IMPROVED IPTV CHANNEL CHANGE TIMES THROUGH MULTICAST CACHING OF PRE-SELECTED CHANNELS

By

THOMAS R. RAY

Bachelor of Science in Electrical Engineering
University of Arkansas
Fayetteville, Arkansas
1991

Master of Science in Electrical Engineering
University of Arkansas
Fayetteville, Arkansas
1994

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December 2014

IMPROVED IPTV CHANNEL CHANGE TIMES

THROUGH MULTICAST CACHING OF

PRE-SELECTED CHANNELS

Dissertation Approved:

Dr. George Scheets

Dissertation Adviser

Dr. Charles Bunting

Dr. Carl Latino

Dr. Mark Weiser

Name: RAY, THOMAS

Date of Degree: DECEMBER, 2014

Title of Study: IMPROVED IPTV CHANNEL CHANGE TIMES THROUGH MULTICAST CACHING OF PRE-SELECTED CHANNELS

Major Field: ELECTRICAL ENGINEERING

Abstract: IPTV has grown in recent years to an estimated 100 million users worldwide. IPTV uses IGMP processes to stream an individual channel to a user until the next channel change when the current channel is stopped and the new selection begins streaming. One of the critical factors determining customer satisfaction is the requirement to have reasonably rapid channel change times of 2 seconds or less, but current channel change times are frequently above that threshold. Numerous research efforts have been ongoing to reduce these times including edge servers, I-frame management, buffering improvements, dynamic video coding, and pre-selecting channels. Channel pre-selection involves sending additional channels in hopes that the user's next selection will already be present at the user's set top box to reduce the channel change time. While this pre-selection technique has previously been proposed, the proposals have been limited in scope, typically based on set top box replacement, and lack specific details regarding the expected channel change reductions attained. This research addressed all of these shortcomings beginning with laboratory testing to verify that the channel change time reduction for successful pre-selection is two times the network delay plus the IGMP processing time which equates to an average of 320 millisecond reduction per channel change. Several pre-selection models were developed and evaluated using theoretical calculations, functional testing, and performance simulations. Sample data was generated to reflect a wide range of user IPTV viewing behavior for use in the performance simulations. The top two models resulted in an average of well over 70% success rates in accurately pre-streaming the user's next selection in the multicast cache output. This approach also has the benefit of being implemented on IPTV provider equipment and would typically only require firmware upgrades without the need for expensive new equipment or changes to existing standards. Operational considerations were also discussed to reduce problems and delays during the implementation phase of the system. Additional applications and future improvements were also presented.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Television viewers have grown to expect high quality service through traditional broadcasting techniques over the many past decades. Whether from cable or satellite services today or even earlier electromagnetic services of the past, users have had an inherent advantage in that all of the offered channels are simultaneously broadcast to their location. This allows for a rapid channel change time whenever the user selects a new channel for viewing, since the user's equipment only has to adjust to the new frequency associated with the next channel for reception. Due to this inherent advantage, users have come to expect fast channel change times from their television service. However, these traditional techniques are limited by fixed channel offerings and unidirectional services. For these reasons traditional providers are limited in the number of High Definition (HD) channels that they can provide.

With the growth of the internet, a wide range of new services have been offered in recent years. Many providers now offer various combinations of traditional data services, Voice over Internet Protocol (VoIP), Internet Protocol TV (IPTV), Video on Demand (VoD), content streaming, gaming, informational services, and mobile access. IPTV has been tightly defined as the transmission of video over broadband IP networks using multicast technology [1]. But it is often more loosely defined to be a service that includes multimedia applications such as TV, video, audio, text, graphics, and data over an IP based network [2]. For the purposes of this paper, the first more tightly worded definition will be used, but it shall be understood that all of these IPTV services are typically offered in a multi-service environment.

IPTV is ushering in an era where all of the offered channels are no longer simultaneously present down to each subscriber's location. Instead typically only the specific channel currently requested is transmitted. This has led to the capability of offering an increasing number of channels and services, since a fixed number of channel offerings that are all simultaneously broadcast would no longer be a limiting factor. The signaling between the user and the network for channel requests is most frequently done using the Internet Group Management Protocol (IGMP), and the inherent delays associated with this channel change time and other quality issues have now come to the forefront as some of the major technical issues associated with IPTV services. Users that have long experienced near instantaneous channel change times have come to expect similar performance from these newly offered IPTV services, so the goal of this research is to offer an improved multicasting method to significantly reduce the average channel change time experienced by users.

1.1 Rise of IPTV

As the number of broadband internet users has steadily increased in recent years due to the availability of Digital Subscriber Lines (xDSL), Fiber-to-the-Home (FTTH), cable modem services, and other Next Generation Network (NGN) offerings, customer acceptance and interest in a wide variety of multimedia services has been high. Providers have made Triple Play services relatively commonplace in which voice, video, and data applications are merged onto a single platform. By then adding remote wireless access onto this package, Quadruple Play services have become available in many metropolitan markets. Providers are now looking further into providing Multiplay packages in which further advanced applications, such as Video-on-Demand and Informational Services, are then merged into a single ubiquitous service. IPTV service has now become a key competitive component for companies looking to distinguish themselves over their competitors.

Worldwide IPTV growth continued over the last several years aided by the increased broadband customer base, technical improvements, introduction of standards, and regulatory reform. Some key metrics associated with this IPTV growth are:

- The number of households worldwide with broadband service was estimated to be 422 million for 2010, with 139 million of those having sufficient bandwidth capabilities to allow reception of IPTV services [3].
- 2011 estimates of worldwide IPTV subscribers were over 100 million [4].
- Europe, Asia, and North America are currently the largest IPTV regions of growth [4] [5].
- European estimates for IPTV subscribers in 2010 topped 11 million [6].
- IPTV usage in Japan has risen by over 50% annually and exceeded the 3 million customer mark by 2012 [7].
- The 2009 outlook for IPTV users in Korea topped 2.5 million [8].
- In the U.S., AT&T's U-Verse program had over 1 million subscribers through 2008 and other competitors were increasing investments in IPTV capabilities [9].

- China had already approached 1 million IPTV customers by the end of 2007 and continues to be one of the fastest growth markets [10].
- Even with an initially low broadband penetration, India represents another large area of IPTV growth and has attracted substantial recent investments [11].

Taken collectively, these metrics clearly indicate the favorable direction that IPTV services have taken in recent years. However, in the U.S. there are still whole market areas that have yet to be penetrated. The Multi-Dwelling Unit (MDU) market represents a large opportunity for IPTV providers, as well as the markets associated with large office buildings and business centers [12]. The MDU space generally refers to apartment complexes, and as such it provides a high concentration of similarly configured locations that could speed IPTV deployment. The high potential of this market is evident by the over 18 million apartment units rented in U.S. annually [13]. Note that this figure excludes single family homes and trailer rentals. Several of the large apartment management companies each offer services to many thousands of apartment complexes, and these companies have now set their sights on providing IPTV and other integrated services [14]. Similarly the high number of office buildings in the U.S., currently standing at well over 700,000, represents another opportunity where providers benefit from both the concentration and uniformity of the configurations [15].

It is important to note that IPTV deployments are made in a wide variety of configurations. Passive Optical Networks are often deployed to residential areas in a much different configuration than is typically implemented at MDU locations. Furthermore, deployments differ with respect to various regions and countries worldwide as a result of differing telecommunications standards, practices, and environments. This research effort will primarily focus on the architecture associated with the MDU deployment space, although many of the concepts and conclusions are applicable to other configurations as well. A typical configuration associated with an MDU deployment of IPTV services is shown on Figure 1.1.

While implementations can differ significantly, it is common for the majority of the offered IPTV channels along with other integrated services to be simultaneously present at a central Head End location for major metropolitan areas. The First Hop Router (FHR) is nearest to the Head End and multicast signals are broadcast down to the Last Hop Router (LHR). The LHR is frequently the last edge Layer 3 device to have a significant

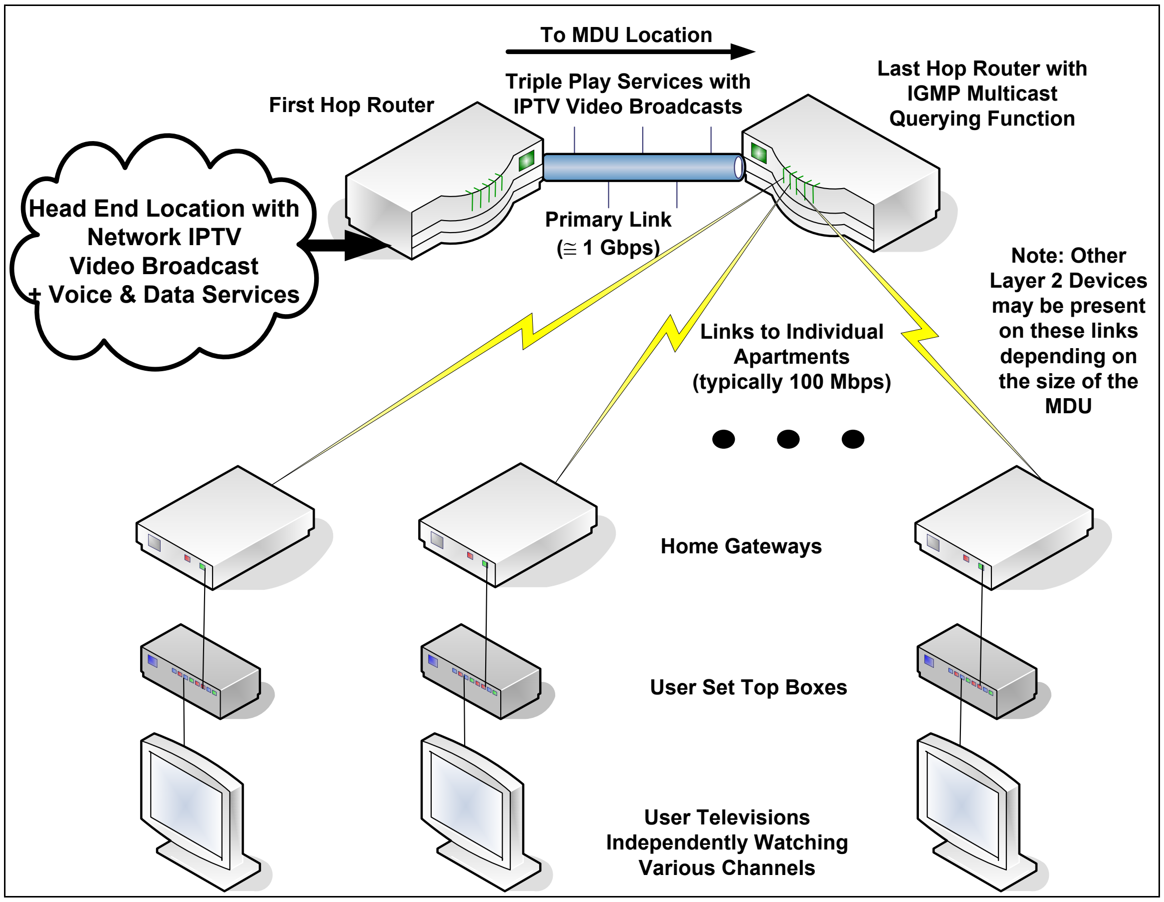

To MDU Location
Triple Play Services with IPTV Video Broadcasts
First Hop Router
Last Hop Router with IGMP Multicast Querying Function
Head End Location with Network IPTV Video Broadcast + Voice & Data Services
Primary Link (≅ 1 Gbps)
Note: Other Layer 2 Devices may be present on these links depending on the size of the MDU
Links to Individual Apartments (typically 100 Mbps)
Home Gateways
User Set Top Boxes
User Televisions Independently Watching Various Channels

Figure 1.1 Typical IPTV Configuration for MDU Deployments

number of offered channels present and is often configured to be the IGMP Querying Device. Other Layer 2 devices may be present between the LHR and the user's Home Gateway, and these devices are typically configured to perform IGMP Snooping. Individual apartments typically have a Home Gateway (HG) to handle the integration of voice, video, and data within the apartment itself. A Set Top Box (STB) is a required device at the customer premises to handle IPTV service, and there are a wide variety of these devices on the market with differing levels of performance and options. All of these various functional devices and concepts will be covered in detail in later sections. Due to the nature of several different apartment complexes having similar configurations and that the individual apartments within each complex are implemented in duplicate fashions, this MDU architecture represents a valuable market to the major providers in this country.

1.2 Popular Video Formats

Live TV video broadcasts are first broken down into small chunks of data and placed into packets for transport across networks for IPTV reception much like standard data has been packetized for internet delivery for years. However, IPTV data is generally placed in User Datagram Protocol (UDP) packets, whereas standard internet data is most frequently placed in Transmission Control Protocol (TCP) packets. TCP is considered a reliable connection that allows for ordering, acknowledgement, timeout, and retransmission of packets. UDP is unreliable in that it does not offer any method of guarantee that packets will arrive at their destinations, so it is used only for applications that can accept some level of error. Since live video streams require a significant amount of bandwidth to be delivered in a time sensitive fashion, most IPTV applications have currently been implemented using UDP connections. To improve quality some STB providers have implemented error concealment techniques to minimize the effects of errors or lost packets at the price of increased complexity and processing delays [16-19].

UDP was formally defined in the standard IETF RFC 768 going back to 1980. The IGMP process was defined in the standards RFC 1112, RFC 2236, and RFC 3376 for versions 1, 2, and 3 respectively, and this process will be covered in detail in the next section. As IPTV usage has increased, the Focus Group of ITU-T issued base specifications in December 2007 and the IPTV-GSI (Global Standard Initiative) was established in early 2008 to continue coordinating recommendations for IPTV specifications. The Motion Pictures Expert Group (MPEG) was formed in 1988 and began issuing formal standards for video coding in 1993 and has steadily issued additional standards ever since. More information on IPTV related standards can be found in Appendix A. SDTV differs from HDTV primarily in the areas of resolution, scanning, and refresh rates. As the state of the art of HDTV has progressed the ITU and other standards organizations have struggled to keep up. While a large number of video formats exist, the most popular video formats are listed in Table 1.1 along with an

Format	Typical Bit Rates
MPEG-2 SDTV	3-6 Mbps
MPEG-2 HDTV	14-20 Mbps
H.264/MPEG-4 SDTV	1-4 Mbps
H.264/MPEG-4 HDTV	6-14 Mbps

Table 1.1 Popular IPTV Video Formats

overview of the bit rates typically associated with each.

The common format of MPEG-2 Transport Stream (TS) is produced with sequence of fixed length 188 byte packets as shown in Figure 1.2 with more details in Appendix A [20]. The video packet stream is formed from video frames that are sized depending on the aspect ratio and resolution. The frame rate can vary depending on applications, and frames are likewise placed into a Group of Pictures (GOP). As the TS is received at the user's location, it is vital for the STB to be able to quickly determine

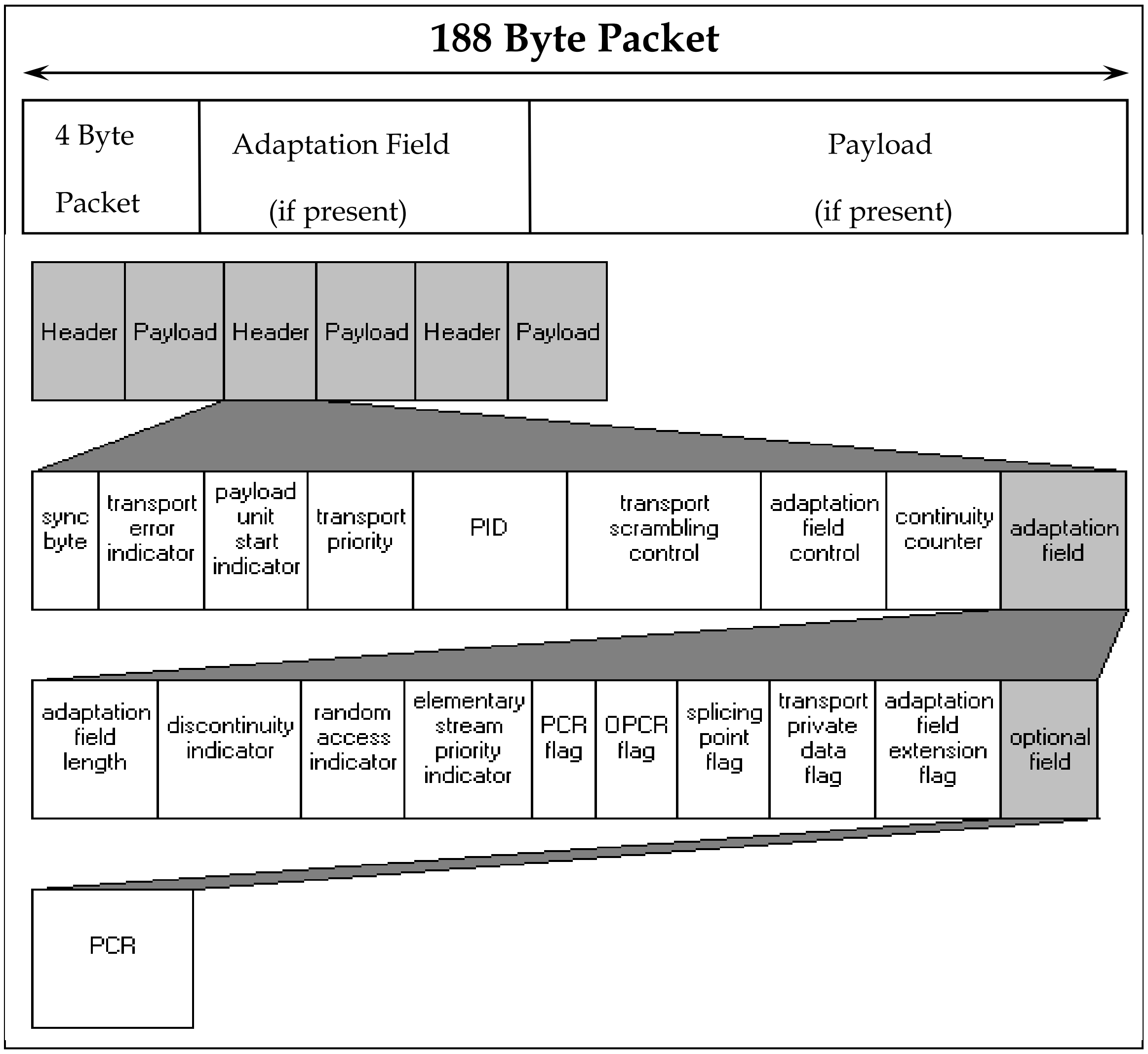


Figure 1.2 Transport Stream Packet Structure and Header [20]

where in a particular video frame the data is intended to occupy and the compression technique. This function is aided through the use of frame types, most commonly the I, P, and B frame types. The Intra-coded frame, known as an I-Frame in MPEG-2 and an Instantaneous Data Refresh (IDR) in MPEG-4, contains all of the information necessary to decode the stream, so they are known as the Key Frames [1][21-22]. Since set top boxes have no way of knowing in advance where the video stream of packets had begun, it is mandatory to first receive one of these Key Frames before any decoding and display can begin. Due to this limitation, an I-Frame serves as a Random Access Point (RAP) into an ongoing video stream. It is also important to keep in mind that I-Framing specifies a full and complete picture. Significant compression can occur through the use of Predictive Frames (P-Frames) that indicate only data that differs from the previous frame [23]. Bi-predictive Frames (B-Frames) go even further by indicating only data that differs from the previous and next frames. The I-Frame rate is considered the most significant type of framing for analysis of channel change time (CCT), since it is initially required for decoding, it is required for later P and B framing to be effective, and it heavily influences the required bandwidth [21-22]. I-Framing optimization and dynamic I-Framing are two well-studied techniques that have been used to improve CCT and will be covered more in the next chapter.

1.3 IGMP Multicasting Protocol

Internet Group Management Protocol (IGMP) is used to facilitate the multicasting of video streams more effectively than traditional transmission methods. The multicasting process is a method to optimize video transmission by reducing the number of duplicate

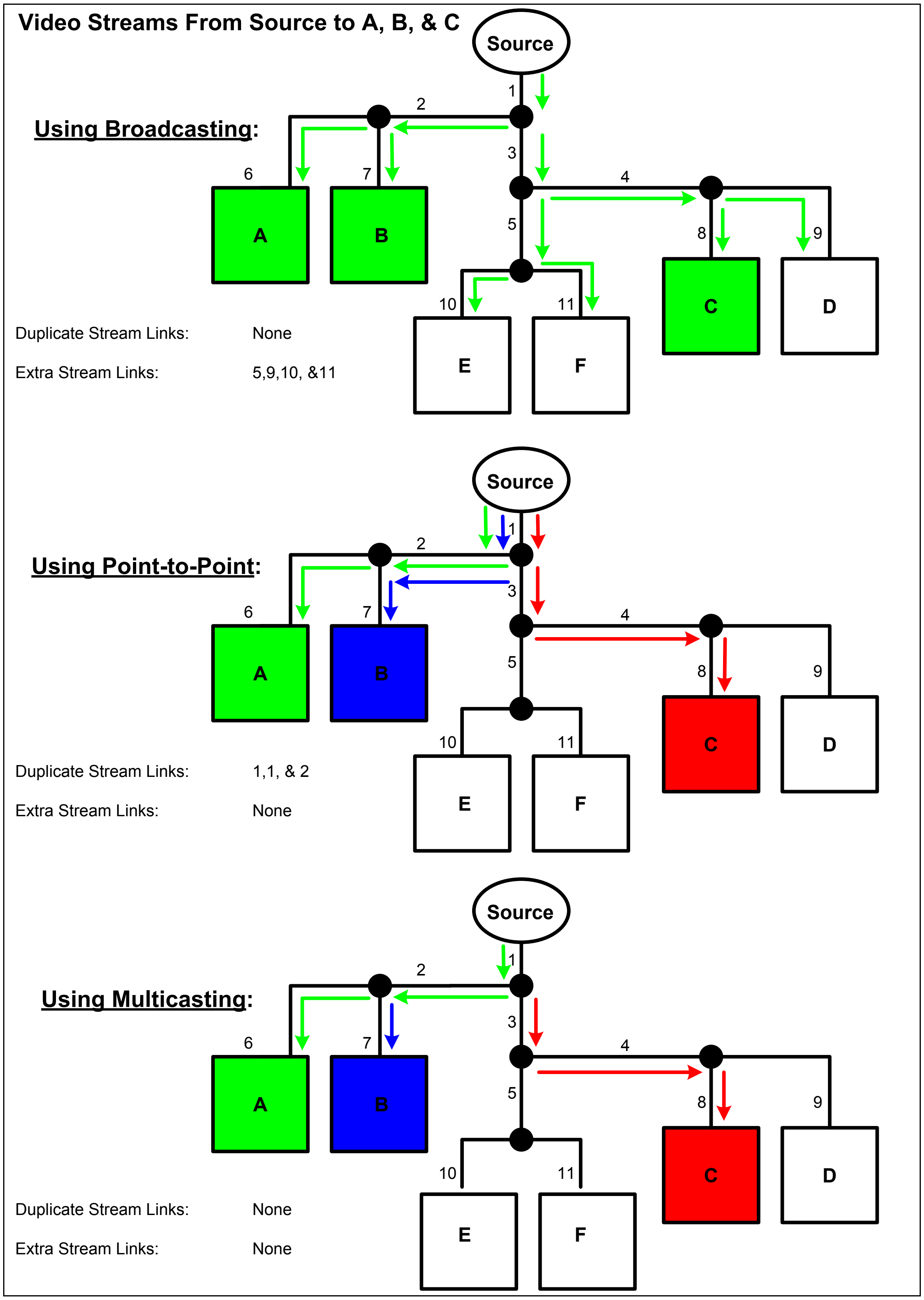
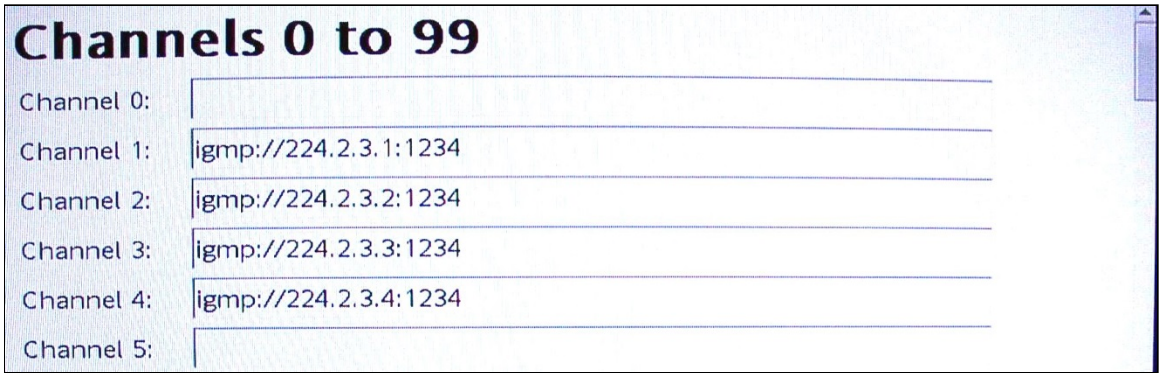

Video Streams From Source to A, B, & C
Using Broadcasting:
Source
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E
F
Duplicate Stream Links: None
Extra Stream Links: 5,9,10, &11
Using Point-to-Point:
Source
Duplicate Stream Links: 1,1, & 2
Extra Stream Links: None
Using Multicasting:
Source
Duplicate Stream Links: None
Extra Stream Links: None

Figure 1.3 Comparisons of Broadcasting, Point-to-Point, and Multicasting Methods

and unnecessary or extra streams. The primary options for video configurations are Broadcasting, Point-to-Point, and Multicasting as depicted in Figure 1.3 with users A, B, and C each requesting to see the same video stream. In the Broadcasting configuration the requested stream is sent to everyone in the network resulting in unnecessary or extra streams on Links 5, 9, 10, and 11 to users D, E, and F. All users are treated identically as indicated by the same green streams. In the Point-to-Point configuration a separate stream is sent to each requesting user, so two duplicate streams result on Link 1 and one duplicate on Link 2. Each requesting user is treated individually as indicated by the different color streams. In the Multicasting configuration common streams are shared and are not sent on to any users not requesting access, so there are no duplicate or unnecessary streams present on the network. It can be thought of as a combination of the positive characteristics of both Broadcasting and Point-to-Point.

To accomplish this multicasting task all of the nodes associated must be running compatible IGMP versions and have all optional parameters properly configured. When IPTV video is transmitted from a multicast source, each TV channel is given a specific IP address and port number. When a user changes to a new channel using his Set Top Box remote control, the STB then sends messages upstream indicating that it no longer needs the current channel and wants to join the new channel. These IGMP messages reflect the IP addresses of the appropriate streams rather than channel numbers. Table 1.2 shows an example of this Channel to IP Address Translation that was a screen shot from an Amino 130 Set Top Box. In the table it is clear for example that Channel 1 is associated with the IGMP format for the stream on IP address 224.2.3.1 using port 1234.

In a typical IPTV multicasting configuration the incoming video streams are first



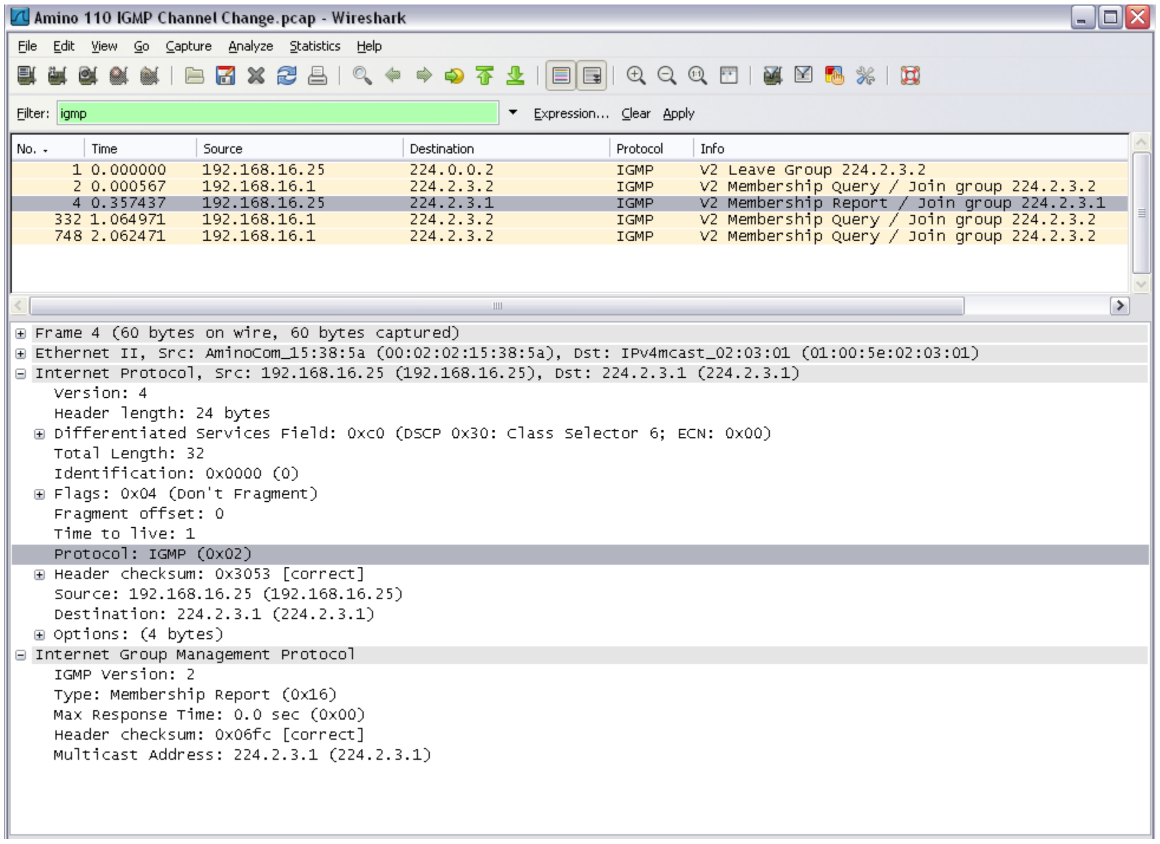
Channels 0 to 99

Channel 0:	
Channel 1:	igmp://224.2.3.1:1234
Channel 2:	igmp://224.2.3.2:1234
Channel 3:	igmp://224.2.3.3:1234
Channel 4:	igmp://224.2.3.4:1234
Channel 5:	

Table 1.2 Sample Channel to IP Address Translation

sent to an IGMP Querying Device, which is frequently providing Layer 3 type functionality. The Querying Device does not pass the videos streams on until it first receives a Join request, and then it only sends the stream on to the particular physical port connected to the requesting party. Multiple ports may request and receive the same video, but currently only ports that have requested a particular stream will receive it to eliminate unnecessary broadcasts to the other ports.

An example of this messaging for a typical channel change from Channel 2 to Channel 1 is shown on Figure 1.4. In this Wireshark IP stream capture the display is filtered to only show the IGMP messages. The user's STB is associated with Source IP address 192.168.16.25 and the IGMP Querying Device has address 192.168.16.1. The first message sent from the STB is to Leave the stream on IP address 224.2.3.2 for Channel 2. The Querying Device sends out several Membership Query Reports to verify that there are no other users still watching that channel before it removes the stream completely. The STB then sends the Join message that was highlighted in Figure 1.4 requesting to receive the IP stream on address 224.2.3.1 which is for Channel 1. In the packet details for this Join message it can be seen that it was from the Amino device with



```
Amino 110 IGMP Channel Change.pcap - Wireshark
Filter: igmp

No.  Time      Source          Destination  Protocol  Info
  1 0.000000   192.168.16.25   224.0.0.2    IGMP      V2 Leave Group 224.2.3.2
  2 0.000567   192.168.16.1    224.2.3.2    IGMP      V2 Membership Query / Join group 224.2.3.2
  4 0.357437   192.168.16.25   224.2.3.1    IGMP      V2 Membership Report / Join group 224.2.3.1
332 1.064971   192.168.16.1    224.2.3.2    IGMP      V2 Membership Query / Join group 224.2.3.2
748 2.062471   192.168.16.1    224.2.3.2    IGMP      V2 Membership Query / Join group 224.2.3.2

⊞ Frame 4 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: AminoCom_15:38:5a (00:02:02:15:38:5a), Dst: IPv4mcast_02:03:01 (01:00:5e:02:03:01)
⊟ Internet Protocol, Src: 192.168.16.25 (192.168.16.25), Dst: 224.2.3.1 (224.2.3.1)
    Version: 4
    Header length: 24 bytes
  ⊞ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)
    Total Length: 32
    Identification: 0x0000 (0)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 1
    Protocol: IGMP (0x02)
  ⊞ Header checksum: 0x3053 [correct]
    Source: 192.168.16.25 (192.168.16.25)
    Destination: 224.2.3.1 (224.2.3.1)
  ⊞ Options: (4 bytes)
⊟ Internet Group Management Protocol
    IGMP Version: 2
    Type: Membership Report (0x16)
    Max Response Time: 0.0 sec (0x00)
    Header checksum: 0x06fc [correct]
    Multicast Address: 224.2.3.1 (224.2.3.1)
```

Figure 1.4 Packet Capture for IPTV Change from Channel 2 to Channel 1

IP version 4 address 192.168.16.25, it is an IGMP version 2 type request, and it is requesting to be a Member or Join multicast stream 224.2.3.1. Note that Leave messages are sent to destination address 224.0.0.2 while Join messages are sent to the stream address of the request. This basic IGMP messaging established the STB as a member of the Channel 1 stream and no longer a member of Channel 2. This process is repeated throughout the network for every IGMP channel change event.

Consider the example configuration shown by Figure 1.5 to examine how the IGMP Querying Device keeps track of the various streams and members. In this example a video source PC is inputting four streams for Channels 1 through 4 to the Layer 3 Switch acting as the Querying Device. The second PC is a DHCP Server to the four Set

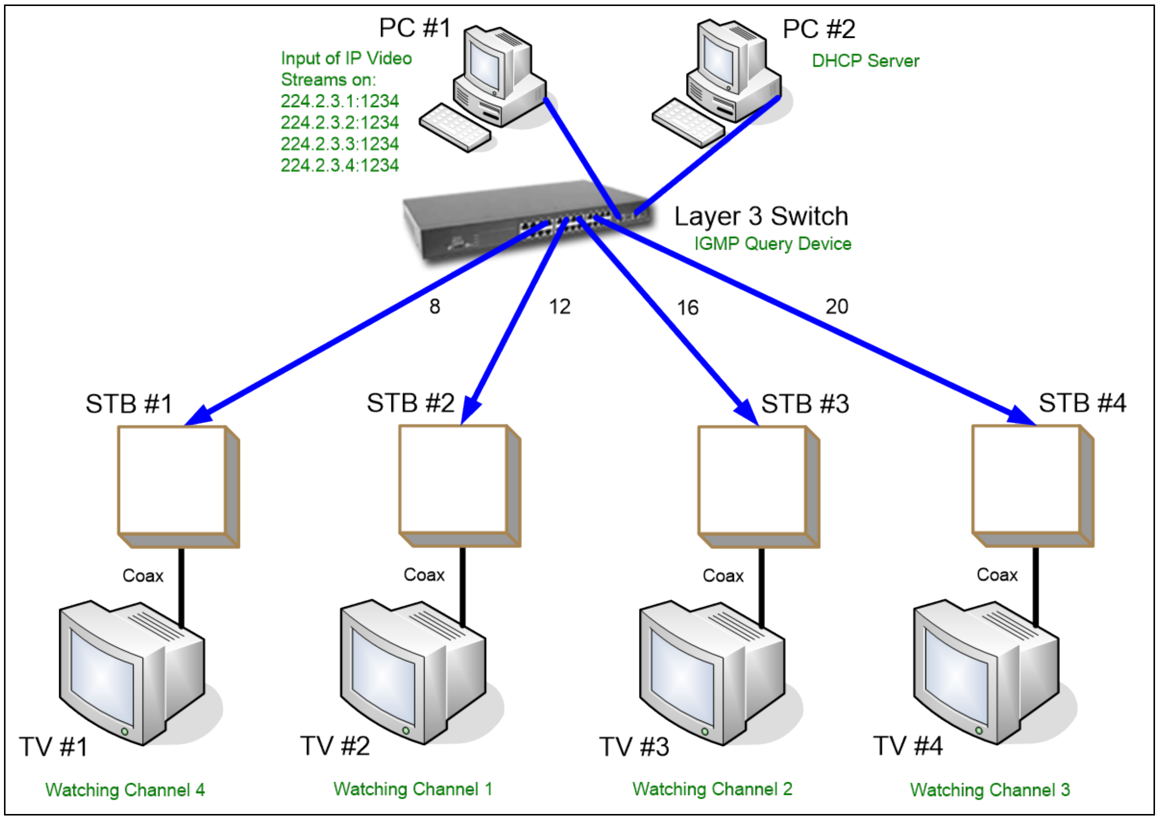
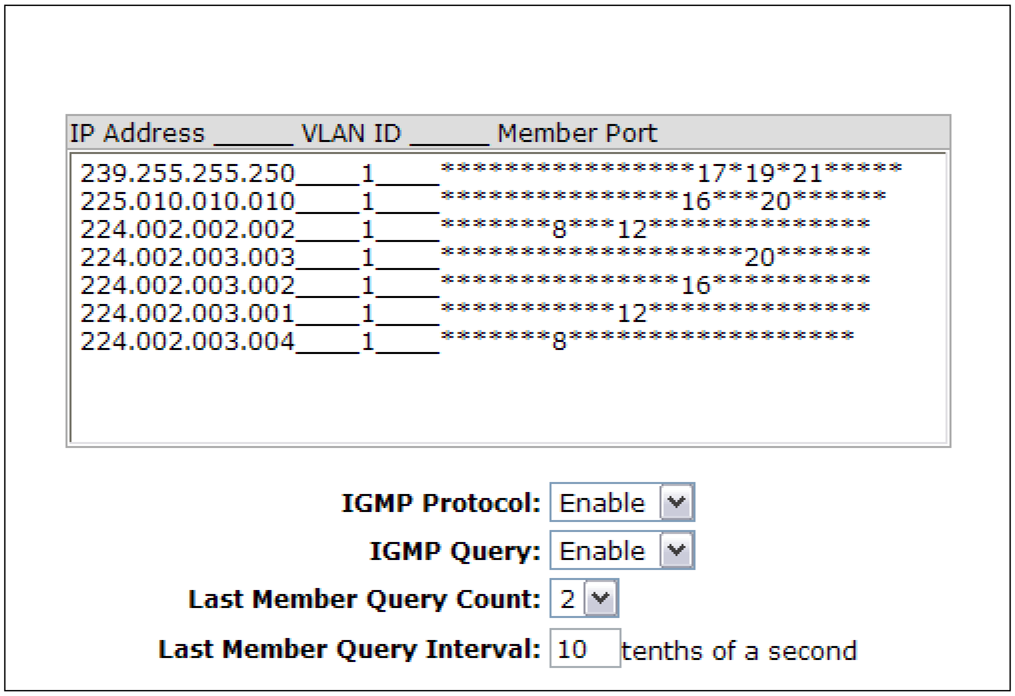


Figure 1.5 Example IGMP Physical Configuration

Top Box/TV pairs connected to the Switch on Ports 8, 12, 16, and 20 for the users.

For the viewing scheme as shown in Figure 1.5 with each STB requesting a different channel, the Querying Device will maintain an IGMP Configuration Table that tracks the members of each IP multicast stream. For the configuration shown in Figure 1.5 the resulting IGMP Configuration Table captured as a screen shot from the Layer 3 Switch is shown on Table 1.3. In this table each multicast stream currently active is listed by its IP address with applicable VLAN details. Each multicast member as established through the IGMP messaging system is listed by its associated physical port connection. Table 1.3 also shows that the Switch is configured for the IGMP Protocol and as the Querying Device with other timing parameters. As expected the table shows that Port 8 is



```
IP Address ______ VLAN ID ______ Member Port
239.255.255.250_____1_____****************17*19*21*****
225.010.010.010_____1_____***************16***20******
224.002.002.002_____1_____*******8***12**************
224.002.003.003_____1_____*******************20******
224.002.003.002_____1_____***************16**********
224.002.003.001_____1_____***********12**************
224.002.003.004_____1_____*******8******************
```

IGMP Protocol: Enable
IGMP Query: Enable
Last Member Query Count: 2
Last Member Query Interval: 10 tenths of a second

Table 1.3 Example IGMP Configuration Table

watching Channel 4 on stream 224.2.3.4 and likewise for the other active ports. The first three multicast IP streams are management streams not associated with actual IPTV video. In a similar fashion other devices on the network may also track the IGMP multicasting traffic in a process known as IGMP Snooping. If there were a Layer 2 Switch or a Gateway between the Layer 3 Switch and the Set Top Boxes of this example, they would optimally be configured for IGMP Snooping functionality to listen and track the multicasting through them. That would allow them to only send on streams to ports that requested it just like the IGMP Querying Device. The primary difference is that in order for the IPTV video to be passed down the IGMP messaging must first be established with the main Querying Device. After that all of the IGMP devices properly configured will maintain optimal multicasting of the streams.

The IGMP functions were first introduced in Version 1 with Join messages but no

Leave messages. Instead a timeout system was used to end streams that were no longer needed [23]. As traffic levels increased it became beneficial to quickly pull down streams that are no longer active, so Version 2 introduced the Leave messaging system. Version 3 then was introduced to allow specification of hosts for receiving and blocking which is useful in the prevention of denial of service attacks and network optimization.

1.4 Need for Short Channel Change Times

Customer expectations are the primary justification for the continued research into shortening IPTV channel change times. Previous researchers in this area have consistently put forth the case that excessive CCTs will be prohibitive with respect to customer satisfaction and IPTV growth [1][2][5][21-22][24-48]. As noted earlier this has not been a significant issue in the past, and with the emergence of IPTV services the CCT has become a major factor in determining the Quality of Experience (QoE) [2][22][37]. The QoE is a relatively new concept that stemmed out of the well known Quality of Service (QoS) analysis with respect to IPTV. While the QoS typically is based on quantitative measurable data, the QoE frequently involves customer impressions as determined by survey results. Results of these customer surveys conclude that viewers rated a CCT of one second as only “fair”, and that this process is a key component of overall QoE [32][33].

Customer expectations have conflicted with the reality of actual CCTs during these early deployment years of IPTV. Clearly the exact times are based on the deployed configurations, equipment involved, broadcast scope, and parameter settings. Recently published results of change times indicate that the numerical values differ substantially based on the specifics, but a representative sample of these results is shown on Table 1.4. From this data we see a considerable range of times reported in actual field cases that are well in excess of customer expectations. Through tradeoffs with minimum customer expectations versus the reality of IPTV field issues, the general consensus for an acceptable CCT target appears to be two seconds or better [1][22][29-31][33][38-40].

Channel Change Times	Source
1.9 - 4.8 sec	Cisco [41]
1.5 - 4 sec	Wikipedia [42]
3 sec	Motorola [30]
0.9 - 70 sec	*Agilent Technologies [2]*

Table 1.4 Sample Reports of Channel Change Times

Given this two second maximum CCT target, the next question becomes how quickly customers will give up on IPTV service if the target is not met. The answer to this question may in fact be quite complex depending on what other TV subscription options they have available to them at that particular moment. Considering the overall benefits of IPTV, the case can be made that customers may soften their stance on CCTs a bit if other larger advantages are present. If the customer can merge data or informational services onto their routine TV broadcasts, then they may expect some minor delays. An example of this would be running a web video conference call or performing online shopping in the corner of their TV screen while watching a network broadcast simultaneously. These added services have forced the Set Top Box manufacturers to develop units that can effectively handle these different voice, video, and data streams through the use of complex middleware. But by introducing this middleware application the average CCT is adversely affected even when the services are not fully utilized. Even with a softened position on change times, customers will still require reasonably seamless performance to meet minimum expectations. This analysis indicates how drivers that add new functionality and advances in one area can further increase the need to keep CCTs sufficiently low to exceed minimum customer QoE.

1.5 Sources of Delay

The existing research base on this topic shows a wide disparity in the descriptions and values associated with CCT delays. Various researchers have categorized and documented the sub-components of the channel change process in methods that were suited to meet the particular needs of their study. Consequently no standard description about the sources of delay in the change time process exists, but by taking a cumulative view of the existing research one can find trends and similarities concerning these delays. Table 1.5 reflects a consensus view of the conventional sources of delay as compiled from the wide body of existing research [1][5][21-22][24][26-30][34][38-39][43-52]. Table 1.5 goes on to show how these sources will be classified for this research. The reasons for classifying the conventional sources of delay are the following:

- It allows for a convenient description of multiple delay sources simultaneously.
- Classifications reflect the overall process from which each delay is a component.
- Classifications for this research are similar to other researcher's schemes.
- This research improves the times associated with the overall classifications rather than the individual sources of delay.

Conventional Descriptions	Conventional Time Delays	Classification for Research	Applicable Delays by Classification
User Selects New Channel	Occurs @ 0 sec	Command Processing	≅ 40 - 50 msec
STB Processes Selection & Stops Decoding Current Channel	≅ 10 msec		
STB Sends Leave Current Channel Message	10 - 20 msec		
STB Sends Join New Channel Message	≅ 20 msec		
Network Latency or Delay (One Way)	30 - 70 msec if Optimal > 1 sec for Complex Configs (600 msec for testing model)	Network Delay	≅ 600 msec
Querying Device Receives and Processes IGMP Requests	≅ 10 msec	IGMP Processing	≅ 20 msec
Encoder Delay and Rate Control	≅ 5 msec		
Aggregation and Streamer Delay for Multiplexing & Rate Shaping	≅ 5 msec		
Network Latency or Delay (One Way)	30 - 70 msec if Optimal > 1 sec for Complex Configs (600 msec for testing model)	Network Delay	≅ 600 msec
De-Jitter Buffering	≅ 300 msec	STB Processing	≅ 1.775 - 2.9 sec
Wait for PAT & PMT	≅ 125 msec		
ECM Delay (If Required)	0 or ≅ 125 msec if used		
Wait for RAP (I-Frame or IDR)	250 msec - 2 sec (300 msec typical)		
MPEG Buffering	1 - 2 sec		
Decode Content & Output Video	≅ 50 msec		
Monitor Response Time & Video Scaling	≅ 10 msec	Display	≅ 10 msec
Cummulative Delay	**≅ 1.855 - 6.68 sec**		**≅ 3.045 - 4.18 sec**

Table 1.5 Consensus View of the Conventional Sources of Delay Followed by the Classification Scheme Used for the Research

While a more detailed description of the individual sources of delay will be given shortly, there are a few issues relating to Table 1.5 that need immediate clarification. The cumulative delay times differ significantly between the Conventional Time Delays and the Applicable Delays by Classification due to how the Network Latency and Wait for RAP items are handled. The Conventional Time Delays reflects the full range of values for these items as documented by the existing resources cited above, but the Applicable Delays by Classification reflects values for these items that are pertinent to the methodology of this research.

The conventional view of the one way Network Latency indicates a 30 to 70 millisecond delay if optimally configured, but this configuration typically indicates an edge server located very near the end user. Since this configuration is not the primary considered case for MDUs and other applications covered by this research, then these optimal values are not considered applicable for the purposes of this analysis. When optimal configurations are not present the conventional view of one way Network Latency shows that this value widely varies from around 50 milliseconds to over 1 second, but the Applicable Delays by Classification reflects a value of 600 milliseconds for the testing purposes of this research in line with previous research into predictive tuning [1][7]. It is understood that the one way Network Latency will vary widely from this 600 millisecond value in real-world applications even when optimal configurations are not present, but this distinction will be made towards the end of this research in the sections handling the potential improvements of this multicast caching technique.

Similarly the Wait for RAP items vary from 250 milliseconds to 2 seconds for the conventional view depending on how the I-Frames are implemented, but the

Applicable Delays by Classification reflect the Wait for RAP item as approximately 300 milliseconds in line with the values typically seen in practice [21-22][32][40-42][48]. The result of these differences in handling these values is the reason for the variance between the conventional range of 1.855 to 6.68 seconds and the applicable range of 3.045 to 4.18 seconds. Note that this applicable range is in line with the industry reported delay times from widespread actual deployments documented previously in Table 1.4.

A closer inspection of the sources of delay will be made by looking at each of the Conventional Descriptions found in Table 1.5. The first item begins the process with the User Selects New Channel at time zero typically using a remote control device. This selection is received followed by the STB Processes Selection and Stops Decoding Current Channel. When the STB receives the new channel request it takes approximately 10 milliseconds to process this request and stop decoding the current channel. This is followed by another 10 to 20 millisecond delay until the STB sends Leave Current Channel Message upstream as covered earlier in the multicasting discussion. Then finally the STB sends Join New Channel Message about 20 milliseconds later to the Querying Device to initiate the channel change by the network. All of these processes are performed by the STB and until the Querying Device receives the Join message then no action is taken with respect to the new channel. For these reasons all of these processes are frequently grouped together in the classification of Command Processing to take a combined delay of 40 to 50 milliseconds.

Once the Join message is sent out by the STB it takes a period of time corresponding to the one way Network Latency or Delay time before it is received by the Querying Device upstream. As discussed previously this delay period ranges from a minimum of 30 milliseconds to over 1 second in duration depending on the configuration, the devices involved, and what other activity is taking place at that time. In order for this delay to be in the minimal range then an edge server located near the end user along with a high performance configuration and a minimal number of intermediate devices is typically required. A valid assumption for networks with a high QoS is to treat one way Network Latency values identically in both transmit and receive directions and constant over time [24]. For this application the item is classified as the Network Delay with the fixed value of 600 milliseconds only being applicable for illustrating the effects of multicast caching with a recognition that actual Network Delays will vary widely.

When a Join message arrives then it takes about 10 milliseconds typically for the Querying Device Receives and Processes IGMP Request function to take place. The encoding and rate control functions followed by the aggregation, streamer, and rate shaping processes yield the next two delays each of around 5 milliseconds. These functions are all carried out by the Querying Device sequentially, so they have been grouped together into the IGMP Processing classification with a typical group delay of about 20 milliseconds. At the end of this process the first packet of video for the new channel selected is being sent out followed by the full video stream.

Another point of note is that these individual sources of delay could also be further divided into even more granularity as needed. For instance if a Layer 3 device manufacturer is looking to optimize the overall IGMP Processing time, they would certainly look at these sources much closer and come up with finer characterizations of the process to include every sub-task such as TCP encapsulation followed by IP encapsulation for instance. Since this research is not geared toward the physical product improvement of STBs or Querying Devices, then we are able to look at them solely as classification groups with the Command Processing for the STB functions and IGMP Processing for the Querying Device functions.

The IGMP Processing is followed by another one way Network Delay cycle for the video packets to make the trip from the Querying Device to the STB. At the end of this second Network Delay the first video packets begin arriving from the Querying Device to the STB for the new channel selected.

With the video packets arriving at the user's set top box the next series of items is grouped into the STB Processing classification. This begins with the De-Jitter Buffering to account for variations in the packet arrival times and under-run conditions adding approximately 300 milliseconds of delay [30][41-42]. The next processing is on the Program Allocation Table (PAT) to provide identification codes and system information for the broadcast programming that is useful for guide functions such as listing the available channels for selection. This is followed by processing the Program Map Table (PMT) which contains information that describes the components of the applicable channel, such as providing the identifiers for the separate video and audio streams of an ongoing channel. These Wait for PAT and PMT functions performed by the STB add a combined delay of about 125 milliseconds [41].

Whenever programming requires encryption, such as with pay channels, then a Conditional Access System is used to prevent unauthorized access. This is done through the use of Entitlement Control Messages (ECM) with embedded keys in them necessary to correctly decrypt the video stream [41][48-50]. This process results in an ECM Delay of approximately 125 milliseconds only when encrypted channels are selected.

As previously discussed the Wait for RAP delay ranges from 250 milliseconds up to 2 seconds depending on how the I-Framing is configured. A typical configuration using a GOP of 15 frames streaming at 25 frames per second results in an average 300 millisecond RAP delay. Since the I-Frames contain the complete picture of a particular frame that is then edited by P and B Frames for subsequent frames, then they are significantly larger in size [51]. So to reduce the Random Access Point wait times by inserting more I-Frames significantly increases the bandwidth required for a particular channel. For instance I-Framing at 0.25 seconds roughly doubles the required channel bandwidth over I-Framing at 2 second intervals [1][21]. This tradeoff between channel bandwidth versus shorter channel change times results in various providers setting up their broadcasts using significantly different I-Framing rates.

As the STB completes the list of items from above, it is now ready to begin handling the actual video stream. This requires the STB to first perform an MPEG Buffering step to allow uninterrupted video downstream to the monitor. Another delay is introduced of approximately 1 to 2 seconds while the STB waits for this buffer to fill [5][21][41-42]. The buffer design and processing power of each STB significantly influences this period. The presence of HD video further requires a larger amount of buffering and slightly higher delays. The last STB delay of another 50 milliseconds is introduced in order to Decode Content and Output Video transforming the stream of packets into the video output format required such as High Definition Multimedia Interface (HDMI). All of these STB functions associated with receiving and preparing the incoming video stream are grouped together for the purposes of this research into the single classification of STB Processing with an associated delay of 1.775 to 2.9 seconds.

Again a STB manufacturer would analyze these functions to a finer granularity with many additional sub-functions included such as managing the Program Clock References and Sequence Headers for timing information and frame rates. Additional STB functions like error concealment can add more delays beyond those specified here. Since this research is not aimed at reducing these specific delays then this section is simplified by the singular classification of STB Processing.

Following the STB Processing the monitor or television introduces the final delay of the Monitor Response Time and Video Scaling. This relatively minor monitor delay of approximately 10 milliseconds allows for handling the various aspect ratios and other formatting associated with the end user's viewing device [30][52].

While the analysis of the individual sources of delay provides a more complete picture of an IPTV channel change, the simplified classifications will be used going forward in this research project for clarity and convenience.

1.6 Objectives of Multicast Caching

Referring back to the last section the simplified classification scheme indicates the following delays associated with a typical IPTV channel change:

- Command Processing
- Network Delay (upstream for new Join request)
- IGMP Processing
- Network Delay (downstream for new video channel)
- STB Processing
- Display

It is the objective of multicast caching to predict to a reasonably high degree of success the channels that are most likely to be changed to next and optimally stream them prior to an upcoming channel change. If this can be successfully accomplished before a user's new channel request input then this new channel would already be streaming to the user. With the delay classifications in mind, this event would differ from the typical case after the Command Processing function since the two Network Delay periods and the IGMP Processing time are minimized as the new channel's video packet stream is already arriving. Once the STB finishes its Command Processing then it is ready to start receiving the new channel as soon as the video arrives. This is the case since multicasting has the Join and Leave functions and no true handshaking occurs.

For all cases where the next channel selection is present in the multicast cache and is therefore being streamed in advance, then theoretically this new system would approximate the following delay classification scheme:

- Command Processing
- STB Processing
- Display

If this can be shown to be a viable system, then this streamlined method would improve the overall CCT by approximately the following time reduction:

CCT Reduction = (2 x Network Delay) + (IGMP Processing)

The difficulties associated with managing time weighted probabilities, bandwidth availability, and optimizing cache effectiveness are the most problematic aspects of this proposed method. Given these significant potential reductions of CCT, it is the objective of this research to put forth the actual means through which these multicast caching problems can be handled effectively and provide a detailed analysis of the workability of the new system.

1.7 Additive Properties of Multiple Methods

An effective multicast caching system still leaves the CCT being dictated by the Command Processing, STB Processing, and Display classifications. Referring back to Table 1.5 it is clear the STB Processing is by far the longest and most significant factor of these three remaining classifications. It is equally important to note that the multicast caching system proposed has no direct impact on the STB Processing delay. Therefore any other methods that independently result in a delay of the STB Processing period would provide additional time reductions to the overall CCT beyond those already potentially achieved by multicast caching.

This additive property of utilizing multiple methods for CCT reduction is easily demonstrated through two ongoing areas of research:

- I-Frame Management
- Buffering Techniques

As noted in the Sources of Delay section, the two most significant factors of the STB Processing delay were the Wait for RAP (I-Frame) and MPEG Buffering items. As will be covered in Chapter II, both of these factors have been researched heavily with many potential improvements proposed. Examples of these are a dynamic I-Framing system that sends an increased number of I-Frames immediately following a channel change and improved physical processors or buffering techniques to reduce the STB MPEG Buffering times. Both of these improvements could be implemented simultaneously with the proposed multicast caching system to effectively reduce the overall CCT by the sum of each of these individual time reductions.

There are a few other proposed methods for CCT reduction that do conflict with or are irrelevant to the multicast caching proposal, and they would not provide additive benefits. However, the methods that do not conflict with multicast caching, particularly those geared toward reducing the STB Processing times, can be implemented in addition to this proposal. That provides the further benefit to this proposed method that it is not primarily an "either/or" proposal that requires a firm decision up front between one particular method versus another in most cases.

1.8 Wasted Bandwidth Fallacy

The concept of sending additional IPTV channels to an end user to reduce the next CCT if one of them is selected next has been referred to as Predictive Tuning, Pre-Buffering, Pre-Selection, or Pre-Fetching in the existing literature [21][25][26][28] [29][34]. These existing approaches will be covered in more detail in the next chapter, but the argument made against this approach is that it wastes bandwidth by sending extra channels that are not in use [21][26]. In the research that supports this approach, no mention or response to this "wasted bandwidth" argument has been found. This research proposal further supports the use of sending these extra channels in certain configurations, but an effort will be made up front to respond directly to the typical criticism of the approach.

It is true that there are many areas of communications networks that frequently have strong limitations on bandwidth usage, and it is the responsibility of network planners to design and operate the network to make the best use of available bandwidth. For this reason networks are frequently “oversubscribed” where the total bandwidth available would not be able to meet the maximum requirements of all users simultaneously. Instead statistical approaches are used in the design to ensure that under most conditions all likely users will be satisfied, but that occasionally high usage conditions will result in some customers not getting their expected service once the defined peak limit has been passed. But what happens during times of minimal network usage or even during normal operating conditions that are well below the peak limit? For most networks, even ones that are heavily oversubscribed, it is not unusual to have significant amounts of idle bandwidth throughout many portions of the network until high peak conditions return.

An even more important point about bandwidth in most networks is that certain layers or functional areas of the network may have significantly differing availability limitations than other areas. It is in connection to this key point that the Multi-Dwelling Unit configuration is important, because it has such a nice potential fit for IPTV services. Reconsider Table 1.1 assuming that the links between the Layer 2 device and the Gateways are configured in the typical 100 Mbps manner, and this entire topic can be explained through the analysis of one of these access links. With the present state of communications technology even users of triple or quadruple play services rarely, if ever, approach the limits of a typical 100 Mbps link and future deployments may go even further in this direction with 1 Gbps links present. Comcast broadband metering services in 2009 indicated that the average residential user consumes 2 to 4 Gigabytes per month and that highly technical “power” users that frequently telecommute from home only raise the rate to 35 Gigabytes per month [53]. Converting this heavy user to an average Mbps consumption rate can be done as follows:

(35 GB/mo)*(8 b/B)*(12 mo/yr)*(1 yr/365 day)*(1 day/86,400 sec) = 106,545 bits/sec

Comcast's average residential user can be similarly converted to yield approximately a 6 to 12 Kbps mean consumption rate. Clearly during peak usage this goes up into the Megabit per second range, but very often this rate is offset by zero usage to yield the low average consumption rate. These surprisingly light usage rates are supported by Cisco's 2010 estimate for North American home IP traffic rates of approximately 142 Kbps, on average, running through its platforms [54]. A 2009 University of California, San Diego study indicates that the total information flow coming to the average American person is 34 Gigabytes daily from all sources (newspapers, TV, radio, internet, etc.) [55]. This study converted written mediums, traditional audio and video sources, plus internet consumption into a single digitized equivalent for overall information. Converting this to bits per second yields a mean information flow to the average U.S. person of 3.15 Mbps from all sources. Further insights into this usage can be gained by the fact that providers customarily limit broadband user bandwidth for traditional data to the range of 5 to 25 Mbps downstream and even less upstream.

The low bit rates associated with voice services play an extremely small role in this overall usage. Given that the service provider will almost certainly limit the customer's downstream data usage, assume for a moment that the customer uses in the range of 10 to 25 Mbps for traditional data service. That still leaves 75 to 90 Mbps available on the link up to this point in the analysis. Consider that 100 Mbps links cannot actually support exactly that rate of traffic continuously, then throw in the minor voice requirement and a large safety factor, and it quickly becomes apparent that this MDU style customer will still have somewhere in the range of 50 to 75 Mbps available for continuous video service as illustrated in Table 1.6.

Typical MDU Link Bandwidth Available	100 Mbps
Data Service	10 - 25 Mbps
Overhead and Safety Margin	15 - 25 Mbps
Voice Service	<< 1 Mbps
Remainder Available for Video Services	**≅ 50 - 75 Mbps**

Table 1.6 Bandwidth Composition Rates for a Sample MDU Customer

With continuously available video bandwidth in the range of 50 to 75 Mbps for a sample MDU link and video streams in the range of 1 to 20 Mbps from Table 1.1, then it follows that if only one channel is being streamed then there will frequently be a significant amount of idle bandwidth on the link. Consider typical SD streams running at 2 or 4 Mbps and HD streams at 9 or 18 Mbps, there may in fact be the ability to run many simultaneous streams in preparation for the next channel change. Given the savings to CCTs of approximately two times the one way Network Delay plus the IGMP Processing time, then this significant time savings can be potentially achieved with little more than the use of otherwise idle traffic flow after the processing features are implemented.

Another key factor to consider is the relative ease at which traffic can be prioritized by Ethernet architecture. Through the use of the three bit precedence field, the four bit Type of Service field, or sometimes through specific Virtual LAN configurations it is common for specific types of packets to receive higher or lower priority treatment with respect to other flowing packets [56]. One recommendation this proposal will make is to have all of the packets associated with the multicast cache of video streams tagged with a low priority, so that these efforts to speed up the CCT would have negligible impact on higher priority existing voice, video, or data packet routing. Under this situation even if a quick burst of higher priority traffic, such as a large data transaction, were to take place while a lower priority cache of multicast video was streaming, then the higher priority data traffic would simply jump to the front of the line. If a higher traffic condition well beyond the normal range available for video services persists then the system would recognize this and reprocess the multicast cache to take advantage only of the bandwidth currently available for video.

Noting that most networks have a significant amount of idle traffic periods, that MDU style architectures have extra capacity built in, that current data usage while growing is still relatively minor, and that prioritization of additional video packets would prevent harmful impacts to existing traffic it is the position of this research proposal that a multicast caching system could be put in place without any significant negative impact. Anytime a portion of a network experiences idle traffic conditions should be considered a lost opportunity for passing some form of communication which can never be regained after the point at which it is lost. Put more bluntly it appears that *not* using idle traffic for low priority operations such as the one being proposed here would actually result in "wasted bandwidth".

1.9 Contributions of this Research

The primary contributions of this research are listed below and will be explained in more detail throughout this document.

- A unique multicast caching system with specific algorithms is presented, with a focus on the MDU style architecture.
- Laboratory testing is conducted to yield a validation of the actual reductions made to channel change times by this process.
- Probability simulations are made to reflect the potential effectiveness of the various proposed algorithms under a wide variety of conditions.
- The use of packet prioritization on the cache of multicast video streams is proposed here to significantly improve the performance of this system over existing methods.
- This research offers a network-based solution over other client-based methods to provide several inherent advantages.
- A detailed analysis of the primary implementation considerations is presented to aid in the actual deployment of this approach.
- A listing of other applications that could benefit from this proposed method is presented along with several areas of further research that could provide future improvements to this field.
- A secondary level contribution is made by the alternative view presented to the traditional "wasted bandwidth" mindset that compels others to start thinking more along the lines of use it or lose it with respect to network bandwidth.

These contributions begin with the unique algorithms associated with this proposal and then go on to offer supporting research from laboratory testing and simulations that provide evidence about the effectiveness of the approach. The novel uses of exponential time weighting to the probability process, the concept of channel bandwidth penalization, and the use of packet prioritization for the cached video streams are unique in this area of research. The network approach alone is a significant departure from existing research that concentrates on client methods, and this new view provides distinct advantages that will become evident as this research is fully presented. The efforts made by this research at identifying and analyzing the main implementation considerations is rarely seen with existing technical proposals in this field that tend to concentrate solely on the mechanics of the new methodology and leave deployment issues up in the air. The use it or lose it mindset proposed here is a departure both in terms of a better way of looking at existing network bandwidth and as a preemptive rebuttal to expected criticisms of the proposal. All of these individual advancements taken in their entirety make up the full contributions of this research.

CHAPTER II

REVIEW OF EXISTING LITERATURE

In reviewing through the existing research and literature on recent advancements to improve IPTV channel change times it was evident that the methods documented generally fell into one of three main groups:

- Optimizing the configurations and parameters of existing technology.
- Proposals for improved methods or processes based on new research.
- Presentations on new hardware developed by manufacturers.

The existing solutions for improved CCT performance from these main groups will be presented in the following sections based on the general technical area of the overall process that was targeted for improvement.

2.1 Edge Servers

In Chapter I an analysis of IGMP message flows and the primary sources of delay associated with CCTs for IPTV applications was presented. It was shown that the Network Delay between the HG and the LHR with IGMP Querying capability is included twice in the CCT to account for the delay required for packets to flow in both directions. A smaller number of hops and reduced complexity on this path directly reduces CCT. For this reason it was documented in [21] that placing an Edge Server as close as possible to the end user is a well verified method for improving CCTs. The Edge Server would handle the IGMP Querying functions and the channels offered would need to be continuously streamed into it to allow for rapid multicasting downstream when requested. This effectively moves the LHR closer to the HG to reduce the Network Delay times. This method proposes using more resources of existing technology to optimize the physical configuration in order to reduced CCTs.

While certain applications definitely fit the profile for the use of this method, the primary drawbacks to its use are the extra expenses associated with more equipment and the additional bandwidth required to have all channels presented continuously down to its edge location. Consider an example case of a small suburban apartment complex with 10 units located near the outside of a large sprawling city. Assume that all of a provider’s 150 offered channels are available at a downtown Head End location with an average bandwidth size of 6 Mbps. To continuously stream all of the 150 channels down to an Edge Server on or near the apartment complex would require a dedicated 900 Mbps link just to serve the video needs of 10 units. Even if all 10 units had two TVs watching completely different channels it would only result in a maximum of 120 Mbps of video as compared to the 900 Mbps rate required for the Edge Server. Add this increased link expense to the actual hardware, installation, and maintenance costs associated with the Edge Server and it would seem unlikely that the service provider would frequently be able to offer competitive pricing for these types of customers. However, if this apartment complex were extremely large or located near other complexes then the provider might well be justified in providing the necessary link sizing and equipment to support an Edge Server at the complex. The factors to be analyzed for Edge Server deployment are the number and type of users, proximity of other users, bandwidth required for all channels offered, link and equipment costs, alternative options available, local pricing, and other market considerations.

This method has technical merit for certain applications, but it must be evaluated on a project-by-project basis to determine the cost effectiveness of its use. Clearly there are many applications and configurations that do not lend themselves to the use of Edge Servers.

2.2 I-Frame Management

Previous discussions showed the importance and tradeoffs associated with the I-Frame rate between the Wait for RAP delay and the channel bandwidth. Due to this importance the management of I-Frames has been a major target of previous research into reducing CCTs. The proposals for I-Frame management generally involve changing the I-Frame rate dynamically, use of enhanced server and STB pairs, or the use of a specialized I-Frame server.

New algorithms have been developed by researchers to optimize the CCT process by dynamically increasing the I-Frame rate for a quick period of time immediately after a channel change request is received [5][21][46-47]. Assuming the effectiveness of these new algorithms is substantiated, field deployment would be made in the form of new LHRs that would dynamically adjust the I-Frame rates. The primary advantage of this technology is the potential effectiveness in reducing the random access point delays and that it is compatible with existing standards and STBs already in widespread use. The main drawback is the increased complexity associated with recalculating and inserting additional I-Frames into existing streams that the LHR would need to undertake to dynamically adjust the I-Frame rate.

By coupling the server functions of the LHR with the STB functions, some manufacturers have been able to offer specialized combinations that optimize CCT for IPTV service [41][48]. The very fact that these specialized combinations are being undertaken by manufacturers such as Cisco underlines the importance of reducing CCTs to achieve increased IPTV markets. Some of the proprietary techniques utilized by these combined solutions involve the use of I-Frame caching, unicast bursting, and dynamic protocol swapping. Unicast bursting is a Cisco technique that sends a quick burst of reference data in unicast fashion to accelerate CCTs, and it is a potential technique under consideration for retransmission applications. Dynamic protocol swapping typically involves the use of RTP immediately after channel changes and then it moves back to UDP transmission later on. These proprietary techniques typically require specialized firmware that is capable of handling the dynamic processes. These solutions offer the advantage of specialized equipment to optimize aspects of IPTV applications while not affecting higher level network operations. The disadvantages are the increased cost and complexity of the proposal and that it is not backwards compatible with existing STB technology. To be effective in the field the solution should be deployed as a combined system rather than using individual components here and there.

Another implementation of dynamic I-Frame management and specialized IPTV equipment is made through the use of I-Frame Servers [57]. This technique has not been adopted for widespread use due to the insertion of a standalone device that operates in parallel with the existing configuration to optimize the rate at which I-Frames are sent.

From a technical standpoint the current state of a user's IPTV viewing is analyzed to determine the optimal I-Frame rate of individual channels. Proposals look at past behavior to determine if a user is currently surfing channels up and down, swapping back and forth between specific channels, or statically watching a single channel. With this knowledge the I-Frame Server will prepare bursts of I-Frames for select channels to be rapidly inserted into the video stream when an appropriate channel selection is received. While the technique offers similar improved performance to dynamic I-Frame management techniques, there is no inherent reason why LHRs cannot incorporate this same functionality themselves to reduce the need for specialized components to be inserted into the system.

2.3 Buffering Techniques

Improvements to the buffering processes involved with IPTV have consistently been undertaken since the delays associated with buffering constitute such a major portion of the CCT. While some researchers have targeted the server side, most of the major gains have come from improvements to STB buffering techniques in recent years. As previously discussed the STB undertakes buffering throughout the de-jittering, PAT, PMT, ECM, RAP, MPEG, and error concealment processes prior to finally decoding the video output. Recent STB buffering improvements have been made through the use of improved hardware, architecture, and algorithms [21][24][41][48].

Some of the STB delays are built into the stream and have to be improved elsewhere, such as delays waiting for I-Frames, PATs, and PMTs to be received. However, even in these cases improved STB processing of these items can make slight improvements to CCTs [21][24]. The use of larger and faster chips with embedded architectural improvements have enabled STB manufacturers to steadily cut down on the significant delays associated with MPEG buffering while still avoiding under-run situations where relatively minor delays upstream can result in a temporary loss of video output [41][48]. In addition to offering speed improvements, the new STBs also offer improved performance for a higher overall customer QoE, such as reducing channel change jerkiness or residual artifacts [41]. Problems associated with audio-to-video synchronization (A/V Sync) stem from the facts that the audio streams tend to be much faster and easier to decode than video and that these two processes are frequently performed on different integrated chips that rarely communicate with each other [58]. This A/V Sync problem is often referred to as the Lip Sync issue where a person's verbal sounds do not match up with their visual facial activity, and it is an entire complex field of study in itself [59-61]. The important point for CCTs is that audio is generally buffered until appropriate timing information correlates it back into its proper position with respect to the lagging video stream.

The improved equipment associated with the size and speed of IPTV buffering processes has made significant improvements to CCTs and QoE benchmarks. The primary disadvantages other than the cost of these improvements stem from the different interpretations of the various standards leading to different implementation methods [58-59]. For this reason engineers are forced to look at the overall IPTV system closely to verify the compatibility of all components in the path.

2.4 Dynamic Video Coding

Several methods to perform dynamic video coding exist and have been utilized to reduce CCTs with varying degrees of success. The most notable of these methods are Scalable Video Coding (SVC), tune-in streams, and tune-in servers. Rate control and shaping functions have to be considered as well for optimal dynamic video coding.

The Joint Video Team (JVT) is a partnership made up of members of the primary standards bodies in this area, which are the MPEG team from the International Organization for Standardization (ISO) and the Video Coding Experts Group (VCEG) from the International Telecommunication Union (ITU). In November 2007 Appendix G of the H.264 Advanced Video Coding (AVC) specification on MPEG-4 video was completed to provide flexible standards for the implementation of SVC [62]. To meet the pressing needs of the emerging handheld device customers, the JVT formalized methods to provide acceptable quality to users based on the specific device involved by considering its resolution, frame rate, and bit rate. SVC is a layered approach to video streaming where a base layer is defined along with one or more enhancement layers [1][16][21][62]. For bandwidth limited situations only the low quality base layer may be sent without the enhancement layers to reduce the overall bit rate. Researchers have concluded that this process can be used dynamically for a short period of time immediately after a channel change has been requested. An SVC capable server can reduce the frame and bit rates temporarily for the new channel requested by sending only the base layer. The user would then see a lower quality image of the new channel quicker than waiting for the higher quality image to decode due to the more rapid transmission of IDRs for the smaller base layer. After a few seconds the server would then transition over to the high quality video by transmitting the base layer along with all enhancement layers.

The dynamic nature of SVC is also affected by the lower layer functions of rate control and shaping that are frequently used to reduce congestion and regulate flow on modern networks. To prevent denial of service and other harmful attacks it is necessary to maintain network links at stable levels through packet buffering, modification, or dropping the identified packets [63]. To limit network congestion video packets are occasionally modified at either the group of pictures, picture, or macroblock levels of granularity [64]. Video frames are often divided up into smaller sub-sections to facilitate processes such as rate shaping and error concealment. An eight by eight array of pixels is known as a block and a two by two array of blocks is known as a macroblock [65-66]. Rate shaping and error concealment algorithms analyze these sub-sections for temporal and spatial characteristics such as motion vectors. Given that rate control and shaping functions may be ongoing, it is imperative that SVC operations on these video sub-sections work in conjunction with the other dynamic video coding processes [38][67].

A tune-in stream is a parallel stream with a higher number of RAPs that some systems use to reduce the CCT. This second stream has a lower resolution but has a much higher I-Frame rate to reduce the RAP delay [21][35][44][46]. The primary disadvantage of tune-in streams is that it requires specialized STB applications to manage the parallel streams and transition between the two.

A tune-in server differs from an edge server in that it operates in parallel to the traditional video path. A specialized tune-in server is positioned upstream to receive available channels, buffer each channel, and send rapid bursts of a new channel upon selection by the user [21][35][47]. The tune-in server would detect a channel change request and then empty its buffer for the requested channel at a higher bit rate using all of the available link bandwidth for a short period of time. In this method the STB would receive the next I-Frame much quicker than otherwise to again reduce the RAP delay. Tune-in servers are designed to operate on existing STBs with little to no interoperability problems. The main drawback is that this extra component must be introduced into the system adding considerable expense and complexity.

These various methods of dynamic video coding have gained acceptance and use in IPTV applications to reduce CCTs primarily through reducing RAP delays. Not all applications can support these methods and significant CCT issues still remain with their use. As mentioned earlier overall CCT solutions tend to be an optimized combination of several techniques that minimize different areas of the entire process.

2.5 Use of RTP Video Streaming

Real-Time Transport Protocol (RTP) is gaining popularity over plain UDP for certain IPTV applications. When used in this fashion RTP is typically run on top of UDP to provide enhanced clocking, checksum support, and packet identification [21-22][27][45][48]. This is accomplished through the following RTP mechanisms:

- Payload Type Identification
- Sequence Numbering
- Time Stamping

Payload type identification assists advanced IPTV applications in quickly identifying the format and type of data encoded, such as audio in the G.711 format. Easier identification of packets streamlines decoding operations and can result in performance enhancements and slightly better CCTs. Sequence numbering gives applications the ability to rapidly identify lost or out of order packets to facilitate error concealment processes. Time stamping of packets along with RTP Control Protocol (RTCP) sender reports allow for the matching clocks to assist with AV Sync functions and de-jitter calculations. By reducing delays associated with de-jitter and AV Sync processes, the RTP time stamping capabilities also provide slight improvements to CCTs. Some researchers suggest the use of RTP or related unicast bursting to provide a dynamic protocol model to go along with the dynamic video coding of the previous section [21][68-70]. Recent research into Time Shifted Sub-channels (TSS), Multicast Assisted Zap Acceleration (MAZA), and their related schemes has been put forward to further address this very issue [71-73]. The only apparent drawback to these approaches is the further need for more interoperability verification on a project level basis.

2.6 Reversing IGMP Leave and Join Messaging Order

During the typical channel change process a STB first sends a Leave message to stop the streaming of the current channel and then it sends a Join message to request the new channel. This order is not set by the standards and was established when tighter bandwidth constraints existed during the early development of DSL. The order has continued to this day as the convention for IGMP channel change processes. Until the LHR receives this Join request no action will be taken to begin streaming the new channel requested. It has been proposed that this order should be reversed so that the Join message is sent out first to begin streaming the requested channel faster thereby resulting in a reduced CCT [28].

The time lag between transmissions of the Leave packet and the Join packet has been documented as ranging from 20 ms up to 200 ms depending on the STB device used [28][41]. This delay equates directly to the potential reduction in CCT by this proposed change of IGMP convention. In laboratory trials that will be presented in later chapters, these results were independently verified by this research with the delay between these outgoing IGMP messages routinely measured in excess of 150 ms.

Excluding the modification efforts required for existing firmware, this change could be made to the IGMP channel change process with little to no impact on other network devices. STB manufacturers could code their devices to behave in the new fashion without violating existing standards or experiencing major interoperability issues. Modern broadband networks would not be expected to experience significant congestion issues by the extended overlap where both the current and new channels are streamed until the Leave message is received. No other apparent problems or documented opposition to making this change were identified, while another 20 to 200 ms could be saved in the overall CCT process with this modified IGMP implementation scheme.

2.7 Adjacent Groups Methods

When a channel change request is made a significant delay occurs before the first packets are received for the new channel as documented earlier. If the new channel was already being streamed to the STB, then those delays would not be present and the STB could begin its processing of the new channel. In Chapter I this delay was identified to be two times the one way Network Delay plus the IGMP Processing time. Researchers have suggested that the elimination of this delay can be achieved in certain circumstances by pre-selecting likely channels for transmission. The Adjacent Groups method proposes sending not only the channel requested but also the next channels up and down from the one requested [25-26].

The analysis of the Adjacent Groups method shows that this proposal would indeed lower CCTs whenever a user next enters the up or down channel button on their remote. The average reduction in CCTs for this method is completely dependent on the user's behavior and channel surfing patterns. Users that do not sequentially change channels would receive no benefit from this proposal whatsoever, while users that only change sequential would always see this reduction in their CCTs [25]. For users that are sequentially changing channels this proposal also provides the benefit of reducing channel misses where a series of changes is made so rapidly that the system cannot respond with a requested channel before the next request is made. A notable point about this method is that these adjacent channels are always being streamed even if a given user's behavior indicates a poor likelihood of success for the technique.

An improvement to this method was then proposed to monitor the user's previous behavior to determine if they are engaged in sequential channel surfing or not [26][38][73]. When the sequential criteria for a given user is met then the adjacent channels will also be streamed, but if the criteria is not met then only the selected channel is streamed. This minor improvement does preserve bandwidth and processing capabilities for non-sequential users that would not benefit from adjacent channel streaming, but again only the sequential surfers would see any benefit from this proposal.

Under these proposals the STB would be responsible for tracking and initiating the requests for the adjacent channels, so it would require no changes of network equipment to implement. However, this reliance on the STB does have the inherent disadvantage of requiring large numbers of clients' equipment to be upgraded prior to implementation.

2.8 Client-Based Statistical Models

Building on the initial concepts put forward in the Adjacent Groups documentation, researchers have suggested several increasingly sophisticated approaches to provide improved channel pre-selections to reduce CCTs [21][26][29-30][34][36][40][73-82]. This began with the modest improvements to the Adjacent Groups method by the suggestion that additional benefits could be gained by giving consideration to the last channel viewed and the most popular channels [26]. Specific algorithms were not introduced in that literature regarding the collection and handling of these additional advancements for this client-based proposal. Another discussion of these concepts was made in [21] primarily by providing an overview of these other proposals with no additional research or results presented for this specific topic.

Further improvements were proposed through the use of a ratings server to provide localized Nielsen-like data on content popularity that would function in tandem with clients' HGs and STBs to generate more reliable pre-selected channel listings [29][75]. Based on the future works section of [29], the process had yet to be implemented or evaluated experimentally as specific algorithms and details were lacking. A fuzzy logic algorithm was proposed for use in ratings server technology in [75].

For residential applications based on Gigabit Passive Optical Network (GPON) technology, another proposal was presented to improve IPTV performance and CCTs by identifying neighborhood peer groups with the localized favorite channels receiving priority treatment [30][83]. GPON allows users to share common links throughout implemented neighborhoods, so this proposal leverages the information collected from each residence to optimize the video broadcasting through conservation of resources based on statistical predictions. This proposal was also not well defined with few specifics provided, and it was only geared towards PON configurations.

Along similar lines a proposal was provided to enhance CCTs for the next generation wireless networks through the use of an extended IGMP system. Fourth generation (4G) users of the proposals in [74][77][84] would be allowed to rapidly change channels to any other channels currently being viewed by other users connected to the same edge server. This method is based on the Worldwide Interoperability for Microwave Access (WiMAX) standards defined by the IEEE 802.16 standard. The use of localized networks sharing common channels was also proposed by [36] for STBs connected to the same edge server. These proposals involve both client and network processes to be upgraded and maintained to share the cache of channels being viewed by the members of the local network. Both proposals also depended solely on simulated or theoretical results and new proxy processes to estimate the potential gains. Improvements to Multicasting Trees were proposed in related research in [85].

The path towards more efficient IPTV pre-selected channel usage was continued in [34] with increased emphasis given toward making the case for this technical innovation. This research introduced the concept for “Intelligent Pre-Fetching” of a list of channels to be transmitted for the purposes of reducing CCTs based on statistical information to be collected in user STBs. The justification for the proposal in this research was compelling, but the method of implementation was unspecific and left for future development.

In related research there have been several recent proposals for the development of innovative Electronic Program Guide (EPG) applications that provide refinements to better determine user preferences. The rationale for these improved processes is to provide viewers with custom EPGs that reflect their favorite channels prominently, to improve customer satisfaction, to assist with network content to be offered, and to lead efforts at improved targeted advertising. One of these approaches proposed using stored viewer STB data coupled with a novel remote control that would allow users to manually input their current mood to generate personalized EPGs [78]. This model was not proposed for the purpose of reducing CCTs, but it did put forth an improved mechanism for determining viewer preferences based on processing these viewer inputs through a neural network that would presumably reside in some unspecified server on the network.

Continuing down the path of improved EPG development, another proposal was made to predict user preferences through the use of Markov Models that make real-time satisfaction calculations. The inputs into this model were proposed to be STB historical data as well as video feedback of viewers' facial expressions as they are themselves being filmed for their reactions to the current channel [79]. This method was not intended for ubiquitous deployment, but it was proposed to advance efforts in the production of Nielsen-like ratings. The researchers in [80] examined another approach to provide improved EPGs by generating user profiles based on geography, time-of-day, and day-of-week analysis of viewer channel preferences. This research introduced a method to perform time-weighting on the collected data through a Bayesian network that operated multiple sliding windows across separate temporal regions of the data. All of these efforts to support improved EPG services were not geared toward reducing CCTs, but rather to achieve the benefits described above for EPG applications. They did each put forth new concepts aimed at determining viewer preferences to support the applications.

This EPG research was brought back to the goal of reducing CCTs by [40][81] with the proposal to use the EPG viewer preference data to generate an improved pre-selection list of channels to be transmitted at any given time. While the background and justification for this proposal was sound, the implementation depended on a workable EPG system. The EPG proposals were merely suggestions of potential methods that could be used at some point in the future if the specific details of the schemes were fully developed. In addition to the effort that would be required to realize this solution, [81] suffers from several cumbersome problems involving the collection of the data to be used in the first place. Not the least of which is the heavy reliance on improved client devices such as complex remotes, STBs, and HGs. Solutions that are further based on manual inputs such as mood indicators or facial feedback would almost assuredly face some significant degree of customer dissatisfaction. The inherent problems associated with a requirement that a current network user must replace and upgrade their components before any of these client-based solutions could be implemented would severely hamper the effectiveness of these proposals. Even for new users the client-based approaches attempt to place increasingly complex functionality into the least complex devices on the IPTV networks.

The only apparent network-based pre-selection approach that was found in the existing research was a proposal made by Lucent engineers in a paper that suggested using historical channel change data to develop customized pre-selection lists of channels to be streamed for CCT reduction [82]. This proposed method would develop channel sets based on popularity, time-of-day, day-of-week, and geography from historical data collected in a standalone Service Control Platform that managed the solution. Possibly due to proprietary concerns, no specific details or algorithms were presented in this proposal and it was essentially just a general outline for a potential future program to assist IPTV development within metro business units.

Even though many of these previous research efforts lacked specific algorithms and details required for implementation and almost all were dependent on client-based modifications, these researchers did present varying degrees of corroboration about the need for improved CCTs and the potential benefits of streaming pre-selected channels.

CHAPTER III

LABORATORY TESTING TO DEMONSTRATE EFFECTS

The concept that IPTV CCTs would be reduced if the next channel to be selected was already streaming in advance of the change request was discussed in earlier chapters. This CCT reduction was theorized to be approximately equal to two times the one way Network Delay between the STB and LHR plus the IGMP Processing time for handling the query request by the LHR that serves as the Querying Device. It was shown that previous researchers have proposed solutions based in part on this concept, but documented results confirming these theoretical reductions were either not performed or not publicly available for review. Given the importance placed on actual verification of these claims about the potential for CCT reductions, it was determined that independent laboratory testing could corroborate these benefits or show flaws to the conceptual understanding of the proposal. The independent testing made during the course of this research into the effectiveness of this method is documented in the following sections.

3.1 Rationale for the Laboratory Testing

The laboratory testing performed during this research project was geared to show the actual time reductions that would be achieved by implementing a scheme by which pre-selected channels were effectively multicast down to the STB in advance of the next channel change. It is important to note that later chapters will detail the actual processes involved with determining the optimal channels to be pre-selected for multicast caching, but this chapter is focused on what CCT reductions would be seen if the pre-selected cache successfully streamed the next channel to be selected. If the next channel requested was not included in the multicast cache of pre-selected channels, then no CCT reduction would be seen from this proposal. Later discussions will simulate and document the expected effectiveness of the cache in correctly determining the next request, but that is not taken into account in this laboratory testing.

The proposed mechanics of this research will be detailed in later sections, but network devices will generally be responsible for determining which channels will be included in the multicast cache for its downstream elements. Consider the case where a LHR acting as the Querying Device for a group of STBs has determined which channels are pre-selected for the multicast cache and is actively streaming them downstream along with the current channel being viewed. Since current Layer 3 devices do not include this functionality, then they are not presently capable of performing this task in a laboratory or field environment. To get around this limitation another STB was connected to the appropriate HG to ask for an additional channel in advance of a monitored channel change request. This activity effectively generated a second video stream down to the HG to reproduce conditions that would occur if the research proposal was fully implemented.

Prior to measuring the results of this cached condition, it was first necessary to perform a series of baseline tests to determine the operation and timing of the different STBs to be used in this laboratory experiment. This was accomplished in both minimal and full lab configurations. The minimal configuration, shown in Figure 3.4 and described in more detail below, was used to indicate the best CCTs attainable for the various STBs available and the Querying Device utilized by minimizing the number of intermediate devices and Network Delays. The full configuration, shown in Figure 3.1, was used to represent conditions typically found in many IPTV applications. The full configuration consisted of two main setups that represented both the non-cached and cached conditions of testing. In the non-cached condition the additional adjacent STB has not requested any video channels, so only the currently watched channel is being streamed to the HG and on to the monitored STB. In the cached condition the adjacent STB has requested and is viewing the next channel, so both the currently viewed channel and the next channel to be requested are being streamed to the HG. Only the specific channels requested by a STB are passed on to it depending on the type of HG used and the parameter settings. The HG used in this research was configured for IGMP Snooping, so the additional channel requested by the adjacent STB was not forwarded on to the monitored STB until it was requested. Under these various conditions the monitored STB then made the request for the next channel while all of the bi-directional packets were captured and the entire process was being recorded. The recorded video of each channel change was then closely analyzed to determine the observable delays and overall CCT associated with each STB. This process was repeated to yield statistically viable results.

3.2 Laboratory Configurations

Figure 3.1 shows the full laboratory configuration that was built specifically for the purposes of this research project. This lab consisted of the following components:

- (2) LanTech GE-24F2GBM Gigabit Managed Network Switches
- Xavi X-550 Home Gateway

- Agilent J2300D WAN Advisor Test Set
- Motorola VIP-1216KK Set Top Boxes
- Amino AmiNET130M Set Top Box
- Amino AmiNET125 Set Top Box
- Westinghouse SK-32H240S LCD HDTV
- Sylvania SST4132 Color TVs
- Personal Computers (PC) with Accessories
- NETGEAR DS104 Hubs
- Miscellaneous cables, optics, power supplies, breakers, and other hardware

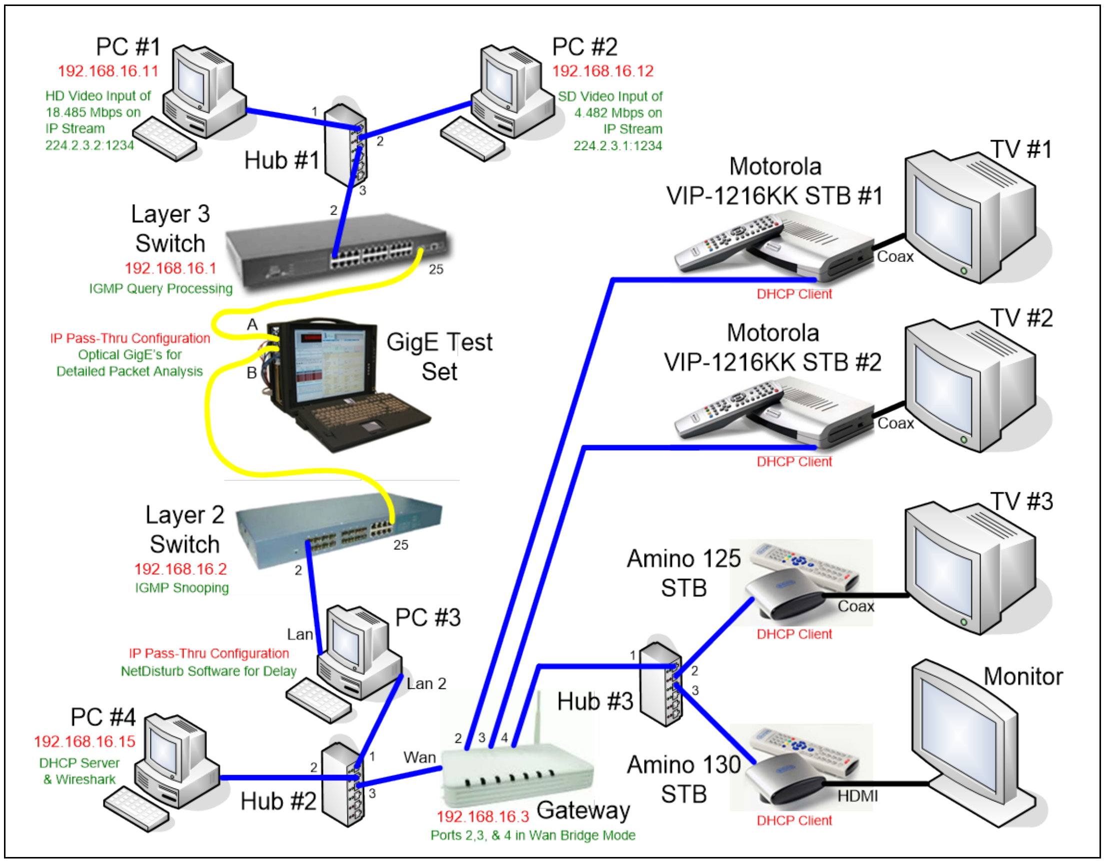


Figure 3.1 Full Laboratory Configuration

As indicated in Figure 3.1 two of the PCs are used as sources for the video inputs with the capabilities of providing multiple SD and HD streams. The LanTech managed switch labeled as "Layer 3 Switch" acts as the LHR with full IGMP Query Processing capabilities and all available video channels being input into it continuously. This LHR switch with partial layer 3 functionality has optical Gigabit interfaces to connect to another LanTech managed switch with the Agilent WAN Advisor in throughput mode in between. The blue lines on Figure 3.1 represent electrical 100 Mbps Ethernet connections, and the yellow lines represent optical GigE connections. This convention will be continued on later diagrams throughout the rest of this paper. This second switch functions as an intermediate layer 2 device and is configured to perform IGMP Snooping processes. The WAN Advisor allowed for the capture and inspection of all packets sent between these two devices, which proved to be beneficial during analysis and troubleshooting phases. PC #3 was configured with NetDisturb network impairment emulator software to allow for the simulation of delays, jitter, and other conditions typically seen in IPTV networks. The NetDisturb emulator was configured to simulate a constant Network Delay of 600 ms in each direction for all packets in line with prior documented laboratory research [1][5][7]. PC # 4 was configured to act as a Dynamic Host Configuration Protocol (DHCP) server that provided IP addresses for the various STBs. PC #4 was also used to capture packets in and out of the Gateway using Wireshark software. The Xavi X-550 Home Gateway was also set up to perform IGMP Snooping and to appropriately route all streams through it using WAN bridge mode to the connected STBs. One of the Motorola STBs was used as STB #1 to simulate a multicast cached condition for one or more pre-selected channels. The other STBs provided three different monitored performance opportunities based on the individual capabilities of these three unique devices. The various NETGEAR hubs allow proper interconnection and numerous test points from which packet inspection was possible. Miscellaneous hardware connected the system together as shown in Figure 3.1 with a -48 VDC power supply sourcing the LanTech switches. The specific IP scheme utilized is also presented with most devices configured for the static IP addresses shown except for the STBs which were configured as DHCP clients. Figure 3.2 shows a photograph of the full laboratory configuration including the Sony DCR-HC40 Digital Video Camera Recorder mounted on a tripod at the bottom of the photo that was used to visually record each

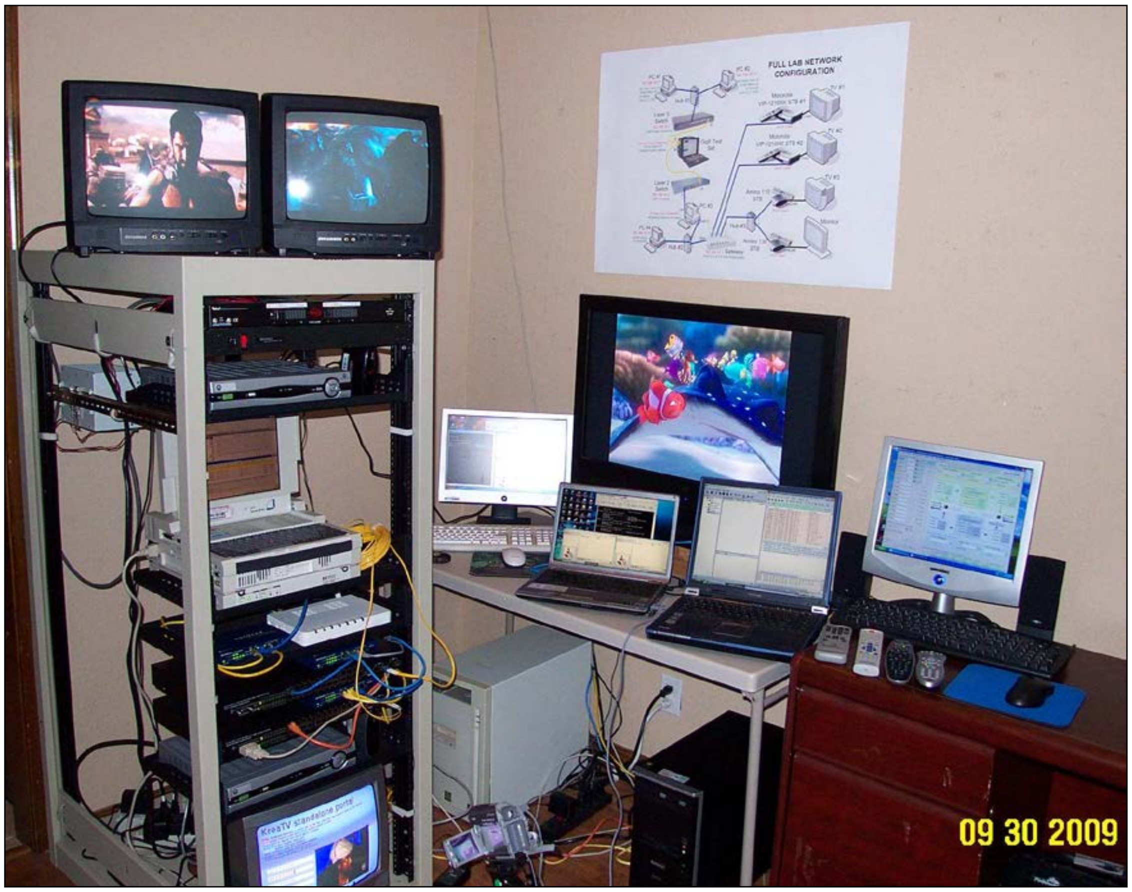


Figure 3.2 Photograph of Full Laboratory Configuration

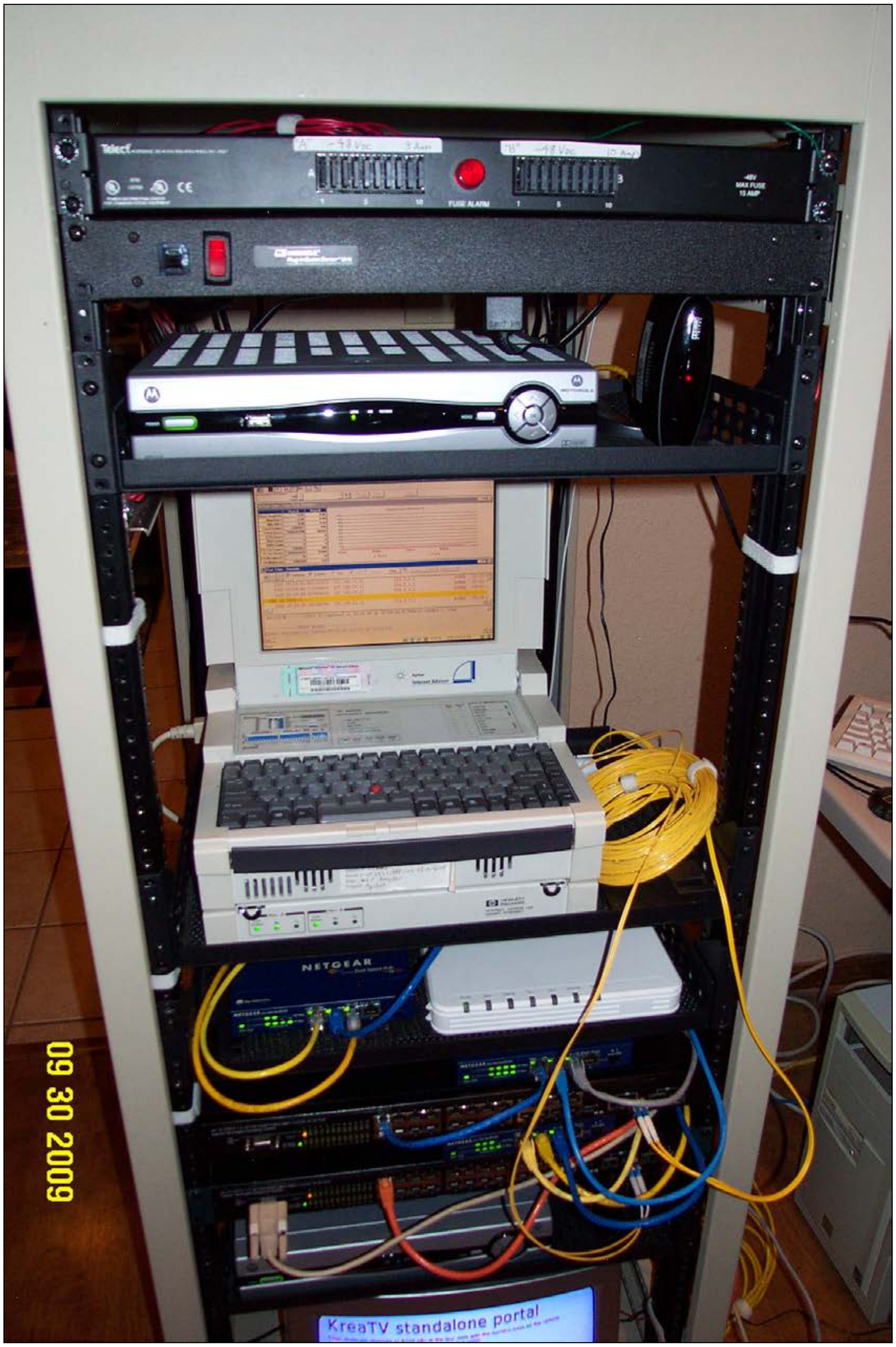


Figure 3.3 Close-Up Photograph of Laboratory Cabinet

tested channel change. Many of the components were either rack mounted or placed on trays within the open telecommunications cabinet toward the left of the photo. Figure 3.3 shows a close-up view of the primary equipment in this cabinet. Appendix B provides specific details on the firmware versions and parameter configurations for each of the major components used in this laboratory testing.

The AmiNET125 STB can only handle SD video streams, but the AmiNET130M STB can process SD or HD channels. The Motorola VIP-1216KK STB can handle SD or HD streams via the KreaTV operating system [86]. Source PCs #1 and #2 used VLC Media Player 0.9.9 software from VideoLAN to multicast SD streams at 4.482 Mbps and HD streams at 18.485 Mbps to serve as various video sources.

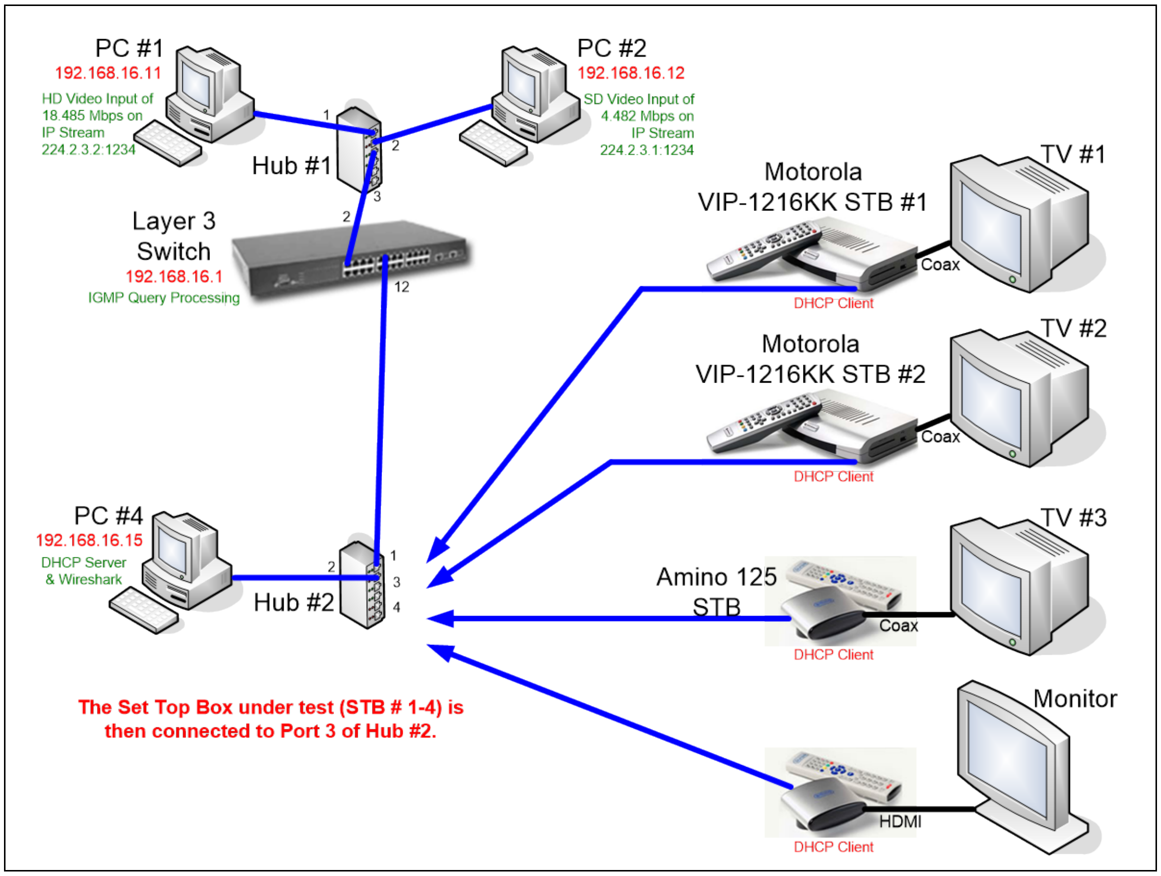

PC #1
192.168.16.11
HD Video Input of 18.485 Mbps on IP Stream 224.2.3.2:1234
PC #2
192.168.16.12
SD Video Input of 4.482 Mbps on IP Stream 224.2.3.1:1234
Hub #1
1
2
3
Layer 3 Switch
192.168.16.1
IGMP Query Processing
2
12
Motorola VIP-1216KK STB #1
TV #1
Coax
DHCP Client
Motorola VIP-1216KK STB #2
TV #2
Coax
DHCP Client
PC #4
192.168.16.15
DHCP Server & Wireshark
Hub #2
1
2
3
4
Amino 125 STB
TV #3
Coax
DHCP Client
Monitor
HDMI
DHCP Client
The Set Top Box under test (STB # 1-4) is then connected to Port 3 of Hub #2.

Figure 3.4 Minimal Laboratory Configuration

The minimum configuration used for the laboratory testing is shown in Figure 3.4 where the WAN Advisor, Layer 2 Switch, PC #3, and Gateway were removed from the network. The minimized path between the STB and the Layer 3 Switch with the IGMP Querying capability consists only of Ethernet cables and a single hub device. PC #4 is still capable of capturing all packets through its hub connection when a channel change process was performed. The testing performed with this minimal configuration yielded baseline data about the best channel change results attainable with this equipment under optimal conditions.

The non-cached series of tests that were performed in both the minimal and full configurations only involved the use of a single STB. Either STB #1, 2, or 3 was used during each of the non-cached tests with no additional video streams ongoing. The STB was initially set up viewing one of the available channels and then the channel was changed to another available channel while the entire process was monitored and recorded. The cached series of tests were initially set up with STB #1 viewing the next channel while one of the other STBs was set to a different current channel. The STB under test was then changed to the channel that STB #1 was viewing while the entire process was monitored and recorded. In this cached environment the next channel to be changed to was already present and streaming since STB #1 had previously requested it. These two series of tests indicate the CCT for a common network topology and the CCT if a multicast cache system correctly pre-selected the next channel to be requested.

Each individual CCT test was recorded using a Sony DCR-HC40 Digital Video Camera Recorder, and these recordings were downloaded to a PC for analysis using Quick Time Player version 7.6.4 to measure the CCT on a frame-by-frame basis. The start of each change request was indicated by a Remote Control LED turning ON, and the end of each request was determined to be the first full frame of video displayed on the TV for the new channel. With the camera mode set to CAMERA-MEMORY these recordings were made at a frame rate of 25 frames per second [87-88]. By setting the Quick Time display mode to Frame Number instead of Standard elapsed time, it was possible to analyze the activity down to the accuracy level attained by the 25 frames per second recording rate [89-90]. This process is described in more detail below, but the accuracy level can be determined using the reciprocal:

1 / (25 frames/second) = 0.04 seconds/frame

Any event that took place between two recorded frames could have occurred as early as immediately after the first frame was displayed or as late as immediately before the third frame was displayed. Figure 3.5 shows an example channel change which began with the LED OFF when the 1st frame was displayed, but it was ON when the 2nd frame

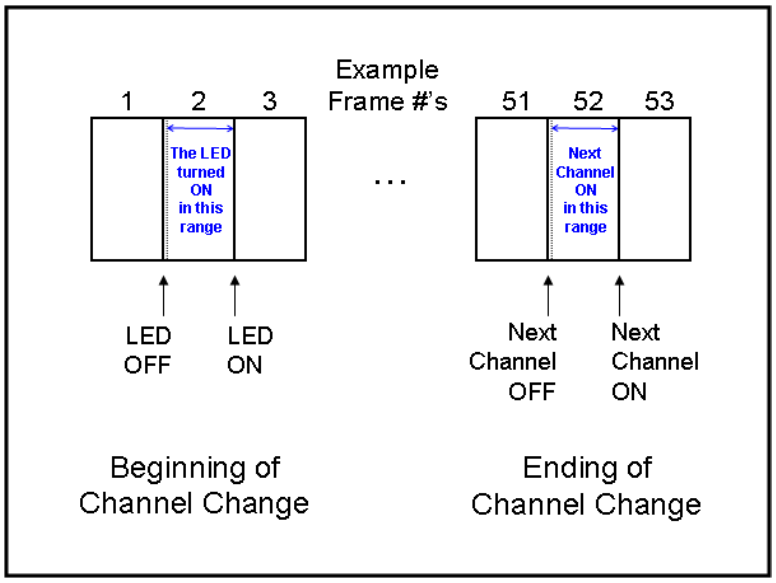


Figure 3.5 Accuracy of Event Calculations Using Recordings

was displayed. Therefore the LED could have turned ON any time after the 1st frame ended and before the 2nd frame ended. Similarly the completion of the channel change could have been any time after the 51st frame ended and before the 52nd frame ended. The CCT is determined by taking the difference in time between these two events. The example activity in Figure 3.5 would be determined by direct observation or calculated through all possible values to determine the full range in the following manner:

CCT_{direct}	= **Ending Frame – Beginning Frame** (direct observation)
	= **52 – 2 frames** (first indications of events)
	= **50 frames**
$CCT_{possible}$	= **Possible End Frames – Possible Beginning Frames**
	= **(52 or 51) – (2 or 1) frames** (possible range per event)
	= **52 – 2 or 52 – 1 or 51 – 2 or 51 – 1 frames**
	= **50 or 51 or 49 or 50 frames**
	= **50 ± 1 frame**
	= **CCT_{direct} ± 1 frame**

These calculations show that direct observation of the first indication of each event leads to a valid CCT with the resulting margin of error for each CCT calculation as follows:

CCT Margin of Error = ± 1 frame = ± 0.04 seconds

Each type of test was performed numerous times to get statistically reliable results for the statistical mean and variance. Thirty independent trials were performed for each test type in accordance with conventional interpretation of the Central Limit Theorem (CLT) [91-96]. A section in a later chapter will explore the concepts of independent and identically distributed (i.i.d.) random variables and how they relate to this research, but in the meantime note that all CCT trials performed in this laboratory testing were considered to be i.i.d. and therefore the CLT was applicable. The CLT provides the following guidance:

> **If $\bar{x}$ is the mean of a sample of size *n* taken from a population that has mean μ and the finite variance σ^2, then**
>
> $$Z = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}}$$
>
> **is a random variable whose distribution function approaches that of the standard normal distribution as $n \to \infty$ [92].**

The sample means and standard deviations (STDev) for each test type were calculated using the following equations with n = 30 [93][95]:

$$\text{Sample Mean}_{\text{CCT}} = \bar{x} = \frac{\sum x}{n}$$

$$\text{Sample Standard Deviation}_{\text{CCT}} = \sigma = \sqrt{\frac{(x - \bar{x})^2}{(n-1)}}$$

Performing 30 independent trials of each type allowed for the application of the CLT, which in turn allowed for a determination about the likelihood of accuracy using the "68-95-99.7 Rule" on the Normal Distribution. This rule states that the intervals defined by one, two, and three standard deviations away from the mean will contain approximately 68%, 95%, and 99.7% of a normally distributed population respectively [93][95][97]. This rule will be used later on to provide accuracy information about the testing results.

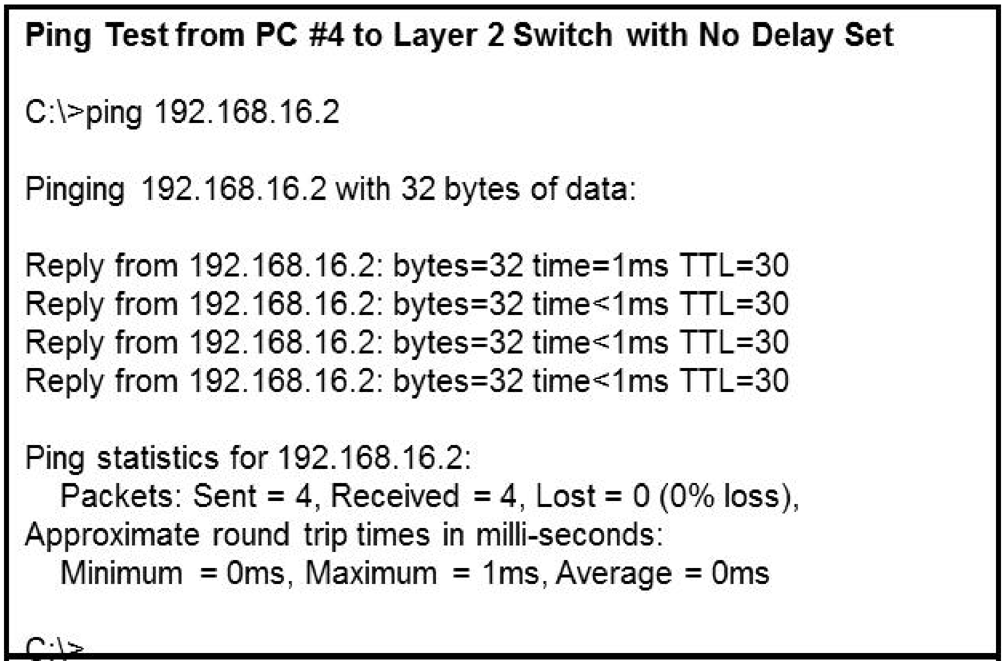
A total of 15 different types of lab tests were performed by varying the minimum and full configurations, the non-cached and cached conditions, SD and HD streaming, and the 3 different types of STBs involved. For each of the 15 different types of tests, 30 trials were performed, recorded, and evaluated to yield reliable results and conclusions about the potential effectiveness of multicast caching techniques.

3.3 Laboratory Testing Methodology

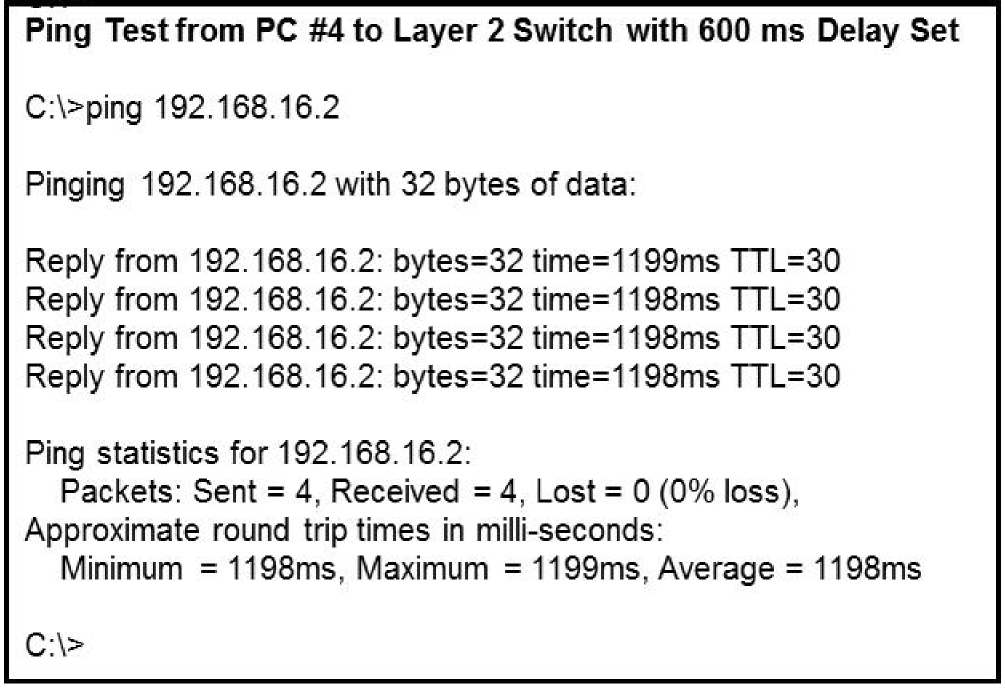
Note that throughout this section the Amino AmiNet 125 is referenced as A125 STB, the Amino AmiNet 130M is referenced as A130 STB, and the Motorola VIP-1216KK is referenced as MSTB for convenient abbreviations. Table 3.1 shows the 15 different types of lab tests that were performed to yield the results found in this section.

Functional Specifics for Each Test			
STB	**SD or HD**	**Minimal or Full**	**With or Without Cache**
A125	SD	Minimal	Without Cache
A130	SD	Minimal	Without Cache
A130	HD	Minimal	Without Cache
MSTB	SD	Minimal	Without Cache
MSTB	HD	Minimal	Without Cache
A125	SD	Full	Without Cache
A130	SD	Full	Without Cache
A130	HD	Full	Without Cache
MSTB	SD	Full	Without Cache
MSTB	HD	Full	Without Cache
A125	SD	Full	With Cache
A130	SD	Full	With Cache
A130	HD	Full	With Cache
MSTB	SD	Full	With Cache
MSTB	HD	Full	With Cache

Table 3.1 List of Tests Performed



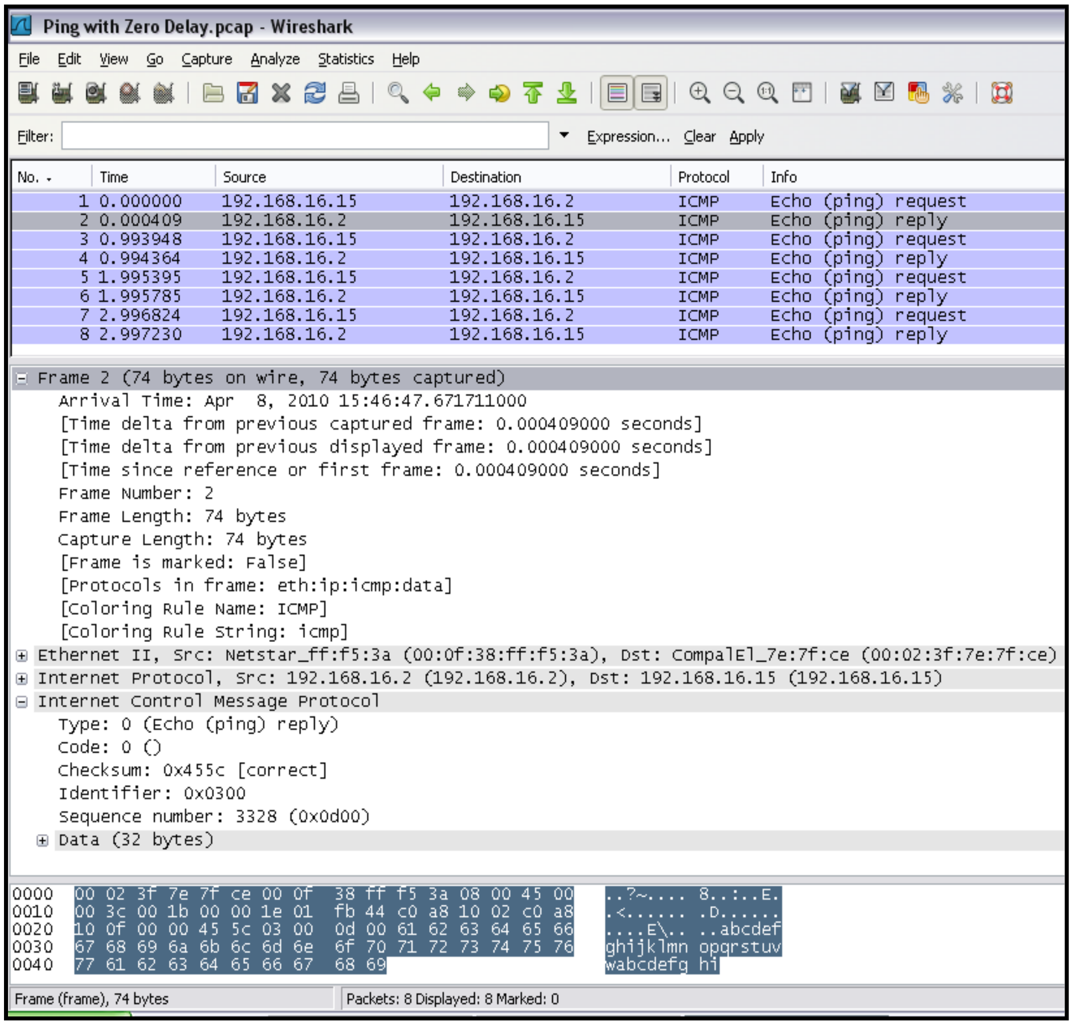
```
Ping Test from PC #4 to Layer 2 Switch with No Delay Set

C:\>ping 192.168.16.2

Pinging 192.168.16.2 with 32 bytes of data:

Reply from 192.168.16.2: bytes=32 time=1ms TTL=30
Reply from 192.168.16.2: bytes=32 time<1ms TTL=30
Reply from 192.168.16.2: bytes=32 time<1ms TTL=30
Reply from 192.168.16.2: bytes=32 time<1ms TTL=30

Ping statistics for 192.168.16.2:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
   Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```



```
Ping Test from PC #4 to Layer 2 Switch with 600 ms Delay Set

C:\>ping 192.168.16.2

Pinging 192.168.16.2 with 32 bytes of data:

Reply from 192.168.16.2: bytes=32 time=1199ms TTL=30
Reply from 192.168.16.2: bytes=32 time=1198ms TTL=30
Reply from 192.168.16.2: bytes=32 time=1198ms TTL=30
Reply from 192.168.16.2: bytes=32 time=1198ms TTL=30

Ping statistics for 192.168.16.2:
   Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
   Minimum = 1198ms, Maximum = 1199ms, Average = 1198ms

C:\>
```

Figure 3.6 Ping Tests to Verify Network Delay Emulator

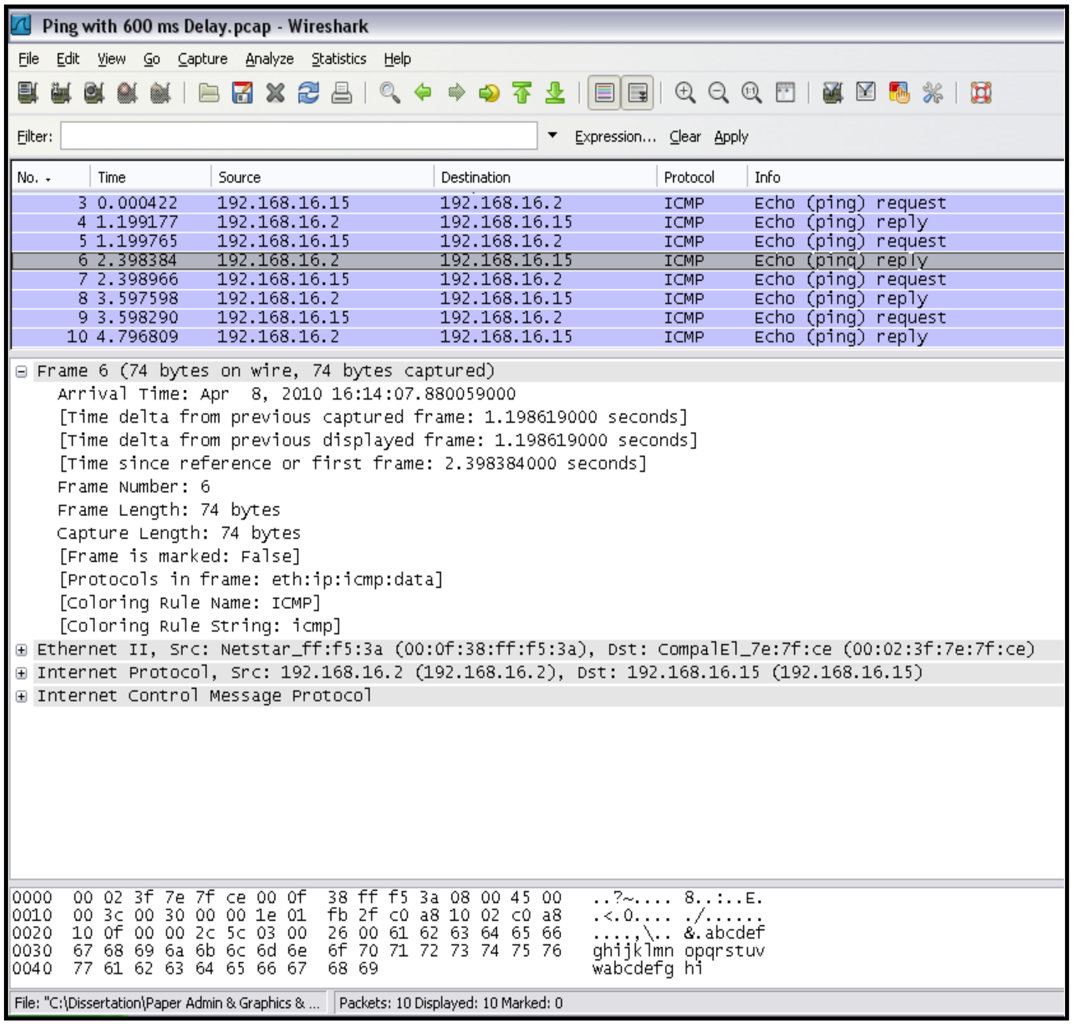
As previously noted the NetDisturb Emulator residing on PC #3 was set to delay packets by 600 ms for each direction to simulate Network Delays. Appendix B provides a more detailed view about how this software was configured to perform these functions. The operation of this software program was verified by running a series of ping tests from PC #4 to the layer 2 switch (see Figure 3.1) with the results shown on Figure 3.6. These two devices were immediately adjacent to PC #3 on each side of the emulator, so any ping test between them would have to make two trips across PC #3 through the emulator. The first ping test was performed with the emulator software set to provide no delay, and the results show ping delays of less than 1 ms. This delay was more precisely shown to be approximately 400 µs by the ping packets captured using Wireshark as displayed on Figure 3.7. The second ping test was run with the emulator set at 600 ms in each direction, and the results show an average round trip delay of 1198 ms.


Ping with Zero Delay.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No.  Time       Source          Destination     Protocol  Info
  1 0.000000   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
  2 0.000409   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
  3 0.993948   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
  4 0.994364   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
  5 1.995395   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
  6 1.995785   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
  7 2.996824   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
  8 2.997230   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply

⊟ Frame 2 (74 bytes on wire, 74 bytes captured)
    Arrival Time: Apr  8, 2010 15:46:47.671711000
    [Time delta from previous captured frame: 0.000409000 seconds]
    [Time delta from previous displayed frame: 0.000409000 seconds]
    [Time since reference or first frame: 0.000409000 seconds]
    Frame Number: 2
    Frame Length: 74 bytes
    Capture Length: 74 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:icmp:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp]
⊞ Ethernet II, Src: Netstar_ff:f5:3a (00:0f:38:ff:f5:3a), Dst: CompalEl_7e:7f:ce (00:02:3f:7e:7f:ce)
⊞ Internet Protocol, Src: 192.168.16.2 (192.168.16.2), Dst: 192.168.16.15 (192.168.16.15)
⊟ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0 ()
    Checksum: 0x455c [correct]
    Identifier: 0x0300
    Sequence number: 3328 (0x0d00)
  ⊞ Data (32 bytes)

0000  00 02 3f 7e 7f ce 00 0f  38 ff f5 3a 08 00 45 00   ..?~.... 8..:..E.
0010  00 3c 00 1b 00 00 1e 01  fb 44 c0 a8 10 02 c0 a8   .<...... .D......
0020  10 0f 00 00 45 5c 03 00  0d 00 61 62 63 64 65 66   ....E\.. ..abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi

Frame (frame), 74 bytes          Packets: 8 Displayed: 8 Marked: 0
```

Figure 3.7 Wireshark Display of Ping Tests with No Delay Set

Another Wireshark display of this test is shown on Figure 3.8 to be approximately 1.1986 seconds of delay. Subtracting the nominal 400 μs required for a zero delay ping test yields around 1.1982 seconds round trip time for the 600 ms delay setup. This verifies the basic operation of the emulator, but it also shows that the combination of emulator and PC clock accuracy does not provide an exact delay of 600 ms in each direction, as a delay of 1.2004 seconds would have been expected. This minor difference of a couple of milliseconds will be noted later as some the test results are presented.


Ping with 600 ms Delay.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time      Source          Destination     Protocol  Info
    3 0.000422   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
    4 1.199177   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
    5 1.199765   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
    6 2.398384   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
    7 2.398966   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
    8 3.597598   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
    9 3.598290   192.168.16.15   192.168.16.2    ICMP      Echo (ping) request
   10 4.796809   192.168.16.2    192.168.16.15   ICMP      Echo (ping) reply
```

```
⊟ Frame 6 (74 bytes on wire, 74 bytes captured)
    Arrival Time: Apr  8, 2010 16:14:07.880059000
    [Time delta from previous captured frame: 1.198619000 seconds]
    [Time delta from previous displayed frame: 1.198619000 seconds]
    [Time since reference or first frame: 2.398384000 seconds]
    Frame Number: 6
    Frame Length: 74 bytes
    Capture Length: 74 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:icmp:data]
    [Coloring Rule Name: ICMP]
    [Coloring Rule String: icmp]
⊞ Ethernet II, Src: Netstar_ff:f5:3a (00:0f:38:ff:f5:3a), Dst: CompalEl_7e:7f:ce (00:02:3f:7e:7f:ce)
⊞ Internet Protocol, Src: 192.168.16.2 (192.168.16.2), Dst: 192.168.16.15 (192.168.16.15)
⊞ Internet Control Message Protocol
```

```
0000  00 02 3f 7e 7f ce 00 0f  38 ff f5 3a 08 00 45 00   ..?~.... 8..:..E.
0010  00 3c 00 30 00 00 1e 01  fb 2f c0 a8 10 02 c0 a8   .<.0.... ./......
0020  10 0f 00 00 2c 5c 03 00  26 00 61 62 63 64 65 66   ....,\.. &.abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                     wabcdefg hi
```

```
File: "C:\Dissertation\Paper Admin & Graphics & ...   Packets: 10 Displayed: 10 Marked: 0
```

Figure 3.8 Wireshark Display of Ping Tests with 600 ms Delay Set

The next series of figures provide a detailed view about how the camera recordings of tested channel changes were used to generate data results. The pictures are screen shots taken with the recorded video being played a frame at a time. Each figure shows a specific frame number in the bottom left corner that is used to coordinate the changes made to the Remote Control LED and the TV display during channel changes.

Figure 3.9 shows the beginning sequence of a channel change using the A125 STB going from one SD video to another. Note that the frame number depicted by the 50F for frame 50 and the Remote Control LED is OFF. Figure 3.10 shows the next frame number 51 recorded as the channel change was made and the LED is now ON.

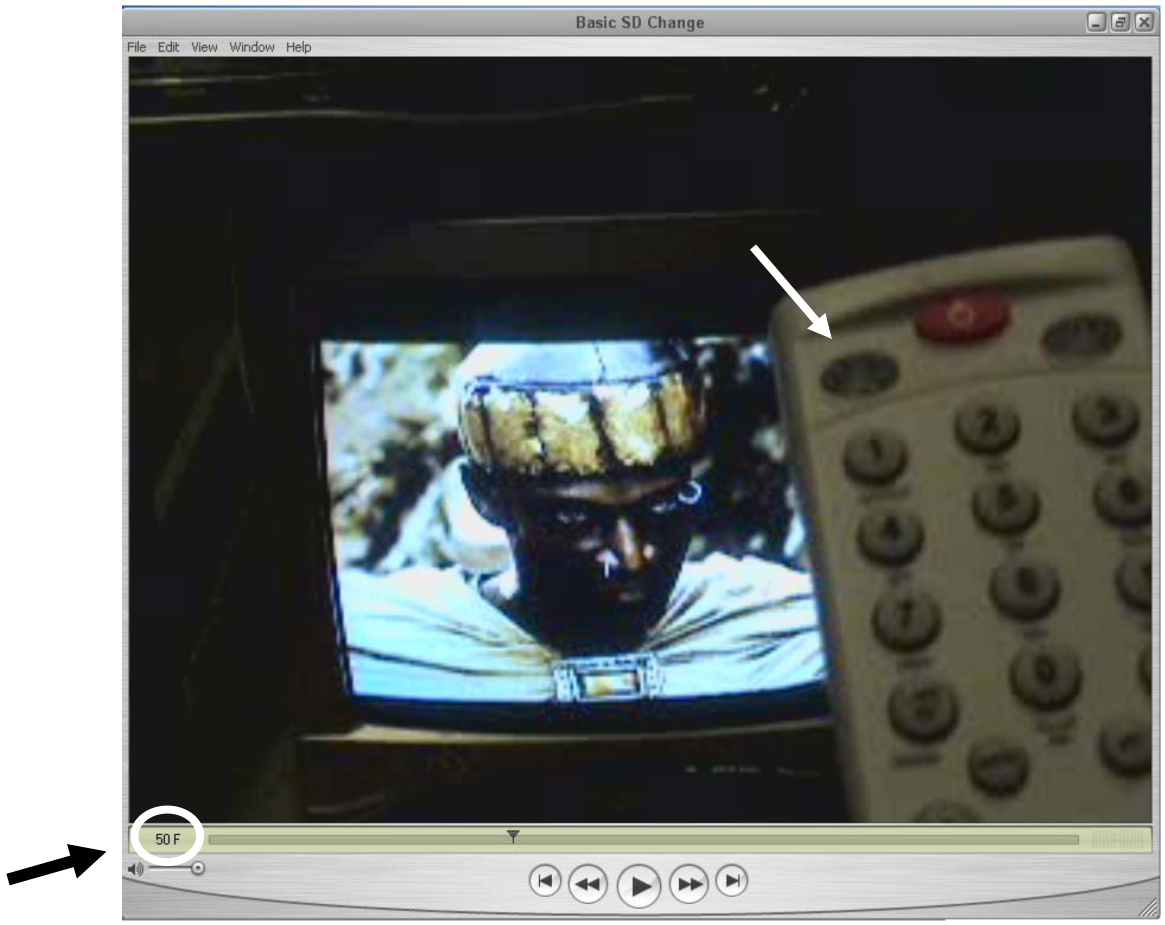


Figure 3.9 Frame 50 Showing Remote LED OFF

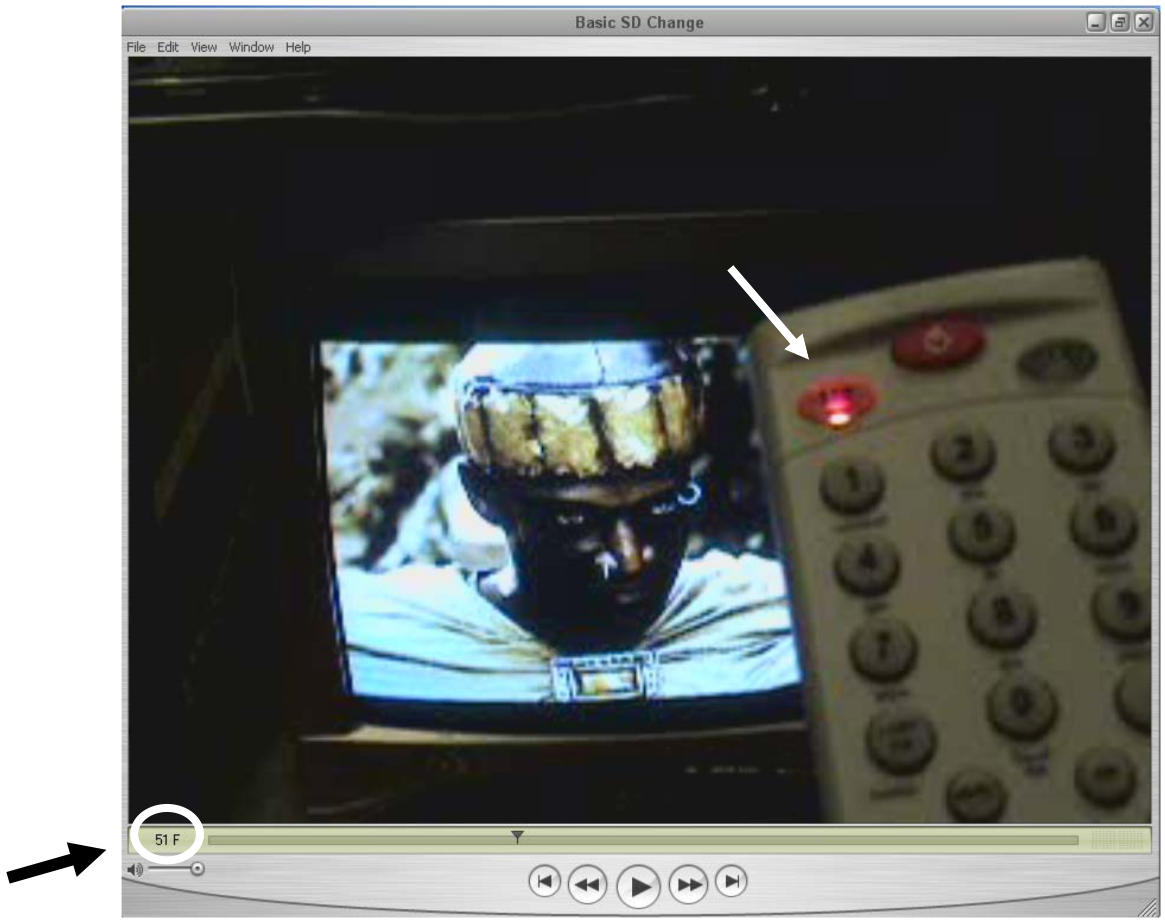


Figure 3.10 Frame 51 Showing Remote LED ON

The sequence from frame 50 with the LED OFF to frame 51 with the LED ON establishes the first detectable time for the start of the channel change as being between frame 50 and 51. A similar sequence of events establishes the time when the channel change was completed. Compare Figure 3.11 with Figure 3.12 to see that the next video channel was fully displayed somewhere between frames 116 and 117. This establishes the first detectable frame for completion of this particular channel change test as frame number 117. It is also apparent by comparing Figure 3.12 with Figure 3.9 that the system has changed to a different video channel. The overall CCT was then made by calculating the number of frames between these transitions and converting from frames to seconds.

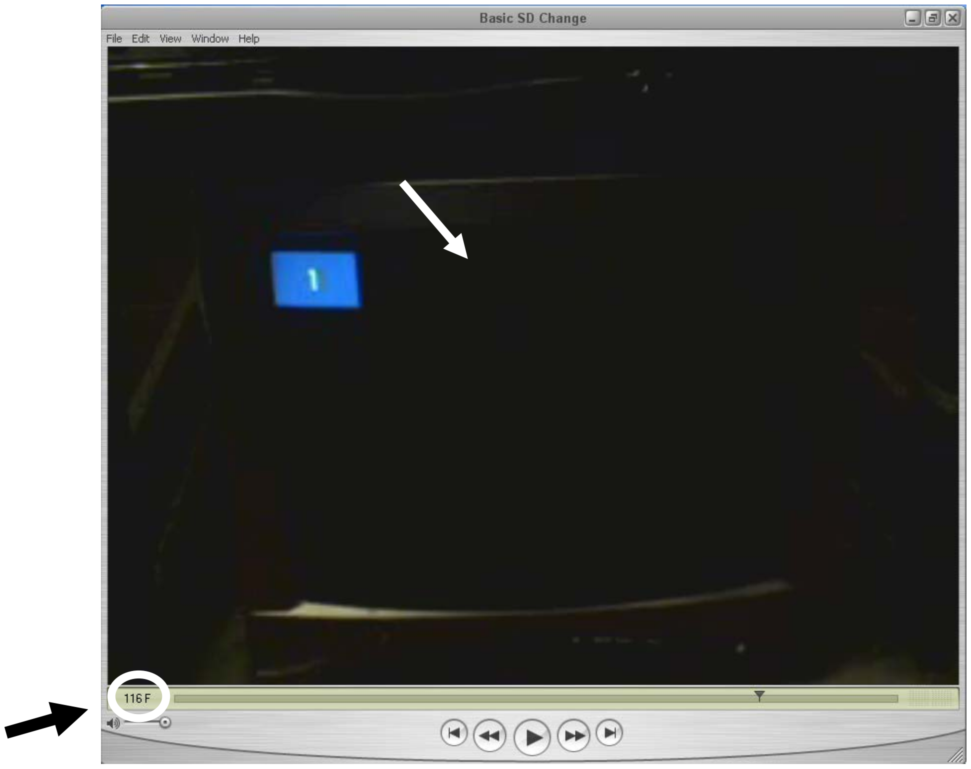


Figure 3.11 Frame 116 Next Channel Not Yet Displayed

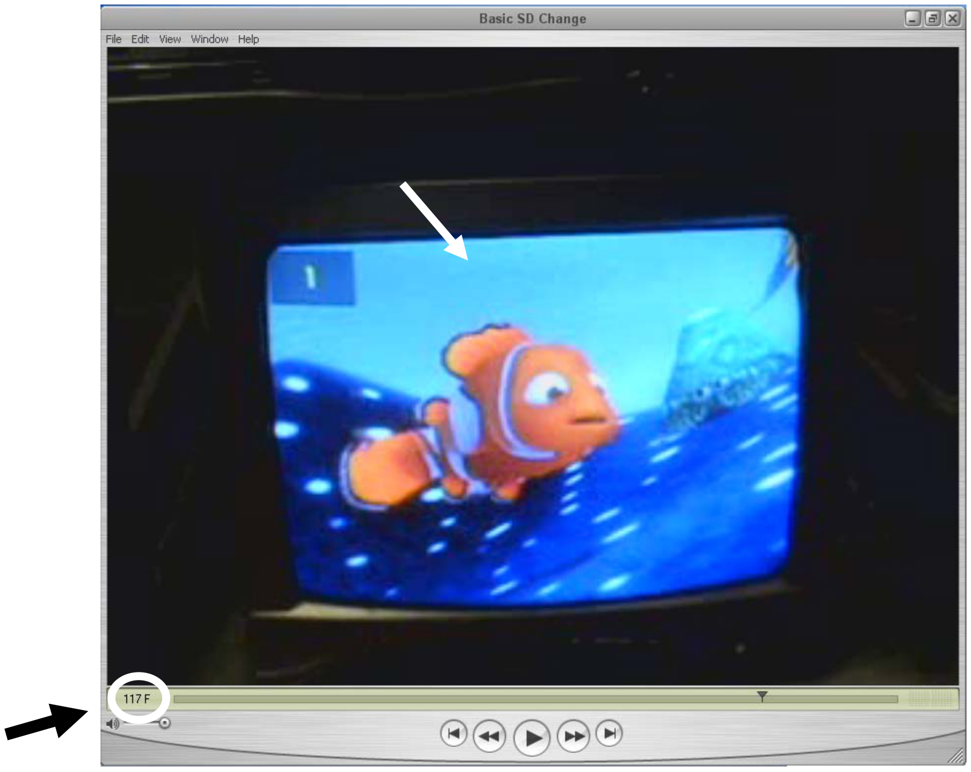


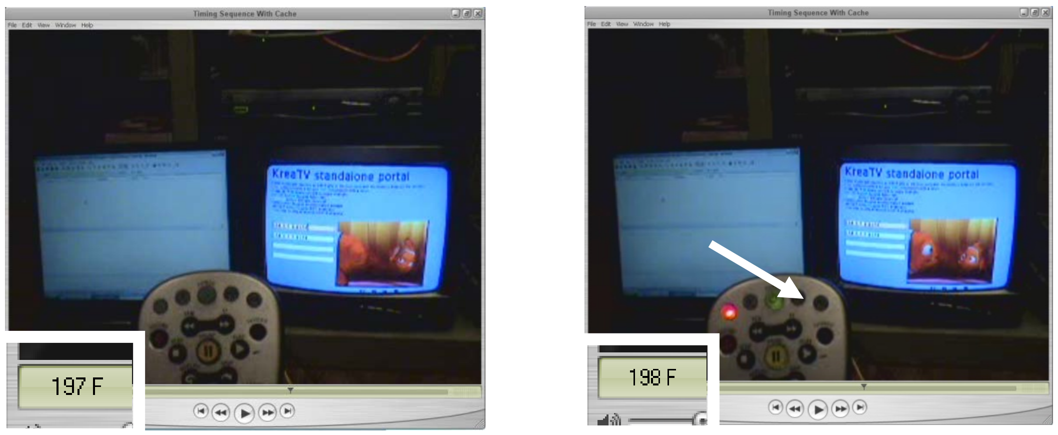
Figure 3.12 Frame 117 Next Channel is Displayed

Configuration:	A125 STB Change to SD Without Caching
1st Transition:	Remote Channel Change Request @ Frame 51
2nd Transition:	Next Channel Displays @ Frame 117

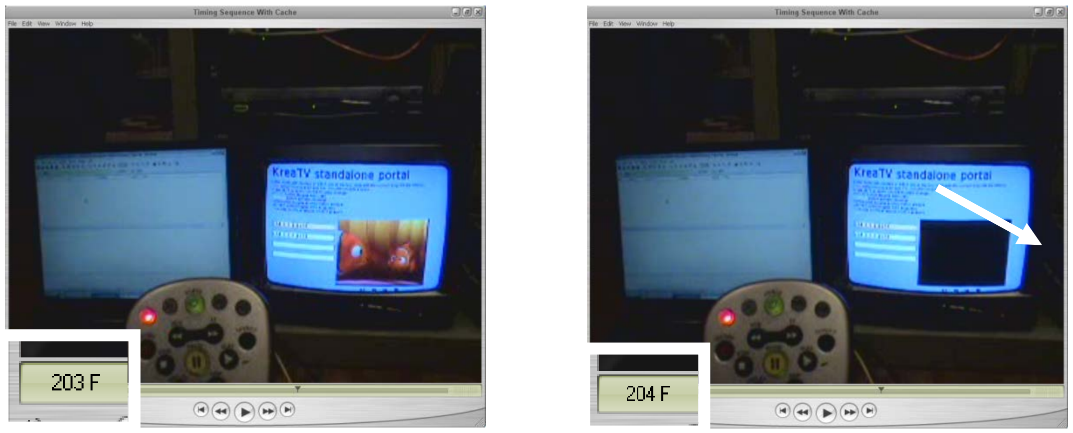
Figure 3.13 A125 STB Example of the CCT Calculation Method

Figure 3.13 completes the CCT calculation for the example trial by subtracting the 1st transition from the 2nd and converting from frames to seconds. This process was repeated for all thirty trials on each of the fifteen test types to generate the lab data sets. Some of the trials were performed with additional data being recorded to simultaneously show all packets transported along with visible STB activity in addition to the Remote Control and TV displays. The data collected from these select trials was used to generate detailed timing sequences for comparison to the previous research and documentation that was presented earlier.

Figures 3.14 through 3.16 represent data recorded during one of the trials selected for the more advanced monitoring techniques. Figure 3.14 establishes the start of the channel change process with the 1st transition at frame 198 as with the last example. Note that a PC monitor with Wireshark running on it is also in view to the left of the TV display. In the dark area above the TV is one of the Motorola VIP-1216KK set top boxes that was used in this trial. This MSTB trial changes from a currently viewed SD channel to an HD channel that has been cached. The MSTB display on the TV shows the KreaTV standalone portal firmware from Motorola used for these STBs during these tests.



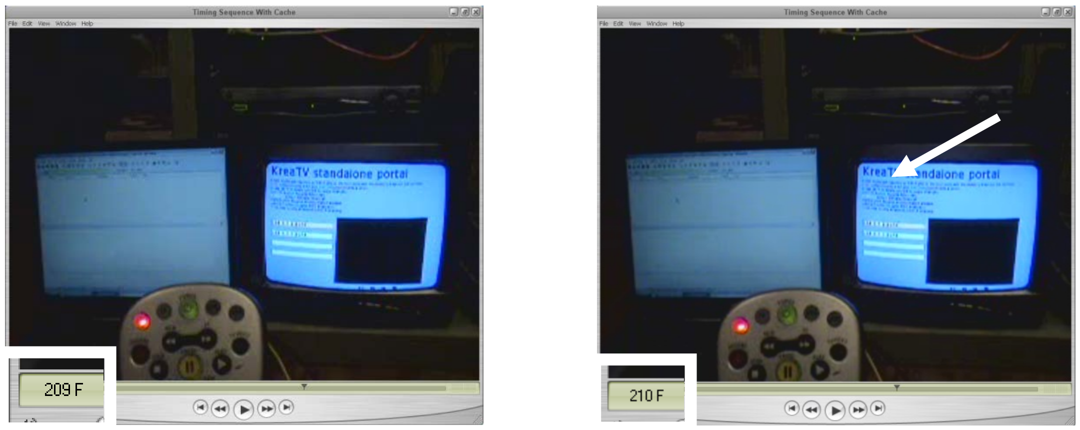
1st Transition showing Remote Channel Change Request @ Frame 198



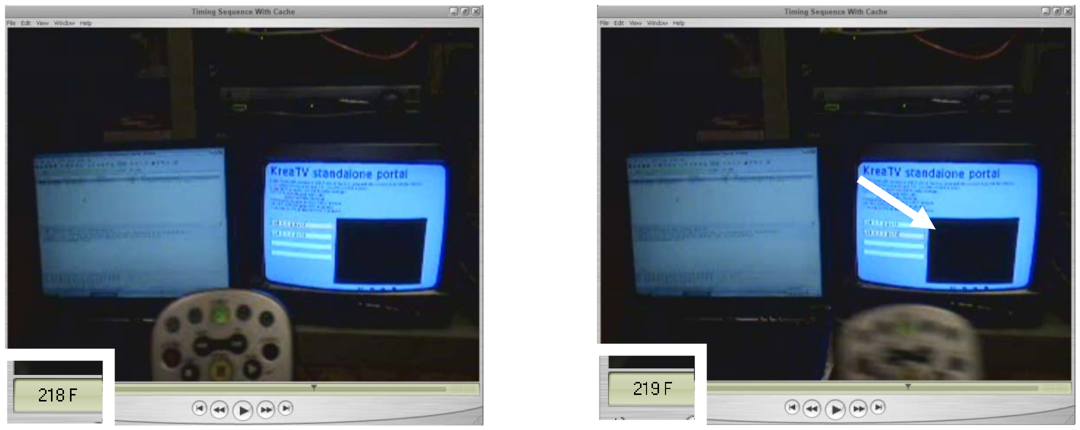
2nd Transition showing Channel 1 Display Dropped @ Frame 204

Figure 3.14 Beginning Transitions of MSTB Example from SD to HD

The 2nd transition on Figure 3.14 reflects the ongoing activity where the STB stops displaying the current channel at frame 204. The process continues on Figure 3.15 with the 3rd transition showing that the 1st IGMP packet was sent out by the MSTB at frame 210. The Wireshark program was filtered to display only IGMP packets during the test so that this transition could be seen. All of the packets have timestamps on them which can now be used to correspond packet timing with the recorded video time using the frame numbers for translation. The 4th transition shown on Figure 3.15 reflects a change in display by the KreaTV program showing the intended move from channel 1 to channel 2 at frame 219.



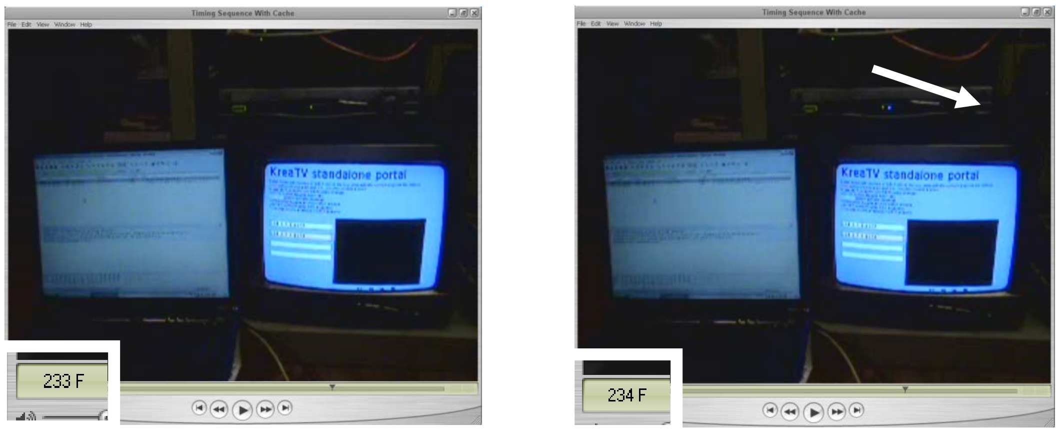
3rd Transition showing First IGMP Leave Ch 1 Packet Sent @ Frame 210



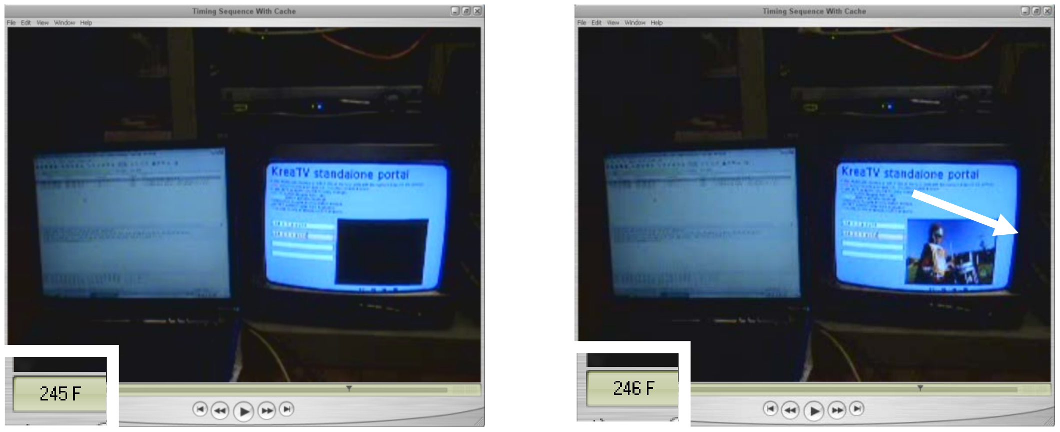
4th Transition showing TV Channel Indicator Moves to Channel 2 @ Frame 219

Figure 3.15 Middle Transitions of MSTB Example from SD to HD

The MSTB has an HD LED to indicate when it is processing high definition videos when the LED is ON. Figure 3.16 shows this activity with the 5th transition indicating that the STB is processing HD video by frame 234. There is an overlapping period of time in which packets are received for both the original channel 1 and the new channel 2. Figure 3.16 goes on to show the 6th transition with the new HD channel fully displayed by frame 246. The channel change process is finally completed a short time later when the last UDP packet from the initial channel 1 is received as will be shown in the upcoming Wireshark data presentation.



5th Transition showing STB High Definition LED Lit @ Frame 234



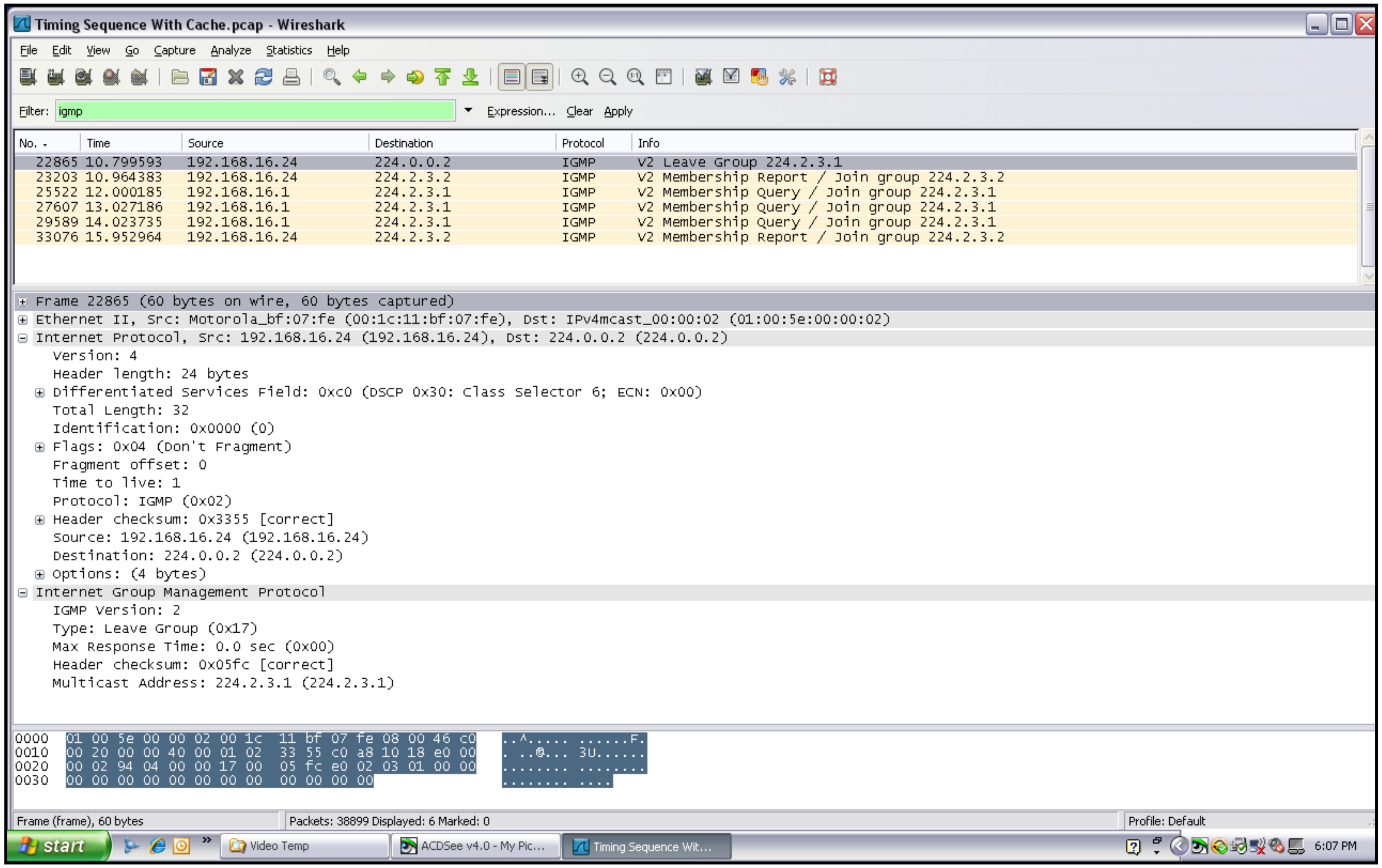
6th Transition showing TV Begins Channel 2 Display @ Frame 246

Figure 3.16 Final Transitions of MSTB Example from SD to HD

In the exact manner documented in the IGMP section of Chapter I, the messaging captured by the Wireshark program shows that the MSTB and the LHR are properly communicating using the IGMP protocols. Figure 3.17 shows all of the IGMP messages captured during this channel change sequence beginning with the first Leave message from the MSTB with 192.168.16.24 as its source IP address. This Leave message was the one that was noted by the 3rd transition at frame 210. The layer 3 device is the LHR with IGMP Querying capabilities for this trial, and it sends out several Membership Report messages as expected followed by the important Join message sent out by the MSTB.

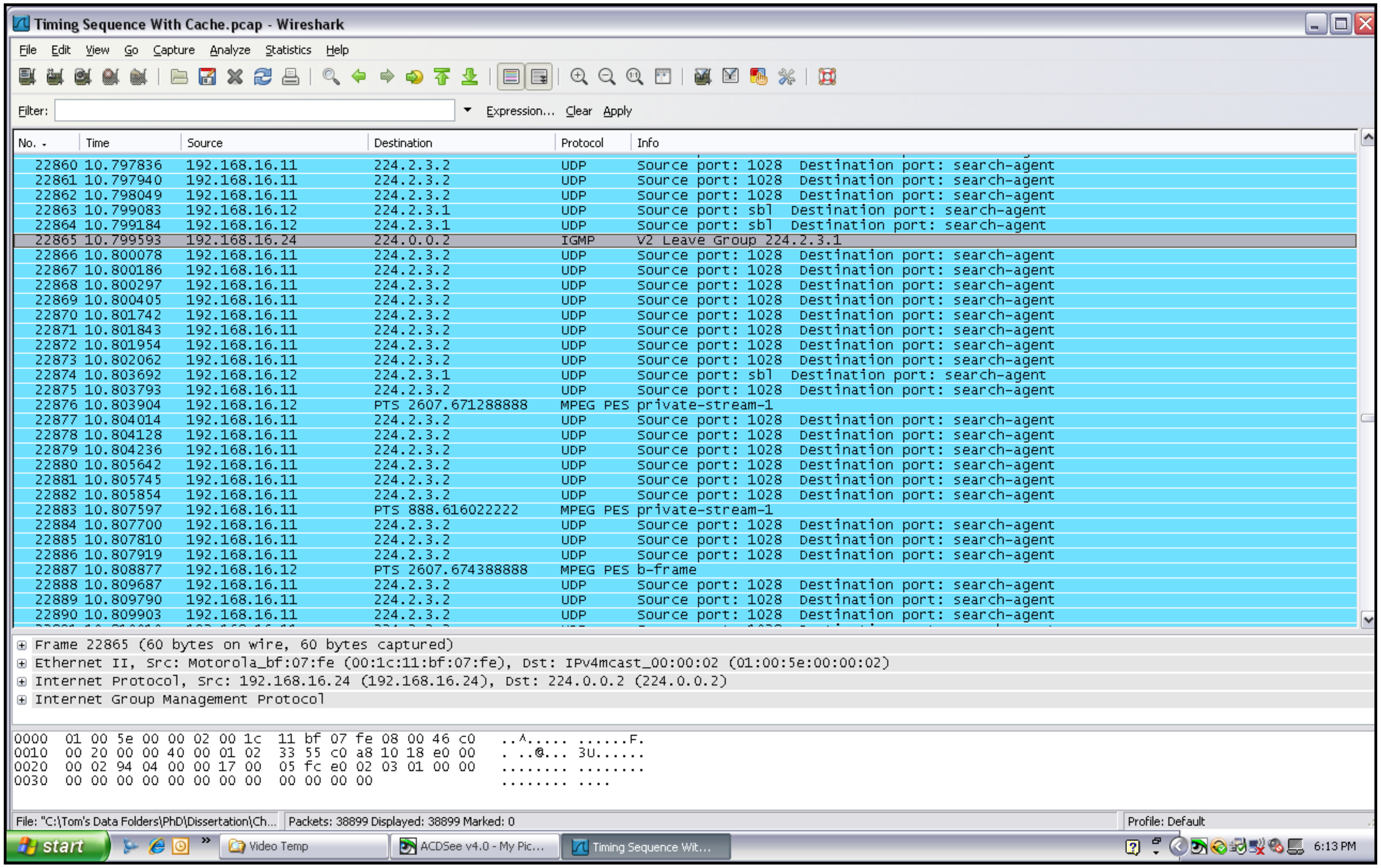
Wireshark on PC #4 was capturing all packets received at its interface, but it normally filtered the display to show only IGMP packets. Figure 3.18 shows the captured Wireshark display without the IGMP filter to show all of the packets captured during the flow. Figure 3.18 displays the moment when the first IGMP Leave message from the MSTB was captured. This figure clearly shows packets with destination addresses of both 224.2.3.1 and 224.2.3.2 indicating that both the original channel 1 and the cached channel 2 videos are streaming. The sources of these streams are the IP addresses associated with PC #2 and PC #1 respectively. This flow also shows one of the periodic B-Frames was received along with the UDP stream and the control messages. The Wireshark capture file was later scrolled down to show when the last of the UDP packets was received for the original SD channel 1 to indicate the formal conclusion of this channel change trial.

Another trial using the same STB with the same objective to change from an existing SD to a new HD channel was made, but this one did not have the new HD channel being cached. The monitoring functions were put into place for this trial and another data set was captured for this non-cached example.


Timing Sequence With Cache.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter: igmp
Expression... Clear Apply

```
No.   Time         Source          Destination   Protocol  Info
22865 10.799593    192.168.16.24   224.0.0.2     IGMP      V2 Leave Group 224.2.3.1
23203 10.964383    192.168.16.24   224.2.3.2     IGMP      V2 Membership Report / Join group 224.2.3.2
25522 12.000185    192.168.16.1    224.2.3.1     IGMP      V2 Membership Query / Join group 224.2.3.1
27607 13.027186    192.168.16.1    224.2.3.1     IGMP      V2 Membership Query / Join group 224.2.3.1
29589 14.023735    192.168.16.1    224.2.3.1     IGMP      V2 Membership Query / Join group 224.2.3.1
33076 15.952964    192.168.16.24   224.2.3.2     IGMP      V2 Membership Report / Join group 224.2.3.2

⊞ Frame 22865 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Motorola_bf:07:fe (00:1c:11:bf:07:fe), Dst: IPv4mcast_00:00:02 (01:00:5e:00:00:02)
⊟ Internet Protocol, Src: 192.168.16.24 (192.168.16.24), Dst: 224.0.0.2 (224.0.0.2)
    Version: 4
    Header length: 24 bytes
  ⊞ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00)
    Total Length: 32
    Identification: 0x0000 (0)
  ⊞ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 1
    Protocol: IGMP (0x02)
  ⊞ Header checksum: 0x3355 [correct]
    Source: 192.168.16.24 (192.168.16.24)
    Destination: 224.0.0.2 (224.0.0.2)
  ⊞ Options: (4 bytes)
⊟ Internet Group Management Protocol
    IGMP Version: 2
    Type: Leave Group (0x17)
    Max Response Time: 0.0 sec (0x00)
    Header checksum: 0x05fc [correct]
    Multicast Address: 224.2.3.1 (224.2.3.1)

0000  01 00 5e 00 00 02 00 1c  11 bf 07 fe 08 00 46 c0   ..^..... ......F.
0010  00 20 00 00 40 00 01 02  33 55 c0 a8 10 18 e0 00   . ..@... 3U......
0020  00 02 94 04 00 00 17 00  05 fc e0 02 03 01 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

Frame (frame), 60 bytes    Packets: 38899 Displayed: 6 Marked: 0    Profile: Default
```

Figure 3.17 IGMP Packet Flow During the MSTB Example



Timing Sequence With Cache.pcap - Wireshark

```
No.   Time      Source          Destination        Protocol  Info
22860 10.797836 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22861 10.797940 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22862 10.798049 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22863 10.799083 192.168.16.12   224.2.3.1          UDP       Source port: sbl  Destination port: search-agent
22864 10.799184 192.168.16.12   224.2.3.1          UDP       Source port: sbl  Destination port: search-agent
22865 10.799593 192.168.16.24   224.0.0.2          IGMP      V2 Leave Group 224.2.3.1
22866 10.800078 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22867 10.800186 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22868 10.800297 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22869 10.800405 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22870 10.801742 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22871 10.801843 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22872 10.801954 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22873 10.802062 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22874 10.803692 192.168.16.12   224.2.3.1          UDP       Source port: sbl  Destination port: search-agent
22875 10.803793 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22876 10.803904 192.168.16.12   PTS 2607.671288888 MPEG PES  private-stream-1
22877 10.804014 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22878 10.804128 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22879 10.804236 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22880 10.805642 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22881 10.805745 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22882 10.805854 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22883 10.807597 192.168.16.11   PTS 888.616022222  MPEG PES  private-stream-1
22884 10.807700 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22885 10.807810 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22886 10.807919 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22887 10.808877 192.168.16.12   PTS 2607.674388888 MPEG PES  b-frame
22888 10.809687 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22889 10.809790 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent
22890 10.809903 192.168.16.11   224.2.3.2          UDP       Source port: 1028  Destination port: search-agent

⊞ Frame 22865 (60 bytes on wire, 60 bytes captured)
⊞ Ethernet II, Src: Motorola_bf:07:fe (00:1c:11:bf:07:fe), Dst: IPv4mcast_00:00:02 (01:00:5e:00:00:02)
⊞ Internet Protocol, Src: 192.168.16.24 (192.168.16.24), Dst: 224.0.0.2 (224.0.0.2)
⊞ Internet Group Management Protocol

0000  01 00 5e 00 00 02 00 1c  11 bf 07 fe 08 00 46 c0   ..^..... ......F.
0010  00 20 00 00 40 00 01 02  33 55 c0 a8 10 18 e0 00   . ..@... 3U......
0020  00 02 94 04 00 00 17 00  05 fc e0 02 03 01 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00               ........ ....

File: "C:\Tom's Data Folders\PhD\Dissertation\Ch...  Packets: 38899 Displayed: 38899 Marked: 0  Profile: Default
```

Figure 3.18 Full Packet Flow During the MSTB Example

Data from both the non-cached and cached trials was then put into the timing sequences shown on Tables 3.2 and 3.3. In the non-cached timing sequence shown on Table 3.2, each of the individual transitions from the visual recording and the packet flow are listed with the corresponding frame number or Wireshark timestamp. Event #3 correlates between the frames and the timestamps since both are known for this first IGMP Leave packet sent out by the MSTB. These times were then converted into a running Test Time set to begin at Event #1 when the Remote Control channel change request was made from channel #1 to channel #2. The Overview section at the bottom categorizes the individual items into applicable groups. An identical process was performed for the cached example noted earlier, and the two timing sequences can be compared to show the differences made by caching the new HD video channel.

Table 3.3 shows transition frame numbers for the events associated with the visual recording that are identical to those presented in Figures 3.14-3.16 earlier. By re-setting Event #1 to be the starting frame, the other transition frame numbers could be used to determine how many frames from the starting frame each of those events occurred. These frame calculations were then converted to Test Time using the 0.04 seconds per frame reference. Event #3 indicated the moment when the first IGMP packet appeared on the Wireshark display at transition frame 210 which was 12 frames or 480 ms from the starting frame. The Wireshark data on Figures 3.15 and 3.16 indicates that Event #3 occurred approximately at timestamp 10.800 seconds, but Event #3 was already known to have occurred at Test Time 0.480 seconds. This knowledge enables all of the timestamps to be converted to Test Times as needed for Events #5, #6, and #9. This same method was used to determine the Test Times for the non-cached trial on Table 3.2.

Using Motorola 1216KK Set Top Box
Channel 1 Standard Definition on stream 224.2.3.1
Channel 2 High Definition on stream 224.2.3.2

Condition: Non-Caching of Next Channel Selection

Event #	**Description**	**Transition Frame #**	**# Frames from Start**	**Wireshark Timestamp**	**Wireshark Difference**	**Test Time**
0	TV Display of Ch 1 in Progress					< 0.000
1	Remote Control Change to Ch 2 Request	334	0			0.000
2	TV Display of Ch 1 Stopped, Screen Blank	339	5			0.200
3	IGMP Leave Ch 1 Packet From STB Sent	342	8	10.275	0.000	0.320
4	TV Channel Indicator Moves from Ch 1 to Ch 2	345	11			0.440
5	IGMP Join Ch 2 Packet From STB Sent			10.446	0.171	0.491
6	First Ch 2 UDP Packet Received			11.646	1.372	1.692
7	STB High Definition Status LED Lit	403	69			2.760
8	TV Display of Ch 2 Begins	411	77			**3.080**
9	Last Ch 1 UDP Packet Received			13.520	3.246	3.566

Overview Section

# From-To	**Function**	**Duration**
1-5	Command Processing	0.491
6	Network Delay and IGMP Processing	1.201
7-8	STB Processing	1.388
1-8	Total Channel Change Time	**3.080**

⇦ **Target Area for Improvement by this Research**

y Ti

Using Motorola 1216KK Set Top Box
Channel 1 Standard Definition on stream 224.2.3.1
Channel 2 High Definition on stream 224.2.3.2

Condition: Caching of Next Channel Selection

Event #	**Description**	**Transition Frame #**	**# Frames from Start**	**Wireshark Timestamp**	**Wireshark Difference**	**Test Time**
0	TV Display of Ch 1 in Progress					< 0.000
1	Remote Control Change to Ch 2 Request	198	0			0.000
2	TV Display of Ch 1 Stopped, Screen Blank	204	6			0.240
3	IGMP Leave Ch 1 Packet From STB Sent	210	12	10.800	0.000	0.480
4	TV Channel Indicator Moves from Ch 1 to Ch 2	219	21			0.840
5	IGMP Join Ch 2 Packet From STB Sent			10.964	0.164	0.644
6	Ch 2 UDP Packets Already Being Sent by Caching			10.966	0.166	0.646
7	STB High Definition Status LED Lit	234	36			1.440
8	TV Display of Ch 2 Begins	246	48			**1.920**
9	Last Ch 1 UDP Packet Received			14.024	3.224	3.704

<u>Overview Section</u>

# From-To	**Function**	**Duration**
1-5	Command Processing	0.644
6	Network Delay and IGMP Processing	0.002
7-8	STB Processing	1.274
1-8	Total Channel Change Time	**1.920**

⇦ Notice Improvement for Target Area with Caching

orat

The Overview section grouped the events listed into the functional processes that were established in Chapter I for the Applicable Delays by Classification of Table 1.5. The Network Delay and IGMP Processing functions for the non-cached example in Table 3.2 show a duration of 1.201 seconds. This is expected based on the fixed 600 ms Network Delay for each direction of the flow plus the IGMP Processing time typically required. Comparing this to the same functionality on Table 3.3 clearly shows that the multicast caching of the new HD video channel reduced this delay to the negligible duration of 0.002 seconds needed for the next UDP packet to appear with the new video channel.

From the Ping testing presented on Figures 3.6-3.8 it was shown that the expected time lag associated with the Network Delay fixed by the NetDisturb software on PC #3 was an average 1198 ms. Table 1.5 listed the typical IGMP Processing delays to be approximately 20 ms. The experimental results of the non-cached trial show the Network Delay and IGMP Processing functions combine for a total delay of 1201 ms. Given the Ping testing results it can be concluded that the IGMP Processing delay for this trial was only 3 ms. In actual practice LHRs are typically responsible for ongoing IGMP Processing for the many users that are connected to the network at any given moment. The data in Table 1.5 was based on observed delays for actual deployments. With many users simultaneously engaged in IGMP multicasting, the LHR IGMP Tables and processes are normally kept at a busy level for the ongoing activity. By contrast these laboratory tests did not have a large number of users connected at any given moment due to the expense required to purchase numerous STBs. Considering the relatively low workload present during these lab trials, it is reasonable to expect the IGMP Processing delays to be lower than those documented by previous research from field observations. Also the data from Table 1.5 on IGMP Processing was a baseline approximation, because the actual delays are dependent on the equipment deployed, the configuration design, and the background activity level.

The conclusion drawn from these timing sequences is that the multicast caching of the next channel to be selected resulted in a CCT reduction equivalent to two times the Network Delay plus the IGMP Processing time required by the LHR for these two trials. This was exactly in line with the expectations established earlier, but it still remained necessary to verify if this was consistent behavior under a variety of conditions.

3.4 Results of Laboratory Testing

Table 3.1 showed the list of laboratory test types that were performed with the specific functional conditions present as indicated. These fifteen different test types were run thirty times each to meet CLT conditions for a total of 450 individual channel change trials. Each trial was filmed and analyzed as covered in the last section to determine the overall CCT per trial. Table 3.4 is an example of one of these test types showing the results of all thirty trials for the MSTB to SD channel changes under the minimal configuration. The results for each of the other fourteen trial types look similar in structure to Table 3.4, and they can be found in Appendix B. Note that all of these results reflect the direct CCT values indicated by the recordings, so the margin of error for all of the results found in this section is plus or minus 0.04 seconds. An examination of Table 3.4 shows that of the thirty trials under this test type the minimum CCT lasted 1.68 seconds or 42 frames and the maximum CCT was 2.00 seconds or 50 frames. Note again that each trial indicates a specific channel change process that was filmed and analyzed for the beginning and ending transitions to determine that individual CCT.

Trial #	# Frames	Time (sec)
1	48	1.92
2	42	1.68
3	49	1.96
4	48	1.92
5	42	1.68
6	46	1.84
7	42	1.68
8	47	1.88
9	50	2.00
10	45	1.80
11	48	1.92
12	43	1.72
13	48	1.92
14	42	1.68
15	45	1.80
16	44	1.76
17	48	1.92
18	42	1.68
19	45	1.80
20	42	1.68
21	48	1.92
22	42	1.68
23	48	1.92
24	48	1.92
25	45	1.80
26	48	1.92
27	45	1.80
28	45	1.80
29	45	1.80
30	49	1.96
Total	1369	54.76
Mean	45.63	1.83
STDev	2.62	0.10
Minimum	42	1.68
Maximum	50	2.00

Table 3.4 Results for all Trials of the MSTB to SD Channel Change with Minimal Configuration

The sample mean time for the data in Table 3.4 was calculated to be 1.83 seconds with a sample standard deviation of 0.10 seconds. Application of the 68-95-99.7 Rule yields the following confidence levels about the associated accuracy intervals:

68% Confident	**1.73 sec < Population Mean < 1.93 sec**
95% Confident	**1.63 sec < Population Mean < 2.03 sec**
99.7% Confident	**1.53 sec < Population Mean < 2.13 sec**

The similar analytical results that were made for each test type were then combined into the summary results found in Table 3.5 reflecting all lab efforts.

	All Times Recorded in Seconds			
Functionality Tested	**Min Time**	**Max Time**	**Mean Time**	**STDev**
A125 STB to SD Using Minimal Config	1.16	1.52	1.40	0.09
A130 STB to SD Using Minimal Config	1.88	2.24	2.06	0.11
A130 STB to HD Using Minimal Config	1.92	2.36	2.13	0.13
MSTB to SD Using Minimal Config	1.68	2.00	1.83	0.10
MSTB to HD Using Minimal Config	1.72	2.12	1.92	0.11
A125 STB to SD via Network Without Cache	2.52	3.04	2.76	0.14
A130 STB to SD via Network Without Cache	3.00	3.56	3.32	0.14
A130 STB to HD via Network Without Cache	3.16	3.72	3.40	0.13
MSTB to SD via Network Without Cache	2.88	3.28	3.07	0.10
MSTB to HD via Network Without Cache	2.92	3.36	3.13	0.12
A125 STB to SD via Network With Cache	1.32	1.80	1.53	0.14
A130 STB to SD via Network With Cache	1.92	2.28	2.13	0.11
A130 STB to HD via Network With Cache	1.96	2.44	2.17	0.13
MSTB to SD via Network With Cache	1.72	2.12	1.86	0.10
MSTB to HD via Network With Cache	1.76	2.20	1.95	0.12

Table 3.5 Summary Results Combined for all Trials

Notice that all of the sample standard deviations fall between 0.09 and 0.14 seconds, so the general accuracy ranges will be similar to those found previously for the data of Table 3.4. But rather than trying to dissect individual conclusions from these combined results, it was considerably easier to see differences and trends by analyzing certain test types by themselves or against opposing test types. Table 3.6 provides an example of this showing only the summary results of all minimal configuration trials. Note again that all minimal configuration trials performed were conducted without multicast caching of the next channel selection. From Table 3.6 it is evident that the STBs have different performance characteristics in terms of CCT, but the mean time for each of these five minimal configurations provides an excellent baseline for later comparison using full configurations to evaluate the effectiveness of specific CCTs. With these minimal configuration results the best attainable CCTs for these specific STBs under given circumstances are now known.

	All Times Recorded in Seconds			
Functionality Tested	**Min Time**	**Max Time**	**Mean Time**	**Standard Deviation**
A125 STB to SD Using Minimal Config	1.16	1.52	1.40	0.09
A130 STB to SD Using Minimal Config	1.88	2.24	2.06	0.11
A130 STB to HD Using Minimal Config	1.92	2.36	2.13	0.13
MSTB to SD Using Minimal Config	1.68	2.00	1.83	0.10
MSTB to HD Using Minimal Config	1.72	2.12	1.92	0.11

Table 3.6 Summary Results for all Minimal Configuration Trials

The primary objective of the laboratory testing was to determine the effectiveness of multicast caching of the next channel selected, so Table 3.7 was compiled to directly show the results of non-caching versus caching trials. In each STB case the mean time reduction was within a few milliseconds of the expected results based on the 600 ms Network Delay for each direction plus the IGMP Processing time. Note that the overall mean time reduction was 1.21 seconds across all five comparison cases. These five comparison cases represent a total of 150 trials (5 x 30) of non-caching versus 150 trials of caching to provide a solid case of statistical significance proving the effectiveness of multicast caching.

All Times Recorded in Seconds Functionality Tested	 Mean Time
A125 STB to SD via Network Without Cache	2.76
A125 STB to SD via Network With Cache	1.53
Time Reduction	1.22
Percent Reduction	44.37%
A130 STB to SD via Network Without Cache	3.32
A130 STB to SD via Network With Cache	2.13
Time Reduction	1.19
Percent Reduction	35.76%
A130 STB to HD via Network Without Cache	3.40
A130 STB to HD via Network With Cache	2.17
Time Reduction	1.23
Percent Reduction	36.26%
MSTB to SD via Network Without Cache	3.07
MSTB to SD via Network With Cache	1.86
Time Reduction	1.22
Percent Reduction	39.64%
MSTB to HD via Network Without Cache	3.13
MSTB to HD via Network With Cache	1.95
Time Reduction	1.17
Percent Reduction	37.59%
Overall Mean Time Reduction	**1.21**
Overall Mean Percent Reduction	**38.72%**

Table 3.7 Summary Results for all Non-Caching Versus Caching Trials

Note that the standard deviations of these sample types are substantially lower than the mean time reductions, so the likelihood of significant time savings near the levels indicated is extremely high. In the case of these lab tests the one way Network Delay of 600 ms led directly to a CCT time savings of two times the Network Delay plus the relatively minor delays associated with the IGMP Processing. For these STBs and conditions this equated to an overall mean time reduction of 38.72% in CCTs. As noted earlier this 600 ms delay was fixed to correspond to previously documented laboratory experiments [1][5][7]. The standard deviations in the range of 0.10 seconds against time reductions more than ten times that amount lead to a confidence level of around 99 percent that the actual time reductions will be between 0.9 and 1.5 seconds per CCT for a 600 ms Network Delay. Actual Network Delays experienced in field applications vary widely from case to case, but previous researchers indicate they are frequently seen in the 100-200 ms range for one way delays (200-400 ms round trip) [39][76]. Given the results demonstrated by this lab testing, the applicable CCT reductions attainable by multicast caching of the next channel selection can in fact be confidently predicted to be the same as theorized earlier:

CCT Reduction ≅ (2 x Network Delay) + (IGMP Processing)

This indicates that if a specific application has the average one way Network Delay of 150 ms, then the expected CCT reduction each time the next channel is successfully cached in advance would be approximately 320 ms depending on the IGMP Processing load [39]. This validated ability to confidently predict the effectiveness of this proposed technique represents one of the milestone contributions listed earlier for this research.

The laboratory data did go on to provide other conclusions that can be of further use on this project. Table 3.8 provides five more comparisons between the minimal configuration results and the cached results using the full configuration. By eliminating the Network Delay and IGMP Processing functions from the overall CCT, the results from caching compared well when pitted against the minimal configuration outcomes. In each case the caching data was only slightly higher than the best-attainable results from the minimal configurations. The mean increase from caching was only 60 ms higher than the minimal configuration for a 3.63% mean increase in CCT. Note that all cached trials had several more hops inserted with the inclusion of the GigE test set, Layer 2 switch, PC#3, and HG, so these higher times are expected over the minimal configurations.

All Times Recorded in Seconds	
Functionality Tested	**Mean Time**
A125 STB to SD Using Minimal Config	1.40
A125 STB to SD via Network With Cache	1.53
Time Increase	0.13
Percent Increase	9.62%
A130 STB to SD Using Minimal Config	2.06
A130 STB to SD via Network With Cache	2.13
Time Increase	0.07
Percent Increase	3.29%
A130 STB to HD Using Minimal Config	2.13
A130 STB to HD via Network With Cache	2.17
Time Increase	0.04
Percent Increase	2.01%
MSTB to SD Using Minimal Config	1.83
MSTB to SD via Network With Cache	1.86
Time Increase	0.03
Percent Increase	1.68%
MSTB to HD Using Minimal Config	1.92
MSTB to HD via Network With Cache	1.95
Time Increase	0.03
Percent Increase	1.53%
Overall Mean Time Increase	**0.06**
Overall Mean Percent Increase	**3.63%**

Table 3.8 Summary Results for all Minimal Versus Caching Trials

But the fact that the cached CCT results approached those of the minimal configurations did further indicate the effectiveness of this method's implementation. If CCT reductions were made by caching but they happened to be significantly higher than those seen in the minimal configurations, then further analysis would be needed to determine the source of those unexpected delays. This was not necessary though since the 60 millisecond mean increase by caching over the minimal results was considered reasonable and expected.

Another comparison analysis was conducted to look into the differences in CCTs for SD versus HD conditions. This is not specific to the study of multicast caching, but it is of importance for modeling techniques that will be presented later. Table 3.9 shows the comparisons between SD and HD results for the six cases studied where that was the only difference involved in the testing. Note again that the A125 STB does not appear in these results because of its inability to process HD video. The results of Table 3.9 indicate a mean increase of 70 ms directly attributable to the need to handle HD video over SD video. The A130 STB had significantly better performance over that of the MSTB, but this was not unexpected due to the many added features and capabilities inherent in the Motorola devices. One important aspect to note about this however is that different STBs have significantly diverse CCTs associated with the processing of HD videos. The other principle of note is the relatively low importance of this aspect when considering the overall CCT as indicated by the mean increase of only 3.19% attributed to the HD video. These concepts will be considered again in later sections when multicast models are being analyzed for effectiveness.

The results documented by this lab testing clearly demonstrated that multicast caching reduced CCT by two times the Network Delay plus the IGMP Processing time.

All Times Recorded in Seconds **Functionality Tested**	**Mean Time**
A130 STB to SD Using Minimal Configuration	2.06
A130 STB to HD Using Minimal Configuration	2.13
Time Increase	0.06
Percent Increase	2.97%
MSTB to SD Using Minimal Configuration	1.83
MSTB to HD Using Minimal Configuration	1.92
Time Increase	0.10
Percent Increase	5.26%
A130 STB to SD via Network Without Cache	3.32
A130 STB to HD via Network Without Cache	3.40
Time Increase	0.08
Percent Increase	2.49%
MSTB to SD via Network Without Cache	3.07
MSTB to HD via Network Without Cache	3.13
Time Increase	0.05
Percent Increase	1.65%
A130 STB to SD via Network With Cache	2.13
A130 STB to HD via Network With Cache	2.17
Time Increase	0.04
Percent Increase	1.69%
MSTB to SD via Network With Cache	1.86
MSTB to HD via Network With Cache	1.95
Time Increase	0.09
Percent Increase	5.10%
Overall Mean Time Increase	**0.07**
Overall Mean Percent Increase	**3.19%**

Table 3.9 Summary Results for all SD Versus HD Trials

These laboratory tests further document that multicast caching approaches the best attainable results for these given conditions. This is not to say that other improvements are not attainable through enhanced STB processing, dynamic I-Framing, or the many other techniques previously discussed. But rather if those improvements are accounted for separately, then the multicast caching method successfully achieves the reductions documented without arbitrarily introducing other delays. The laboratory results further documented that different STBs handle HD video in substantially different ways, but that the overall introduction of HD video did not dramatically increase CCTs during these trial cases.

CHAPTER IV

NEW CHANNEL CHANGE PREDICTION MODELS FOR ANALYSIS

Chapter III was devoted to showing what the reductions in CCT would be if the proposed system was able to correctly pre-stream the next channel to be selected. Another major objective of an effective multicast caching system is to correctly identify the next channel selected by a user with a high probability. The system would not be considered effective overall if the next channel selected was rarely included in the pre-streamed cache. Several models were developed to effectively identify the channels to be included in the multicast cache over time. Since this system does not currently exist, it was necessary to either build it for field testing or to create computer simulations of the system in operation. Some thought was given to actually building the system in a layer 2 device that was based on open-source software, but the expense and complexity of completing that task proved to be a large obstacle to overcome. Even if this open-source testing device were to be actually created, a unique IPTV environment with numerous users willing to participate in the testing would have to be secured and managed. In the end it became clear that a computer simulation approach made more sense for the purposes of this research project.

For clarity it is worth restating that the lab testing previously presented showed what the CCT reductions would be if the system correctly pre-streamed the next channel to be selected. Throughout this research the currently updated list of channels to be pre-streamed is considered to be the “multicast cache” of channels.

A series of computer simulations were developed to test the effectiveness of the various models that were created to identify the channels to be selected for multicast caching. The simulations were designed to determine the likelihood of successfully caching the next channel to be selected, resulting in significant CCT reductions as documented by the lab testing. Whenever the system incorrectly identifies the next channel to be selected then there would be no corresponding reduction in CCT, also known as the zap time or channel change latency [44][70][73][75].

The high level steps that are required for successful computer simulation of multicast caching are the following:

- **Create New Channel Change Prediction Models for Analysis**
- **Develop Sample Data Sets**
- **Simulate the Models Using the Sample Data Sets as Inputs**

These high level steps are presented in order over the next three chapters. This chapter will present the different models developed during this research to simulate multicast caching of pre-selected channels to improve IPTV CCT performance. Chapter V is devoted to the development of the sample data sets, and the results of the computer simulations will then be covered in detail in Chapter VI. For now the important principle to consider is that each model represents a different view of the tradeoffs between complexity and performance. Some of the models are quite simplistic and do not yield optimal caching results, but they are extremely easy to implement. Other models provide better caching performance, but they are more complex to develop and maintain. Another option put forth by this research is that a potential system implementation could be made using a mixed model approach. In this case the system would have two or more models running simultaneously, and it would compare the resulting cumulative weighted probabilities of the cache selections to determine which model was optimal for use at that specific moment in time.

These models were developed for functional simulation through programs written using the Microsoft Visual C++ .NET development platform. The algorithm for each model will be presented and discussed, and the C++ source code for each model can be found in Appendix D. Many of the variable conventions, error checking routines, output programming, and other required C++ functions found in the appendix will be ignored in this chapter to focus on the core mechanics of each algorithm.

4.1 Tracking the Channel History

A key prerequisite for each model to function is the availability of historical data related to recent channel change behavior. For this purpose a channel change history buffer was created to track the most recent channel changes in order back to a specified limit for each output port on the device. This buffer is used in conjunction with the known data rate for each channel to provide critical information that is used in determining the channels to be cached. Each time a new channel is selected, it is placed into the most recent position while all other previous changes are incremented down one position in the buffer. This research is not focused on the dwell time, or how long each channel is viewed, and no tracking of this metric is included in this current proposal.

With this channel change history buffer it is possible to calculate the past probabilities of each channel. This can be done by adding up the number of times each channel appears in the buffer and dividing by the total number of entries in the buffer. These probabilities could then be used directly in calculations to fill the multicast cache with channels that are likely to be selected next provided that the basic historical trends are continued. However, many different areas of communications, such as packet addressing, make use of time-weighting algorithms to provide a means of giving the most recent events higher priority than older events. By adjusting the weighting parameters, the balance between maintaining a useful historical record and identifying new trends can potentially lead to improvements in channel change prediction over just the raw probabilities calculated directly from the history buffer.

4.2 Time-Weighting the Channel History for Ranking

The widespread usage of many common packet-based protocols, most notably TCP, include methods for handling error control. To prevent network congestion these methods frequently involve time-weighting techniques for improving the analysis of tell-tale metrics. These methods depend upon ongoing measurements of the round-trip time (RTT) for packet handshaking processes and adaptive improvements of this measurement give higher weight to the most recent measurements. The smooth round-trip time (SRTT) estimate is frequently calculated as shown in Equation 4-1 [98-99]. In this equation, K indicates the applicable segment and $\boldsymbol{\alpha}$ is a constant smoothing parameter with a value in the range ($0 < \boldsymbol{\alpha} < 1$). Since both $\boldsymbol{\alpha}$ and ($1 - \boldsymbol{\alpha}$) are less than one, each successive term in this equation is smaller than the previous term.

$$\mathbf{SRTT(K+1) = (1 - \alpha)RTT(K + 1) + \alpha(1 - \alpha)RTT(K) + \alpha^2(1 - \alpha)RTT(K - 1) + \ ... \ + \alpha^K(1 - \alpha)RTT(1)}$$

Equation 4-1

Error control advancements developed by Van Jacobson in the late 1980’s expanded on the exponential smoothing techniques for calculating SRTT. Jacobson’s algorithms calculated a retransmission timeout (RTO) that resulted in an exponential back-off of retransmissions [98-99]. The implementation of this algorithm meant that the RTO would grow exponentially for each successive retransmission, so devices would wait for increasing periods of time to allow the network to recover from congestion.

Numerous other exponential weighting algorithms have been developed for a wide range of other purposes, including many physical science and financial applications [100-103]. The common theme in these applications was the need to provide higher significance to more recent events. This theme was clearly present during this research as well, since the determination of channels to be placed into a multicast cache would naturally change over time. If time-weighting was not considered at all, then a user that finds a new favorite channel that had not been viewed before would likely have to wait for prohibitively long periods of time before the new behavior was taken into account. Other common conditions could easily end up with the same negative result, such as when providers modify channel lineups.

The mathematical mechanics involved in time-weighting channel change history queues are slightly different than those described above for network congestion. There is no need in the case of multicast caching for determining average times or comparing delays. The approach taken in this research was to exponentially modify the weight associated with each channel change such that each entry down the history queue had a slightly reduced value for use in overall calculations. The following exponential weight formula was applied to an array with the same quantity of sequential entries that are in the historical buffer:

$$W_j = \alpha^j$$ **Equation 4-2**

where **W** was the weight, ***j*** was the change queue entry number, and **α** was the constant weighting parameter with a value (0 < **α** < 1). Analyses on how to set the maximum historical buffer limit **N** will be presented in the next chapter, but for now consider the case with the limit fixed at 2000 entries. The constant weighting parameter (**α**) defines a tradeoff between not providing enough and providing too much emphasis on the most recent channel changes. Smaller values of **α** result in a greater weight being applied to the more recent channel changes, but if **α** is set too low then only the last few channel changes will dictate the cache results. If on the other hand **α** is set too high, then the most

α Parameter =	**0.5**	**0.9**	**0.98**	**0.99**	**0.995**	**1**
Entry *j*	**Resulting Weight Listings**					
1	0.500000	0.900000	0.980000	0.990000	0.995000	1
2	0.250000	0.810000	0.960400	0.980100	0.990025	1
3	0.125000	0.729000	0.941192	0.970299	0.985075	1
4	0.062500	0.656100	0.922368	0.960596	0.980150	1
5	0.031250	0.590490	0.903921	0.950990	0.975249	1
6	0.015625	0.531441	0.885842	0.941480	0.970373	1
7	0.007813	0.478297	0.868126	0.932065	0.965521	1
8	0.003906	0.430467	0.850763	0.922745	0.960693	1
9	0.001953	0.387420	0.833748	0.913517	0.955890	1
10	0.000977	0.348678	0.817073	0.904382	0.951110	1
17	0.000008	0.166772	0.709322	0.842943	0.918316	1
110	0.000000	0.000009	0.108360	0.331033	0.576154	1
380	0.000000	0.000000	0.000463	0.021947	0.148857	1
573	0.000000	0.000000	0.000009	0.003155	0.056575	1
1151	0.000000	0.000000	0.000000	0.000009	0.003122	1
2000	0.000000	0.000000	0.000000	0.000000	0.000044	1

Table 4.1 Exponential Weight Listings for Sample α Parameters

recent data is not weighted heavily enough to drive cache results in a timely manner for new viewer behavior. Table 4.1 illustrates the weight tradeoffs for various values of **α**.

The exponential weights in Table 4.1 clearly fall off quickly for the first two listings with **α** = 0.5 and **α** = 0.9, while for **α** = 0.995 weights fall off only 5% throughout the first 10 entries. The last column with **α** = 1 is equivalent to the non-weighted raw data found in the original historical buffer. So with **α** = 1, the previous 2000th channel change would carry the exact same weight as the most recent 1st change. This table goes on to show the buffer entry associated with the **α** parameter when the weight value first drops below 0.00001 or one thousandth of one percent. For the purposes of illustration, this level was used to show when the entry contributions have become negligible due to the low weight values. From these observations it is evident that a relatively high level must be used for the **α** parameter to retain some degree of significance for earlier changes. Following experimental trials, the decision was made to use a constant value of **α** = 0.98 during the simulation phases of this research because it fell in the middle of the tradeoff region between rapidly moving a new favored channel up the probability list while still allowing for the contribution of several hundred channel changes in the final results. The weighted probabilities for each channel are determined by Equation 4-3 below:

$$\chi_i = \frac{\sum_{j=1}^{N} (\alpha^{j} \text{ for buffer entries of channel } i)}{\sum_{j=1}^{N} \alpha^{j}}$$

Equation 4-3

where $\boldsymbol{X_i}$ is the weighted probability of channel $\boldsymbol{i}$, **N** is the historical buffer size, and **α** is the constant weighting parameter for each historical buffer position $\boldsymbol{j}$. Later sections will discuss how the actual field implementation would use feedback mechanisms and logic criteria to adjust the **α** parameter along with the size **N** of the channel change queue to dynamically react to different circumstances over time.

4.3 Shared Routines

The following terms are widely used in the discussions to follow:

Channel Change History Buffer: This is a 1 x N array that sequentially tracks in order the recent history of channel change selections made up to the limit of N changes. For the purposes of the simulations, N was fixed at a constant 2000 as discussed in Chapter V.

Channel List Arrays: The Channel List arrays consist of three separate 1 x 150 arrays for the remaining available channel numbers, bandwidths of each channel, and the time-weighted channel probabilities respectively.

Confirmed Channel Arrays: These consist of three 1 x 150 arrays containing the channel number, bandwidth, and time-weighted probability of all channels that have been confirmed into the current multicast cache for pre-streaming.

Sample Data Sets: A series of simulated channel activities was made to generate example data representing snap shots of expected input conditions that were then input into each model for comparative performance analysis. There were a total of 18 sample data sets created as described in the next chapter, and each of these data sets were input under three different conditions to each model for a total of 54 evaluations.

Input Data Set: This refers to the specific data that is being evaluated for each run of a model. In actual field use, the input data set would directly be the channel change history data and related current information as updated after each channel change. During this research the input data set was equivalent to the specific sample data set that was currently selected for evaluation.

The basic pattern and flow of the caching models is as follows in Figure 4.1:

- **Load the channel list arrays with the desired input data set**
- **Determine the current channel being viewed**
- **Determine the total bandwidth (BW) and available BW**
- **Perform the specific model algorithm on the various data arrays**
- **Load cache channel selections into the confirmed channel arrays**
- **Log results**

Figure 4.1 Basic Pattern Flow of the Caching Models

These high-level steps are performed on each model using a combination of shared routines and custom modeling routines. The shared routines used by all of the models to perform a variety of common tasks, such as data input and output, are detailed in Appendix C. The specific model algorithm is incorporated into each custom channel change prediction model routine.

The different models developed during this research are described in the next six sections that follow. The detailed code for each of these models can be found in Appendix D. The fundamental metric of interest here is the cumulative weighted probability of all the cached channels.

4.4 Probability-Only Model

Note that the sample data sets to be discussed in detail in the next chapter represent multiple simulated snap-shot views of data for different instants in time that would typically be present in the weighted probability queues. In actual implementation these probability queues would be routinely updated after each channel change of a fully functional device. During the following discussions on the various models note that the input data set has already been sorted from top to bottom based on the time-weighted probability of each available channel.

Of all the models developed during this research project the model with the least amount of complexity is the Probability-Only Model. The Probability-Only Model follows the basic pattern of flow outlined in Figure 4.1. Following the completion of the first three steps of the pattern the Channel List arrays have been populated, details about the current channel are known, and the current available bandwidth has been calculated.

The model works through the Channel List arrays from the top down always looking at the channel that currently has the highest probability, and the only criteria to be met to move a channel to the Confirmed Channels arrays is that the channel's bandwidth is less than or equal to the remaining available bandwidth. If a channel under analysis meets the criteria, then the channel is added to the Confirmed Channels arrays and removed from the Channel List arrays. If the channel bandwidth is higher than the remaining bandwidth, then the routine removes the channel from the Channel List arrays and begins processing the next channel in the array. The process is continued until the remaining bandwidth is less than the minimum channel bandwidth of 2 Mbps or until the Channel List arrays are fully emptied.

This model has the benefits of not being complex to develop, of having a minimal processor requirement to implement, and it has a reasonably effective caching success. As will be shown with the simulation results in Chapter VI, the Probability-Only Model does exhibit a moderately high probability of successfully caching the next channel to be selected. However, an intuitive view of this routine indicates that it sacrifices a

Channel	BW	Weighted Probability
21	18	0.10
84	2	0.09
17	4	0.08
67	2	0.07
55	9	0.06

Table 4.2 Example of Probability-Only Model Tradeoffs

significant degree of improved caching results for reduced complexity. Consider the example in Table 4.2 for the hypothetical condition where only 19 Mbps of remaining bandwidth are available for additional confirmed channel caching. The Probability-Only Model will simply pick the channel with the highest probability as long as it will fit into the remaining bandwidth. For this example the model would pick channel 21 first, but the 18 Mbps required for that channel means that no additional channels could then be selected. In this case the Probability-Only Model would result in an estimated 10% chance of the user picking this pre-streamed channel as his next channel choice. Note however, that all four of the other channels (84, 17, 67, and 55) would fit into the same 19 Mbps of available bandwidth for a cumulative weighted probability calculation of (0.09 + 0.08 + 0.07 + 0.06) = 0.3 or 30% as long as channel 21 is ignored. This would result in a channel cache that is three times more likely to be successful than the results obtained from the Probability-Only Model for this example. However, the modeling calculations required to perform the intuitive selection process that was just outlined are significantly more complex. The Probability-Only Model has the benefit of being an easily implemented fully functional model that could be useful on inexpensive applications or for devices with extremely low processing capabilities.

4.5 High Definition Model

The High Definition (HD) Model was developed to target HD video channels for caching in precedence over SD channels. The algorithmic process for the HD Model follows the basic pattern flow of the Probability-Only Model except that channels are sorted first by decreasing probability, and then they are resorted by decreasing bandwidth. The resulting Channel List arrays end up with the 18 Mbps channels at the top of the arrays in decreasing order of probability. These will be followed by the channels with a bandwidth of 9 Mbps, then those at 4 Mbps, and finally the 2 Mbps channels. The caching process is then completed in the same fashion as the Probability-Only Model until the available bandwidth limit is fully utilized. The resulting cache list will not typically have a large number of entries since it selects channels with the highest bandwidth first when filling the available bandwidth.

The simulation results for this model will be detailed in Chapter VI, but the cumulative weighted probabilities of the selected channels in the cache were significantly lower than the other models tested. It is therefore clear that this model would only be considered for use in applications that have a compelling need to select the highest bandwidth channels first.

4.6 Maximum Channels Model

The Maximum Channels Model is essentially the opposite of the High Definition Model in that preference is given to channels with the lowest bandwidth first. The algorithmic process of the Maximum Channels Model follows the HD Model, except that the Channel List array sorting is performed first by ordering the channels from the highest to the lowest probability followed by a resorting of all channels from lowest to highest bandwidth. The resulting Channel List arrays will then have all of the 2 Mbps channels at the top of the arrays in decreasing order of probability. These will be followed by the channels with a bandwidth of 4 Mbps, then those at 9 Mbps, and finally the 18 Mbps channels. The caching process is then performed in the same fashion as the previous models until the Confirmed Channel arrays fully utilize the available bandwidth.

The resulting output cache list from this model ends up with the largest possible number of channels since it selects channels with the lowest bandwidth first. However, the cumulative weighted probability of the cached channels tested well below the Probability Only Model in the simulations covered in Chapter VI. This can intuitively be expected since any channels with a highly weighted probability that use more than 2 Mbps are sorted down below all of the 2 Mbps channels.

4.7 Exhaustive Search Model

A computer program could theoretically use an exhaustive search technique to find the absolute best possible video cache selection for any given input data set. This may not be workable in real time due to the large number of calculations that must be performed using the expected processing power present at this layer of a network. This idea quickly ran into the reality of the huge number of caching combinations associated with 150 offered channels.

An attempt to reduce this complexity was made by considering only a lower available bandwidth option of 50 Mbps to reduce the possibilities to be calculated. If the currently viewed channel was at the minimum of 2 Mbps then there would only be a maximum of 48 Mbps available for caching. The maximum possible number of cached channels would then be 48 Mbps divided by a worst case of 2 Mbps or 24 maximum channels. The minimum number of channels would be determined by the condition with a currently viewed channel using 18 Mbps leaving 32 Mbps for caching. This would result in a minimum of 3 channels cached using up 31 Mbps of the available bandwidth with one channel at 18 Mbps, one at 9 Mbps, and one at 4 Mbps. Therefore the number of possible caching combinations with at least 3 channels and no more than 24 channels out of a possible 149 total channels would be calculated as follows:

Number of Possible Cache Combinations for the 50 Mbps Example

$$= \left(\frac{(149!)}{(3!)(146!)} \right) + \left(\frac{(149!)}{(4!)(145!)} \right) + \ldots + \left(\frac{(149!)}{(23!)(126!)} \right) + \left(\frac{(149!)}{(24!)(125!)} \right)$$

$$\cong 4 \times 10^{27} \textbf{ Combinations} \qquad \textbf{Equation 4-4}$$

The processing time required for this task makes this idea completely unworkable for the purposes of this project. Therefore any workable exhaustive search technique would have to look at a reduced number of possible combinations.

In this model, the maximum number of channels to be included in the cache for any single pass of the routine was limited to six based on sample testing that allowed for reasonably rapid results using the available research computers. The exhaustive search routine would then look through all combinations of 1 to 6 channels out of the 149 possibilities and select the combination of channels with the highest weighted probability of being selected to be included in the cache. Those channels would be removed from the Channel List arrays and if the remaining bandwidth permitted additional channels the process would be repeated for a second pass of up to six additional channels. If there was still any remaining bandwidth of 2 Mbps or higher then the routine was run again for a third time. The complete process allows for a total of up to 18 channels to be cached by this model. However these 18 cached channels are not necessarily the absolute optimum set of 18 channels due to the high number of combinations that would be required for a full 18 channel exhaustive search. Instead this represents the best six channels in terms of weighted probabilities, followed by the next best six channels, again followed by the next best six channels.

The maximum number of combinations to be checked by any single iteration of the Exhaustive Search Model can be calculated as follows:

Maximum Number of Combinations to be Checked per Iteration

$$= \left(\frac{(149!)}{(1!)(148!)} \right) + \left(\frac{(149!)}{(2!)(147!)} \right) + \ldots + \left(\frac{(149!)}{(5!)(144!)} \right) + \left(\frac{(149!)}{(6!)(143!)} \right)$$

$$\cong 1.43 \times 10^{10} \text{ Combinations} \qquad \textbf{Equation 4-5}$$

A comparison of this maximum number of combinations per iteration (1.43 x 10^{10}) with the total number of combinations required to implement the full optimal exhaustive search (4 x 10^{27}) indicates that this iterative approach is significantly more feasible to implement and simulate.

While the Exhaustive Search Model does not yield the absolute best case cache possible as initially hoped, it did consistently provide a reasonably high cumulative weighted probability result of its multicast cache as will be shown by the simulation results in chapter VI.

4.8 Probability Divided by Bandwidth Model

Recall the discussion earlier about how the Probability Only Model made no caching considerations regarding a channel's bandwidth usage other than whether it would fit into the remaining bandwidth available. In that model an 18 Mbps channel with a slightly higher probability was favored over a channel with much lower bandwidth usage but a lower probability. The Probability Divided by Bandwidth Model takes into account both the probability and bandwidth of each channel prior to cache selection.

Consider a channel with a weighted probability equal to 0.1 and a bandwidth of 18 Mbps. The Probability Divided by Bandwidth Model sorts the Channel List array based on each channel's probability (***X**ᵢ*) divided by its bandwidth (***R**ᵢ*) as defined in the sort field below:

$$\text{Sort Field} = \frac{\chi_i}{R_i}$$ **Equation 4-6**

The channel under consideration would have a value of (0.1 / 18) or 0.005556 whereas another channel with the same probability but using only 2 Mbps of bandwidth would have a value of (0.1 / 2) or 0.05. In effect, channels are penalized proportional to their bandwidth usage in this model prior to caching selection, and this concept of bandwidth penalization proved to be the key contribution introduced by this model. The mathematical basis for this model stems from prior research that utilizes analogous penalization techniques in the finance and rate distortion disciplines where entries are penalized by cost or bitrate respectively [104-105].

This model introduces a modest amount of additional complexity over the Probability Only Model, but in doing so it can make much more informed caching selections based on both the probability and bandwidth of each channel. The performance of this model during the simulation phase of this research showed that these more informed decisions resulted in a significantly higher probability of predicting the next channel selection.

<u>4.9 Sample Moment Model</u>

The Sample Moment Model uses additional mathematical parameters for channel sorting prior to the cache selections. Estimates of the first moment are calculated using traditional methods to yield the sample mean ($\bar{x}$) of the weighted probabilities ($\boldsymbol{X_i}$) for a channel population ($\boldsymbol{n}$) as follows [106]:

$$\text{Sample Mean} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} \chi_i$$ **Equation 4-7**

The Sample Moment Model works in the same manner as the Probability Divided by Bandwidth Model except that the sort field is no longer simply the probability divided by bandwidth term. Instead the sort field is calculated for each channel in the following manner:

$$\text{Sort Field} = \frac{\chi_i - \bar{x}}{R_i^{pf}}$$ **Equation 4-8**

where $\boldsymbol{X_i}$ is the weighted probability of channel $\boldsymbol{i}$, $\bar{x}$ is the sample mean of the weighted probabilities ($\boldsymbol{X_i}$) of all available channels ($\boldsymbol{n}$), $\boldsymbol{R_i}$ is the bandwidth or rate of channel $\boldsymbol{i}$, and $\boldsymbol{pf}$ is the penalty factor applied to the bandwidth. The following analysis steps through the development of this sort field and the individual contributions of each term.

Subtracting off the weighted probability sample mean ($\bar{x}$) places more emphasis on channels with higher probabilities. This is true since channels with weighted probabilities below the mean become negative and channels with probabilities slightly higher than the mean approach zero. This change directly affects the sorting order. Figure 4.2 shows a hypothetical example that focuses on only two of 11 channels available in this case for simplicity where channel #7 uses one half the bandwidth of channel #11, but #11 has a bit less than twice the probability of #7. Channel #7 appears first in the list when sorted by probability divided by bandwidth, but when these channels are sorted by the probability minus the mean term divided by the bandwidth then #11 rises above #7 in the sort order. Channel #11, being farther positive from the mean weighted probability than Channel #7 and hence, based on past history, a more popular selection, receives more emphasis and is penalized less for its higher bandwidth requirements than occurs in the Probability Divided by Bandwidth Model of Section 4.8.

From the Probability Divided by Bandwidth Model it was evident that dividing by the bandwidth effectively penalized channels operating at higher rates. The bandwidth penalty factor (***pf***) was introduced to explore the impact of this penalty with regard to the channels chosen for caching. Systematically varying the penalty factor in the range of 0.5

Example Sample Moment Model Channel Details:

Rank #	Xi
1	0.01
2	0.02
3	0.04
4	0.06
5	0.07
6	0.10
7	0.11
8	0.12
9	0.13
10	0.15
11	0.19
Total Sum =	1.00000

Sample Mean Probability of all (n=11) Channels = $\bar{x} \approx 0.09091$

#	X_i	R_i	X_i / R_i	$(X_i - \bar{x})$	$(X_i - \bar{x})$ / R_i
4	0.06	2	0.03	-0.0309	-0.0155
6	0.1	4	0.025	0.0091	0.0023

Sorted by X_i / R_i

#	Xi	R_i	Xi / R_i
4	0.06	2	0.03
6	0.1	4	0.025

Sorted by $(X_i - \bar{x})$ / R_i

#	Xi	R_i	$(X_i - \bar{x})$ / R_i
6	0.1	4	0.0023
4	0.06	2	-0.0155

Figure 4.2 Example Showing the Effects of the Mean Term

to 2.0 during initial testing indicated that improved results could indeed be obtained. The ***pf*** values that provided the best results for the simulations were calculated at 1.01 for 50 Mbps and 1.19 for 65 and 75 Mbps video bandwidths, but in field use this value might need to be periodically updated using the feedback loop processes as described in the next section. This model had the highest performance results of all models as discussed in Chapter VI.

4.10 Mixed Model Approach and Feedback Loops

The simulation results that are fully presented in Chapter VI show that specific caching models end up with higher cumulative weighted probability results on average. Not surprisingly there are instances where, for a specific input data set, models with lower results on the average outperform models that, on average, have better results. It is worth noting that the overall effectiveness of the system might be slightly increased with a mixed-model approach where multiple models are running simultaneously. The multicast cache would then be selected through the model with the highest cumulative cache probability.

The vast majority of calculations covered in this chapter for the various models are accomplished outright on the input data representing the channel change history buffer and other known quantities. Besides channel change probabilities, there are a few additional parameters that that could be periodically updated over time through the use of feedback mechanisms. The candidates for updates through feedback loops are the following:

- **Size of the Channel Change History Buffer (N)**
- **Constant Weighting Parameter (α)**
- **Bandwidth Penalty Factor (*pf*)**

The size (N) of the channel change history buffer is a tradeoff between having enough information for meaningful probability analysis against memory and processor resources to maintain the data. Consider a situation where the oldest entries on the history buffer had virtually no impact to the cumulative probability calculations per channel, then the history buffer might be reduced to save memory and processor resources with negligible resulting impact.

The constant weighting parameter (**α**) was shown to impact the effect of past history on channel caching results. This parameter works in conjunction with the size of the history buffer to represent viewer channel changing behavior. Whenever the weighting parameter is changed, then the history buffer length may be adjusted as well.

Likewise the bandwidth penalty factor (***pf***) from the Sample Moment Model would likely benefit from a feedback approach. In particular the ***pf*** would need to be optimized whenever a user adjusts the total bandwidth available to video services or when caching performance falls below expected limits. Short term changes in bandwidth may not necessarily call for rigid adjustment of the ***pf***, but any lengthy changes in available bandwidth should trigger an update to this parameter.

These feedback loops would be triggered by logical assessment of statistical fields and status flags. The exact nature of these fields will be subject to system hardware limitations as covered in Chapter VII, but the basic parameters illustrated with Figure 4.3 would likely satisfy the needs of the feedback system. The approach relies upon several User-Defined Parameters, Tracking Statistics, Metrics, and Flags to develop the logic required for the feedback loops. The three feedback-adjustable variables would have associated optimization routines that could be triggered by the presence of negative flag conditions. The Flag Conditions could be updated after each channel change based on logic associated with the defined parameters, statistics, and metrics.

The primary considerations about the size of the channel change history buffer are that it is large enough to effectively track the entire statistically significant channel changes and that it is not so large that it unnecessarily slows down the processing times required to determine the channel caches. The minimum statistical significance of the buffer can be easily determined by looking at the weight of the oldest entry in the table to see if it is above a minimum default value. The values of these weights are only changed when the constant weight parameter (**α**) is modified. The processing time to determine

User-Defined Parameters:

Video Bandwidth Allotted Mbps	65 Mbps
Cache % Deviation Trigger Limit	10%
Failed Cache Processing % Limit	1%
Recent Change Limit	100
pf Adjustment Limit Mbps	5 Mbps

Tracking Statistics:

Total # Caches Counter	Caching Success Counter	Caching Fail Counter	Caching Success %	Cache Avg. Cumm. Weighted Probability %	Failed Cache Processing %	Failures Inside Recent Change Limit
4731	3071	1660	64.9	68.3	0.08	1

Metrics and Flags:

Current Video Bandwidth Available Mbps	65.0
Channel Change History Buffer Size	2000
Current Constant Weight Parameter **α**	0.98
Current BW Penalty Factor ***pf***	1.19
Channel Change History Buffer Flag	Y
Constant Weight Parameter **α** Flag	Y
BW Penalty Factor ***pf*** Flag	Y

Figure 4.3 Example of Feedback Statistics Display

the multicast cache after each channel change should ideally always be less than the time before the next channel change. This will be further discussed in Chapter VII, but if the next channel change occurs before the previous cache processing is completed then the cache would not have time to be updated for that channel change. This situation may not always have a direct impact on the caching effectiveness since the caches are often not radically changed even after processing, but nevertheless the number of these events should be minimized. The “Failed Cache Processing %” field reflects the percentage of times that the next channel change is received before the caching process is complete. Therefore, the channel change history buffer size flag would be set to “N” triggering a size adjustment whenever the **α** parameter is changed or when the “Failed Cache Processing %” field exceeds the user-defined “Failed Cache Processing % Limit” value.

The constant weight parameter (**α**) would be subject to adjustment when the expected effectiveness of the system consistently exceeds actual system performance. As proposed this adjustment would be triggered whenever the difference from the expected “Cache Avg. Cumm. Weighted Probability %” to the actual “Caching Success %” is greater than the defined limit established by the “Cache % Deviation Trigger Limit” parameter. This condition indicates that the expected caching success rate is being overestimated by the caching model and the “Constant Weight Parameter **α** Flag” is set to N to initiate the **α** adjustment routine. The defined “Recent Change Limit” parameter and the “Failures Inside Recent Change Limit” counter would be used by the adjustment routine to analyze the frequency that newly popular channels go unnoticed by the model.

The bandwidth penalty factor (***pf***) value would be adjusted when using the Sample Moment Model and significant changes are made to the amount of video bandwidth that is available for use or when unexpectedly poor performance is detected. The "Current Video Bandwidth Available Mbps" field reflects the actual bandwidth available for use by the system over the period of time that the field covers. This period would need to be determined during the manufacturing process for each type of device, but using a one minute period to determine the average current bandwidth availability would ensure that the ***pf*** is not altered based on instantaneous bandwidth fluctuations while still keeping the system reasonably up to date. If the "Current Video Bandwidth Available Mbps" field drops below the defined "Video Bandwidth Allotted Mbps" parameter by more than the limit established in the defined "***pf*** Adjustment Limit Mbps" parameter, then the "BW Penalty Factor ***pf*** Flag" would be set to N to trigger the ***pf*** adjustment routine. The ***pf*** value would also be subjected to an adjustment routine for unexpectedly poor caching success rates if the **α** adjustment routine fails to achieve cache results within the trigger limits specified earlier.

Additional details about potential feedback routines are shown in Appendix D. These feedback routines could be designed to provide higher likelihoods of caching effectiveness over time, but as discussed earlier the system should not typically be expected to achieve 100% success rates. Each of these feedback routine proposals takes comparisons of varied adjustments to select the optimal parameter level available for the observed conditions. Additional considerations and practical implications of these feedback processes will be further discussed during the implementation considerations of Chapter VII. Potential improvements and areas of future research, such as analysis of the time in which channel changes occur in terms of time-of-day, day-of-week, and month-of-year or the geographical viewer location, are included in Chapter VII as well.

CHAPTER V

SAMPLE DATA SETS DEVELOPED FOR THE PROJECT

This chapter explains how the collections of sample data sets were developed for this project. Eighteen individual sample data sets reflecting a variety of situations were created for use by the models covered in the last chapter. Each model would then perform three different simulations on each data set with the bandwidth available for video service set to 50, 65, and 75 Mbps respectively. This led to 54 unique simulations performed by each of the six models yielding the results presented in Chapter VI.

5.1 Variables to be Managed

This proposed multicast caching system would require several variable factors to be set as either fixed or adjustable values over time through the feedback processes. The channel change history buffer would have to be created and maintained to track the exact order of recent channel changes along with the current status over some definitive period of time. A separate process would maintain information about the bandwidth required for each available channel. Another process would keep track of the currently available bandwidth that the system has through the appropriate network connections, and a related process would determine how much of this available bandwidth could be devoted to IPTV video usage based on user preferences or other specified criteria. The models would then use the channel change history buffer, the currently viewed channel, the required bandwidth for each channel, and the bandwidth presently available for IPTV video applications to determine which channels should be included in the current multicast cache.

The sample data sets for this project were designed to provide these characteristics that the models needed to make the caching selections. The sample data sets had to meet a wide variety of requirements to maintain validity throughout the simulation process. The following primary variables had to be considered and handled appropriately throughout the creation of the sample data sets:

- **Number of Channels Offered (n)**
- **Mix of SD and HD Channels**
- **Bandwidth Required Per Channel (R_i)**
- **Total Bandwidth Available for IPTV Video Applications**
- **Size of the Channel Change History Buffer (N)**
- **Favorite Channel Listings per User**
- **Constant Weighting Parameter Value (α)**

These main variables all had to be defined, tracked, and modified as needed to cover various potential conditions that the multicast caching system would likely encounter in real-world applications.

The number of channels offered by IPTV service providers varies significantly based on equipment capabilities, network limitations, broadcast licensing, geographic location, service pricing, user agreements, and other factors. Preliminary tests were performed with higher and lower channel availability numbers which clearly indicated that modeling results were inversely proportional to the number of channels offered. Lower numbers of channels allowed for higher success rates while higher numbers of channels decreased the success rates if all other factors were held constant. Given the overall number of variables that had to be accounted for and the knowledge that the number of channels offered gave direct insight into the proportional expectations of caching, the decision was made to hold the number of offered channels constant during this research. IPTV providers typically offer from100 to 200 channels [32][107-109], so the channel availability for this project was fixed midway between these levels at **n**=150 channels in line with previous research assumptions [109].

The mix of SD and HD channels offered also differs widely by provider and region [107]. To reflect these current and near-term variations, three bandwidth mixes were created with differing levels of SD and HD channels. This was accomplished first by assigning four popular bandwidth levels of 2, 4, 9, and 18 Mbps as the available bandwidth rates for various SD and HD channels. The three bandwidth mixes were then assigned different percentages to each of the four bandwidth rates as shown in Table 5.1.

Mix A reflects a current provider offering limited HD services, while Mix B indicates a provider offering a moderate level of HD services. Mix C would reflect a

BW Rate (in Mbps)	**Mix A**	**Mix B**	**Mix C**
2	50.0%	40.0%	35.0%
4	40.0%	30.0%	15.0%
9	7.5%	25.0%	40.0%
18	2.5%	5.0%	10.0%

Table 5.1 Channel Bandwidth Probability Used in Simulations

provider that has at least half of the channels offered at full HD quality. Clearly these three mixes do not cover the full range of all options possible by the industry, but they do provide significantly different views of SD and HD offerings to expose any serious modeling flaws with respect to this variable.

With the number of channels fixed at **n**=150 and three bandwidth mixes defined, the next step was to assign one of the four bandwidth rates to each of the 150 channels. This was accomplished by creating a series of pseudo-random numbers from 0 to 1 that were applied against a percentage weighting table reflecting the bandwidth rates for each channel and each mix.

The total bandwidth available for IPTV video applications was also modified through three different values. The basic MDU case with a total customer link bandwidth of 100 Mbps served as the baseline broadband rate, so the resulting bandwidth rates available for video were between 50 and 75 Mbps as defined much earlier in Table 1.6. Three video bandwidth options were defined with the values of 50, 65, and 75 Mbps to reflect differing customer expectations based on their background processes and applications. These options would allow users to reserve from 25 to 50 Mbps for their other applications associated with voice, data, overhead, and safety margins.

Another variable to be considered was the size of the channel change history buffer (**N**). Beginning with the most recent and then working backwards, the question was how many previous channel changes should be tracked in the history. Published studies indicate that viewers change channels an average of 25 times per day, although some users reach levels of a few hundred changes per day [27]. As noted in the previous chapter, the channel history is used to determine the relative probability that a channel may be selected next, so a reasonably high buffer was required to improve the confidence level of the calculated probabilities. If all of the variables to be considered were allowed to vary throughout the simulation process, then the complexity of the process would increase exponentially. The channel change history buffer was fixed to sequentially track the most recent 2000 channel changes. Additional details, such as the time-of-day and day-of-week, were not included in this channel change history buffer. An analysis about the inclusion of those and other inputs derived from the many potential fields of data that could be collected about each viewer will be discussed in Chapter VII.

Every viewer has an individual list of popular or favorite channels that varies by age, sex, time of day, day of week, season of year, and other factors. For example, the sports channels might expect to see an increase in game day viewership during the fall football season. Table 5.2 shows a portion of sample data reflecting both the rank and channel number for each entry. This list goes from a rank of 1 to 150 with a decreasing probability assigned to represent the most favorite down to the least favorite channel. The concept of randomly determining the channel ranking order will be covered in the next section, and the methods used for determining the actual probability numbers shown will also be a major topic of discussion later in this chapter.

For clarity the rank refers to the user's favorability of channels which is also referred to as the channel popularity by some researchers [108], and the channel number refers to the actual user number selections via remote or guide. The first entry on Table 5.2 has a rank of one indicating that channel number 79 is this user's favorite channel with the highest probability of selection. The P(x) column in Table 5.2 indicates the

Rank	Channel #	P(x) =	F(x) =
1	79	0.095163	0.095163
2	107	0.086107	0.181269
3	127	0.077913	0.259182
4	7	0.070498	0.329680
5	90	0.063789	0.393469
6	95	0.057719	0.451188
7	136	0.052226	0.503415
8	101	0.047256	0.550671
9	76	0.042759	0.593430
10	120	0.038690	0.632121
11	149	0.035008	0.667129
12	133	0.031677	0.698806
13	110	0.028662	0.727468
14	123	0.025935	0.753403
15	21	0.023467	0.776870
16	19	0.021234	0.798103
17	35	0.019213	0.817316
18	128	0.017385	0.834701
19	96	0.015730	0.850431
20	10	0.014233	0.864665
21	53	0.012879	0.877544
22	147	0.011653	0.889197
23	12	0.010544	0.899741
24	78	0.009541	0.909282
25	137	0.008633	0.917915
26	115	0.007811	0.925726
27	86	0.007068	0.932794
28	104	0.006395	0.939190
29	13	0.005787	0.944977
30	48	0.005236	0.950213
31	121	0.004738	0.954951
32	32	0.004287	0.959238
33	46	0.003879	0.963117
34	116	0.003510	0.966627
35	15	0.003176	0.969803
36	20	0.002874	0.972676
37	59	0.002600	0.975276
38	118	0.002353	0.977629
39	80	0.002129	0.979758
40	102	0.001926	0.981684
41	11	0.001743	0.983427
42	92	0.001577	0.985004
43	143	0.001427	0.986431
44	31	0.001291	0.987723
45	100	0.001168	0.988891
46	63	0.001057	0.989948
47	44	0.000957	0.990905
48	61	0.000866	0.991770
49	47	0.000783	0.992553
50	71	0.000709	0.993262
51	140	0.000641	0.993903
52	139	0.000580	0.994483
53	69	0.000525	0.995008
54	38	0.000475	0.995483
55	37	0.000430	0.995913
56	142	0.000389	0.996302
57	126	0.000352	0.996654
58	55	0.000318	0.996972
59	91	0.000288	0.997261
60	23	0.000261	0.997521
61	146	0.000236	0.997757
62	132	0.000213	0.997971
63	114	0.000193	0.998164
64	36	0.000175	0.998338
65	14	0.000158	0.998497
66	4	0.000143	0.998640
67	52	0.000129	0.998769
68	111	0.000117	0.998886
69	148	0.000106	0.998992
70	24	0.000096	0.999088
71	122	0.000087	0.999175
72	33	0.000079	0.999253
73	74	0.000071	0.999324
74	108	0.000064	0.999389
75	144	0.000058	0.999447
76	109	0.000053	0.999500
77	106	0.000048	0.999547
78	29	0.000043	0.999590
79	25	0.000039	0.999629
80	117	0.000035	0.999665
81	9	0.000032	0.999696
82	5	0.000029	0.999725
83	16	0.000026	0.999751
84	93	0.000024	0.999775
85	119	0.000021	0.999797
86	94	0.000019	0.999816
87	98	0.000018	0.999833
88	138	0.000016	0.999849
89	70	0.000014	0.999864
90	64	0.000013	0.999877
91	28	0.000012	0.999888
92	66	0.000011	0.999899
93	113	0.000010	0.999909
94	82	0.000009	0.999917
95	56	0.000008	0.999925
96	87	0.000007	0.999932
97	22	0.000006	0.999939
98	129	0.000006	0.999945
99	60	0.000005	0.999950
100	65	0.000005	0.999955
101	99	0.000004	0.999959
102	45	0.000004	0.999963
103	40	0.000004	0.999966
104	30	0.000003	0.999970
105	58	0.000003	0.999972
106	17	0.000003	0.999975
107	57	0.000002	0.999977
108	73	0.000002	0.999980
109	51	0.000002	0.999982
110	68	0.000002	0.999983
111	1	0.000002	0.999985
112	75	0.000001	0.999986
113	150	0.000001	0.999988
114	112	0.000001	0.999989
115	72	0.000001	0.999990
116	85	0.000001	0.999991
117	54	0.000001	0.999992
118	26	0.000001	0.999992
119	77	0.000001	0.999993
120	130	0.000001	0.999994
121	131	0.000001	0.999994
122	84	0.000001	0.999995
123	39	0.000000	0.999995
124	83	0.000000	0.999996
125	2	0.000000	0.999996
126	41	0.000000	0.999997
127	62	0.000000	0.999997
128	49	0.000000	0.999997
129	81	0.000000	0.999998
130	6	0.000000	0.999998
131	67	0.000000	0.999998
132	103	0.000000	0.999998
133	3	0.000000	0.999998
134	34	0.000000	0.999998
135	88	0.000000	0.999999
136	43	0.000000	0.999999
137	145	0.000000	0.999999
138	141	0.000000	0.999999
139	134	0.000000	0.999999
140	27	0.000000	0.999999
141	42	0.000000	0.999999
142	8	0.000000	0.999999
143	89	0.000000	0.999999
144	124	0.000000	0.999999
145	105	0.000000	0.999999
146	135	0.000000	1.000000
147	50	0.000000	1.000000
148	97	0.000000	1.000000
149	125	0.000000	1.000000
150	18	0.000000	1.000000

Table 5.2 Sample Favorite Channel Listing

probability assigned to each channel entry. The F(x) column provides the cumulative distribution of probabilities for each entry down the list in the table.

Each viewer exhibits individual favorite channel listing behavior with respect to not only the order of favorite channels but also the shape or profile of how many channels a viewer regularly watches. A viewer that "surfs" through a large portion of the 150 channels offered would have a much lower cumulative probability through the highly ranked channels than a viewer that only watches a small selection of channels. This characteristic defines the shape distribution of how the favorite channels are watched with respect to different viewers. While the same general type of the distribution may be applicable across a wide range of viewers, the specific shape details about that distribution will be different across the population. These topics will be discussed in more detail in upcoming sections, but the notable implication is that the distribution needed to be varied at some point during the creation of the sample data sets to reflect these differences in viewer's channel changing behavior.

The channel change history will be used to determine how often channels have been selected over time to develop probabilities about future selection. With the use of a relatively large history buffer (**N**=2000), the notion that the most recent events reflect the current probabilities better than older events led to research into the time-weighting process presented in the last chapter. With the adoption of the time-weighting scheme it became necessary for the constant weighting parameter (**α**) to become another variable to be considered. During this research the **α** parameter was analyzed and assigned the fixed value of 0.98 as noted earlier, but in actual practice the feedback mechanisms previously discussed in the last chapter would adjust the weights to maintain the system at optimal levels over time.

The variables discussed in this section formed the basic criteria used to create the sample data sets and to perform the model simulations. Some of the variables were fixed to constants while others were varied over a range of values. Each variable had to be considered individually and with respect to the other variables for validity of the assumptions used during the creation of the sample data sets.

5.2 Notes on Independence of Samples

One key area of concern for this research is whether user channel changes are or can be assumed to be i.i.d. from change to change. The major components of user behavior with respect to channel changing are the duration or dwell time of watching particular channels and the frequency with which individual channels are selected. This research is not particularly concerned with the duration of watching particular channels except for how it leads to determining the expected time between user selections thus impacting the overall sizing of the channel change history buffer covered in Chapter IV as well as the processing implications to be discussed in Chapter VII. This project is primarily focused on understanding the popularity or frequency with which individual channels are selected.

Regarding the popularity of channels, it is intuitively clear that the actual channel changing process is not independent as readily evidenced by the fact that many users loyally watch specific programs. In fact a wide variety of dependencies contribute to individual user IPTV behavior including age, sex, ethnicity, locale, income, and education. Most of the previous research into channel pre-fetching did not address the i.i.d. issue directly or even mention that level of detail about their simulations [21][29-30][34]. Therefore the key question of concern here is whether the assumption of i.i.d. channel selections as applied here is valid.

In [26] and [77] the i.i.d. assumption was directly documented and used regarding simulated channel changes being independent of one another. In order for each channel change to be an independent event, the probability of the change cannot be affected in any way by previous changes. In real-world applications it is obvious that this is not actually the case. Consider the user that is swapping back and forth between two channels for news, business, weather, sports, or other programming. The probability of the next channel change would then not be an independent event since the swapping pattern defines the next channel with a much higher probability over all others. Similarly, the user that increments or decrements channels up or down to get to the intended destination does not behave in an i.i.d. fashion. However, researchers are still left with the fact that viewer behavior is unique and there are no known references to actual user behavior in the open literature. Actual channel change behavior information is available from companies such as Nielsen, but not at a reasonably affordable fee. Hence the choice has been made here to proceed in a manner common with previous studies, and assume that aspects of channel changes are statistically independent. It is important to note that this assumption has been previously made in peer-reviewed documentation of this application even though this alone does not guarantee its validity.

If deployed by an IPTV provider who purchases channel change information from a company such as Nielsen, future research could be conducted into using state processes based on Bayesian, Markovian, or Brownian models to reflect some of these channel change dependencies [79][110]. These advanced techniques could take into account current or recent states that a system exhibits for a more informed decision.

The actual implementation plan for multicast caching calls for the historical tracking and analysis of each downstream IPTV port identified for usage within the system. Non-independent events would still be reflected in the historical record associated with each user. Consider a two STB household with one TV in the parents' bedroom and one in the children's bedroom. If the children are frequently watching kids' programming in their room, then those channels would be recorded much more heavily in the history associated with that port than on the parents' port. In another example, consider a small apartment complex with only one Hispanic resident frequently watching Spanish language channels. Clearly those channels would not show up much in the other residents' history, but they would accurately be reflected in the history of the Hispanic resident. This individualized approach would capture the history of certain non-independencies in real-world channel changes under the implementation plan proposed. The caching selection models would still correctly cache the next dependently selected channel for all cases in which that channel otherwise met the weighted probability criteria used for caching.

Some of the potential concerns associated with relying on the i.i.d. assumption can be easily mitigated. For actual implementation one key proposal was mentioned previously about the additive properties of multiple methods where this caching system could be employed in conjunction with other existing techniques that have been previously developed by other researchers. For example, other researchers have already proposed systems to pre-select the last channel viewed or to identify if a user is incrementing or decrementing channels and then cache channels accordingly [25-26][38][73]. Additional logic processes could be created to properly integrate these alternative methods effectively, but the approach of using multiple methods would help identify and manage several of the common non-independent situations that frequently arise.

5.3 Exponential Distributions

If an IPTV service provider was offering 150 channels to its customers, then the uniform distribution would accurately describe a viewer's channel change behavior only if the user selected channels in a purely random fashion or continuously changed through all 150 channels systematically. In reality users have certain favorite channels that they tend to watch much more frequently than others. Some users may only watch a very small number of channels while others have a longer list of favorites. Previous researchers have made use of different distributions to model viewer behavior with varying degrees of success. Attempts to model user behavior typically aim at providing baseline or average channel changing representations, and it is widely recognized that some users will be outliers that fall outside of the norm. The following distributions have been used recently to model various aspects of user channel changing behavior:

- **Uniform [5][27][57]**
- **Exponential [26][108-109]**
- **Zipf [5][28][57][108-109]**
- **Poisson [5][26][28]**
- **Gamma [26]**
- **Weibull [108]**

Most of these previous researchers did not elaborate on why or how each PDF was selected and used, but in [108] a comprehensive discussion was made into identification of the most valid methods of modeling user channel changing behavior. The goal of that research was to come up with a series of models that suitably reflected actual IPTV usage across a variety of metrics. This team composed mostly of AT&T researchers compiled trace data from over two million anonymous STBs across four time zones over a seven day period along with associated network device logs, programming information, and other pertinent details. Specifically with regard to modeling the popularity of channel selections, their research found that the exponential and Zipf distributions were the top two PDF performers. They stated that while the Zipf distribution worked well for the top few highly popular channels, the exponential distribution achieved a better fit for the larger body of available channels. For these reasons the exponential distribution was selected to provide the basis for user channel popularity representations during this project in line with this most comprehensive analysis on the topic.

The PDF for an exponential random variable with parameter $\lambda > 0$ is defined as follows [91][110]:

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Equation 5-1

The Cumulative Distribution Function (CDF) of this exponential random variable is defined as follows:

$$F(a) = \int_0^a \lambda e^{-\lambda x} dx = 1 - e^{-\lambda a}, \qquad a \geq 0$$

Equation 5-2

In this study, this exponential distribution is used for the purpose of creating valid sample data sets to be used as simulation inputs, but in actual field use the models will

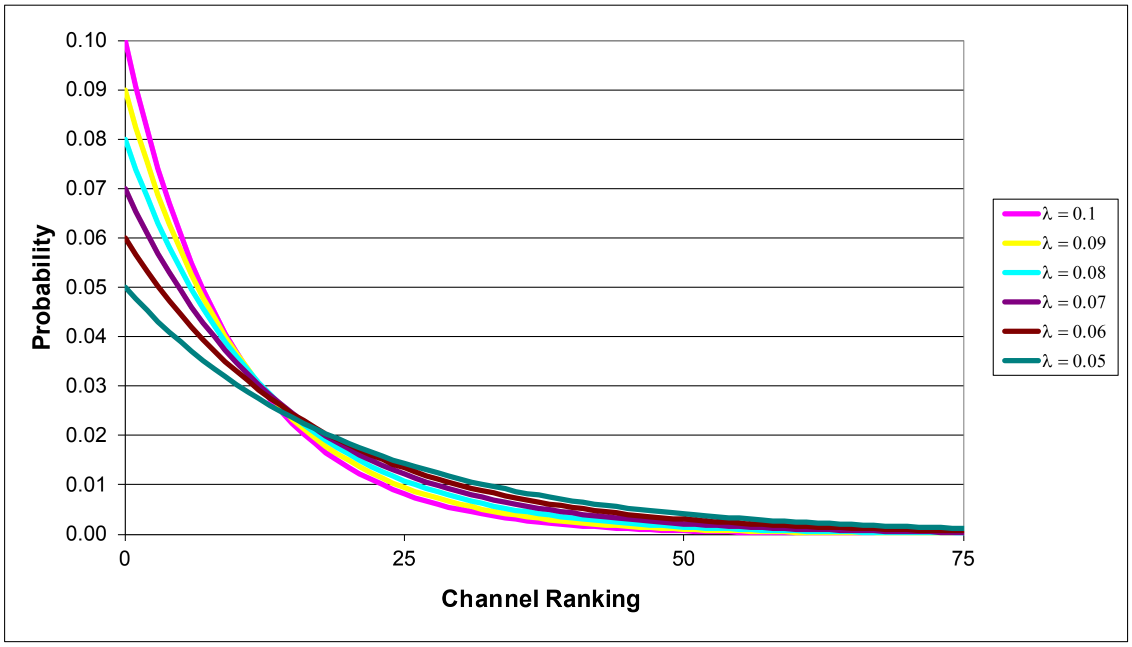


Figure 5.1 Sample PDFs for the Exponential Distribution

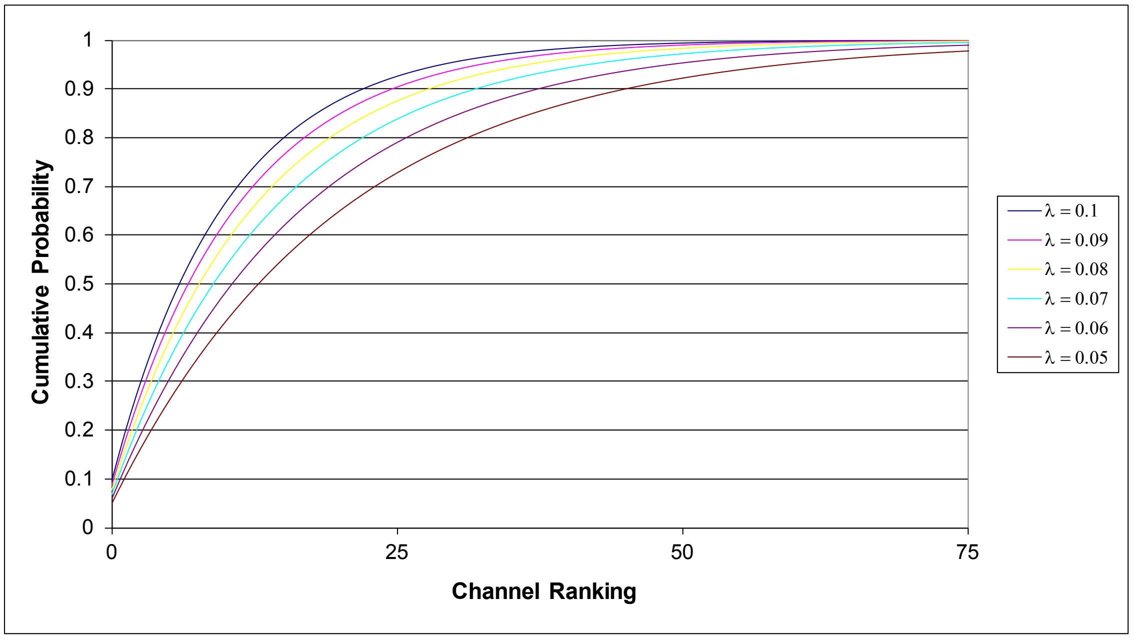


Figure 5.2 Sample CDFs for the Exponential Distribution

only be concerned with the historical data collected on a per port basis. The resulting sample data sets derived from these exponential PDFs represent an acceptable fit for channel popularity for simulation, but the actual system implementation does not have a direct requirement for individual users to follow the exponential distribution.

Basic user behavior about the number of favorite channels that are typically watched over some historical period of time was modeled by varying the λ parameter of the exponential distribution. Figure 5.1 shows the exponential for a variety of λ parameter values with the x-axis indicating the channel ranking and the probability of each channel is indicated by the y-axis. Note that the favorite channels are ranked first in all cases and naturally have the highest probability of being selected. From the PDFs shown on Figure 5.1, it is evident that the higher λ parameter values have a steeper initial slope corresponding to a user that typically watches only a few channels on a regular basis. The lower λ parameter values would be associated with viewers that watch or "surf" many different channels over the historical period.

Figure 5.2 provides a view of the sample exponential CDFs for these same λ values to offer another representation of these concepts. Consider the 0.1 λ distribution at the approximate point where it crosses the 0.90 or 90% mark and note that this point corresponds to just below the cumulative probabilities of the first 25 ranked channels. However, it takes the cumulative probabilities of almost the first 50 ranked channels for the 0.05 λ distribution to reach the 90% mark. These λ values were the specific limits used during the creation of the sample data set for this project, and they exhibit CDFs similar to those presented by the AT&T research team [108]. Lower λ values were discarded based on documented user behavior indicating that the vast majority of viewers find less than one third of the total channels offered to be their favorites [27][108-109]. Higher λ values were similarly discarded as the documentation indicates most users surf through more than just a few channels with regularity. By generating sample channel changes based on the various defined λ parameter settings of the exponential distribution, each sample data set was created to reflect some of these differences seen in viewer channel popularity within the primary range of expectations.

5.4 Creation of the Sample Data Sets

Each sample data set was created through a series of Visual Basic macro routines run on a Microsoft Excel spreadsheet. These routines performed the following functions:

- **Defined specific parameter settings for the next data set**
- **Randomly determined the video channel line-up order**
- **Determined the bandwidths of each video channel for all three mixes**
- **Simulate channel changes based on the parameter settings**

Following the completion of these routines the spreadsheet shows the ranking of video channels, bandwidths assigned to each channel for all three mixes, current channel selected, and a history of the simulated channel changes. These provide the key elements to test the different models' performance and resulting channel cache. The detailed spreadsheet format and the actual macro routines for these processes are found in Appendix E, but the basic steps associated with each function are explained below.

Each sample data set was created to reflect the specific parameters that were assigned to match the conditions desired for that unique trial. The three specific parameters defined at this point were the exponential distribution parameter (**λ**), the number of channel changes simulated, and the constant weighting parameter (**α**). The **λ** parameter was varied in the range of 0.05 to 0.1 to correspond with expected viewer behavior about the channel popularity of different users as described earlier. The number of generated channel changes was set to **N**=2000 throughout the sample data set creation for a fixed size of channel change history buffer. The constant weighting parameter was also fixed at **α**=0.98 during the time-weighting process as discussed in Chapter IV.

The data set spreadsheet contained a column listing the numbers from 1 to 150 to indicate the channel ranking. Another unique number from 1 to 150, representing an actual video channel, is then randomly assigned to each channel ranking. The bandwidth for each video channel was also randomly assigned based on the proportional rates defined for each of the three bandwidth mixes. The mix routine was set up to run three iterations per channel to provide the output for each bandwidth mix. Table 5.3 shows an example portion of the sample data creation spreadsheet to illustrate the process.

Column A	**Column C**	**Column M**
1	79	2
2	107	2
3	127	4
4	7	4
5	90	2
6	95	4
7	136	2
8	101	9
9	76	4
10	120	2
Channel Rank	**Video Channel #**	**BW Mix A**

Table 5.3 Example Bandwidth Rate Assignment Data

The Channel Rank list from 1 to 10 in Column A is a subset of the data that would normally range up to 150. Column C shows the randomly assigned Video Channel numbers. In Table 5.3, the bandwidths shown in Column M were randomly assigned using the probabilities associated with Mix A of Table 5.1. A similar procedure was used to generate channel preference tables for the other two bandwidth mixes.

An exponentially distributed CDF, defined as F($\boldsymbol{i}$), was then generated for all of the channels using Equation 5-3 below. In this equation the variable $\boldsymbol{i}$ represents the channel rank from 1 to 150 and λ is the defined exponential parameter.

$$\mathbf{F}(\boldsymbol{i}) = \mathbf{1 - e}^{-\lambda \boldsymbol{i}}$$

Equation 5-3

Table 5.4 shows the CDF values in column B calculated for λ = 0.1. Note that only the first and last five ranked channels are shown on this table with the remaining 140 channels omitted for clarity.

Column A	Column B	Column C
1	0.095162582	79
2	0.181269247	107
3	0.259181779	127
4	0.329679954	7
5	0.39346934	90
↓	↓	↓
146	0.999999544	135
147	0.999999587	50
148	0.999999626	97
149	0.999999662	125
150	0.999999694	18
Channel Rank	**Cummulative Distribution F(*i*)**	**Video Channel #**

Table 5.4 Example Exponential Channel Data

A separate macro routine was used to generate the series of channel changes for each sample data set. In this routine, 2000 pseudo-random numbers from zero to one are sequentially generated and individually compared with the cumulative distribution values in column B to determine the channel selections. Whatever range each pseudo-random number fell between dictated that specific channel change. Following the completion of this channel change macro, the associated sample data set spreadsheet had a change queue showing the sequence of these 2000 channel changes listed by video channel number. The currently viewed channel is identified at the top of the queue. This queue adds to the previous data collected showing the parameter settings, the video channel ranking order, and the channel bandwidths for each mix. The sample data set is complete at this point with the channel change history buffer filled, but additional processes were undertaken to perform the time-weighting functions and to simplify the format of the sample data for convenient use by the various models.

5.5 Time-Weighting of the Sample Data Sets

The rationale for time-weighting the channel change history was presented in Chapter IV along with the mathematical process and an analysis of the trade-offs involved in determining the constant weighting parameter ($\boldsymbol{\alpha}$).

Using the fixed constant weighting parameter of $\boldsymbol{\alpha} = 0.98$, a new column was created on the sample data set spreadsheet populated with 2000 progressively decreasing values of the weight from Equation 4-2, $\mathbf{W}_j = \boldsymbol{\alpha}^j$. It was then necessary to apply the weights to the channel change history buffer to calculate new time-weighted probabilities. The standard probability for a particular channel selection would typically

Channel Selected	Selection # (*j*)	Weight Equation	Weight
42	324	$W_j = (0.98)^{324}$	0.0014363111251
42	486	$W_j = (0.98)^{486}$	0.0000544343181
42	565	$W_j = (0.98)^{565}$	0.0000110339946
42	655	$W_j = (0.98)^{655}$	0.0000017909340
42	763	$W_j = (0.98)^{763}$	0.0000002020672
42	847	$W_j = (0.98)^{847}$	0.0000000370242
42	849	$W_j = (0.98)^{849}$	0.0000000355581
42	1104	$W_j = (0.98)^{1104}$	0.0000000002059
42	1150	$W_j = (0.98)^{1150}$	0.0000000000813
42	1471	$W_j = (0.98)^{1471}$	0.0000000000001
		Subtotal =	**0.0015038453085**

Table 5.5 Example of the Time-Weighting Process

be calculated by the summation of all of the times the channel was selected in the history divided by the total number of entries or 2000 in this case. To calculate the time-weighted probability it is necessary to sum up all of the specific weight values associated with the relative position of each selection of a channel. This summation of the specific weights of a channel is then divided by the total summation of all 2000 weights. Table 5.5 illustrates a portion of this process for determining the weighted probability of example channel number 42 from a sample data run. Note that Table 5.5 indicates that channel 42 was selected a total of 10 times during the trial, and its last selection was on the 324th most recent channel change in this example. Using the standard methods for determining the probability of this channel number 42 example results in the following calculation:

$$P_{\text{Standard}}(x = 42) = \frac{(\text{Times \#42 Selected})}{(\text{Total \# of Selections})} = \frac{(10)}{(2000)} = 0.005$$

Equation 5-4

However, calculating the time-weighted probability of channel 42 requires the summation of weights associated with its selections and the total sum of weights. Note that the total summation of all 2000 weights associated with the constant weighting parameter value of **α** = 0.98 is equal to a value of 49. Using the sub-total of weights for channel 42 from Table 5.5 and the total sum of all weights equal to 49, the weighted probability for example channel 42 is calculated as follows:

$$P_{\text{Weighted}}(x = 42) = \frac{(\text{Subtotal of all Weights for \#42 Selections})}{\sum_{j=1}^{N} \alpha^{j}}$$

Equation 5-5

$$= \frac{(0.0015038)}{(49)} = 0.0000307$$

The significant difference between the standard probability and weighted probability is primarily attributed to the fact that channel 42 was not included in any of the most recent 323 channel changes. This same time-weighting process was performed on all 150 channels for each sample data set.

The remaining sample data set tasks involved the necessary formatting and storing of the sample data for convenient use by the various caching model programs that were written and compiled in Visual C++. The resulting Minimized Sample Data Format is shown in Appendix E.

5.6 Summary

The overall number of sample data sets created was 18 in total. This allowed for six different values of the exponential **λ** parameter from 0.05 to 0.1 with each of these different values in turn producing three sample data sets corresponding to the different bandwidth mixes A, B, and C. The modeling programs then performed three simulations on each of the 18 data sets corresponding to tests using 50, 65, and 75 Mbps of available video bandwidth levels. This produced a total of 54 individual simulation runs for each channel caching model as presented in the next chapter.

CHAPTER VI

SIMULATIONS

This chapter provides a comprehensive view of the simulations used to test each of the multicast caching models. The rationale, methodology, and results of the Functionality Testing are presented in the first section. This is followed by a discussion explaining the Performance Simulations. Simulation predictions will then be presented followed by the results of the Performance Simulations with an analysis of the outcomes.

6.1 Functionality Testing

The goal of the Functionality Testing is to verify that each model is operating correctly, while the Performance Simulation to be discussed later on will provide empirical data to analyze the prediction capabilities of each model. The Functionality Testing involves running a multicast caching model through a series of simulated channel changes. This basic test verifies the proper operation of a model by checking that the following functions are correctly performed and updated as the test progresses:

- The various input parameters, such as the constant weighting parameter and the total bandwidth, are properly utilized.
- The current channel number and its bandwidth are correctly identified.

- The output channel cache is correctly calculated for the available bandwidth, weighted probabilities, and model algorithm.
- The output channel cache is properly modified by each successive channel change selected.
- The net cached weighted probabilities are correctly calculated.
- The system correctly tracks the successfulness of the cache prediction outcomes throughout the simulation.

This testing involves preparing an initial test data set with the channel change history buffer filled up with 2000 events based on the specified variable settings. A sequence of four channel changes is then generated with additional data sets created after each change. These new data sets are then run into each model in succession with the caching results recorded for analysis. The particular variable settings used for the Functionality Testing are shown on Figure 6.1 below.

The process of creating the initial test data set begins using these variable settings to result in the channel ranking and bandwidth values for the 150 channels used during

Exponential Lambda (λ) =	0.1
Constant Weight Parameter (**α**) =	0.98
Channel Change History Buffer Size (**N**) =	2000
Number of Channels Offered =	150
BW Mix Option =	C
BW Penalty Factor (***pf***) =	1.19
Total BW Available for Video (in Mbps) =	65

Figure 6.1 Variable Settings for Functionality Testing

this testing. The channel change history buffer is then populated by generating **N**=2000 channel changes based on one of the lambda value settings (**λ** = 0.10). The time-weighting function was then completed using the constant weight parameter value (**α** = 0.98), and the test data set is then represented through an initial minimized data format. The remaining test data sets are created in the same manner by generating single channel

Rank	Channel #	Bandwidth C
1	91	2
2	38	18
3	33	9
4	5	4
5	18	2
6	37	2
7	133	2
8	47	2
9	106	9
10	67	18
11	94	9
12	51	2
13	19	2
14	139	2
15	39	9
16	69	9
17	137	2
18	52	9
19	131	2
20	89	9
21	130	2
22	49	4
23	123	4
24	101	9
25	45	2
26	43	9
27	58	2
28	125	18
29	71	2
30	10	18
31	60	9
32	146	9
33	112	9
34	59	18
35	35	4
36	93	18
37	86	2
38	80	4
39	20	4
40	11	9
41	74	4
42	28	2
43	29	9
44	73	2
45	128	9
46	99	9
47	143	18
48	96	2
49	129	2
50	82	9
51	62	2
52	27	9
53	136	2
54	13	2
55	26	9
56	110	9
57	65	2
58	145	2
59	107	2
60	102	9
61	126	9
62	14	18
63	61	4
64	150	2
65	64	9
66	78	9
67	87	2
68	21	9
69	15	9
70	100	2
71	31	9
72	25	9
73	79	2
74	56	2
75	1	2
76	117	9
77	127	9
78	144	2
79	6	4
80	148	4
81	8	2
82	111	9
83	24	4
84	42	2
85	57	4
86	140	9
87	132	2
88	55	2
89	90	4
90	135	2
91	2	2
92	115	2
93	41	2
94	54	2
95	105	9
96	32	2
97	116	9
98	70	9
99	3	9
100	36	9
101	53	4
102	66	9
103	12	9
104	97	2
105	72	2
106	23	2
107	113	2
108	122	9
109	103	18
110	48	2
111	138	2
112	147	2
113	75	4
114	108	9
115	109	9
116	120	2
117	44	2
118	9	9
119	84	2
120	121	18
121	83	2
122	88	9
123	30	9
124	46	2
125	85	2
126	95	2
127	16	9
128	4	9
129	92	9
130	17	9
131	119	18
132	81	2
133	68	4
134	98	9
135	34	9
136	142	9
137	104	2
138	149	2
139	40	2
140	22	2
141	63	9
142	118	4
143	134	2
144	114	9
145	7	4
146	77	2
147	50	2
148	124	2
149	141	9
150	76	9

Figure 6.2 Example of Initial Channel Ranking and Bandwidth List

changes and recollecting the resulting information. Ultimately five test data sets are represented by five minimized data formatted files, and these files are then input sequentially into each caching model. An example of the channel ranking and bandwidth listing was shown on Figure 6.2 and an example of the initial minimized data format file after the time-weighting process is shown in a partial listing on Figure 6.3.

Channel #	Time-Weighted Probability	BW C
37	0.09172	2
33	0.07423	9
106	0.07129	9
67	0.06989	18
47	0.06501	2
5	0.06221	4
38	0.05967	18
91	0.05448	2
133	0.05162	2
69	0.03806	9
39	0.03728	9
18	0.03539	2
43	0.03178	9
89	0.03135	9
51	0.02514	2
139	0.02352	2
94	0.02267	9
19	0.02025	2
52	0.01953	9
101	0.01624	9
131	0.01541	2
129	0.01507	2
137	0.01331	2
11	0.01207	9
60	0.01139	9
125	0.01076	18
130	0.00488	2
71	0.00463	2
45	0.00374	2
49	0.00171	4
58	0.00139	2
146	0.00102	9
126	0.00079	9
10	0.00076	18
59	0.00073	18

Figure 6.3 Partial List of Initial Entries in Minimized Data Format

The sequence of the initial 2000 channel changes followed by the four additional single selections used in this testing is partially shown through the latest entries in the channel change history buffer on Figure 6.4. Each model is run sequentially through these five test inputs to generate five corresponding output caches for evaluation.

Buffer Bins	Channels Queued
Test 4	67
Test 3	93
Test 2	45
Test 1	37
Initial 1	39
Initial 2	89
Initial 3	52
Initial 4	5
Initial 5	37
Initial 6	69
Initial 7	47
Initial 8	33
Initial 9	106
Initial 10	67
Initial 11	133
Initial 12	37
Initial 13	91
Initial 14	106
Initial 15	129
Initial 16	47
Initial 17	101
Initial 18	67
Initial 19	5
Initial 20	47
Initial 21	131
Initial 22	33
Initial 23	37
Initial 24	33
Initial 25	51
Initial 26	11
Initial 27	133
Initial 28	91
Initial 29	60
Initial 30	38

Figure 6.4 Partial List of Entries in the Channel Change History Buffer

Test Status	Initial 2000		1st Test		2nd Test		3rd Test		4th Test	
Current Channel Number	39		37		45		93		67	
Current Channel BW	9		2		2		18		18	
	Cache	BW	Cache	BW	Cache	BW	Cache	BW	Cache	BW
Cached Channels & Associated BW	37	2	47	2	37	2	37	2	37	2
	47	2	91	2	47	2	47	2	47	2
	91	2	133	2	91	2	91	2	91	2
	133	2	18	2	133	2	133	2	133	2
	18	2	5	4	18	2	18	2	18	2
	5	4	51	2	5	4	5	4	5	4
	51	2	139	2	51	2	51	2	51	2
	139	2	19	2	139	2	45	2	45	2
	19	2	33	9	19	2	139	2	139	2
	33	9	106	9	33	9	19	2	19	2
	106	9	131	2	106	9	33	9	33	9
	131	2	129	2	131	2	106	9	106	9
	129	2	137	2	129	2	131	2	131	2
	137	2	69	9	137	2	129	2	129	2
	69	9	39	9	69	9	137	2	137	2
	45	2	45	2	39	9				
Cache Weighted Probability	0.686		0.667		0.703		0.637		0.669	
BW of Cache	55		62		62		46		46	
BW Remaining	1		1		1		1		1	
Cache Actions and Channel Movements	Cache set after initial 2000 channel changes		Removed #37 Added #39		Removed #45 Added #37		Removed #69 Removed #39 Added #45		No Change to Cache	
Successful Prediction of Previous Cache	N/A		Yes		Yes		No		No	

Figure 6.5 Functionality Testing Summary Results for the Sample Moment Model

Figure 6.5 shows the Functionality Testing Summary Results for the Sample Moment Model. A careful analysis of these results shows that the Sample Moment Model did meet all of the testing criteria identified earlier. In particular, it is evident that the current channel is properly updated and the caches do change as expected. This summary also provides another avenue to understand the mechanics of the multicast caching system. For example, when the current channel uses a higher bandwidth, as in the 3rd and 4th tests, then the total cached bandwidth exhibits a corresponding reduction.

All of the models did pass the Functionality Testing showing that the basic operation of the models proceeded correctly. This was not a particularly surprising result since when one of the models was properly debugged and fully operational, then the other models followed suit due to the heavy use of the same Shared Routines. The major changes between the models were with the implementation of the specific prediction algorithms. The Performance Simulation was next designed to evaluate the expected successfulness of these unique prediction algorithms.

6.2 Performance Simulation Methodology

The Performance Simulations involve running each of the multicast caching models on the full complement of sample data sets at specified available bandwidth rates. As seen in Chapter V, the sample data sets consist of 18 files that represent 6 different exponential λ values at three different bandwidth mixes. The minimized data format of each sample data set was created as described in Chapter V with each one detailing the specific channel rankings, bandwidths, time-weighted probabilities, and currently viewed channel. Each model performed three simulations on each of the 18 sample data sets using the rates of 50, 65, and 75 Mbps for the total available bandwidth for video services. This led to a total of 54 individual simulations per model.

Each simulation run amounted to one of the 18 minimized data format files being input to populate the historical buffers and related variables allowing the selected model to calculate the Confirmed Channel arrays and output the cached channel selections. This same process was repeated for each of the 54 simulations on all six models for a total of 324 individual runs. Each run represented a snap-shot in time of various conditions in which the model had to make decisions about which channels to pre-select. In real-world implementation this would take place on all IPTV device ports after each channel selection yielding a continuously updated multicast cache for each port.

A sample of the text format of the model output file is shown in Figure 6.6. As seen in this figure, the output file indicates the model used, the input data set file name, the specified total bandwidth available for video services, the channels in the Confirmed Channel arrays that make up the multicast cache, information on the currently viewed channel, and other attributes about the cache. It is clear that the example in Figure 6.6 is from a Sample Moment Model simulation with the first sample data set chosen as the input file (i1.txt) with 50 Mbps of total bandwidth available for video services. In this case, the 17 channels that were included in the multicast cache are listed by channel number, time-weighted probability, and channel bandwidth. A check is made to see that the 50 Mbps of total bandwidth is fully utilized by the 48 Mbps in the cache plus the 2 Mbps used by the currently viewed channel number 55 resulting in 0 Mbps of remaining bandwidth. The net expected weighted probability is calculated as shown in Equation 6-1 for the cached channels.

$$\text{Cache Net Weighted Probability} = \frac{\sum(\text{Cached Probabilities})}{(1 - \text{Current Channel Probability})}$$

$$= \frac{(0.6203)}{(1-0.0271)} = 0.6376$$

Equation 6-1

```
SAMPLE MOMENT MODEL OUTPUT


Input Filename =          i1.txt

Total Bandwidth =         50


Confirmed Channels List for Multicast Cache:


Chan    Prob    BW

10      0.0461  2

50      0.0438  2

88      0.0386  2

137     0.062   4

72      0.0326  2

60      0.0311  2

143     0.0556  4

73      0.0533  4

3       0.049   4

29      0.025   2
```

Figure 6.6 Example Format of a Model Output File

Equation 6-1 reflects the observation that the possibilities for the next channel selection should not include the current channel. The cache net expected weighted probability is therefore dependent on the other 149 channels that make up the full set of potential candidates for the next selection.

All of the 54 simulation output files for each model were saved into the results folder for that model. The completed data was then input into a separate Microsoft Excel file for a Cumulative Summary for each model. These summaries provide a quantitative means of analyzing each model based on all outcomes, performance by bandwidth rate, and performance by λ value. The key findings on the Cumulative Summaries were then input into a master Performance Simulation Results Summary for comparative analysis on the overall performance of the various models. Predicted results of the simulations followed by summary results and an analysis of the key findings are presented in the next sections. The complete detailed Performance Simulation can be found in Appendix F.

<u>6.3 Predicted Performance Simulation Results</u>

For caching predictions it is first necessary to determine the BW utilized by the currently viewed channel to then derive the remaining size of the channel cache. The average channel size per mix can be calculated from the data previously presented in Table 5.1 as shown in the following equations:

$$\begin{aligned} AverageChannelSizeMixA = \mu_{R_A} &= \sum_{forallR_x} (R_x \times \%_x) \\ &= (2\times 0.5) + (4\times 0.4) + (9\times 0.075) + (18\times 0.025) \\ &= 3.725 Mbps \end{aligned}$$

Equation 6-2

$$AverageChannelSizeMixB = \mu_{R_B} = \sum_{forallR_x}(R_x \times \%_x)$$
$$= (2\times 0.4) + (4\times 0.3) + (9\times 0.25) + (18\times 0.05)$$
$$= 5.15Mbps$$

Equation 6-3

$$AverageChannelSizeMixC = \mu_{R_C} = \sum_{forallR_x}(R_x \times \%_x)$$
$$= (2\times 0.35) + (4\times 0.15) + (9\times 0.4) + (18\times 0.1)$$
$$= 6.7Mbps$$

Equation 6-4

Using the average channel size per mix as the estimate of the expected size of the currently viewed channel the BW available for video caching is then computed as the selected total BW rate (50, 65, or 75 Mbps) minus the currently viewed channel size above. These milestones along with the mechanics of each model will be used to determine the number of channels that can be included in the resulting cache.

PROBABILITY ONLY MODEL PREDICTIONS:

The Probability Only Model makes use of the calculated probabilities of all offered channels based on the history buffer. The simulations repeatedly fill this buffer based on the use of six lambda (λ) values in the exponential equation. Figure 6.7 below shows an example of this exponential function after normalization with the probability of all the 149 available remaining channels equal to 1.0 as required for the PDF.

Each available channel represented on *x*-axis of Figure 6.7 is shown in order of probability or favorite channel status. Figure 6.8 depicts that the #1 favorite channel in this case has a probability of P(x=1) = 0.0952. Note that for Mix A each favorite channel has BW probabilities associated with it as defined in Figure 5.1, and these different BW probabilities can be used to define a 2nd Order view of this distribution based on the four

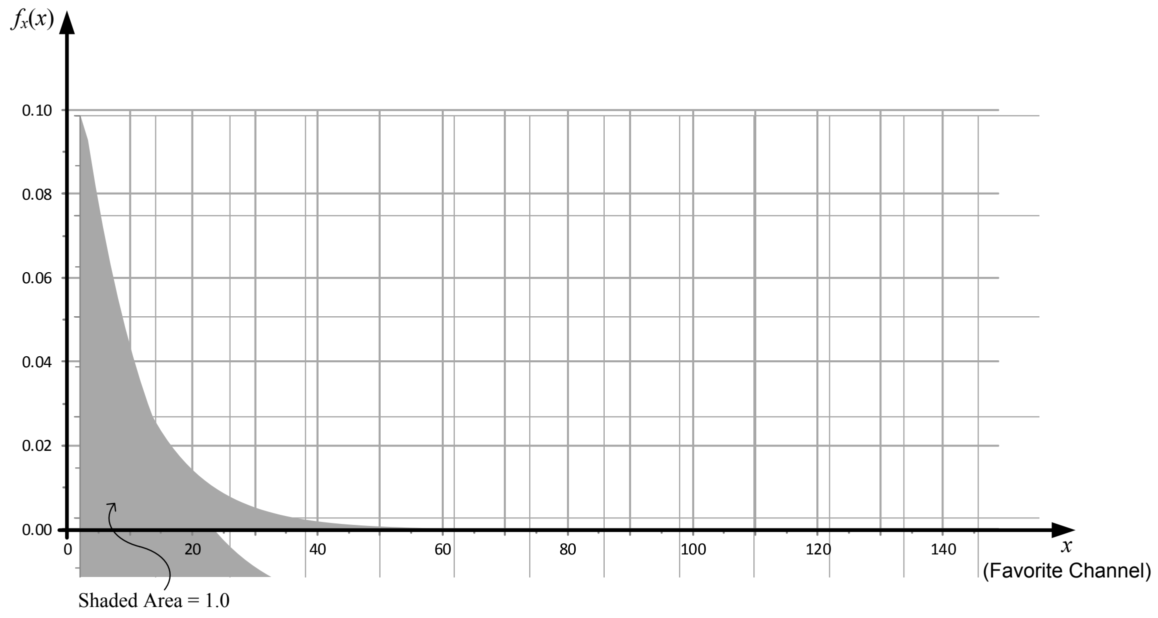


Figure 6.7 Full PDF for Exponential Distribution with λ = 0.10 on Mix A

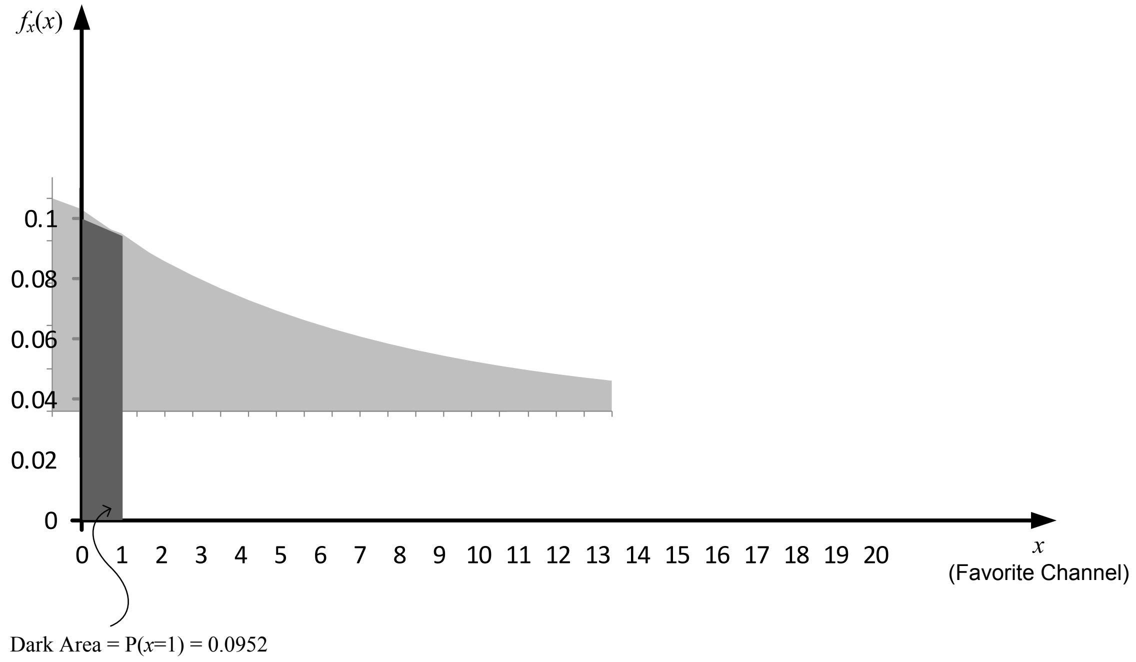


Figure 6.8 Partial View of λ = 0.10 Exponential with #1 Favorite Highlighted

BW options available. Figure 6.9 shows this 2nd Order PDF with its respective BW probabilities determined as the area under each BW portion equal to the rate probabilities assigned in Figure 5.1.

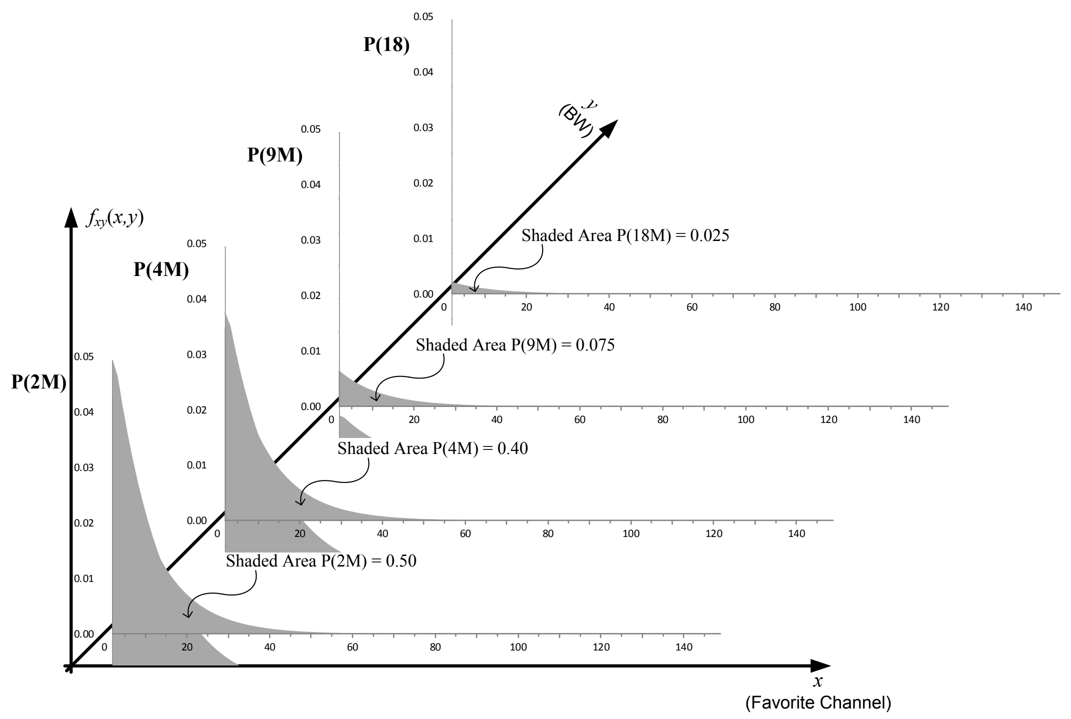


Figure 6.9 Probability Only Model 2nd Order PDF for Exponential λ = 0.10, Mix A

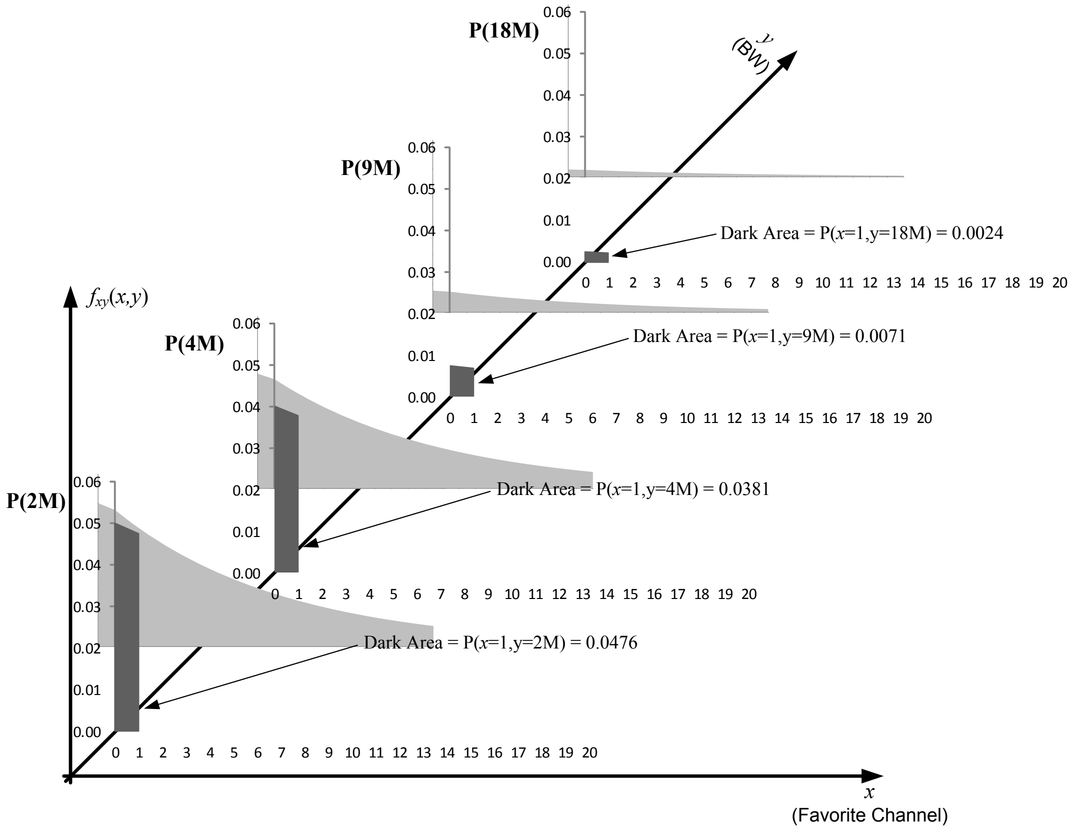


Figure 6.10 Partial 2nd Order PDF of λ = 0.10 and #1 Favorite Highlighted by BW

Note that the combined probabilities of the four portions of the 2nd Order PDF in Figure 6.9 are still equal to 1.0 for the combined full PDF. Again we can look at each individual channel in the 2nd Order view to then find the resulting BW probability for each of the four rates possible. Figure 6.10 more clearly shows the individual probabilities by BW for favorite channel #1. It also follows that the probability of favorite #1 can be defined by the summation shown in Equation 6-5 below.

$$\begin{aligned} P(x=1) &= P(x=1, y=2M) + P(x=1, y=4M) + P(x=1, y=9M) + P(x=1, y=18M) \\ &= (0.0476) + (0.0381) + (0.0071) + (0.0024) \\ &= 0.0952 \end{aligned}$$

Equation 6-5

Using this concept applied to the Probability Only Model allows us to calculate the resulting net cache probability as follows in Equation 6-6 with ***m*** channels cached.

$$\begin{aligned} &P(cache) = \\ &= \sum_{k=1}^{m} P(x=k, y=2M) + P(x=k, y=4M) + P(x=k, y=9M) + P(x=k, y=18M) \end{aligned}$$

Equation 6-6

The remaining task required for Probability Only Model prediction is to determine the expected number of channels ***m*** included in the caches associated with each mix and total video BW rate available. This expected number of cached channels is calculated by subtracting the current channel size from the total video BW and then dividing by the average channel size. The decimal remainder from this calculation must also be taken into account as the model mechanics will choose smaller BW sized channels as needed to maximize the available total video BW. Table 6.1 shows the results

Selected	Expected # of Channels Cached @ Selected Rates		
Mix	50 Mbps	65 Mbps	75 Mbps
Mix A	13	17	20
Mix B	10	13	15
Mix C	8	10	11

Table 6.1 Probability Only Model Expected Number of Cached Channels

of these calculations for each mix and total available BW rate.

With the number of cached channels from Table 6.1 repeatedly assigned as variable ***m*** in Equation 6-6, the cache net probabilities for each simulated conditions was determined using Excel to calculate the results found in Table 6.2 for the Probability Only Model. Note the significant increase in cache probability found with higher total video BW rates. It can also be seen that the cache probability increases for channel mixes containing fewer HD channels, such as Mix A. Users that have fewer favorite channels, depicted by higher λ values, also result in a higher cache probability.

λ value	0.10	0.09	0.08	0.07	0.06	0.05	Average
Mix A @ 50 Mbps	0.7275	0.6896	0.6465	0.5975	0.5417	0.4782	0.6135
Mix A @ 65 Mbps	0.8173	0.7835	0.7433	0.6958	0.6395	0.5729	0.7087
Mix A @ 75 Mbps	0.8647	0.8347	0.7981	0.7534	0.6989	0.6325	0.7637
Mix B @ 50 Mbps	0.6321	0.5934	0.5507	0.5034	0.4512	0.3937	0.5208
Mix B @ 65 Mbps	0.7275	0.6896	0.6465	0.5975	0.5417	0.4782	0.6135
Mix B @ 75 Mbps	0.7769	0.7408	0.6988	0.6501	0.5935	0.5279	0.6647
Mix C @ 50 Mbps	0.5507	0.5132	0.4727	0.4288	0.3813	0.3299	0.4461
Mix C @ 65 Mbps	0.6321	0.5934	0.5507	0.5034	0.4512	0.3937	0.5208
Mix C @ 75 Mbps	0.6671	0.6284	0.5852	0.5370	0.4832	0.4233	0.5540
Average per λ	0.7106	0.6741	0.6325	0.5852	0.5314	0.4700	0.6006
Average for all Mix A	0.6953						
Average for all Mix B	0.5996						
Average for all Mix C	0.5070						
Average for all 50 Mbps	0.5268						
Average for all 65 Mbps	0.6143						
Average for all 75 Mbps	0.6608						
Average Probability for all Conditions	0.6006						

Table 6.2 Predicted Results for the Probability Only Model

PROBABILITY DIVIDED BY BANDWIDTH MODEL PREDICTIONS:

The Probability Divided by BW Model simulation results can be predicted in much the same way. Figure 6.11 shows a partial 2nd Order PDF of the Probability Divided by BW Model distribution following normalization for Mix A. Note that the combined probability of the four 2nd Order BW components again equals 1.0. These derived BW probabilities were then used in Equation 6-7 to determine the average BW for each channel to be included in the cache for Mix A. Calculations were performed for Mixes B and C respectively as shown in Equations 6-8 and 6-9. Dividing the total available cache BW by the average cache channel size resulted in the expected number of channels cached for each condition as presented in Table 6.3.

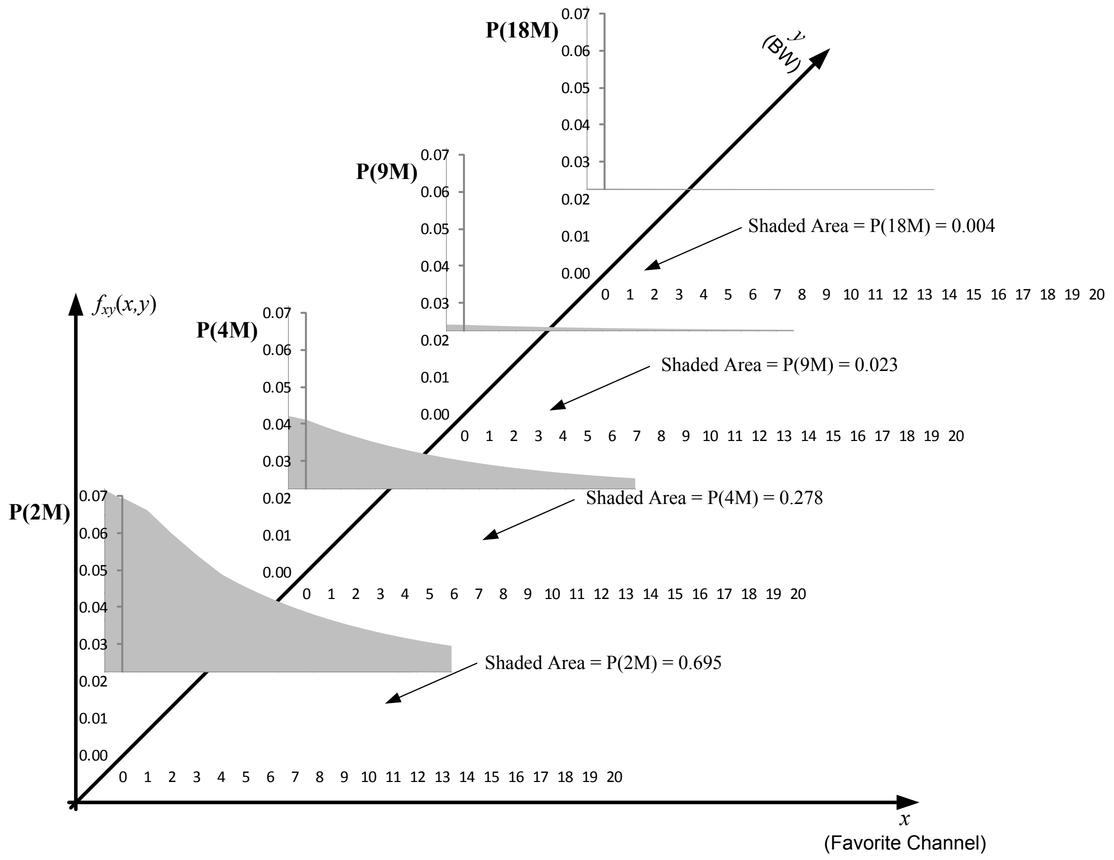


Figure 6.11 2nd Order PDF of Probability Divided by BW Model λ = 0.10, Mix A

$$
\begin{aligned}
AverageCacheChannelSizeMixA = \mu_{R_A} &= \sum_{forallR_x}(R_x \times \%_x) \\
&= (2\times 0.695)+(4\times 0.278)+(9\times 0.023)+(18\times 0.004) \\
&= 2.78Mbps
\end{aligned}
$$

Equation 6-7

$$
\begin{aligned}
AverageCacheChannelSizeMixB = \mu_{R_B} &= \sum_{forallR_x}(R_x \times \%_x) \\
&= (2\times 0.655)+(4\times 0.245)+(9\times 0.091)+(18\times 0.009) \\
&= 3.27Mbps
\end{aligned}
$$

Equation 6-8

$$
\begin{aligned}
AverageCacheChannelSizeMixC = \mu_{R_C} &= \sum_{forallR_x}(R_x \times \%_x) \\
&= (2\times 0.667)+(4\times 0.143)+(9\times 0.169)+(18\times 0.021) \\
&= 3.81Mbps
\end{aligned}
$$

Equation 6-9

Selected Mix	Expected # of Channels Cached @ Selected Rates 50 Mbps	65 Mbps	75 Mbps
Mix A	17	22	26
Mix B	14	18	21
Mix C	11	15	18

Table 6.3 Probability Divided by BW Model Expected Number of Cached Channels

Using the expected number of cached channels from Table 6.3 into Equation 6-6 on spreadsheet data values corresponding to the normalized 2nd Order PDF from Figure 6.11 allowed for the calculation of the net cache probability for each condition. These simulation predictions for the Probability Divided by BW Model are presented in Table 6.4 below. Again note the improved expected caching results when higher total video BW is allowed for caching. It can also be seen in Table 6.4 that fewer HD channels in a mix result in higher cache probability for this model. Users with higher λ values

λ value	0.10	0.09	0.08	0.07	0.06	0.05	Average
Mix A @ 50 Mbps	0.8173	0.7835	0.7433	0.6958	0.6395	0.5729	0.7087
Mix A @ 65 Mbps	0.8892	0.8619	0.8280	0.7856	0.7330	0.6675	0.7942
Mix A @ 75 Mbps	0.9257	0.9037	0.8751	0.8380	0.7900	0.7279	0.8434
Mix B @ 50 Mbps	0.7534	0.7163	0.6737	0.6247	0.5684	0.5037	0.6400
Mix B @ 65 Mbps	0.8347	0.8021	0.7631	0.7164	0.6605	0.5938	0.7284
Mix B @ 75 Mbps	0.8775	0.8489	0.8136	0.7701	0.7164	0.6504	0.7795
Mix C @ 50 Mbps	0.6671	0.6284	0.5852	0.5370	0.4832	0.4233	0.5540
Mix C @ 65 Mbps	0.7769	0.7408	0.6988	0.6501	0.5935	0.5279	0.6647
Mix C @ 75 Mbps	0.8347	0.8021	0.7631	0.7164	0.6605	0.5938	0.7284
Average per λ	0.8196	0.7875	0.7493	0.7038	0.6494	0.5846	0.7157
Average for all Mix A	0.7821						
Average for all Mix B	0.7160						
Average for all Mix C	0.6490						
Average for all 50 Mbps	0.6343						
Average for all 65 Mbps	0.7291						
Average for all 75 Mbps	0.7838						
Average Probability for all Conditions	0.7157						

Table 6.4 Probability Divided by BW Model Expected Simulation Results

representing fewer favorite channels result in higher cache probabilities here as well. These key trends are consistent with the results seen in the Probability Only Model and reflect important concepts to consider when analyzing the performance of this proposed system.

SAMPLE MOMENT MODEL PREDICTIONS:

The Sample Moment Model simulation results were also predicted using the 2nd Order PDF procedure. Each lambda value was used to generate six different exponential PDFs, and these were then normalized through only the first 149 values to yield the X_i values for all of the possible next channel selections. The Sort Field from Equation 4-8 was then used to generate new sort data associated with each channel for this model. For λ=0.10 and Mix A the following data values were used in the Sort Field:

$\boldsymbol{X_i}$ for each channel (from the normalized exponential PDF)

$\bar{\boldsymbol{x}}$ = 0.00671 (calculated using Equation 4-7)

μ_R = 3.725 (from Equation 6-2)

pf = 1.19 (initial penalty factor)

The resulting list of 149 calculated values from the Sort Field was then shifted positive (using calculated Shift Factor = 0.001403) to remove all negative values. This was followed by a normalization of all values to result in a new Full PDF for the Sample Moment Model (dividing each value by calculated Full Normalizing Factor = 0.209102). The individual BW probabilities were then calculated per channel using these new Full PDF probabilities, the available BW rates (2, 4, 9, & 18), and the Mix A BW probabilities from Table 5.1 to generate the 2nd Order PDFs shown on Figure 6.12.

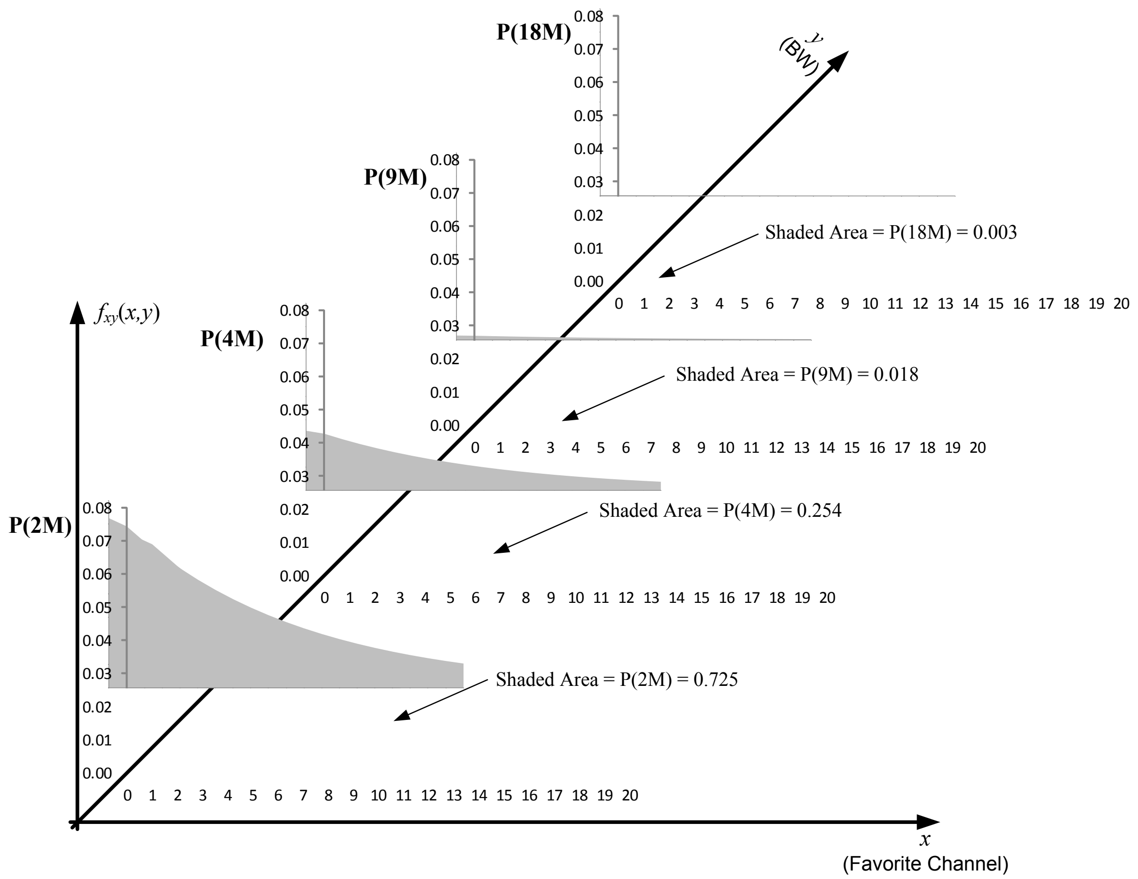


Figure 6.12 Partial 2nd Order PDF of Sample Moment Model with λ = 0.10, Mix A

The sum of all the individual BW probabilities for each channel yielded the new BW probabilities of selection for this example as follows:

P(2M) = 0.72498	P(4M) = 0.25421
P(9M) = 0.01816	P(18M) = 0.00265

Equations 6-10, 6-11, and 6-12 show how the various 2nd Order BW probabilities were used for each mix to calculate the average BW for channels cached with this model. The process again divided the total available cache BW by the average cache channel size resulting in the expected number of channels cached for the various conditions encountered as presented in Table 6.5 with supplemental details in Appendix F.

$$\begin{aligned} AverageCacheChannelSizeMixA = \mu_{R_A} &= \sum_{forallR_x} (R_x \times \%_x) \\ &= (2 \times 0.725) + (4 \times 0.254) + (9 \times 0.018) + (18 \times 0.003) \\ &= 2.68 Mbps \end{aligned}$$

Equation 6-10

$$\begin{aligned} AverageCacheChannelSizeMixB = \mu_{R_B} &= \sum_{forallR_x} (R_x \times \%_x) \\ &= (2 \times 0.693) + (4 \times 0.228) + (9 \times 0.072) + (18 \times 0.006) \\ &= 3.06 Mbps \end{aligned}$$

Equation 6-11

$$\begin{aligned} AverageCacheChannelSizeMixC = \mu_{R_C} &= \sum_{forallR_x} (R_x \times \%_x) \\ &= (2 \times 0.714) + (4 \times 0.134) + (9 \times 0.136) + (18 \times 0.015) \\ &= 3.46 Mbps \end{aligned}$$

Equation 6-12

Selected Mix	Expected # of Channels Cached @ Selected Rates 50 Mbps	65 Mbps	75 Mbps
Mix A	17	23	27
Mix B	15	20	23
Mix C	13	17	20

Table 6.5 Sample Moment Model Expected Number of Cached Channels

As with the previous predictions the expected number of cached channels from Table 6.5 entered in Equation 6-6 was used on the spreadsheet values corresponding to the normalized PDF from Figure 6.12 to yield the net cache probabilities for each condition. The simulation predictions for the Sample Moment Model are presented in Table 6.6. Again the same trends noted earlier regarding increasing total available video caching BW, reducing the HD channel presence, and increasing the exponential λ values are found to increase cache net probabilities for this model.

λ value	0.10	0.09	0.08	0.07	0.06	0.05	Average
Mix A @ 50 Mbps	0.8173	0.7835	0.7434	0.6960	0.6401	0.5750	0.7092
Mix A @ 65 Mbps	0.8997	0.8738	0.8412	0.8003	0.7492	0.6861	0.8084
Mix A @ 75 Mbps	0.9328	0.9120	0.8847	0.8492	0.8030	0.7437	0.8542
Mix B @ 50 Mbps	0.7769	0.7408	0.6989	0.6503	0.5941	0.5298	0.6651
Mix B @ 65 Mbps	0.8647	0.8347	0.7982	0.7536	0.6996	0.6347	0.7642
Mix B @ 75 Mbps	0.8997	0.8738	0.8412	0.8003	0.7492	0.6861	0.8084
Mix C @ 50 Mbps	0.7275	0.6896	0.6466	0.5977	0.5422	0.4800	0.6139
Mix C @ 65 Mbps	0.8173	0.7835	0.7434	0.6960	0.6401	0.5750	0.7092
Mix C @ 75 Mbps	0.8647	0.8347	0.7982	0.7536	0.6996	0.6347	0.7642
Average per λ	0.8445	0.8141	0.7773	0.7330	0.6797	0.6161	0.7441
Average for all Mix A	0.7906						
Average for all Mix B	0.7459						
Average for all Mix C	0.6958						
Average for all 50 Mbps	0.6628						
Average for all 65 Mbps	0.7606						
Average for all 75 Mbps	0.8090						
Average Probability for all Conditions	0.7441						

Table 6.6 Sample Moment Model Expected Simulation Results

ADDITIONAL MODEL PREDICTIONS:

Recall that the Exhaustive Search Model uses “brute force” techniques to systematically look at various channel caching combinations. This approach did not lend itself to theoretical methods to predict its performance, so no prediction attempts were undertaken in this research on the Exhaustive Search Model.

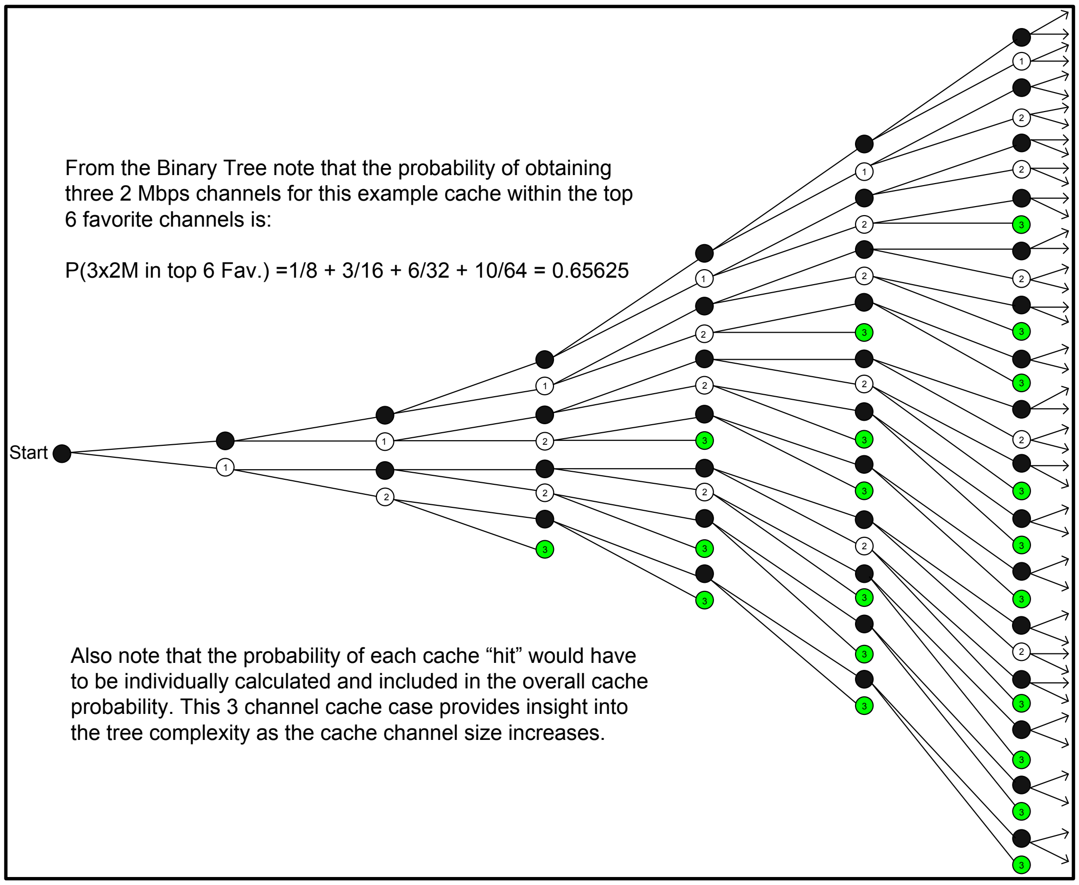
Due to the relatively high percentage assigned for 2 Mbps channels by each mix, the Maximum Channels Model is expected to only select these lower BW channels until all available caching BW has been fully utilized. The expected number of cached channels was calculated for this model using the original average channel size per mix from Equations 6-2, 6-3, and 6-4 along with this known 2 Mbps cached channel size to yield the data found in Table 6.7 below. This model also falls under the same conditions driving the Full and 2nd Order Exponential PDFs as the Probability Only Model that was seen in the example for Mix A presented in Figures 6.7 and 6.9 previously.

However, the Maximum Channels Model’s expected constraint to only cache 2 Mbps channels effectively eliminates our previous ability to take advantage of the 2nd Order BW probabilities from Figure 6.9, because only the 2 Mbps channels would be selected for caching in direct contradiction to the BW probabilities from the 2nd Order PDF. Therefore another technique is required to make an effective predictive analysis of the Maximum Channels Model.

Selected	Expected # of Channels Cached @ Selected Rates		
Mix	50 Mbps	65 Mbps	75 Mbps
Mix A	23	30	35
Mix B	22	29	34
Mix C	21	29	34

Table 6.7 Maximum Channels Model Expected Number of Cached Channels

The most promising technique analyzed that may yield accurate predictions for the Maximum Channels Model was the use of an event tree. Since the decision on whether or not to cache each favorite channel was based primarily on if it was a 2 Mbps channel or not, then this event tree approach could be followed through all of the various caching possibilities until the cache was filled to the number of channels listed above in Table 6.7. Figure 6.13 shows a partial example event tree through the first six favorite channels or tree layers for this model on Mix A. This particular example represents one of the easiest cases to attempt given its low cache number of 3 channels and the basic 2 Mbps channel probability of 0.5 present in Mix A. But even in this limited example it is



From the Binary Tree note that the probability of obtaining three 2 Mbps channels for this example cache within the top 6 favorite channels is:
P(3x2M in top 6 Fav.) =1/8 + 3/16 + 6/32 + 10/64 = 0.65625
Start
Also note that the probability of each cache “hit” would have to be individually calculated and included in the overall cache probability. This 3 channel cache case provides insight into the tree complexity as the cache channel size increases.

Figure 6.13 Event Tree Example for Maximum Channels Model to Obtain a 3 Channel Cache on Mix A with P(2M) = 0.5

clear that the size and complexity of this partial tree increases with each favorite channel included in the tree. The complete event tree would actually have branch layers all the way out to favorite channel #149, but in practice this may not be fully necessary since the probabilities fall off dramatically and may eventually be ignored as successive layers are added. The event tree also grows considerably as the size of the cache increases, but this method does represent a potential predictive solution and could be favorably assisted by event tree software.

Selected Mix	Expected # of Channels Cached @ Selected Rates 50 Mbps	65 Mbps	75 Mbps
Mix A	3	5	6
Mix B	5	4	6
Mix C	4	4	5

Table 6.8 HD Model Expected Number of Cached Channels

The expected number of cached channels is low for the HD Model as calculated using the processes described earlier and as presented in Table 6.8. This model's complexity is derived not by a high number of cached channels, which is not present, but by the model's mechanics which will start filling in lower BW channels into the cache as room allows. A similar computer assisted event tree approach could again be used for accurate predictions of the HD Model, but this is left for future research in this field.

6.4 Performance Simulation Results

The Performance Simulation Results Summary is shown on Figure 6.14 with the mean expected weighted probabilities listed for all input samples by the bandwidth rate and exponential λ for each of the six models developed in Chapter IV. All of the expected weighted probabilities listed are the mean values per identified attribute cached. For instance, the mean net expected probability of 0.7193 for all samples and conditions simulated by the Sample Moment Model is the average summation of the weighted probabilities of the caches for all of the 54 simulations. The mean expected weighted probabilities listed for each bandwidth rate include all of the λ values simulated at that rate per model. The mean weighted probabilities of the λ values include all bandwidth rates simulated at that specific λ value.

Figure 6.14 lists the models in decreasing order of overall performance rates. The Sample Moment Model had the highest overall performance, and it had the highest success level for all bandwidth rates and for all but one of the λ values in comparison to all of the other models. The Probability Divided by Bandwidth Model also performed well during the simulations proving that it was a viable model for use across the full spectrum of tested conditions. The HD Model was the only model simulated that would typically have less than a 50% success rate with a mean expected weighted probability for all input samples of only 23.94%.

A Cumulative Summary for the top performing Sample Moment Model can be seen on Figure 6.15. This summary shows the performance results for all 54 test runs of this model, and it is displayed in a multiple ways by rate and by λ value. It is evident that if a user provides the system more bandwidth for video caching, then the success rate of pre-selecting the next channel to be viewed increases. Also evident is the predicted result that users with a smaller number of favorite channels, as expressed by an increased λ value, have better prediction success rates from the caching models. Of particular note to examine is that only three of the 54 simulations for this model indicate an expected

Model Simulated:	Sample Moment Model	Probability Divided by Bandwidth Model	Probability Only Model	Exhaustive Search Model	Maximum Channels Model	HD Model
Mean Probability for all Samples =	0.7193	0.7180	0.6719	0.6601	0.5809	0.2394
Mean Prob. for 50 Mbps Rate =	0.6457	0.6442	0.5877	0.5964	0.5008	0.2070
Mean Prob. for 65 Mbps Rate =	0.7345	0.7342	0.6881	0.6726	0.5965	0.2602
Mean Prob. for 75 Mbps Rate =	0.7775	0.7754	0.7399	0.7114	0.6454	0.2510
Mean Prob. by Lambdas: λ = 0.05	0.6093	0.6080	0.5373	0.5514	0.4155	0.1973
λ = 0.06	0.6602	0.6620	0.6081	0.6142	0.5456	0.1962
λ = 0.07	0.7782	0.7774	0.7377	0.7180	0.6611	0.2195
λ = 0.08	0.7418	0.7418	0.7221	0.7053	0.5923	0.2701
λ = 0.09	0.7337	0.7277	0.6890	0.6683	0.6157	0.1902
λ = 0.10	0.7925	0.7908	0.7371	0.7036	0.6554	0.3631

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.6376	0.6152	0.3476	0.6535	0.5868	0.5386	0.8058	0.7019	0.6148	0.8091	0.7427	0.4538	0.7764	0.6315	0.5714	0.8290	0.7188	0.5887
Mean Prob. for 50 Mbps Rate =	0.6457																	
Probability at 65 Mbps Rate =	0.7347	0.6946	0.4381	0.7504	0.6558	0.6079	0.8931	0.7738	0.7070	0.8832	0.8189	0.5671	0.8533	0.7451	0.6581	0.9120	0.8306	0.6976
Mean Prob. for 65 Mbps Rate =	0.7345																	
Probability at 75 Mbps Rate =	0.7866	0.7275	0.5014	0.8068	0.6979	0.6439	0.9302	0.8071	0.7699	0.9225	0.8540	0.6246	0.8778	0.7849	0.7043	0.9490	0.8646	0.7424
Mean Prob. for 75 Mbps Rate =	0.7775																	

Mean Probability for all Samples = 0.7193

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.5335	0.5929	0.7075	0.6686	0.6598	0.7122
Mean Prob. by λ for 65 Mbps Rate =	0.6225	0.6713	0.7913	0.7564	0.7522	0.8134
Mean Prob. by λ for 75 Mbps Rate =	0.6718	0.7162	0.8357	0.8004	0.7890	0.8520
Mean Prob. for all Rates by λ =	0.6093	0.6602	0.7782	0.7418	0.7337	0.7925

am

success rate of less than 50% for the pre-selection outcome. The mean weighted probability outcome of the simulations run at the highest rate of 75 Mbps for users with only about 25 favorite channels ($\lambda = 0.1$) ended up with successful outcomes of greater than 85%. The Sample Moment Model overall result of 71.93% mean net weighted probability indicates that the vast majority of the time this model would be expected to successfully pre-select the next channel to be viewed and result in a CCT reduction of [(2 x Network Delay) + (IGMP Processing)] for all successful caches. The Cumulative Summaries covering this same level of detail for all of the other models can be found in Appendix F.

Table 6.9 shows a comparison between the predicted and simulated results for the models that were fully predicted. The Sample Moment Model and Probability Divided by Bandwidth Model simulation results were within 2.5% of the predicted expected results. The Probability Only Model simulated at over 7% higher than expected. This deviation could potentially be attributed to a reliance on averaging techniques used in several key computations for the predicted results. Prior to any actual attempted field deployment of this model, additional predictive analysis should focus on theoretical techniques that eliminate this potential source of error in determining the expected results.

The key conclusions to be drawn from the Performance Simulation results are that several different modeling techniques each have reasonably high expectations of

Model	Predicted	Simulated	% Difference
Sample Moment	0.7441	0.7193	-2.48%
Probability Divided by BW	0.7157	0.7180	0.23%
Probability Only	0.6006	0.6719	7.13%

Table 6.9 Comparison Between Predicted and Simulated Results

successfully predicting the next channel to be viewed and that higher bandwidth rates for video services dramatically improves the caching results. The Performance Simulation results were all within 7.13% or less of the predicted results. The Probability Only Model relying solely on the weighted probabilities would be expected to correctly predict and cache the next channel selection more than 60% of the time on average. By introducing the key concept of penalizing and rewarding channel sort fields based on their bandwidth usage, the Probability Divided by Bandwidth Model increased the expected system effectiveness to well above 70% on average. The additional complexity and data parameters added into the modeling process allowed the Sample Moment Model to become the overall top performer with gains to above 74% average predicted effectiveness and a near 72% success rate simulated across all conditions tested.

The remaining chapters of this research document lay out additional proposals on how to implement the system, cover additional improvements that could be investigated in the future, and discuss other applications where this same caching approach may be beneficial.

CHAPTER VII

IMPLEMENTATION OF MULTICAST CACHING

The previous chapters provided the justification, rationale, details, and evaluations of this multicast caching proposal. The sections in this chapter identify and analyze the key issues that must be considered in order to successfully implement this proposal on actual provider networks. Several suggestions are also made concerning potential improvements to the overall system that could be further developed through additional research in the future.

7.1 Operational Considerations

The proposed system of multicast caching does not require a direct change in any of the applicable standards, but all of the required devices will have to be capable of integrating the model routines and handling the multiple IPTV streams. Router devices from each vendor have distinct features and firmware that may or may not allow easy integration of the multicast caching system without major engineering redesign. These individual devices will have to be evaluated, upgraded, and redesigned as needed to be able to support the features of the proposed system. Based on laboratory testing with a variety of STBs, there does not appear to be interoperability issues pertaining to the additional multicast streams as these client devices simply ignore the other streams that are not selected for viewing. Client HGs may or may not require modifications primarily dependent on whether the device is used in bridge or router mode and on whether IGMP Snooping is available. But due to the fact that this proposed system does not rely heavily on upgrades to customer client devices, the primary efforts to implement the system involve planned modifications to the network side of the IPTV process.

Any other CCT reduction methods that are intended to be used simultaneously with the multicast caching system would have to be analyzed for proper integration as well. In many cases such as with I-frame management, these methods involve CCT reductions in areas totally unaffected by pre-fetching the next channel for viewing, so the integration would not likely be of major concern. However, some of the other CCT improvements may directly affect the design and performance of the multicast caching system, such as the methods involving specialized tune-in servers.

One operational consideration concerns how the system behaves during the initial start-up condition. At the time of the initial start-up, the channel change history buffer would be empty and there would be no data available for the models to predict the next channel change. With each new channel change the history buffer slowly gains more information for the models to use in their cache determination. This expected condition will consistently improve over time until the history buffer has enough entries to reflect the user's preferences. During this process the proposed system would only be expected to provide "best effort" performance based on the limited amount of information in the history buffer. Users should be made aware of this initial condition through customer documentation and product warnings so they will not expect optimal CCTs until the system has had time to gather their preferences.

The device or port reset condition is an area that requires some consideration. As the system would typically be implemented to numerous customers that are fed IPTV streams from multiple ports on the same LHR edge device, each port of the device would run distinct instances of the model routine. Care should be taken during the design phase to make sure that normal resets of affected devices do not void out the channel change historical buffers and associated logs for each port. An alternative would be to have software capabilities to restart the multicast caching system for individual ports or for the entire device based only on verified technician inputs to the device. Power down scenarios along with soft and hard resets of an entire routing device should not result in the loss of the multicasting information. A full reset of a device to factory default conditions would be expected to clear out all of the caching system information and result in the system behaving the same as its initial start-up condition.

The pre-selected channels streamed as part of the multicast cache are designed to improve CCT, but they do fill up a significant portion of the unused available bandwidth. It is possible that in certain situations a burst of data or other traffic may need to receive preferential treatment over the cached channels. This would occur more frequently with users that have higher-end devices with larger data rate limits. A large part of this issue could be addressed up front with the rate selection parameter set to a lower rate for video services, such as lowering the rate from 75 Mbps to 65 Mbps. In this case optimal CCT performance would be sacrificed to reserve a larger amount of bandwidth for non-video services. More granularity could be added to the process by allowing video bandwidth usage as define by a user to be assigned in 1 Mbps increments. A dynamic approach to this problem would be to prioritize the multicast cache to much lower levels than other traffic. This would allow a burst of data packets to have priority over the multicast cache packet stream. If the burst continued for long enough to be reflected in the available bandwidth rate fields, then the system would automatically reduce the size of the cache to maintain appropriate rates for the other non-video services. For short term burst situations that did not affect the available bandwidth rate fields, the cached traffic could be preempted by other traffic through use of a variety of standardized prioritization and QoS techniques within the Internet Protocol (IP). One of these techniques is the use of the Type of Service (TOS) field within the IPv4 header by adjusting the Precedence Bits (P-Bits) and TOS subfield for the different types of traffic [98-99][111]. The TOS field was upgraded to the Traffic Class field in the IPv6 header with expanded capability to distinguish differentiated services. Virtual Local Area Networks (VLAN) can also be used to designate different routing and treatment of identified packets [112-113]. By assigning the multicast cache a lower precedence through the P-Bits or tagging it with a specific VLAN, the higher priority traffic packets can be directed to jump ahead of the cached stream packets in the packet processing queues. The use of these techniques would require further system evaluation and testing to quantify the capabilities, functionality, and reliability of the process for the specific devices deployed, although the concept has been standardized in general for many years. Further details about these prioritization techniques can be found in Appendix G.

A variety of performance logs and metrics for each instance of the multicast caching system should be kept along with overall device-level logs. These indicators should reflect key aspects of the system’s performance including historical items such as the overall caching success rates, the numerical values of model variables, and the processing load required for the system. Some of these caching related metrics could be designed to be routinely uploaded to higher-level equipment or a Network Management System (NMS) to allow for network level performance analysis. Thresholds could be established to trigger warnings and alarms based on poorly performing network components or unusual conditions.

7.2 Processing and Memory Implications

A key objective of an effective multicast caching system is that it stays relatively current with respect to viewer preferences. The optimal situation would be that the system can update the cache of pre-selected channels in between each channel change. Table 7.1 shows the timing results for a series of processing tests run on the Sample Moment Model. These processing tests were run on a modest personal computer with the following capabilities and configuration:

Operating System	**Linux – Fedora Release 12**
Linux Kernel	**Version 2.6.32**
Processor	**Dual Core T5270 @ 1.4 GHz each**
RAM	**2 GB**

The Sample Moment Model routine was recompiled to run on this Linux-based machine, and the Linux “time” utility command was run on the executable program of this model to return the actual amount of time spent processing the application for each of the eighteen tested inputs. Tracks of the CPU usage verified that only one of the Dual Core processors was used to run the application. These tests using only moderate CPU

Test #	Processing Time (ms)
1	15
2	11
3	13
4	9
5	8
6	13
7	12
8	13
9	9
10	13
11	11
12	12
13	12
14	11
15	12
16	11
17	11
18	10
Average (ms)	**11.4**

Table 7.1 Processing Tests of the Sample Moment Model

speeds indicate that the processing requirements of this proposed system are not prohibitively large, and the system would typically have enough time to update the cache between channel changes given the 11.4 msec recorded average.

This tested average processing time is meant only as a benchmark to determine the processing feasibility of the system. Several factors would be different during an actual deployment that would affect the resulting processing times seen in the field. In addition to different CPUs and processing speeds, a field system would be designed so that it would not have the additional requirements used in this testing to open a file, read in the input data, write out the output data, and close a file. The channel change history buffer and all key variables could simply be kept in Random Access Memory (RAM) for quick access, although the system could periodically save this data for safekeeping. The channel probability updates and the time-weighting routines were simulated in this test through insertion of additional coding steps to fill representative arrays since these functions were already completed in the minimized sample data format. The Sample Moment Model as written in C++ through this research was not fully optimized from a software perspective, so professional re-development of the routine would likely result in additional processing improvements. Nevertheless, these tests indicate that the Sample Moment Model does not have prohibitively high processing demands, and the general processing clock cycle requirements per port for each channel change event are approximated as follows:

(11.4 msec) x (1.4 G cycles/s) = 15.96 M cycles / port channel change

This result approximates the number of processing cycles that the current model program requires to update the multicast cache after each channel change on a per port basis. This estimate provides an insight into the processing requirements for a wide range of equipment using a multiple of this result by the desired number of ports a particular device needs to support.

The speed of the processing would be enhanced by retaining all of the data in RAM for immediate access for field deployments. This requires an analysis of RAM sizing to verify that each device will not be overburdened by excessive memory requirements. In an analysis similar to the processing tests, the peak RAM usage required by the systems was tested on the same Linux-based personal computer. These RAM tests were conducted by comparing the peak memory usage near the end of the test with the usage at the beginning of the program run. The additional coding used to perform these tests was inserted into the appropriate positions of the Sample Moment Model, and the details of this routine can be seen in Appendix G. The Linux structure “getrusage( )” was the key component of this routine, and it provided the value of the “maxrss” variable indicating the set size in Kilobytes of memory usage at queried points in time. With the system variables fixed at the same levels used in the Chapter VI simulations, this testing consistently approximates the peak RAM requirements as follows:

Peak RAM Usage of Sample Moment Model = 264 KB per port

If the underlying system variables are changed, such as the history buffer size or number of channels offered, then the RAM requirements will be affected. However, these per port approximations do not indicate that the memory requirements for this system are prohibitively large. Furthermore, a streamlined professional software development plan would likely reduce the RAM usage significantly for actual implementation. Additional ongoing technical advancements, such as multi-core processing and embedded design optimization, will only increase the ability of network devices to be able to handle these system requirements.

While the processing and RAM tests validate the general ability of the system to be designed so that it routinely stays up to date, it would still be prudent for the system to not be functionally required to always meet that objective. If a viewer is rapidly changing channels at the same time that the processor is extremely busy with higher priority tasks, then a cache update time-out could be triggered that retains the previous cache through that particular channel change event and the “Failed Cache Processing %” metric field would be increased to reflect that the system did not complete the cache update in time. As noted earlier, there may or may not be changes to the cache in between channel changes based on the model results, so a slightly out of date cache would be highly preferential over no channels cached at all. As discussed in Chapter IV, the “Failed Cache Processing %” flagged set to No or an adjustment of the constant weighting parameter (**α)** would trigger the feedback adjustment routines to modify the size of the channel change history buffer to help decrease the processing demands. Proper design of the system would alleviate most of these concerns for the vast majority of channel change events.

There are several tasks of lower time sensitivity than updating the channel cache, and these tasks can be limited to run only when overall CPU demands are low. These tasks include the variable feedback adjustment routines and uploading performance metrics to higher network elements. Allowing these routines to run only when CPU usage is low ensures that these activities do not undermine the basic functionality of the system. A series of self-checks to verify process integrity could be similarly regulated based on the criticality of the situation. For instance, an alarm indicating that a key process has stopped would have a higher priority of resolution than sending out a periodic update to a process that has seen a period of inactivity. Clearly users will not be operating the system continuously and inactivity would be a common occurrence, but key processes should not close down unexpectedly. Careful consideration of decision flows such as these will further ensure that the multicast caching system operates effectively and does not impair other device functionality.

<u>7.3 Limitations</u>

The key limitation to understand first is that this proposed system does not expect to be 100% effective. From the simulation results based on a reasonably high bandwidth availability and 150 channels in the lineup, the top performing model was unsuccessful at predicting the next channel selection over a quarter of the time. From the laboratory testing it is known that there will be no reduced CCT for this system unless the multicast cache includes the next channel selected.

Two of the main issues that directly affect the success rate of the system are the bandwidth available for video services and the number of channels offered. It was shown in Chapter VI that the system's success rate increased by over 13% for the Sample Moment Model when the video bandwidth was increased from 50 Mbps to 75 Mbps. Similarly the system would see a decreased success rate if the number of channels offered increases, such as from 150 to 300. The guidance here is that the system should not be expected to maintain high success rates at reducing CCT if the channel lineup is dramatically increased without additional bandwidth being made available for the cached channels.

While the system operates within the scope of existing standards, each applicable device within the IPTV network will have to be capable of handling its part of the system. LHR devices will have to be able to run an instance of the selected model for each IPTV port on the device. These routers will have to meet the processing and memory requirements covered earlier in addition to being capable of streaming all the cached video channels. Additional capabilities to maintain logs, manage packet priorities, and update feedback processes must be present within these devices as well. These tasks do not conflict with the existing standards, but the functionality is not currently present on deployed equipment and would need development. Based on the reasonable processing and memory requirements analyzed earlier, it is expected that numerous existing devices could support the requirements through firmware updates to handle the implementation of the system without the need for wholesale hardware retrofits. This would not be an issue whatsoever for all new deployments once the functionality has been verified for the installation kit in laboratory testing and field trials. NMS systems would need additional software features to track network caching performance and metrics as well as being able to manage system alarms and warnings. With the overall focus of this proposal geared toward the network side, it is also worth noting that the entire network does not need to be upgraded simultaneously. The multicast caching system could be rolled out systematically over time to different areas of the network in a controlled fashion with minimal customer impact.

As will be discussed in Chapter VIII, this multicast caching technique has potential applications higher up in the network that are unrelated to the specific goal of CCT reduction. Some of these potential applications focus more on optimizing bandwidth usage, so it is possible that additional upgrades would be required at different levels of the network to handle the additional functionality if this path is selected.

7.4 Potential Improvements

There are several areas of improvement available beyond this current project scope including some potential gains to model development and some gains associated with other related processes. The recognition of existing special conditions could be incorporated into the system rather easily to take advantage of previous methods used for CCT reduction. Two clear examples of this potential improvement would be to recognize when a viewer is incrementing or decrementing channels sequentially and to identify when a user is toggling back and forth repeatedly between two channels. Another improvement would be to include functionality designed to optimize the critical periods of “high surfing” channel changes, such as during advertisements. These situations have already been covered by previous researchers [26][38][73], so when these conditions are identified then the channel pre-selection process could utilize overriding priorities for which channels to cache first. It was shown during the model development for this research that gaining additional information about each individual user results in a more knowledgeable prediction of their future channel change selections. Further research could continue down this path through even more advanced mathematical algorithms or by including non-mathematical facts about the current viewer. The next few paragraphs will explore some of the most promising directions that could be systematically explored in the future.

Premium services utilizing attributes such as time-of-day (TOD), day-of-week (DOW), and month-of-year (MOY) viewing histories could be offered to gain insight into a preferential user’s periodic viewing tendencies. For example, if the user has routinely watched a particular local news channel at 10:00 p.m., then a higher probability could be assigned to that channel during those nightly time-slots. It is intuitively obvious that these parameters would lead to more informed predictions of channel change selections, but several factors would be introduced into the probability criteria that must be addressed. Initially the channel change history buffer would need to be divided into bins representing the different time-slot parameters under consideration. A series of logic decisions would be required to integrate these new sorting parameters into the overall prediction process. A new weighting process would be required to determine how much emphasis should be put upon each of these new parameters. Consider the example where a viewer watched a particular channel at a certain hour the night before, but they did not watch that channel at all the previous week. This example may receive some higher weighting based on the previous night's activity, but perhaps not as much as if they consistently watched it nightly or even each week during that same time-slot. All of these types of weighting decisions would then need to be integrated into the overall probability calculations in some fashion to come up with the new sorting field for determining the multicast cache. It is clear that a path could be developed using these parameters for more informed decisions, but the additional complexities introduced must be well managed for an overall improvement to the system.

Another potential premium service for consideration would allow the system to gain specific knowledge about an individual viewer. If authorized by the user, key attributes could be included into the decision criteria that take into account individual factors such as the user's age, sex, education, and ethnicity. One method to help manage this could be a quick user identification from a known menu of previous users. For instance, a TV in the family room of a household may be periodically watched by different people at different times, so they could initially identify themselves when the STB is accessed from a simple menu listing the family viewing options. If it is known that a young child is watching the TV at a certain time, then the channel predictions could be quite different versus those made when the parent is watching. This parameter could be contingent on a premium user's acceptance and input. Other options could be taken into account to indicate if the TV is located in other types of facilities such as a hotel room or a business. The geographic location could be taken into account as well for weighing viewership expectations of sporting events or regionally oriented programs. For instance, viewers in Massachusetts are probably much more likely to watch a Boston Red Sox baseball game than viewers in New Mexico. An elaborate history could further be developed that tracks not only the channel numbers but also the actual programs associated with each viewing.

While there may be clear privacy concerns in some cases, even more in-depth knowledge of a user could be utilized by integrating other business unit information into the decision process. Consider the situation where a large MDU apartment complex has entered into a business arrangement with a telecommunications company to obtain bulk-rate IPTV services along with their other utilities. In these MDU cases, the apartment owner may have detailed knowledge of a viewer's occupation, salary, financial history, and credit that could be authorized for use in the caching process. Furthermore, the telecommunications provider could conceivably use a viewer's internet activity to gain insight into an individual's interests and hobbies. These "big brother" cases may sound a bit far-fetched at first glance right now, but an increasingly large effort at targeted advertising has been ongoing for several years already to pave the way forward for these types of individualized services [114]. All of these potential user identification parameters covered above do introduce additional complexities and further judgmental weighting problems that must be considered and overcome in order to add them into the caching process. Consider the situation where many of the above parameters are known about a user, then one key element of concern would be the logical decisions about how much emphasis must be placed on each parameter during full integration and prediction of the next cache list. Factors such as timing, memory, and processor requirements must be considered as well in order for these types of advanced criteria to be injected into the vast number of edge ports within a network.

Many of the existing CCT reduction techniques, such as dynamic I-Frame management, could be simultaneously used along with this proposed multicast caching system. A key existing technique that could hold significant promise for simultaneous use with this proposal would be the methods of dynamic video coding discussed in Chapter II. The concept of tune-in streams that use less bandwidth could be extremely beneficial to the channel caching process by allowing many more channels to be cached within a given bandwidth limit. This hybrid solution provides all of the CCT reductions associated with the dynamic video coding methods while at the same time significantly improves the effectiveness of the multicast caching solution. This particular hybrid approach represents one of the most recommended and optimistic areas of potential CCT improvement for future development given the steady rise in the number of offered IPTV channels.

CHAPTER VIII

ADDITIONAL APPLICATIONS

The models first presented in Chapter IV contain various algorithms and processes to maximize pre-streaming channels down to a user to reduce their CCTs. The top performing models predict the next channel to be selected while maximizing the overall probability of the pre-streamed cache for the next channel change. There are several other potential applications for this type of process in which a subset of video channels or live data streams can be pre-selected for transmission over a limited amount of bandwidth.

8.1 Downstream Video in the Network Backbone

Consider the example configuration depicted in Figure 8.1 showing a portion of a large future IPTV network with several downstream links. In this example the New York main office is the central transmission site for all of the 1500 network channels following advertising insertions and other edits, such as logos, as desired. The primary downstream long-haul links to the regional offices are at a rate of 10 Gbps. Links to various other smaller sites and hubs are at different rates depending on the size of the site demand, link availability, and cost. Additional links down to MDU locations similarly run at a variety of rates based on the number of users and cost concerns. The links within the MDU to

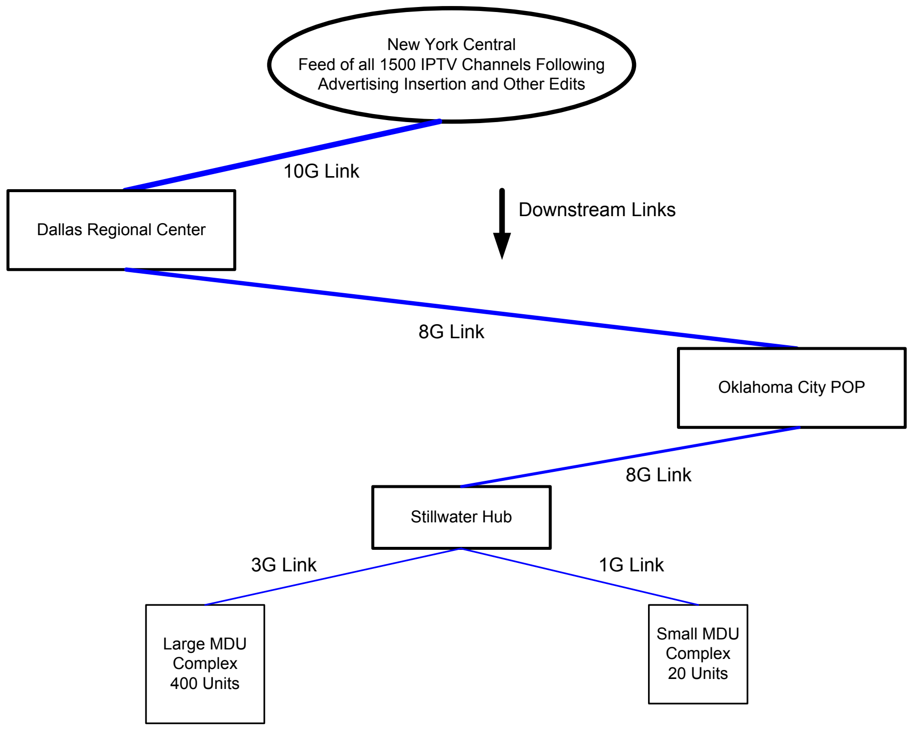


Figure 8.1 Example IPTV Downstream Links

individual users are assumed to be at standard 100 Mbps rates. Even as the cost for bandwidth has significantly decreased over time these various network links still represent enormous ongoing cost expenditures to the IPTV providers, so they consistently take significant steps to minimize this burden [115]. For this example let us assume that the average channel BW is 5.1 Mbps as in the Mix B case from Chapter V. Table 8.1 shows these downstream links by link BW, maximum channel carrying capacity, and planned number of channels to be carried simultaneously. Notice that all of the major markets have been designed to have all 1500 offered channels present to their location, but this is not needed or affordable when it comes to the destination markets like most MDUs with only a few hundred units or less.

In Table 8.1 we see that the 400 unit MDU has plans to receive 582 simultaneous channels while the 20 unit MDU is only sized to receive 194 channels. While the

Link Starting Location	Link Ending Location	BW (Gbps)	Max. # Channels	Planned # Channels
New York	Dallas	10	1941	1500
Dallas	Oklahoma City	8	1553	1500
Oklahoma City	Stillwater	8	1553	1500
Stillwater	Large MDU (400 units)	3	582	582
Stillwater	Small MDU (20 units)	1	194	194

Table 8.1 Downstream Link Metrics

provider sees these plans as more than enough to handle the expected demand from the respective MDUs, the channel limitations below the offered number of 1500 requires that certain channels be selected for streaming. If an MDU user picks a channel outside of the currently streamed list then clearly the CCT is expected to take longer. So the provider and its customers would clearly benefit from a system that would make the most effective use of its IPTV link capacity and increase the probabilities of customer QoE. The top performing model algorithms presented in this research would be well suited for this downstream video application.

8.2 Upstream Video Broadcasting

In a future IPTV network application a large number of live channels offered may tend to frequently exceed the limits of most network links. This basic oversubscription could be built into the provider's business plan by guaranteeing certain channels, offering "best effort" at providing others, and offering premium services to pay on demand for certain channel coverage. Figure 8.2 shows an example of an upstream portion of a network in which the source feeds exceed the upstream link limits. These source feeds could still be retained at the Regional Centers for viewing within the area even as they are not fully streamed back to the central New York office. But the ongoing routine decisions

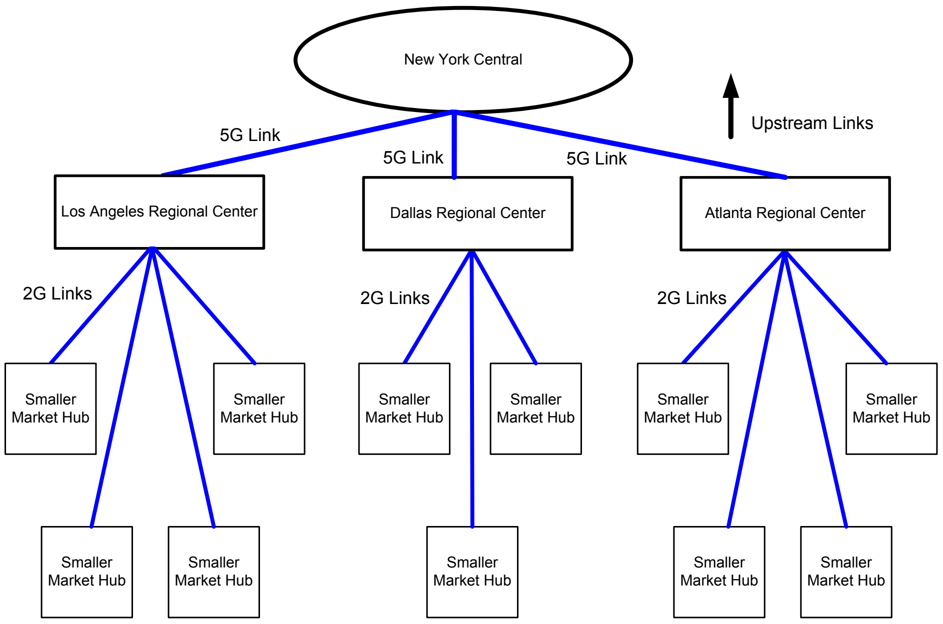


Figure 8.2 Example IPTV Upstream Links

regarding which channels to send on upstream form another suitable application for the caching algorithms presented in this research.

8.3 Remote Sensing and Other High Bandwidth Applications

Several fields of science have situations where a large amount of live data is captured at remote locations, but the transmission capacity from the remote location back for analysis may not be large enough to handle all of the live data feeds. In these cases the data is instead analyzed remotely with summaries uploaded periodically. This situation arises sometimes by design but sometimes by accident. Examples of this were the Mars Rovers and other space probes that sustained damage to one of their communications links and were forced to handle large amounts of data over the remaining links not sized for the entire data stream. If an operator selects specific data to be forwarded from a remote location, then that request should pre-empt most normal traffic. But over time in these BW limited cases, it is imperative that the communications link be utilized to its maximum capability. A couple of the caching algorithms presented in this research could take into account the BW price for certain channels or feeds in this case to maximize the overall remote transmission.

Another potential application of these algorithms involves managing video feeds back to control centers or security surveillance. Consider a corporation with numerous factories, buildings, or stores in which a large number of surveillance cameras are continuously recording ongoing operations. The control center may have the ability to manually view individual cameras over a remote link, but as the size of the network feeds grow then manually configuring which camera should be transmitted over the link poses an increasing burden. The caching algorithms could simplify this task while still allowing for manual override viewing as needed. Military command and control centers as well as airlines and hospitals could also be potential beneficiaries of this technology for their high BW situations.

CHAPTER IX

CONCLUSIONS

An analysis about the need for rapid channel change times on growing IPTV networks was initially presented, and the underlying processes that govern channel changing were examined in detail. This research advocated pre-streaming a multicast cache of video channels over available bandwidth to users in advance of their next channel change. The benefits of the multicast cache were verified in laboratory testing for cases in which the system had included the next channel change in the pre-streamed cache. This benefit was documented to typically be in the range of 220 to 420 msec in actual field practice for each correctly pre-streamed event.

Six different caching models were developed for a variety of situations, and mathematically based predictions were presented for several of these models. Each model was run through a series of simulations using sample data sets as inputs to the model covering a wide variety of expected channel changing behavior. All six of the models successfully passed the functionality simulations, and showed varying degrees of success in the performance testing. Feedback mechanisms were presented that would tune field systems for optimal performance. Implementation considerations such as processing and memory requirements were also presented to bridge the gap between this research and actual field deployment of the systems proposed. It was also shown that implementation of this system could take place primarily on network devices and potentially through firmware upgrades without the need for wholesale replacements or changes to existing IPTV standards. Related topics requiring further research or areas of potential improvement were discussed. Several additional potential applications of the models and algorithms were also presented.

The Probability Only Model makes caching decisions based on each channel's probability of selection as calculated from a history of channel changes stored in a device buffer over time. The concept of time-weighting the history buffer was proposed to give higher influence to more recent activity versus older events. Functionality and performance simulations of this model required the development of the sample data sets with a key element being the use of the exponential distribution to characterize user favorite channel behavior across six different exponential curves. Predictions of the model's performance in the simulations focused on PDFs of multiple random variable distributions in which each favorite channel's probability of selection along with its BW probability was analyzed. The Probability Only Model was predicted to correctly cache the next channel change roughly 60% of the time across the entire spectrum of conditions expected. The actual performance simulation results for this model over the 54 various simulation runs showed a 67% cache probability rate.

The Probability Divided by Bandwidth Model introduced the concept of rewarding and penalizing channels based on their low or high bandwidth rates. Similar predictions based on composite PDFs of this model indicate that it was expected to be successful in 71.6% of the performance simulations, which ended up being extremely close to the simulated results.

The Sample Moment Model was introduced with a sort field based on the channel probability minus the mean then divided by the bandwidth to a power factor. This sort field and algorithm was shown to be the most effective of those presented with predicted results of 74.4% success and simulated results of near 72% cache probability.

The Maximum Channels Model was presented as a method of optimally caching the highest number of channels. Performance predictions for this model were complicated in such a way that the composite PDF methods of prediction were not usable. An alternative predictive approach was presented using event trees, but this method was left for future research. The actual performance simulations did report this model to end up with an average of 58% cache probability.

The HD Model was initially developed prior to the laboratory testing which showed that HD channels only experienced minor increases in channel change times. But this model was geared to cache the highest BW channels first, and as such it resulted in a smaller cache size. The model performed poorly in the performance simulations with an expected success rate of only 24%.

The Exhaustive Search Model was developed in hopes of providing a benchmark for maximum model effectiveness by examining all possible channel cache combinations. The processing required to fully achieve those results was beyond the limits of this research, so a streamlined system was developed in which the best combination of up to six cached channels was determined in successive attempts using a “brute force” approach. A mathematical prediction was not undertaken for this technique, but the model did result in an average of 66% cache probability through the simulations.

The top performing of these models indicate that well over 70% of the time it is expected that users would experience significant channel change time reductions in the range of several hundred milliseconds if this proposed system was fully implemented on IPTV networks.

REFERENCES

[1] Y. Lee, J. Lee, I. Kim, H. Shin, “Reducing IPTV Channel Switching Time Using H.264 Scalable Video Coding,” IEEE Transaction on Consumer Electronics, Vol. 54, No. 2, May 2008.

[2] J. Zhang, Y. Wang, B. Rong, “QoS/QoE Techniques for IPTV Transmissions,” 2009 IEEE Symposium on Broadband Multimedia, TR2009-010, August 2009, Mitsubishi Electric Research Laboratories, Inc., 2009.

[3] “IPTV: The Global Picture,” November 9, 2009, http://www.emarketer.com/Reports/All/Em_iptv_sep06.aspx.

[4] “Global IPTV: Market Analysis and Forecast to 2011” RNCOS Industry Research Solutions, November 2007, http://www.asdreports.com/shopexd.asp?id=113.

[5] H. Joo, H. Song, D. Lee, I. Lee, “An Effective IPTV Channel Control Algorithm Considering Channel Zapping Time and Network Utilization,” IEEE Transactions on Broadcasting, Vol. 54, No. 2, June 2008.

[6] “42.2 Million Digital TV Users, 11.3 Million IPTV Users in Europe by 2010,” March 3, 2005, ZDNet Research, http://blogs.zdnet.com/ITFacts/?p=577.

[7] “Japan IPTV Market Outlook”, Research on Asia, Group, Inc., December 2008, http://www.researchandmarkets.com/reports/668313.

[8] “IPTV In Practice – The Korean Experience,” October 1, 2008, http://www.onscreenasia.com/article-3864-iptvinpracticethekoreanexperience-onscreenasia.html.

[9] “USA – Convergence – Triple Play & Quadruple Play,” Buddle.com, January 24, 2009, http://www.budde.com.au/Research/USA-Convergence-Triple-Play-Quadruple-Play.html.

[10] “China Advances Towards 1 Million IPTV Users,” Television Asia, October 1, 2007, http://www.encyclopedia.com/doc/1G1-169961018.html.

[11] K. Shah, “Telecom Giants Enter Fledgling IPTV Market,” November 19, 2007, http://www.expresscomputeronline.com/20071119/market04.shtml.

[12] B. Schechner, “ABI Research Finds Multi-Dwelling Units Providing Major Opportunity for Urban DSL Deployment in North America,” ABI Research, November 20, 2006, http://www.businesswire.com/news/home/20061120005536/en/ABI-Research-Finds-Multi-Dwelling-Units-Providing-Major.

[13] “Quick Facts: Apartment Stock,” National Multi-Family Housing Council, December, 2013, http://www.nmhc.org/Content.cfm?ItemNumber=55494.

[14] “RealPage Offers Triple Play in MDUs Using BitBand,” IPTV Pavilion, April 15, 2008, http://connectedplanetonline.com/iptvpavilion/technology/realpage-bitband-mdus-0415/index.html.

[15] J. Michaels, “Office Buildings: How Large Are They?,” January 3, 2001, http://www.eia.doe.gov/emeu/consumptionbriefs/cbecs/pbawebsite/office/office_howlarge.htm.

[16] P. Lambert, P. Debevere, J. Cock, J. Macq, N. Degrande, D. Vleeschauwer, R. Wall, “Real Time Error Concealing Bitstream Adaptation Methods for SVC in IPTV Systems,” J Real-Time Image Processing, 2009, pages 79-90.

[17] J. Seiler, A. Kaup, “Adaptive Joint Spatio-Temporal Error Concealment for Video Communication,” IEEE, Multimedia and Signal Processing, 978-1-4244-2295-1/08, 2008, pages 229-234.

[18] J. Suh, Y. Ho, “Recovery of Motion Vectors for Error Concealment,” IEEE TENCON, 0-7903-5739-6/99, 1999, pages 750-753.

[19] “Using IPTV Packet Loss Correction to Extend the Range of IPTV Services and Reduce Support Calls,” Digital Fountain, pages 43-48, http://www.iptvmagazine.com/2006_12/IPTVMagazine_2006_12_packet_loss_correction.htm.

[20] D. Kucera, “Introduction to MPEG-2 Compression and Transport Streams,” November 2002, Tektronix, Inc., pages 1-55.

[21] H. Fuchs, N. Farber, “Optimizing Channel Change Time in IPTV Applications,” IEEE, 2008.

[22] J. Lin, W. Lei, S. Bai, L. Li, “The Implementation of Fast Channel Switching in IPTV,” IEEE, Computer Society, 978-0-7695-3804-4/09, 2009, pages 684-688.

[23] “IGMP: Internet Group Management Protocol Overview,” Javvin Company, Network Management and Security, http://www.javvin.com/protocolIGMP.html.

[24] I. Kopilovic, M.Wagner, “A Benchmark for Fast Channel Change in IPTV,” IEEE, June 3, 2008.

[25] "C. Cho, I. Han, Y. Jun, H. Lee, "Improvement of Channel Zapping Time in IPTV Services Using the Adjacent Groups Join-Leave Method," The 6th International Conference on Advanced Technology, IEEE, October 4, 2004, pp. 971-975.

[26] C. Gan, P. Lin, C. Chen, "A Novel Pre-Buffering Scheme for IPTV Service," Elsevier Computer Networks, Vol. 53, 2009, pages 1956-1966.

[27] A. Begen, N. Glazebrook, W. Steeg, "Reducing Channel-Change Times with the Real-Time Transport Protocol," IEEE Internet Computing, May/June 2009, pp. 40-47.

[28] M. Sarni, B. Hilt, P. Lorenz, "A Novel Channel Switching Scenario in Multicast IPTV Networks," 2009 Fifth International Conference on Networking and Services, IEEE, 2009, pages 396-401.

[29] J. Lee, G. Lee, S. Seok, B. Chung, "Advanced Scheme to Reduce IPTV Channel Zapping Time," APNOMS 2007, LNCS 4773, pp. 235–243, 2007.

[30] M. Montpetit, H. Calhoun, "Adding the Community to Channel Surfing - A New Approach to IPTV Channel Change," IEEE, 978-1-4244-2309-5/09, 2009.

[31] "QoS-Enabled IP Networks for Assured Multiplay QoE," Nokia Siemens Networks, Technical White Paper, 2008, pages 1-20.

[32] M. Luby, J. Miller, "The Impact of Packet Loss on an IPTV Network," Digital Fountain, January 2007.

[33] "Case Study: How Network Conditions Impact IPTV QoE," Agilent Technologies, Inc., 2008, www.agilent.com/find/n2x.

[34] S. Mandal, M. MBuru, "Intelligent Pre-Fetching to Reduce Channel Switching Delay in IPTV Systems," Texas A & M, 2008, http://students.cs.tamu.edu/skmandal/research/channelswitching.pdf.

[35] C. Sasaki, A. Tagami, T. Hasegawa, S. Ano, "Rapid Channel Zapping for IPTV Broadcasting with Additional Multicast Stream," IEEE Communications Society, 978-1-4244-2075-9/08, 2008, pages 1760-1766.

[36] Y. Song, T. Kwon, "Fast Channel Change IPTV System for Enhanced User Experience," 2009 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, 978-0-7695-3762-7/09, 2009, pages 815-818.

[37] S. Malkos, E. Ucar, R. Akdeniz, "Analysis of QoE Key Factors in IPTV Systems: Channel Switching," IEEE, 978-1-61284-832-7/11, 2011.

[38] J. Koo, K. Chung, "Adaptive Channel Control Scheme to Reduce Channel Zapping Time of Mobile IPTV Service," IEEE Transactions on Consumer Electronics, Vol. 57, No. 2, May 2011, pages 357-365.

[39] D. Manzato, N. Fonseca, "A Survey of Channel Switching Schemes for IPTV," IEEE Communications Magazine, August 2013, pages 120-127.

[40] C. Lee, C. Hong, K. Lee, "Reducing Channel Zapping Time in IPTV Based on User's Channel Selection Behaviors," IEEE Transactions on Broadcasting, Vol. 56, No. 3, September 2010, pages 321-330.

[41] R. Rajah, "Optimizing Channel Change Time," Cisco Systems, Inc., 2008, pages 1-20, http://www.teamlightbulb.com/Rajah%20Cisco.pdf.

[42] "Zap Time," Wikipedia, http://en.wikipedia.org/wiki/Zap_time.

[43] F. Hodis, "IPTV Challenges and Metrics," Electro-Optical Engineering Inc., Application Note 174.1AN, 2008.

[44] A. Azgin, Y. Altunbasak, "A Semi-Distributed Fast Channel Change Framework for IPTV Networks," IEEE, 978-1-4577-0638-7/11, 2011.

[45] A. Begen, N. Glazebrook, W. Ver Steeg, "A Unified Approach for Repairing Packet Loss and Accelerating Channel Changes in Multicast IPTV," IEEE, 978-1-4244-2309-5/09, 2009.

[46] P. Siebert, T. Van Caenegem, M. Wagner, "Analysis and Improvements of Zapping Times in IPTV Systems," IEEE Transactions on Broadcasting, Vol. 55, No. 2, June 2009, pages 407-418.

[47] H. Uzunalioglu, "Channel Change Delay in IPTV Systems," IEEE, 978-1-4244-2309-5/09, 2009.

[48] "Delivering Video Quality in Your IPTV Deployment," Cisco Systems, Inc., White Paper, C11-380207-00, July 2008, pp. 1-9.

[49] "MPEG Program Allocation Table," IPTV Industry Dictionary, http://www.iptvdictionary.com/IPTV_Dictionary_MPEG_Program_Allocation_Table_PAT_Definition.html.

[50] "A Guide to MPEG Fundamentals and Protocol Analysis," Tektronix MPEG Tutorial, 2011, http://www.img.lx.it.pt/~fp/cav/Additional_material/MPEG2_overview.pdf.

[51] P. Egli, "Video Over IP: Video Codecs Overview," Peter Egli Publications, Rev. 1.10, 2003, pages 1-11.

[52] "Westinghouse LCD HDTV Monitor Specifications," http://www.westinghousedigital.com/details.aspx?itemnum=111#VALUE.

[53] T. Mastrangelo, "Bandwidth Metering: A Path to Empowerment?," Penton Media Inc., January 10, 2010, http://telephonyonline.com/residential_services/commentary/bandwidth-metering-empowerment-1215/.

[54] "Study to Assess Broadband Bandwidth Usage and Key Trends in Europe," Ventura Team LLP for the Fibre to the Home Council Europe, February 2008, pages 2-36, http://www.bloobble.com/broadband-presentations/presentations?itemid=2647.

[55] R. Bohn, J. Short, "How Much Information? 2009 Report on American Consumers," Global Information Industry Center, University of California, San Diego, December 9, 2009.

[56] E. Hall, "Implementing Prioritization On IP Networks," Network Computing, http://www.networkcomputing.com/915/915ws1.html.

[57] H. Song, D. Lee, H. Joo, "Method for Reducing Channel Change Time of Internet Protocol Television (IPTV) and IPTV Service Provision Server for Implementing the Same," Patent application number: 20080310518, http://www.faqs.org/patents/app/20080310518.

[58] P. Waddell, "Quality Issues in Audio/Video," Harmonic Inc., IEEE BTS Tutorial, September 11, 2008.

[59] J. Lias, "HDMI's Lip Sync and Audio-Video Synchronization for Broadcast and Home Video," Simplay Labs, LLC, August 15, 2008, pages 1-5, http://www.digitalhomedesignline.com/howto/210101788;jsessionid=UCTOE45ZX4HJVQE1GHPSKH4ATMY32JVN?pgno=1.

[60] P. Waddell, G. Jones, A. Goldberg, "Audio/Video Synchronization Standards and Solutions," Advanced Television Systems Committee, http://www.atsc.org/cms/pdf/audio_seminar/12%20-%20JONES%20-%20Audio%20and%20Video%20synchronization-Status.pdf.

[61] "Audio and Video Synchronization: Defining the Problem and Implementing Solutions," Linear Acoustic Inc., White Paper, 2004, Rev. 1, pages 2-8.

[62] I. Richardson, "White Paper: A Technical Introduction to H.264/AVC," Vcodex, http://www.vcodex.com/images/uploaded/342454811635986.pdf.

[63] S. Jacobs, A. Eleftheriadis, "Real-Time Dynamic Rate Shaping and Control for Internet Video Applications," Columbia University, Department of Electrical Engineering, March 1, 1997, pages 1-9.

[64] L. Merritt, R. Vanam, "Improved Rate Control and Motion Estimation for H.264 Encoder," IEEE, 1-4244-1437-7/07, 2007, pages 309-312.

[65] J. Cao, F. Li, J. Guo, "Spatial and Temporal Adaptive Error Concealment Algorithm for MPEG-2," IEEE, 0-7803-8253-6/04, 2004, pages 285-288.

[66] W. Bretl, "Introduction to MPEG 2 Video Compression," Zenith Electronics Corp., http://www.bretl.com/mpeghtml/mpeg2vc1.HTM.

[67] T. Anselmo, D. Alfonso, "Buffer-Based Constant Bit-Rate Control for Scalable Video Coding," STMicroelectronics, Advanced Systems Technology, November 2007, http://www.eurasip.org/Proceedings/Ext/PCS2007/defevent/papers/cr1072.pdf.

[68] D. Banodkar, K. Ramakrishnan, S. Kalyanaraman, A. Gerber, O. Spatscheck, "Multicast Instant Channel Change in IPTV Systems," COMSWARE 2008, IEEE, 978-1-4244-1797-1, 2008.

[69] "Fast Channel Changing in RTP," Internet Streaming Media Alliance, Technical Working Paper, Version 1.0, December 17, 2007, pages 1-14.

[70] A. Azgin, Y. Altunbasak, "Exploiting Unicast vs. Multicast Delivery Tradeoffs to Improve the Latency Performance for IPTV Channel Change," The 9th Annual IEEE Consumer Communications and Networking Conference, IEEE, 978-1-4577-2071-0/12, 2012, pages 748-753.

[71] K. Tian, Y. Cheng, X. Shen, "Fast Channel Zapping with Destination-Oriented Multicast for IP Video Delivery," IEEE Transactions on Parallel and Distributed Systems, IEEE, 1045-9219/12, 2012.

[72] Y. Bejerano, P. Koppol, "Improving Zap Response Time for IPTV," IEEE INFOCOM 2009, IEEE, 978-1-4244-3513-5/09, 2009, pages 1971-1979.

[73] A. Azgn, Y. Altunbasak, "Channel Reordering with Time-shifted Streams to Improve Channel Change Latency in IPTV Networks," 2011, http://arxiv.org/ftp/arxiv/papers/1111/1111.3711.pdf.

[74] S. Cheng, C. Chou, G. Horng, T. Chang, "Fast IPTV Channel Switching Using Hot-View and Personalized Channel Preload Over IEEE 802.16e," EURASIP Journal on Wireless Communications and Networking, 2012.

[75] M. Ahmad, N. Rehman, J. Qadir, A. Baig, H. Majeed, "Prediction-Based Channel Zapping Latency Reduction Techniques for IPTV Systems – A Survey," 2009 International Conference on Emerging Technologies, IEEE, 978-1-4244-5632-1/09, 2009, pages 466-470.

[76] F. Ramos, "Mitigating IPTV Zapping Delay," IEEE Communications Magazine, August 2013, pages 128-133.

[77] E. Lee, S. Park, J. Lee, "Cross-Layer Design of Internet Group Management Protocol for Mobile IPTV Services in WiMAX," The 5th International Conference on Systems and Networks Communications, IEEE Computer Society, 978-0-7695-4145-7/10, 2010, pages 294-299.

[78] S. Hsu, M. Wen, H. Lin, C. Lee, "AIMED – A Personalized TV Recommendation System," EurolTV, LNCS 4471, 2007, pages 166-174.

[79] M. Yamamoto, N. Nitta, N. Babaguchi, "Automatic Personal Preference Acquisition from TV Viewer's Behaviors," IEEE, 978-1-4244-2571-6/08, 2008, pages 1165-1168.

[80] J. Lim, S. Kang, M. Kim, "Automatic User Preference Learning for Personalized Electronic Program Guide Applications," Journal of the American Society for Information Science and Technology, Vol. 58(9), 2007, pages 1346-1356.

[81] Y. Kim, J. Park, H. Choi, S. Lee, J. Park, J. Kim, Z. Ko, "Reducing IPTV Channel Zapping Time Based on Viewer's Surfing Behavior and Preference," 2008 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 978-1-4244-1648-6, June 3, 2008, pages 1-6.

[82] J. Caja, "Optimization of IPTV Multicast Traffic Transport over Next Generation Metro Networks," The 12th International Telecommunications Network Strategy and Planning Symposium, IEEE, 3-8007-2999-7, November 2006, pages 1-6.

[83] J. Yoon, S. Park, S. Choe, "Implementation of EIGMP for Fast IPTV Channel Change in GEPON," IEEE Transactions on Consumer Electronics, Vol. 57, No. 2, May 2011, pages 484-491.

[84] H. Joo, J. Song, "QoE-Aware Mobile IPTV Multicast System Over WiMAX Network," The 9th Annual IEEE Consumer Communications and Networking Conference, 2012, pages 759-764.

[85] D. Bailey, J. Chen, S. Radhadrishnan, S. Karabuk, "Multi-Source IPTV Networks: Zap Time and Bandwidth Optimization," 2013 International Conference on Computing, Networking, and Communications, IEEE, 978-1-4673-5288-8/13, 2013, pages 665-670.

[86] "Motorola Mobility Release Latest Enhancements to its KreaTV Operating System for IPTV & Hybrid Set-tops," Motorola, May 2012, http://cloud-computing.tmcnet.com/news/2012/05/18/6310820.htm.

[87] "Camera Operations Guide DCR-HC30/HC40," Sony Corporation, 3-088-339-11(1), 2004.

[88] T. Ray, "Testing Performed to Resolve Sony DCR-HC40 Issue," Oklahoma State University, Electrical and Computer Engineering Department, Dissertation Notes, March 31, 2010, pages 1-5, http://hcove.net/Published_Literature.html.

[89] "QuickTime 7.6 User Guide," Apple Publishing 018-1270/2009-01-01, Apple Inc., 2009.

[90] "QuickTime Tutorial – Appendix A: QuickTime Confidential," Cycling '74, 2010, http://www.cycling74.com/docs/max5/tutorials/jit-tut/jitterappendixa.html.

[91] A. Papoulis, S. Pillai, Probability, Random Variable, and Stochastic Processes, 4th Edition, McGraw-Hill Companies, Inc., New York, 2002, pages 81-83, 101-105, and 278-279.

[92] R. Johnson, Probability and Statistics for Engineers, 7th Edition, Pearson Education, Inc., New Jersey, 2005, pages 212-213.

[93] J. Newmark, Statistics and Probability in Modern Life, 6th Edition, Harcourt Brace College Publishers, Florida, 1997, pages 134, 312, and 472-473.

[94] W. Bolstad, Introduction to Bayesian Statistics, 2nd Edition, John Wiley and Sons, Inc., New Jersey, 2007, page 131-133.

[95] B. Ostle, L. Malone, Statistics in Research, 4th Edition, Iowa State University Press, Iowa, 1988, pages 20-22 and 77-79.

[96] Z. Smith, C. Wells, "Central Limit Theorem and Sample Size," University of Massachusetts Amherst, October 2006, pages 1-22, http://www.umass.edu/remp/Papers/Smith&Wells_NERA06.pdf.

[97] D. Hunter, "Statistics 200 Honors – Homework Solutions," Penn State University, Statistics Department, September 3, 1999, pages 1-4, http://www.stat.psu.edu/~dhunter/200h/fall99/solutions/sol01.pdf.

[98] W. Stallings, High-Speed Networks and Internets, 2nd Edition, Prentice-Hall, Inc., New Jersey, 2002, pages 56-57, 315-330, 437-439, and 622-626.

[99] W. Stallings, Data and Computer Communications, 7th Edition, Pearson Education, Inc., New Jersey, 2004, pages 604-607 and 683-698.

[100] S. Sorin, "Exponential Weight Algorithm in Continuous Time," University of Pierre and Marie Curie, Paris, Dept. of Mathematics, Springer-Verlag, 2007, pages 1-13, http://www.math.jussieu.fr/~sorin/articles/exponentional%20weight%20algorithm%20in%20continious%20time.pdf.

[101] T. Li, “On Exponentially Weighted Recursive Least Squares for Estimating Time-Varying Parameters,” IBM T. J. Watson Research Center, December 9, 2003, pages 1-19, http://www.research.ibm.com/people/t/thl/myhtml/parafilter/papers/rls.pdf.

[102] R. Nau, “Averaging and Exponential Smoothing Models,” Duke University, Decision 411 - Statistical Forecasting, 2005, http://www.duke.edu/~rnau/411avg.htm.

[103] C. Murphy, “Moving Averages Tutorial,” University of Honk Kong, 2006, http://courses.jmsc.hku.hk/jmsc7008spring2012/files/2010/02/MovingAverages.pdf.

[104] S. Kim, Y. Ho, “A Fast Mode Decision Algorithm for H.264 Using Statistics of the Rate Distortion Cost,” IET Digital Library, 2008, http://digital-library.theiet.org/content/journals/10.1049/el_20083714?crawler=true.

[105] A. Polinsky, S. Shavell, “Enforcement Costs and the Optimal Magnitude and Probability of Fines,” The University of Chicago Press, Journal of Law and Economics, Vol. 35, No. 1, April 1992, pages 133-148.

[106] “Moment (in Statistics),” Science Encyclopedia, Jrank.org, http://science.jrank.org/pages/51491/moment-(in-statistics).html.

[107] M. Hallam, T. Rarick, “Video Transport and Distribution for IPTV Networks,” Tellabs, Inc., White Paper, 74.1716E, Revision A, October 2006, pages 1-7.

[108] T. Qiu, Z. Ge, S. Lee, J. Wang, J. Xu, Q. Zhao, “Modeling User Activities in a Large IPTV System,” 2009 Internet Measurement Conference, November 2009, pages 430-441, http://www2.research.att.com/~slee/pubs/iptv_imc09.pdf.

[109] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, X. Amatriain, “Watching Television Over an IP Network,” 2008 Internet Measurement Conference, October 2008, pages 71-83, http://an.kaist.ac.kr/~mycha/docs/imc05-cha.pdf.

[110] S. Ross, Introduction to Probability Models, 8th Edition, Academic Press, California, 2003, page 10-14, 23-36349-389, and 601-632.

[111] M. Hassan, R. Jain, High Performance TCP/IP Networking, Pearson Education, Inc., New Jersey, 2004, pages 12-29.

[112] D. Hucaby, S. McQuerry, “VLANs and Trunking,” Cisco Press, October 25, 2002, http://www.ciscopress.com/articles/article.asp?p=29803.

[113] S. Varadarajan, “Virtual Local Area Networks,” Ohio State University, Department of Computer and Information Science, August 14, 1997, http://www.cs.wustl.edu/~jain/cis788-97/ftp/virtual_lans/index.htm.

[114] S. Chang, M. Cevher, “An Investigation and Conceptual Models of Podcast Marketing,” Springer-Verlag, 2007, pages 264-275, http://www.springerlink.com/content/xj3224j16616452p/.

[115] J. Berthold, “Optical Networking for Data Center Interconnects Across Wide Area Networks,” 17th IEEE Symposium on High Performance Interconnects, IEEE, 1550-4794/09, 2009, pages 149-153.

[116] J. Maisonneuve, M. Deschanel, J. Heiles, W. Li, H. Liu, R. Sharpe, Y. Wu, “An Overview of IPTV Standards Development,” IEEE Transactions on Broadcasting, Vol. 55, No. 2, June 2009, 315-328.

[117] O. Levin, “IPTV Standards Perspective,” Joint ITU-T Workshop and IMTC Forum, May 2006, http://www.itu.int/ITU-T/worksem/h325/200605/presentations/s4p3-levin.pdf.

[118] D. Boswarthick, “ETSI IPTV Standards,” IPTV World Forum Asia, December 2008, http://docbox.etsi.org/zArchive/TISPAN/Open/Information/NGN_Presentations/IPTV%20Asia%20Dec08/Boswarthick_for_IPTV_Asia_Dec08.pdf

APPENDICES

APPENDIX A

Appendix A.1 List of Acronyms

4G	Fourth Generation of Cellular Wireless Standards
AVC	Advanced Video Coding
A/V Sync	Audio to Video Synchronization
B-Frames	Bi-Predictive Frames
BW	Bandwidth
CCT	Channel Change Time
CDF	Cumulative Distribution Function
CLT	Central Limit Theorem
DHCP	Dynamic Host Configuration Protocol
DOW	Day-of-Week
ECM	Entitlement Control Messages
EPG	Electronic Program Guide
FHR	First Hop Router
FTTH	Fiber-to-the-Home
GOP	Group of Pictures
GPON	Gigabit Passive Optical Network
GSI	Global Standard Initiative
HD	High Definition
HDMI	High Definition Multimedia Interface
HG	Home Gateway
IDR	Instantaneous Data Refresh
I-Frames	Intra-coded Frames
IGMP	Internet Group Management Protocol
i.i.d.	Independent and Identically Distributed
IP	Internet Protocol
IPTV	Internet Protocol Television
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JVT	Joint Video Team
LAN	Local Area Network
LHR	Last Hop Router
MAZA	Multicast Assisted Zap Acceleration

MDU	Multi-Dwelling Unit
MOY	Month-of-Year
MPEG	Motion Pictures Expert Group
NGN	Next Generation Network
NMS	Network Management System
PAT	Program Allocation Table
P-Bits	Precedence Bits
PC	Personal Computer
PDF	Probability Density Function
P-Frames	Predictive Frames
PMT	Program Map Table
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RAP	Random Access Point
RTCP	Real-Time Transport Control Protocol
RTO	Retransmission Timeout
RTP	Real-Time Transport Protocol
RTT	Round-Trip Time
SRTT	Smooth Round-Trip Time
STB	Set Top Box
STDev	Standard Deviation
SVC	Scalable Video Coding
TCP	Transmission Control Protocol
TOD	Time-of-Day
TOS	Type of Service
TS	Transport Stream
TSS	Time Shifted Sub-channels
UDP	User Datagram Protocol
VCEG	Video Coding Experts Group
VLAN	Virtual Local Area Network
VoD	Video on Demand
VoIP	Voice over Internet Protocol
WAN	Wide Area Network
WiMAX	Worldwide Interoperability for Microwave Access (802.16)
xDSL	Digital Subscriber Line

Appendix A.2 Primary Standards Related to IPTV

Additional IPTV Standards are detailed in references [116-118].

International Telecommunication Union (ITU):

ITU-T H.264	ITU-T G.9970	ITU-T G.9954
ITU-T H.222	ITU-T G.9963	ITU-T G.9951-3
ITU-T G.9972	ITU-T G.9960-1	ITU-T G.987

International Organization for Standardization / International Electrotechnical Commission (ISO/IEC):

ISO/IEC MPEG-4 AVC	ISO/IEC 13818	ISO/IEC 11172

Institute of Electrical and Electronics Engineers (IEEE):

IEEE 802.16	IEEE 802.3
IEEE 802.11	IEEE 802.1

Advanced Television Systems Committee (ATSC):

ATSC A/91	ATSC A/64	ATSC A/54
ATSC A/90	ATSC A/63	ATSC A/53
ATSC A/72	ATSC A/58	ATSC A/52
ATSC A/65	ATSC A/57	

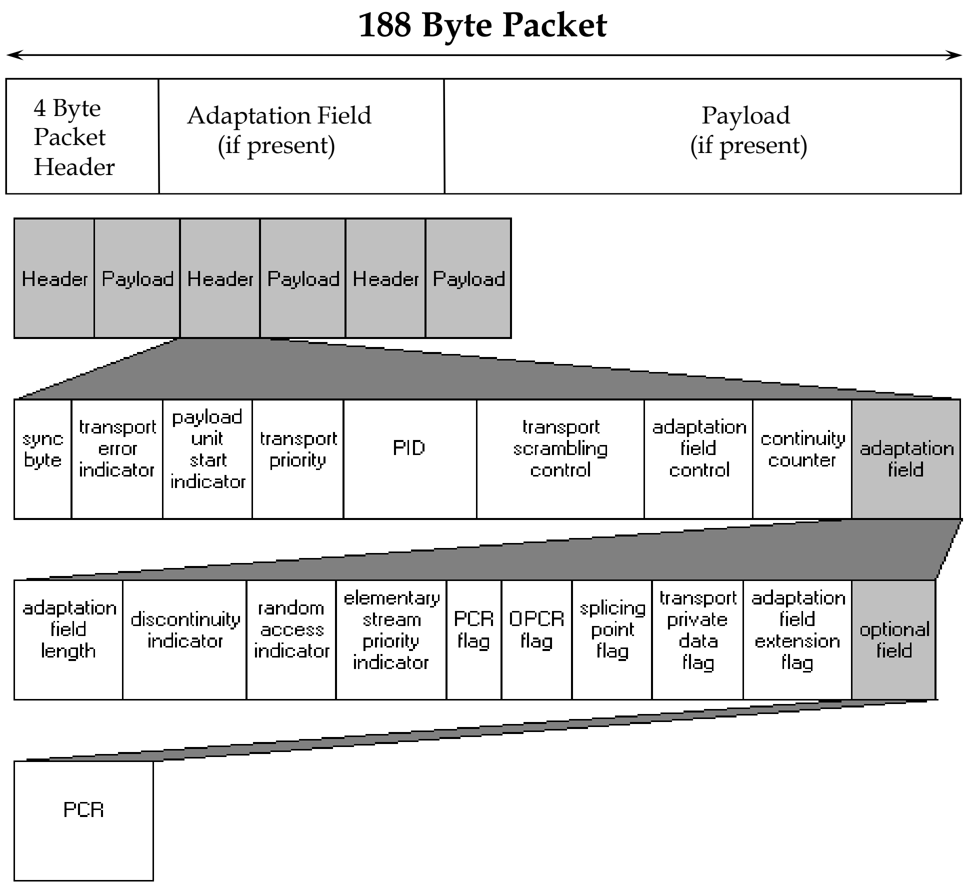
European Telecommunications Standards Institute (ETSI):

ETSC TS 185 011	ETSC TS 182 027
ETSC TS 185 009	ETSC TS 182 010
ETSC TS 185 007	ETSC TS 181 016
ETSC TS 185 003	ETSC TS 181 014
ETSC TS 183 065	ETSC TS 143 069
ETSC TS 183 064	ETSC TS 101 211
ETSC TS 183 053	ETSC TS 101 162
ETSC TS 182 028	ETSC TS 101 154

Internet Engineering Task Force (IETF):

RFC 3550	RFC 1122
RFC 3376	RFC 1112
RFC 2250	RFC 768
RFC 2236	

Appendix A.3 MPEG-2 Transport Stream Packet Structure



MPEG-2 Transport Stream Packet Structure [20]

The MPEG-2 Transport Stream flows at a constant bit rate with fixed length of 188 Byte packets. Each packet contains only one type of data (video, audio, guide, etc). The 4 Byte Packet Header consists of the following fields:

Sync Byte - 8 bits in form 0x47

Transport Error Indicator - 1 bit flag set if packet has uncorrectable error

Payload Unit Start Indicator - 1 bit flag for PES or PSI data otherwise zero

Transport Priority – 1 bit flag set for priority over packets with same PID

PID – 13 bit Packet Identifier

Transport Scrambling Control – 2 bit field for not scrambled or using even or odd key

Adaptation Field Control - 2 bits for adaptation field, payload, or combination

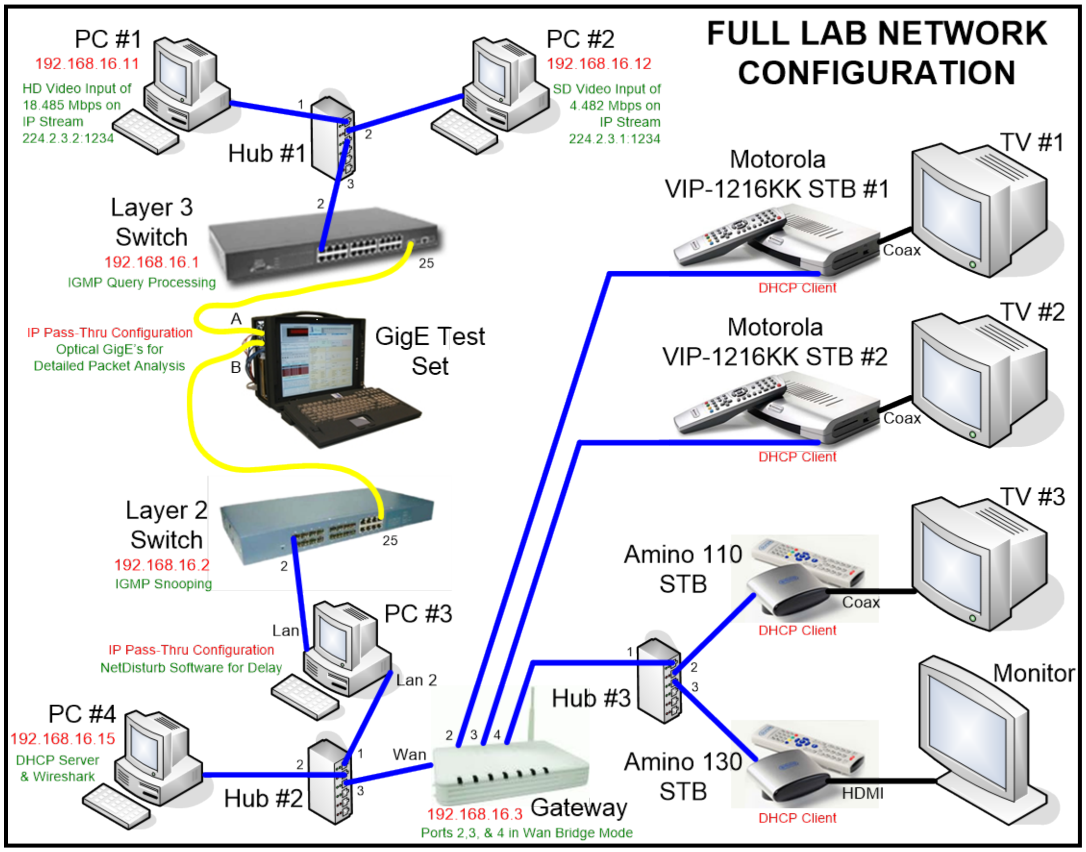
Continuity Counter – 4 bit field incremented when payload present only

Adaptation Field – 0 or more bits depending on flag settings with following subfields

- Adaptation Field Length – 8 bits to describe remaining size of field to follow
- Discontinuity Indicator – 1 bit flag if packet off with continuity counter or PCR
- Random Access Indicator – 1 bit flag if PES starts a video/audio sequence
- Elementary Stream Priority Indicator – 1 bit flag for higher priority
- PCR Flag – 1 bit set if packet contains a PCR field
- OPCR Flag – 1 bit set if packet contains an OPCR field
- Splicing Point Flag – 1 bit flag set if splice countdown field present
- Transport Private Data Flag – 1 bit flag if private data bytes are present in packet
- Adaptation Field Extension Flag – 1 bit flag if adaptation field extension present
- Optional Field -
    - PCR – 33+9 bits for program clock reference for A/V sync and timing
    - OPCR – 33+9 bits for original program clock reference
    - Splice Countdown – 8 bits to indicate packets to next splicing point
    - Stuffing Bytes – variable length used only as needed

APPENDIX B

Appendix B.1 Laboratory Equipment Details

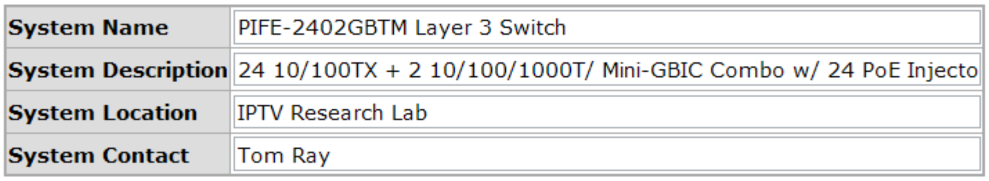


Full Laboratory Network Configuration

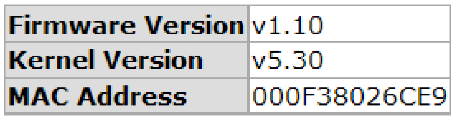
LanTech GE-24F2GBM Gigabit Managed Network Switches:

24 10/100TX + 2 10/100/1000T/ Mini-GBIC Combo w/ 24 PoE Injector Managed Industrial Switch

LanTech GE-24F2GBM Gigabit Managed Network Switch

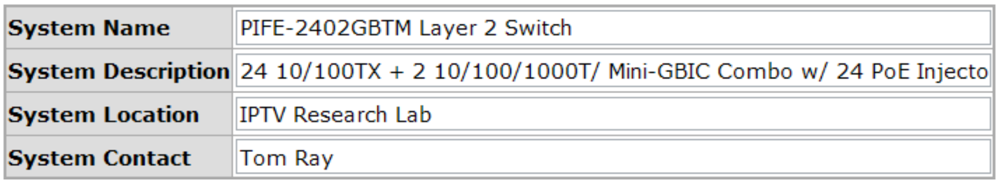


System Name	PIFE-2402GBTM Layer 3 Switch
System Description	24 10/100TX + 2 10/100/1000T/ Mini-GBIC Combo w/ 24 PoE Injecto
System Location	IPTV Research Lab
System Contact	Tom Ray

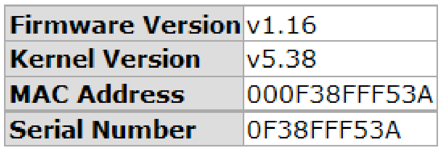


Firmware Version	v1.10
Kernel Version	v5.30
MAC Address	000F38026CE9

LanTech Layer 3 Switch General Information

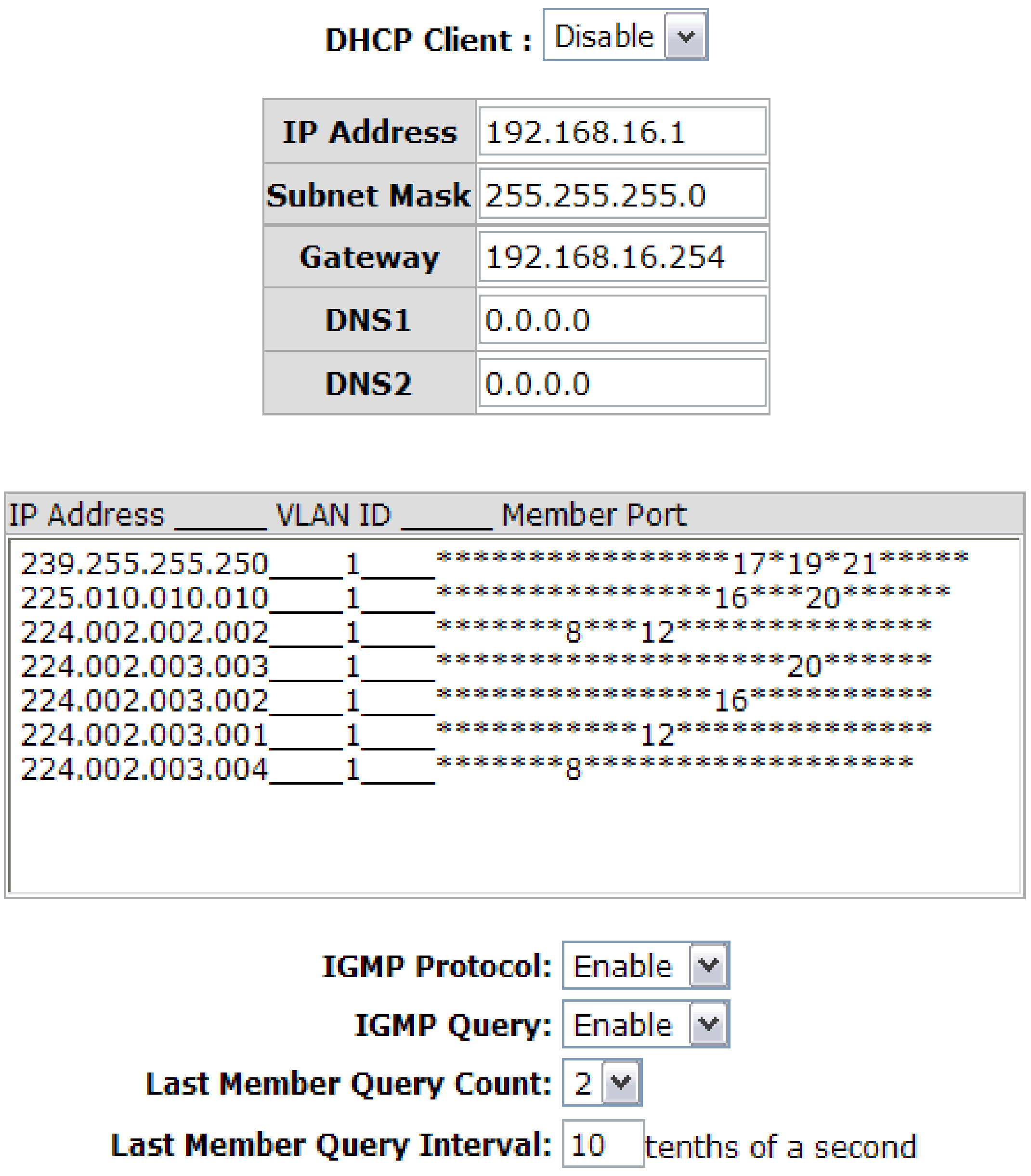


System Name	PIFE-2402GBTM Layer 2 Switch
System Description	24 10/100TX + 2 10/100/1000T/ Mini-GBIC Combo w/ 24 PoE Injecto
System Location	IPTV Research Lab
System Contact	Tom Ray



Firmware Version	v1.16
Kernel Version	v5.38
MAC Address	000F38FFF53A
Serial Number	0F38FFF53A

LanTech Layer 2 Switch General Information



DHCP Client : Disable

IP Address	192.168.16.1
Subnet Mask	255.255.255.0
Gateway	192.168.16.254
DNS1	0.0.0.0
DNS2	0.0.0.0

```
IP Address ______ VLAN ID ______ Member Port
239.255.255.250_____1_____****************17*19*21*****
225.010.010.010_____1_____***************16***20******
224.002.002.002_____1_____*******8***12**************
224.002.003.003_____1_____*******************20******
224.002.003.002_____1_____***************16**********
224.002.003.001_____1_____***********12**************
224.002.003.004_____1_____*******8******************
```

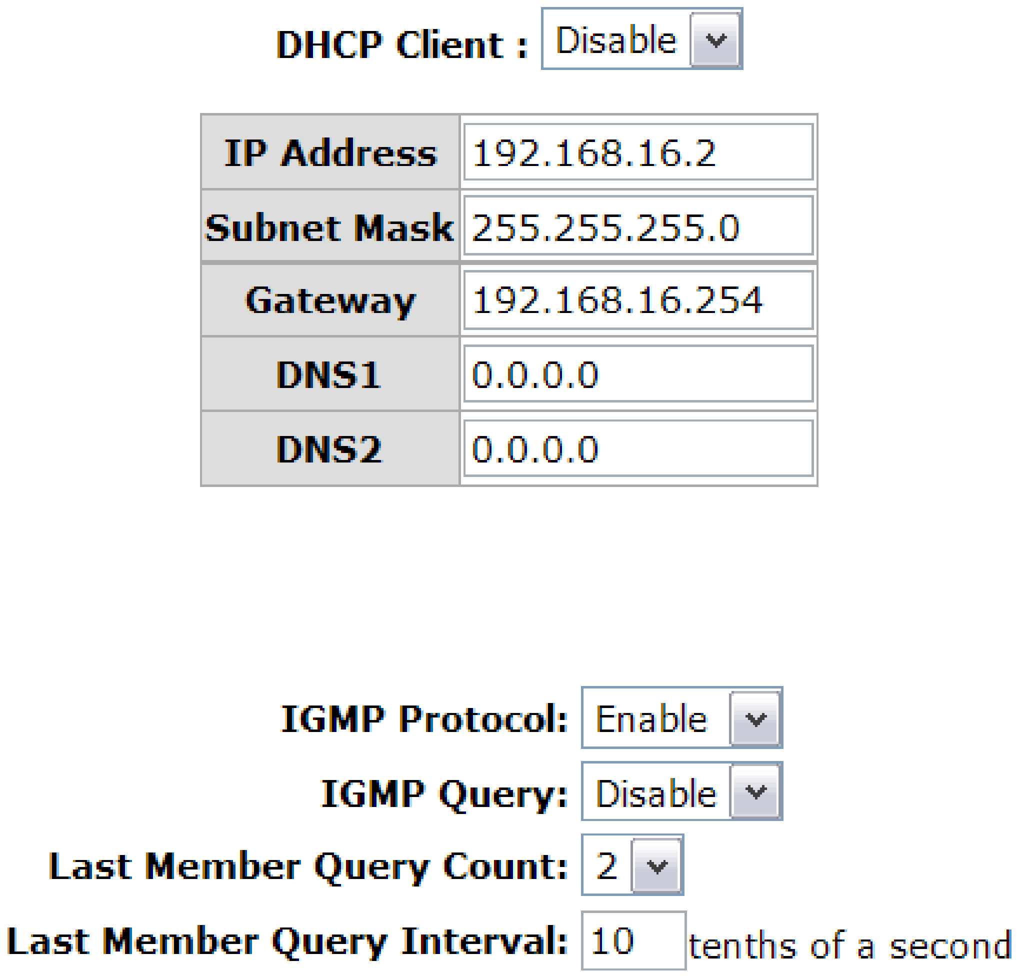
IGMP Protocol: Enable

IGMP Query: Enable

Last Member Query Count: 2

Last Member Query Interval: 10 tenths of a second

LanTech Layer 3 Switch IP and IGMP Configuration



DHCP Client : Disable

IP Address	192.168.16.2
Subnet Mask	255.255.255.0
Gateway	192.168.16.254
DNS1	0.0.0.0
DNS2	0.0.0.0

IGMP Protocol: Enable

IGMP Query: Disable

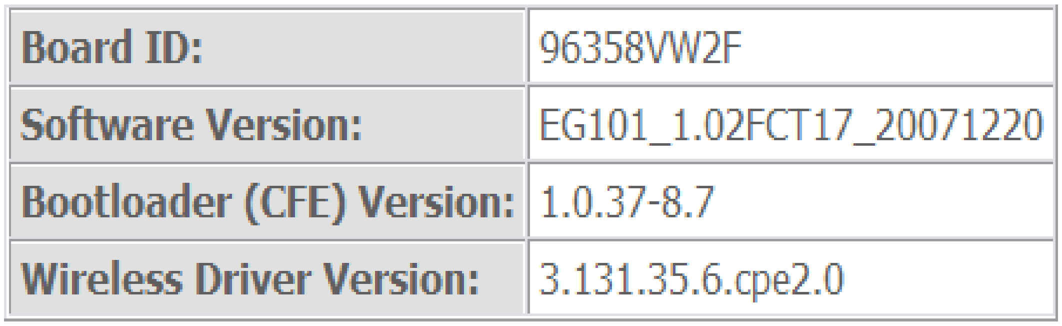
Last Member Query Count: 2

Last Member Query Interval: 10 tenths of a second

LanTech Layer 2 Switch IP and IGMP Configuration

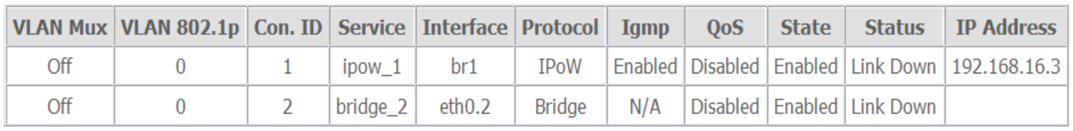
Xavi X-550 Home Gateway:

Xavi X-550 Home Gateway

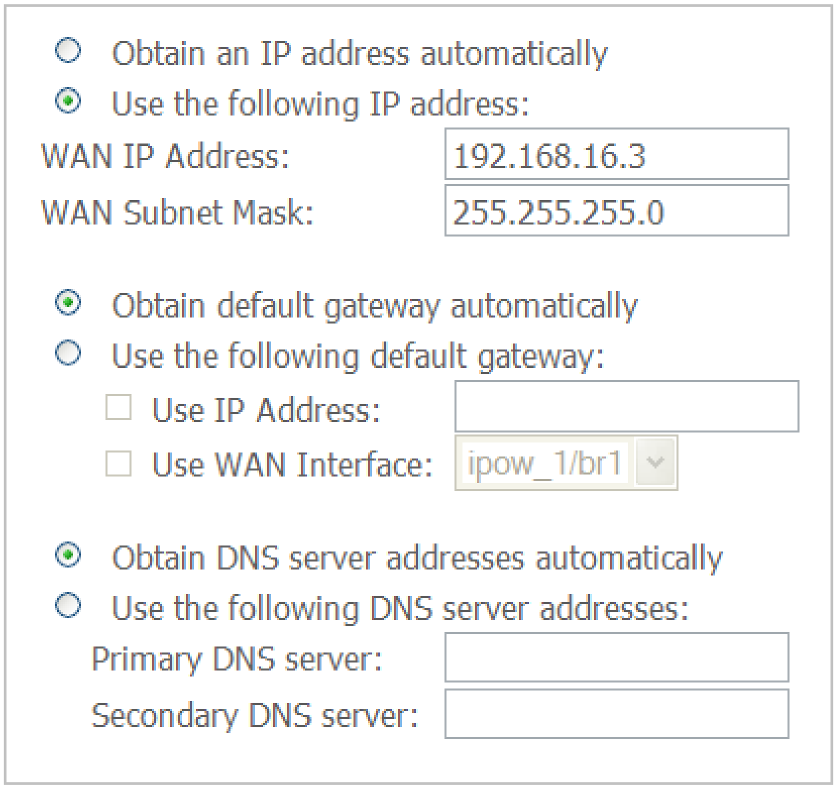


Board ID:	96358VW2F
Software Version:	EG101_1.02FCT17_20071220
Bootloader (CFE) Version:	1.0.37-8.7
Wireless Driver Version:	3.131.35.6.cpe2.0

X-550 General Information



VLAN Mux	VLAN 802.1p	Con. ID	Service	Interface	Protocol	Igmp	QoS	State	Status	IP Address
Off	0	1	ipow_1	br1	IPoW	Enabled	Disabled	Enabled	Link Down	192.168.16.3
Off	0	2	bridge_2	eth0.2	Bridge	N/A	Disabled	Enabled	Link Down	



○ Obtain an IP address automatically
● Use the following IP address:

WAN IP Address:	192.168.16.3
WAN Subnet Mask:	255.255.255.0

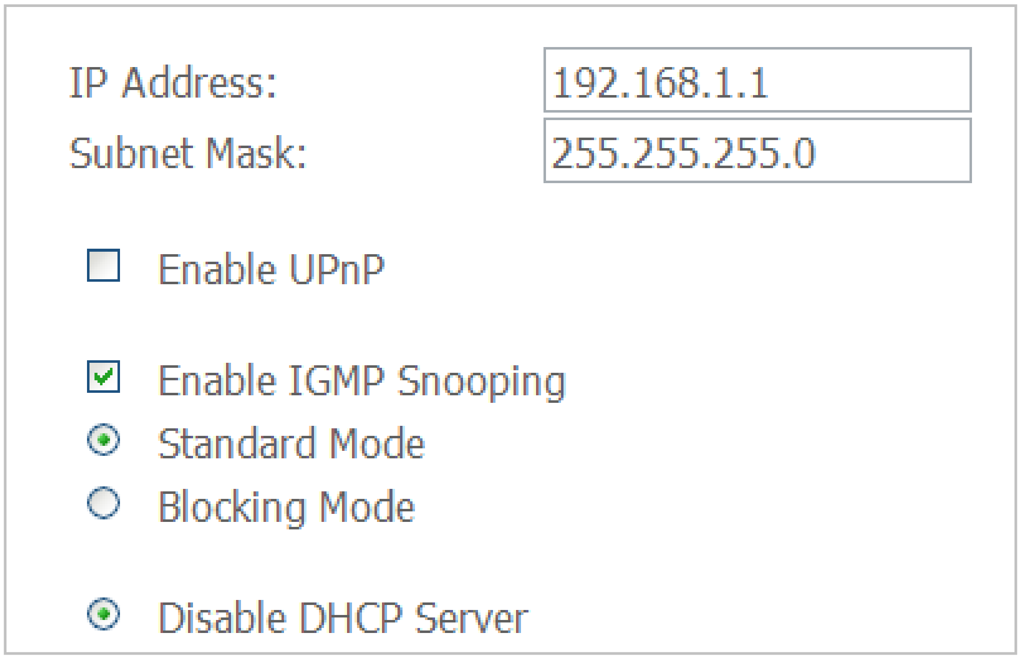
● Obtain default gateway automatically
○ Use the following default gateway:

☐ Use IP Address:	
☐ Use WAN Interface:	ipow_1/br1

● Obtain DNS server addresses automatically
○ Use the following DNS server addresses:

Primary DNS server:	
Secondary DNS server:	

X-550 WAN Configuration



IP Address:	192.168.1.1
Subnet Mask:	255.255.255.0

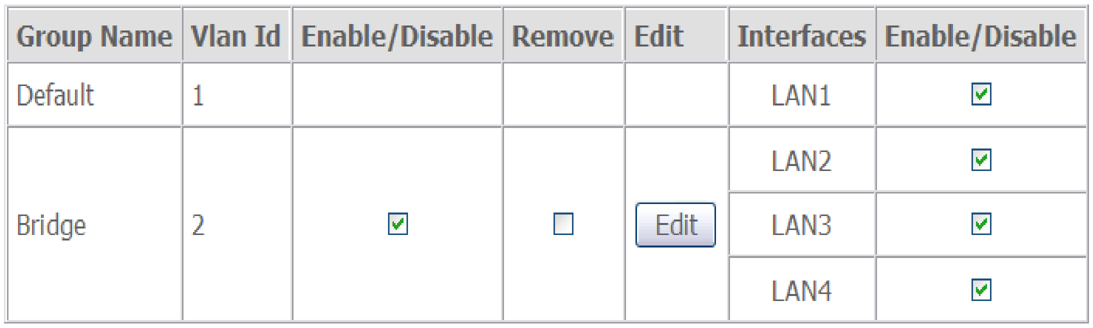
☐ Enable UPnP

✓ Enable IGMP Snooping

● Standard Mode

○ Blocking Mode

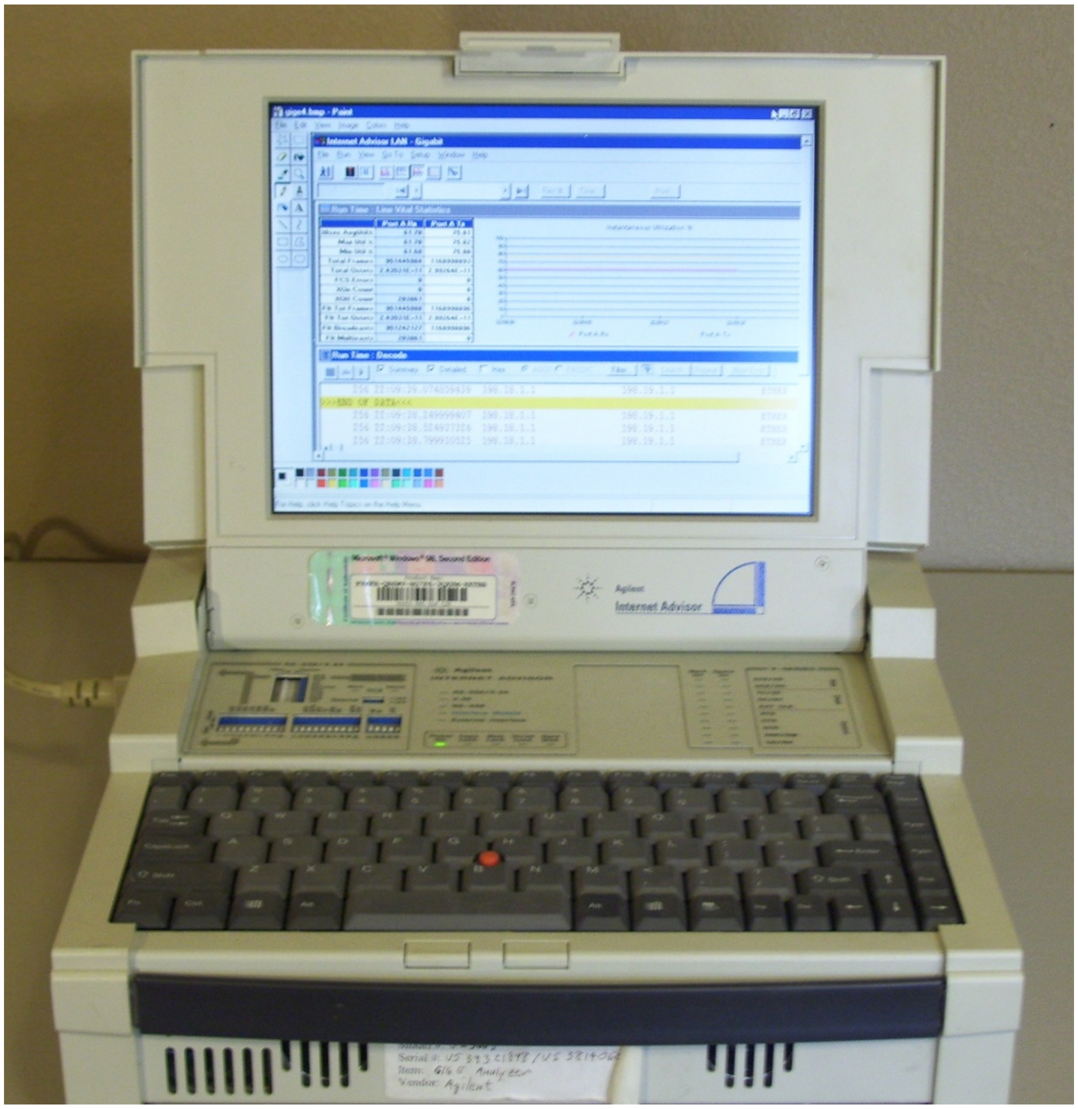
● Disable DHCP Server



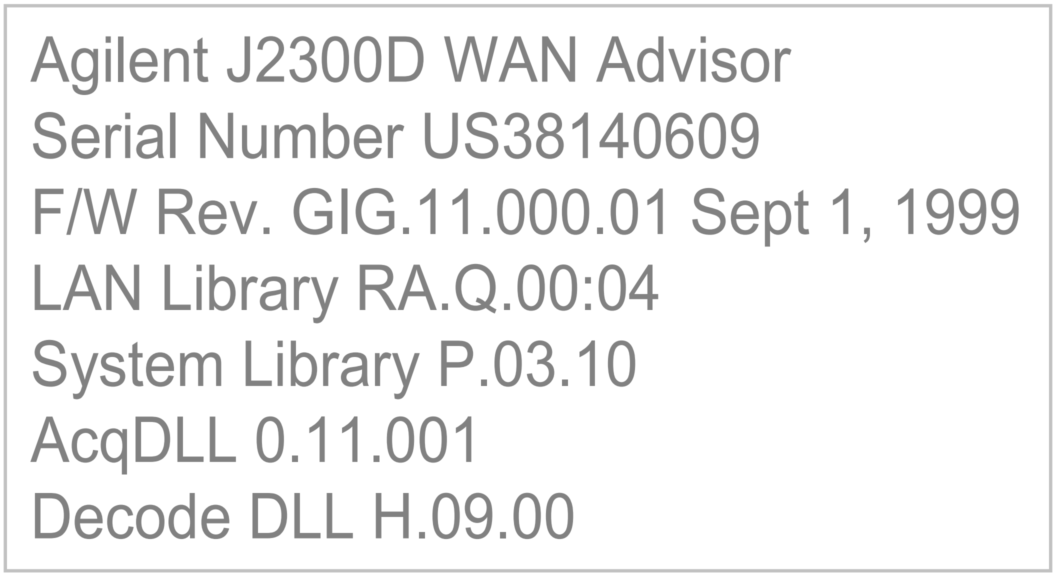
Group Name	Vlan Id	Enable/Disable	Remove	Edit	Interfaces	Enable/Disable
Default	1				LAN1	✓
Bridge	2	✓	☐	Edit	LAN2	✓
					LAN3	✓
					LAN4	✓

X-550 LAN Configuration and Port Mapping

Agilent J2300D WAN Advisor Test Set:

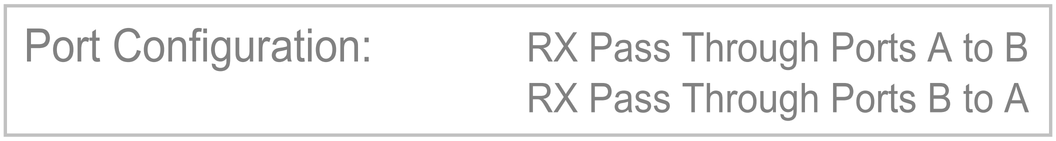


Agilent J2300D WAN Advisor Test Set



Agilent J2300D WAN Advisor
Serial Number US38140609
F/W Rev. GIG.11.000.01 Sept 1, 1999
LAN Library RA.Q.00:04
System Library P.03.10
AcqDLL 0.11.001
Decode DLL H.09.00

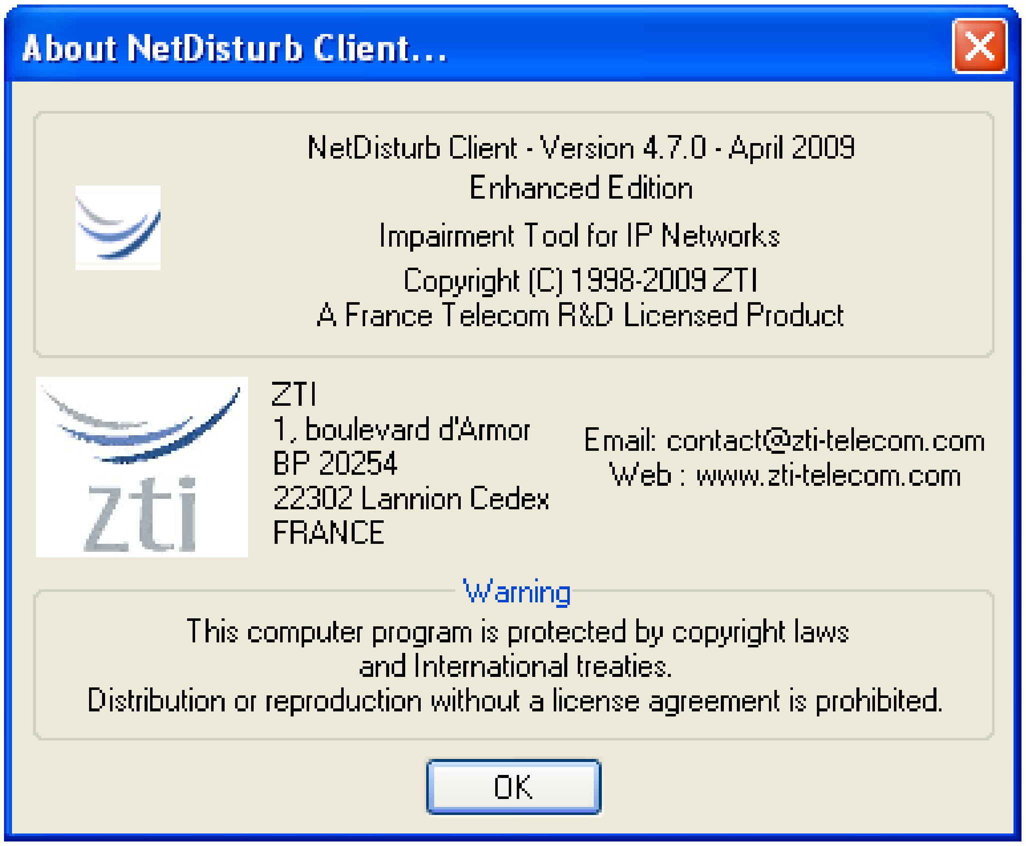
Agilent WAN Advisor General Information



```
Port Configuration:          RX Pass Through Ports A to B
                             RX Pass Through Ports B to A
```

Agilent WAN Advisor Port Configuration

Appendix B.2 NetDisturb Software Configuration



About NetDisturb Client...

NetDisturb Client - Version 4.7.0 - April 2009
Enhanced Edition

Impairment Tool for IP Networks

Copyright (C) 1998-2009 ZTI
A France Telecom R&D Licensed Product

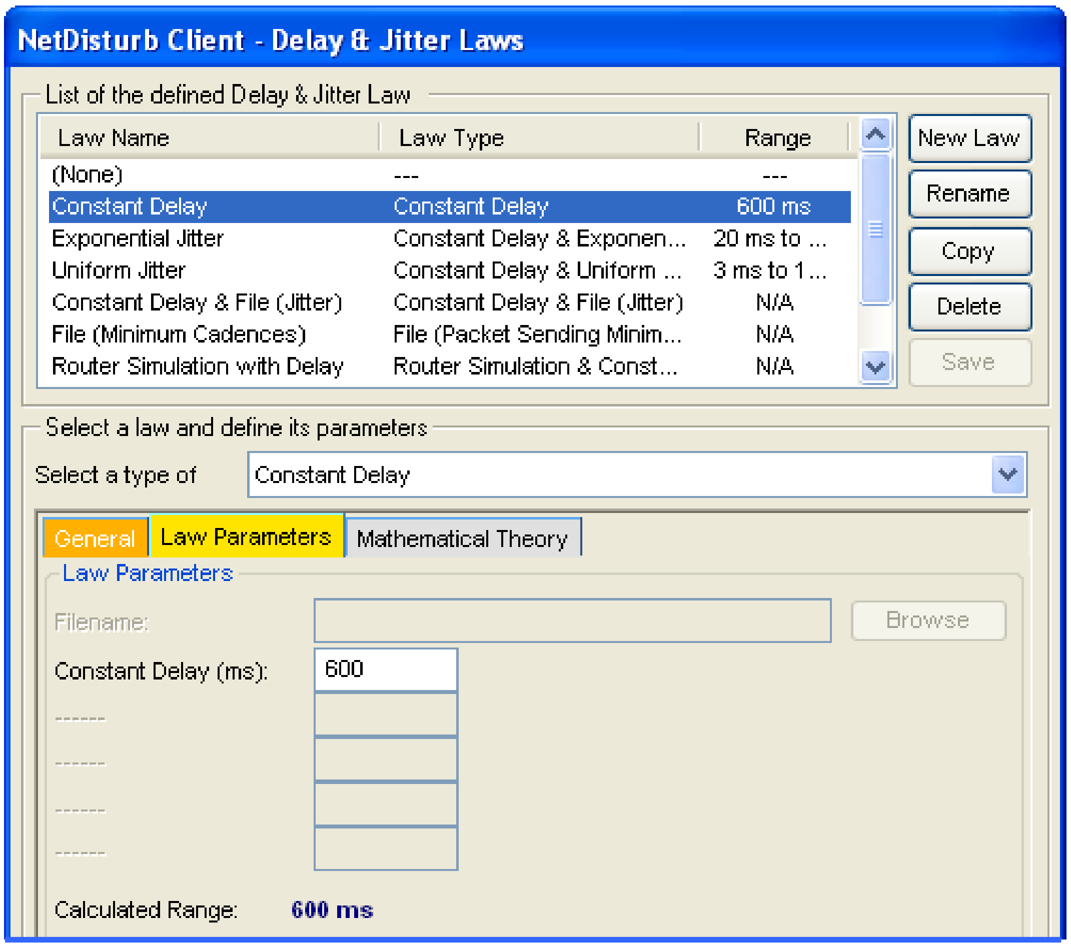
ZTI
1, boulevard d'Armor
BP 20254
22302 Lannion Cedex
FRANCE

Email: contact@zti-telecom.com
Web : www.zti-telecom.com

Warning

This computer program is protected by copyright laws
and International treaties.
Distribution or reproduction without a license agreement is prohibited.

NetDisturb Software General Information


NetDisturb Client - Delay & Jitter Laws
List of the defined Delay & Jitter Law
New Law
Rename
Copy
Delete
Save
Select a law and define its parameters
General
Law Parameters
Mathematical Theory
Browse

Law Name	Law Type	Range
(None)	---	---
Constant Delay	Constant Delay	600 ms
Exponential Jitter	Constant Delay & Exponen...	20 ms to ...
Uniform Jitter	Constant Delay & Uniform ...	3 ms to 1...
Constant Delay & File (Jitter)	Constant Delay & File (Jitter)	N/A
File (Minimum Cadences)	File (Packet Sending Minim...	N/A
Router Simulation with Delay	Router Simulation & Const...	N/A

Select a type of: Constant Delay

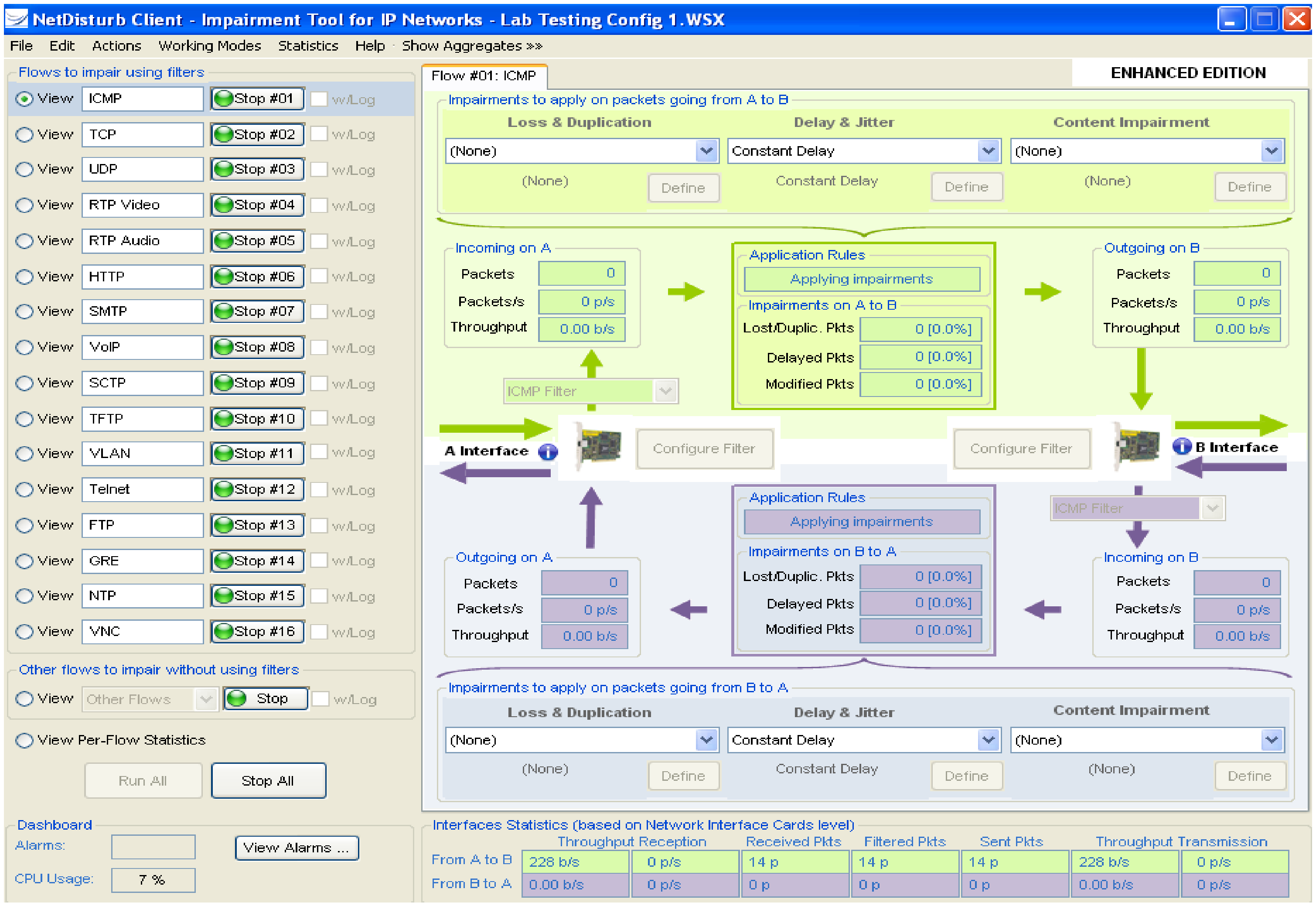
Law Parameters

Filename:

Constant Delay (ms): 600

Calculated Range: **600 ms**

NetDisturb 600 ms Constant Delay Configuration


NetDisturb Client - Impairment Tool for IP Networks - Lab Testing Config 1.WSX

File Edit Actions Working Modes Statistics Help Show Aggregates >>

Flows to impair using filters

[✓ View]	ICMP	Stop #01	w/Log
View	TCP	Stop #02	w/Log
View	UDP	Stop #03	w/Log
View	RTP Video	Stop #04	w/Log
View	RTP Audio	Stop #05	w/Log
View	HTTP	Stop #06	w/Log
View	SMTP	Stop #07	w/Log
View	VoIP	Stop #08	w/Log
View	SCTP	Stop #09	w/Log
View	TFTP	Stop #10	w/Log
View	VLAN	Stop #11	w/Log
View	Telnet	Stop #12	w/Log
View	FTP	Stop #13	w/Log
View	GRE	Stop #14	w/Log
View	NTP	Stop #15	w/Log
View	VNC	Stop #16	w/Log

Other flows to impair without using filters

View Other Flows Stop w/Log

View Per-Flow Statistics

Run All Stop All

Dashboard

Alarms: View Alarms ...

CPU Usage: 7 %

Flow #01: ICMP ENHANCED EDITION

Impairments to apply on packets going from A to B

Loss & Duplication	Delay & Jitter	Content Impairment
(None)	Constant Delay	(None)
(None)	Constant Delay	(None)

Incoming on A: Packets 0; Packets/s 0 p/s; Throughput 0.00 b/s

Application Rules: Applying impairments

Impairments on A to B: Lost/Duplic. Pkts 0 [0.0%]; Delayed Pkts 0 [0.0%]; Modified Pkts 0 [0.0%]

Outgoing on B: Packets 0; Packets/s 0 p/s; Throughput 0.00 b/s

ICMP Filter

A Interface Configure Filter Configure Filter B Interface

ICMP Filter

Application Rules: Applying impairments

Outgoing on A: Packets 0; Packets/s 0 p/s; Throughput 0.00 b/s

Impairments on B to A: Lost/Duplic. Pkts 0 [0.0%]; Delayed Pkts 0 [0.0%]; Modified Pkts 0 [0.0%]

Incoming on B: Packets 0; Packets/s 0 p/s; Throughput 0.00 b/s

Impairments to apply on packets going from B to A

Loss & Duplication	Delay & Jitter	Content Impairment
(None)	Constant Delay	(None)
(None)	Constant Delay	(None)

Interfaces Statistics (based on Network Interface Cards level)

	Throughput Reception		Received Pkts	Filtered Pkts	Sent Pkts	Throughput Transmission	
From A to B	228 b/s	0 p/s	14 p	14 p	14 p	228 b/s	0 p/s
From B to A	0.00 b/s	0 p/s	0 p	0 p	0 p	0.00 b/s	0 p/s

NetDisturb Packet Impairment Configuration

Appendix B.3 Results of Laboratory Trials

Functional Specifics for Lab Testing				
Configuration	**STB**	**SD or HD**	**Minimal or Full**	**With or Without Cache**
A	A125	SD	Minimal	Without Cache
B	A130	SD	Minimal	Without Cache
C	A130	HD	Minimal	Without Cache
D	MSTB	SD	Minimal	Without Cache
E	MSTB	HD	Minimal	Without Cache
F	A125	SD	Full	Without Cache
G	A130	SD	Full	Without Cache
H	A130	HD	Full	Without Cache
I	MSTB	SD	Full	Without Cache
J	MSTB	HD	Full	Without Cache
K	A125	SD	Full	With Cache
L	A130	SD	Full	With Cache
M	A130	HD	Full	With Cache
N	MSTB	SD	Full	With Cache
O	MSTB	HD	Full	With Cache

List of Lab Tests Performed

Each lab testing configuration (A through O) above consisted of 30 independent channel change trials. The results for each configuration and trial are found on the following pages. The summary results and analysis of these tests can be found in Chapter III.

Amino 125 STB to SD Using Minimal Config		
Trial #	**# Frames**	**Time (sec)**
1	35	1.40
2	37	1.48
3	36	1.44
4	35	1.40
5	29	1.16
6	34	1.36
7	34	1.36
8	38	1.52
9	35	1.40
10	32	1.28
11	35	1.40
12	38	1.52
13	36	1.44
14	32	1.28
15	35	1.40
16	36	1.44
17	35	1.40
18	36	1.44
19	32	1.28
20	36	1.44
21	33	1.32
22	36	1.44
23	33	1.32
24	38	1.52
25	34	1.36
26	38	1.52
27	36	1.44
28	32	1.28
29	38	1.52
30	36	1.44
Total	1050.00	42.00
Average	35.00	1.40
STDev	2.20	0.09
Minimum	29.00	1.16
Maximum	38.00	1.52

Lab Testing Results for Configuration A

Amino 130 STB to SD Using Minimal Config		
Trial #	**# Frames**	**Time (sec)**
1	56	2.24
2	54	2.16
3	50	2.00
4	53	2.12
5	48	1.92
6	48	1.92
7	48	1.92
8	49	1.96
9	56	2.24
10	49	1.96
11	55	2.20
12	49	1.96
13	52	2.08
14	50	2.00
15	51	2.04
16	53	2.12
17	52	2.08
18	48	1.92
19	48	1.92
20	51	2.04
21	56	2.24
22	51	2.04
23	53	2.12
24	51	2.04
25	54	2.16
26	47	1.88
27	53	2.12
28	56	2.24
29	54	2.16
30	53	2.12
Total	1548.00	61.92
Average	51.60	2.06
STDev	2.81	0.11
Minimum	47.00	1.88
Maximum	56.00	2.24

Lab Testing Results for Configuration B

Amino 130 STB to HD Using Minimal Config		
Trial #	**# Frames**	**Time (sec)**
1	54	2.16
2	54	2.16
3	55	2.20
4	53	2.12
5	53	2.12
6	48	1.92
7	57	2.28
8	48	1.92
9	50	2.00
10	53	2.12
11	56	2.24
12	59	2.36
13	58	2.32
14	56	2.24
15	50	2.00
16	48	1.92
17	49	1.96
18	56	2.24
19	57	2.28
20	56	2.24
21	56	2.24
22	48	1.92
23	53	2.12
24	49	1.96
25	56	2.24
26	50	2.00
27	54	2.16
28	54	2.16
29	53	2.12
30	51	2.04
Total	1594.00	63.76
Average	53.13	2.13
STDev	3.31	0.13
Minimum	48.00	1.92
Maximum	59.00	2.36

Lab Testing Results for Configuration C

Motorola STB to SD Using Minimal Config		
Trial #	**# Frames**	**Time (sec)**
1	48	1.92
2	42	1.68
3	49	1.96
4	48	1.92
5	42	1.68
6	46	1.84
7	42	1.68
8	47	1.88
9	50	2.00
10	45	1.80
11	48	1.92
12	43	1.72
13	48	1.92
14	42	1.68
15	45	1.80
16	44	1.76
17	48	1.92
18	42	1.68
19	45	1.80
20	42	1.68
21	48	1.92
22	42	1.68
23	48	1.92
24	48	1.92
25	45	1.80
26	48	1.92
27	45	1.80
28	45	1.80
29	45	1.80
30	49	1.96
Total	1369	54.76
Mean	45.63	1.83
STDev	2.62	0.10
Minimum	42	1.68
Maximum	50	2.00

Lab Testing Results for Configuration D

Motorola STB to HD Using Minimal Config		
Trial #	**# Frames**	**Time (sec)**
1	49	1.96
2	48	1.92
3	48	1.92
4	45	1.80
5	51	2.04
6	47	1.88
7	51	2.04
8	48	1.92
9	46	1.84
10	43	1.72
11	50	2.00
12	48	1.92
13	45	1.80
14	43	1.72
15	51	2.04
16	47	1.88
17	48	1.92
18	51	2.04
19	45	1.80
20	48	1.92
21	52	2.08
22	43	1.72
23	51	2.04
24	48	1.92
25	50	2.00
26	45	1.80
27	48	1.92
28	50	2.00
29	49	1.96
30	53	2.12
Total	1441.00	57.64
Average	48.03	1.92
STDev	2.74	0.11
Minimum	43.00	1.72
Maximum	53.00	2.12

Lab Testing Results for Configuration E

Amino 125 STB to SD via Network Without Cache		
Trial #	**# Frames**	**Time (sec)**
1	69	2.76
2	69	2.76
3	66	2.64
4	64	2.56
5	69	2.76
6	66	2.64
7	67	2.68
8	72	2.88
9	69	2.76
10	74	2.96
11	75	3.00
12	75	3.00
13	70	2.80
14	76	3.04
15	69	2.76
16	67	2.68
17	64	2.56
18	66	2.64
19	70	2.80
20	63	2.52
21	72	2.88
22	65	2.60
23	69	2.76
24	73	2.92
25	69	2.76
26	69	2.76
27	66	2.64
28	67	2.68
29	72	2.88
30	67	2.68
Total	2069.00	82.76
Average	68.97	2.76
STDev	3.45	0.14
Minimum	63.00	2.52
Maximum	76.00	3.04

Lab Testing Results for Configuration F

Amino 130 STB to SD via Network Without Cache		
Trial #	**# Frames**	**Time (sec)**
1	87	3.48
2	81	3.24
3	84	3.36
4	88	3.52
5	84	3.36
6	81	3.24
7	86	3.44
8	86	3.44
9	78	3.12
10	81	3.24
11	81	3.24
12	76	3.04
13	84	3.36
14	87	3.48
15	81	3.24
16	75	3.00
17	87	3.48
18	81	3.24
19	89	3.56
20	84	3.36
21	83	3.32
22	84	3.36
23	85	3.40
24	84	3.36
25	78	3.12
26	84	3.36
27	81	3.24
28	84	3.36
29	87	3.48
30	78	3.12
Total	2489.00	99.56
Average	82.97	3.32
STDev	3.58	0.14
Minimum	75.00	3.00
Maximum	89.00	3.56

Lab Testing Results for Configuration G

Amino 130 STB to HD via Network Without Cache		
Trial #	**# Frames**	**Time (sec)**
1	87	3.48
2	87	3.48
3	90	3.60
4	90	3.60
5	84	3.36
6	87	3.48
7	84	3.36
8	79	3.16
9	84	3.36
10	81	3.24
11	87	3.48
12	81	3.24
13	81	3.24
14	87	3.48
15	87	3.48
16	87	3.48
17	81	3.24
18	87	3.48
19	81	3.24
20	82	3.28
21	87	3.48
22	87	3.48
23	84	3.36
24	85	3.40
25	85	3.40
26	84	3.36
27	93	3.72
28	86	3.44
29	87	3.48
30	79	3.16
Total	2551.00	102.04
Average	85.03	3.40
STDev	3.35	0.13
Minimum	79.00	3.16
Maximum	93.00	3.72

Lab Testing Results for Configuration H

Motorola STB to SD via Network Without Cache		
Trial #	**# Frames**	**Time (sec)**
1	75	3.00
2	78	3.12
3	82	3.28
4	75	3.00
5	75	3.00
6	78	3.12
7	81	3.24
8	76	3.04
9	77	3.08
10	75	3.00
11	78	3.12
12	75	3.00
13	81	3.24
14	72	2.88
15	78	3.12
16	75	3.00
17	78	3.12
18	78	3.12
19	81	3.24
20	75	3.00
21	78	3.12
22	75	3.00
23	75	3.00
24	78	3.12
25	73	2.92
26	78	3.12
27	78	3.12
28	78	3.12
29	75	3.00
30	75	3.00
Total	2306.00	92.24
Average	76.87	3.07
STDev	2.42	0.10
Minimum	72.00	2.88
Maximum	82.00	3.28

Lab Testing Results for Configuration I

Motorola STB to HD via Network Without Cache		
Trial #	**# Frames**	**Time (sec)**
1	76	3.04
2	78	3.12
3	75	3.00
4	78	3.12
5	81	3.24
6	78	3.12
7	73	2.92
8	81	3.24
9	81	3.24
10	81	3.24
11	81	3.24
12	75	3.00
13	75	3.00
14	75	3.00
15	75	3.00
16	84	3.36
17	81	3.24
18	81	3.24
19	77	3.08
20	75	3.00
21	78	3.12
22	73	2.92
23	84	3.36
24	78	3.12
25	78	3.12
26	81	3.24
27	77	3.08
28	75	3.00
29	81	3.24
30	78	3.12
Total	2344.00	93.76
Average	78.13	3.13
STDev	3.06	0.12
Minimum	73.00	2.92
Maximum	84.00	3.36

Lab Testing Results for Configuration J

Amino 125 STB to SD via Network With Cache		
Trial #	**# Frames**	**Time (sec)**
1	40	1.60
2	42	1.68
3	37	1.48
4	36	1.44
5	45	1.80
6	40	1.60
7	39	1.56
8	33	1.32
9	37	1.48
10	36	1.44
11	43	1.72
12	34	1.36
13	43	1.72
14	37	1.48
15	37	1.48
16	34	1.36
17	40	1.60
18	43	1.72
19	42	1.68
20	36	1.44
21	43	1.72
22	33	1.32
23	39	1.56
24	34	1.36
25	42	1.68
26	34	1.36
27	42	1.68
28	36	1.44
29	40	1.60
30	34	1.36
Total	1151.00	46.04
Average	38.37	1.53
STDev	3.61	0.14
Minimum	33.00	1.32
Maximum	45.00	1.80

Lab Testing Results for Configuration K

Amino 130 STB to SD via Network With Cache		
Trial #	**# Frames**	**Time (sec)**
1	57	2.28
2	51	2.04
3	54	2.16
4	57	2.28
5	54	2.16
6	54	2.16
7	51	2.04
8	51	2.04
9	54	2.16
10	51	2.04
11	48	1.92
12	54	2.16
13	57	2.28
14	51	2.04
15	54	2.16
16	50	2.00
17	54	2.16
18	51	2.04
19	48	1.92
20	54	2.16
21	57	2.28
22	55	2.20
23	54	2.16
24	57	2.28
25	57	2.28
26	57	2.28
27	54	2.16
28	54	2.16
29	51	2.04
30	48	1.92
Total	1599.00	63.96
Average	53.30	2.13
STDev	2.85	0.11
Minimum	48.00	1.92
Maximum	57.00	2.28

Lab Testing Results for Configuration L

Amino 130 STB to HD via Network With Cache		
Trial #	**# Frames**	**Time (sec)**
1	56	2.24
2	51	2.04
3	49	1.96
4	51	2.04
5	60	2.40
6	55	2.20
7	54	2.16
8	54	2.16
9	57	2.28
10	54	2.16
11	51	2.04
12	57	2.28
13	54	2.16
14	51	2.04
15	51	2.04
16	57	2.28
17	51	2.04
18	57	2.28
19	51	2.04
20	61	2.44
21	51	2.04
22	51	2.04
23	60	2.40
24	57	2.28
25	54	2.16
26	54	2.16
27	54	2.16
28	52	2.08
29	54	2.16
30	57	2.28
Total	1626.00	65.04
Average	54.20	2.17
STDev	3.16	0.13
Minimum	49.00	1.96
Maximum	61.00	2.44

Lab Testing Results for Configuration M

Motorola STB to SD via Network With Cache		
Trial #	**# Frames**	**Time (sec)**
1	51	2.04
2	48	1.92
3	45	1.80
4	45	1.80
5	45	1.80
6	48	1.92
7	45	1.80
8	45	1.80
9	48	1.92
10	45	1.80
11	51	2.04
12	43	1.72
13	45	1.80
14	48	1.92
15	51	2.04
16	48	1.92
17	44	1.76
18	48	1.92
19	46	1.84
20	47	1.88
21	45	1.80
22	45	1.80
23	53	2.12
24	45	1.80
25	45	1.80
26	44	1.76
27	45	1.80
28	45	1.80
29	45	1.80
30	44	1.76
Total	1392.00	55.68
Average	46.40	1.86
STDev	2.49	0.10
Minimum	43.00	1.72
Maximum	53.00	2.12

Lab Testing Results for Configuration N

Motorola STB to HD via Network With Cache		
Trial #	**# Frames**	**Time (sec)**
1	54	2.16
2	48	1.92
3	45	1.80
4	51	2.04
5	44	1.76
6	51	2.04
7	48	1.92
8	48	1.92
9	45	1.80
10	48	1.92
11	51	2.04
12	51	2.04
13	48	1.92
14	51	2.04
15	48	1.92
16	45	1.80
17	51	2.04
18	51	2.04
19	48	1.92
20	48	1.92
21	51	2.04
22	45	1.80
23	51	2.04
24	54	2.16
25	48	1.92
26	45	1.80
27	55	2.20
28	44	1.76
29	48	1.92
30	48	1.92
Total	1463.00	58.52
Average	48.77	1.95
STDev	3.00	0.12
Minimum	44.00	1.76
Maximum	55.00	2.20

Lab Testing Results for Configuration O

APPENDIX C

Appendix C.1 Shared Routines for Channel Change Prediction Models

Libraries and Standard Declarations:

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <cstdio>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

using std::cout;
using std::cin;
using std::endl;
using namespace std;
```

Variable Definitions:

```
// Variables used for file Input/Output and to exit program
char input_file [1] [MAX3];
char output_file [1] [MAX3];
char Quit1, Quit2;

// Variables used to bring in Input Data Set and get them in properly formatted arrays
const int MAX2=8;
const int MAX3=15;
char chan_t [150] [MAX2];
char prob_t [150] [MAX2];
char chan_n [1] [MAX2];
char bw_t [150] [MAX2];
int chan_in [150], bw_in [150];
float prob_in [150];

// Variables used for Channel List Arrays
int Qchan_ls [150];             // For the List of Channels
int Qbw_ls [150];               // For the Bandwidths of Channels
float Qprob_ls [150];           // For Time-Weighted Probabilities of Channels
```

```
// Variables used for Confirmed List Arrays
int Qchan_cf [150];             // Channel #s in Confirmed List
int Qbw_cf [150];               // Channel Bandwidths in Confirmed List
float Qprob_cf [150];           // Channel Probabilities in Confirmed List
int Conf_chan_count;            // Count of Channels in Confirmed List

// Variables used for the current channel information and Penalty_Factor
int Curr_chan_number, Curr_chan_bw;
float Curr_chan_prob [1];
double Penalty_Factor[1];

// Variable to track BW status of system and arrays
int BW_avail, BW_total, Rem_BW_avail [1];

// Variables used during sorting processes
int Qchan_sort [150], Qbw_sort [150], Qsortc_calc[1];
float Qprob_sort [150], Qdivided_sort [150], Qsort_calc [1];
float Qlastd_sort[1], Qsortd_calc[1], Qdivided_ls [150];

// Variables used to hold attributes about the Confirmed Channel Arrays
int Curr_conf_BW [1], test_cf_bw [1];
float Net_Confirmed_Probm [1];

// Variables used to track current best calculation results
int Best_bw [1], Final_Best_bw [1], Qchan_best [150], Qbw_best [150];
float Best_prob [1], Best_Prob_Hold1 [1], Final_Best_prob [1];
float Qprob_best [150];

// Variables used to temporarily hold data during various conversions and comparisons
int Qbw_temp [1], Qchan_temp2 [150], Qbw_temp2 [150];
float Qsort_temp[1], Qprob_temp [1];
float Qprob_temp2 [150], Qdivided_temp2 [150];
double Qtempbw[1];

// Variables used as counters by the Shared Routines and Prediction Models
int i, j, k, m, n, p, q, r, s, t, u, v, w, x, y, z, cc, BW_sum_init [1], BW_sumt, Chan_max;
```

Shared Routines:

void **LS_Sort()**
// This routine sorts the Channel List Arrays based on the sort terms calculated by the specific prediction model. The sort terms must be calculated prior to this routine and populated in the "Qdivided_ls" array. The routine re-orders the Channel List Arrays by moving the channel number, bandwidth, and probability in descending order for each successive sort term.

```
{	for(j=0; j<150; j++)	{
		Qchan_sort[j] = 0;
		Qprob_sort[j] = 0;
		Qbw_sort[j] = 0;
		Qdivided_sort[j] = 0;
		Qchan_temp2[j] = Qchan_ls[j];
		Qprob_temp2[j] = Qprob_ls[j];
		Qbw_temp2[j] = Qbw_ls[j];
		Qdivided_temp2[j] = Qdivided_ls[j];
	}
	for(k=0; k<150; k++)  {
		Qchan_ls[k] = 0;
		Qprob_ls[k] = 0;
		Qbw_ls[k] = 0;
		Qdivided_ls[k] = 0;
	}
	Qsortd_calc[0] = 0;
	for(q=0; q<150; q++)  {
		for(s=0; s<150; s++)		{
			if (Qdivided_temp2[s] > Qsortd_calc[0])		{
				Qsortd_calc[0] = Qdivided_temp2[s];
				Qsortc_calc[0] = Qchan_temp2[s];
			}
		}
		Qchan_sort[q] = Qsortc_calc[0];
		for(r=0; r<150; r++)	{
			if(Qchan_sort[q] == Qchan_temp2[r])		{
				Qprob_sort[q] = Qprob_temp2[r];
				Qbw_sort[q] = Qbw_temp2[r];
				Qdivided_sort[q] = Qdivided_temp2[r];
				Qchan_temp2[r] = 0;
				Qprob_temp2[r] = 0;
				Qbw_temp2[r] = 0;
				Qdivided_temp2[r] = 0;
			}
		}
		Qsortd_calc[0] = 0;
	}
	for(t=0; t<150; t++)	{
		Qchan_ls[t] = Qchan_sort[t];
		Qprob_ls[t] = Qprob_sort[t];
		Qbw_ls[t] = Qbw_sort[t];
		Qdivided_ls[t] = Qdivided_sort[t];
	}
}
```

```
void LS_Strip_Chan_0s()
// Routine strips all channels that are outside limits or have null values.
{       for(r=0; r<150; r++)    {
                if(Qchan_ls[r] > 150)  {
                        Qchan_ls[r] = 0;
                }
        }
        w=0;
        while(w<150)  {
                for(m=0; m<150; m++)            {
                        if (Qchan_ls[m] == 0) {
                                p=1;
                                for(n=m; n < 150; n++)          {
                                        if(Qchan_ls[n+p] == 0) {
                                                p++;
                                        }
                                        else {
                                                Qchan_ls[n] = Qchan_ls[n+p];
                                                Qbw_ls[n] = Qbw_ls[n+p];
                                                Qprob_ls[n] = Qprob_ls[n+p];

                                        }
                                }
                        }
                }
                w++;
        }
        for(m=0; m<150; m++)            {
                if (Qchan_ls[m] == 0 || Qchan_ls[m] > 150)  {
                        Qchan_ls[m] = 0;
                        Qbw_ls[m] = 0;
                        Qprob_ls[m] = 0;
                }
        }
}

void LS_Clean_BW_and_Prob()
// Cleans up BW and Probability values outside limits following other manipulations.
{       for(m=0; m<150; m++)            {
                if (Qchan_ls[m] <= 0 || Qchan_ls[m] > 150)  {
                        Qchan_ls[m] = 0;
                        Qbw_ls[m] = 0;
                        Qprob_ls[m] = 0;
                        Qdivided_ls[m] = 0;
                }
        }
```

```
          for(m=0; m<150; m++)          {
if (Qbw_ls[m]!=2 && Qbw_ls[m]!=4 && Qbw_ls[m]!=9 && Qbw_ls[m]!=18)    {
                        Qchan_ls[m] = 0;
                        Qbw_ls[m] = 0;
                        Qprob_ls[m] = 0;
                        Qdivided_ls[m] = 0;
                }
          }
          for(m=0; m<150; m++)          {
                if (Qprob_ls[m]<=0 || Qprob_ls[m] >= 1)      {
                        Qchan_ls[m] = 0;
                        Qbw_ls[m] = 0;
                        Qprob_ls[m] = 0;
                        Qdivided_ls[m] = 0;
                }
          }
          for(m=0; m<150; m++)          {
                if (Qchan_ls[m] == 0) {
                        p=1;
                        for(n=m; n<150; n++)   {
                                if (Qchan_ls[n+p] != 0)   {
                                        Qchan_ls[n] = Qchan_ls[n+p];
                                        Qbw_ls[n] = Qbw_ls[n+p];
                                        Qprob_ls[n] = Qprob_ls[n+p];
                                        Qdivided_ls[n] = Qdivided_ls[n+p];
                                }
                                else   {
                                        p++;
                                }
                        }
                }
          }
}
```

void **Confirmed_Count()**

```
// Verifies the confirmed count of cached channels.
{         Conf_chan_count = 0;
          for(m=0; m<150; m++)          {
                if (Qprob_cf[m] == 0) {
                        Qchan_cf[m] = 0;
                }
          }
          for(i=0; i<150; i++)    {
                if (Qchan_cf[i] != 0)   {
                        Conf_chan_count = Conf_chan_count + 1;
                }
          }
}

void Confirmed_BW_and_Prob_Count()
// Verifies the confirmed BW and Probability of all cached channels.
{         Curr_conf_BW[0] = 0;
          Final_Best_bw[0]=0;
          Final_Best_prob[0]=0;
          for(j=0; j<150; j++)    {
                    if (Qchan_cf[j] != 0 && Qchan_cf[j] < 151) {
                              Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_cf[j];
                              Final_Best_bw[0] = Final_Best_bw[0] + Qbw_cf[j];
                              Final_Best_prob[0] = Final_Best_prob[0] + Qprob_cf[j];
                    }
          }
          Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
          cout<<"Complete"<<endl;
}

void LS_to_Confirmed()
// Moves channels from sorted list to confirmed arrays for caching.
{         j=0;
          Curr_conf_BW[0]=0;
          Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
          for(i=0; i<150; i++)    {
                    if (Qbw_ls[i] <= Rem_BW_avail[0])            {
                              Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
                              Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
                              Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
                              Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
                              j++;
                    }
          }
          Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
}

void LS_Check_Confirmed()
// Routine to check the validity of the confirmed cache channels.
{         for(j=0; j<150; j++)    {
                    for(k=0; k<150; k++)  {
                              if (Qchan_ls[j] == Qchan_cf[k])        {
                                        Qchan_ls[j] = 0;
                                        k=150;
                              }
                    }
```

```
		}
}
```

Typical Usage and Order of Shared Routines:

```
// Initially open Input Data Set file and read in data
// Query and track Total Bandwidth available for video
// Determine and track details about current channel
// Calculate Bandwidth Available for caching
// Populate Channel List Arrays from Input Data Set details
// Calculate new sort term for all channels based on the specific prediction model used
```

// The shared routines are typically run in a dual pass order. The first pass sorts the Channel List Arrays and then verifies the integrity and details about the Channel List and Confirmed Channel Arrays. The second pass populates the cached channels into the Confirmed Channels Arrays, cleans up the Channel List Arrays for all channels confirmed, and calculates key metrics about the cached channels.

```
// 1st Pass Run Prior to Sorting Algorithm Routine

LS_Sort();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Confirmed_Count();
Confirmed_BW_and_Prob_Count();

// 2nd Pass Run After Sorting Algorithm Routine

LS_to_Confirmed();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Confirmed_Count();
Confirmed_BW_and_Prob_Count();

// Verify all Bandwidth Available for caching fully utilized

// Output Final Results and Exit Program
```

Appendix C.2 Input, Initialization, Output, and Related Routines

<u>Typical Input Section Routine and Additional Subroutines</u>:

```
// Libraries, Standard Declarations, and Variables per Appendix C
int main(void) {
// Read input data from entered file into arrays
cout << "Enter the Name of the Input File: " <<'\t'<<'\t';
cin >> input_file[0];
cout << endl;
cout << "Enter the Name of the Output File: " <<'\t'<<'\t';
cin >> output_file[0];
cout << endl;
 ifstream infile;
infile.open(input_file[0]);
if(!infile)                {
        cout<<"unable to open infile";
        exit(1);
        }
ofstream outfile;
outfile.open(output_file[0]);
if(!outfile)       {
        cout<<"unable to open outfile";
        exit(1);
        }
k=0;
for(j=0; j<450; j++)    {
        infile.getline(chan_t[k],MAX2);
        infile.getline(prob_t[k],MAX2);
        infile.getline(bw_t[k],MAX2);
        k=k+1;
        j=j+2;
}
infile.getline(chan_n[0], MAX2);
for(m=0; m<150; m++)  {
        chan_in[m] = atoi (chan_t[m]);
        prob_in[m] = atof (prob_t[m]);
        bw_in[m] = atoi (bw_t[m]);
}
Curr_chan_number = atoi (chan_n[0]);
for(n=0; n<150; n++)  {
        if (chan_in[n] == Curr_chan_number)            {
                Curr_chan_bw = bw_in[n];
                Curr_chan_prob[0] = prob_in[n];
                n = 150;
        }
}
// Read entered BW Total into variable for use
cout << "Enter the Amount of Total Bandwidth: "<<'\t'<<'\t';
cin >> BW_total;
BW_avail = BW_total - Curr_chan_bw;
BW_sumt = 0;
BW_sum_init[0] = 0;
while (BW_sumt <= BW_avail)  {
        for(n=0; n<150; n++)  {
                if (BW_sum_init[0] + bw_in[n] <= BW_avail) {
                        BW_sum_init[0] = BW_sum_init[0] + bw_in[n];
                        Chan_max = n+1;
                }
                else    {
                        BW_sumt = BW_avail + 1;
                }
        }
}

// Populates Q List, places initial confirmed entries to Q Confirmed, and deletes those
from the remaining list

for(m=0; m<150; m++)  {
        Qchan_ls[m] = chan_in[m];
        Qprob_ls[m] = prob_in[m];
        Qbw_ls[m] = bw_in[m];
}
for(n=0; n<150; n++)  {
        if (Qchan_ls[n] == Curr_chan_number)        {
                for(p=n; p<150; p++)  {
                        Qchan_ls[p] = Qchan_ls[p+1];
                        Qprob_ls[p] = Qprob_ls[p+1];
                        Qbw_ls[p] = Qbw_ls[p+1];
                }
                n = n-1;
        }
}
```

Typical Output Section Routine and Additional Subroutines:

```
outfile <<"MODEL OUTPUT"<<endl;
outfile <<endl<<"Input Filename = "<<'\t'<<'\t'<<'\t'<<input_file[0];
outfile <<endl<<"Total Bandwidth = "<<'\t'<<'\t'<<'\t'<<BW_total<<endl;
outfile <<endl<<endl;
outfile <<"Confirmed Channels List for Pre-Selected Queue:";
outfile <<endl<<endl;
outfile <<"Chan"<<'\t'<<"Prob"<<'\t'<<"BW"<< endl;
for (i=0; i<150; i++)   {
        if (Qchan_cf[i] != 0)   {
                outfile <<Qchan_cf[i]<<'\t'<<Qprob_cf[i]<<'\t'<<Qbw_cf[i]<< endl;
        }
}
outfile <<endl<<endl;
outfile <<"Current Channel Number = "<<'\t'<<'\t'<<Curr_chan_number<<endl;
outfile <<"Current Channel Probability = "<<'\t'<<'\t'<<Curr_chan_prob[0]<<endl;
outfile <<"Current Channel Bandwidth = "<<'\t'<<'\t'<<Curr_chan_bw<<endl<<endl;
outfile<< "Channel List Probability Mean = "<<'\t'<<Mean_pr[0]<<endl;
outfile<< "Confirmed List Bandwidth = "<<'\t'<<'\t'<<Final_Best_bw[0]<<endl;
outfile<< "Remaining Bandwidth Available = "<<'\t'<<Rem_BW_avail[0]<<endl<<endl;
outfile<< "Gross Confirmed List Probability = "<<'\t'<<Final_Best_prob[0]<<endl;
outfile<< "Net Confirmed List Probability = "<<'\t'<<Net_Confirmed_Prob[0]<<endl;
infile.close();
outfile.close();
return 0;
}
```

APPENDIX D

Appendix D.1 Sample Moment Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables

int main(void) {
{       Mean_pr[0] = 0;
        for(p=0; p<150; p++)  {
                Mean_pr[0] = Mean_pr[0] + Qprob_ls[p];
        }
        Mean_pr[0] = (Mean_pr[0] / 149);
}
Confirmed_Count();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
{       Penalty_Factor[0] = 1.19;
        for(p=0; p<150; p++)  {
                Qtempbw[0] = Qbw_ls[p];
                BWfactor[0] = pow( Qtempbw[0], Penalty_Factor[0] );
                                // Sample_Moments Sort Field
                Qdivided_ls[p] = ((Qprob_ls[p] - Mean_pr[0]) / BWfactor[0]);
                                // Avoid negatives to reuse existing Shared Routines
                Qdivided_ls[p] = (Qdivided_ls[p] + 1);
        }
}
{       for(j=0; j<150; j++)    {
                Qchan_sort[j] = 0;
                Qprob_sort[j] = 0;
                Qbw_sort[j] = 0;
                Qdivided_sort[j] = 0;
                Qchan_temp2[j] = Qchan_ls[j];
                Qprob_temp2[j] = Qprob_ls[j];
                Qbw_temp2[j] = Qbw_ls[j];
                Qdivided_temp2[j] = Qdivided_ls[j];
        }
        for(k=0; k<150; k++)  {
                Qchan_ls[k] = 0;
                Qprob_ls[k] = 0;
                Qbw_ls[k] = 0;
                Qdivided_ls[k] = 0;
        }
        Qsortd_calc[0] = 0;
        for(q=0; q<150; q++)  {
                for(s=0; s<150; s++)            {
                        if (Qdivided_temp2[s] > Qsortd_calc[0])                 {
                                Qsortd_calc[0] = Qdivided_temp2[s];
                                Qsortc_calc[0] = Qchan_temp2[s];
                        }
                }
                Qchan_sort[q] = Qsortc_calc[0];
                for(r=0; r<150; r++)    {
                        if(Qchan_sort[q] == Qchan_temp2[r])         {
                                Qprob_sort[q] = Qprob_temp2[r];
                                Qbw_sort[q] = Qbw_temp2[r];
                                Qdivided_sort[q] = Qdivided_temp2[r];
                                Qchan_temp2[r] = 0;
                                Qprob_temp2[r] = 0;
                                Qbw_temp2[r] = 0;
                                Qdivided_temp2[r] = 0;
                        }
                }
                Qsortd_calc[0] = 0;
        }
        for(t=0; t<150; t++)    {
                Qchan_ls[t] = Qchan_sort[t];
                Qprob_ls[t] = Qprob_sort[t];
                Qbw_ls[t] = Qbw_sort[t];
                Qdivided_ls[t] = Qdivided_sort[t];
        }
}
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
LS_to_Confirmed();
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];
Net_Confirmed_Prob[0] = ((Final_Best_prob[0])/(1 - Curr_chan_prob[0]));

// Output Final Results
}
```

Appendix D.2 Probability Divided by BW Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables as other models

int main(void) {
Confirmed_Count();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
{    for(p=0; p<150; p++)  {
          Qdivided_ls[p] = (Qprob_ls[p] / Qbw_ls[p]);
     }
}
LS_Sort_by_Divided();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
LS_to_Confirmed();
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

// Output Final Results as with other models
}
```

Appendix D.3 Probability Only Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables as other models

int main(void) {
Confirmed_Count();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
LS_to_Confirmed();
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

// Output Final Results
}
```

Appendix D.4 Maximum Channels Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables as other models

int main(void) {
Confirmed_Count();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
```

```
Confirmed_Count();
j=0;
Curr_conf_BW[0]=0;
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];

          // Starts caching channels based on lowest BW
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 2 && Qbw_ls[i] <= Rem_BW_avail[0])                {
          Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
          Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
          Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
          Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
          Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
          j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 4 && Qbw_ls[i] <= Rem_BW_avail[0])                {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 9 && Qbw_ls[i] <= Rem_BW_avail[0])                {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 18 && Qbw_ls[i] <= Rem_BW_avail[0])               {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

// Output Final Results
}
```

Appendix D.5 HD Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables as other models

int main(void) {
Confirmed_Count();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Confirmed_Count();

         // Starts caching channels based on highest BW
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 18 && Qbw_ls[i] <= Rem_BW_avail[0])              {
         Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
         Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
         Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
         Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
         Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
         j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                              {
    if (Qbw_ls[i] == 9 && Qbw_ls[i] <= Rem_BW_avail[0])               {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                          {
    if (Qbw_ls[i] == 4 && Qbw_ls[i] <= Rem_BW_avail[0])            {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
for(i=0; i<150; i++)                                          {
    if (Qbw_ls[i] == 2 && Qbw_ls[i] <= Rem_BW_avail[0])            {
        Qchan_cf[Conf_chan_count + j] = Qchan_ls[i];
        Qprob_cf[Conf_chan_count + j] = Qprob_ls[i];
        Qbw_cf[Conf_chan_count + j] = Qbw_ls[i];
        Curr_conf_BW[0] = Curr_conf_BW[0] + Qbw_ls[i];
        Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
        j++;
    }
}
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

// Output Final Results
}
```

Appendix D.6 Exhaustive Search Model Sorting Routine

```
// Input data from file, populate Q arrays, & initialize variables as other models

// Subroutine to determine best 6 available channels per pass
void Six_Iterations()
                        // First Iteration
{       Best_Prob_Hold1[0] = Best_prob[0];
        Best_prob[0] = 0;
        for(j=0; j<Total_Iterations; j++)       {
                Qbw_temp[0] = Qbw_ls[j];
                if (Qbw_temp[0] <= Rem_BW_avail[0])      {
                        Qprob_temp[0] = Qprob_ls[j];
                        if (Qprob_temp[0] > Best_prob[0])   {
                                Qchan_best[0] = Qchan_ls[j];
```

```
                              Best_prob[0] = Qprob_temp[0];
                              Best_bw[0] = Qbw_ls[j];
                         }
                    }
               }
                              // Second Iteration
               for(j=0; j<Total_Iterations; j++)        {
                    for(k=j+1; k<Total_Iterations; k++)   {
                         Qbw_temp[0] = Qbw_ls[j] + Qbw_ls[k];
                         if (Qbw_temp[0] <= Rem_BW_avail[0])      {
                              Qprob_temp[0] = Qprob_ls[j] + Qprob_ls[k];
                              if (Qprob_temp[0] > Best_prob[0])    {
                                   Qchan_best[0] = Qchan_ls[j];
                                   Qchan_best[1] = Qchan_ls[k];
                                   Best_prob[0] = Qprob_temp[0];
                                   Best_bw[0] = Qbw_temp[0];
                              }
                         }
                    }
               }
                              // Third Iteration
          for(j=0; j<Total_Iterations; j++)        {
               for(k=j+1; k<Total_Iterations; k++)   {
                    for(m=k+1; m<Total_Iterations; m++)          {
                         Qbw_temp[0] = Qbw_ls[j] + Qbw_ls[k] + Qbw_ls[m];
                         if (Qbw_temp[0] <= Rem_BW_avail[0])      {
                             Qprob_temp[0] = Qprob_ls[j] + Qprob_ls[k] +Qprob_ls[m];
                                   if (Qprob_temp[0] > Best_prob[0])    {
                                        Qchan_best[0] = Qchan_ls[j];
                                        Qchan_best[1] = Qchan_ls[k];
                                        Qchan_best[2] = Qchan_ls[m];
                                        Best_prob[0] = Qprob_temp[0];
                                        Best_bw[0] = Qbw_temp[0];
                                   }
                              }
                         }
                    }
               }
                              // Fourth Iteration
          for(j=0; j<Total_Iterations; j++)        {
            for(k=j+1; k<Total_Iterations; k++)        {
               for(m=k+1; m<Total_Iterations; m++)          {
                 for(n=m+1; n<Total_Iterations; n++)        {
              Qbw_temp[0] = Qbw_ls[j] + Qbw_ls[k] + Qbw_ls[m] +Qbw_ls[n];
                    if (Qbw_temp[0] <= Rem_BW_avail[0])      {
                       Qprob_temp[0]=Qprob_ls[j]+Qprob_ls[k]+Qprob_ls[m]+Qprob_ls[n];
```

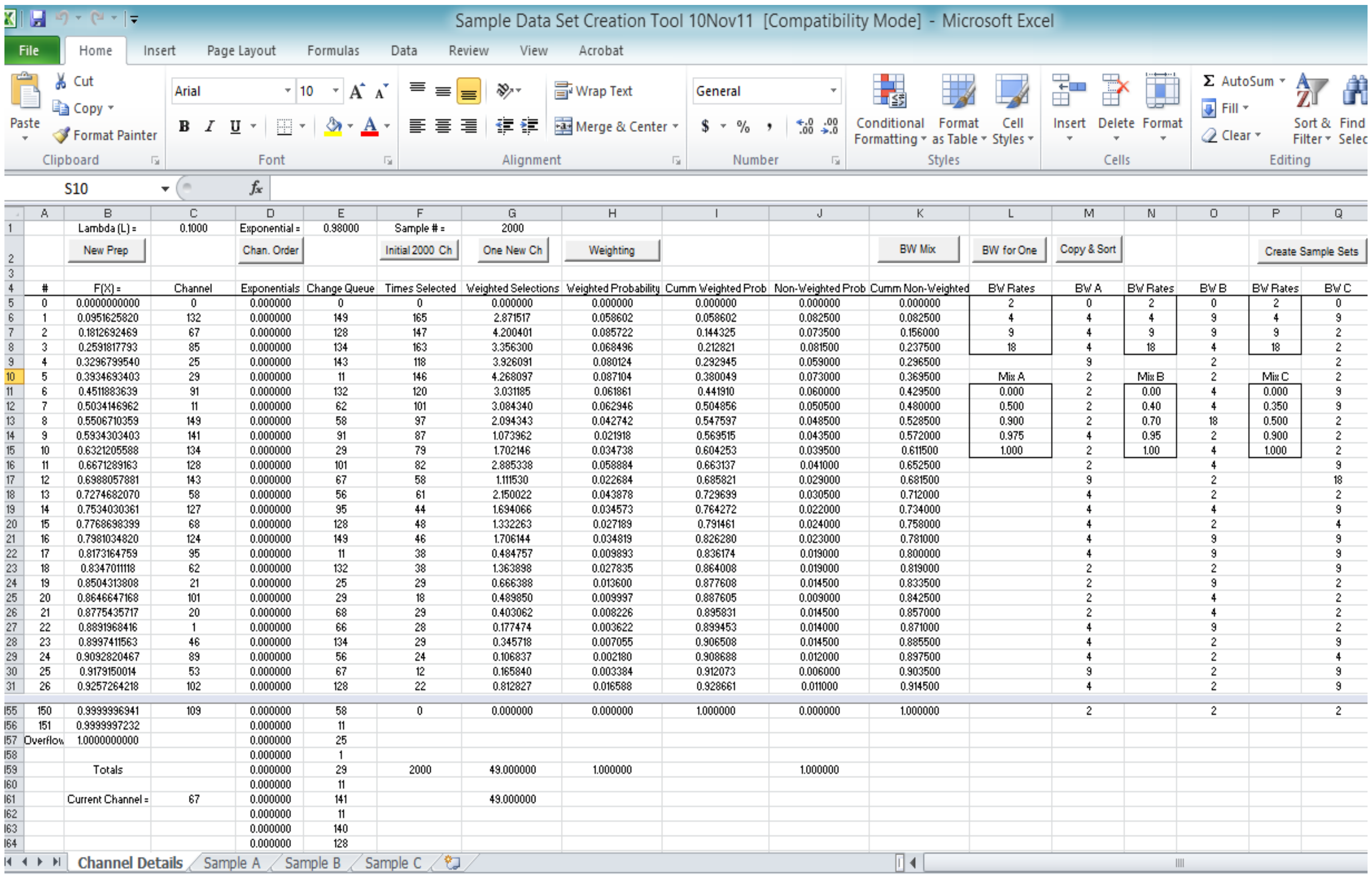
```
                        if (Qprob_temp[0] > Best_prob[0])    {
                                                Qchan_best[0] = Qchan_ls[j];
                                                Qchan_best[1] = Qchan_ls[k];
                                                Qchan_best[2] = Qchan_ls[m];
                                                Qchan_best[3] = Qchan_ls[n];
                                                Best_prob[0] = Qprob_temp[0];
                                                Best_bw[0] = Qbw_temp[0];
                                        }
                                }
                        }
                    }
                }
            }
                            // Fifth Iteration

            for(j=0; j<Total_Iterations; j++)        {
                for(k=j+1; k<Total_Iterations; k++)        {
                    for(m=k+1; m<Total_Iterations; m++)        {
                        for(n=m+1; n<Total_Iterations; n++) {
                            for(p=n+1; p<Total_Iterations; p++)  {
                Qbw_temp[0]=Qbw_ls[j]+Qbw_ls[k]+Qbw_ls[m]+Qbw_ls[n]+Qbw_ls[p];
    if (Qbw_temp[0] <= Rem_BW_avail[0])        {
    Qprob_temp[0]=Qprob_ls[j]+Qprob_ls[k]+Qprob_ls[m]+Qprob_ls[n]+Qprob_ls[p];
                                if (Qprob_temp[0] > Best_prob[0])    {
                                        Qchan_best[0] = Qchan_ls[j];
                                        Qchan_best[1] = Qchan_ls[k];
                                        Qchan_best[2] = Qchan_ls[m];
                                        Qchan_best[3] = Qchan_ls[n];
                                        Qchan_best[4] = Qchan_ls[p];
                                        Best_prob[0] = Qprob_temp[0];
                                        Best_bw[0] = Qbw_temp[0];
                                            }
                                        }
                                }
                            }
                        }
                    }
            }
                                // Sixth Iteration

            for(j=0; j<Total_Iterations; j++)        {
                    for(k=j+1; k<Total_Iterations; k++)   {
                        for(m=k+1; m<Total_Iterations; m++)        {
                            for(n=m+1; n<Total_Iterations; n++) {
                                for(p=n+1; p<Total_Iterations; p++)  {
                                    for(q=p+1; q<Total_Iterations; q++)  {
```

```
Qbw_temp[0]=Qbw_ls[j]+Qbw_ls[k]+Qbw_ls[m]+Qbw_ls[n]+Qbw_ls[p]+Qbw_ls[q];
                                        if (Qbw_temp[0] <= Rem_BW_avail[0])      {

Qprob_temp[0]=Qprob_ls[j]+Qprob_ls[k]+Qprob_ls[m]+Qprob_ls[n]+
Qprob_ls[p]+Qprob_ls[q];
                                        if (Qprob_temp[0] > Best_prob[0])    {
                                                Qchan_best[0] = Qchan_ls[j];
                                                Qchan_best[1] = Qchan_ls[k];
                                                Qchan_best[2] = Qchan_ls[m];
                                                Qchan_best[3] = Qchan_ls[n];
                                                Qchan_best[4] = Qchan_ls[p];
                                                Qchan_best[5] = Qchan_ls[q];
                                                Best_prob[0] = Qprob_temp[0];
                                                Best_bw[0] = Qbw_temp[0];
                                                                }
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }
Best_prob[0] = Best_prob[0] + Best_Prob_Hold1[0];
}

// Subroutine to load best six channels into confirmed arrays
void Best_to_Confirmed()
{
        j=0;
        for(i=0; i<6; i++)       {
                if (Qbw_best[i] == 2) {
                        Qchan_cf[Conf_chan_count + j] = Qchan_best[i];
                        Qprob_cf[Conf_chan_count + j] = Qprob_best[i];
                        Qbw_cf[Conf_chan_count + j] = Qbw_best[i];
                        j++;
                }
                else if (i<4 && Qbw_best[i] == 4)     {
                        Qchan_cf[Conf_chan_count + j] = Qchan_best[i];
                        Qprob_cf[Conf_chan_count + j] = Qprob_best[i];
                        Qbw_cf[Conf_chan_count + j] = Qbw_best[i];
                        j++;
                }
                else if (i<2 && Qbw_best[i] == 9)     {
                        Qchan_cf[Conf_chan_count + j] = Qchan_best[i];
                        Qprob_cf[Conf_chan_count + j] = Qprob_best[i];
                        Qbw_cf[Conf_chan_count + j] = Qbw_best[i];
                        j++;
                }
        }
        for(w=0; w<150; w++)          {
                Qchan_best[w] = 0;
                Qprob_best[w] = 0;
                Qbw_best[w] = 0;
        }
}

int main(void) {

LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Best_prob[0] = 0;
Best_bw[0] = 0;


// First Phase

Total_Iterations_Count();
Six_Iterations();
LS_BW_and_Prob_to_Best();
Confirmed_Count();
Best_to_Confirmed();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Confirmed_BW_and_Prob_Count();

//  Second Phase

Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Total_Iterations_Count();
Six_Iterations();
LS_BW_and_Prob_to_Best();
Confirmed_Count();
Best_to_Confirmed();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Confirmed_BW_and_Prob_Count();
```

```
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

//  Third Phase

Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Total_Iterations_Count();
Six_Iterations();
LS_BW_and_Prob_to_Best();
Confirmed_Count();
Best_to_Confirmed();
Confirmed_BW_and_Prob_Count();
LS_Check_Confirmed();
LS_Strip_Chan_0s();
LS_Clean_BW_and_Prob();
Rem_BW_avail[0] = BW_total - Curr_chan_bw - Curr_conf_BW[0];
Confirmed_BW_and_Prob_Count();
Rem_BW_avail[0] = BW_avail - Curr_conf_BW[0];

// Output Final Results
}
```

APPENDIX E

Appendix E.1 Visual of Sample Data Set Creation Tool



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Lambda (L) =	0.1000	Exponential =	0.98000	Sample # =	2000										
2		New Prep		Chan. Order		Initial 2000 Ch	One New Ch	Weighting			BW Mix	BW for One	Copy & Sort			Create Sample Sets	
3																	
4	#	F(X) =	Channel	Exponentials	Change Queue	Times Selected	Weighted Selections	Weighted Probability	Cumm Weighted Prob	Non-Weighted Prob	Cumm Non-Weighted	BW Rates	BW A	BW Rates	BW B	BW Rates	BW C
5	0	0.0000000000	0	0.000000	0	0	0.000000	0.000000	0.000000	0.000000	0.000000	2	0	2	0	2	0
6	1	0.0951625820	132	0.000000	149	165	2.871517	0.058602	0.058602	0.082500	0.082500	4	4	4	9	4	9
7	2	0.1812692469	67	0.000000	128	147	4.200401	0.085722	0.144325	0.073500	0.156000	9	4	9	9	9	2
8	3	0.2591817793	85	0.000000	134	163	3.356300	0.068496	0.212821	0.081500	0.237500	18	4	18	4	18	2
9	4	0.3296799540	25	0.000000	143	118	3.926091	0.080124	0.292945	0.059000	0.296500		9		2		2
10	5	0.3934693403	29	0.000000	11	146	4.268097	0.087104	0.380049	0.073000	0.369500	Mix A	2	Mix B	2	Mix C	2
11	6	0.4511883639	91	0.000000	132	120	3.031185	0.061861	0.441910	0.060000	0.429500	0.000	2	0.00	4	0.000	9
12	7	0.5034146962	11	0.000000	62	101	3.084340	0.062946	0.504856	0.050500	0.480000	0.500	2	0.40	4	0.350	9
13	8	0.5506710359	149	0.000000	58	97	2.094343	0.042742	0.547597	0.048500	0.528500	0.900	2	0.70	18	0.500	2
14	9	0.5934303403	141	0.000000	91	87	1.073962	0.021918	0.569515	0.043500	0.572000	0.975	4	0.95	2	0.900	2
15	10	0.6321205588	134	0.000000	29	79	1.702146	0.034738	0.604253	0.039500	0.611500	1.000	2	1.00	4	1.000	2
16	11	0.6671289163	128	0.000000	101	82	2.885338	0.058884	0.663137	0.041000	0.652500		2		4		9
17	12	0.6988057881	143	0.000000	67	58	1.111530	0.022684	0.685821	0.029000	0.681500		9		2		18
18	13	0.7274682070	58	0.000000	56	61	2.150022	0.043878	0.729699	0.030500	0.712000		4		2		2
19	14	0.7534030361	127	0.000000	95	44	1.694066	0.034573	0.764272	0.022000	0.734000		4		4		9
20	15	0.7768698399	68	0.000000	128	48	1.332263	0.027189	0.791461	0.024000	0.758000		4		2		4
21	16	0.7981034820	124	0.000000	149	46	1.706144	0.034819	0.826280	0.023000	0.781000		4		9		9
22	17	0.8173164759	95	0.000000	11	38	0.484757	0.009893	0.836174	0.019000	0.800000		4		9		9
23	18	0.8347011118	62	0.000000	132	38	1.363898	0.027835	0.864008	0.019000	0.819000		2		2		9
24	19	0.8504313808	21	0.000000	25	29	0.666388	0.013600	0.877608	0.014500	0.833500		2		9		2
25	20	0.8646647168	101	0.000000	29	18	0.489850	0.009997	0.887605	0.009000	0.842500		2		4		2
26	21	0.8775435717	20	0.000000	68	29	0.403062	0.008226	0.895831	0.014500	0.857000		2		4		2
27	22	0.8891968416	1	0.000000	66	28	0.177474	0.003622	0.899453	0.014000	0.871000		4		9		2
28	23	0.8997411563	46	0.000000	134	29	0.345718	0.007055	0.906508	0.014500	0.885500		4		2		9
29	24	0.9092820467	89	0.000000	56	24	0.106837	0.002180	0.908688	0.012000	0.897500		4		2		4
30	25	0.9179150014	53	0.000000	67	12	0.165840	0.003384	0.912073	0.006000	0.903500		9		2		9
31	26	0.9257264218	102	0.000000	128	22	0.812827	0.016588	0.928661	0.011000	0.914500		4		2		9
155	150	0.9999996941	109	0.000000	58	0	0.000000	0.000000	1.000000	0.000000	1.000000		2		2		2
156	151	0.9999997232		0.000000	11												
157	Overflow	1.0000000000		0.000000	25												
158				0.000000	1												
159		Totals		0.000000	29	2000	49.000000	1.000000		1.000000							
160				0.000000	11												
161		Current Channel =	67	0.000000	141		49.000000										
162				0.000000	11												
163				0.000000	140												
164				0.000000	128												

Appendix E.2 Sample Data Set Creation Tool Key Macros

Below are the key Excel macro routines that control the Sample Data Set Creation Tool.

```
Sub Button1_Click()
' New Prep clears various data cells for a fresh run

   Sheets("Weighting Comparisons").Select
   Range("B2:G151").Value = ""
   Range("I2:K151").Value = ""
   Range("H2").Select
   Sheets("Sample A").Select
   Range("A1:E151").Value = ""
   Range("A1").Select
   Sheets("Sample B").Select
   Range("A1:E151").Value = ""
   Range("A1").Select
   Sheets("Sample C").Select
   Range("A1:E151").Value = ""
   Range("A1").Select
   Sheets("Channel Details").Select
   Range("C2").Select
   Application.CutCopyMode = False
End Sub


Sub Button2_Click()
' Chan Order creates channels #'s 1 to 150

   Range("C6:C155").Value = ""
   rowCounter = 6
For i = 1 To 150
reChannel:
   generatedChannel = Int((150 - 1 + 1) * Rnd + 1)
      For Row = 6 To 155
         checkChannel = Range("C" & Row).Value
         If generatedChannel = checkChannel Then GoTo reChannel
      Next Row
    Range("C" & rowCounter).Value = generatedChannel
   rowCounter = rowCounter + 1
Next i
   Range("E2").Select
End Sub
```

```
Sub Button3_Click()
' **Initial 2000 Ch** randomly selects 2000 channels to populate History Buffer

Limit = Range("G1").Value + 6
Range("E6:E2005").Value = "0"
Range("F6:F155").Value = "0"
Range("G6:G155").Value = "0"
Range("H6:H155").Value = "0"
Range("I6:I155").Value = "0"
Range("J6:J155").Value = "0"
Range("K6:K155").Value = "0"
chcnt = 6
Do While (chcnt < Limit)
        generatedRandnum = Int((1000000000 - 0 + 1) * Rnd + 0)
        currentRandnum = generatedRandnum / 1000000000
   For i = 6 To 156
        currentFx = Range("B" & i).Value
        If currentRandnum > currentFx Then
           GoTo Newi
        Else
           If Range("C" & i).Value = Range("E" & (chcnt - 1)).Value Then
              chcnt = chcnt - 1
            Else
              Range("E" & chcnt).Value = Range("C" & i).Value
              chcnt = chcnt + 1
              i = 156
           End If
        End If
Newi:
   Next i
Loop
Range("C161").Value = Range("E" & Limit - 1).Value
   Range("I2").Select
End Sub


Sub Button4_Click()
' **One New Ch** makes a single selection and increments History Buffer

'Delect Oldest Channel and Decrement All Others in Buffer
   Range("E6").Select
   Selection.ClearContents
   Range("E7:E2005").Select
   Range("E2005").Activate
   Selection.Cut
   Range("E6").Select
   ActiveSheet.Paste
   Range("E2005").Select
'Select One New Random Channel
  For j = 0 To 10
         chcnt = Limit - 1
         generatedRandnum = Int((1000000000 - 0 + 1) * Rnd + 0)
         currentRandnum = generatedRandnum / 1000000000
         For i = 6 To 156
             currentFx = Range("B" & i).Value
             If currentRandnum > currentFx Then
                    GoTo Newi
                 Else
                    If Range("C" & i).Value = Range("E" & (chcnt - 1)).Value Then
                          GoTo Newj
                    Else
                           Range("E" & chcnt).Value = Range("C" & i).Value
                          i = 156
                          j = 10
                    End If
               End If
Newi:
         Next i
Newj:
Next j
Range("C161").Value = Range("E" & Limit - 1).Value
Range("I2").Select
End Sub


Sub Button5_Click()
' Weighting peforms the exponential time-weighting function

      'Times Selected & Weighted Selections
Limit = Range("G1").Value + 6
For j = 6 To 155
   Channel2Count = Range("C" & j).Value
   For k = 6 To Limit
      If Channel2Count = Range("E" & k).Value Then
         Range("F" & j).Value = Range("F" & j).Value + 1
         Range("G" & j).Value = Range("G" & j).Value + Range("D" & k).Value
      End If
   Next k
Next j
   'Weighted, Cumm Weighted, Non-Weighted, & Cumm Non-Weighted Probabilities
For n = 6 To 155
   Range("H" & n).Value = Range("G" & n).Value / Range("G159").Value
```

```
    Range("I" & n).Value = Range("I" & n - 1).Value + Range("H" & n).Value
    Range("J" & n).Value = Range("F" & n).Value / Range("F159").Value
    Range("K" & n).Value = Range("K" & n - 1).Value + Range("J" & n).Value
Next n
    Range("J2").Select
End Sub


Sub Button6_Click()
' BW Mix sets up the channel BWs per mix ratios

' Determine Channel Bandwidth For Mix A
    Sheets("Weighting Comparisons").Select
    Range("B2:G151").Value = ""
    Range("I2:K151").Value = ""
    Sheets("Channel Details").Select
    Range("M6:M155").Value = "0"
For i = 6 To 155
    genrand2 = Int((1000 - 0 + 1) * Rnd + 0)
    rand2 = genrand2 / 1000
    For j = 11 To 15
        If rand2 > Range("L" & j).Value Then
            If rand2 < Range("L" & (j + 1)).Value Then
              Range("M" & i).Value = Range("L" & (j - 6)).Value
            End If
        End If
    Next j
    If Range("M" & i).Value = 0 Then
        i = i - 1
    End If
Next i
    Range("M6:M155").Select
    Selection.Copy
    Sheets("Weighting Comparisons").Select
    Range("E2").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Sheets("Channel Details").Select

' Determine Channel Bandwidth For Mix B
Range("O6:O155").Value = "0"
For i = 6 To 155
    genrand2 = Int((1000 - 0 + 1) * Rnd + 0)
    rand2 = genrand2 / 1000
    For j = 11 To 15
        If rand2 > Range("N" & j).Value Then
```

```
            If rand2 < Range("N" & (j + 1)).Value Then
              Range("O" & i).Value = Range("N" & (j - 6)).Value
            End If
        End If
    Next j
    If Range("O" & i).Value = 0 Then
        i = i - 1
    End If
Next i
    Range("O6:O155").Select
    Selection.Copy
    Sheets("Weighting Comparisons").Select
    Range("F2").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Sheets("Channel Details").Select

' Determine Channel Bandwidth For Mix C
Range("Q6:Q155").Value = "0"
For i = 6 To 155
    genrand2 = Int((1000 - 0 + 1) * Rnd + 0)
    rand2 = genrand2 / 1000
    For j = 11 To 15
        If rand2 > Range("P" & j).Value Then
            If rand2 < Range("P" & (j + 1)).Value Then
              Range("Q" & i).Value = Range("P" & (j - 6)).Value
            End If
        End If
    Next j
    If Range("Q" & i).Value = 0 Then
        i = i - 1
    End If
Next i
    Range("Q6:Q155").Select
    Selection.Copy
    Sheets("Weighting Comparisons").Select
    Range("G2").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Sheets("Channel Details").Select
    Range("J2").Select
        Application.CutCopyMode = False
End Sub
```

APPENDIX F

Appendix F.1 Sample Moment Model Example Calculations of 2nd Order PDF

Sample Moment Model Full & 2nd Order PDF Calculations for λ=0.1 & Mix A

Lambda (λ)	pf	$\bar{x}$	Shift Factor	Full Norm	BW Norm
0.10	1.19	0.00671	-0.001403	0.209102	0.3023

Mix A %s:	0.500	0.400	0.075	0.025	Average CH BW	Avg Cache CH BW
					3.725	2.678

				BW Rates (Mbps)					
#	Exponential	Exponential$_{norm}(X_i)$	Full PDF$_{norm}$	2	4	9	18	All BW Probability	Cumulative Prob.
1	0.09048374	0.09516261	0.09516308	0.06899128	0.02419126	0.00172807	0.00025247	0.09516308	0.0951631
2	0.08187308	0.08610669	0.08610711	0.06242589	0.02188916	0.00156363	0.00022845	0.08610711	0.1812702
3	0.07408182	0.07791256	0.07791294	0.05648527	0.01980613	0.00141483	0.00020671	0.07791294	0.2591831
4	0.06703200	0.07049820	0.07049854	0.05110999	0.01792132	0.00128019	0.00018704	0.07049854	0.3296817
5	0.06065307	0.06378941	0.06378971	0.04624623	0.01621588	0.00115836	0.00016924	0.06378971	0.3934714
6	0.05488116	0.05771904	0.05771931	0.04184531	0.01467274	0.00104813	0.00015313	0.05771931	0.4511907
7	0.04965853	0.05222635	0.05222659	0.03786320	0.01327644	0.00094839	0.00013856	0.05222659	0.5034173
8	0.04493290	0.04725636	0.04725657	0.03426004	0.01201302	0.00085814	0.00012537	0.04725657	0.5506739
9	0.04065697	0.04275932	0.04275951	0.03099976	0.01086983	0.00077647	0.00011344	0.04275951	0.5934334
10	0.03678794	0.03869023	0.03869040	0.02804974	0.00983543	0.00070258	0.00010265	0.03869040	0.6321238
11	0.03328711	0.03500837	0.03500852	0.02538046	0.00889946	0.00063572	0.00009288	0.03500852	0.6671323
12	0.03011942	0.03167688	0.03167701	0.02296518	0.00805256	0.00057523	0.00008404	0.03167701	0.6988093
13	0.02725318	0.02866243	0.02866254	0.02077975	0.00728626	0.00052049	0.00007604	0.02866254	0.7274718
14	0.02465970	0.02593484	0.02593494	0.01880230	0.00659288	0.00047095	0.00006881	0.02593494	0.7534068
15	0.02231302	0.02346681	0.02346690	0.01701302	0.00596548	0.00042614	0.00006226	0.02346690	0.7768737
145	0.00000005	0.00000005	0.00000002	0.00000001	0.00000000	0.00000000	0.00000000	0.00000002	1.0000000
146	0.00000005	0.00000005	0.00000001	0.00000001	0.00000000	0.00000000	0.00000000	0.00000001	1.0000000
147	0.00000004	0.00000004	0.00000001	0.00000001	0.00000000	0.00000000	0.00000000	0.00000001	1.0000000
148	0.00000004	0.00000004	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.0000000
149	0.00000003	0.00000004	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.0000000
Totals	0.9508329	1.0000000000	1.0000000000	0.724979412	0.254208440	0.018159088	0.002653060	1.0000000000	1.0000000

Appendix F.2 Detailed Performance Simulation Cumulative Summaries

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.6376	0.6152	0.3476	0.6535	0.5868	0.5386	0.8058	0.7019	0.6148	0.8091	0.7427	0.4538	0.7764	0.6315	0.5714	0.8290	0.7188	0.5887
Mean Prob. for 50 Mbps Rate =	0.6457																	
Probability at 65 Mbps Rate =	0.7347	0.6946	0.4381	0.7504	0.6558	0.6079	0.8931	0.7738	0.7070	0.8832	0.8189	0.5671	0.8533	0.7451	0.6581	0.9120	0.8306	0.6976
Mean Prob. for 65 Mbps Rate =	0.7345																	
Probability at 75 Mbps Rate =	0.7866	0.7275	0.5014	0.8068	0.6979	0.6439	0.9302	0.8071	0.7699	0.9225	0.8540	0.6246	0.8778	0.7849	0.7043	0.9490	0.8646	0.7424
Mean Prob. for 75 Mbps Rate =	0.7775																	

Mean Probability for all Samples = 0.7193

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.5335	0.5929	0.7075	0.6686	0.6598	0.7122
Mean Prob. by λ for 65 Mbps Rate =	0.6225	0.6713	0.7913	0.7564	0.7522	0.8134
Mean Prob. by λ for 75 Mbps Rate =	0.6718	0.7162	0.8357	0.8004	0.7890	0.8520
Mean Prob. for all Rates by λ =	0.6093	0.6602	0.7782	0.7418	0.7337	0.7925

Sample Moment Model Cumulative Summary

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.6376	0.6195	0.3514	0.6554	0.5868	0.5477	0.8061	0.7019	0.6148	0.8091	0.7427	0.4538	0.7764	0.6129	0.5436	0.8290	0.7188	0.5887
Mean Prob. for 50 Mbps Rate =	0.6442																	
Probability at 65 Mbps Rate =	0.7347	0.6844	0.4404	0.7504	0.6565	0.6130	0.8941	0.7738	0.7069	0.8832	0.8187	0.5671	0.8533	0.7451	0.6541	0.9120	0.8306	0.6976
Mean Prob. for 65 Mbps Rate =	0.7342																	
Probability at 75 Mbps Rate =	0.7870	0.7342	0.4830	0.8068	0.6954	0.6464	0.9296	0.7999	0.7699	0.9225	0.8539	0.6246	0.8778	0.7857	0.7003	0.9506	0.8646	0.7255
Mean Prob. for 75 Mbps Rate =	0.7754																	

Mean Probability for all Samples = 0.7180

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.5362	0.5966	0.7076	0.6686	0.6443	0.7122
Mean Prob. by λ for 65 Mbps Rate =	0.6199	0.6733	0.7916	0.7564	0.7508	0.8134
Mean Prob. by λ for 75 Mbps Rate =	0.6681	0.7162	0.8331	0.8003	0.7879	0.8469
Mean Prob. for all Rates by λ =	0.6080	0.6620	0.7774	0.7418	0.7277	0.7908

Probability Divided by Bandwidth Model Cumulative Summary

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.5742	0.5422	0.3092	0.5517	0.5216	0.4699	0.7895	0.5848	0.5853	0.8091	0.7253	0.3729	0.6871	0.6130	0.4639	0.8064	0.6936	0.4786
Mean Prob. for 50 Mbps Rate =	0.5877																	
Probability at 65 Mbps Rate =	0.6950	0.6273	0.3184	0.7217	0.6011	0.5761	0.8931	0.7541	0.6593	0.8600	0.8147	0.5383	0.8201	0.7299	0.5801	0.9090	0.7364	0.5505
Mean Prob. for 65 Mbps Rate =	0.6881																	
Probability at 75 Mbps Rate =	0.7545	0.6530	0.3620	0.7808	0.6486	0.6012	0.9302	0.7252	0.7176	0.9105	0.8562	0.6114	0.8810	0.7720	0.6543	0.9444	0.8461	0.6688
Mean Prob. for 75 Mbps Rate =	0.7399																	
Mean Probability for all Samples =	**0.6719**																	

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.4752	0.5144	0.6532	0.6358	0.5880	0.6595
Mean Prob. by λ for 65 Mbps Rate =	0.5469	0.6330	0.7688	0.7377	0.7100	0.7320
Mean Prob. by λ for 75 Mbps Rate =	0.5899	0.6769	0.7910	0.7927	0.7691	0.8198
Mean Prob. for all Rates by λ =	0.5373	0.6081	0.7377	0.7221	0.6890	0.7371

Probability Only Model Cumulative Summary

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.5855	0.5470	0.3414	0.5667	0.5481	0.5121	0.7813	0.5894	0.5949	0.7832	0.7388	0.4538	0.6498	0.6130	0.5178	0.7829	0.6044	0.5258
Mean Prob. for 50 Mbps Rate =	0.5964																	
Probability at 65 Mbps Rate =	0.6741	0.5969	0.4286	0.7012	0.6204	0.5554	0.8675	0.6843	0.6704	0.8383	0.7807	0.5021	0.7275	0.6974	0.6036	0.8810	0.6694	0.6083
Mean Prob. for 65 Mbps Rate =	0.6726																	
Probability at 75 Mbps Rate =	0.7248	0.6537	0.4102	0.7720	0.6829	0.5693	0.9106	0.6916	0.6720	0.8617	0.8209	0.5686	0.7877	0.7737	0.6444	0.8784	0.7131	0.6688
Mean Prob. for 75 Mbps Rate =	0.7114																	
Mean Probability for all Samples =	**0.6601**																	

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.4913	0.5423	0.6552	0.6586	0.5935	0.6377
Mean Prob. by λ for 65 Mbps Rate =	0.5665	0.6256	0.7407	0.7070	0.6762	0.7196
Mean Prob. by λ for 75 Mbps Rate =	0.5963	0.6747	0.7581	0.7504	0.7352	0.7535
Mean Prob. for all Rates by λ =	0.5514	0.6142	0.7180	0.7053	0.6683	0.7036

Exhaustive Search Model Cumulative Summary

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.4244	0.4773	0.2695	0.5242	0.5436	0.4265	0.6286	0.5488	0.4931	0.4997	0.6458	0.3495	0.6517	0.5625	0.3938	0.6579	0.4898	0.4285
Mean Prob. for 50 Mbps Rate =	0.5008																	
Probability at 65 Mbps Rate =	0.4381	0.4864	0.3298	0.5291	0.6009	0.4945	0.7987	0.6655	0.5632	0.7431	0.7675	0.3587	0.8184	0.6536	0.4000	0.8055	0.6713	0.6131
Mean Prob. for 65 Mbps Rate =	0.5965																	
Probability at 75 Mbps Rate =	0.4386	0.5414	0.3338	0.6519	0.6350	0.5050	0.8411	0.7204	0.6906	0.8235	0.7818	0.3611	0.8603	0.6993	0.5014	0.8960	0.6816	0.6547
Mean Prob. for 75 Mbps Rate =	0.6454																	
Mean Probability for all Samples =	**0.5809**																	

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.3904	0.4981	0.5568	0.4983	0.5360	0.5254
Mean Prob. by λ for 65 Mbps Rate =	0.4181	0.5415	0.6758	0.6231	0.6240	0.6966
Mean Prob. by λ for 75 Mbps Rate =	0.4379	0.5973	0.7507	0.6555	0.6870	0.7441
Mean Prob. for all Rates by λ =	0.4155	0.5456	0.6611	0.5923	0.6157	0.6554

Maximum Channels Model Cumulative Summary

Sample Data Set Number:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
λ value:	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.10	0.10	0.10
Probability at 50 Mbps Rate =	0.2157	0.0495	0.1000	0.2637	0.0835	0.1865	0.2245	0.3269	0.2479	0.3550	0.2401	0.1437	0.0988	0.2010	0.0947	0.2756	0.3433	0.2755
Mean Prob. for 50 Mbps Rate =	0.2070																	
Probability at 65 Mbps Rate =	0.3931	0.1246	0.1214	0.3037	0.1584	0.1104	0.4567	0.1703	0.0877	0.4954	0.1815	0.2244	0.2825	0.1854	0.1774	0.5273	0.3922	0.2908
Mean Prob. for 65 Mbps Rate =	0.2602																	
Probability at 75 Mbps Rate =	0.4317	0.1548	0.1851	0.4471	0.1347	0.0780	0.3388	0.1047	0.0180	0.4628	0.1891	0.1387	0.3088	0.0842	0.2787	0.5137	0.4130	0.2362
Mean Prob. for 75 Mbps Rate =	0.2510																	
Mean Probability for all Samples =	**0.2394**																	

λ value:	0.05	0.06	0.07	0.08	0.09	0.1
Mean Prob. by λ for 50 Mbps Rate =	0.1218	0.1779	0.2665	0.2463	0.1315	0.2981
Mean Prob. by λ for 65 Mbps Rate =	0.2130	0.1908	0.2382	0.3004	0.2151	0.4034
Mean Prob. by λ for 75 Mbps Rate =	0.2572	0.2199	0.1538	0.2635	0.2239	0.3876
Mean Prob. for all Rates by λ =	0.1973	0.1962	0.2195	0.2701	0.1902	0.3631

High Definition Model Cumulative Summary

APPENDIX G

Appendix G.1 Packet Prioritization

Recommended IP packet tagging processes to prioritize network traffic for this system are the following:

- IP Type of Service (TOS) Field
- IP Traffic Class Field
- Precedence Bits (P-Bits)
- Virtual Local Area Networks (VLAN)

The multicast cache packets could be given significantly lower priority such that they would be preempted if cases of higher priority packet bursting. Full application details on these processes are available in the industry standards listed in Appendix A.2 and in references [98-99][111-113].

Appendix G.2 Operational Testing Code for Linux Platform

The Processing and Memory Testing as discussed in Chapter VII was conducted on the Linux computer as described in the chapter. This testing required additional C++ code to capture the critical details involved in the tests, and this code was then recompiled to operate on the Linux platform. The following critical code lines were inserted into the Sample Moment Model at key positions to facilitate the testing.

```
// Routine to set up structures and output key metrics required for testing
void process(struct rusage *p, char *when)
{
 printf("%s\n", when);
 printf(" /* user time used */  %8d  %8d\n",  p->ru_utime.tv_sec,p->ru_utime.tv_usec );
 printf(" /* system time used */ %8d  %8d\n",  p->ru_stime.tv_sec,p->ru_stime.tv_usec );
      printf(" /* integral sset size */      %8d\n",  p->ru_maxrss          );
      printf(" /* integral shared memory size */      %8d\n",  p->ru_ixrss          );
      printf(" /* integral unshared data  */         %8d\n",  p->ru_idrss          );
      printf(" /* integral unshared stack  */        %8d\n",  p->ru_isrss          );
      printf(" /* page reclaims */                  %8d\n",  p->ru_minflt         );
      printf(" /* page faults */                   %8d\n",  p->ru_majflt         );
      printf(" /* swaps */                         %8d\n",  p->ru_nswap          );
      printf(" /* block input operations */          %8d\n",  p->ru_inblock        );
      printf(" /* block output operations */          %8d\n",  p->ru_oublock        );
         printf(" /* messages sent */                  %8d\n",  p->ru_msgsnd         );
      printf(" /* messages received */               %8d\n",  p->ru_msgrcv         );
      printf(" /* signals received */               %8d\n",  p->ru_nsignals       );
      printf(" /* voluntary context switches */       %8d\n",  p->ru_nvcsw          );
      printf(" /* involuntary  */                   %8d\n",  p->ru_nivcsw         );
}
        int who= RUSAGE_SELF;
        struct rusage usage;
        struct rusage *pjr=&usage;

// Processing time report
ret=getrusage(who,pjr);
        process(pjr, "-------------before");

// Memory Time Report requires operator to address PC statistics at this point
cout << "Calculate RAM then hit any key to continue: " <<'\t'<<'\t';
cin >> input_key[0];
```

VITA

Thomas R. Ray

Candidate for the Degree of

Doctor of Philosophy

Thesis: IMPROVED IPTV CHANNEL CHANGE TIMES THROUGH MULTICAST CACHING OF PRE-SELECTED CHANNELS

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2014.

Completed the requirements for the Master of Science in Electrical Engineering at University of Arkansas, Fayetteville, Arkansas in 1994.

Completed the requirements for the Bachelor of Science in Electrical Engineering at University of Arkansas, Fayetteville, Arkansas in 1991.

Experience:

Self-employed as a Consulting Engineer for last 11 years.

Graduate Teaching Assistant at Oklahoma State University 2009.

Director of Network Engineering at Sycamore Networks 1999-2003.

Senior Manager of Implementations at MCI WorldCom 1997-1999.

Senior Engineer at WilTel 1993-1996.

Graduate Research Assistant at the University of Arkansas 1991-1993.

Professional Memberships:

Member, IEEE and Communications Society since 1989.