. A typical Web session starts with the Web client sending a request and ends when the server returns all requested content to the client. Because Web sessions generally only last for a couple seconds, CATNAP has only a small chance to treat Web flows before they terminate. The short duration of Web flows increases the implementation difficulties. However, because of the short duration of typical Web flows, there is no punishment even when CATNAP inaccurately classifies Web flows into a wrong category and mistakenly treats them. Thus, CATNAP should provide similar performance as DropTail, which is the bottom line of treating Web flows.

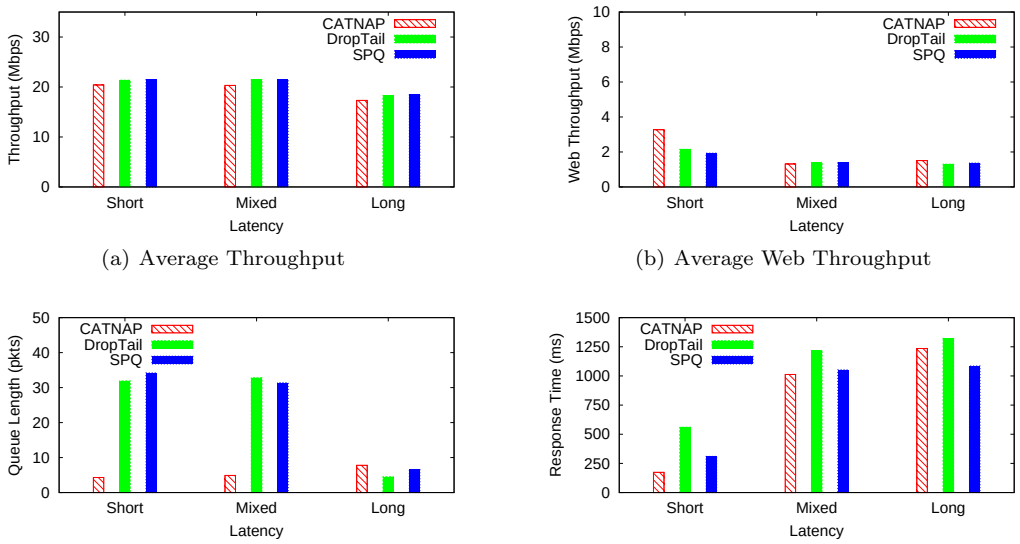
In our simulation or experimental study [35], Web flows are not able to survive with DropTail or SPQ under the pressure of heavy load UDP flows. The Web clients' requests keep timing out because UDP packets fill up the DropTail or SPQ AP's queue. Thus, this study only runs foreground Web flows with background FTP flows.

It is difficult to compare performance of Web flows with other types of foreground flows with the same metric because of the Web flows' short life cycle. It may only complete a couple of Web sessions in 120 second simulation time because our Web traffic model includes users' reading time [9]. Therefore, the Web flow simulation extends the simulation time to 600 seconds: FTP background flows kicks off at the 5th second and ends at the 595th second; the Web client sends its first Web request at the 10th second and stops to send requests after the 590th second. To avoid cold start effects, this study choose the traffic between the 10th second and the 590th second for analysis.

Figure 6.32 presents the result when the foreground Web flows and the background FTP flow are concurrently running on IEEE 802.11g links. As Figure 6.32(a) shows, all three APs provide similar throughput with the three latency configurations. Because of the short duration of Web flows, the background FTP flow contributes the dominant part of the throughput. Figure 6.32(b) shows the throughput for Web flows only. However, because the Web session duration includes the server response time, Web throughput is lower than FTP throughput. CATNAP provides 30% more Web throughput than the DropTail in Figure 6.32(b). Similar to our other simulation results, Figure 6.32(c) shows that CATNAP maintains a smaller queue than DropTail or SPQ.

This study chooses the Web response time as the main performance metric for Web flows. The Web response time is the time period between the time when the Web client sent a request and the

time when the Web client receives last object from the Web server. Figure 6.32(d) shows that both CATNAP and SPQ have a similar and a smaller response time than DropTail.



(a) Average Throughput

(b) Average Web Throughput

(c) Queue Length

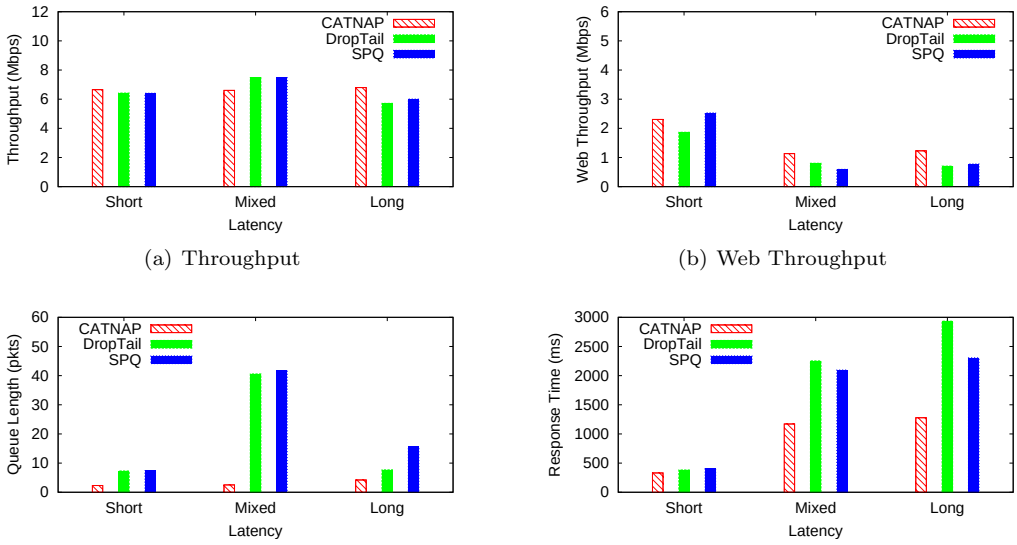
(d) Average Web Response Time

Figure 6.32: Web Flow vs FTP over IEEE 802.11g

Figure 6.33 shows the simulation results when the foreground Web flows and the FTP background flow are concurrently running with IEEE 802.11b links. Over the slower IEEE 802.11b link, CATNAP performs similar as it does with IEEE 802.11g link, specifically, CATNAP provides about a 0.5 seconds smaller Web responses time than DropTail under the long and mixed latency configurations.

6.4 Extended Results

Section 6.3 discusses the simulation results when a foreground flow currently runs with one background flow under three different latency settings. To reduce the number of variables in simulations, the simulation cases in Section 6.3 only has three different latency settings, and two flows (a foreground flow and a background flow). This section demonstrates the performance of CATNAP, DropTail and SPQ under a variety of simulation configurations.



(a) Throughput

(b) Web Throughput

(c) Queue Length

(d) Web Response Time

Figure 6.33: Web Flow vs FTP over IEEE 802.11b

6.4.1 Variety of Latency Settings

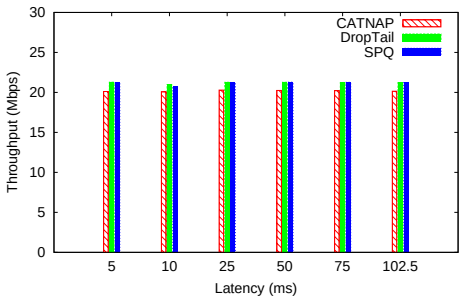
Simulations with three latency settings are not enough to validate the performance of CATNAP, especially for delay sensitive flows such as game flows. The application server involved in residential users' network activities may be all around the world, with end-to-end latencies over a large range. Thus, this section demonstrates how different network latencies affect the performance of CATNAP, DropTail and SPQ.

The foreground flow latency (L_1) varies from 5 ms, 10 ms, 25 ms, 50 ms, 75 ms, and 102.5ms, where 10 ms and 102.5 ms[7] are two modes of one way latencies measured by Maier *et al.* over German residential networks [139]. Because a single FTP flow with a large one-way latency (102.5 ms) is not able to saturate IEEE 802.11 links, the simulations in this section stress the wireless link with a background FTP flow with 10 ms one-way latency. Meanwhile, the simulations in this section only run with IEEE 802.11g since Section 6.3 showed that most of results with IEEE 802.11b links are similar to the simulations with IEEE 802.11g links. Additionally, to avoid repetitive results, we only choose game and video flows as foreground flow since the game flows are sensitive to latencies and video flows usually are not sensitive to latencies because of playout buffers.

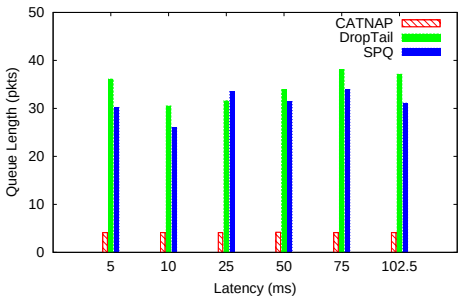
[7] In [139], Maier *at al.* measured round-trip time and the two modes are 20 ms and 205 ms.

Game Flows with Different Latencies

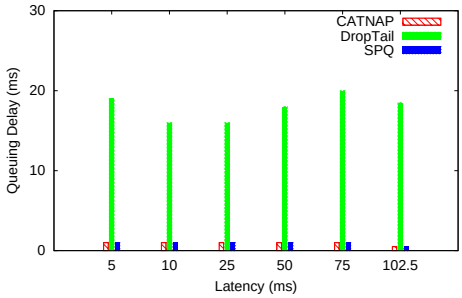
Figure 6.34 shows the simulation results as the latency increases between the game server and client. With the increment of latency of the foreground game flow, all three APs provide similar cumulative throughput of 20 Mbps because the background FTP flow dominates throughput. However, in Figure 6.34(a) the cumulative throughput on CATNAP is 1 Mbps or so less than the other two since CATNAP controls the queue length in a conservative manner. Among the three APs, SPQ introduces the smallest queuing delay because of its strict priority nature. As Figure 6.34(c) shows, the queuing delay of game flows on CATNAP is the same as the game flow on SPQ while DropTail imposes 15-20 ms more delays on game flows. Under all latency settings, SPQ provides the best game quality, CATNAP provides similar quality as SPQ, and DropTail has the lowest game quality even in the simulations with short latency settings. Note, because game quality is sensitive to delay, game flows for all APs drop into fair quality in simulations with large latencies, such as 102.5 ms, although SPQ and CATNAP provide a little bit better game quality than DropTail does.



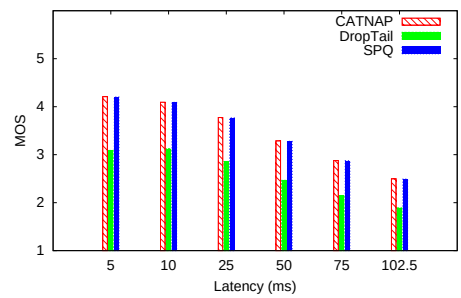
(a) Throughput



(b) Queue Length



(c) Queuing Delay of Game Application

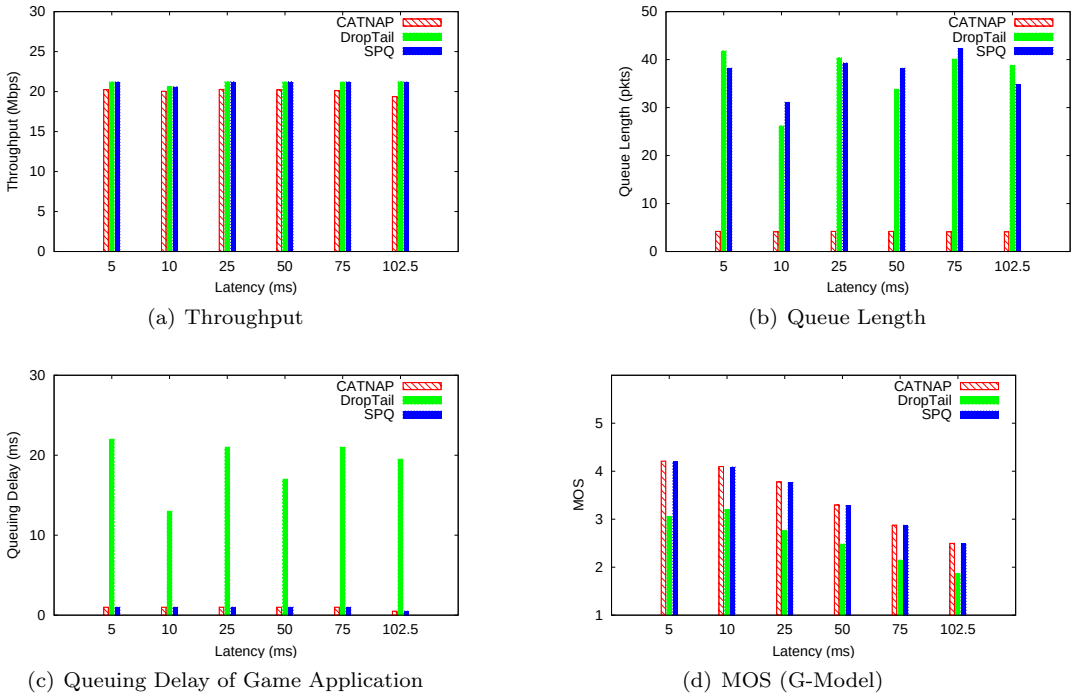


(d) MOS (G-Model)

Figure 6.34: Server Latency Cases (Game w/UDP)

As Figure 6.35 shows, the TCP-based game flows have similar performance to their UDP counterparts. DropTail still provides the poorest game quality in all simulation cases, while SPQ and

CATNAP achieve similar performance.



(a) Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

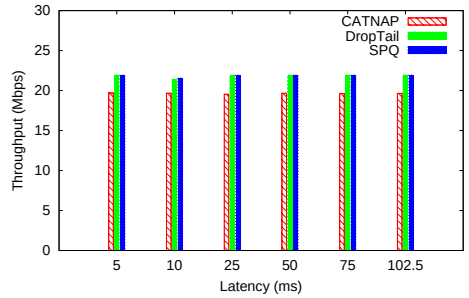
(d) MOS (G-Model)

Figure 6.35: Server Latency Cases (Game w/TCP)

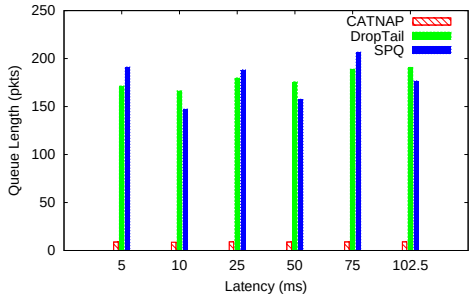
Video Applications with Different Latencies

Figure 6.36 shows the simulation results of foreground UDP video flows under different latency settings. As in the video simulation in Section 6.3, the 10th percentile of frame rate is chosen as the quality metric for end-of-video quality. Figure 6.36(d) shows that all three APs yield a similar video play out frame rate.

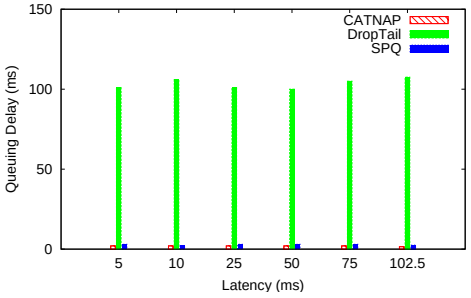
Moreover, Figure 6.36(c) shows that DropTail introduces around 100 ms delay for video flows, while CATNAP and SPQ only introduce 2-3 ms delay. The reason is that the background FTP packets fill up the queue inside DropTail. Although the SPQ queue is also filled with FTP packets, it prioritizes the video flows and does not impair the video performance. Figure 6.36(c) shows that CATNAP has the potential to provide better performance for interactive video applications, such as video conferences, which have less tolerance to large delays.



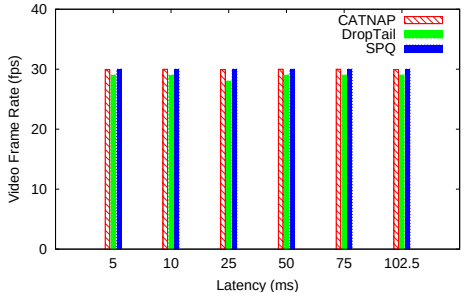
(a) Throughput



(b) Queue Length



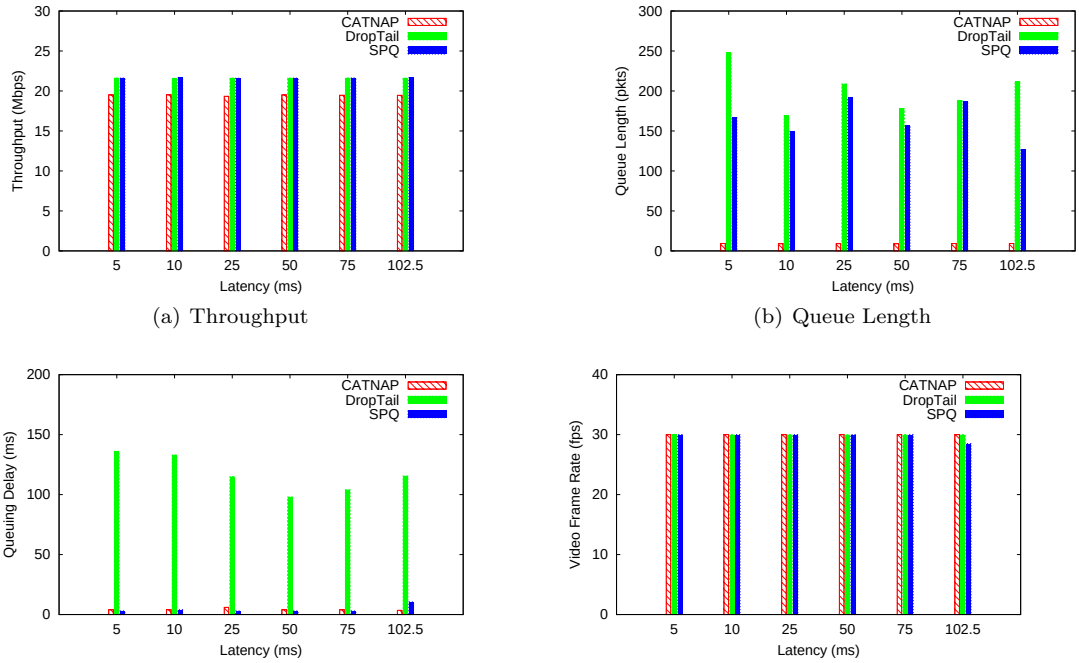
(c) Queuing Delay of Video Application



(d) 10*th* Percentile of Video Frame Rate

Figure 6.36: Server Latency Cases (video w/UDP)

Figure 6.37 shows the simulation results of TCP-based video flows under different latency settings. The results of TCP-based video flows are very similar to the results of UDP-based flows.



(a) Throughput

(b) Queue Length

(c) Queuing Delay of Video Application

(d) $10th$ Percentile of Video Frame Rate

Figure 6.37: Server Latency Cases (video w/TCP)

Summary

This section used two indicator applications to assess CATNAP : games which are sensitive to delay, and videos which are not too sensitive to delay, to evaluate over a large range of latency settings. Since the results are similar to simulation results in Section 6.3, we did not repeat all simulation combinations over different server latency settings.

6.4.2 Range of DropTail Queue Capacities

The queue capacity inside network devices affects the performance of network applications running through them [35,168]. Generally, a large queue capacity is able to assimilate occasionally bursty traffic and improve the TCP throughput, but it may harm the quality of time sensitive applications, e.g., game applications.

Table 6.5 summaries the queue capacities used in in Section 6.3 where DropTail and SPQ choose the queue capacity based on bandwidth delay products, and CATNAP sets its queue capacity as sufficiently large as 1000 packets. However, our earlier studies [35,168] show that the queue capacity in residential APs is between 35 packets to 100 packets, much smaller than the queue capacity used in Section 6.3. Therefore, this section evaluates the performance of DropTail, the default queue controller for residential wireless APs, under a variety of queue capacity settings.

To reduce the number of independent variables, this section uses only the UDP-based game flows as foreground flows, and the FTP flows as background flows because the quality of games is most sensitive to delay. The latency between both clients and servers is set as 10 ms. DropTail queue capacities of are chosen as 35, 50, 100, 200, 400, and 900 packets, where the 35 packet queue is the smallest one found in [35], and the 900 packet queue is the largest one used in Section 6.3.

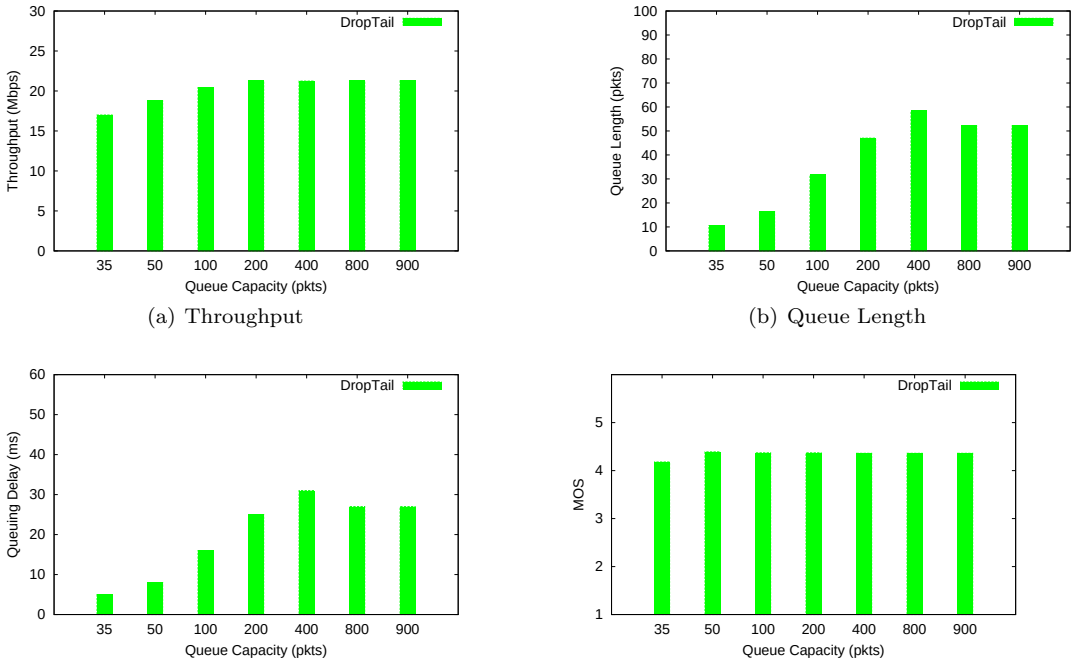
Figure 6.38(a) shows the cumulative throughput on DropTail with different queue capacities. The cumulative throughput increase as the DropTail queue capacity increases. However, after the queue capacity is larger than 200 packets, the cumulative throughput stop increasing. Because the background FTP flow is the dominant part of the throughput, the large queue capacity helps to improve the bursty TCP throughput.

Figure 6.38(b) presents the average queue length [8] during the simulation life cycle. The average queue length also increases as the queue capacity increases. However, after the queue capacity is greater than 400 packets, the average queue length stays around 50 packets.

Figure 6.38(c) shows the queuing delay for the foreground game flow. As the queue capacity and queue length increase, the queuing delay of the game flow also increases. When the AP queue capacity is greater than 200 packets, the queuing delay stays around 20 ms delay because the queue length becomes stable.

The queuing delay is almost two times the link latency between the game client and server, but the MOS value of the game flow does not decrease much. Although the game is sensitive to delay,

[8] Queue length is average queue length over one second.



(a) Throughput

(b) Queue Length

(c) Queuing Delay of Game Application

(d) $10th$ Percentile of MOS Value

Figure 6.38: Game (w/UDP) on DropTail with Different Queue Capacities

30 ms delay (20 ms queuing delay and 10 ms link latency) does not dramatically degrade game performance because of the G-Model [136].

However, Figure 6.21(d) in Section 6.3 shows that CATNAP maintains the MOS value as 4.39 at all three latency settings, which is slightly better than DropTail that range of 4.18 to 4.30. Overall, CATNAP provides similar or better performance with game applications than DropTail, regardless of its queue capacity.

6.4.3 Multiple Foreground Applications

To control the number of random variables, the simulation in the previous section only involves two flows: one background flow and one foreground flow. Thus, this section compares the performance between CATNAP and DropTail with multiple flows. Note, this section excludes SPQ, because SPQ is a static priority scheme and without any common acceptable standard for flow priorities.

Setup

Palazzi *et al.* have the assumption that the typical American middle class family consists of two working parents and two teenage kids [105,126,169][9], and four computers are concurrently running. Thus, the simulation setup in this section is: one AP, one gateway, four wireless clients, and four Internet servers.

One wireless node serves as the destination for the background FTP flow. The one way latency between the FTP client and server is 10 ms, which is half of the shorter mode of RTTs which is measured over German residential networks [139][10]. This simulation uses the FTP background flow to stress the IEEE 802.11g links.

The remaining three wireless nodes are used as the destination for the UDP-based game, UDP-based VoIP and TCP-based video flows, respectively. CATNAP classifies the game and VoIP flows as non-response-based, interactive, and non-greedy, and the video flow as response-based, non-interactive, and non-greedy. To reduce the number of simulation variables, the foreground flows are set with the same one-way latency as 50 ms.

The background FTP flow starts the 5th second of the simulation and ends at the 115th second. The VoIP, game and Video Flow start at the 9.7th seconds with 0.3 seconds interval, and they terminate at 110th second. To avoid the effect of cold start, the results are calculated with the simulation data between the 15th second and 105th seconds to avoid the cold start and cool down effects.

The queue capacity of DropTail is calculated based on bandwidth delay products: 50 packets (small) and 450 packets (large) respectively. The queue capacity of CATNAP is 1000 packets, as same as the CATNAP queue capacity in Section 6.3.

Refined Estimated Link Capacity Algorithm

In Section 6.2, CATNAP provides lower cumulative throughput than DropTail, especially when interactive flows exist. One possible reason is that CATNAP underestimates the effective throughput, over-treating the non-interactive flows. After revisiting the CATNAP effective capacity estimator (Algorithm 1), we decided to refine Algorithm 1 to improve the performance of CATNAP.

[9] According to the 2010 Census data, the average household size is 2.63, slightly growing from 2.62 in the 2007 Census.

[10] Maier *et al.* report 20 ms and 205 ms as two modes of RTTs over Germany residential network are 20 ms and 205 ms respectively.

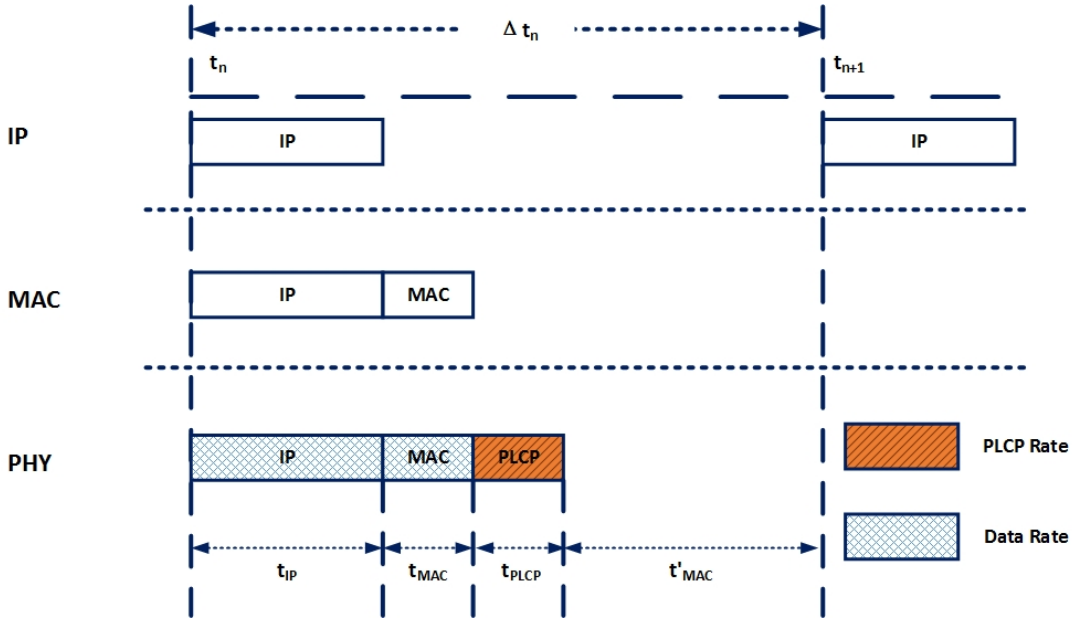


Figure 6.39: IP Packet Transmission

Figure 6.39 gives the detailed timing information of an IP packet transmission process inside a wireless driver, where t_{PLCP} is the transmission time for the PLCP header; t_{MAC} is the transmission time for the MAC layer header including CRC; t_{IP} is the transmission time for the IP packet excluding the MAC and PLCP headers; and t'_{MAC} is the overhead to receive MAC layer ACK frames and possible time used for MAC layer retransmission. Note, the actual sending time (Δt_n) for packet p_n measured in Algorithm 1 is $\Delta t_n = t_{n+1} - t_n$. Algorithm 1 calculates the effective capacity C as:

$$\begin{aligned} C &= \frac{lengthp(n)+H_{MAC}}{t_m+t+t'_m} \\ &= \frac{lengthp(n)+H_{MAC}}{\Delta t_n - t_{PLCP}} \\ &\approx \frac{L_{IP}}{\Delta t_n - t_{PLCP}} && \text{, when } L_{IP} >> L_{MAC} \\ &= \frac{L_{IP}}{\Delta t_n} && \text{, used in effective capacity estimator (Algorithm 1)} \end{aligned} \tag{6.2}$$

Comparing Figure 6.39 and Eq 6.2, one can see that Algorithm 1 works well when the IP packet size $length(p_n)$ is much larger than the MAC layer header length. For example, compared to a 1500 byte long FTP packet, the 34 byte MAC layer header is negligible. But for the 80 byte long Quake IV game packets, the MAC layer header is significant when calculating the effective capacity. Thus,

Algorithm 13 simply improves the CATNAP effective capacity estimator by considering the length of MAC header and PLCP overhead.

Algorithm 13 Estimate Link Effective Capacity (refined)

At the end of each epoch,

1: $H_{MAC} \leftarrow 34$ bytes ▷ MAC Layer header including CRC.
2: $t_{PLCP} \leftarrow 48$ ms ▷ Transmission time used for PLCP header
3: $n \leftarrow get\_total\_pkt\_sent()$ ▷ get total number of packet sent in this epoch.
4: **if** $n > 0$ **then**
5: $C_{epoch} \leftarrow \frac{\sum_{k=1}^{n}(get\_pkt\_length(p_k)+H_{MAC})}{\sum_{k=1}^{n}(\Delta t_k} - t_{PLCP})$ ▷ Instantaneous Capacity in i_{th} epoch
6: $C \leftarrow (1-\omega) \times get\_link\_capacity() + \omega \times C_{epoch}$
7: **else**
8: $C \leftarrow get\_link\_capacity()$ ▷ No packet was transmitted in i_{th} epoch, C_i is same as C_{i-1}.
9: **end if**
10: $set\_link\_capacity(C)$ ▷ Save the link capacity into the CATNAP system variable tab.

According to the IEEE 802.11 standard [2, 141], the physical layer header is sent at a lower rate (the preamble rate) where the MAC header and MAC payload are transmitted at the higher IEEE 802.11 rate. Based on IEEE 802.11g standard [2], the long and short PLCP preamble and the long PLCP header are transmitted at 1 Mbps, and the short PLCP header is be transmitted at 2 Mbps. However, the NS-2 implementation only supports one PLCP data rate which is used to transmit both the PLCP preamble and the PLCP header. Thus, in this study, we use our NS-2 simulator with the PLCP Header length as a 48 bit (short preamble length) and PLCP rate as 1 Mbps respectively.[11] Therefore, the PLCP overhead per frame is calculated in Equation 6.3

$$t_{PLCP} = \frac{H_{PLCP}}{R_{PLCP}} = \frac{48\ bits}{1\ Mbps} = 48\mu s \tag{6.3}$$

Results of Multiple Foreground Applications

Table 6.7 and Table 6.8 summarize the results when three foreground flows and one background flow are concurrently running with DropTail and CATNAP. We have several key observations from these two tables:

[11] The NS-2 implementation is inconsistent with the IEEE 802.11 standard. In addition to the possible 1 Mbps or 2 Mbps PLCP rate listed above, some researchers argue that the PLCP data rate should be set as 6 Mbps - the base date rate for the IEEE 802.11g network.

Table 6.7: Throughput Comparison between Bandwidth Estimators

Application	Throughput (Mbps)			
	DropTail		CATNAP	
	50 pkt queue	450 pkt queue	Orig Estimator	Refined Estimator
Background FTP	17.26	19.72	16.58	18.48
VoIP w/UDP	0.080	0.08	0.0800	0.08
Game w/UDP	0.01	0.01	0.01	0.01
Video w/TCP	1.66	1.65	1.66	1.66
Total	19.00	21.46	18.23	20.34

Table 6.8: QoS comparison between Bandwidth Estimators

Application	Quality			
	DropTail		CATNAP	
	50 pkt queue	450 pkt queue	Orig Estimator	Refined Estimator
Background FTP(Mbps)	17.26	19.72	16.58	18.48
VoIP w/UDP (MOS)	4.36	4.03	4.37	4.37
Game w/UDP (MOS)	2.52	1.32	3.28	3.28
Video w/TCP (fps)	16.00	29.00	30.00	30.00

1. CATNAP provides better QoS for interactive applications, especially for game applications which are more sensitive to delay. From the view of interactive applications' quality, CATNAP with either the original or refined bandwidth estimator performs better than DropTail.

2. CATNAP with refined bandwidth estimator provides higher cumulative throughput than CATNAP with the original estimator. Both implementations of CATNAP have lower cumulative throughput than DropTail with a large queue, but CATNAP with refined capacity estimator provides better cumulative throughput than DropTail with small queue capacity.

3. The queue capacity of DropTail significantly affects its performance. The bigger queue capacity ensures the bulk file transfer applications achieve higher throughput, while the smaller queue can provide better performance for interactive applications, such as Game and VoIP. However, it is not easy to decide the "best" queue capacity for DropTail.

4. CATNAP does not fully take advantage of a large queuing buffer space even with the refined bandwidth estimator. The cumulative throughput of DropTail with 450 packets is 1.1 Mbps (5%) higher than CATNAP with the refined estimator, and 3.2 Mbps higher than CATNAP with the original estimator. Thus, there are still a possibility to improve the throughput of

CATNAP, but would require a more accurate bandwidth estimator. We leave this as future work.

6.5 Summary

This section summarizes the simulation results Section 6.3 and Section 6.4, and provides a general overview of performance comparison between CATNAP and SPQ. SPQ provides a performance ceiling for an AP over a residential network. In reality, however, it is impractical for SPQ to reach its best performance without having apriori knowledge on each flow's characteristics. Thus, this summary section uses the result of DropTail as a baseline to calculate the improvement from CATNAP.

CATNAP always improves the foreground application quality in the presence of a high volume UDP flow. A TCP-based flow will be terminated when a high volume UDP flow stresses the wireless link [17], and DropTail yields extremely low performance. Although DropTail provides higher downlink throughput than CATNAP for some simulation cases, the quality of foreground applications is always poor with DropTail because the foreground application cannot survive when competing with high volume UDP flows. Since DropTail performs so poorly when a high volume UDP flow is present, this section only focuses on the simulation results with FTP flows as a background application and excludes results involving the UDP-based flows in the summary section. Interested readers can refer to Table A.1 to Table A.4 in Appendix for further information.

Table 6.9 and Table 6.10 summarize the quality (QoS) improvement of foreground applications, and Table 6.11 and Table 6.12 compare throughput differences between CATNAP and DropTail over high capacity links (11g) and low capacity links (11b) respectively. Appendix lists the complete tables with numerical values. The positive ratio means that CATNAP performs better than DropTail, while the negative number means that CATNAP performs worse than DropTail. Note, bold fonts highlight the results when CATNAP differs by 10% more than DropTail. Several key observations are made from these four tables:

1. CATNAP improves the quality (QoS) of foreground applications on both high capacity (IEEE 802.11g) links and low capacity (IEEE 802.11b) links, especially for interactive applications with tight latency requirements such as First Person Shooter (FPS) games [162,170]. However, CATNAP does not significantly improve the quality of VoIP applications over IEEE 802.11g

Table 6.9: QoS improvement for CATNAP over DropTail on IEEE 802.11g Links.

Background	Foreground	Quality	Latency Settings		
			short	*mixed*	*long*
FTP	FTP	Throughput	1.3%	**104.4%**	**18.4%**
		Jain's Fairness	1.0%	**23.8%**	6.4%
	Game w/UDP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	**58.1%**	**52.4%**	**15.7%**
	Game w/TCP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	**54.5%**	**55.6%**	**53.7%**
	VoIP w/UDP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	1.1%	1.8%	1.6%
	VoIP w/TCP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	0.2%	0.6%	0.9%
	Video w/UDP	Throughput	0.0%	0.0%	0.0%
		Fr Rate(10% tail)	0.0%	0.0%	0.0%
		Video Fr Delay	**89.5%**	**97.6%**	**75.0%**
	Video w/TCP	Throughput	0.0%	0.0%	0.0%
		Fr Rate(10% tail)	3.4%	3.4%	0.0%
		Video Fr Delay	**80.9%**	**95.6%**	**77.8%**
	Web	Throughput	**23.5%**	**41.3%**	**75.7%**
		Response Time (sec)	**13.6%**	**48.1%**	**56.4%**

Table 6.10: QoS improvement for CATNAP over DropTail on IEEE 802.11b Links.

Background	Foreground	Quality	Latency Settings		
			short	*mixed*	*long*
FTP	FTP	Throughput	-8.8%	**46.5%**	**106.4%**
		Jain's Fairness	4.2%	**25.0%**	7.6%
	Game w/UDP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	**32.6%**	**113.9%**	**93.7%**
	Game w/TCP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	**34.7%**	**117.7%**	**111.2%**
	VoIP w/UDP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	5.3%	**25.4%**	**11.6%**
	VoIP w/TCP	Throughput	0.0%	0.0%	0.0%
		MOS (10% tail)	5.2%	**29.6%**	**12.7%**
	Video w/UDP	Throughput	1.4%	-1.4%	0.7%
		Fr Rate(10% tail)	**11.5%**	7.4%	**11.1%**
		Video Fr Delay	**50.0%**	**96.9%**	**96.4%**
	Video w/TCP	Throughput	0.0%	0.0%	0.0%
		Fr Rate(10% tail)	7.4%	**20.0%**	3.4%
		Video Fr Delay	**14.3%**	**93.8%**	**78.7%**
	Web	Throughput	**75.8%**	**63.8%**	**115.7%**
		Response Time (sec)	**54.6%**	**55.0%**	**57.9%**

Table 6.11: Cumulative Throughput Improvement for CATNAP over DropTail on IEEE 802.11g Links.

Background	Foreground	Improvement		
		short	*mixed*	*long*
FTP	FTP	-1.7%	2.0%	-2.7%
	Game w/UDP	-4.2%	-5.1%	-2.1%
	Game w/TCP	-2.8%	-6.0%	-2.1%
	VoIP w/UDP	-1.5%	-5.6%	0.7%
	VoIP w/TCP	-2.1%	-5.0%	2.5%
	Video w/UDP	-5.9%	**-12.3%**	-2.2%
	Video w/TCP	-6.8%	**-13.1%**	-7.3%
	Web	-4.4%	-5.4%	-5.1%

Table 6.12: Cumulative Throughput Improvement for CATNAP over DropTail on IEEE 802.11b Links.

Background	Foreground	Improvement		
		short	*mixed*	*long*
FTP	FTP	-4.0%	3.1%	1.0%
	Game w/UDP	9.6%	-9.6%	**-16.2%**
	Game w/TCP	**10.1%**	**-10.9%**	-0.6%
	VoIP w/UDP	6.1%	-5.4%	-5.9%
	VoIP w/TCP	8.5%	**-10.8%**	-9.9%
	Video w/UDP	6.5%	**-13.2%**	**-10.6%**
	Video w/TCP	8.9%	**-10.8%**	**12.0%**
	Web	3.7%	**-11.6%**	**13.1%**

links, because the quality of VoIP degrades only when one way delay is greater than 173ms. The average playout frame rate is also not improved much by CATNAP with high capacity IEEE 802.11g links, but the video frame queuing delay is significantly reduced. However, because VoIP is more sensitive to bursty loss and its quality is not sensitive to delay relative to typical AP queue sizes, the quality of VoIP is not improved much by CATNAP.

2. CATNAP generally provides a slightly lower cumulative downlink throughput than DropTail. Primarily, throughput is not the objective of CATNAP. Moreover, the original downlink capacity estimator, described in Algorithm 1, excludes PLCP headers and MAC layer headers when estimating downlink available capacity. The PLCP or MAC layer overhead is insignificant compared to the transmission times for "full" packets, but these PLCP or MAC layer overhead should not be excluded for "non-full" packets, especially for tiny packets from interactive flows, such as VoIP or game flows. Therefore, Algorithm 13 amends the downlink

capacity calculation by subtracting the PLCP and MAC layer overhead, the corresponding result are listed in Table 6.7.

3. CATNAP reduces the queuing delay for game, VoIP and video applications for most simulation cases. Video applications with CATNAP and DropTail yield similar average frame rate over either high link capacity (11g) or low link capacity (11b), but CATNAP introduces smaller overhead for video frame queuing delays than DropTail.

Chapter 7

Future Work

This chapter considers several possible areas of future work based on this thesis.

7.1 Implementation

Chapter 6 shows the evaluation and validation of CATNAP with the NS-2.33 network simulator. One important reason that we implement CATNAP with NS-2 instead of an actual AP is that network simulators are able to evaluate CATNAP under various network conditions. Implementing CATNAP with the Open Source tools, e.g. Openwrt [56] could provide realistic results, but evaluating AP performance could not be done as thoroughly as in simulation. As Chapter 6 discussed, it is difficult to analyze wireless traffic collected from uncontrolled environments because many factors degrade the performance of wireless APs.

Initially, we tried to implement CATNAP as a host AP [168] based on a Intel P4 desktop in 2010. OpenWrt [56] is an alternative development tool, but OpenWrt only supported LinkSys WRT54g router. However, the Linux based host AP provides more development flexibility than OpenWrt. The per-flow queue, response-base/non-response-based classifier, and interactive/non-interactive classifier were functional as a prototype. However, in 2010, the Linux kernel did not support wireless IEEE 802.11g card very well. We tried several wireless cards from different vendors including Netgear, Belkin, Trendnet and Lucent, and encountered various technical difficulties. For example, the greedy/non-greedy classifier needs information on link capacity, but most driver APIs only can provide current physical link speeds. We enhanced the Madwifi driver to collect wireless

layer and physical layer information, but the Madwifi driver became unstable with heavy loads.

CATNAP could be implemented with OpenWrt or other wireless network development kit. In detail, the implementation of a prototype CATNAP consists of several steps:

- **Implement CATNAP with a virtual machine:** Directly implementing CATNAP with OpenWrt might be difficult because CATNAP is a cross layer solution. CATNAP needs modifications with wireless layer drivers as well as IP and transport layers implementation. However, several CATNAP modules can be implemented inside IP and Transport layer. These IP and transport layer modules can be implemented on virtual machines to save debug time.
- **Implement CATNAP with a Host AP:** Host AP [168] is a Linux based desktop with wireless network card, acting as a wireless access point. The Host AP provides more computational power, memory resources and debugging compatibility. Implementing CATNAP on a host AP could be helpful to tune the parameters of CATNAP, reduce its resource usage, and improve its performance.
- **Implement CATNAP with OpenWrt:** The OpenWrt tool kit supports a variety of hardware [56]. Several OpenWrt compatible devices, e.g., SMC 7908ISP-A and Freescale MPC85xx (p1020wlan), are based on multi-core CPUs with similar architectures to Intel X86. These APs can be the first candidates when porting CATNAP from a Host AP.

CATNAP is a cross layer solution, and the majority of its modules can be implemented within the IP and Transport layer. Theoretically, CATNAP can be built upon any hardware supported by OpenWrt. Only the downlink capacity estimator must be integrated within the MAC or physical layer, and it depends on the hardware. The "push" and "delay" can be implemented with IEEE 802.11e standard by simply mapping the CATNAP classification results into TOS bits or IEEE 802.11e traffic categories (TC).

7.2 Resouce Consumption

CATNAP could be implemented in commerically sold APs, but consideration would need to be made for the hardware resources required. Wireless APs are resource limited devices. For example, among 200 APs supported by OpenWrt, only six of them are built with dual core CPU and only few of them have 1G RAM [56].

Although CATNAP is designed for resource limited devices, it still introduces memory and CPU overheads. CATNAP requires additional $O(N)$ memory, where N is number of current flows, to keep per flow statistics information for each active flow, such as packet rate, RTT, classification results, EWMA packet length, and start and last active timestamp. However, the per flow information is small, and 1 KB space is sufficient to store one flow record. Meanwhile, unlike core network routers, residential wireless APs do not need to handle thousands of concurrent flows. Our home wireless measurement study only observed 10-20 concurrent flows occasionally. Thus, in real implementation, CATNAP only needs up to 50 KB additional memory space to keep per flow information. In addition to per flow information, CATNAP tracks system variables, such as total number of active flows, downlink capacity, and cumulative packet arriving rate, but these system wide information are independent with the number of flows and the number of packets, and requires 1 KB additional memory.

The CATNAP modules are divided into two categories from the implementation perspective: packet-based or epoch-based. The computational complexity of packet-based modules such as the interactive/non-interactive classifier is $O(M)$ where M is the number of packets per seconds; the computational complexity of epoch-based modules such as the greedy/non-greedy classifier is $O(N)$ where N is the number of flows per epoch. Our NS-2 implementation integrates the packet-based modules into the uplink and downlink queue controller and triggers the epoch-based modules with a timer (soft interrupt). An AP implementation could use a similar approach. Computation-heavy operations such as recalculating CRC after modifying the advertised window size for each TCP packet can be done with hardware to save CPU cycles.

However, the challenging part is to estimate the CPU consumption under a heavy loaded system.

7.3 Identify Paced Traffic

In addition to response-based/non-response-based, interactive/non-interactive, and greedy/non-greedy flows, packet inter-arrival timing information can be also used to identify the type of applications [22, 30]. Packets can be transmitted in bursts on a paced basis or an as-available basis [30]. While the former pattern often indicates both throughput and response-oriented applications, the latter pattern is indicative of applications that require a steady data rate to limit jitter, e.g., video streaming and VoIP. We assume that packet inter-arrival time is usually dominated by the ap-

plications themselves, but it can also be affected by the wireless network conditions. Congestion and queuing on the devices along the path from servers to the residential clients are potential impediments to correct determination of this classification, but the long-term nature of the timing information of these applications makes classification feasible.

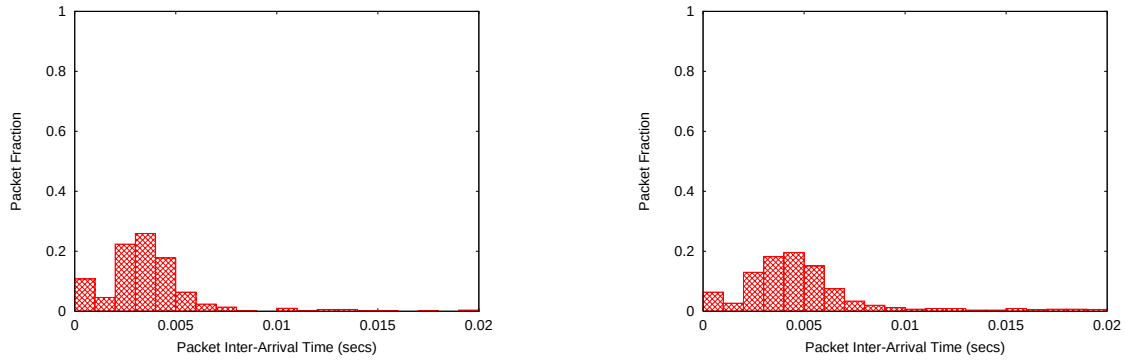
Paced flows should be treated with smooth method: sometime push and sometime delay. However, detecting response-based traffic or inspecting packet length is not effective in differentiating video streaming flows from other flows. High quality video can be transferred over TCP with full packets as in FTP, and our current implementation of CATNAP might classified high quality video flows as non-interactive and/or greedy. However, high quality video flows are different from FTP flows in the nature of packet inter-arrival timing information.

For the file transfer, the limiting factor is typically the TCP congestion control mechanisms. It is well known that TCP congestion control, combined with ACK compression, induces burstiness in traffic patterns. However, streaming traffic typically involves continuous data transfer over a long time period[1], and its data rate is determined by the video or audio codec. Thus, streaming traffic has a more regular traffic pattern and transmission rate than does file transfer. Hence, streaming traffic is expected to be less bursty in terms of packet inter-arrival time than bulk downloading data[2].

Roughan *et al.* [22] propose a method to differentiate streaming traffic by studying the ratio of mean and standard deviation of inter-arrival times, referred to as the inter-arrival variability metric. Their result shows the inter-arrival variability appears to be a good method for differentiating data transfer from streaming. However, the bursty nature of the wireless medium introduces additional variance into the packet inter-arrival time. Therefore, one potential improvement is to develop a new flow classification metric based on the various moments of inter-arrival times.

Figure 7.1 compares packet inter-arrival time between a streaming flow over UDP (the data flows from server to client)[3] and an FTP download (the data flow from server to client) over a wireless link in our preliminary study. The downstream packets are captured at the wireless client instead of at an AP over a residential network. In Figure 7.1, bin size is selected as 0.001 second. Louma *et al.* [171] use a similar approach, with different bin size, to identify real time video traffic from FTP traffic. However, as Figure 7.1 shows, the distribution of packet inter-arrival times of

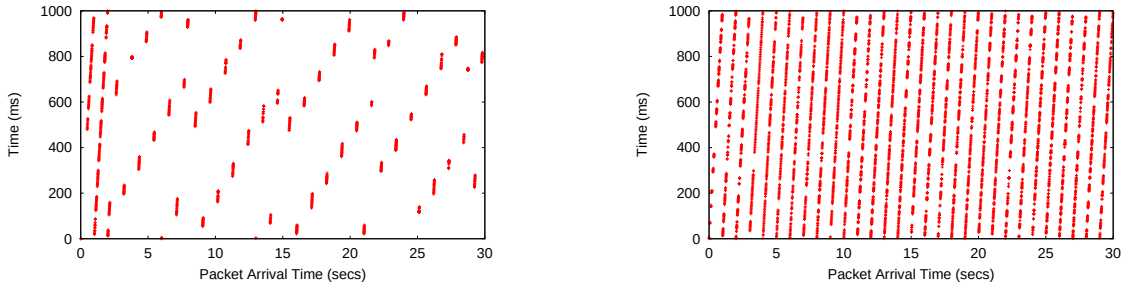
[1] The median video clip length on the Internet is 2 minutes [38].
[2] A VBR codec may be bursty because of the nature of information content, but this burstiness appears in the packet size, not in the packet inter-arrival times [22].
[3] The video used in our preliminary study is encoded at 300 Kbps by Windows Media Encoder v9.0.



(a) Streaming Application (b) FTP

Figure 7.1: Packet Inter-Arrival Time Distribution of a Streaming and an FTP Application.

these two different applications are difficult to differentiate.



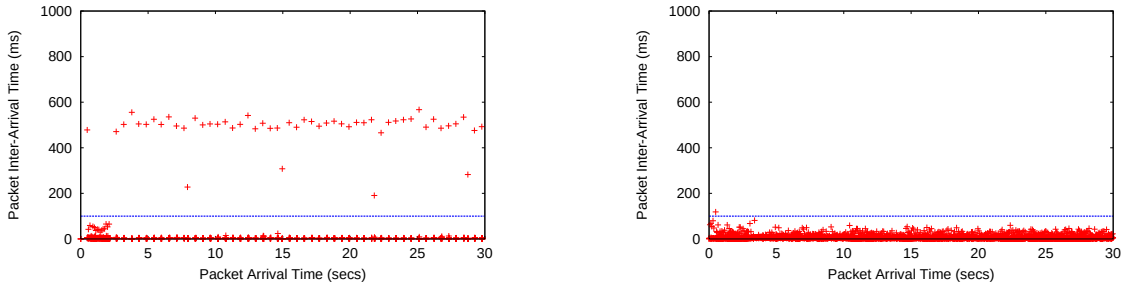
(a) Streaming Application (b) FTP

Figure 7.2: Two Example Dot-Strip Plots for the Packet Inter-Arrival Time for a Windows Media Streaming with UDP Flow and an FTP Data Flow.

Figure 7.2 shows the same dataset in dot-strip plot graphs. The plot shows the arrival time (in seconds) of a packet along the x-axis, and to make the timings of individual packets more obvious, displays the milliseconds along the y-axis. The major advantage of dot-strip plot graph is that the patterns from the packet inter-arrival time plots can be visually identified by human eyes. In Figure 7.2 (a), after the initial buffering phase, the packets in the streaming application are transferred in paced groups, while the FTP flow (in Figure 7.2 (b)) does not show similar characteristics. To study the gap pattern between paced packet groups in a streaming flow, a threshold is introduced to quantitatively analyze the gaps between packet groups.

Figure 7.3 shows the packet inter-arrival time on the y-axis, and packet arrival time along the x-axis for the same data set with a threshold (the horizontal line in both graphs). The packets above the horizontal line arrive 100 milliseconds or more after its previous packet. The preliminary

threshold in Figure 7.3 is arbitrarily selected, but one could study the sensitivity of classification to the threshold selection.



(a) Streaming Application (b) FTP

Figure 7.3: Two Examples of Packet Timing Information.

In addition to the gap between packet groups, other timing information such as a moving average of packet inter-arrival time can be used to differentiate paced and as-available traffic [22]. Thus, CATNAP could use a statistical-based method to differentiate paced traffic (such as streaming video and audio) from other best effort flows and attempt to reduce the jitter of paced flows when congestion occurs.

7.3.1 Smooth the Flow

Packets in flows that need a consistent rate with little jitter can be smoothed (pushed or delayed) to maintain a consistent rate as the network load varies. This treatment is appropriate for streaming video (VoD) and audio (VoIP) applications that need bandwidth estimation and low delay jitter.

Smoothing the flow is the most complex of the four treatments. Smoothing can be done with both push and/or delay operations or separate queues. For instance, using a separate queue to support video streaming applications, packets from the streaming server can be placed in the queue (buffer), and the wireless AP dequeues and transmits packets from this queue periodically. This method minimizes jitter on the wireless client and improves quality.

The smoothing operation might require additional information to decide the drain rate for streaming applications. When a flow is categorized as a paced flow[4], the CATNAP classifier already knows the arrival rate before placing packets into a separate queue. Thus, the AP can buffer the incoming traffic and periodically dequeue the packets at the same rate as the arrival rate. Between

[4]It is typically a streaming audio or video flow.

the transmission of paced packets, the non-paced flows such as file downloading still have a chance to be transmitted.

7.4 Improve RTT Estimator

Simulation results show that CATNAP treats the FTP flows more fairly than DropTail because CATNAP sets the TCP advertised windows size with estimated fair share rate. Because the advertised window size calculation is based on the RTT estimation, CATNAP could perform better if we can improve the current RTT estimator.

Algorithm 2 measures the round time time (RTT) for TCP flows based on the TCP SYN/ACK three-way handshake - it simply sets the RTT of UDP flows on the average RTT for Internet flows reported in [153]. Thus, the RTT of TCP flows is only measured once when the TCP connection is established. After that, even when RTT changes due to queuing delay variations or routing changes, the CATNAP treater still calculates the advertised window size based on the obsolete RTT measured at the first time.

Therefore, CATNAP might have a better performance if we can improve the Algorithm 2 by measuring RTT during their entire lifetime of TCP streams. However, passive estimation of RTT at the middle observation point is not a trivial problem [172, 173]. In addition to the SYN/ACK estimation, Jiang and Dovrolis [173] propose a *Slow-Start* algorithm which is based on the slow-start phase of TCP. The Slow-Start algorithm only works with TCP flow starts with at least five consecutive segments, and the first four of them are MSS packets. Thus, their method does not work with TCP flows with small segments, or TCP flows with unusual MSS settings. Jaiswal *et al.* [172] and But *et al.* [174] present algorithms to continuously estimate RTT for TCP flows. But their algorithms are inaccurate under certain conditions [174] though they can be used to predict an upper bound of the actual RTT.

On the other hand, it is even more difficult to estimate RTT for non-response-based or UDP flows at the mid-point, especially for UDP video streaming flows, because there might not exist reverse flows for them. However, we might use some other information to estimate the RTT of non-response-based flows. For example, RTSP [94] specifies the video stream can be sent through either UDP or TCP, but the control signaling are always transmitted over a TCP connection. In this case, we might estimate the RTT of UDP flows by measuring paired TCP flows.

7.5 Self-Tuning Epoch

The value of the epoch used in Algorithm 1 and Algorithm 8 is empirically decided by a set of simulations. Ideally, the size of the epoch should be half of the shortest RTT observed by CATNAP. However, because our current implementation of RTT estimator only estimates once when the TCP connection is established and the one time measured RTT might be inaccurate, we decide to choose a fixed epoch in order to reduce the implementation complexity. A too short epoch time would introduce too much computational overhead for flows with long RTTs, while a too large epoch time would not be able to detect the fluctuation of flow rate promptly. Thus, it is promising to improve the performance of CATNAP by introducing a scheme to dynamically choose the value of epoch based on the shortest RTT observed.

7.6 Identify Interactive flows with TCP PUSH flag

Some other packet level information can help to more accurately identify the type of applications. For example, according to different application types, the client may have a different number of packets flagged with the PUSH flag. By using the PUSH flag, the TCP sender tells the operating system and the receiving peer that all buffered data needs to be sent to the receiving applications. In interactive applications, when the client sends a command to the server, the client sets the PUSH flag and waits for the server's response. Without the PUSH flag, this process would be delayed by the operating system on the receiver which may continue to wait for additional data. Thus, interactive applications tend to have a larger fraction of non-full packets with the PUSH flag.

Chapter 8

Conclusions

As the last-mile link speed is increasing and wireless devices become widely deployed, wireless access points are becoming the bottleneck device along the Internet path in near future. Improving the quality of realtime applications, such as network games and VoIP over residential wireless networks is becoming increasingly important. A wireless access point acts as the central communication point for a residential network, connecting all network devices inside home through a shared Internet connection. However, compared with enterprise class APs, residential APs are usually resource limited: low CPU computation capabilities and small memory size.

This dissertation presents a cross-layer solution, Classification And Treatment iN Access Point (CATNAP), which can be implemented over residential access points to 1) provide an automatic flow classification scheme without any manual user interventions; 2) minimize queuing delays to meet quality of service (QoS) requirements of delay sensitive applications such as Web, VoIP, game and streaming video. CATNAP is an active queue management approach which is designed for low end devices, such as residential APs: both classification and treatment modules consume small amount of resources. CATNAP classifies flows into treatment-based categories without any manual user configurations or training with historic datasets.

CATNAP meets a few requirements for "smart" devices. Its pre-configured parameters remove the burden from home users to configure their APs. Typically, home users are not required to know any detailed information of their applications, such as which ports are used. Meanwhile, CATNAP is an approach that does not require any modification to end hosts, and it is compatible with existing TCP/IP based network applications. Moreover, because CATNAP is independent

from any particular application, it is able to classify and treat any future applications which behave similarly to applications in the eight categories.

A thorough simulation study demonstrates that CATNAP is able to improve multimedia application quality with a wide range of network configurations, especially applications such as games and VoIP whose quality is sensitive to delay. In most simulation cases, CATNAP is able to provide better performance than DropTail and similar performance to SPQ which needs prior knowledge on flow types. Especially for simulation cases where a non-response-based, greedy flow is a background flow, CATNAP provides even better performance than SPQ. Web flows cross the boundary between the interactive and non-interactive flows. There is only small opportunity to treat Web flows because of their short life cycle. However, CATNAP can improve the Web quality with smaller response time.

This dissertation makes an effort to investigate current wireless network usage at home. By analyzing traces gathered during the home wireless measurement study, we provide a better understanding of home wireless network usage, with a set of reality-based parameters to tune NS-2 simulation.

This study shows that

- CATNAP is a "plug-and-play" solution, which automatically classifies and treats home applications, without any modification to end devices or applications.
- CATNAP generally provide better QoS than DropTail. CATNAP lowers the queuing delay for time sensitive applications by effectively controlling queue length. Results show that CATNAP improves QoS for interactive applications, VoIP and games, as well as reduces the response time for Web applications.
- Results show that CATNAP tames greedy traffics efficiently, especially for misbehaved high volume UDP flows. Meanwhile, CATNAP treats greedy TCP flows more fairly than DropTail.

Bibliography

[1] Akamai Technologies, "The State of the Internet, 3rd quarter, 2013 Report," Tech. Rep. 3, Akami Technologies, Inc, 2013.

[2] IEEE Computer Society LAN MAN Standard Committee, "IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," August 1999.

[3] Pablo Brenner, "A Technical Tutorial on the IEEE 802.11 Protocol," Tech. Rep., BreezeCom Wireless Communications, 1992.

[4] Andrew Tanenbaum, *Computer Networks: 4th Edition*, Prentice-Hall, Inc., Upper Saddle River, NJ, 2002.

[5] Qiang Ni, Lamia Romdhani, and Thierry Turletti, "A Survey of QoS Enhancements for IEEE 802.11 Wireless LAN: Research Articles," *Wireless Communincations & Mobile Computing*, vol. 4, no. 5, pp. 547–566, 2004.

[6] Wi-Fi Alliance, "Wi-Fi Certified for WMM - Support for Multimedia Applications with Quality of Service in Wi-Fi Networks," Tech. Rep., Wi-Fi Alliance, September 2004.

[7] Andrew Moore and Konstantina Papagiannaki, "Toward the Accurate Identification of Network Applications," in *Proceedings of the 6th Passive and Active Measurement Workshop (PAM)*, Boston, MA, March/April 2005.

[8] Choong-Soo Lee, *A Credit-based Home Access Point (CHAP) to Improve Application Quality on IEEE 802.11 Networks*, Ph.D. thesis, Worcester Polytechnic Institute, May 2010.

[9] Jeongeun Julie Lee and Maruti Gupta, "A New Traffic Model for Current User Web Browsing Behavior," Tech. Rep., Intel Research White Paper, 2007.

[10] Wikipedia, "List of WLAN Channels," February 2014, On line at http://en.wikipedia.org/wiki/List_of_WLAN_channels, Last Access on March 3rd, 2014.

[11] Globespanvirata.com, "Isl3893 Licencing," Online at: http://isl3893.sourceforge.net/GPL_letter_pub.pdf, Last Access on March 3rd, 2008.

[12] Stefan Mangold, Sunghyun Choi, Peter May, Ole Klein, Guido Hiertz, and Lothar Stibor, "IEEE 802.11e Wireless LAN for Quality of Service," in *Proceedings of European Wireless Conference*, Florence, Italy, February 2002.

[13] Johan Cimen, "Packet Scheduling in Wireless Network IEEE802.11 Using Linux," M.S. thesis, Ume University, Ume, Sweden, April 2004.

[14] Jae Won Chung, *Congestion Control for Streaming Media*, Ph.D. thesis, Worcester Polytechnic Institute, June 2005.

[15] Anil Kumar Singh and Bharat Mishra, "Comparative Study on Wireless Local Area Network Standards," *International Journal of Applied Engineering and Technology*, vol. 2, no. 7, July 2012.

[16] Rahul Bhoyar, Mangesh Ghonge, and Suraj Gupta, "Comparative Study on IEEE Standard of Wireless LAN/Wi-Fi 802.11 a/b/g/n," *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 2, no. 7, July 2013.

[17] Jon Gretarsson, Feng Li, Mingzhe Li, Ashish Samant, Huahui Wu, Mark Claypool, and Robert Kinicki, "Performance Analysis of the Intertwined Effects between Network Layers for 802.11g Transmissions," in *Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP)*, Montreal, Canada, October 2005.

[18] Marco Roccetti, Claudio Palazzi, Stefano Ferretti, and Giovanni Pau, *Encyclopedia of Wireless and Mobile Communications*, chapter Wireless Home Entertainment Center: Protocol Communications and Architecture, Auerbach Publications, Taylor & Francis Group, London (UK), 2007.

[19] Tiantian Guo, Jianfei Cai, and Chuan Heng Foh, "Scalable Video Adaptation in Wireless Home Networks with a Mixture of IPTV and VoD Users," in *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, Houston, TX, December 2011.

[20] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Philadelphia, PA, August 2005, pp. 229–240.

[21] Augustin Soule, Kave Salamatia, Nina Taft, Richard Emilion, and Konstantina Papagiannaki, "Flow Classification by Histograms: or How to Go on Safari in the Internet," *SIGMETRICS Performance Evaluation Review*, vol. 32, no. 1, pp. 49–60, 2004.

[22] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield, "Class-of-Service Mapping for QoS: a Statistical Signature-based Approach to IP Traffic Classification," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, Taormina, Sicily, Italy, October 2004.

[23] Jeffrey Erman, Anirban Mahanti, Martin Arlitt, Ira Cohen, and Carey Williamson, "Offline/Realtime Traffic Classification Using Semi-Supervised Learning," *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194–1213, 2007.

[24] Andrew Moore and Denis Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," *SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1, pp. 50–60, 2005.

[25] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill, "Flow Clustering Using Machine Learning Techniques," in *Proceedings of the 5th Passive and Active Measurement Workshop (PAM)*, Antibes Juan-les-Pins, France, April 2004.

[26] Sebastian Zander, Thuy Nguyen, and Grenville Armitage, "Automated Traffic Classification and Application Identification Using Machine Learning," in *Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN)*, Sydney, Australia, November 2005, pp. 250–257.

[27] Isara Anantavrasilp and Thorsten Scholer, "Automatic Flow Classification Using Machine Learning," in *Proceedings of the 15th International Conference on Software, Telecommunications and Computer Networks (SoftCom)*, Split-Dubrovnik, Croatia, September 2007, pp. 1–6.

[28] Jeffrey Erman, Anirban Mahanti, and Martin Arlitt, "Internet Traffic Identification Using Machine Learning," in *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, San Francisco, CA, November/December 2006.

[29] Samara Lynn, "Apple AirPort Extreme Base Station (A1521)," June 2013, On line at http://www.pcmag.com/article2/0,2817,2421124,00.asp, Last Access on May 1st, 2014.

[30] Mark Claypool, Robert Kinicki, and Craig Wills, "Treatment-Based Traffic Signatures," in *Proceeding of IMRG (IETF Internet Measurement Research Group) Workshop on Application Classification and Identification (WACI)*, Cambridge, MA, October 2007.

[31] Ashish Patro, Srinivas Govindan, and Suman Banerjee, "Observing Home Wireless Experience through WiFi APs," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking (MobiCom)*, Miami, FL, September 30 - Octber 04 2013, pp. 339–350.

[32] Dragos Niculescu, Samrat Ganguly, Kyungtae Kim, and Rauf Izmailov, "Performance of VoIP in a 802.11 Wireless Mesh Network," in *Proceedings of the 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Catalunya, Spain, April 2006, pp. 1–11.

[33] Jeonggyun Yu, Sunghyun Choi, and Jaehwan Lee, "Enhancement of VoIP over IEEE 802.11 WLAN via Dual Queue Strategy," in *Proceedings of the IEEE International Conference on Communications (ICC)*, Paris, France, June 2004, vol. 6.

[34] Feng Li, Jae Chung, Mingzhe Li, Huahui Wu, Mark Claypool, and Robert Kinicki, "Application, Network and Link Layer Measurements of Streaming Video over a Wireless Campus Network," in *Proceedings of the 6th Passive and Active Measurement Workshop (PAM)*, Boston, MA, April 2005.

[35] Feng Li, Mingzhe Li, Rui Lu, Huahui Wu, Mark Claypool, and Robert Kinicki, "Measuring Queue Capacities of IEEE 802.11 Wireless Access Points," in *Proceedings of the Fourth IEEE International conference on Broadband Communications, Networks and Systems (BROADNETS)*, Raleigh, NC, September 2007.

[36] Guojun Jin and Brian Tierney, "Netest: a Tool to Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput," in *Proceedings of the International Conference on Information Technology: Research and Education (ITRE)*, Newark, NJ, August 2003, pp. 578–582.

[37] Mingzhe Li, *Using Bandwidth Estimation to Optimize Buffer and Rate Selection for Streaming Multimedia over IEEE 802.11 Wireless Networks*, Ph.D. thesis, Worceter Polytechnic Institute, December 2006.

[38] Mingzhe Li, Mark Claypool, Robert Kinicki, and James Nichols, "Characteristics of Streaming Media Stored on the Web," *ACM Transaction on Internet Technology*, vol. 5, no. 4, pp. 601–626, 2005.

[39] Huahui Wu, *ARMOR - Adjusting Repair and Media Scaling with Operations Research for Streaming Video*, Ph.D. thesis, Worcester Polytechnic Institute, May 2006.

[40] Long Le, Jay Aikat, Kevin Jeffay, and F. Donelson Smith, "Differential Congestion Notification: Taming the Elephants," in *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)*, Berlin, Germany, October 2004, pp. 118–128.

[41] David Kotz and Kobby Essien, "Analysis of a Campus-wide Wireless Network," *Wireless Networking*, vol. 11, pp. 115–133, 2005.

[42] Jalel Ben-Othman, Souheila Bouam, and Farid Nait-Abdesselam, "802.11 Qos Cross-Layer Protocol Based Propagation Conditions Adaptation," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks (LCN)*, Clontarf Castle, Dublin, Ireland, October 2007, pp. 698–702.

[43] Regiane Barbosa, Josiane Rodrigues, Simone Fraiha, Hermnio Gomes, and Gervsio Cavalcante, "An Empirical Model for Propagation Loss Prediction in Indoor Mobile Communications Using Pade Approximant," in *Proceedings of the BMO/IEEE MTT-S International Conference on Microwave and Optoelectronics*, Braslia, Brasil, July 2005.

[44] Joshua Robinson and Edward Knightly, "A Performance Study of Deployment Factors in Wireless Mesh Networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, AK, May 2007, pp. 2054–2062.

[45] Mathias Kurth, Anatolij Zubow, and Jens-Peter Redlich, "Multi-Channel Link-Level Measurements in 802.11 Mesh Networks," in *Proceedings of the International Conference on Wireless Communications and Mobile Computing*, Vancouver, British Columbia, Canada, 2006, pp. 937–944.

[46] Marcelo Carvalho and Jose Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-Hop Networks," in *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, Atlanta, GA, November 2003, pp. 146–155.

[47] Proxim Corporation, "A Detailed Examination of the Environmental and Protocol Parameters that Affect 802.11g Network Performance," Tech. Rep., Proxim Wireless Networks, 2003.

[48] Wen-Tsuen Chen, Bo-Bin Jian, and Shou-Chih Lo, "An Adaptive Retransmission Scheme with QoS Support for the IEEE 802.11 MAC Enhancement," in *Proceedings of the IEEE 55th Vehicular Technology Conference (VTC)*, Birmingham, AL, May 2002, vol. 1, pp. 70–74.

[49] Francisco Micó, Pedro Cuenca, and Luis Orozco-Barbosa, "QoS Mechanisms for IEEE 802.11 Wireless LANs," in *Proceedings of the 7th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC)*, Toulouse, France, June/July 2004, pp. 609–623.

[50] Mathieu Lacage, Mohammad Manshaei, and Thierry Turletti, "IEEE 802.11 Rate Adaptation: a Practical Approach," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Venice, Italy, 2004, pp. 126–134.

[51] Ad Kamerman and Leo Monteban, "WaveLAN II: A High Performance Wireless LAN for Unlicensed Band," *Bell Labs Technical Journal*, 1997.

[52] Gavin Holland, Nitin Vaidya, and Paramvir Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001, pp. 236–251.

[53] IEEE Computer Society LAN MAN Standard Committee, "IEEE 802.11n-2009Amendment 5: Enhancements for Higher Throughput," Standard, October 2009.

[54] Cisco Inc, "802.11ac: The Fifth Generation of Wi-Fi," Tech. Rep., Cisco and/or its affiliates, January 2014.

[55] Anatolij Zubow and Robert Sombrutzki, "Adjacent channel interference in IEEE 802.11n," in *Proceeding of the IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, April 2012, pp. 1163–1168.

[56] OpenWrt, "OpenWrt Table of Hardware.," Online at http://wiki.openwrt.org/, Last Access on December 3rd, 2013.

[57] IEEE Computer Society LAN MAN Standard Committee, "IEEE 802.11e/D3.0, *Draft Supplement to Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS)*," Standard, May 2002.

[58] Quang Ni, "Performance Analysis and Enhancements for IEEE 802.11e Wireless Networks," *IEEE Network*, vol. 19, no. 4, pp. 21–27, July/August 2005.

[59] Javier del Prado Pavon and Sai Shankar Nandagopalan, "IEEE 802.11e MAC Signaling to Support Schedule QoS," October 2013, US Patent 8,553,703.

[60] Bob Braden, David Clark, and Scott Shenker, "RFC 1633 Integrated Services in the Internet Architecture: An Overview," June 1994, On line at http://andrew2.andrew.cmu.edu/rfc/rfc1633.html. Last Access on March 9, 2008.

[61] ITTC in IP QoS research, "IP Quality of Service : An Overview," 1997, Online at http://qos.ittc.ku.edu/ipqos/ip_qos.htm, Last Access on March 8, 2008.

[62] Bob Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin, "RFC 2205 - Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification," September 1997, On line at http://www.faqs.org/rfcs/rfc2205.html. Last Access on March 9, 2008.

[63] Markus Peuhkuri, "IP Quality of Service," Tech. Rep., Laboratory of Telecommunications Technology, Helsinki University of Technology, 1999.

[64] Juha Heinanen, Fred Baker, Walter Weiss, and John Wroclawski, "RFC 2597 - Assured Forwarding PHB Group," June 1997, On line at http://www.ietf.org/rfc/rfc2597.txt Last Access on March 9, 2008.

[65] Gahng-Seop Ahn, Andrew Campbell, Andras Veres, and Li-Hsiang Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, New York, NY, June 2002, pp. 457–466.

[66] Sebastian Zander, Thuy Nguyen, and Grenville Armitage, "Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics," in *Proceedings of the 6th Passive and Active Network Measurement (PAM)*, Boston, MA, April 2005, pp. 325–328.

[67] Wei Wei, Bing Wang, Chun Zhang, James Kurose, and Donald Towsley, "Classification of Access Network Types: Ethernet Wireless LAN, ADSL, Cable Modem or Dialup?," in *the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Miami, FL, March 2005, pp. 1060–1071.

[68] Srinivasan Keshav, "Packet-pair flow control," *IEEE/ACM Transactions on Networking*, 1994.

[69] Andrew Moore, Denis Zuev, and Michael Crogan, "Discriminators for Use in Flow-Based Classification," Tech. Rep. RR-05-13, Department of Computer Science, Queen Mary, University of London, Mile End Road, London E1 4NS, UK, August 2005.

[70] IANA, "Internet Assigned Numbers Authority (IANA) ," Online at http://www.iana.org/assignments/port-numbers. Last Access on Januaray 10th, 2014.

[71] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and Kimberly Claffy, "The CoralReef Software Suite as a Tool for System and Network Administrators," in *Proceedings of the 15th USENIX Conference on System Administration (LISA)*, San Diego, CA, 2001, pp. 133–144.

[72] Steven Lin and Nick McKeown, "A Simulation Study of IP Switching," *SIGCOMM Computer Communication Review*, vol. 27, no. 4, pp. 15–24, 1997.

[73] Mika Ilvesmaki, Kalevi Kilkki, and Marko Luoma, "Packets or Ports - the Decisions of IP Switching," *Broadband Networking Technologies*, vol. 3233, pp. 53–64, November 1997.

[74] Alistair King, "Corsaro," Online at http://www.caida.org/tools/measurement/corsaro/, Last Access on October 30th, 2013.

[75] Tanja Zseby, Alistair King, Nevil Brownlee, and kc Claffy, "The Day After Patch Tuesday: Effects Observable in IP Darkspace Traffic," in *Passive and Active Network Measurement Workshop (PAM)*, Hong Kong, China, March 2013, PAM 2013.

[76] Jawad Khalife, Amjad Hajjard, and Jesus Daz-Verdejo, "Performance of OpenDPI in Identifying Sampled Network Traffic," *Journal of Networks*, vol. 8, no. 1, pp. 71–81, January 2013.

[77] Tomasz Bujlow, Valentn Carela-Espaol, and Pere Barlet-Ros, "Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification," Tech. Rep. UPC-DAC-RR-CBA-2014-1, Department of Computer Architecture, Universitat Politcnica de Catalunya, Barcelona, Spain, January 2014.

[78] Demetres Antoniades, Michalis Polychronakis, Spiros Antonatos, Evangelos Markatos, Sven Ubik, and Arne sleb, "Appmon: An Application for Accurate per Application Network Traffic Characterization," in *Proceedings of BroadBand Europe*, Geneva, Switzerland, December 2006.

[79] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Srinivasan Seshan, and David Wetherall, "802.11 User Fingerprinting," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, Montreal, Quebec, Canada, 2007, pp. 99–110.

[80] Jason Franklin, Damon McCoy, Parisa Tabriz, Vicentiu Neagoe, Jamie Van Randwyk, and Douglas Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium (USENIX-SS)*, Vancouver, BC, Canada, 2006, pp. 12–12.

[81] Antonio Nucci, "Skype: the Future of Traffic Detection and Classification," Tech. Rep., PipelinePub.com, September 2006.

[82] Guowu Xie, Marios Iliofotou, Thomas Karagiannis, Michalis Faloutsos, and Yaohui Jin, "ReSurf: Reconstructing Web-surfing Activity from Network Traffic," in *Proceedings of the International Dederation for Information Processing (IFIP) Networking Conference*, Brooklyn, NY, May 2013, pp. 1–9.

[83] Kai Yang, Binqiang Wang, and Zhen Zhang, "A Method of Identifying P2P Live Streaming Based on Union Features," in *Proceedings of the 4th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, May 2013, pp. 426–429.

[84] Tom Mitchell, *Machine Learning*, McGraw-Hill Education, December 1997, ISE Editions.

[85] Kc Claffy, *Internet Traffic Characterization*, Ph.D. thesis, University of California at San Diego, 1997.

[86] Balachandra Krishnamurthy and Jennifer Rexford, "Web Protocol and Practice," Chapter 10: Web Workload Characterization, Addison-Wesley, 2001.

[87] Maureen Chesire, Alec Wolman, Geoffrey Voelker, and Henry Levy, "Measurement and Analysis of a Streaming Media Workload," in *Proceedings of USNIX Symposium on Internet Technologies and Systems*, San Francisco, CA, March 2001.

[88] Paul Barford, Jeffery Kline, David Plonka, and Amos Ron, " A Signal Analysis of Network Traffic Anomalies," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, Novemeber 2002.

[89] Nigel Williams, Sebastian Zander, and Grenville 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.

[90] Jun Zhang, Chao Chen, Yang Xiang, Wanlei Zhou, and Yong Xiang, "Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions," *Journal of IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 5–15, January 2013.

[91] Nigel Williams, Sebastian Zander, and Grenville Armitage, "Evaluating Machine Learning Algorithms for Automated Network Application Identification," Tech. Rep., Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Melbourne, Australia, April 2006.

[92] Nigel Williams, Sebastian Zander, and Grenville Armitage, "Evaluating Machine Learning Methods for Online Game Traffic Identification," Tech. Rep., Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Melbourne, Australia, 2006.

[93] CISCO, "NetFlow, Version 9 ," Online at: http://www.cisco.com/en/US/products/ps6645/products_ios_protocol_option_home.html, Last Access on October 30th, 2013.

[94] Henning Schulzrinne, Anup Rao, and Robert Lanphier, "Real Time Streaming Protocol (RTSP), Request for Comments 2326, 1998," Online at: ftp://ftp.isi.edu/in-notes/rfc2326.txt, Last Access on October 30th, 2013.

[95] Mark Hall and Geoffrey Holmes, "Benchmarking Attribute Selection Techniques for Discrete Class Data Mining," *IEEE Transaction on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437–1447, 2003.

[96] Mark Hall, *Correlation-based Feature Selection for Machine Learning*, Ph.D. thesis, Department of Computer Science, Waikato University, Hamilton, New Zealand, 1999.

[97] Tom Dunnigan and George Ostrouchov, "Flow Characterization for Intrusion Detection," Tech. Rep., Oak Ridge National Laboratory, November 2000.

[98] Thuy T. T. Nguyen, Grenville Armitage, Philip Branch, and Sebastian Zander, "Timely and Continuous Machine-learning-based Classification for Interactive IP Traffic," *Journal of IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 6, pp. 1880–1894, December 2012.

[99] Jun Zhang, Yang Xiang, Yu Wang, Wanlei Zhou, Yong Xiang, and Yong Guan, "Network Traffic Classification Using Correlation Information," *Jounal of IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 104–117, January 2013.

[100] Yu Jin, Nick Duffield, Jeffery Erman, Patrick Haffner, Subhabrata Sen, and Zhi-Li Zhang, "A Modular Machine Learning System for Flow-Level Traffic Classification in Large Networks," *Journal on ACM Transactions on Knowledge Discovery Data*, vol. 6, no. 1, March 2012.

[101] Bernd Ameel, Kathleen De Kerpel, Hugo Canière, Christophe TJoen, Henk Huisseune, and Michel De Paepe, "Classification of Two Phase Flows Using Linear Discriminant Analysis and Expectation Maximization Clustering of Video Footage," *International Journal of Multiphase Flow*, vol. 40, pp. 106–112, 2012.

[102] David Aha, Dennis Kibler, and Marc Albert, "Instance-Based Learning Algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[103] Wei Wei, Sharad Jaiswal, James Kurose, and Donald Towsley, "Identifying 802.11 Traffic from Passive Measurements Using Iterative Bayesian Inference," in *Proceedings of the 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Catalunya, Spain, April 2006.

[104] Valeria Baiamonte, Konstantina Papagiannaki, and Gianluca Iannaccone, "Detecting 802.11 Wireless Hosts from Remote Passive Observations," in *Proceedings of the 6th International IFIP-TC6 Networking Conference on Ad Hoc and Sensor Networks, Wireless Networks, and Next Generation Internet (NETWORKING)*, Atlanta, GA, May 2007, pp. 356–367.

[105] Claudio Palazzi, Giovanni Pau, Macro Roccetti, and Mario Gerla, "In-Home Online Entertainment: Analyzing the Impact of the Wireless MAC-Transport Protocols Interference," in *Proceedings of the Internation Conference on Wireless Networks, Communications and Mobile Computing*, Maui, HI, June 2005.

[106] Lucas DiCioccio, Renata Teixeira, and Catherine Rosenberg, "Measuring and Characterizing Home Networks," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, London, United Kingdom, June 2012, pp. 383–384.

[107] Igor Canadi Igor, Paul Barford, and Joel Sommers, "Revisiting Broadband Performance," in *Proceedings of the ACM Conference on Internet Measurement Conference (IMC)*, Boston, MA, November 2012, pp. 273–286.

[108] Miguel Rio, Tom Kelly, Mathieu Goutelle, Richard Hughes-Jones, and Jean-Philippe Martin-Flatin, "A Map of the Networking Code in Linux Kernel 2.4.20," Tech. Rep. DataTAG-2004-1, DataTag project (IST-2001-32459), March 2004.

[109] Minh-Son Nguyen and Quan Le-Trung, "Integration of Atheros ath5k device driver in wireless ad-hoc router," in *Proceedings of the International Conference on Advanced Technologies for Communications (ATC)* , Ho Chi Minh City, Vietnam, 2013, IEEE, pp. 609–613.

[110] Ren Ping Liu, Gordon J Sutton, and Iain B Collings, "A New Queueing Model for QoS Analysis of IEEE 802.11 DCF with Finite Buffer and Load," *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, pp. 5374–5374, 2013.

[111] Guosong Tian and Yu-Chu Tian, "QoS Differentiation in IEEE 802.11 Wireless Local Area Networks for Real-Time Control," in *Proceedings of the 19th Real-Time and Embedded Technology and Applications Symposium (RTS)*, Philadelphia, PA, April 2013, pp. 33–36.

[112] Jukka Manner, Markku Kojo, Aki Laukkanen, Mika Liljeberg, Tapio Suihko, and Kimmo Raatikainen, "Exploitation of Wireless Link QoS Mechanisms in IP QoS Architectures," in *Proceedings of SPIE on Quality of Service over Next-Generation Data Networks*, Denver, CO, August 2001, vol. 4524, pp. 273–283.

[113] Dan Siemon, "Queueing in the Linux Network Stack," *Linux Journal.*, July 2013.

[114] Matias Bjørling, Jens Axboe, David Nellans, and Philippe Bonnet, "Linux Block IO: Introducing Multi-Queue SSD Access on Multi-Core Systems," in *Proceedings of the 6th International Systems and Storage Conference (SYSTOR)*, Haifa, Israel, July 2013, p. 22.

[115] Madhavapeddi Shreedhar and George Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Cambridge, MA, 1995, pp. 231–242.

[116] Sally Floyd and Van Jacobson, "Link-sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, 1995.

[117] Jae Chung and Mark Claypool, "Dynamic-CBT and ChIPS - Router Support for Improved Multimedia Performance on the Internet," in *Proceedings of the 8th ACM International Conference on Multimedia (MULTIMEDIA)*, Marina del Rey, CA, 2000, pp. 239–248.

[118] Mark Parris, Kevin Jeffay, and Donelson Smith, "Lightweight Active Router-Queue Management for Multimedia Networking," in *Proceedings of the SPIE Conference on Multimedia Computing and Networking*, San Jose, CA, Janunary 1999, pp. 162–174.

[119] Sally Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, 1994.

[120] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[121] Dong Lin and Robert Morris, "Dynamics of Random Early Detection," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Cannes, France, September 1997, pp. 127–137.

[122] Kang Yong Lee, Jinsul Kim, Kee Seong Cho, and Ho-Jin Lee, "Traffic Aware QoS Scheduling for IEEE 802.11e HCCA WLAN," *IEICE Transactions on Communications*, vol. 96, no. 2, pp. 639–642, 2013.

[123] Vishal Sharma, Jagjit Malhotra, and Harsukhpreet Singh, "Quality of Service (QoS) Evaluation of IEEE 802.11 WLAN Using Different PHY-Layer Standards," *Optik-International Journal for Light and Electron Optics*, vol. 124, no. 4, pp. 357–360, 2013.

[124] Abhishek Kumar, "Traffic Sensitive Active Queue Management," M.S. thesis, Worcester Polytechnic Institute, December 2003.

[125] Mark Claypool, Robert Kinicki, and Abhishek Kumar, "Traffic Sensitive Active Queue Management," in *Proceedings of the 25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Miami, FL, March 2005.

[126] Claudio Palazzi, Giovanni Pau, Marco Roccetti, Stefano Ferretti, and Mario Gerla, "Wireless Home Entertainment Center: Reducing Last Hop Delays for Real-Time Applications," in *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE)*, Hollywood, CA, 2006, p. 67.

[127] Maodong Li, Zhenzhong Chen, and Yap-Peng Tan, "Cross-layer Optimization for SVC Video Delivery over the IEEE 802.11e Wireless Networks," *Journal of Visual Communication and Image Representation*, vol. 22, no. 3, pp. 284–296, 2011.

[128] Lei Guo, Enhua Tan, Songqing Chen, Zhen Xiao, Oliver Spatscheck, and Xiaodong Zhang, "Delving into Internet Streaming Media Delivery: A Quality and Resource Utilization Perspective," in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC)*, Rio de Janeriro, Brazil, 2006, pp. 217–230.

[129] Lachlan Andrew, Stephen Hanly, and Rami Mukhtar, "Active Queue Management for Fair Resource Allocation in Wireless Networks," *IEEE Transaction on Mobile Computing*, vol. 7, no. 2, pp. 231–246, February 2008.

[130] Lachlan Andrew, Stehphen Hanly, and Rami Mukhtar, "CLAMP: Active Queue Management at Wireless Access Points," in *Proceedings of European Wireless*, Nicosia, Cyprus, April 2005.

[131] Feng Li, Mingzhe Li, Rui Lu, Huahui Wu, Mark Claypool, and Robert 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, April 2006.

[132] Mingzhe Li, Mark Claypool, and Bob Kinicki, "Wireless Sniffing by Example – How to Build and Use an IEEE 802.11 Wireless Network Sniffer," Tech. Rep. WPI-CS-TR-05-19, Department of Computer Science at Worcester Polytechnic Institute, November 2005.

[133] Mingzhe Li, Feng Li, Mark Claypool, and Robert E. Kinicki, "Weather forecasting: predicting performance for streaming video over wireless LANs," in *Proceedings of the 15th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Stevenson, WA, June 2005, pp. 33–38.

[134] Catherine Boutremans, Gianluca Iannaccone, and Christophe Diot, "Impact of Link Failures on VoIP Performance," in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Miami, FL, 2002, pp. 63–71.

[135] Sebastian Zander and Grenville 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 (NOSSDAV)*, Stevenson, WA, June 2005, pp. 13–18.

[136] Frank Wattimena, Robert Kooij, Jeroen Van Vugt, and Kamal Ahmed, "Predicting the Perceived Quality of a first person shooter: the Quake IV G-model," in *Proceedings of 5th ACM SIGCOMM Workshop on Network and System Support for Games (NETGAME)*, Singapore, Singapore, 2006, pp. 42–45.

[137] Akamai Technologies, "The State of the Internet, 4th quarter, 2009 Report," Tech. Rep. 4, Akami Technologies, Inc, 2009.

[138] Akamai Technologies, "The State of the Internet, 3rd quarter, 2012 Report," Tech. Rep. 3, Akami Technologies, Inc, 2012.

[139] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman, "On Dominant Characteristics of Residential Broadband Internet Traffic," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC)*, Chicago, IL, 2009.

[140] Aaron Schulman, Dave Levin, and Neil Spring, "On the Fidelity of 802.11 Packet Traces," in *the 9th Passive and Active Measurement conference (PAM)*, Cleveland, OH, April 2008, pp. 132–141.

[141] Broadcom Inc, "The New Mainstream Wireless LAN Standard," Tech. Rep., Broadcom white paper 802.11g-WP104-R, 2003.

[142] Fadel Adib and Dina Katabi, "See through Walls with WiFi!," in *Proceedings of the ACM SIGCOMM*, Hong Kong, China, April 2013, pp. 75–86.

[143] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu, "Characterizing Residential Broadband Networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC)*, San Diego, CA, October 2007, pp. 43–56.

[144] Srikanth Sundaresan, Walter de Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè, "Broadband Internet Performance: a View from the Gateway," in *Proceedings of the ACM SIGCOMM Conference*, Toronto, Ontario, Canada, August 2011, pp. 134–145.

[145] Nevil Brownlee and kc Claffy, "Understanding Internet Traffic Streams: Dragonflies and Tortoises," *Communications Magazine, IEEE*, vol. 40, no. 10, pp. 110–117, 2002.

[146] Robert Cole and Josh Rosenbluth, "Voice over IP Performance Monitoring," *SIGCOMM Computer and Communication Review*, vol. 31, no. 2, pp. 9–24, 2001.

[147] Konstantina Papagiannaki, Mark Yarvis, and W.Steven Conner, "Experimental Characterization of Home Wireless Networks and Design Implications," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Barcelona, Spain, April 2006, pp. 1–13.

[148] Kenneth Calvert, Keith Edwards, Nick Feamster, Rebecca Grinter, Ye Deng, and Xuzi Zhou, "Instrumenting Home Networks," in *Proceedings of the 2011 ACM SIGCOMM Workshop on Home Networks (HomeNets)*, Toronto, Canada, August 2011, vol. 41, pp. 84–89.

[149] Lucas DiCioccio, Renata Teixeira, and Catherine Rosenberg, "Impact of Home Networks on End-to-End Performance: Controlled Experiments," in *Proceedings of the 2010 ACM SIGCOMM Workshop on Home networks (HomeNets)*, New Delhi, India, August 2010, pp. 7–12.

[150] Broadcom, "High-Speed 802.11A+G AP/Router MIPS + VPN Processor," Online at: http://www.broadcom.com/collateral/pb/94704AGR-PB00-R.pdf, Last Access on April 11, 2014.

[151] IBM, "PowerPC 405 Processor Core," Online at: http://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/, Last Access on April 11, 2014.

[152] APProsoftware.com, "RTL8186 Wireless LAN Gateway System Specification," Tech. Rep., APProsoftware Inc., 2004.

[153] IEPM/NPM team, "PingER WorldWide History 2013," July 2013, On line at http://www-iepm.slac.stanford.edu/pinger/, Last Access on July 2013.

[154] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer, "Equation-Based Congestion Control for Unicast Applications," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Stockholm, Sweden, August/September 2000, pp. 43–56.

[155] Anna Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, Sivaramakrishnan Muthukrishnan, and Martin Strauss, "Fast, Small-Space Algorithms for Approximate Histogram Maintenance," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, Montreal, Quebec, Canada, 2002, pp. 389–398.

[156] Geert Van Der Auwera, Prasanth David, and Martin Reisslein, "Traffic Characteristics of H.264/AVC Variable Bit Rate Video," *IEEE Communications Magazine*, 2007.

[157] "The Network Simulator - NS-2 ," Online at: http://www.isi.edu/nsnam/ns/, Last Access on April 10, 2014.

[158] "Network Simulator 2," March 2008, On line at http://www.isi.edu/nsnam/ns/ns-build.html, Last Access on April 10, 2014.

[159] "Tstat: TCP Statistic and Analysis Tool," October 2009, On line at http://tstat.tlc.polito.it/traces.shtml, Last Access on April 10, 2014.

[160] James Kinicki and Mark Claypool, "Traffic analysis of avatars in Second Life," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Braunschweig, Germany, June 2008, pp. 69–74.

[161] "Network Monitoring And Measurements: Traffic Traces," June 2004, On line at http://traffic.comics.unina.it/Traces/ttraces.php, Last Access on April 10, 2014.

[162] Anthony Cricenti and Branch Philip, "ARMA(1,1) modeling of Quake4 Server to Client Game Traffic," in *Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games*, Melbourne, Australia, 2007, NetGames, pp. 70–74.

[163] Tanja Lang and Grenville Armitage, "An NS2 Model for the Xbox System Link Game Halo, (TR-030613A)," Tech. Rep., Swinburne University of Technology, Melbourne, Australia, 2003.

[164] Kyoungwon Suh, Daniel Figueiredo, Jim Kurose, and Dan Towsley, "Characterizing and Detecting Skype-Relayed Traffic," in *the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Barcelona, Spain, April 2006, pp. 1–12.

[165] Mohd Nazri Ismail, "Analyzing of MOS and Codec Selection for Voiice over IP Technology," Tech. Rep., Department of MIIT, University of Kuala Lumpur (UniKL), Malasyia, 2009.

[166] Geert Van Der Auwera, Prasanth T. David, and Martin Reisslein, "Traffic and Quality Characterization of Single-Layer Video Streams Encoded with the H.264/MPEG4 Advanced Video Coding Standard and Scalable Video Coding Extension," *ACM Transaction on Internet Technology*, vol. 54, pp. 698–718, 2008.

[167] Patrick Seeling, Martin Reisslein, and Beshan Kulapala, "Network Performance Evaluation Using Frame Size and Quality Traces of Single-Layer and Two-Layer Video: A Tutorial.," *IEEE Communications Surveys and Tutorials*, vol. 6, no. 1-4, pp. 58–78, 2004.

[168] Feng Li, Mark Claypool, and Robert Kinicki, "QFind Methodology with a Host AP," Tech. Rep. WPI-CS-TR-06-03, Computer Science Department at Worcester Polytechnic Institute, June 2006.

[169] Claudio Palazzi, Stefano Ferretti, Giovanni Pau, Macro Roccetti, and Mario Gerla, "What's in that Magic Box? The Home Entertainment Center's Special Protocol Potion, Revealed," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 4, pp. 1280–1288, November 2006.

[170] Mark Claypool, "Warcraft III Online Trace," 2003, Online at http://perform.wpi.edu/downloads/war3/war3-traces.zip, Last Access on April 11, 2014.

[171] Marko Luoma, Mika Ilvesmki, and Markus Peuhkuri, "Source Characteristics for Traffic Classification in Differentiated Services Type of Networks," in *the Internet III: Quality of Service and Future Directions (VV01), SPIE*, Boston, MA, September 1999.

[172] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, and Donald F. Towsley, "Inferring TCP Connection Characteristics Through Passive Measurements.," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOMM)*, HongKong, China, March 2004, vol. 3.

[173] Hao Jiang and Constantinos Dovrolis, "Passive Estimation of TCP Round-Trip Times," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 75–88, July 2002.

[174] Jason But, Urs Keller, David Kennedy, and Grenville Armitage, "Passive TCP Stream Estimation of RTT and Jitter Parameters," in *Proceedings of the 30th IEEE Conference on Local Computer Networks(LCN)*, Sydney, Australia, November 2005.

Appendix: Summary of Simulation Results

Table A.1: Cumulative Throughput (Mbps) with IEEE 802.11g Links

Background	Foreground	short			mixed			long		
		CATNAP	DropTail	%	CATNAP	DropTail	%	CATNAP	DropTail	%
FTP	FTP	22.05	22.45	-1.7%	23.01	22.56	+2.0%	21.26	21.87	-2.7%
	NFS w/UDP	25.76	30.90	**-16.6%**	-	-	-	25.86	30.94	**-16.4%**
	Game w/UDP	20.08	20.97	-4.2%	20.15	21.23	-5.1%	17.08	17.45	-2.1%
	Game w/TCP	20.05	20.63	-2.8%	19.96	21.24	-6.0%	17.06	17.63	-3.2%
	VoIP w/UDP	19.95	20.26	-1.5%	20.31	21.52	-5.6%	16.87	16.75	+0.7%
	VoIP w/TCP	20.20	20.64	-2.1%	20.15	21.22	-5.0%	18.85	18.39	+2.5%
	Video w/UDP	19.19	20.40	-5.9%	19.02	21.70	**-12.3%**	18.74	18.90	-2.2%
	Video w/TCP	19.20	20.60	-6.8%	18.80	21.65	**-13.1%**	17.98	19.41	-7.3%
	Web	20.40	21.35	-4.4%	20.31	21.47	-5.4%	17.32	18.27	-5.1%
NFS w/UDP	Game w/UDP	30.47	30.61	-0.4%	-	-	-	30.47	30.62	-0.4%
	Game w/TCP	30.44	30.60	-0.5%	-	-	-	30.46	30.52	-0.2%
	VoIP w/UDP	30.43	30.58	-0.5%	-	-	-	30.45	30.59	-0.5%
	VoIP w/TCP	30.40	30.65	-0.8%	-	-	-	30.45	30.67	-0.7%
	Video w/UDP	30.54	30.14	+1.3%	-	-	-	30.48	30.16	+1.1%
	Video w/TCP	30.44	30.91	-1.2%	-	-	-	30.24	30.95	-2.3%

Table A.2: Quality of Foreground Applications with IEEE 802.11g Links

Background	Foreground	Quality	short			mixed			long		
			CATNAP	DropTail	%	CATNAP	DropTail	%	CATNAP	DropTail	%
FTP	FTP	Tput(Mbps)	11.60	11.45	+1.3%	12.10	5.92	**+104.4%**	10.53	8.89	**+18.4%**
		Fairness	0.99	0.98	+1.0%	0.99	0.80	**+23.8%**	1.00	0.94	+6.4%
	NFS w/UDP	Tput(Mbps)	13.74	30.75	*-55.3%*	-	-	-	13.28	30.91	*-57.0%*
		Fairness	0.99	0.51	**+94.1%**	-	-	-	0.99	0.50	**+98.0%**
	Game w/UDP	Tput(Mbps)	0.013	0.013	0%	0.013	0.013	0%	0.013	0.013	0%
		MOS (10% tail)	4.08	2.58	**+58.1%**	2.50	1.64	**52.4%**	2.50	2.16	**+15.7%**
	Game w/TCP	Tput(Mbps)	0.018	0.018	0%	0.018	0.018	0%	0.018	0.018	0%
		MOS (10% tail)	4.08	2.64	**54.5%**	2.49	1.60	**55.6%**	2.49	1.62	**53.7**
	VoIP w/UDP	Tput(Mbps)	0.08	0.08	0%	0.08	0.08	0%	0.08	0.08	0%
		MOS (10% tail)	4.39	4.34	1.1%	4.38	4.30	1.8%	4.38	4.31	1.6%
	VoIP w/TCP	Tput(Mbps)	0.088	0.088	0%	0.088	0.088	0%	0.088	0.088	0%
		MOS (10% tail)	4.39	4.38	+0.2%	4.34	4.31	+0.6%	4.34	4.30	0.9%
	Video w/UDP	Tput(Mbps)	1.42	1.42	0%	1.42	1.42	0%	1.42	1.42	0%
		10% Fr Rate(fps)	29.0	29.0	0%	29.0	29.0	0%	30.0	30.0	0%
		Fr Delay (sec)	0.002	0.019	**-89.5%**	0.002	0.085	**-97.6%**	0.002	0.008	**-75.0%**
	Video w/TCP	Tput(Mbps)	1.47	1.47	0%	1.47	1.47	0%	1.47	1.47	0%
		10% Fr Rate(fps)	30.0	29.0	3.4%	30.0	29.0	3.4%	30.0	30.0	0%
		Fr Delay (sec)	0.004	0.021	**-80.9%**	0.004	0.091	**-95.6%**	0.004	0.018	**-77.8%**
	Web	Tput(Mbps)	2.31	1.87	**+23.5%**	1.13	0.80	**+41.3%**	1.23	0.70	**+75.7%**
		Resp. Time (sec)	0.331	0.383	**-13.6%**	1.170	2.253	**-48.1%**	1.278	2.934	**-56.4%**
NFS w/UDP	Game w/UDP	Tput(Mbps)	0.013	0.010	**+30.0%**	-	-		0.013	0.010	**+30.0%**
		MOS (10% tail)	4.11	3.36	**+22.3%**	-	-		2.51	1.05	**+139.0%**
	Game w/TCP	Tput(Mbps)	0.015	0.015	0%	-	-		0.015	0.052*	0%
		MOS (10% tail)	4.12	2.51	**+64.1%**	-	-		2.51	1.05	**+139.0%**
	VoIP w/UDP	Tput(Mbps)	0.080	0.061	**+31.1%**	-	-		0.080	0.060	**+33.3%**
		MOS (10% tail)	4.39	2.56	**+71.5%**	-	-		4.34	1.21	**+258.7%**
	VoIP w/TCP	Tput(Mbps)	0.085	0.037	**+129.7%**	-	-		0.085	0.010	**+750.0%**
		MOS (10% tail)	4.39	2.25	**+95.1%**	-	-		4.34	0.93	**+366.7%**
	Video w/UDP	Tput(Mbps)	1.42	0.79	**+79.7%**	-	-		1.47	0.77	**+91.0%**
		10% Fr Rate(fps)	30.0	26.0	**+15.4%**	-	-		30.0	26.0	**+15.4%**
		Fr Delay (sec)	0.002	0.035	**-94.3%**	-	-		0.004	0.349	**-98.9%**
	Video w/TCP	Tput(Mbps)	1.47	0.25	**+488.0%**	-	-		1.47	0.16	**+818.0%**
		10% Fr Rate(fps)	30.0	9.0	**+233.3%**	-	-		30.0	1.6	**1775.0%**
		Fr Delay (sec)	0.004	0.036	**-88.9%**	-	-		0.004	0.349	**-98.9%**

Table A.3: Cumulative Throughput (Mbps) with IEEE 802.11b Links

Background	Foreground	short			mixed			long		
		CATNAP	DropTail	%	CATNAP	DropTail	%	CATNAP	DropTail	%
FTP	FTP	7.25	7.55	-4.0%	7.92	7.68	+3.1%	7.89	7.81	+1.0%
	NFS w/UDP	8.08	9.30	*-13.1%*	-	-	-	8.30	9.39	*-11.6%*
	Game w/UDP	6.63	6.05	+9.6%	6.60	7.30	-9.6%	6.07	7.25	*-16.2%*
	Game w/TCP	6.64	6.03	**+10.1%**	6.60	7.41	*-10.9%*	7.06	7.10	-0.6%
	VoIP w/UDP	6.59	6.21	+6.1%	6.99	7.39	-5.4%	6.92	7.35	-5.9%
	VoIP w/TCP	6.61	6.09	+8.5%	6.60	7.40	*-10.8%*	6.50	7.22	-9.9%
	Video w/UDP	6.72	6.31	+6.5%	6.78	7.81	*-13.2%*	7.01	7.84	*-10.6%*
	Video w/TCP	6.75	6.20	+8.9%	6.81	7.64	*-10.8%*	7.66	6.84	**+12.0%**
	Web	6.66	6.42	+3.7%	6.61	7.48	*-11.6%*	6.80	6.01	**+13.1%**
NFS w/UDP	Game w/UDP	9.23	9.26	-0.3%	-	-	-	9.23	9.26	-0.3%
	Game w/TCP	9.22	9.39	-1.8%	-	-	-	9.19	9.39	-2.1%
	VoIP w/UDP	9.19	9.26	-0.8%	-	-	-	9.19	9.26	-0.8%
	VoIP w/TCP	9.19	9.39	-2.1%	-	-	-	9.22	9.39	-1.8%
	Video w/UDP	9.37	9.39	-0.2%	-	-	-	9.37	9.39	-0.2%
	Video w/TCP	9.39	9.39	0%	-	-	-	9.40	9.39	+1.1%

Table A.4: Quality of Foreground Applications with IEEE 802.11b Links

Background	Foreground	Quality	short			mixed			long		
			CATNAP	DropTail	%	CATNAP	DropTail	%	CATNAP	DropTail	%
FTP	FTP	Tput(Mbps)	3.53	3.87	-8.8%	4.00	2.73	**+46.5%**	4.19	2.03	**106.4%**
		Fairness	1.00	0.96	+4.2%	1.00	0.80	**+25.0%**	0.99	0.92	+7.6%
	NFS w/UDP	Tput(Mbps)	4.53	9.39	*-48.2%*	-	-		4.39	9.39	*-53.2%*
		Fairness	0.96	0.50	**+92.0%**	-	-		0.99	0.50	**98.0%**
	Game w/UDP	Tput(Mbps)	0.013	0.013	0%	0.013	0.013	0%	0.013	0.013	0%
		MOS (10% tail)	4.03	3.04	**+32.6%**	2.46	1.15	**113.9%**	2.46	1.27	**+93.7%**
	Game w/TCP	Tput(Mbps)	0.018	0.018	0%	0.018	0.018	0%	0.018	0.018	0%
		MOS (10% tail)	4.04	3.00	**+34.7%**	2.46	1.13	**117.7%**	2.45	1.16	**111.2%**
	VoIP w/UDP	Tput(Mbps)	0.08	0.08	0%	0.08	0.08	0%	0.08	0.08	0%
		MOS (10% tail)	4.39	4.17	+5.3%	4.34	3.46	**+25.4%**	4.34	3.89	**+11.6%**
	VoIP w/TCP	Tput(Mbps)	0.088	0.088	0%	0.088	0.088	0%	0.088	0.088	0%
		MOS (10% tail)	4.39	4.17	+5.2%	4.34	3.35	**+29.6%**	4.34	3.85	**+12.7%**
	Video w/UDP	Tput(Mbps)	1.42	1.40	+1.4%	1.42	1.44	-1.4%	1.43	1.42	+0.7%
		10% Fr Rate(fps)	29.0	26.0	**+11.5%**	29.0	27.0	+7.4%	30.0	27.0	**+11.1%**
		Fr Delay (sec)	0.006	0.012	**-50.0%**	0.006	0.192	**-96.9%**	0.006	0.166	**-96.4%**
	Video w/TCP	Tput(Mbps)	1.47	1.47	0%	1.47	1.47	0%	1.47	1.47	0%
		10% Fr Rate(fps)	29.0	27.0	+7.4%	30.0	25.0	**+20.0%**	30.0	29.0	+3.4%
		Fr Delay (sec)	0.012	0.014	**-14.3%**	0.012	0.195	**-93.8%**	0.013	0.061	**-78.7%**
	Web	Tput(Mbps)	3.27	1.86	**+75.8%**	1.31	0.80	**+63.8%**	1.51	0.70	**+115.7%**
		Resp. Time (sec)	0.174	0.383	**-54.6%**	1.013	2.253	**-55.0%**	1.235	2.934	**-57.9%**
NFS w/UDP	Game w/UDP	Tput(Mbps)	0.013	0.006	**+116.7%**	-	-	-	0.013	0.006	**+116.7%**
		MOS (10% tail)	4.07	3.60	**+13.1%**	-	-	-	2.48	1.16	**+113.8%**
	Game w/TCP	Tput(Mbps)	0.018	0.010	**+80.0%**	-	-	-	0.019	0.013	**46.2%**
		MOS (10% tail)	4.26	3.08	**+38.3%**	-	-	-	2.52	1.46	**72.6%**
	VoIP w/UDP	Tput(Mbps)	0.080	0.019	**+321.1%**	-	-	-	0.080	0.019	**+321.1%**
		MOS (10% tail)	4.39	1.42	**+209.2%**	-	-	-	4.34	0.52	**+731.4%**
	VoIP w/TCP	Tput(Mbps)	0.080	0.00	**+100.0%**	-	-	-	0.079	0.005	**+100.0%**
		MOS (10% tail)	3.85	0	**+100.0%**	-	-	-	3.97	≈ 0	**+100.0%**
	Video w/UDP	Tput(Mbps)	1.42	0.21	**+576.1%**	-	-	-	1.43	0.20	**+615.0%**
		10% Fr Rate(fps)	27.0	9.0	**+300.0%**	-	-	-	26.0	9.0	**+300.0**
		Fr Delay (sec)	0.010	0.026	**-61.5%**	-	-	-	0.010	0.256	**-96.1%**
	Video w/TCP	Tput(Mbps)	1.47	0.25	**+488.0%**	-	-	-	1.47	0.01	**100%**
		10% Fr Rate(fps)	30.0	1.0	**100%**	-	-	-	30.0	-	**100%**
		Fr Delay (sec)	0.014	0.050	**-72.0%**	-	-	-	0.015	-	**100%**