Streaming Video Performance and Enhancements in Resource-Constrained Wireless Networks

Justin James Madigan

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Streaming Video Performance and Enhancements in Resource-Constrained Wireless Networks
by

Justin James Madigan

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Fei Hu
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
July 2006

Approved By:

________________________
Dr. Fei Hu
Primary Advisor – R.I.T. Dept. of Computer Engineering

________________________
Dr. Xiaojun Cao
Secondary Advisor – R.I.T. Dept. of Information Technology

________________________
Dr. Marcin Lukowiak
Secondary Advisor – R.I.T. Dept. of Computer Engineering
Title: Streaming Video Performance and Enhancements in Resource-Constrained Wireless Networks

I, Justin James Madigan, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Justin James Madigan
Justin James Madigan

Date
Acknowledgements

I would like to thank Dr. Fei Hu for his constant help and support during this thesis, as well as my advisors Dr. Cao and Dr. Lukowiak for all their instruction and assistance.
Abstract

Streaming video is an increasingly popular application in wireless networks. The concept of a live streaming video yields several enticing possibilities: real-time video conferencing, television broadcasting, pay-per-view movie streaming, and more. These ideas have already been explored via the internet and have met with mixed success, largely due to the shortcomings of the underlying network. Taking streaming video to wireless networks, then, poses several significant challenges. Wireless networks are inherently more susceptible to failures and data corruption due to their unstable communications medium. This volatility suggests serious drawbacks for any implementation of streaming video. Video frame errors, jitter, and even complete sync loss are entirely conceivable in a wireless environment. Many of these issues have been undertaken and several approaches to mediation or even solution of these problems are underway. This thesis proposes to use advanced simulation techniques to properly exhaustively permute many vital parameters within a UMTS network and uncover, if they exist, bottlenecks in UMTS performance under considerable network load. This is accomplished via a described testing plan with simulation environment. Additionally this thesis proposes a new UDP-like transport layer specially optimized for streaming media over resource-constrained networks, tested to work with significant improvements under the UMTS cellular networking system. Finally this thesis provides several innovative new methods in the furtherance of the field of streaming media research in resource-constrained and cellular environments. Overall this thesis makes several important contributes to an exciting and ever-growing field of active research and discussion.
# Table of Contents

Abstract ................................................................................................................................. iv

Table of Contents ................................................................................................................ v

List of Figures ........................................................................................................................ vi

List of Tables ........................................................................................................................ ix

Glossary ................................................................................................................................. x

Chapter 1  Introduction ....................................................................................................... 1

Chapter 2  Background ....................................................................................................... 12

Chapter 3  Impacts of Varied Parameters on Video Performance in UMTS Networks ............ 23

Chapter 4  Enhanced Transport Layer for End-to-End Video Latency and Jitter Reduction in Wireless and High-noise Environments ............................................................................. 44

Chapter 5  Other Advanced Improvements in Time Synchronization and Overhead Reduction for Streaming Video ............................................................................................................. 62

Chapter 6  Conclusion ....................................................................................................... 73

Bibliography ......................................................................................................................... 75
List of Figures

Figure 1: Streaming Video and Buffer Underrun ........................................3
Figure 2: The OSI Model for the modern Protocol Stack (image from [9])...........7
Figure 3: Case 1: A piece of frame data is corrupt (image from [11])...................8
Figure 4: Case 2: Corrupt data with Resync Markers [11]................................8
Figure 5: Case 3: Corrupt data with bidirectional resynchronization markers and reversible variable length encoding [11]...............................................9
Figure 6: Worldwide Cellular Telephone Usage, image from [16].........................13
Figure 7: UMTS Network Cell Distribution [19].............................................14
Figure 8: UMTS Network showing the GSM Components [19]...........................16
Figure 9: UMTS capabilities versus Previous Technologies, image from [20]..........17
Figure 10: Mobile Services and their relationships with current uses (left - current, right – predicted future) [21].................................................................18
Figure 11: Mobile Telecommunication Subscribers Worldwide [22].....................19
Figure 12: An Example UMTS Network [23].................................................20
Figure 13: Images from the Jurassic Park movie displaying its various sceneries and vivid colors (Tracefile used) [27].........................................................25
Figure 14: Node Model Flowchart of operation for the video called party (Video receiver) ........................................................................................................26
Figure 15: Node Model Flowchart of operation for the video calling party (Video sender) .....................................................................................................27
Figure 16: Simulation Setup for Parametric Testing..............................................31
Figure 17: End-to-End Delay under RNC System Utilization of 10%, 50% and 90% (respectively, left-to-right) .......................................................... 32

Figure 18: RNC Multiple CPU Simulation Results, left – 1 CPU, center – 20 CPUs, right 32 CPUs .................................................................................................................. 37

Figure 19: Simulation Performance Results with all Components using 32 CPUs ....... 38

Figure 20: IP Buffer Modification Simulation Results, Left to right, 16 MB, 64 MB, 128 MB, 256 MB IP buffer memory ............................................................................................................ 39

Figure 21: Forwarding Efficiency Protocol Simulation Results .................................. 40

Figure 22: Simulation Results of All Components with 32 CPU and 256 MB IP buffer. 42

Figure 23: Simulation Results from Datagram Forwarding Buffer at 1000 Packets ....... 43

Figure 24: The UDP/IP Pseudo-header from RFC 768 [30]........................................ 47

Figure 25: UDP Lite headers from [28], closely related to the proposed approach (field names are the same) ........................................................................................................... 48

Figure 26: Ordinary UDP versus Proposed UDP, differences highlighted .............. 48

Figure 27: (Transmitter Side) Critical section of UDP code comparison: Left new UDP, Right - traditional UDP ........................................................................................................... 52

Figure 28: Simulation Setup for Ordinary UDP versus Proposed UDP ................. 54

Figure 29: End-to-End Delay Simulation Results Left, Original UDP, Right, Proposed UDP ......................................................................................................................... 57

Figure 30: Jitter Simulation Results - Left, Original UDP, Right, Proposed UDP ....... 59

Figure 31: Cristian's Remote Clock Reading Method, image from [35] .................... 64

Figure 32: Time Synchronization Timing Diagram .................................................... 66

Figure 33: A Single Time Synchronization Update .................................................... 67
List of Tables

Table 1: UMTS Network Cell Designations and Parameters ............................................. 15
Table 2: Specific Parameters and their Explored Permutations........................................ 29
Glossary

1G  First Generation Cellular Telephone Technology. Analog cellular communications with no capacity for data transmission.


3G  Third Generation Cellular Telephone Technology. Allows for video and voice communications as well as data transfers such as downloads from the internet and e-mail.

Bandwidth  The data rate achieved when communicating between devices.

Buffering  Temporary storing of information for future use.

CDMA  Code Division Multiple Access. A form of multiplexing often seen in cellular telephones that allows multiple users to share the same frequency range by assigning them unique codes.

FDMA  Frequency Division Multiple Access. A form of multiplexing that assigns each user a unique frequency to communicate on with their intended receiver.

GSM  Global System for Mobile Communications. The cellular telephone technology used in most parts of the world outside the United States and the most popular cellular standard in the world.
IP
Internet Protocol. The commonly used networking layer used in packet-switched networks such as the Internet.

Kbps
Kilobits per second. A data transfer rate equivalent to 1000 bits per second.

Mbps
Megabits per second. A data transfer rate equivalent to 1,000,000 bits per second or 1,000 kilobits per second.

MP3
MPEG-1 Audio Layer 3. Lossy compression for digital audio encoding, a popular music format in many media players known for its small file size but impressive quality.

MPEG-4
A popular standard used to compress audio and video content. Used in many cellular telephones and portable media players.

OPNET
A popular network simulation tool from OPNET, Inc.

OSI Model
Open Systems Interconnection Model. A layered networking protocol stack design commonly used in the Internet and many other networked devices.

Protocol Stack
An implementation of a networking protocol suite, such as the OSI model.

QoS
Quality of Service. The ability to guarantee data rates to individual users of a network. Difficult to achieve under times of heavy network load but important for reliable content delivery.
<table>
<thead>
<tr>
<th><strong>Streaming</strong></th>
<th><strong>Streaming Media</strong></th>
<th><strong>TCP</strong></th>
<th><strong>TDMA</strong></th>
<th><strong>Time Synchronization</strong></th>
<th><strong>UDP</strong></th>
<th><strong>UMTS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The concept of delivering media to be immediately consumed and discarded rather than stored for later use.</td>
<td>Any multimedia that is streamed; that is any multimedia sent to a user to be immediately consumed and then disposed of as opposed to a fully downloaded video that must be downloaded in its entirely before consumption.</td>
<td>Transmission Control Protocol. A transport layer protocol used to ensure delivery in unreliable environments. It is very reliable but costly in terms of network overhead.</td>
<td>Time Division Multiple Access. A method of multiplexing whereby groups of communicating network users are given specific time slots to transmit within. Outside their timeslot they must remain silent.</td>
<td>The process whereby two computers synchronize their internal clocks so that both clocks read the exact same time.</td>
<td>User Datagram Protocol. A simplistic transport-layer protocol used for unreliable transmission of information across a network.</td>
<td>Universal Mobile Telecommunications System. A new breakthrough cellular technology that promises to provide worldwide service as well as increased data rates, full multimedia support and constant internet connection.</td>
</tr>
</tbody>
</table>
Chapter 1  Introduction

In the past twenty years, the computer networks and computer usage in general has become increasingly geared towards media. Modern desktop computers often are labeled “media centers” if they possess sufficient media presentation capabilities. The trend has continued, with the concept of a “media center” extending to personal, mobile devices. One popular example is the Motorola Razr v3 phone which is capable of MPEG-4 playback, serves as an MP3 jukebox and can capture pictures and video with a built-in camera. [1] Although it may seem extreme, this type of functionality is common in today’s mobile devices. The popularity of these and similar handheld media devices indicates there is a strong consumer preference for devices that can deliver media to the end-user on command. With an increasing market push for on-demand media, the foundational substructure of today’s cellular and wireless networks must incorporate streaming media functionality. Streaming media is the term used specifically for media that is immediately consumed upon deliverance to its intended party. [2] This is defined in contrast to downloaded media which must be fully downloaded by the end user before it can be displayed. Streaming media allows a user to “buffer” only a small portion of the media that is currently being played. By discarding sections of media not currently in use, the user does not need to wait for the entire multimedia file to download. This method allows faster delivery and consumption, but places heavy demands on the delivery mechanism and forces strict quality of service (QoS) requirements on the network over which the content is being delivered. These challenges have plagued the industry for quite some time.
The main challenge is consistent delivery of whatever media is currently being consumed. Videos, as an example, are generally complex to deliver to an end-user and even under modern advanced compression techniques, can be quite large in file size. If the video is not streamed to an end-user with a consistent data rate, the user’s video experience is interrupted eventually by a buffer underrun. This is a frustrating, though non-fatal error caused by insufficient reception of information from the streaming source.

The concept of a buffer underrun is explained in the diagram below. Essentially the concept is simple. Providing streaming content over any transmission medium is prone to delays, latencies, and unforeseen stoppages. This assumption has to do with the variability of computing itself: it is not known precisely when interference may occur, or other undesirable phenomenon. These can interrupt data transmission and cause unwanted delays, but they are unavoidable. Over the internet, a myriad of factors can be the cause of unwelcome delay. Factors such as traffic, IP buffer space, and other concerns can cause variable delay when delivering content. These factors are assumed by any client accepting streaming media and therefore the client buffers a certain amount which varies depending on application. This buffer is stored in the event that a streaming server can not deliver the media in a timely fashion, or at least fast enough that the client need not wait for the next packet of data to arrive. Once this buffer is filled sufficiently, playback begins. Should the streaming server be unable to deliver content in a timely manner, the client will be forced to commence playback from the buffer. However, if the buffer should become empty and still no streaming content has arrived from the server, the client will have no choice but to interrupt the user’s video until more content has arrived.
Streaming Video, step-by-step

1. After the initial handshaking between the client and the server, the server begins transmitting streaming video to the client.

2. Under the assumption that the streaming server will not be able to consistently deliver the video, the client buffers some video to prevent "skipping" in the end-user's video experience.

3. When the buffer has filled to an acceptable size, the client begins playback of the streaming content. Should the server be unable to deliver content, the client will play video from the buffer.

4. If the server still does not deliver video and the buffer is empty, the buffer has underrun and the client can deliver no more video until the streaming server provides some.

Figure 1: Streaming Video and Buffer Underrun
Through much of the early nineties, when streaming video was still coming about, much frustration was seen at the slow response of technologies such as RealNetworks™ RealPlayer, mostly due to the large amount of buffering required before playback could begin [3]. Since RealPlayer was the premiere content delivery mechanism, users could do little but complain at the poor performance they received. The problem was twofold: CPU power was relatively low compared to what is required to adequately stream video, and the internet connections were largely dial-up at 56kbps. These low bandwidth connections are unsuitable for streaming large videos or other space-intensive media. With the rise of broadband internet and computing power becoming cheaper, streaming media flourished. [4] Additional reasons included the commercial interest in streaming video, and an increase in research behind compression technologies, making higher quality video smaller in size, thus easing the requirements of bandwidth on the media servers. With these advances, streaming media increased over 600% from 1998 to 2003 according to [5].

Streaming media has far from reached perfection; streaming content is still relatively low quality and delays are still frequently experienced by content consumers. With these challenges still in play, a further dimension is added with the rise of wireless devices. Streaming media over a wired device allows content providers to rely on the consistency of their transmission medium to a reasonable extent. This assumption is not valid in the wireless arena. Wireless transmissions are subject to a host of difficulties, including interference, jamming, reflections, diffraction, Doppler fading, distortion and scattering. All these factors result in a characteristically unpredictable data delivery experience when
exchanging information over a wireless network. With this volatility inherent in the transmission medium, it is difficult to conceive how something with strict QoS requirements such as streaming media could ever flourish in a wireless environment. Although the challenge may seem daunting, the demand for streaming content has shown to be incredibly high. With the advent and subsequent rise of cellular telephones and their dual nature as media centers, the demand for streaming media over wireless networks has shown to be extremely high.

Reliable content delivery is the most pressing concern in the field of streaming media over wireless devices. Since the wireless environment is so prone to errors, failures, interference and interruptions, guaranteeing any type of bandwidth for any purpose becomes troublesome. Many excellent methods have been suggested for combating this, such as CDMA codes [6] which are resistant to noise. However, the simple fact of the situation is that the wireless environment is noisy and error-prone. Reliable delivery is simply not possible, so many modern methods rely upon redundancy (which increases overhead and delays) and fault tolerance (which may or may not increase delays, depending on implementation). Redundancy techniques repeat pieces of information more than once in the hopes that if one piece is damaged, one of its copies will survive. Failing that, perhaps the correct data can be pieced together from the redundant copies, discarding the incorrect sections. This method is useful but far too taxing on resources to be acceptable in an environment where not only are resources scarce, but the application demand on resources is very high. Video and audio encoding/decoding are intensive operations for a CPU and require real-time delivery of data in a streaming situation. This
makes on time content delivery an absolutely necessity when dealing with streaming media. Hence using techniques which will cause overhead increase are typically simply unacceptable. Overheads must be kept to a minimum for a viable solution to a streaming media problem.

Current protocols, such as TCP [7] have robust implementations that are resistant to packet loss and packet error. However, their overhead has shown to be simply unbearable in the case of streaming media, especially in resource-constrained environments and wireless environments, as discussed in [3]. Even UDP, the simplest transport protocol [8], is not entirely ideal for streaming video. This will be discussed in depth later on, but there are certain fundamental issues underlying the typically seen protocol stack (shown below). The main purpose and drive of the protocol stack as it stands is to deliver content reliably and correctly at the expense of bandwidth and delays. Although the balance is not excessively tipped and indeed mediation can occur, the main focus is reliable content delivery at whatever cost is necessary. The opposite approach is required for streaming media. Users are far more content with a slightly distorted image in a video that lasts for milliseconds than a stoppage of video while rebuffering can occur every ten seconds or so.
The protocol stack contains significant amounts of checksumming and other error-catching codes which immediately flag packets as “corrupt” or “bad” and upon such a designation packets are summarily discarded. This type of redundant data integrity verification is useful for ensuring that data arrives correctly, but not useful for situations in which corrupt data is better than no data at all. There must be significant changes to existing protocol structure if streaming media is to take hold successfully since the specific challenges facing streaming media and regular data exchange are vastly different.

Figure 2: The OSI Model for the modern Protocol Stack (image from [9])
MPEG-4 is a popular format for many handheld devices, such as the Apple video iPod [10] and many popular cellular phones (such as the Motorola Razr v3 [1]). Each individual picture in an MPEG-4 video is referred to as a “frame.” Each frame is made up of a series of 8x8 pixel squares called “blocks.” A group of blocks, or GOB, is a row of blocks. A GOB also contains a header in MPEG-4 (although this is optional, it is typically used if any errors in video transmission are expected) which can aid in resynchronization if a section is corrupt or unusable. Because MPEG-4 videos are encoded with a variable bit-rate, if a piece of a frame is corrupt then the frame would ordinarily be unusable (see illustration below).

![Figure 3: Case 1: A piece of frame data is corrupt (image from [11])](image)

However, to combat this, a known data pattern is inserted every so often as a “marker” to allow resynchronization if errors occur. This allows the decoder to “pick up” the frame as soon as a bit marker is identified. This is illustrated below.

![Figure 4: Case 2: Corrupt data with Resync Markers [11]](image)
Although this method leaves undecodable data it is clearly preferable to the option of simply losing an entire frame. This method is actually improved upon in the MPEG-4 specification when reversible variable length encoding is used by allowing backward decoding. That means when a resynchronization marker is found, the frame can be regenerated backwards starting from the first good resynchronization marker. This process is illustrated below.

![Diagram of frame structure](image)

**Figure 5: Case 3: Corrupt data with bidirectional resynchronization markers and reversible variable length encoding [11]**

Clearly from the diagrams shown, frame loss is avoidable even with corrupt data. This means that when faced with the decision to accept a corrupt packet or reject it entirely due to unknown corruptions within the packet, the advantageous choice is to accept the packet and allow the built-in error correction mechanisms to take hold. It should be duly noted that this is not the only error-correcting method available to MPEG-4, and other methods exist to layer atop this method which make the MPEG-4 encoding even more robust. Nonetheless the point remains entirely valid that clearly packets are better off used than discarded in MPEG-4, even with corruption.

This thesis exists under a threefold purpose: to better understand the limitations of the UMTS system under heavy network conditions, to improve streaming video transmission
in the UMTS network, and to propose new ideas that will provide exiting improvements to the field of streaming media. Toward this end, this thesis is structured as follows: This chapter has provided an introduction to streaming media and background information regarding MPEG-4 videos. Chapter 2 provides an introduction to the concepts behind UMTS and its underlying substructure, ideas, and development plans. Chapter 3 provides comprehensive and detailed network simulation results showing the performance of an OPNET-simulated UMTS network under heavy traffic conditions. Chapter 4 introduces a proposed UDP scheme that considerably reduces delays and jitter in the overall performance of streaming videos, especially under conditions of heavy environmental noise. Chapter 5 contains many new ideas ranging from time synchronization to protocol enhancements that can potentially improve the performance of streaming media in wireless and cellular networks. These ideas presented in Chapter 5 are outside the general scope of this thesis but should provide ample new ground for future research into their respective topics.

This thesis is important because of the ever-growing demand for more streaming content over wireless and cellular devices. Currently not enough is known regarding potential bottlenecks in network system, bottlenecks that could have the capability to bring down a network during times of heavy traffic and network load. This thesis provides this in-depth analysis in the new cellular technology, UMTS. UMTS promises to be a worldwide cellular solution and provide greatly enhanced data rates for streaming media. Clearly with such lofty goals, stressful simulations are important to properly tax the network system to ensure no weak points exist. This thesis also provides an important new
technological improvement: an enhanced transport layer protocol to provide better performance in streaming media, specifically streaming video. This should prove to be an important stepping stone in the construction of a truly seamless cellular/wireless streaming media solution. Finally this thesis provides many exciting new ideas for the improvement of the existing telecommunications systems. As such this thesis covers many extremely important ideas currently being actively researched by many members of the multimedia and networking community and presents innovative new ideas to contribute to this field.

In summary this section has shown that streaming media is an important growing technology with serious challenges. A brief history of streaming media, its shortcomings and the current resolutions to those shortcomings has been discussed. With the proliferation of handheld wireless devices, a new need for streaming media over resource-constrained wireless devices has been identified and discussed. The current protocol stack has been shown to be insufficient and ill-suited to the needs and challenges associated with real-time streaming media and specific examples of these shortcomings have been expressed. Finally an introduction to the MPEG-4 codec and the sensitivity of streaming media to packet loss has been discussed with specific examples given in the MPEG-4 codec. The next chapter will introduce the platform upon which this thesis is based: the UMTS cellular network technology as well as provide some foundational concepts which will be used in the remainder of the thesis and development of the new ideas proposed herein.
Chapter 2  Background

Currently, the European and Asian cellular telephone markets are saturated with a technology known as the Global System for Mobile Communications, or GSM. GSM is capable of 9.6 kbps of user data rate for services such as videotext, facsimile and teletext [12]. This kind of data transfer rate is unacceptable for streaming video, which requires significantly more in terms of available bandwidth. With GSM technology nearing the end of its lifecycle, a newer technology has come to replace the aging GSM, and this technology is called the Universal Mobile Telecommunications System, or UMTS. UMTS is an exciting new technology, capable of 1920 kbps: a data rate which far outweighs the GSM capability. [13] Although users can currently expect rates closer to 384 kbps, Japan is implementing a 3 Mbps upgrade to its currently active UMTS networks. [14] Clearly this newly available bandwidth opens the door for a myriad of new uses for the cellular telephone network in Europe and Asia.

One of the key strengths of UMTS is its infrastructure which is constructed to intentionally and permanently meld the internet with telecommunications. The UMTS network has specific provisions to allow users to connect to the internet anytime they wish. Not only does this open the door for more multimedia content to be available to the cell phone user, but it also provides a doorway for video chatting (live video conferencing between communicating parties). If this technology continues to develop and new technologies arise that can meet the growing demand for streaming media, mobile telecommunications could easily become entirely done through live video
conferencing. This makes UMTS the ideal pathway to the future as it supports higher data rates and internet connection anytime.

Cellular telephone usage is immense and growing daily. [15] The map below shows worldwide cellular telephone usage and clearly shows the widespread penetration that cellular telephones have. As of October 2005, 194.5 million Americans [17] were subscribed to a cellular telephone service. This constitutes 65% of the population of the United States. With the cellular telephone numbers as they are and continuing to grow, clearly there is a high demand for cellular service and should a new streaming video over telecommunications networks service arise, demand for it too would immediately be immense. Because of this worldwide demand, the underlying foundation of any new service must be carefully designed and expertly crafted to ensure the best possible performance to the greatest number of customers. Clearly as described in the previous chapters this entails a serious and very immense amount of groundwork.

Figure 6: Worldwide Cellular Telephone Usage, image from [16]
The design of UMTS is not only to be meshed with the internet but also to provide worldwide coverage [18]. To provide backward compatibility, UMTS has built-in functionality to allow 2G phones to communicate over the UMTS network in a process known as inter-network roaming [19]. To accommodate the various speeds, distances, and other factors involved in consistent network communication, the UMTS network has been designed with a variety of “cells” which cover different distances. See the illustration below.

Figure 7: UMTS Network Cell Distribution [19]

These cells have a variety of capabilities, outlined below. The general purpose of each cell is to adequately service its region of coverage.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bitrate</strong></td>
<td>≤ 2048 kbps</td>
</tr>
<tr>
<td><strong>Terminal Speed</strong></td>
<td>&lt; 10 km/hr</td>
</tr>
<tr>
<td><strong>Cell radius</strong></td>
<td>10 – 50 m</td>
</tr>
</tbody>
</table>

**Pico Cell**

*Building-size network, applicable to airports, coffee shops, office buildings, etc.*

<table>
<thead>
<tr>
<th>Bitrate</th>
<th>2048 kbps – 384 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal Speed</td>
<td>120 km/hr</td>
</tr>
<tr>
<td>Cell radius</td>
<td>500 m</td>
</tr>
</tbody>
</table>

**Micro Cell**

*Coverage for densely populated metropolitan areas, such as downtowns*

<table>
<thead>
<tr>
<th>Bitrate</th>
<th>384 kbps – 144 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal Speed</td>
<td>120 – 500 km/hr</td>
</tr>
<tr>
<td>Cell radius</td>
<td>2000 – 6000 m</td>
</tr>
</tbody>
</table>

**Macro Cell**

*Suitable for expansive, less densely populated areas such as farmland, rural land and suburbs*

<table>
<thead>
<tr>
<th>Bitrate</th>
<th>384 kbps – 144 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal Speed</td>
<td>&lt; 1000 km/hr</td>
</tr>
<tr>
<td>Cell radius</td>
<td>Worldwide</td>
</tr>
</tbody>
</table>

**World cell**

*Satellite-enabled worldwide coverage to reach even remote areas and desolate landscapes as well as mountains, oceans and other uninhabitable regions*
This logical division of services and bandwidth actually is a very interesting strongpoint for the UMTS network. By logically dividing the available services and providing more coverage where it is most likely to be used, UMTS can avoid wasting resources on areas not likely to often be used. Further UMTS can wisely partition the existing resources to be more heavily available in areas that are more likely to require them.

To provide an easy transition, UMTS utilizes much of the core GSM technology. The modular approach of UMTS (see the illustration below) allows the core and access networks to be developed separately and furthermore their separation also allows updates and development to proceed on those units independently.

![UMTS Network showing the GSM Components](image)

*Figure 8: UMTS Network showing the GSM Components [19]*
The UMTS network's increased bandwidth opens the door for an immense amount of potential services. The vast improvement over previous technologies can only be appreciated by comparison, see the figure below. Not only does this open the door for video conferencing, but also interactive shopping, mobile bill-pay, online banking, instant emergency services, mobile classes and coursework, and much more [20]. These groundbreaking new services could potentially revolutionize the worldwide cellular telephone market.

<table>
<thead>
<tr>
<th>Services</th>
<th>2G</th>
<th>PSTN</th>
<th>ISDN</th>
<th>2G+</th>
<th>UMTS/3G</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-mail file 10 kbyte</td>
<td>8 sec</td>
<td>3 sec</td>
<td>1 sec</td>
<td>0.7 sec</td>
<td>0.04 sec</td>
</tr>
<tr>
<td>Web page 9 kbyte</td>
<td>9 sec</td>
<td>3 sec</td>
<td>1 sec</td>
<td>0.8 sec</td>
<td>0.04 sec</td>
</tr>
<tr>
<td>Text file 40 kbyte</td>
<td>33 sec</td>
<td>11 sec</td>
<td>5 sec</td>
<td>3 sec</td>
<td>0.2 sec</td>
</tr>
<tr>
<td>Large report 2 Mbyte</td>
<td>28 min</td>
<td>9 min</td>
<td>4 min</td>
<td>2 min</td>
<td>7 sec</td>
</tr>
<tr>
<td>Video clip 4 Mbyte</td>
<td>48 min</td>
<td>18 min</td>
<td>8 min</td>
<td>4 min</td>
<td>14 sec</td>
</tr>
<tr>
<td>Film with TV Quality</td>
<td>1100 hr</td>
<td>350 hr</td>
<td>104 hr</td>
<td>52 hr</td>
<td>&gt;5 hr</td>
</tr>
</tbody>
</table>

Figure 9: UMTS capabilities versus Previous Technologies, image from [20]
With this new technology come many new challenges. Because of the high demand for the services that will be offered and the immense number of users which is only predicted to increase, and expected to exceed 600 million by 2010 (see figure below). By adding so many services to an already full architecture any initiative must be adequately prepared for the kinds of immense traffic patterns that will ensue.
To properly accommodate the new services and their associated consumer load, the UMTS architecture will use the standard OSI model protocol stack with UDP support for video transmission and other error tolerant applications. The current UDP transport protocol is the most popular way to transmit video over any internet-based network due to its simplicity; however important changes can be made to this structure to improve upon its effectiveness in a video transmission role or indeed in any role where there is loss sensitivity but error and corruption tolerance.

Currently, UMTS is a new technology with immense complexity underneath its superficial structure (see below). It is an ambitious new development and as such it brings with it a significant amount of risk. Much of this risk can be attributed to the fact that there are no comprehensive parametric permutation and modification simulation results. Such a set of results would be immensely useful in understanding exactly how
this UMTS system will respond to low-level traffic generation and traffic fluctuations. This data could easily be used to properly understand why certain components in the UMTS system perform worse than others. These system components could be adequately modified to prevent the negative impact of the traffic changes during periods of high demand.

Figure 12: An Example UMTS Network [23]
An additional but vital benefit of such comprehensive parametric testing would be the identification of bottlenecks in the UMTS system. Currently it is not known if bottlenecks even exist – if they do, and they are severe, UMTS network operation could be hindered significantly if these bottlenecks begin affecting performance. For example, if a pipeline in the UMTS network is utilized more than initially anticipated, network performance could be delayed to the entire nearest cell (of any size designation) because of the bottleneck effect. However, if these bottlenecks in performance were identified, they could be dealt with beforehand so that they never affect overall performance. This thesis will deal with this important issue later on.

Of course this area is a hotbed of new research due not only to its ambitiousness but also to its immense potential rewards. New ideas are often proposed and should be at as high a frequency as possible to keep the flow of new information and work into the area at a high level. Promoting continual research in the area is always an important and effective way of ensuring the future success of any new technology. This thesis will additionally propose some new, unexplored ideas that should have a positive impact on the field of UMTS.

In summary the UMTS network has been explained along with its motivations, substructure, underlying layout and foundational technologies. UMTS is a promising architecture with a vast number of entirely new applications ranging from immediate medical care to online shopping. The expansive bandwidth that UMTS will offer coupled with its revolutionary technological foundation will yield a new era in
telecommunications and the internet as we know it. With this comes fresh challenges associated with UMTS have been discussed along with the new ideas this thesis will explore in the furtherance of the UMTS technology.
Chapter 3  Impacts of Varied Parameters on Video Performance in UMTS Networks

To correctly investigate any phenomenon or idea that is difficult to implement physically, a proper simulation is required. Without a robust and objective simulation to compare other results against, it is impossible to gauge the performance of one algorithm or idea against another. The industry-standard network simulation tool, OPNET Modeler, is an ideal choice for the investigation of UMTS traffic operation and its parameters for many reasons. First, it is a widely used and recognized simulation tool in the networking field. Second, it recently has been upgraded with full UMTS simulation capabilities [24]. Hence OPNET is the tool of choice for the investigations herein.

Ideally, any network would exhibit the same performance under any level of load on the system. In reality this is not possible since resources are always limited. Thus attention must be paid to maximizing the performance seen by the user at all times even under times of heavy strain on the physical system. To best accomplish this, the system cannot be constrained by any one system component – that is, there can be no “bottleneck” in the system. Current research into bottlenecks of UMTS uses methods such as monitoring TCP traffic to identify bottleneck links. [25] The approach taken here, however, is to attempt to identify core components of the system that may cause a bottleneck in system performance. If such performance-critical components could be identified, a best-effort approach could be taken to ensure that their utilization does not exceed some “critical mass” – that is, a utilization level past which network performance suffers dramatically.
This could significantly improve performance of the network in times of heavy load and could be used with great success in high-traffic areas.

Although a single point of failure is entirely possible, the potential for a set of circumstances under which a bottleneck occurs should not be dismissed. For example, perhaps the system performs adequately under a high strain exhibited upon the CPU of an RNC. Assume additionally that the system performs adequately under a heavy strain on the IP traffic buffers of the RNC. We might be tempted to logically conclude that the system should perform adequately if both the aforementioned conditions are simultaneously met; however this assumption would be specious. Perhaps testing would reveal that although the system performance does not severely degrade under load from one condition or the other, but under both the performance is dismal. These sorts of potentialities need to be accounted for in order to properly determine specific failure characteristics for a given network.

Streaming video is a network-intense operation. Even modern-day MPEG4 and related video formats require significant amounts of bandwidth. According to [26], to stream an hour of video compressed at 300 kbits/sec requires 300 Mbit/sec (125.68 GiB/hr) of bandwidth to stream to a thousand viewers. These requirements raise serious concerns. Thus, streaming video is used to test the performance of the UMTS network under varying loads in order to identify bottlenecks in the system performance.
Unfortunately, there is only a “video-conferencing” application configuration in OPNET with no native support for outside video. In order to properly stream video, an acceptable trace file should be chosen. For this project, the verbose trace file from the movie Jurassic Park was chosen as the sample trace to stream from the different UMTS nodes. The movie contains various sceneries, landscapes, colors and other phenomenon that make it a useful tool in exploiting the video transmission techniques found in UMTS and other related networks.

Figure 13: Images from the Jurassic Park movie displaying its various sceneries and vivid colors (Tracefile used) [27]
In order to send the video properly, the OPNET code must be modified to accept foreign video. To do this, the gna_mgr_video_calling/gna_mgr_video_called objects must be modified as they perform the video conferencing simulation in OPNET. A node model file (shown below) shows the flowchart operation of the video conferencing simulation calling and called party.

Figure 14: Node Model Flowchart of operation for the video called party (Video receiver)
Figure 15: Node Model Flowchart of operation for the video calling party (Video sender)

Specific modifications must be made to this existing substructure in order to properly accommodate streaming video from an actual MPEG trace file. The essential underlying technique used herein was originally developed by Sailaja Yagnavajhala of Clarkson University. During the transmission of what is normally “simulated” video conferencing data, actual data must be transmitted. The changes associated with this process require several incremental alterations to be accomplished properly in an OPNET simulation. These modifications are summarized below.
Video Transmitter

- During a connection attempt:
  - When moving from “connect” to “idle” state, initialize the tracefile state variable with the external tracefile array

- In the send state:
  - Set the frame size to the proper MPEG4 tracefile’s specified value
  - Send the frame type specified
  - Send the frame number specified
  - Ensure that the sending stops when the video has ended

Video Receiver

- During a connection attempt:
  - When moving from “connect” to “idle” state, open the received tracefile in write mode to accept received data
  - Initialize the tracefile state variable which contains the state array

- In the SEND state
  - Initialize and send the tracefile summarization/application data that will be sent with each packet

- In the RECEIVE state
  - Reconstruct the trace file information from the received data and write it to the file

From this, it is clear that not only must the main steps of the video transmission change, but also its reception and reconstruction. The output file (saved as received_trace.dat) can then be compared against its expected counterpart (in this case, the trace file data from the Jurassic park video). This allows easy detection of errors or miscommunications between the two parties.
With this modified infrastructure in place, the modification of parameters may begin. The parameters in this case were chosen based on items that are likely to be over utilized during heavy network congestion. Things like CPU usage, IP packet buffers, and background system utilization are of particular importance. The variables and their altered values may be seen in the table below along with a summarized rationale. Further explanations will be proffered with the results when discussed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>New Values</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background System Utilization</td>
<td>0%</td>
<td>10%, 50%, 90%</td>
<td>Test the effects of modifying background utilization to determine if the modification of the usage of any one component will adversely affect system performance more than the others.</td>
</tr>
<tr>
<td>GGSN</td>
<td>50%</td>
<td>90%</td>
<td></td>
</tr>
<tr>
<td>RNC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGSN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>1</td>
<td>20, 32</td>
<td>Determine if multiprocessing capability in the RNC would significantly improve performance in heavy traffic. Expected result would be improved performance of the multiprocessor system due to its ability to handle multiple tasks/connections simultaneously.</td>
</tr>
<tr>
<td>RNC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Buffer Space</td>
<td>16 MB</td>
<td>64 MB, 128 MB, 256 MB</td>
<td>Ascertain the effects of additional buffer space for incoming packets. Expected result would be fewer dropped packets and hence better performance.</td>
</tr>
<tr>
<td>RNC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datagram Forwarding Efficiency Protocol RNC</td>
<td>Disabled</td>
<td>Enabled</td>
<td>Determine if using a datagram forwarding efficiency algorithm provided in OPNET will improve performance.</td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>1 CPU, 32</td>
<td></td>
<td>Determine the effects of</td>
</tr>
</tbody>
</table>
The simulation setup was a simple two-node UMTS system exchanging heavy amount of normal resolution video traffic. This encompasses the same type of traffic seen under a many-node scenario (in this case, however, two nodes form multiple connections and exchange data rather than multiple nodes; however the effect seen from the RNC and similar components is identical).
The first explored topic was the background system utilization, as per the aforementioned table. The purpose here was to ascertain whether or not the performance degradation seen from background system utilization on any particular UMTS device would be greater or less when compared with other devices under a similar strain. In other words, this was a simple way to root out "bottlenecks" in the system by letting OPNET simulate
background system utilization. The obtained graphs from the simulation may be viewed below.

Figure 17: End-to-End Delay under RNC System Utilization of 10%, 50% and 90% (respectively, top left, right, bottom)

These results clearly indicate, when looking left to right in figure 5, the occurrence of a single cutoff value – specifically, performance seems relatively stable outside the range of 10% system utilization. In other words, after about 10% of system utilization the heavy
system load no longer sees performance degradation from the further utilization. This is logical since this indicates that the system can adequately handle the traffic simulated with only 90% of the system resources and will not see further degradation given the system resources until after 90% utilization. Thus the system utilization in itself is not enough to constitute a “bottleneck” in the UMTS system; however an important lesson is revealed through this experiment: system utilization affects performance even at low usage levels. Although these performance impacts are not drastic, they do exist and clearly this indicates that in even heavier traffic loads the performance impact would be exacerbated. System utilization should be kept at a minimal wherever possible, although with high traffic areas this may be an unrealistic expectation.

The next investigated phenomenon is the number of CPUs in the RNC and in all components. These results, shown graphically below, indicate the differences. According to the data, an optimal situation is found in the single CPU scenario. The 20 CPU scenario sees an actual performance degradation and the 32 CPU scenario seems to recover slightly. This is most likely because the level of exploitable parallelism in the given simulation scenario is reached in the single CPU scenario. In other words, there is very little parallelism to exploit herein. Perhaps in a truly multi-client scenario, one in which there were many simultaneous connections coming from autonomous clients in varying locations with different demands this scenario might play out differently under the various numbers of available computing units. Nonetheless at this time it seems the single core RNC performs the best in the given scenario.
Object: A of Network
average (in Video Calling Party Packet End-to-End Delay (sec))

Object: B of Network
average (in Video Called Party Packet End-to-End Delay (sec))
Object: A of Network
average (in Video Calling Party, Packet End-to-End Delay (sec))
Object: B of Network
average (in Video Called Party, Packet End-to-End Delay (sec))
A full multiple CPU load was given (that is, all components were given 32 CPUs) and the results displayed below. In this scenario, the clear result is virtually identical to the single CPU scenario. Thus it is very clear that at no level is there exploitable parallelism for the explored case. Hence in situations of single-stream video traffic between communicating nodes, it does not appear that there are significant, if any, gains from multiple processing at any level since there does not appear to be any exploitable parallelism.
If packets are coming too quickly at any component that is required to forward them, it is conceivable that such a component could become clogged and be forced to drop packets accordingly. Therefore, it stands to reason that increasing this buffer size could result in better performance due to fewer dropped packets. This would translate to the final end-to-end delay, which incorporates packet loss. However, the results in this case indicate no measurable gains whatsoever from the IP buffer increase. Thus although in this case there is little forwarding to be done and thus not a reasonable amount of exploitable forwarding space to be used by the increased space and hence no performance impact is seen. Nonetheless it is entirely possible in a scenario where more forwarding were required that
these results might be different. A more powerful scenario environment and processing elements would be required to properly test this phenomenon.

Even with the previous results, perhaps it is not the buffer space that is an issue but the forwarding protocol that might be used. OPNET has a built-in forwarding efficiency
protocol which can be used. Although no specific details are given in the OPNET documentation, the results supposedly can enhance forwarding performance in packet-switched networks. The results of implementing this protocol are shown below. In this case we actually see degradation in performance, albeit a slight one. This protocol should not be used, however we do find an interesting phenomenon here – performance seems relatively hinged on the forwarding protocol. Further exploration into this phenomenon would be useful since such sensitivity was found in this simple test. There is the possibility that the forwarding protocol could hold significant ability to improve performance and yield the desired “bottleneck” this research is searching for.

![Figure 21: Forwarding Efficiency Protocol Simulation Results](image-url)
The next investigation centers upon the concept of a dual impact of two previously explored elements: the multiple CPU option alongside the IP buffer option. Perhaps a source of performance degradation in the multiple CPU scenarios wasn’t simply nonexistent parallelism but the increased speed of the 32 CPU processing was producing packets too fast for the buffers to handle and thus packets were being dropped as a result. In such an event, a bottleneck would have been inadvertently created. To test this, giving the entire scenario a complement of 32 CPUs and 256 MB of IP buffer space should show performance improvement if such a bottleneck had been created. The results show an actual slight performance improvement over the single CPU, 16 MB IP buffer space scenario. However this improvement is not enough to warrant a bottleneck discovery, although further research into this phenomenon may yield further intriguing results. The probable cause of the performance improvement here is simply the faster processing ability coupled with the buffer space improvements have removed a few packet drops somewhere in the simulation. The results are not drastic enough to be convincing.
The final phenomenon investigated is the potential for a datagram forwarding buffer of 1000 packets as opposed to the default value of 10. Instead of buffering at the IP level, this would buffer at the datagram, or transport layer. In this case the idea is presented that at the UDP layer, perhaps corruption or packet loss is being seen and thus by increasing the available buffer space for UDP packets, this could be partially alleviated. The results are given below and show a very good matching between the sender and receiver end-to-end delay. There is also a slight performance improvement evident in the graph. This shows that perhaps at the UDP layer there are some exploitable performance
enhancement techniques. The following chapter of this thesis will present a potential method for UDP level modifications toward overall streaming video performance improvement.

\[ \text{Object: A of Network average (in Video Calling Party Packet End-to-End Delay (sec))} \]

\[ \text{Object: B of Network average (in Video Called Party Packet End-to-End Delay (sec))} \]

\[ \text{Figure 23: Simulation Results from Datagram Forwarding Buffer at 1000 Packets} \]

In conclusion, a comprehensive set of parametric tests were performed on the UMTS network under a video traffic load. Several conceptual bottlenecks were explored including the concept of a CPU-bound bottleneck, memory-bound bottleneck, and a combination thereof. No singular bottleneck was found for the case that was explored,
although the possibility of bottlenecks in different scenarios was suggested during various parts of the simulations. The most significant result was the UDP datagram forwarding protocol enhancements which yielded a slight but noticeable change – besides a small performance improvement an end-to-end delay match was found between sender and receiver. This means that there isn’t any packet loss or other phenomenon hindering one side more than the other (sender or receiver). The following chapter of this thesis explores a potential way to exploit the ramifications of this result toward better streaming video performance in UMTS networks.

Chapter 4 Enhanced Transport Layer for End-to-End Video Latency and Jitter Reduction in Wireless and High-noise Environments

In the previous chapter, explorations into possible bottlenecks were made. Resulting from these explorations, the potentiality for improvement of end-to-end delay in UMTS delivering streaming video content was found at the UDP layer. This chapter deals with a potential way of improving the performance of the overall system by modifying the UDP layer.

Many different methods have been attempted to improve performance of videos across congestion-prone networks. Among these methods is an approach known as UDP Lite [28], a modified version of which is the focus of this chapter. UDP lite is a conceptually
straightforward method of improving performance through the reduction of packet loss. The concept behind UDP lite is the specification of data as being partially free of corruption. If such a “corruption threshold” is met, the packet is accepted regardless of the fact that some corruption may indeed exist in the packet payload. This method has been implemented with success toward cellular video, with significant performance improvements being seen in [29].

In many different algorithms for the resource-constrained transmission of voice and data, packet loss is difficult to recover from. For example, in many video compression algorithms, “key frames” are sent periodically to update the exact video frame information, but generally only changes in the video from frame to frame are transmitted. This means that only some metadata about the current video status is being sent from end to end rather than entire frames. This can be envisioned as a simple set of sequential numbers, each depending on the previous number. If a number in the sequence is lost completely, the result is the sequence becomes unintelligible to the receiver, since the sequence depends on the correctness of the previous number. It is similar in video – a lost packet may manifest as visual corruption or loss of synchronization. In this sense, it is important to minimize packet loss whenever and however possible in video transmission schemes.

Various schemes for preventing total packet loss have been suggested, such as piggybacking [31], forward error correction (FEC) [32] and even Reed-Solomon coding. [33] In piggybacking a lower-quality or degraded version of the previous packet is sent
alongside the current packet. If a single packet is lost, the lower quality version that is “piggybacked” is used instead of the high quality packet. This method is difficult and unacceptable to maintain in a highly resource-constrained environment due to the amount of overhead. Forward error correction is complex and causes computational overhead on the transmitter and the receiver since the code must be computed and checked. Reed Solomon coding is based on over-sampling data [35] and also will result in overhead. An ideal method for use in a resource-constrained environment would improve performance without requiring significant additional overhead.

The figure below shows the UDP and IP pseudo headers [30] as seen in RFC 768. In order to accomplish the stated goal of improved performance in streaming video without significantly impacting the amount of overhead required, a change must be made in the existing header structure. To maintain compatibility with other applications, this change should not alter the underlying substructure of the header format; rather it should be a change which uses the existing fields in a perhaps more clever or optimized way in regards to real-time streaming video. The change suggested here is similar to that of UDP lite: change the length field in the IP header to a “checksum coverage” field and alter the checksum to be a partial checksum. It is true according to [28] that real-time video and audio protocols are more sensitive to packet loss than to partial packet corruption. Furthermore many protocols contain header information or important details at the front of the packet that are important. Thus it stands to reason that if the first few bytes of the packet might be deemed trustworthy insofar as the data it contains, that the packet itself might be salvaged by the error correction and fault tolerance built into the higher layers.
(namely, the video or audio decompression scheme in use in the application layer). The proposed scheme, then, would provide a checksum coverage field indicating how much of the packet (beginning from the start of the UDP layer header) is covered by the indicated checksum. The checksum, then, is replaced by this partial checksum. By specifying the packet information as being partially error-free, a corrupt packet would be allowed to proceed to the higher layers and the higher level fault tolerance and error correction capabilities would be relied upon to mend the damage in such a way that would be transparent to the end user.

| Source address | Destination address |
| Source port | Destination port |
| Zero | Proto | UDP length |
| Length | Checksum |

Figure 24: The UDP/IP Pseudo-header from RFC 768 [30]

To properly understand this approach, a flow chart is shown with the specific process highlighted. The method proposed here is a strictly transport-layer protocol, thus the transport layer methodology is described herein. Other protocols are certainly usable above or below this one. The UDP-Lite Pseudo header block diagram is also given as the fields are very similar to the proposed approach.
The specific protocol will now be discussed at length to point out areas of difference. For outgoing information, first the port numbers are stored as they will be used. This includes
the source and destination port, just as in regular UDP. Next the checksum is set to 0. This is a requirement of the protocol and prevents the checksum from being erroneously included in any checksum calculations. The checksum coverage field is used to determine the number of bytes of the UDP packet to checksum. The checksum coverage field specifies the number of bytes, beginning with the first byte of the UDP header. A legal nonzero coverage value \textit{must} be greater than 7 since the UDP header is 8 bytes and the checksum \textit{must} cover the header. This is done to ensure the proper header values. Essentially this signifies that the header itself is considered universally vital to operation regardless of the specific application. A zero coverage value indicates the entire packet is included in the checksum, making the approach identical to ordinary UDP in this case. The checksum is computed through 16-bit additions of each 16-bit word until the checksum is complete, pursuant to the aforementioned rules. The one’s complement of this sum is taken, and the result of that operation is stored as the checksum. With this, the packet may be passed to a lower layer. For incoming data, the reverse process is done. The checksum is stored and the checksum field set to zero, the checksum is calculated using the specified rules and the result (without the one’s complement) is added to the checksum. If the result is all ones (0xFFFF) then there are no errors in the section specified by the checksum coverage field and the data may be passed on to the higher layer for further processing.

The difference between the protocols is subtle, but its impacts on performance can be profound. As discussed earlier, specifying data as partially insensitive to errors is often a better choice than specifying data as wholly insensitive to errors, as UDP currently does.
This change allows higher layers to use their error detecting and correcting protocols while not entirely discarding the lower layer of protection provided at the UDP layer. In the standard UDP lite, which has slight differences compared to the proposed model, performance has increased significantly in real-time audio-visual data transmission with sources reporting improvements in end-to-end delay of 26% and 50% less packet loss than traditional UDP [33]. Hence although the differences between the protocols may seem trivial, its impacts are certainly far from it.

This code was implemented in OPNET for simulation. The crucial aspects of the code will be explained here, along with a quick tutorial on their functionality. Below is a comprehensive flowchart illustrating and explaining the critical section of code. A side-by-side comparison of the traditional UDP code and new proposed UDP code is provided. This is the critical section of code as far as the new UDP is concerned — it is the checksum creation and storage for the UDP layer. Although many other changes and tweaks were necessary elsewhere in OPNET to prevent other layers from using their own data integrity verification methods and thus overriding what is being done here, this section of code is crucial and indeed forms the backbone of the new approach. The idea here is that the old checksum algorithm is simple but blunt: checksum the entire packet, including the UDP header. The new, improved approach is to checksum the UDP header plus only a small portion of the packet. This piece of the packet is thus free from errors and can be trusted. With enough of the beginning portion of the packet verified as correct, the error-correcting nature of the MPEG-4 layer above the transport layer can be held to sufficiently detect and correct the various errors that may or may not exist in the
remainder of the packet. The code section explained below highlights various important differences between the previously used approach to UDP and the proposed approach.

The new UDP code is summarized here. First the length is computed from the packet size. Next the checksum is computed based on the checksum length specified either by the user default (which is 20 bytes, including the 8 byte UDP header) or by a 0 which indicates checksumming the whole packet. The checksum length to actually compute is found by taking the previously determined value and subtracting 8 (bytes) which are the UDP header and must be part of the checksum. Next the checksum is computed and the one’s complement is taken so that it can be passed to the next layer.

On the receiving end, the code is essentially identical save for the fact that it is entirely reversed. In the same way as in the transmitter, the checksum is computed; this time after the received checksum is saved. Note that any checksum coverage values of 1 to 7 are deemed invalid and the packet is immediately discarded, this applies also to packets with checksum coverage larger than the packet itself. Provided the packet passes these “sanity checks”, after the checksum is computed it is added to the received checksum. If the result is all 0’s with an overflow then the data is verified up until the point that the checksum coverage specifies. At this point the data can be considered partially insensitive to errors up until the point of checksum coverage. It is then accepted and sent to the next higher layer for processing (IP layer).
To properly test that the protocol will provide an improvement over traditional UDP, the end-to-end delay was specifically tested in order to ascertain the particular improvements in performance that the proposed model will offer over traditional UDP. While packet
loss will likely be reduced, the metrics of choice for this study will be end-to-end delay and jitter. Ideally a significant reduction in both is desired.

The simulation environment will compare the two UDP implementations in a high-noise environment. Two individual terminals will exchange the Jurassic Park verbose video trace file described in the previous chapter. The scenario can be seen below. Three minutes of simulation will be performed with a high update interval for accuracy (update every 100,000 events).
The results of the simulation provide an objective means for testing the effectiveness of the proposed protocol. Provided below are the graphs of the end-to-end delay and the jitter for the original UDP and the proposed UDP. The results indicate that a reduction in
jitter of at most 5% was achieved and a reduction in end-to-end delay of 10% was achieved (on average).
Object: A of Network

average (in Video Calling Party Packet End-to-End Delay (sec))

Object: B of Network

average (in Video Called Party Packet End-to-End Delay (sec))
Figure 29: End-to-End Delay Simulation Results - Top, Original UDP, Bottom, Proposed UDP
Figure 30: Jitter Simulation Results - Top, Original UDP, Bottom, Proposed UDP
The end-to-end delay reduction is readily apparent from the graph. It is clearly seen that as the simulation proceeds, the effects of the lightweight UDP protocol proposed here become more pronounced and cause improved performance compared to the standard UDP implementation. The most telling aspect is seen approximately two minutes into the simulation, where the differences are most pronounced and become more so as the simulation progresses. Notice also in traditional UDP the end-to-end delay gap widens toward the end of the simulation. This does not occur in the lightweight UDP and thus the delays remain closer to the same values throughout the simulation. Analysis of the graph provided by OPNET yields an average improvement of approximately 10%.

The jitter reduction is less overt, and at best is only 5%. The jitter is already so low in the regular UDP and the remaining simulation that this reduction is hardly apparent in the graphs. This may seem inadequate however there is quite a bit of client-side buffering done by the UEs thus they can withstand the end-to-end delay increase of UDP in this case; however in other cases where end-to-end delay was higher, this would likely not be the case. Since only two UEs were used in this simulation, it is likely that the individual processing elements of the UMTS network could easily handle the data streams passing through and therefore virtually no jitter was seen during the simulation. The fact that jitter was reduced at all with the proposed UDP is actually an intriguing accomplishment. Since there is client-side buffering and a sufficient amount of video must come across, all the jitter indicates that the delay does not significantly change during the simulation, which is expected since the noise level is held relatively constant (albeit high). Since information was able to pass through the simulation, the jitter was relatively low since
buffering on both sides would prevent significant jitter at the cost of increased end-to-end delay. The jitter reduction that is seen with the proposed protocol is likely a result of momentary changes in delay or lost packets. Although the jitter reduction is small, the fact that any jitter reduction at all was achieved is impressive since there is client-side buffering built in to the OPNET simulation.

Thus the results of the simulation show that indeed there is an improvement in the performance of the proposed scheme over the traditional UDP approach. The purpose and stated intent of the new protocol is end-to-end delay reduction in high-noise environments and this has obviously been achieved with significant results even in a non-congested network. These results are clearly achieved with end-to-end delay reductions averaging 10%. Further since this baseline case shows improvement, further research with this protocol is warranted although large-scale simulation entails highly complex parametric calibration utilizing real-world scenarios and further research which is outside the scope of this thesis. Nonetheless with these impressive results highlighting the bare essential baseline case, there is much cause for interest in the further pursuit and exploration of this protocol.
Chapter 5  Other Advanced Improvements in Time Synchronization and Overhead Reduction for Streaming Video

This chapter explores several potential methods for improving the performance of streaming video including time synchronization enhancements and real-time transfer protocol modifications. These ideas are complicated to implement and difficult to objectively quantify the effectiveness of their results and therefore are not implemented or tested directly in this thesis, however their implementation details are given so that future work may benefit from the groundwork laid herein. Nevertheless the theoretical underpinnings and expected benefits of these approaches are provided in sufficient detail as to quell most uncertainties regarding their actual performance. Some preliminary testing is indeed done on some of the time synchronization methodologies however further work is needed to solidify these concepts as not only theoretically but also realistically sound. Overall the ideas presented here should provide a fertile ground for future work in the field of streaming video.

The first idea presented here has to do with time synchronization. In any computer system, clocks are used to determine system times and the relative time at which events occur. When multiple systems must communicate, they rely on the accuracy of their counterpart’s clock in relation to their own when communicating time-sensitive data. Often in streaming video, timestamps are used to ensure in-order delivery, such as in the
case of the popular streaming audio/video protocol, RTP [34]. In order for these timestamps to be meaningful to a third party, they must be synchronized with that third party’s clock. Ideally this wouldn’t be an issue, however all clocks run at different frequencies of operation as they deviate from the frequency of the “perfect clock.” Such a perfect clock operating eternally at the exactly correct frequency obviously exists only in fantasy, but its implications are far-reaching. Since there is no perfect clock, only clocks that aspire to be perfect, synchronization is needed at some point if meaningful timestamps are to be exchanged.

Computer clocks are usually made from quartz crystal, which must be accurately machined, with the precision being exemplified by the use of units such as parts-per-million. Even with accurate machining, a frequency deviation of 0.001% will yield a clock that drifts by one second per day [35]. From a computers’ standpoint, this may as well be an eternity as decisions are routinely made on the millisecond level and timestamps will likely need to be accurate within such a mark. Thus we have an absolute need for synchronization.

Most synchronization methods operate on the simple concept of message passing, where a sender and a receiver exchange messages and estimate the transmission time of the messages they send to provide an accurate reading of a foreign clock. Synchronization in this manner is known as Cristian’s method [36] and was introduced quite some time ago. Cristian’s method passes a timestamp from sender to receiver and then estimates the round trip time of the message as half the total trip time. This is a rough estimate and is
not useful for mediums such as wireless where round trip times are unstable and difficult to accurately predict. An improved method must avoid these difficulties to meet the challenges of today.

![Figure 31: Cristian's Remote Clock Reading Method, image from [35]](image)

The method proposed here avoids the troublesome synchronization issues such as round trip estimation by using a different message passing approach. Instead of synchronizing on a single message, the sender fires several distinct messages to the receiver with whom he wishes to synchronize. The sender knows the expected delay between the messages as this may be determined from the first packet or perhaps set as a standard value for all units to use. Now, this delay could be varied due to changes in round-trip time as a result of the instability of a wireless medium or congestion in various switching elements, but this will be averaged out later in the process. The receiver continually receives synchronization messages, likely piggybacked on top of existing data being streamed between the two nodes. The receiver computes a running average of the differences in time between the messages, effectively creating a reasonable estimate of the trip time of a message. The receiver, upon receiving the synchronization blocks, fires off a response to which the original sender may synchronize upon. This method is less invasive than outright synchronization as it can be piggybacked on other messages. Additionally the
overhead is low since a running average is used. This running average means the more communication that nodes participate in, the better their synchronization will be. Hence heavily communicating nodes such as those exchanging video information with significant detail should remain well synchronized, which is ideal for high quality video. "Sloppier" or less exact synchronization might be seen between nodes only exchanging voice data, but this is acceptable as these modes do not require as tight synchronization. This method of automatic overhead adjustment and self-adapting rate control mean the reduction in overhead should be palpable from this method.

A simple explanation of this is given and flow-chart-style pseudo-code provided in the figure below. Essentially it can be understood as the figure suggests. First, the receiver knows the difference in the send times of the repeated synchronization messages (all of which contain the current time according to the sender). Thus when the receiver gets the second message, and the third message, and so on, it can deduce the time that the message took to transmit by adding 10 ms to the initially received time and comparing this against the current time (according to the sender). The receiver then takes a running average of these to estimate the round-trip time accurately. After a sufficient number have been collected, the receiver will synchronize with the sender. Additionally the uncertainty may be gathered based on the difference in the received times of the messages. Messages that delay more than 10 ms are known to have been unnaturally delayed during transmission. Messages that appear to arrive "sooner" than 10 ms were simply not impeded in the same way (or perhaps at all) that the previous messages were.
The pseudo-code for this style of operation can be seen below, although its contents are essentially as described previously. The operation is relatively straightforward. Note that this method assumes good prior synchronization. In order to function effectively, the messages need to have a baseline known good time to start from. There are numerous available methods for doing initial synchronization quickly and effectively, this method simply keeps clocks continuously synchronized with very little effort. As the pseudo-code explains, a delay measurement is taken and stored for standard deviation calculations later, which yield the overall uncertainty of the obtained synchronization. This uncertainty is due to variations in message delay between sending of messages. The running average can continuously be taken and the clocks kept synchronized in this fashion. The figure below explains a single synchronization step more clearly.
Receive_message( msg );
If( msg.isKeepSync() ) {
    // get next sync message
    Old_rcv_time = msg.time(); // the time inside the message
    Receive_message( msg );
    msgcnt = 1;
    // begin message reception loop
    While( msg.isTimingPacket() ) { // while the message is a
        // time sync update
        // update delay collection with new information
        Delay.store(msg.arrivalTime() - Old_rcv_time);
        // trip time is now minus when the message was sent
        Avg_triptime = currentTime() - msg.time();
        Old_rcv_time = msg.arrivalTime();
        msgcnt++;
    }
    // get final average delay
    Uncertainty = Stdev( delay ); // the uncertainty calculation
    Avg_triptime /= msgcnt; // average trip time
}

Figure 34: Time Synchronization Pseudocode
An initial and simplistic implementation of this method was done on mica2 TelosB motes and some very basic results gathered. Synchronization in this method was found to be somewhat localized and short-lived. Results showed that, on average, accuracy within 2 ms was reasonable under low message passing loads. These results are neither exceptionally good nor bad, and additionally the testing was done under near-ideal conditions. The synchronization was short-lived in the sense that drift was readily apparent immediately if message passing stopped since the synchronization relies on continuous reception of passed messages. The localization of the approach indicated that the synchronization exists specifically between two nodes and thus is not easily transferable to another third party. These preliminary results mean the method is reasonable for use in a voice/video transmission situation between two nodes, but not useful for widespread synchronization. This is acceptable for the stated purposes; however an ideal algorithm would be adaptable to many different applications and scenarios. Nonetheless this method has met with reasonable success in its initial testing.
Future work on this synchronization method should focus on adaptability and reduction of overhead. The overhead primarily introduced here is done so due to the accuracy required in the timestamp. Future revisions could easily circumvent this by making the timestamps relative to the previous, thereby requiring only a single, initial timestamp exchange and a series of smaller timestamp updates. Overall this is a promising new method that should be investigated further to discover what specific and quantifiable advantages it may have over existing methods.

Currently UMTS takes advantage of a protocol known as RTP, or Real-Time Transfer Protocol. This is a protocol which can be attached to the application layer or the transport layer, depending on implementation [34] and provides the following services according to RFC 1889:

- Sequence Numbers
- Time Stamping
- Payload identification
- Delivery Notification

Through these benefits RTP is said to be useful in real-time video and audio streaming due to its ability to correctly identify data to allow various QoS protocols to adjust network operation accordingly. [38] The actual header format of RTP is shown below. This is a well-entrenched technology, and forms the technological foundation of VoIP. [40] However, this technology is typically insensitive to packet loss but very sensitive to delays according to [41]. This makes it ideal to be used in conjunction with the
previously proposed UDP scheme, and herein is proposed an idea to improve the mixed performance of the two protocols.

One possible way to improve performance at the expense of features is to remove the SSRC and CSRC fields. Currently, a synchronization source or SSRC is a 32-bit randomly assigned identifier that allows a source of RTP packets to be identified without relying on a network address. The contributing source list (CSRC) comprises up to fifteen 32-bit identifiers which contributed to a payload. For example, in a VoIP conversation these could be the individuals participating in a teleconference phone call. The receiving protocol could then extract the voice of each member individually to raise or lower volumes or mute certain members accordingly. Although this functionality is certainly useful, it does not have much direct use when streaming video, even if the video is coming from multiple sources. The SSRC identifier is certainly useful, but significant amounts of overhead can be saved by eliminating this part of the header while
maintaining the sequence numbering and payload identification. Thus the slimmed down header does not require the CC field (which is the number of CSRC packets) and is thus even smaller. To maintain backwards compatibility with earlier RTP protocols, the version bits (V) could be set to 111 to indicate that the reduced header RTP is in use. This would alleviate much of the overhead that RTP requires.

Of course, to implement this in reality requires a significant effort since RTP receptors must be reprogrammed to correctly interpret the new packet format. Furthermore the benefits of this approach are not currently known. Future work may implement these ideas to ascertain their benefits in a real-life scenario; however the fact that overhead reduction is seen by removing features that do not positively affect performance suggests that the end result would likely be correct functionality alongside positive end-to-end delay and performance improvements.

One final proposal to enhance performance from the end-user’s point of view lies in the concepts of Forward Error Correction (FEC). Now, FEC approaches indeed add overhead [42] which may cause the concept of their implementation affecting a performance improvement seem illogical. However, recalling that the new UDP protocol established here relies on acceptance of packets with some corruption, using an error-controlling code such as an FEC approach will cause less distortion on the user’s end as the “averaging of noise” approach taken by an FEC will cause overall less distortion. However, it remains to be seen whether or not such an approach will affect a reasonable improvement in video quality since MPEG-4 already contains significant amounts of
error correction and tolerance capabilities. However, there is certainly much hope for such an approach. The end effects, of course, require further study to fully understand.

Unfortunately these approaches require further study and much more time to fully implement in order to get a working prototype and simulation running. Once this is accomplished and the groundwork is laid, future work can begin and show the benefits of these approaches. In this chapter, a discussion of lightweight time synchronization methods for the benefit of streaming video was discussed along with methods for implementation and various adjustments for the improvement of performance. Methods for synchronization payload delivery were discussed, such as piggybacking. A technique for reduction of RTP overhead and decreasing of delays associated with using RTP was discussed. By discarding functionality not directly related to video and audio transmission performance, significant amounts of bandwidth can be spared, opening up that bandwidth to be used more effectively. Finally the use of forward error correction for the potential for improvement of video performance was discussed, and some basic underlying concepts explored. Overall this section provides a framework upon with performance enhancements may be based in future work and should yield many new useful protocols and developments.
Chapter 6 Conclusion

Streaming media over wireless and cellular networks is an exciting and fascinating upcoming field with diverse challenges and fertile ground for new ideas. The boom of the telecommunications industry with regards to cellular telephones and a general shift from household to portable electronics has fueled a trend toward interactive portable multimedia devices. With this new technology come the challenges of reliable content delivery, acceptable end user experience and maintenance of high video quality even under stressful network conditions. With the immense user base that cellular telephones already have and the network conditions that already exist, providing an entirely new multimedia experience for the end user is an important and difficult new problem. Furthermore, the Universal Mobile Telecommunications System promises to be the next big cellular telephone movement with worldwide communications possible and improved data rates. Being so new, it is not known if this technology can withstand the demanding network conditions brought about by streaming video and other network-intensive streaming multimedia. This thesis has shown through comprehensive simulations that there is no specific bottleneck in the network which could cause serious breakdown during high traffic conditions, reinforcing the idea that UMTS is capable of handling even severe traffic loads. This thesis also proposed a modification to the UDP transport layer which will provide improved end-to-end delay and decreased jitter. Complex simulation results yielded a maximum 10% drop in end-to-end delay and a maximum 5% drop in jitter. Finally the thesis closed with several new ideas to improve the subsystems within streaming video. These ideas included time synchronization and protocol improvements. The topic of streaming media is currently very active and an incredibly
rich and diverse area of research. This thesis has provided several new ideas that should prove useful in the continual development of this amazing field.
Bibliography


[14] Ibid.


[19] Ibid.


[40] Ibid.
