ACTIVESTB: AN EFFICIENT WIRELESS RESOURCE MANAGER IN HOME NETWORKS

A Thesis

by

VARRIAN DURAND HALL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2006

Major Subject: Computer Science

ACTIVESTB: AN EFFICIENT WIRELESS RESOURCE MANAGER IN HOME NETWORKS

A Thesis

by

VARRIAN DURAND HALL

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Eun Jung Kim
Committee Members,	Anxiao (Andrew) Jiang
	Deepa Kundur
Head of Department,	Valerie E. Taylor

December 2006

Major Subject: Computer Science

ABSTRACT

ActiveSTB: An Efficient Wireless Resource Manager
in Home Networks. (December 2006)

Varrian Durand Hall, B.S., Mississippi State University

Chair of Advisory Committee: Dr. Eun Jung Kim

The rapid growth of new wireless and mobile devices accessing the internet has led to an increase in the demand for multimedia streaming services. These home-based wireless connections require efficient distribution of shared network resources which is a major concern for the transport of stored video. In our study, a set-top box is the access point between the internet and a home network. Our main goal is to design a set-top box capable of performing network flow control in a home network and capable of quality adaptation of the delivered stream quality to the available bandwidth. To achieve our main goal, estimating the available bandwidth quickly and precisely is the first task in the decision of streaming rates of layered and scalable multimedia services. We present a novel bandwidth estimation method called *IdleGap* that uses the NAV (Network Allocation Vector) information in the wireless LAN. We will design a new set-top box that will implement *IdleGap* and perform buffering and quality adaptation to a wireless network based on the *IdleGap's* bandwidth estimate. We use a network simulation tool called NS-2 to evaluate *IdleGap* and our ActiveSTB compared to traditional STBs. We performed several tests simulating network conditions over various ranges of cross traffic with different error rates and observation times. Our simulation results reveal how *IdleGap* accurately estimates the available bandwidth for all ranges of cross traffic (100Kbps ~ 1Mbps) with a very short observation time (10 seconds). Test results also reveal how our novel ActiveSTB outperforms traditional STBs and provides good QoS to the end-user by reducing latency and excess bandwidth consumption.

DEDICATION

To my mother

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. E. J. Kim, for her guidance and support throughout the course of this research; and thanks also to Dr. Karen Butler and Dr. Anxiao Jiang for your support. Heung Ki Lee is a Ph.D. student who is the technical lead of this research, so I also thank Heung Ki Lee for his guidance.

I'd like to thank all of my friends, colleagues, and department faculty and staff for helping me enjoy and appreciate my experience at Texas A&M University. I especially thank the A&M faculty who helped to ensure my successful graduation, and a special thanks to Mona Osborne for her support and encouraging words during difficult times. All of this work could not have been done were it not for the generosity of the National Science Foundation's Louis Stokes Alliance for Minority Participation (LSAMP) Bridge To Doctorate(BTD) Fellowship.

Finally, thanks to my mom and dad for helping me to be the person I am today.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

This research provides a solution to networking issues related to multimedia video streaming in the wireless home environment. Competitive pricing of home-based network electronic devices has caused the home network to rapidly increase in complexity. Home networks consist of various network devices from multiple vendors and different hardware generations that are added to the home over time. Also, internet access in home environments has significantly increased and is deeply heterogeneous. The rapid and widespread usage of the internet has given rise to an increase in demand for audio and video streaming [1]. Although there has been a significant increase in user requests for streaming audiovisual information over the internet, the quality of streamed multimedia content still requires significant improvement in order to be accepted as an alternative by the mass television audiences. In order to support such a large number of wireless clients, techniques that allow fast access and transport of wireless streaming data are essential.

In our study, we mainly focus on the LAN/WAN shown in Figure 1.1. In this figure, an Internet-based STB is an interface between a wired network and a wireless network and serves as a bottleneck between the server and heterogeneous client devices. Thus, the quality (or bandwidth) of the delivered stream to such devices is limited to the bottleneck bandwidth between the server and the client [2]. Even though wired networks can provide high and stable bandwidths, fragile wireless networks cannot. Therefore, for layered streaming services, it is very critical for the STB to know the available wireless network's bandwidth in order to efficiently distribute stored requested information to heterogeneous clients (or devices).

This thesis follows the style of *IEEE Transactions on Multimedia*.

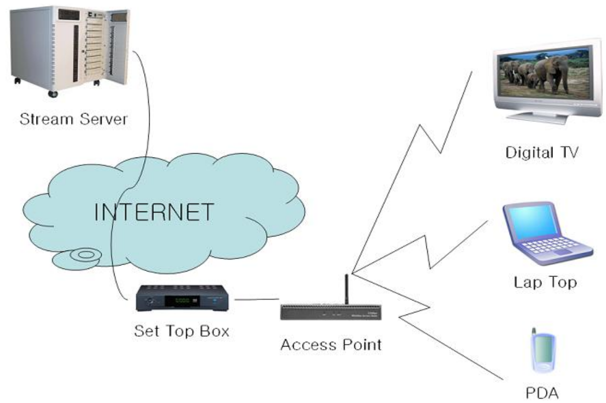

Figure 1.1 Streaming services through a STB.

In wireless networks, the IEEE 802.11 protocol in Distributed Co-ordination Function (DCF) mode, based on CSMA/CA algorithm, has become very popular. Previous works [3,4,5] based on bandwidth estimation in wired environments are not applicable to wireless networks that use the DCF protocol. These methods require probing time which adds delay in processing; however, multimedia streaming is a soft real-time service where each frame is delay-sensitive. Swiftness and availability of each frame is critical for real time systems; therefore, during bandwidth deviations, the rate of the transmitted multimedia streams should change expeditiously. The accuracy of previous bandwidth estimation methods, *Spruce*[3] and *ProbeGap*[5], is dependent on probing time and the volume of the packets for probing. *ProbeGap* produces good estimates at low cross traffic rates (2 Mbps cross traffic regardless of the cross traffic packet size); however, it significantly overestimates available bandwidth when the cross traffic is high (4 Mbps cross traffic generated with 300-byte packets) [5]. Another reason why *Spruce* and *ProbeGap* are not practical for real time video streaming applications is because the influence by cross traffic on probe packet sequences causes probe packets in sequences to be split up or even lost**.** On the other hand, our real-time bandwidth estimation tool, *IdleGap*, for a wireless network is independent of cross traffic. *IdleGap* estimates the available bandwidth via the ratio of free time or idle time in the wireless links. To get the ratio of idle time in a wireless network, *IdleGap* uses information from network management at the low layer. This info provides us with an efficient and fast method for estimating the available bandwidth.

The increase in streaming video has led to many technical challenges that must be addressed in the two areas of video-coding and networking. One such method that addresses the challenges associated with streaming video and networking is scalability. Scalability plays a crucial role in delivering the best possible video quality over unpredictable “best-effort” networks. In Chapter II we will discuss how our ActiveSTB, a set-top box, will utilize scalability and *IdleGap* in the delivery of the best quality data to end-user.

A set-top box (STB) is a device that converts some external signal source to content that can be displayed on screen, .i.e., receives the television signal, runs the interactive applications and transfers the digital TV signal to the TV [6]. Set-top boxes (STBs) are becoming key devices in home entertainment networks, not only to receive digital television (DTV), but also as residential gateways to deliver multiple services as well [7]. STBs in home networks have gained in flexibility and modularity, therefore, the functionality of the STB may be distributed between a main device and several peripherals, all interconnected by an ethernet or wireless network [8].

The goal of our STB will be to forward a partially buffered(or cached) stream to a client to allow a user to view quality playback of video while simultaneously performing quality-adaptation to changes in network bandwidth This research will demonstrate the uniqueness of our novel STB called ActiveSTB that utilizes *IdleGap* [9] to acquire the real-time available wireless bandwidth. This research will demonstrate how our ActiveSTB improves network performance and QoS to end-user by (1) efficiently caching the layer encoded video, (2) decreasing excess bandwidth consumption, and (3) reducing latency to end-user. We start by streaming a single video object, introduce packet loss, and monitor both the quality of video objects sent to user and the ActiveSTBs' ability to adapt to the available bandwidth from STB to end-user. The overall success of our ActiveSTB is dependent upon its unique ability to estimate the available wireless bandwidth.

CHAPTER II

BACKGROUND

Bandwidth Estimation in Broadband Networks

Since the introduction of Cprobe [10], a method for estimating bandwidth using Internet Control Message Protocol (ICMP) packet trains, many tools have been suggested. Cprobe uses packet trains to estimate the current congestion along a path. Cprobe bounces a short stream of echo packets off a target server and records the time between the receipt of the first packet and the receipt of the last packet. Dividing the number of bytes sent by this time yields a measure of available bandwidth. In order to tolerate packet drops and possible re-ordering of packets, Cprobe uses results of four eparate 10-packet streams when calculating the available bandwidth. Cprobe's successors *Spruce* and *IGI* use the interval of consecutive probe packets, since the interval or gap between probe packets is increased in heavy cross traffic. *Spruce* and *IGI* are both designed around the probe gap model which assumes a single bottleneck [11]. *Spruce* samples the arrival rate at the bottleneck queue before the first packet departs the queue. *Spruce* calculates the number of bytes received at the queue between two probes for the inter-probe spacing at the receiver. *Spruce* then computes the available bandwidth as the difference between the path capacity and the arrival rate at the receiver bottleneck [12]. The *IGI* algorithm sends a sequence of packet trains with an increasing initial gap, from the source to the destination host. *IGI* monitors the difference between the average source (initial) and destination (output) gap and terminates when it becomes zero. At that point, the packet train is operating at the turning point [11]. *Topp* [13] and *Pathload* [14] are also based on the rate of incoming packets. The comparison of the incoming rate from the sender side to the outgoing rate at the receiver side reveals the incoming rate to be less than or equal to the available bandwidth of the probing link. In *Probegap*[5], the link's idle time is the milestone for bandwidth estimation of a wireless network; however, *ProbeGap* also must send several probe packets over a specified interval.

All of the methods outlined above introduce additional traffic into the link and all require a probing sequence time to send and process the probing packets. To account for lost probes, additional probes are sent requiring more processing and filtering out of bad estimates. As a result, most of these methods may not be applicable to certain applications requiring instant bandwidth estimates, and if the link is congested many probes may not reach destination. Specifically, strict time bounds required of multimedia applications impose upper limits on delay and jitter in addition to the usual performance metrics of throughput and packet loss.

Bandwidth Estimation in Wireless Networks

Real time bandwidth estimation is a very challenging problem for real-time applications in wireless networks. There are two factors making this problem unique. First, unlike wired networks, traditional FIFO is not used to schedule bandwidth among connections in wireless networks. To avoid collisions in wireless networks, nodes are arranged in a distributed manner. This arrangement causes previously discussed bandwidth estimation methods in wired networks using intervals or rates to be inapplicable for bandwidth estimation in wireless networks. Second, the probing time required by these methods in determining the available bandwidth should be minimized for time-sensitive multimedia streaming services.

J. Padhye et al [5] and Mark Davis [15] suggested that the idle time in a wireless link can be a major milestone in estimating the available bandwidth as follows. Let *C* be the capacity of the wireless network[1]. *Idle_rate* indicates the rate at which the link is idle. Then the available bandwidth (B_{avail}) can be obtained by the following product:

$$B_{avail} = C \times Idle\_rate \tag{2.1}$$

---

[1] It can be changed by the negotiated data rate between a wireless node and the access point.

However, previous methods like *ProbeGap*[5] and *Topp*[13] using this formula required too much overhead for bandwidth estimation to be used in a real-time system. In *ProbeGap*, too much time elapsed probing the link and analyzing probing data, and results showed multiple incorrect estimates in heavy traffic. *Topp* utilized too much time in order to capture whole packets in the network and acquiring node information from captured packets. For real-time applications such as multimedia streams, it is impractical to use these methods; therefore, in this research we introduce an efficient method known as *IdleGap* that utilizes the *Idle_rate* in determining the available bandwidth. As will be shown later, *IdleGap* doesn't require introducing any probes to the link and is immune to cross traffic.

Cross Layer Feedback

For efficient mobile device communication and interaction, cross layer feedback is performed by a mobile device accessing its own protocol stack layers that contain information from transmitted packets. Cross layer feedback allows interaction between a layer and any other layer in the protocol stack. Packet information retrieval across the protocol stack layers, i.e. cross layering, provides very useful information about mobile devices in a wireless network. Several studies [15, 16, 17, 18] have revealed interaction across layers aid in improving a system. Samarth H. Shah [16] et al proposes the use of a centralized *Bandwidth Manager* (BM), which obtains from each flow its channel time proportion (CTP) requirements, at the start of its session. It uses this information to gauge what proportion of unit channel time each flow should be allotted. Samarth's et al system takes advantage of *cross-layer interaction* between the application/middleware and link layers. Mark Davis [15] suggested an 802.11 management method that processes the captured frame to obtain the available bandwidth. Davis's method describes a WLAN traffic probe that operates at the MAC layer and is capable of producing real time information on resource usage on a per-station basis. For a QoS-sensitive application, a different priority at the MAC layer may be assigned based on the applications [17]. Robert L. Carter [18] uses bandwidth probing to measure bandwidth and congestion at the application level. All of these methods infer the ability to gather, compute, and share useful information for bandwidth estimation across the OSI layers.

Caching

There has been a handful of studies that show how a proxy server can improve service quality for multimedia streaming services [19,20,21,22]. In [20], a proxy server caches parts of a multimedia stream such as the initial part of the video. For excessively high data rate transmissions, a proxy server stores a part of the highly transmitted multimedia data [19]. According to a client's request for a low quality multimedia stream, the proxy server degrades the bit rate of the cached stream to improve the hit rate [21,22]. To the best of our knowledge, methods for supporting multiple heterogeneous multimedia terminals simultaneously have not been studied so far. Therefore, in this research we suggest how an intelligent STB that caches data like a proxy can enhance the performance of multimedia streaming services to multiple wireless heterogeneous multimedia.

Set-Top Boxes

Earlier we discussed the problems and solutions of bandwidth estimation in wireless networks, we now discuss the set-top box (STB) and its use in delivering streaming data in wireless LANS. Typically an STB receives a request from a client, retrieves the requested multimedia data from the server, and forwards it to the multimedia terminal. During this process, the STB can cache portions of the stream and forward the cached stream data to multimedia terminals through a shared resource known as the wireless channel. The STB caches and forwards the streaming data between two different networks, wired and wireless networks, in order to reduce negative effects of network traffic such as late packets. The more resources assigned to handle the streams, the less jitter the terminal will experience within the network. The wireless channel is a limited shared resource available for servicing heterogeneous multimedia streams. Therefore, a simple and effective allocation strategy for the STB cache and the wireless channel is critical to improving the quality of the video streams delivered through the STB and the wireless network. In general, the streaming services with high quality may require more resources than the ones with low quality. Unfortunately, the amount of resources required for each case is not fully understood yet, so currently our research focuses on how to manage the resources for heterogeneous streaming services in this environment.

A set-top box is a device combining the functionality of analog cable converter boxes (tuning and descrambling) and computers (navigation, interaction, and display). Today's set-top boxes have four major components: a network interface, an MPEG decoder, graphics overlay, and a presentation engine.

- The network interface provides downstream and upstream interfaces over one or more physical connections.
- The decoder converts MPEG encoded data into audio and video. Additionally, MPEG subsystem may demultiplex application and control data from an MPEG transport stream.
- The graphics overlay provides at least one graphics plane, bitmap operations, and optional chromakey mixing.
- The preservation engine consists of a CPU, a minimum of two megabytes of memory, and a lightweight, real-time operating system. The client portion of the application runs in this subsystem. The application is controlled through the use of a simple remote control with buttons or a joystick. There is no keyboard in the basic system[23].

Recent successful deployments of IPTV-over-DSL in Europe and Asia have proven that telecom companies can successfully enter the market for television services. Last year networking giant Cisco acquired set top box manufacturer Scientific Atlanta (SA). Recently, set top box manufacturer Motorola agreed to buy Kreatel, the Swedish manufacturer of IPTV set top boxes. This combination makes for a triple play solution for carrier networks and the digital home. The medium of delivery, the Internet, has also shown itself to be capable of delivering quality video and entertainment. As a result, the digital home consumer market has rapidly grown, and both Motorola and Cisco were aware of how the STB would play a key role in the digital home consumer market. According to The Diffusion Group (TDG), a Plano-based consulting firm, these acquisitions will now give Motorola and SA a global non-IP STB space to tap into. This surely will result in increased growth in the STB market. Table 2.1 below shows the projections for STB market share numbers[24].

Table 2.1 Global IPTV Set-Top Box Forecasts.

IPTV STB Volume	2005	2006	2007	2008	2009	2010
STB Volume Top 2 Vendors(Units Shipped)	94,530	295,875	864,285	3,913,560	6,093,872	9,031,860
STB Volume for the Other Vendors (Units Shipped)	850,770	1,676,625	2,016,665	978,390	910,579	1,003,540
Market Share of Top 2 Vendors	10%	15%	30%	80%	87%	90%

Set-top box designers are being asked to support an array of new audio, video and image formats as their products evolve into more open, networked devices. IPTV set-top boxes may be enabled with the functions of personal video recorders (PVR), digital media adapters (DMA), voice over IP (VoIP), videophones and more[25]. Due to the heterogeneous nature of home based networked devices, each new device with additional functionality, layers on different requirements. IPTV and VoD depend on streaming media over a wide area network (WAN) while media applications such as PVR and DMA add a media source in a home LAN environment. Some of these applications will need to run independently and in parallel with streaming media. In order to satisfy users with a desirable experience, Set-top boxes typically require the following:

(1) receive streaming compressed audio/video over a network

(2) process the stream containers
(3) decode the streams
(4) present a synchronized audio/video output to the listener or viewer

Although, the future looks great for VO/IP, VoD, and IPTV STBs, a major challenge in the delivery of broadband is the Quality of Service (QoS) of streaming media to end-users. Here are some factors that can affect the performance of streaming media:

(1) Limited Bandwidth
(2) Server congestion
(3) Packet loss and concealment
(4) Jitter and timing drift.
(5) Variable broadband data rates,
(6) Latencies and delays impairing two-way communications,
(7) Changes in head-end video encoders producing interoperability problems with set-top decoders,
(8) Devices installed on networks the service provider doesn't control, or in parts of the network that perform below standard, subjecting it to uneven and unoptimized quality of service (QoS).

For IP video transmitted using the UDP protocol, packet loss can cause significant QoS reduction. A simple video stream can be severely degraded with low levels of packet loss, due to error propagation effects. Video quality is often represented in terms of PSNR - Peak Signal to Noise Ratio, which is a measure of the RMS(root mean square) error between the original and reconstructed video sequences. Generally a PSNR less than 20dB is regarded as unwatchable, and this level is reached for MPEG2 with a loss rate of less than 1 percent. Although all of the issues highlighted require a solution, the most important solution is the ability of the set top box to adapt to the limited available bandwidth.

Telchemy [26], a leader in VoIP and IPTV performance management, offers a lightweight software agent called VQmon/SA-VM that can be integrated into set-top boxes. VQmon/SA-VM transmits metrics back to service providers during video transmissions. The following are the feedback metrics:

(1) VSTQ score, providing data on video transmission quality

(2) VQS Score, providing an estimate of user perceived quality

Although this method provides a unique solution for management of service provider to STB transmissions, it does not provide a solution for STB to end-user link management. In this research our ActiveSTB has this unique ability. As shown earlier in Figure 1.1, an STB resides between the server and multimedia terminals, and relays the data flow from the server to the terminals and vice versa. Although the cost of the STB limits its functionality, a simple strategy implemented within the STB can improve the quality of multimedia services dramatically.

Scalable Coding and Multimedia

When the volume of multimedia data to be transmitted to terminals is too massive, some networks or terminals can not support these high rate transmissions. In multimedia, various qualities of multimedia streams can be supported in terminals using Scalable Video Coding. Scalability plays a crucial role in delivering the best possible video quality over unpredictable "best-effort" networks. Video scalability provides an application with the ability to adapt to the video quality of changing network conditions or unpredictable bandwidth variations due to heterogeneous access-technologies of the receivers [27], thus, scalability aids in content delivery to heterogeneous devices. Scalable coding provides a hierarchical coding scheme to manage multimedia streams. The transmission of scalable video involves the transmission of interdependent layers with different priorities. In scalable multimedia streaming, one stream is divided into several layers which includes one base layer and several enhancement layers. Layers in the MPEG-4 streams are handled equally in the network, even though end nodes may handle these layers differently. Data corruption during the transmission of a layer invalidates the layers with lower priorities yet to be transmitted. In this case, to prevent unnecessary assignment of the shared resources, our ActiveSTB provides early dropping of inefficient or invalidated layers. Before the ActiveSTB forwards cached multimedia data to multimedia terminals, the ActiveSTB drops a corrupted layer along with its associated lower priority layer(s). This stream reduction reduces bandwidth consumption during transmissions within the wireless network.

Note that the base layer is essential for decoding the stream, while the enhancement layers just improve the quality of the decoded stream. This hierarchical scheme can be expanded to streaming data in networks. If an intermediate node, such as a set-top box(STB), can distinguish each layer from other layers, it can apply a different strategy or assign different amounts of resources to each layer. For example, the base layer is assigned more resources, while the enhancement layers are assigned relatively less resources. Even though wired networks can provide high and stable bandwidths, fragile wireless networks are not as reliable. Therefore, for layered streaming services, it is very critical for a STB to know the available wireless network bandwidth.

CHAPTER III

IDLEGAP

Network Allocation Vector

A condition known as the hidden node problem can sometimes occur in wireless networks. The hidden node problem is when two nodes in a wireless network share the same Access Point (AP) and are unable to communicate with each other. In this situation, one node will not know whether the other node is already using the shared resource, i.e., the wireless channel. A solution to the hidden node problem involves each node using its Network Allocation Vector (NAV) that shows how long other nodes are allocated the link in the IEEE 802.11 DCF MAC protocol. The Network Allocation Vector is used within IEEE 802.11 networks to prevent Stations from accessing the wireless medium and causing contention. The NAV is an indicator maintained by each Station, of time periods when transmission will not be initiated even though the Stations CCA (Clear Channel Assessment) function does not indicate traffic on the medium. Although a node may be unreachable from other active nodes, the node can determine whether or not another node is already using the wireless network by checking its NAV. Figure 3.1 below shows the hidden node condition. Node B2 is hidden from SRC and B1 nodes. In Figure 3.1, when the sender(SRC) sends an RTS (Request To Send) to the receiver (AP), node B1 that is reachable from the sender, updates its NAV. However, node B2 does not update its NAV, because it is not reachable from sender. Only when the AP sends CTS (Clear To Send) can node B2 updates its NAV. The idle time in the wireless network can then be estimated from the NAV information.

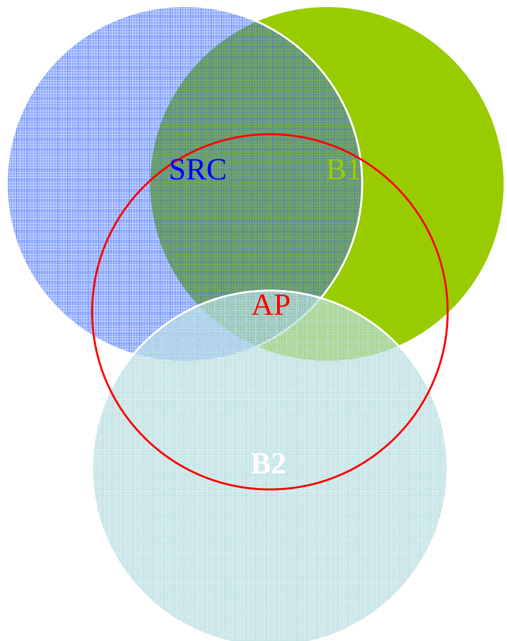

Figure 3.1 Hidden node condition. B2 is hidden from SRC and B1.

Idle Rate Estimation in Wireless Link

All nodes in a WLAN share the same resource; i.e., a wireless channel. If a node in a WLAN is utilizing this resource, any other node(s) should await the release of the wireless channel. During a transmission in a WLAN, a node can be one of the following: sender, receiver or on-looker. If a node transmits data to another node, it is sender. A node is a receiver if receiving data. Finally, when a node does not join the transmission, it is an onlooker.

The busy time of the wireless link can be estimated by adding up all the transactions of nodes in the network as depicted in Equation (3.1). Here T_{busy} is the busy time which is the sum of all the transaction times $TT(i, j)$ of the wireless link over elapsed time T_{elpsd}. $TT(i, j)$ indicates the transaction time between nodes *i* and *j* at some elapsed time T_{elpsd}.

$$T_{busy} = \frac{1}{2} \times \sum_{i=1}^{n} \sum_{j=1}^{n} TT(i, j) \tag{3.1}$$

Unfortunately, we can not determine all the transaction times from all nodes in the network. In addition, obtaining several nodes transaction information can increase network traffic, hence affecting current traffic on the network. Therefore, we propose a method to obtain all the necessary information from one node in a wireless network as follows. The transaction time $TT(i, j)$ of node *i* can be obtained via the sum of the sending time, ST_{ij}, from node *i* to node *j* when node *i* is a sender, and the receiving time, RT_{ij}, of node *i* receiving data from node *j*, when node *i* is a receiver:

$$TT(i, j) = ST_{ij} + RT_{ij} \tag{3.2}$$

During the transaction time between nodes *i* and *j*, we can get the on-looking time from a node *k's* NAV info that is updated during *i* and *j's* transactions

$$OT_k = TT(i, j) \tag{3.3}$$

where OT_k is the on-looking time at node *k* and *k* not equal to *i*,*j*. Therefore, we can estimate the busy time, T_{busy}, of the wireless link via any node *k* in the network as shown in Equation (3.4):

$$T_{busy} = ST_k + RT_k + OT_k \tag{3.4}$$

We can then obtain *Idle_rate* using the busy time T_{busy} and the total elapsed time T_{elpsd}:

$$Idle\_rate = 1 - \frac{T_{busy}}{T_{elpsd}} \tag{3.5}$$

Figure 3.2 below shows 4 nodes individual *Sender*, *Receiver*, and *On-looker* times during three transmissions from three different nodes. NAV updates the Defer Times, and the Link Busy and Idle times are used to calculate *Idle_rate*.

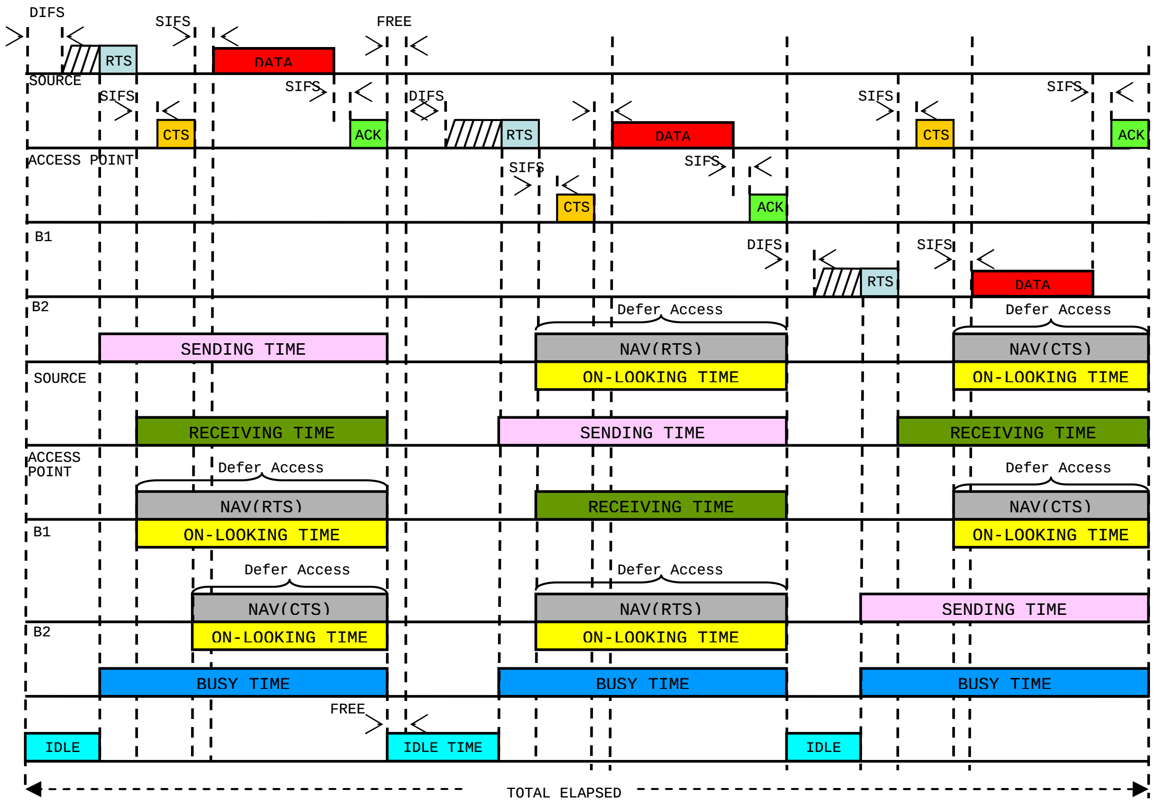

Figure 3.2 Timing diagram revealing NAV info.

System Model

We propose to add an *Idle-Module* in the MAC layer of a wireless node. This module obtains the busy time, T_{busy}, from (a) and (b) in Figure 3.3. The transaction time of a node can be obtained through accessing outgoing and incoming packets $(ST_k + RT_k)$ between the Network layer and the Link and MAC Layer shown in (b). The Idle-Module also gets the on-looking time (OT_k) from the NAV shown in (a). The updating process of the NAV triggers the *Idle-Module* to update its value. An application can access the *Idle-Module* to get the idle rate $(\boldsymbol{1 - T_{busy} / T_{elpsd}})$. Then applying the idle rate and link capacity *C* to Equation (2.1), the estimated bandwidth of the link can be calculated with minimal effort. We call this method *IdleGap* [9]:

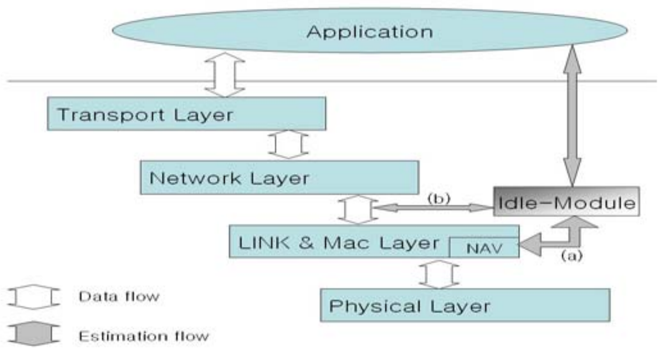

Figure 3.3 Architecture of Idle-Module

Experimental Results

To verify the performance of our *IdleGap* method, network simulations were conducted using NS-2. As shown in Figure 3.4, there are seven nodes including three wired nodes, three wireless nodes and an AP. In the wired network, the capacity of the link was set to 10Mbps, while the capacity in wireless network was set to 1Mbps.

In Figure 3.4, communication in the simulation via the AP involves three connections: Wired Node 1 to Wireless Node 2, Wired Node 2 to Wireless Node 1, and Wired Node 3 to Wireless Node 3. Wired Nodes 1 and 2 generate the cross traffic, while the algorithm generates timestamps from packets received by Wireless Node 3 via packets sent from Wired Node 3 to estimate the available bandwidth. We compare *IdleGap* to *ProbeGap* [5] and *Spruce* [3] which have been shown to out perform previous bandwidth estimations methods.

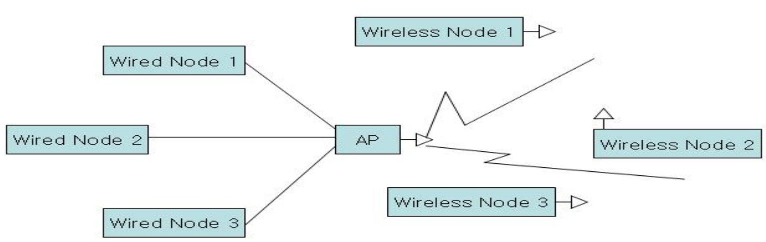

Figure 3.4 Simulation environment.

Experiments with Increasing Cross Traffic

Figure 3.5 shows the estimated available bandwidth value for each algorithm. The capacity of the wireless network in our simulation is 1 Mbps. Probing time for each algorithm is 1000 seconds and 200 probing packets are allowed. In light cross traffic, *ProbeGap* produces bandwidth estimates reflective of measured available bandwidth values. However, it shows multiple transition points over 200Kbps cross traffic. In the original *Spruce* paper, the intra-pair gap is set to the transmission time of the narrow link [3]. This causes the underestimation of the link's available bandwidth. Therefore, the intra-pair-gap was calibrated to reflect the available 1.0 Mbps with no cross traffic. Even after the calibration, *Spruce* overestimated the bandwidth severely with more than 0.5Mbps cross traffic. The reason is due to high drop rates with heavy cross traffic. Thus, the estimated bandwidth value becomes polluted and can be a contributing factor to the overestimation of the available bandwidth. The *IdleGap*, which uses NAV to estimate bandwidth, shows the closest match to the real bandwidth. Note that after 0.6Mbps cross traffic, saturation occurs due to the overhead of the wireless network such as defer time and RTS/CTS.

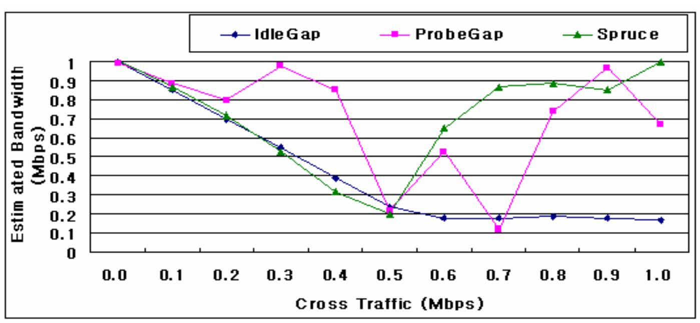

Figure 3.5 Estimated bandwidth with cross traffic.

Experiment with Different Observation Times

In this experiment, we vary the observation time to estimate the available bandwidth. Since we focus on the results during the observation period, the cross traffic is set to 10Kbps, where all three schemes are able to estimate the bandwidth accurately as shown in Figure 3.6. *ProbeGap* and *Spruce* send the probes at intervals of 5 seconds [5]. Figure 3.6 shows the estimated values of the available bandwidth for *ProbeGap*, *Spruce* and *IdleGap* between observation periods of 10 and 500 seconds. Until 250 seconds, *ProbeGap* and *Spruce* record values not reflective of measured available bandwidth. After 250 seconds, *ProbeGap* and *Spruce* values are near the measured bandwidth values. However, *IdleGap* generates values reflective of measured bandwidth for all periods. Therefore, we can conclude that *IdleGap* provides accurate estimations with short observation times.

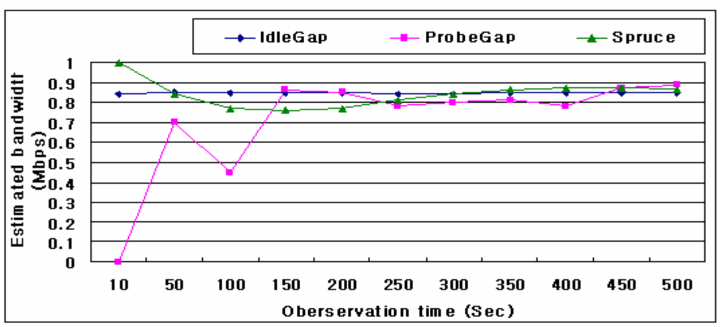

Figure 3.6 Estimated bandwidth with different observation times.

CHAPTER IV

ACTIVESTB

Transmission Management

The multimedia stream is composed of GoVs (Group of Video Objects), each of which is normally composed of several VoPs (Video Object Planes) or frames and each frame is divided into layers. Each frame size varies which means each GoV size varies; therefore, variable bit rate (VBR) stream transmissions would be much more efficient than constant bit rate (CBR) stream transmissions. Another reason why CBR is less efficient than VBR is because packets transmitted via CBR can arrive late at end-user decoder since transmissions are not based on the GoV interval. The GoV interval is derived from decoding time stamps in the frames which make up the GoV. Therefore, having varied frame sizes yields varied or variable GoV sizes over an interval, hence VBR transmission. Since VBR is based on the frame rate, we sometimes refer to VBR as frame rate. The GoV or *gov* interval is normally equal to the number of frames, *n*, minus 1 times the frame rate. This is the case because each frame needs to arrive at end-user's decoder prior to the frames DTS. We determine the actual GoV interval from the cached GoV stream data as shown below.

```
pkt = 1st pkt in cached GoV(g)
do this while pkt != null
        if start dts < 0 || > pkt dts
          then start dts = pkt dts
        if end dts < pkt dts || end dts < 0
           end dts = pkt dts
        pkt = next pkt
gov_interval = end dts – start dts
start dts = end dts
g = g + 1
gov =  GoV(g)
```

Once the constant GoV(g) interval, *gov_interval*, is determined, GoV(g)'s packet transmission coefficient, *trnsmssn_coeff*, is calculated in units of seconds per byte. The transmission time, *trnsmssn_time*, for each packet(p) of GoV(g) is then calculated as shown below:

gov_size = size of GoV(g)
trnsmssn_coeff = gov_interval / gov_size
trnsmssn_time(packet(p)) = size of packet(p) x *trnsmssn_coeff*

ActiveSTB acts as a proxy server or gateway to downstream heterogeneous clients. ActiveSTB performs quality-adaptation to changes in network bandwidth, while simultaneously forwarding the partially cached stream allowing a user to view quality playback of video. The ActiveSTB stores the stream in layers which aids in reducing the packet loss rate when transmitting the layer encoded stream at variable bit rates. ActiveSTB is designed to efficiently manage the wireless transmission of the multimedia stream by (1) extracting information from buffered data, (2) *Early Dropping*, (3) estimating the available bandwidth, and (4) transmitting the stream at a variable bit rate (VBR). Today's STBs transmit multimedia stream data at a constant bit rates (CBR) irregardless of stream size and without bandwidth estimation. As mentioned earlier, CBR is less efficient than VBR because packets can arrive late at end-user decoder since transmissions are not based on the GoV interval which is determined from decoding time stamps in the frames which make up a GoV. Additionally, the number of bytes sent at the constant bit rate could exceed the available bandwidth. This could also happen when using VBR in our ActiveSTB, however, our ActiveSTB adjusts the GoV size to the max size allowable for the available bandwidth. Since we know the available bandwidth (*abw*) and the *gov_interval*, we can calculate the max stream size that can be transmitted over the duration of the *gov_interval* at this *abw*. See calculations below:

$$abw \leftarrow Idle\_Gap$$

$$abw = \frac{\max\_bytes}{gov\_interval} \times \frac{Mbits}{125000\,bytes}$$

$$\max\_bytes = abw \times gov\_interval \times \frac{125000\,bytes}{Mbits}$$

If the *gov_*size exceeds *max_bytes*, we repeatedly adjust or adapt the GoV size to the available bandwidth by removing layers from individual frames within the GoV. We call this process *quality adaptation* to the available wireless links bandwidth.

ActiveSTB acts as a proxy server or gateway to downstream heterogeneous clients. ActiveSTB performs quality-adaptation to changes in network bandwidth, while simultaneously forwarding the partially cached stream allowing a user to view quality playback of video. The ActiveSTB stores the stream in layers which aids in reducing the packet loss rate when transmitting the layer encoded stream at variable bit rates. Loss in layered streaming service has two classifications: *Indirect Loss* and *Direct Loss*. *Direct Loss* is when layered data is not transmitted successfully from server to STB, while *Indirect Loss* is the removal of data or layers corrupted by *Direct Loss* from the extracted cached stream. The *Indirect Loss* process of removing corrupted data from the ActiveSTB cache is called *Early Dropping*. This enables our ActiveSTB to efficiently manage wireless link during the transmission of layers. *Early Dropping* in the ActiveSTB saves cache space and reduces client-side *Indirect Loss* in wireless networks. The ActiveSTB obtains the available bandwidth estimate via ACK responses from mobile nodes containing the available bandwidth estimate. The mobile nodes execute the lightweight *IdleGap* software module to calculate the bandwidth estimate. If the stream size is greater than the max stream size calculated over the stream interval using the bandwidth estimate, layers in each frame of the stream are removed until stream size is less than or equal to the max stream size. This reduction of the stream to be transmitted reduces the consumption of the available bandwidth. Thus, the ActiveSTB's execution of "Early Dropping" and bandwidth estimation will be shown to reduce client-side *Indirect Loss*, wireless link bandwidth consumption, and improves QoS to end-user.

Early Dropping

Before transmitting a cached layer to a client, the ActiveSTB validates the desired layer, and if invalid, drops or removes layer or layers from cache. Table 4.1 below shows the variables used for stream packet transmission and validation equations.

Table 4.1 Packet transmission and validation calculation variables.

G	*multimedia stream of 'g' GoV's.*
L_g	*layer(l) at GoV(g)*
$P_{g,l,s}$	*The s^{th} packet of layer(l) in GoV(g)*
$NG(G)$	*Number of GoV's in multimedia stream.*
NL_g	*Number of layers in GoV(g)*
$NP_{g,l}$	*Number of packets in layer(l) in GoV(g)*
PLR	*Packet Loss Rate*
$Size(l)$	*size of layer(l).*

In Equation (4.1) below, the server divides multimedia data at *layer(l)* in GoV(g) into several packets for transmitting:

$$L_g = \bigcup_{i=0}^{NP_{g\,l}} P_{g\,l\,i} \tag{4.1}$$

When a layer is complete with all available packets, the multimedia terminal can then decode this data. The loss of one packet within a VoP(Video of Pictures) or Frame causes other packets in the same layer and higher to be discarded. This is "Early Dropping". The safe transmission of all packets in a *layer(l)* ensures that the multimedia data at *layer(l)* is valid for decoding in Equation (4.2):

$$Valid(L_g) = \prod_{i=0}^{NP_{g\,l}} (1 - PLR(P_{g\,l\,i})) \tag{4.2}$$

Scalable multimedia has a hierarchical structure: an enhancement layer requires a lower layer including base layer for decoding scalable multimedia as shown in Equation (4.3):

$$Valid(L_g)\ for\ decoding = \prod_{l=0}^{L_g} \prod_{i=0}^{NP_{g\,l}} (1 - PLR(P_{g\,l\,i})) \tag{4.3}$$

To reduce wireless channel consumption, we can filter out incomplete layers before transmitting data over the shared wireless channel as shown in Equation (4.5), while complete layers are transmitted to multimedia terminal through wireless channel as shown in Equation (4.4):

$$Complete(NG(G)) = \sum_{g=0}^{NG(G)} \left(\sum_{L=0}^{NL_g} \left(Size(L) \times \prod_{l=0}^{L} \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli})) \right) \right) \tag{4.4}$$

$$Filtered(NG(G)) = \sum_{g=0}^{NG(G)} \left(\sum_{L=0}^{NL_g} Size(L) \times \left(1 - \prod_{l=0}^{L} \prod_{i=0}^{NP_{gl}} (1 - PLR(P_{gli})) \right) \right) \tag{4.5}$$

In Figure 4.1, each GoV in the stream contains 4 layers which include 1 base layer and three enhancement layers. The STB notices that third layer in GoV G1 is incomplete, so both third and forth layer are *early dropped* and not forwarded to client. This method of early dropping saves on bandwidth consumption by reducing the amount of bad or useless data transmitted to client.

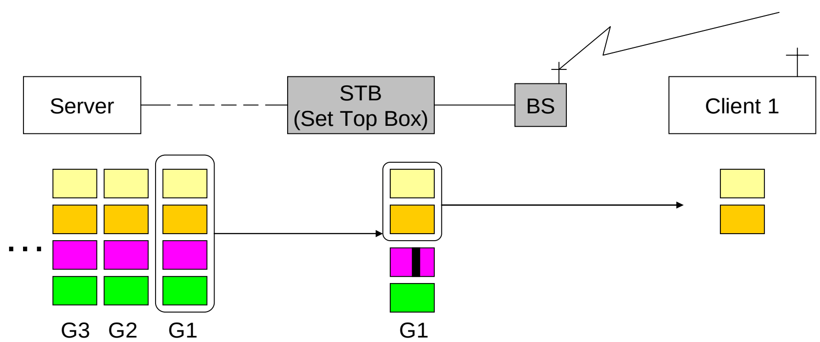

Figure 4.1 Early dropping scheme.

Implementation

The STB should decide how much data should be cached for each stream and how much data for each stream to be transmitted to multimedia terminals via the wireless channel. If the STB only handles homogeneous connections, a management scheme based on the size of cached data may be the best choice. However, since the STB handles heterogeneous connections with different bit rates and qualities, we suggest an active STB that extracts DTS or *dts* (decoding time stamp) information from the cached multimedia stream to deal with different bit rates and qualities.

In our simulation, 33ms is the duration of a frame. Tests will show that frame rate usage in the ActiveSTB, as expected, is much more efficient than the constant bit rate transmissions used by the *Basic* and *Enhanced Basic* STB. The nomenclature for tests results recorded in the figures and configuration settings associated with all STBs are defined in Tables 4.2 and 4.3. The *EnhancedSTB* was created to show how our ActiveSTB not only out performs a *BasicSTB*, but also an enhanced STB. FM is used in conjunction with bandwidth estimation; however, no bandwidth estimation is used when CBR is used.

Table 4.2 Measurement definitions.

Measurements	
Y	Enabled
N	Disabled
ED	Early Drop
FM	Frame Rate
CBR	Constant Bit Rate

Table 4.3 Set-top box measurement settings.

ActiveSTB	***EnhancedSTB***	***BasicSTB***
ED(Y)_FM	ED(Y)_CBR	ED(N)_CBR
	ED(N)_FM	

In our tests, both the *BasicSTB* and *EnhancedSTB* cache a portion of the stream; however, *BasicSTB* uses CBR and does not early drop, extract *dts* info, or estimate the available bandwidth (EABW). *EnhancedSTB* either does ED with CBR and no bandwidth estimation (BWE) or does no ED with Frame Rate which uses BWE. Our results will show how ActiveSTB outperform both Basic and Enhanced STBs.

Network Simulation

Currently, we have a technique for dividing and merging of MPEG-4 streams. In Figure 4.2, the Divider divides the original stream into several layers, and then logs the size and decoding time information of each layer in terms of GoP (Group of Picture). The NS-2 network simulation is conducted using the log file generated by the Divider, and the results of successful or unsuccessful layered streaming transmissions are generated and analyzed. The Merger merges the streamed multimedia data from the NS-2 results into an MPEG-4 stream.

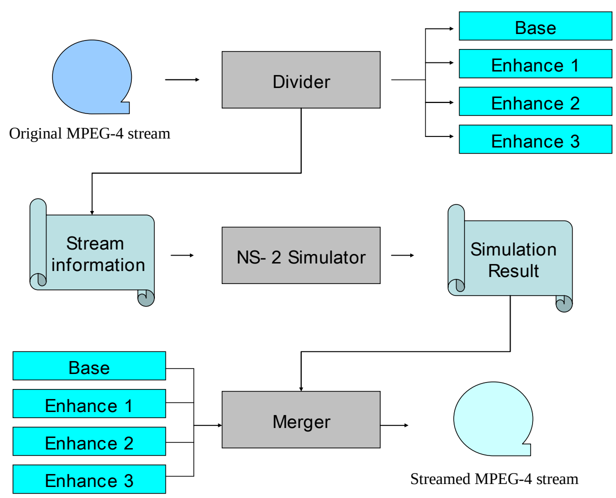

Figure 4.2 Simulation diagram.

Figure 4.3 shows the NS-2 network simulation scenario and the ActiveSTB modules *validate*, *adjust stream*, and *get abw*; note *abw* means available bandwidth. The modules are shown in the flow chart in Figure 4.4. In Figure 4.3, we simultaneously begin transmission of the layer encoded video and injection of cross traffic into LAN from two sources. We also induce *Direct Loss* in ActiveSTB by varying the packet loss rate during transmission. This process causes *Indirect Loss* in the ActiveSTB and *Indirect Loss* packets are not forwarded to client. We induce *Direct Loss* at the client by increasing the cross traffic in the link. This of course causes *Indirect Loss* to occur at the client also. For the simulation, cross traffic values ranged from 0 to 0.8 Mbps, and error rates or packet loss ranged from 0 to 25%.

The performance of our ActiveSTB and the basic and enhanced STBs in various scenarios is verified using NS-2. Our results will compare the original stream size to the decoded data at the end-user. We recorded and show results for *Indirect Loss*, packets decoded, latency, and Communication Efficiency. These results reveal how ActiveSTB reduces excess bandwidth consumption.

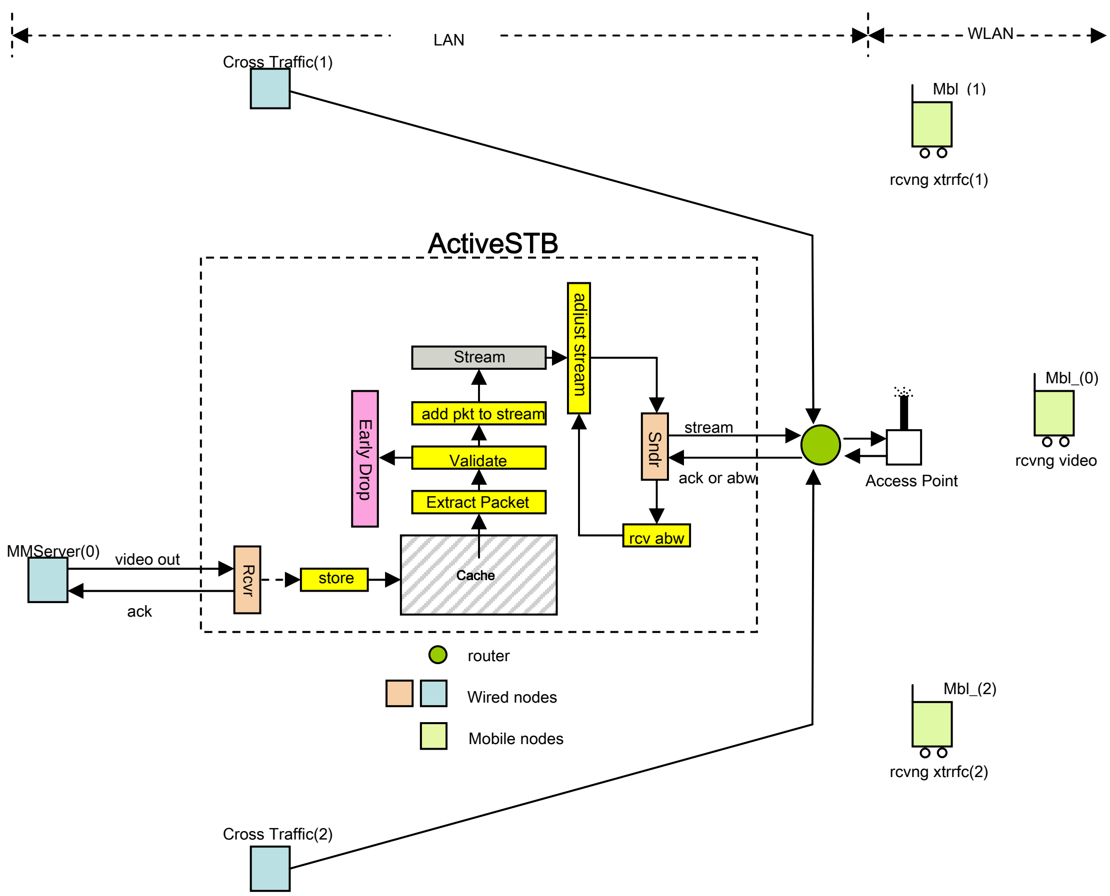

Figure 4.3 NS-2 Simulation scenario and ActiveSTB modules

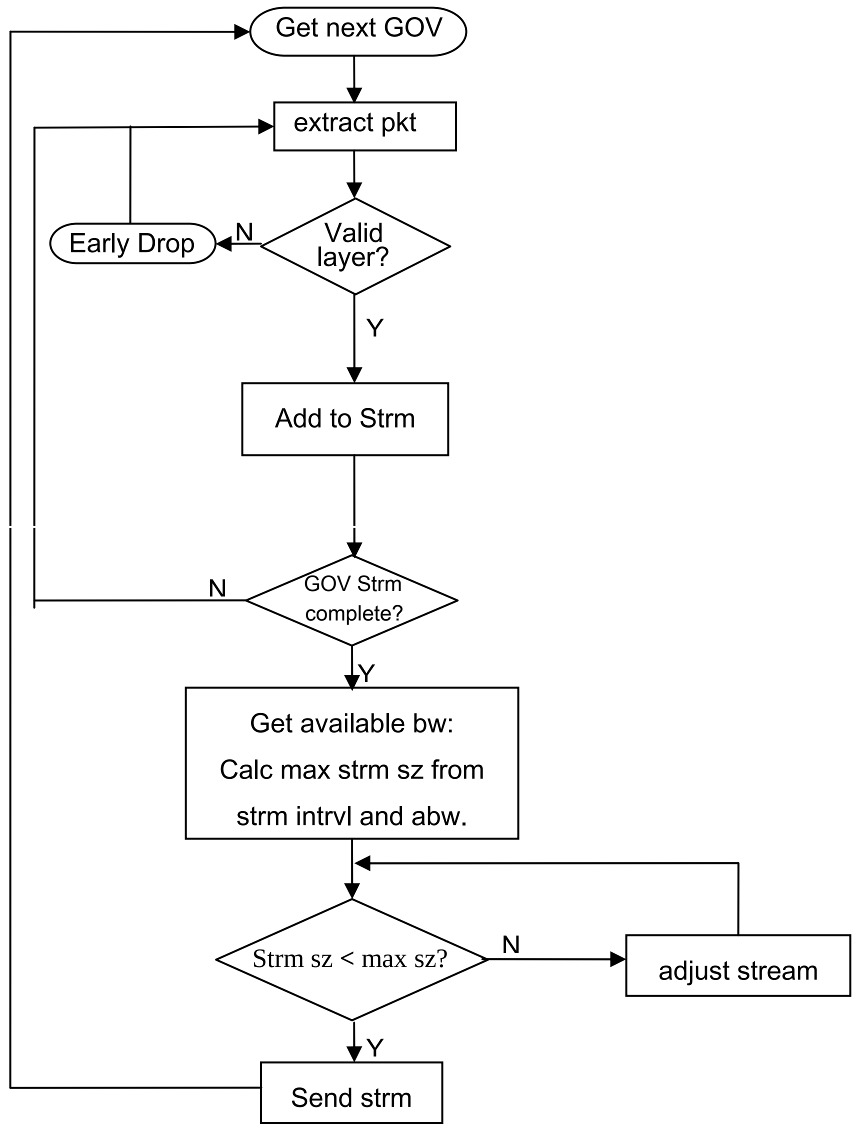

Figure 4.4 ActiveSTB flow chart.

Experimental Results

The figures in the following sections compare our ActiveSTB to a basic STB (*BasicSTB*), and an enhanced basic STB (*EnhancedSTB*). As previously mentioned, our ActiveSTB not only caches data, but performs link management by discarding (early dropping) useless data, using VBR transmissions, and by estimating the available bandwidth prior to forwarding the partially cached stream. As discussed earlier, since data is layer encoded and stored in frames, our ActiveSTB forwards the stream using the frame rate.

Packets Decoded with Increasing Cross Traffic and Fixed Error Rates

In this section, we will highlight how ActiveSTB compared to current STBs, decodes more data as the cross traffic increases. Figures 4.5 and 4.6 are results for a small 363K byte file named Suzie. Figures 4.7 and 4.8 are results for a larger 1.3M byte file named Foreman. Figures 4.7 and 4.8 reveals our ActiveSTB yielding more efficient and higher performance transmissions during high and low cross traffic of the larger Foreman file video compared to *BasicSTB* and *EnhancedSTB* (ref Table 4.2 and 4.3). The ActiveSTBs performance for the smaller Suzie file as shown in Figures 4.5 and 4.6 begin to outperform the other STBs only after exceeding 0.25M bits of cross traffic, whereas the ActiveSTBs performance for the larger Foreman file as shown Figures 4.7 and 4.8, outperforms the other STBs at all levels. Since in the real world most users tend to view larger video file sizes, the ActiveSTB is perfect solution for realtime environments. Thus, the remaining sections will only have results from the larger Foreman file tests.

Figures 4.5 through 4.8 reveal how our ActiveSTB measurements consistently outperform the *EnhancedSTBs* and the *BasicSTB* as the cross traffic increases. The figures for different Error Rates show the number of packets decoded in the end-user's decoder. The higher the number of packets decoded, the better the quality of the streamed data to user. Higher quality is associated with transmission of higher layers and little to no transmission of useless data thereby reducing *Indirect Loss*. The max link bandwidth is 1Mbps. The results also reveal that as the link becomes heavily congested when the cross traffic exceeds 0.75 Mbps, all STBs decode nearly the same number of packets. This is due to the fact that the amount of data decoded during heavy congestion for any scheme is very low and the viewing quality is also very low. Nevertheless, even with a 10% packet loss rate of data sent to the ActiveSTB, Figure 4.8 shows the ActiveSTB outperforming the other STBs. Along with early dropping of layer or layers due to packet loss, the ActiveSTB also reduces the stream size based on the ABW. On the other hand, the *BasicSTB* further congests the link by not early dropping and not adjusting the stream size to the ABW. The *EnhancedSTB* ED(N)-FM setting sends a higher number packets too because it doesn't early drop, and so does *EnhancedSTB* ED(Y)_CBR setting because it uses CBR. Hence, more packets are sent to receiver, but due to congestion many are discarded as *Indirect Loss* and the number decoded is small. However, during heavy congestion the ActiveSTB drastically reduces the stream size prior to sending, so very little *Indirect Loss* occurs and due to very small stream size, the quality of the data received may still be very low. Due to the use of BWE and ED in our ActiveSTB as shown in Figures 4.5 through 4.8, during 0.4 to 0.8Mbits of cross traffic, our ActiveSTB decoded an average of 32% to 56% more packets than the basic or enhanced STB. Figures 4.9 and 4.10 on page show the average performance improvement of ActiveSTB compared to all other STBs. Later, I will show how the ActiveSTB's active link management will result in low end-user decoder *Indirect Loss* and Latency.

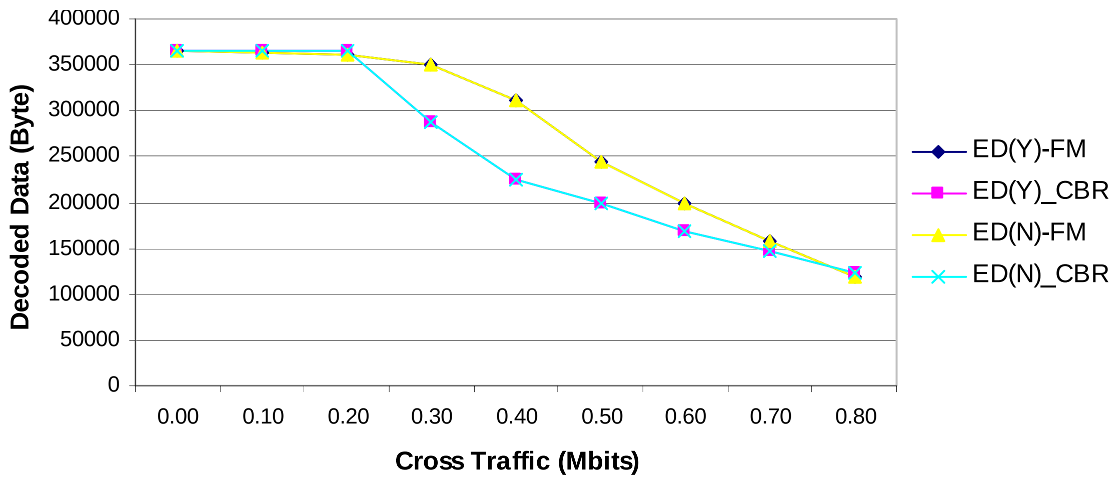

Figure 4. 5 Suzie video file(363K). Packets decoded vs cross traffic, 0% error rate.

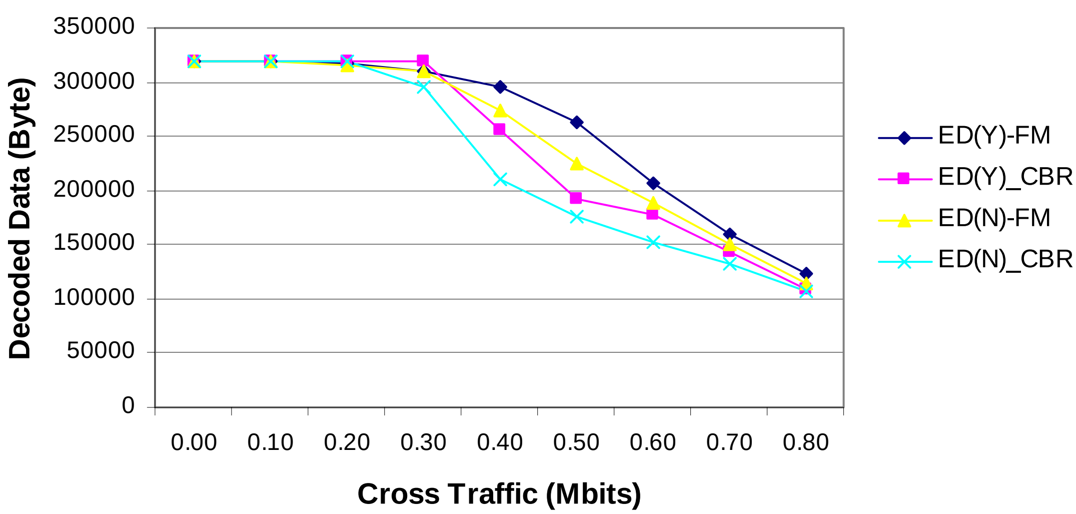

Figure 4. 6 Suzie video file(365K). Packets decoded vs cross traffic, 5% error rate.

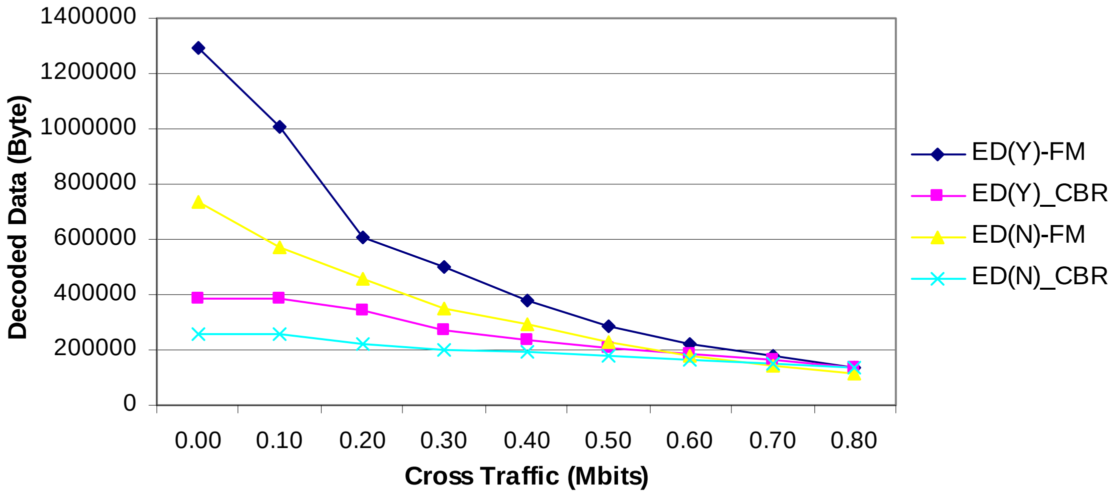

Figure 4. 7 Foreman video file(1.3M). Packets decoded vs cross traffic, 5% error rate.

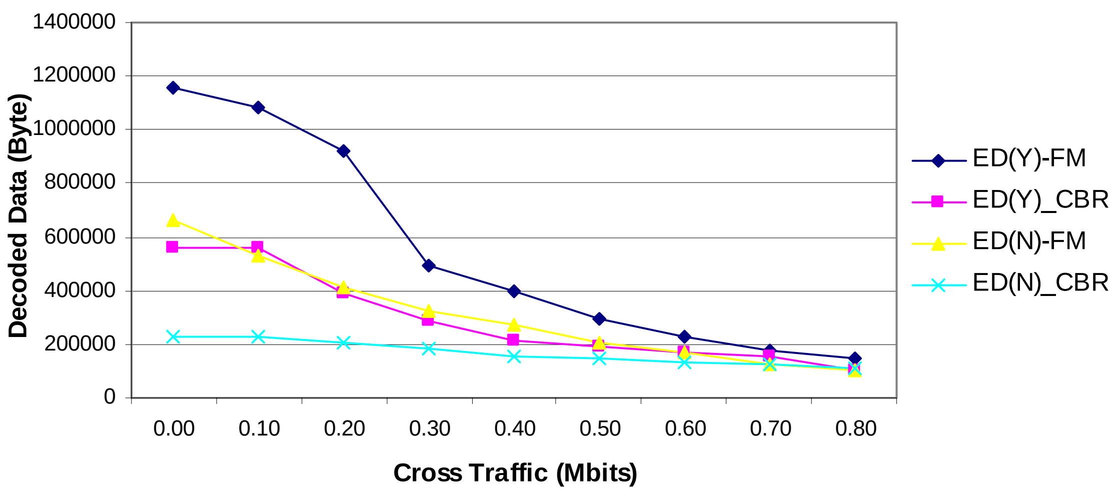

Figure 4. 8 Foreman video file(1.3M). Packets decoded vs cross traffic, 10% error rate.

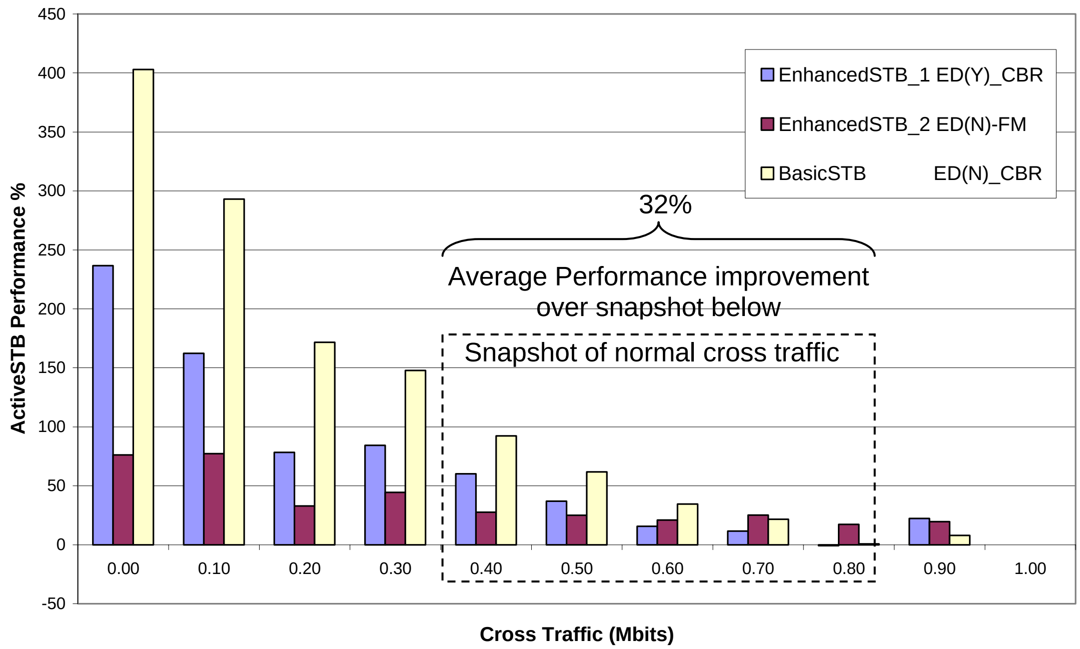

Figure 4. 9 ActiveSTB performance during 0.4 to 0.8Mbits cross traffic, 5% error rate.

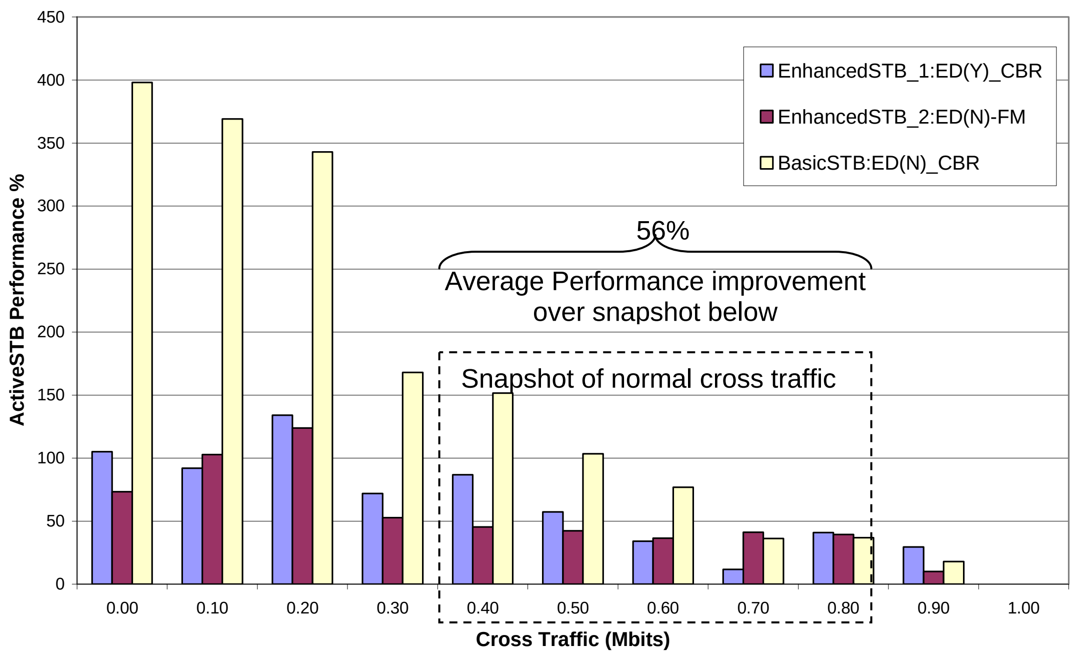

Figure 4. 10 ActiveSTB performance during 0.4 to 0.8Mbits cross traffic, 10% error rate

Indirect Loss with Increasing Cross Traffic and Error Rates

In this section, we will demonstrate how ActiveSTB decreases the amount of *Indirect Loss* at the end-user as cross traffic increases. The decrease in *Indirect Loss* reduces bandwidth consumption, end-user memory consumption and processing and decoding time. As mentioned earlier, *Indirect Loss* occurs at the ActiveSTB or in the end-user's decoder when requisite packets are lost during transmission. *Indirect Loss* is useless transmitted packets. Since our ActiveSTB performs early dropping and bandwidth estimation which prevents transmission of useless data, one can see in Figures 4.11 and 4.12 the relatively low to negligible amount of *Indirect Loss* occurring at end-user's decoder for ActiveSTB. Although Frame Rate (FM) is superior to Constant Bit Rate (CBR), in the two figures, ED(Y)-FM and ED(Y)_CBR have similar low *Indirect Loss*, because both perform early dropping. The ActiveSTB gains revealed in Figures 4.11 and 4.12 are in the savings in memory and bandwidth consumption. Bandwidth is saved due to a very small amount of *Indirect Loss* occurring, and memory is saved because the useless packets are not stored.

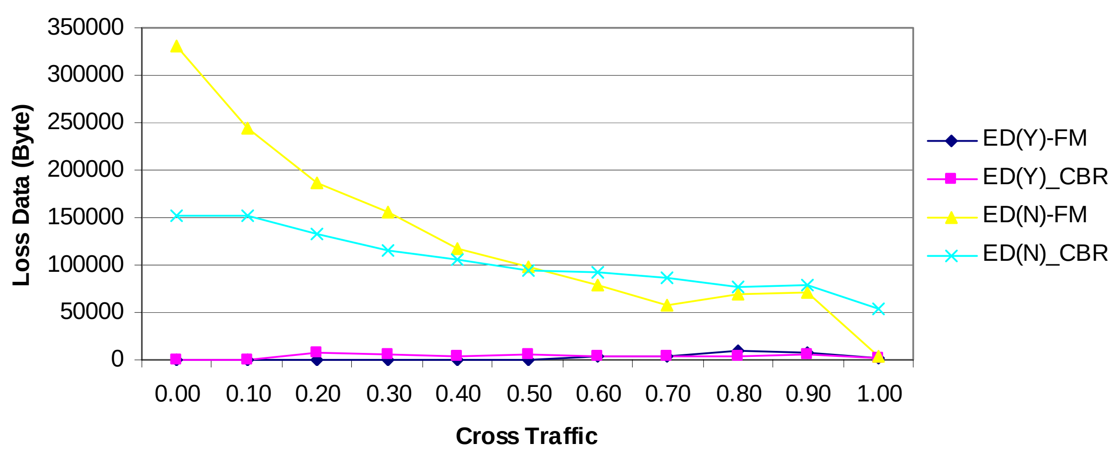

Figure 4. 11 *Indirect Loss* vs cross traffic, 15% error rate.

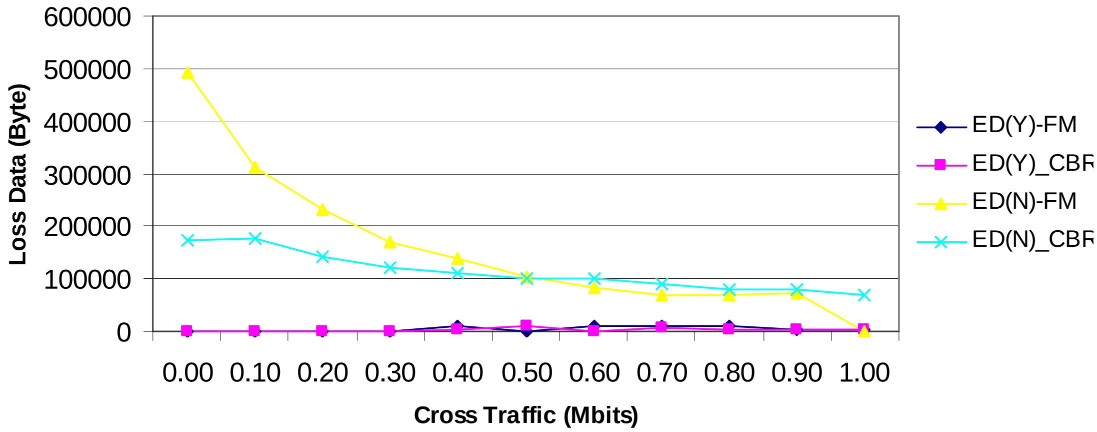

Figure 4. 12 *Indirect Loss* vs cross traffic, 20% error rate.

Notice in Figure 4.11 with an error rate of 15%, indirect loss ranges from 0 to 300,000 bytes. The range in Figure 4.12 is from 0 to 500,000 bytes. As cross traffic increases, these values gradually decrease, however, the fact that higher amounts of indirect loss are occurring in the other STB settings of ED(N)-FM and ED(N)_CBR, reveals the higher amounts of useless data transmitted that unnecessarily consume bandwidth. Since little to no indirect loss is occurring in ActiveSTB, latency or delays in packet transmissions is automatically reduced. In the next section, we will discuss ActiveSTBs affect on latency.

Latency with Increasing Cross Traffic and Fixed Error Rates

This section reveals how the ActiveSTB reduces latency. The Latency results below indicate the average one-way packet transmission delay time over the entire duration of the stream at increasing levels of cross traffic. Latency increases as the link becomes more and more congested with cross traffic or when the rate of the transmission of the stream exceeds the link rate. Figures 4.13 and 4.15 show measurements recorded for Latency as cross traffic increases at fixed error rate settings. The results below confirm low latency results for the ActiveSTB due to use of ED, BW estimation, and Frame Rate(same as Variable Bit Rate) transmissions. Low *Indirect Loss* indicates that most packets received are decoded. On the other hand, higher *Indirect Loss* means most packets received are discarded. Since ED and BW estimation reduces the stream size when necessary, latency is decreased. The results below also show how the usage of Frame Rate(or VBR) outperforms CBR. Figure 4.13 shows measurements recorded for an error rate of 15%. Figure 4.12 shows measurements taken during a higher error rate or packet loss rate of 25% during video transmissions to our ActiveSTB(ED(Y)-FM). Here, *Early Dropping* significantly reduces latency for not only Frame Rate, but also for the less efficient CBR transmissions. Notice in Figure 4.13, the 2nd best performer is *EnhancedSTB_2*(ED(N)-FM), but then notice in Figure 4.14 with the higher error rate how *EnhancedSTB_2*(ED(N)-FM), becomes the 3rd best performer replaced by *EnhancedSTB_1*(ED(Y)-CBR),. This coincides with the earlier observation revealing how when the error increases, *early dropping* becomes more and more important.

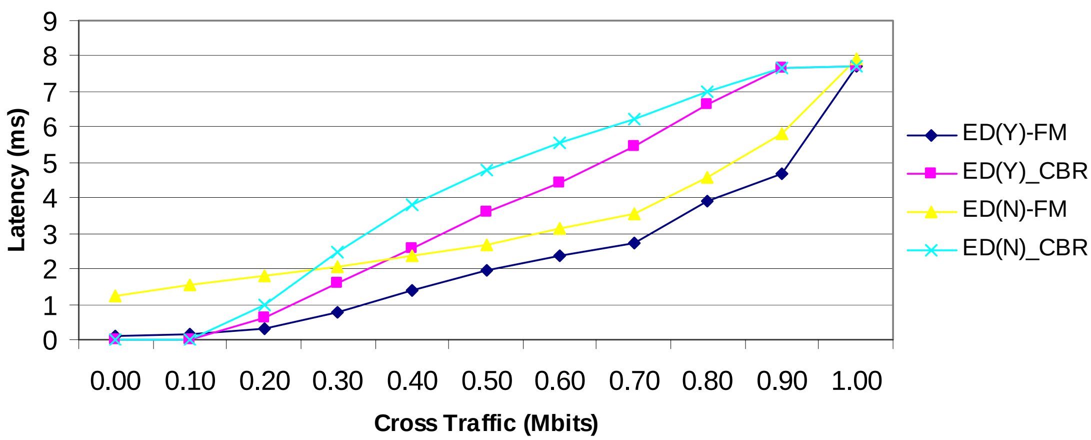

Figure 4. 13 Latency vs cross traffic, 15% error rate.

Notice in Figure 4.14 how *EnhancedSTB_1*(ED(Y)-CBR) outperforms *EnhancedSTB_2*(ED(N)-FM). This reveals that CBR with early dropping and no bandwidth estimation is better than FM with bandwidth estimation and no early dropping. This confirms the importance of early dropping and ActiveSTB's excellent performance when combining early dropping with bandwidth estimation. Finally, in Figure 4.14 when we average the measurements acquired during link crosstraffic of 0.4 to 0.8Mbps, ActiveSTB's average performance improvement compared to

EnhancedSTB_1(ED(Y)-CBR) is 78%. This means ActiveSTB(ED(Y)-FM) has even greater performance compared to the remaining STBs since *EnhancedSTB_1* has the 2nd best performance as shown in Figure 4.14.

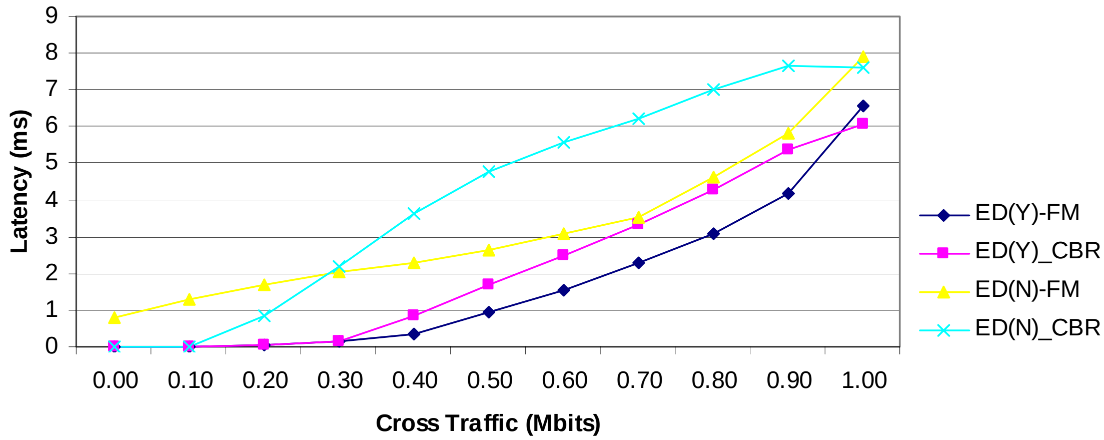

Figure 4. 14 Latency vs cross traffic, 25% error rate.

Communication Efficiency

This section shows how our ActiveSTB improves the communication efficiency to the end-user. Figures 4.15 and 4.16 display the ratio of the decoded data to sent data measurements taken over increasing cross traffic. Data is sent from the Multimedia Server and received at end-user's decoder. Recall ED means early drop, FM means frame rate, and CBR means constant bit rate. The results below again confirm the benefits of early dropping when using Frame Rate(FM) or CBR. Notice in Figure 4.15 when the error rate is low, 5%, the FM settings with and without ED, ED(Y)-FM and ED(N)-FM, outperform the CBR settings with and without ED. During higher packet loss rates as shown in Figure 4.16, the opposite occurs: both the ED(Y)-FM and ED(Y)_CBR outperform ED(N)-FM and ED(N)_CBR revealing again the significance of early dropping. This observation reveals how the use of bandwidth estimation and FM, with or without ED, produces better results than CBR during low packet loss rates, however, as packet loss rates increase due to congestion, early dropping enables both FM and CBR to perform better than the others. The best performer over all increasing packet loss rates and cross traffic is the ActiveSTB which is ED(Y)-FM. This shows our ActiveSTB consistently delivers better quality data than an enhanced or basic STB over increasing error rates and cross traffic. Notice how during the higher error rate of 15% as shown Figure 4.16, if we average the measurements taken during link crosstraffic of 0.4 to 0.8 Mbps, the ActiveSTB(ED(Y)-FM) ratio of decoded data to ActiveSTB sent data measurements outperform the *EnhancedSTBs* (ED(Y)_CBR, ED(N)-FM) by an average of 95% and outperforms the *BasicSTB* (ED(N)_CBR) by and average of 464%.

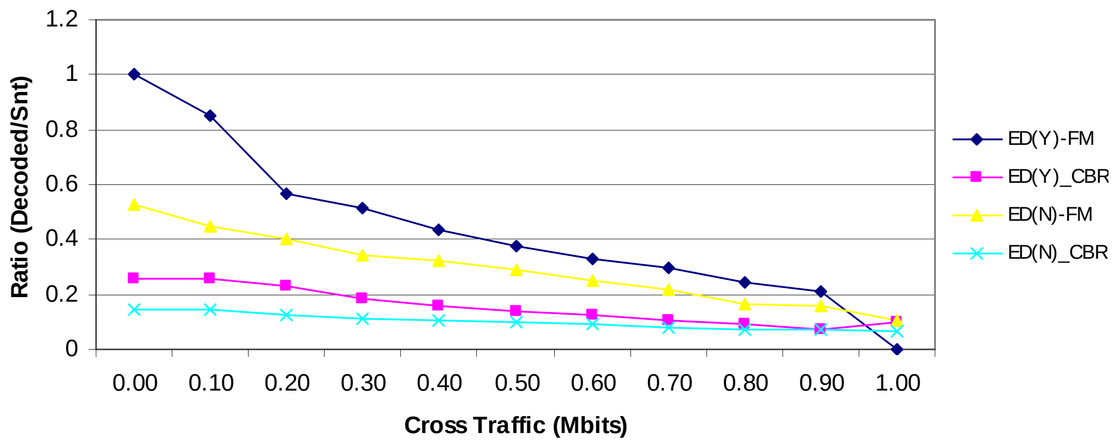

Figure 4. 15 Ratio vs cross traffic, 5% error rate.

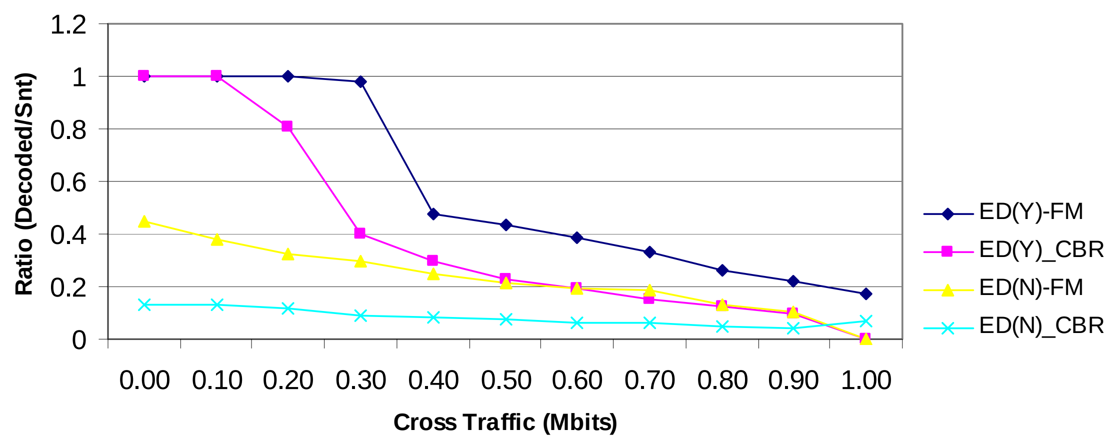

Figure 4. 16 Ratio vs cross traffic, 15% error rate.

CHAPTER V

SUMMARY AND CONCLUSION

Summary

As discussed previously, the internet and wireless home networks have undergone rapid growth which has led to an increase in streaming video. This increase has necessitated attention to bandwidth efficiency of service to end-user. Thus, optimal link management is a requisite for bandwidth efficiency. The most challenging aspect of multimedia streaming services is the adapting the bit rate of multimedia stream according to the network status; therefore we presented a method, *IdleGap*, to estimate the available bandwidth of a wireless link. *IdleGap* is a lightweight software module that is easily implemented in the mac layer of a wireless node. *IdleGap* quickly estimated the link's bandwidth by accessing busy and idle time info from the node's NAV info in the mac layer. *IdleGap* was shown to (a) be applicable to real-time applications such as multimedia streaming services, (b) be simple and effective in estimating the available bandwidth and (c) incur low overhead. The simulation result revealed how *IdleGap* outperformed other probing and bandwidth estimation methods like *ProbeGap* and *Spruce*. Our results also revealed how IdleGap bandwidth estimates closely matched the real bandwidth and yielded estimates with minimal delay time. Hence, we showed how *IdleGap* when implemented in our newly presented ActiveSTB, improved the quality of the streamed media to client.

Conclusion

In this research we have also addressed and resolved several issues involved with streaming media in home networks via STBs. Traditional STBs provide user interaction, forward data to be viewed; however, they do not buffer part of the stream, *early drop*, or estimate the available bandwidth. In a wireless home environment with multiple wireless peripherals, proper forwarding of data to devices over limited bandwidth is key in quality viewing of data. We considered an improvement of data streamed to wireless home environments by focusing on the bottleneck in the link which is an STB. We designed an ActiveSTB to consider and overcome the challenges of ineffective usage of the shared wireless channel, reducing latency, and improving bandwidth efficiency. Our results demonstrated our ActiveSTB overcoming these issues by its use of extracting layer info from cached data, early dropping and bandwidth estimation via *IdleGap*. Measurements recorded reveal our ActiveSTB improves bandwidth efficiency of the layer encoded stream sent to the end-user and contributed to 32 to 56% more packets being decoded at the end-user than traditional STBS. Our ActiveSTB efficiently utilized the available bandwidth by removing corrupted data from the stream and adjusting the stream size to the available bandwidth. These results were shown to improve the quality of the stream sent to end-user over increasing cross traffic and packet loss rates. Latency in the transmission due to cross traffic was drastically reduced, and the reduction in the transmission of corrupted data reduced excess bandwidth consumption. Based on results, we believe the methods implemented in our ActiveSTB module will greatly enhance the quality of data streamed to end-user, thus contributing to increased wireless home network usage and an increasing growth in the STB market.

REFERENCES

[1] R. Rejaie, Y. Haobo, M. Handley, D. Estrin, “Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet**,**” in *Proc. of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, 26-30 March 2000 pp. 980 - 989.

[2] G. Rubino, M. Varela and J. Bonnin, “Controlling multimedia QoS in the future home network using the PSQA metric,” *The Computer Journal*, Vol. 49(2):137-155, 2006

[3] J. Strauss, D. Katabi, and F. Kaashoek, “A measurement study of available bandwidth estimation tools,” in *Internet Measurement Conference*, ACM Press, pp. 39 – 44, 2003.

[4] M. Jain and C. Dovrolis. “Pathload: A measurement tool for end-to-end available bandwidth.” in *Proc. of Passive and Active Measurements Workshop*, (Fort Collins, CO), 2002, pp. 14 – 25.

[5] K. Lakshiminarayanan, V. N. Padmanabhan, and J. Padhye, “Bandwidth estimation in broadband access networks,” in *Internet Measurement Conference*, ACM Press, pp. 314 – 321, 2004.

[6] R. Pulles, P. Sasno, “Demonstrations: A set top box combining MHP and MPEG-4 interactivity,” in *Proc. of the 2nd European Union Symposium on Ambient Intelligence EUSAI '04*. November 2004. Endhoven, Netherlands, ACM Press, pp 31 – 34.

[7] F.T.H. den Hartog, M. Balm, C.M. de Jong, J.J.B. Kwaaitaal. "Convergence of residential gateway technology: analysis of evolutionary paths," in *IEEE Consumer Communications and Networking Conference*, pp.1-6, Jan. 2004.

[8] W. T. Ng and H. A. Chan. "Streaming multimedia content over home network with an intelligent controller,” in *The 29th Annual Conf. of the IEEE Industrial Electronics Society*, Vol. 2, Nov. 2003, pp. 1802-1807.

[9] H. K. Lee, V. Hall, K. H. Yum, K. I. Kim and E. J. Kim "Bandwidth estimation in wireless LANs for multimedia streaming services," in *Proceedings of the 2006 International Conference on Multimedia & Expo (ICME)*, Canada 2006.

[10] R. L. Carter and M. E. Crovella, “Dynamic server selection using bandwidth probing in wide-area networks,” Technical Report TR-96-007, Boston University Computer Science Department, 1996.

[11] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth techniques," in Proceedings of *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, Vol. 21(2), 2003, pp. 879 – 894.

[12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Internet Measurement Conference*, 2003, pp. 39 – 44.

[13] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Global Telecommunications Conference*, Vol. 1, 27 Nov.-1 Dec. 2000 pp. 415 – 420.

[14] M. Jain and C. Dovrolis. "Pathload: A measurement tool for end-to-end available bandwidth." in *Passive and Active Measurements(PAM) Workshop*, Fort Collins, Co, March, 2002, pp. 14 – 25.

[15] M. Davis, "A wireless traffic probe for radio resource management and QoS provisioning in IEEE 802.11 WLANs," in *Proc. of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM Press, pp. 234 – 243, 2004.

[16] Samarth H. Shah, Kai Chen, and Klara. Nahrstedt, "Dynamic bandwidth management in single-hop ad hoc wireless networks", in *Mobile Networks and Applications*, 2005, Kluwer Academic Publishers, Vol. 10 Issue 1 – 2, pp. 199 – 217.

[17] G. Xylomenos and G. C. Polyzos, "Quality of service support over multi-service wireless internet links," in *Computer Networks*, Vol. 37(5), 2001, pp. 601 – 615.

[18] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Technical Report TR-96-007, Boston University Computer Science Department, 1996.

[19] Y. Wang, Z. L. Zhang, D. Du, and D. Su, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. of IEEE INFOCOM*, 1998, pp. 660 – 667.

[20] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM*, 1999, Vol. 3, pp. 1310 – 1319.

[21] B. Shen, S. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," in *IEEE Trans. on Multimedia, Special Issue on Streaming Media*, 2004, Vol. 6, Issue 2, pp. 375 – 386.

[22] P. Schojer, L. Boszormenyi, H. Hellwagner, B. Penx, and S. Podlipnig, "Architecture of a quality based intelligent proxy (QBIX) for MPEG-4 videos," in *Proc. of World Wide Web*, May, ACM Press, 2003, pp. 394 – 402.

[23] A. Laursen, J. Olkin, M. Porter, “Oracle media server: providing consumer based interactive access to multimedia data,” in *Proc. of the 1994 ACM SIGMOD international conference on management of data*, 1994, Vol. 23 Issue 2, pp. 470 – 477.

[24] C. Dixon, “For IPTV set top box vendors, the consolidation dance has started,” Accessed January 2006: http://www.tdgresearch.com/tdg_opinions_for_IPTV_set_top_box_vendors.htm

[25] S. Robertson and R. Rivin, “Designing IPTV set-top boxes without getting boxed in,” Accessed August 2006: http://www.analog.com

[26] Telchemy Application Note, “IPTV Performance Management,” Accessed June 2006: http://www.telchemy.com/appnotes/Managing_IPTV_Service_Quality.pdf#search=%22pdf%20set%20top%20box%20quality%20of%20service%22

[27] H. Radha, M. van der Schaar, and Y. Chen, “The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP,” in *Proc. of IEEE Transactions on Multimedia*, Vol. 3, No. 1, 2001, pp. 53-67.

VITA

Name: Varrian Durand Hall

Address: Department of Computer Science, Texas A&M University
College Station, TX 77843-2258

Email Address: cystar123@gmail.com

Education: M.S. Computer Science
Texas A&M University 2006

B.S. Mississippi State University 1994
Emphasis: Microprocessors and Computer Aided Digital Design

A.A.S. Electronics Engineering Technology
Hinds Community College 1989

Academic Research:

H. K. Lee, V. Hall, K. H. Yum, K. I. Kim and E. J. Kim "Bandwidth estimation in wireless LANs for multimedia streaming services," in *Proceedings of the 2006 International Conference on Multimedia & Expo (ICME).*

Graduate Research Assistant**,** Texas A&M University, Spacecraft Technology Center, June 2005 to September 2005

- Used Matlab to evaluate, modify and test pattern recognition algorithms used to detect Space Shuttle when approaching ISS.
- Used AGI STK (Satellite Tool Kit) software to create orbits and render images of shuttle performing flip maneuver done prior to docking with ISS.

Digital Image Final Project, Morphology, December 2005
Created a GUI Morphing program to allow users to enter several photographs, select item in each photograph to morph, and view a sequence of the selected images in each photograph morphing into each other. Process involved implementing complex algorithms and mathematical equations and implementation of a feature based technique.

Digital Image and Digital Signal Processing Summer Project, June 2005 to August 2005
Researched digital image and digital signal processing theorems and equations.

NASA JPL (Jet Propulsion Laboratory), Pasadena, CA, May 2005
Met with Dr. Eddie Tunstel, a senior robotics engineer and head of Spacecraft/Rover Engineering Team (SRET), for the Mars Exploration Rovers (MER). Discussed MER project, current research, robotic engineering principles, and future planned missions.