Wireless Communication Options for a Mobile Ultrasound System

By:

Brett William Dickson

A Thesis

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Electrical Engineering

by

____

August 2008

APPROVED:

____

Prof. Peder C. Pedersen, Major Advisor

____

Prof. Alexander M. Wyglinski, Committee Member

____

Prof. Andrew G. Klein, Committee Member

Abstract

A mobile ultrasound system has been developed, which makes ultrasound examinations possible in harsh environments without reliable power sources, such as ambulances, helicopters, war zones, and disaster sites. The goal of this project was to analyze three different wireless communication technologies that could be integrated into the ultrasound system for possible utilization in remote data applications where medical information may be transmitted from the mobile unit to some centralized base station, such as an emergency room or field hospital. By incorporating wireless telecommunication technology into the design, on site medical personnel can be assisted in diagnostic decisions by remote medical experts.

The wireless options that have been tested include the IEEE 802.11g standard, mobile broadband cards on a 3G cellular network, and a mobile satellite terminal. Each technology was tested in two phases. In the first phase, a client/server application was developed to measure and record general information about the quality of each link. Four different types of tests were developed to measure channel properties such as data rate, latency, inter-arrival jitter, and packet loss using various signal strengths, packet sizes, network protocols, and traffic loads. In the second phase of testing, the H.264 Scalable Video Codec (SVC) was used to transmit real-time ultrasound video streams over each of the wireless links to observe the image quality as well as the diagnostic value of the received video stream.

The information gathered during both testing phases revealed the abilities and limitations of the different wireless technologies. The results from the performance testing will be valuable in the future for those trying to develop network applications for telemedicine procedures over these wireless telecommunication options. Additionally, the testing demonstrated that the system is currently capable of using H.264 SVC compression to transmit VGA quality ultrasound video at 30 frames per second (fps) over 802.11g while QVGA resolution at frame rates between 10 and 15 fps is possible over 3G and satellite networks.

Preface

I would like to thank the Telemedicine and Advanced Technology Research Center (TATRC) for providing essential resources throughout the duration of this research. This thesis work was supported under Contract No. DAMD17-03-2-0006, ”Real-Time Troop Physiological Status Monitoring System Using a Common Wireless Network", from the TATRC. The financial support is gratefully acknowledged.

Additonally, I would like to acknowledge Philip J. Cordeiro for the valuable tutelage and assistance that was provided, especially in the beginning stages of this project. Lastly, the mentoring and guidance provided by Professor Peder C. Pedersen was essential to the success of this research work, and is greatly appreciated.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Ultrasound imaging is a safe medical imaging modality that uses sound waves to allow the observation of internal anatomical structures, such as tissues and organs. Ultrasound imaging works by emitting impulses of sound energy along thin acoustic beams into the human body, and reconstructing the echoes of the original sound wave into a viewable image. Compared to other medical imaging technologies, such as CAT scans, MRI and X-Ray, ultrasound imaging can most easily be adapted to a portable environment due to relatively low power and size constraints. As part of an on-going research project, a mobile ultrasound system has been developed that can be housed in a number of different configurations such as a wearable vest or a small handheld bag. For this reason, it has been named a “reconfigurable ultrasound system.”

The reconfigurable unit allows for imaging to take place in environments lacking stable power sources where ultrasound technology has not previously been a possibility. Some of these environments include ambulances, disaster sites, war zones and rural medicine. In most of these settings, it may be necessary to transmit image data from the mobile ultrasound unit to some base station, either for long term storage, or to be viewed by a more highly experienced physician. Because ultrasound is an interactive imaging method that requires training and experience on the part of the ultrasonographer, guidance from experienced medical personnel will greatly benefit the remote sonographer who may not be sufficiently skilled in ultrasound.

This thesis enables expansion of the original ultrasound design by examining a number of wireless transmission possibilities that could be employed in remote data transfer applications. The wireless options that were analyzed include the IEEE 802.11 standards, mobile broadband cards on a 3G cellular network, and lastly, a satellite network. To perform a thorough analysis of each wireless option, two test phases were conducted. In the first phase, basic channel characteristics, such as bandwidth, latency and packet loss were measured for each communication option. In the second phase, ultrasound images were transmitted in real-time to obtain a qualitative assessment regarding any degradation in received image quality due to constraints exerted by the wireless link.

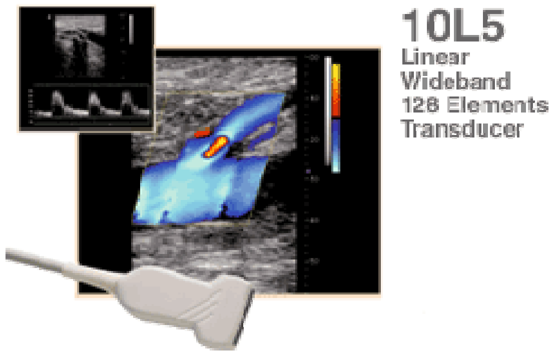
In this introduction chapter, background information relevant to the thesis is presented. Topics such as ultrasound imaging technology and telemedicine are discussed. Also, a complete overview of the reconfigurable ultrasound design is given.

1.1 Ultrasound Technology Background

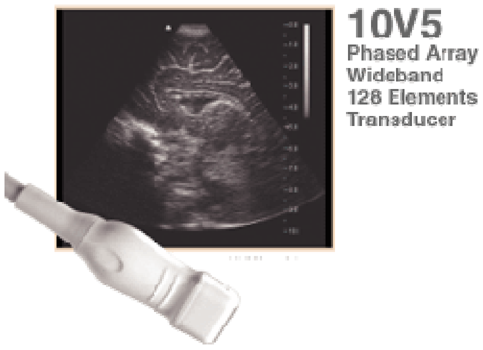
Ultrasonography is a medical imaging technique used to visualize internal anatomical structures in the human body such as muscles, tissues and organs, as well as identify the presence of trauma, injuries and fluid accumulation. To obtain an ultrasound image, pulses of sound waves are emitted into the body by means of an ultrasound array transducer containing a large number of array elements. Echoes are then produced and reflected back to the transducer whenever the sound waves encounter interfaces between organs or tissue structures exhibiting changes in acoustic impedances. The greater the difference between acoustic impedances, the larger the echo. The depth (or range) of the tissue interface producing the echo can be determined by measuring the time between the transmission of the incident sound pulse and the reception of the echo from the tissue structures.

Although the term ultrasound refers to acoustic energy with frequencies greater than the threshold of human hearing (20 KHz), the frequency range used in diagnostic ultrasonography is generally between 1 and 18 MHz [1]. Different types of transducers are used depending on the type of exam being performed. Transducers can differ in the number of array elements in the head of the probe, the shape of the probe as well as the frequency range of the emitted sound pulses. Figure 1 displays some common ultrasound transducers that are used today.

(a) 4-2 MHz Convex Array Transducer [2]



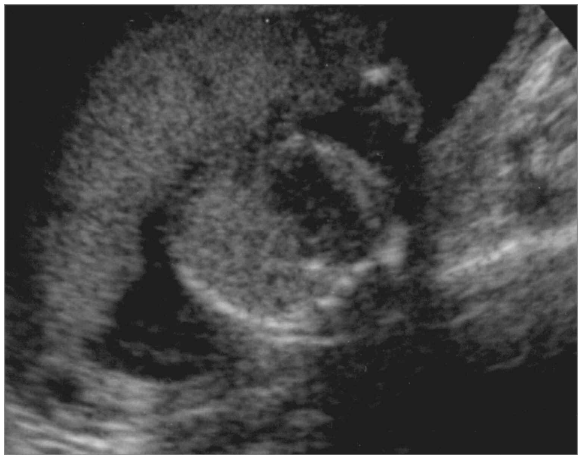
(b) 10-5 MHz Linear Array Transducer [2]



(c) 10-5 MHz Phased Array Transducer [2]

Figure 1: Example Ultrasound Transducers

As the frequency of the sound waves increases, the size of the corresponding wavelength will decrease. This leads to higher resolution imaging; however, the higher frequencies are not able to penetrate as deeply into the body as lower frequencies. For this reason, superficial structures such as muscles, tendons, testes and breasts are imaged at higher frequencies, generally between 7 and 18 MHz. Deeper structures, such as liver and kidneys, require lower frequencies (1 to 6 MHz), thus leading to poorer resolution [1]. Figure 2 shows two ultrasound images taken at different frequencies. It is evident that the resolution of the image on the right, which was taken at a higher frequency, is more detailed than that of the image on the left.


(a) 3.5 MHz Ultrasound Image [3]

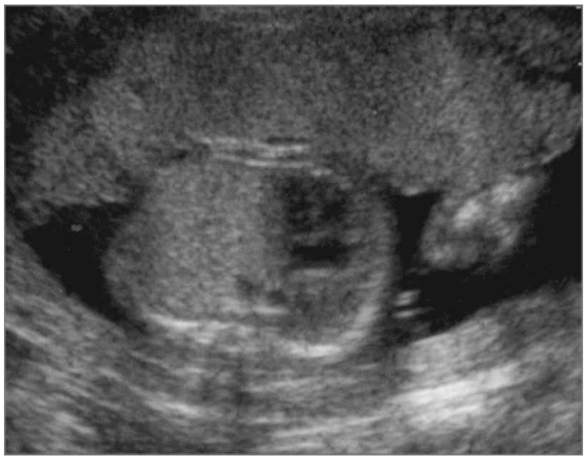
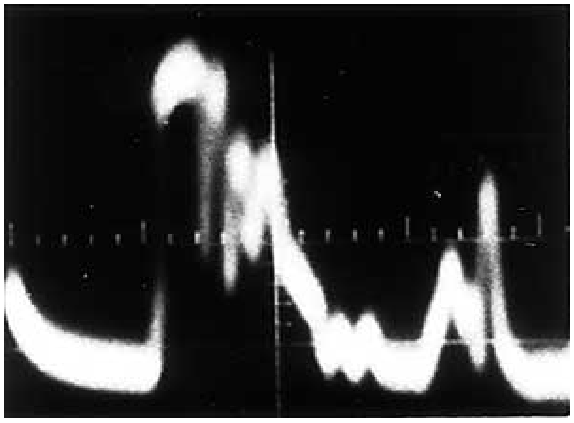
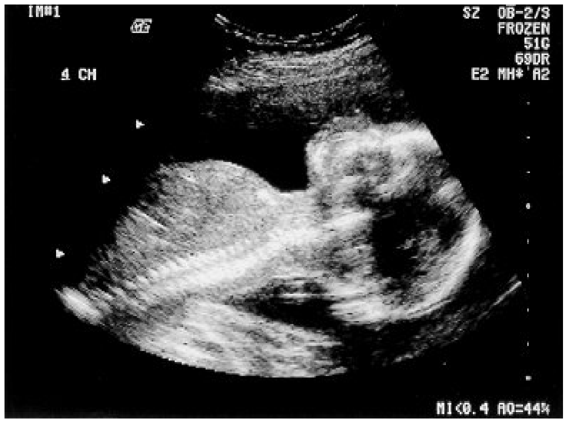

(b) 5.0 MHz Ultrasound Image [3]

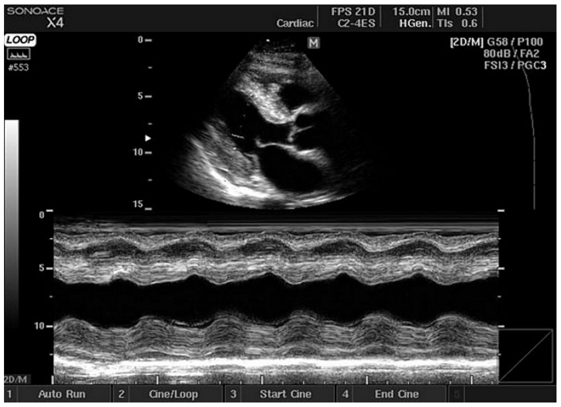
Figure 2: Comparison of Ultrasound Images at Different Frequencies

In medical ultrasonography, four primary imaging modes are used. The first mode, which was developed in the 1950's, is called *A-mode* where the "A" stands for amplitude. A-mode, which is the simplest type of ultrasound image presentation, uses a single transducer head to place a single scan a line through the body. The image is constructed by plotting the envelope of the received RF echo as a function of depth [4]. The next imaging mode, called *B-mode* or brightness mode, improves upon A-mode imaging by using a linear array of transducer elements to steer the ultrasound beam, creating scan a plane through the body [4]. The result of the received echoes is a two-dimensional image of the scanned plane with image brightness representing the amplitude of the echoes.

In *M-mode* imaging, where the "M" stands for motion, successive B-Mode images are acquired, allowing the sonographer to observe how points along a given scan line behave as a function of time [1]. This imaging mode is useful when examining organs that are in motion such as the heart valves. The last common imaging technique is called *Doppler Mode*. This mode takes advantage of the Doppler effect that occurs when a sound wave encounters a moving object. The movement of the structure will produce a Doppler shift in the frequency of the returned echoes. There are a few different imaging techniques that take advantage of the Doppler Effect including Power Doppler, Color Doppler and Pulsed Wave Doppler. Most of the Doppler imaging techniques are used to characterize blood flow in vessels and tissues [4]. Figure 3 shows an ultrasound image acquired using each of the aforementioned imaging modes.


(a) A-Mode Ultrasound Image [5]


(b) Prenatal B-Mode Image [6]


(c) M-Mode Image of Heart [7]

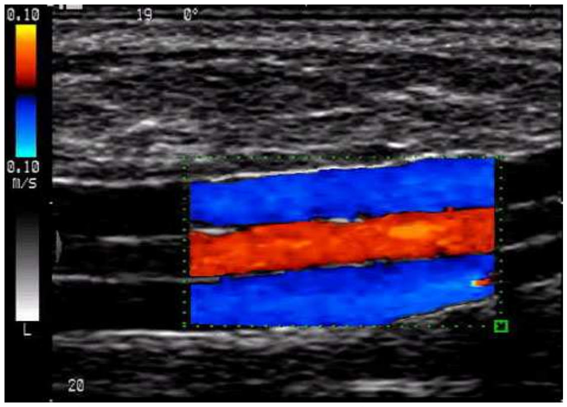

(d) Color Doppler Image of Carotid Artery [8]

Figure 3: Ultrasound Images in Different Imaging Modes

Recent advances in ultrasound imaging has led to the possibility of combining different imaging modes for certain types of exams as well as using multiple scan planes to produce three and four-dimensional ultrasound images. Other relatively recent new imaging methods include tissue harmonic imaging, imaging with contrast agents, and tissue elasticity imaging. Ultrasound is a useful imaging tool as it rarely inflicts any discomfort to the patient and does not have any known side effects. Ultrasound imaging is commonly used in the following medical specialties [1]:

- Cardiology
- Endocrinology
- Gastroenterology
- Gynecology
- Obstetrics
- Ophthalmology
- Urology
- Musculoskeletal (tendons, muscles, and nerves)
- Vascular studies
- Emergency Medicine
- Surgery

1.2 Telemedicine

The term *telemedicine* has adopted a number of different definitions throughout its short history. Evolving communication technologies has been a central factor in the ever-expanding uses of telemedicine. Currently, a widely recognized definition of the term is, “the provision of healthcare services, clinical information, and education over a distance using telecommunication technology [34].”

Currently, telemedicine is a rapidly growing market both in the United States and globally. As of 2006, about 3,500 hospitals, schools and other facilities were using some form of telemedicine, which represents a 75% increase from the year 2000 [32]. There have been many documented cases of successful telemedicine applications throughout recent decades. Initially, telemedicine was used primarily in applications where traditional healthcare services could not be provided, such as disaster relief, mobile military camps, and rural health centers. The telemedicine market figures to continue to grow as healthcare providers are attempting to use telemedicine as a practical alternative to traditional office visits. Future uses of telemedicine will enable patients with chronic diseases an efficient way to receive medical services and could perhaps empower patients to become participants in managing their own health [32]. This section will provide a brief history of the evolution of telemedicine citing some specific cases of telemedicine operations. It will then present a taxonomy of the different dimensions of telemedicine which can be used to distinguish how one application differs from another.

1.2.1 History

One of the main factors in the evolution of telemedicine has been advancements in the development of telecommunications technology. As the capacity and reliability of communication channels have progressed over recent decades, so have the practical uses of telemedicine. Another factor that has contributed to the growth of telemedicine has been the miniaturization of computers and electronic devices. Computer miniaturization has made it possible for medical instruments to become portable, and in some cases wearable, which has created opportunities to countless new telemedicine applications [33].

In 1956, Wittson and Dutton from the Nebraska Psychiatric Institute used closed circuit television to transmit live therapy sessions to students. The initial purpose of the project was to educate students; however, additional applications were soon developed that allowed the university's psychiatric department to interact with a state mental institution about 100 miles away [31]. In a high profile case in the 1960's, the National Aeronautics and Space Administration (NASA) used telecommunication technology to monitor the health of astronauts during space missions [35].

In the 1970's, many pilot projects were started through government funding, but most programs were terminated before they had a chance to become mature [34]. In one such case called the *Logan Airport Project*, Massachusetts General Hospital was linked to the medical center at Logan airport via a microwave connection. The purpose of the project was to deliver primary and specialist care to airport employees [31].

In the 1980's and 1990's, advancements in telecommunication technology allowed for more opportunities to use telemedicine as a practical method to deliver healthcare. One of the most popular areas of telemedicine during this time period was disaster relief in both the civilian and military sector. In 1985, NASA used the Advanced Technology-3 (ATS-3) communications satellite for voice communications during a disaster aid effort following an earthquake in Mexico City. The ATS-3 link was vital in this effort because all land-based forms of communications were destroyed during the earthquake [33]. After Hurricane Hugo hit the Virgin Islands in March of 1990, the Alabama Army National Guard Mobile Surgical Hospital used a prototype Battlefield Computed Radiology scanner during the relief effort. They used an International Maritime Satellite (INMARSAT) terminal to transmit images acquired from the scanner to hospitals in Washington, D.C. and Georgia for medical support [33].

With the advent of digital networks such as the Integrated Services Digital Network (ISDN), telemedicine applications expanded from isolated pilot projects in countries with advanced communications technology to developing nations that desperately require medical care [34]. In one case from 1996, the United States Department of Defense established a medical network in Bosnia that connected remote medical centers to hospitals in the U.S. The project used an ISDN network structure to send X-rays, ultrasound, CT scans and other medical images to field hospitals for diagnostic support [33].

Today, rapid advancements in both wired and wireless communication technology are paving the way for telemedicine to become a practical option for many countries and organizations. For example, the *British Lancashire Ambulance Project* uses multiple 3G cellular lines to transmit images to a hospital. The project, which was developed for ischemic stroke, uses one cellular line to transmit vital signals while another transmits slow-scan images at about 15 frames per minute [35]. Telemedicine has found a niche in most all medical fields, and should continue to evolve and become even more prevalent in upcoming years. The next section will examine the various dimensions of a telemedicine application.

1.2.2 Taxonomy

Telemedicine applications can be vastly different depending on the field of use and reason for the medical effort. For this reason, Chatterjee *et al*. [34] propose a taxonomy of telemedicine that is aimed at identifying the various dimensions of telemedicine and telehealth. This taxonomy is helpful when trying to classify a telemedicine effort and determine how one telemedicine application differs from another. The five dimensions that make up this taxonomy are: Application Purpose, Application Area, Environmental Setting, Communication Infrastructure and Delivery Options.

Application Purpose

The *Application Purpose* is the reason that the exchange of medical information is necessary in the first place. Generally, the application purpose falls into one of two categories: clinical or non-clinical. Clinical applications refer to actual medical situations where decisions regarding the care of a patient must be made. The *Committee on Evaluating Clinical Applications of Telemedicine* published a report in 1996 classifying clinical usages of telemedicine into the following six categories [38]:

- Triage / Initial urgent evaluation
- Supervision of primary care
- Provision of specialty care
- Consultation
- Monitoring
- Use of remote information to support or guide care for specific patients

Since then, advancements in technology are paving the way for a patient to be cared for through communication channels without the intervention of a local supervisor. Although this is currently not a common application in telemedicine, it looks as if it could be a popular clinical purpose in the future [34].

Non-clinical purposes refer to cases that do not involve decisions about care for patients. Some non-clinical applications include professional medical education, patient education, research, public health and administrative. Although there may be possible non-clinical applications using WPI's mobile ultrasound system, this thesis is focused more on the clinical purposes of the system.

Application Area

The *Application Area* refers to the medical field in which care is being provided. Both the type and amount of medical information that must be exchanged depend greatly on the medical field in use. Some areas may require visual or audio data while text may be sufficient in other areas. For example, the type of data required in a psychiatric medical application is likely much different than that required in the obstetrics domain. The following is a list of some possible application areas for telemedicine; however, it is by no means a comprehensive list.

- Neurology
- Cardiology
- Radiology
- Pediatrics
- Surgery
- Pathology
- Psychiatry
- Dermatology
- Obstetrics
- Gynecology
- Rheumatology
- Otolaryngology

Environmental Setting

The *Environmental Setting* refers to the physical environment that the physician or patient will be using during the telemedicine procedure. In the majority of telemedicine applications, data is transmitted from the treatment site to some centralized base station either for storage or professional review. Most of the time, a large scale hospital serves as the base station because of the vast medical resources available at the site. The site from which data must be sent can vary greatly depending on the telemedicine scenario. For example, in triage efforts, medical information may be sent while the patient is being transported to a medical facility. In this case, the environmental setting could be an ambulance, helicopter or mobile telemedicine vehicle (MTV). In other applications, a rural health center or a navigating sea vessel may serve as the environmental setting of the telemedicine event. In any case, the most important factor to look at when evaluating the setting is the physical distance between the two locations. The range between both of the sites will narrow down the communication possibilities that exist when exchanging data. Figure 4 illustrates some of the different environmental settings that may come up in various telemedicine applications.

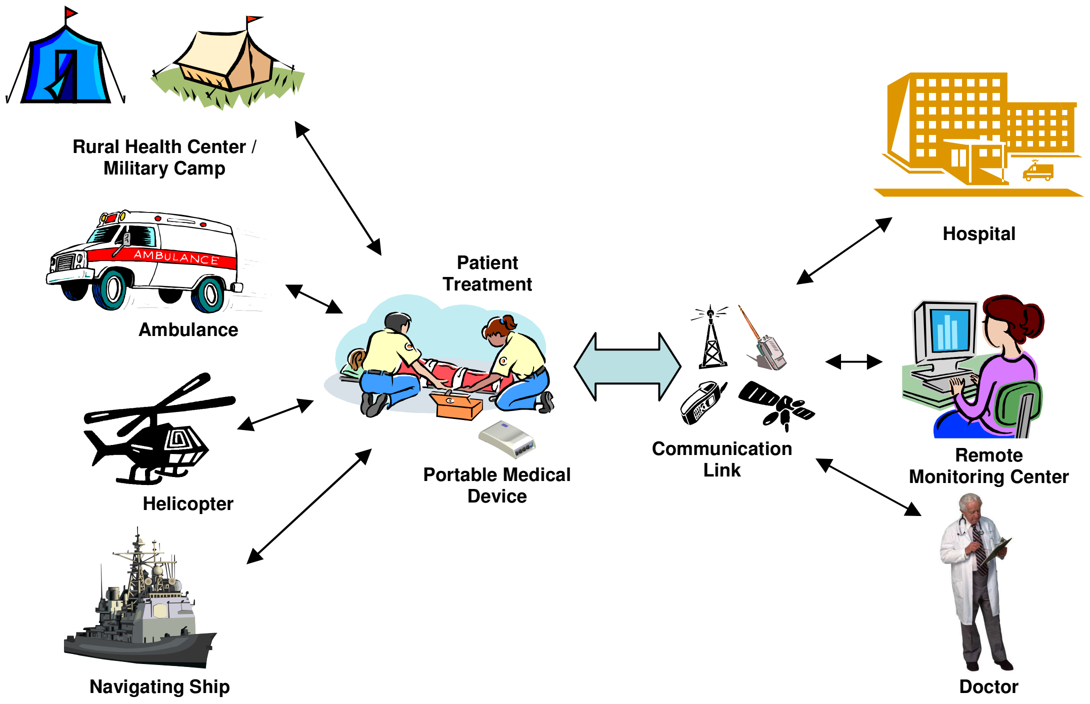


Figure 4: Possible Environmental Settings in Telemedicine

<u>**Communication Infrastructure**</u>

The *Communication Infrastructure* refers to the telecommunication channels that are available to transmit and receive data. Communication infrastructures are either wired or wireless depending on the type of telecommunication technology being utilized. Additionally, a combination of wired and wireless networks is possible Wired networks use twisted pair cables, coaxial cables or fiber optic lines while wireless technologies utilize radio frequency waves to send and receive data. Many times, the telemedicine application will dictate the type of telecommunication technology that is required. For example, it would be impossible for a helicopter or ship to send data through a wired network to a hospital. On the other hand, it would be senseless for a rural health clinic with wired internet connectivity to use wireless technology to send medical information to the base station.

The most important characteristic of the communication infrastructure for telemedicine applications is its bandwidth. The amount of available bandwidth on a channel will be the limiting factor for the type of services that can be performed on the network. Insufficient bandwidth may make it impossible to perform high quality network applications such as audio conferencing or streaming video. Other network characteristics such as latency and jitter will directly affect the quality of such applications.

Since wired telecommunication channels are typically more reliable than wireless channels, wired infrastructures are generally used when possible. Although they are more reliable, wired infrastructures normally come at a higher cost than wireless technologies; especially if dedicated lines need to be deployed. Unlike wireless networks, physical distance is not a major concern for wired links. Although there will always be some amount of signal degradation over a wired medium, repeaters and hubs can be used to regenerate the transmitted signal. The following table shows some popular wired telecommunication technologies along with their maximum supported bandwidths.

Table 1: Wired Telecommunication Technologies [39]

Technology	Bandwidth
ISDN (Dial Up)	64 kbps
DSL	64 kbps – 1.544 Mbps
T1	1.544 Mbps
T3	44.7 Mbps
Ethernet	10/100/1000 Mbps
Fiber Optic Cable	1 Gbps +

When wired infrastructures are not available, wireless links can serve as an alternative. The main concern with wireless links is the physical range that data can be sent. Depending on the electromagnetic frequency being used, different wireless technologies have different physical ranges that data can reliably be sent and received. Other concerns for wireless technologies include interference from other signals in the same frequency range, as well as signal fading due to multipath effects [40]. Both of these events will cause signal degradation at the receiver decreasing the overall reliability of the link. The following table shows some of the current wireless technologies available for telemedicine applications. Also included in the table are some technologies that are either experimental or still under development, and may be utilized in future telemedicine applications.

Table 2: Current and Future Wireless Telecommunication Technologies [36] [17] [41] [42]

Technology	Standard		Max Downlink (Mbps)	Max Uplink (Mbps)	Range	Typical Downlink (Mbps)	Current Wide Deployment
WiFi	802.11b		11.0	11.0	~30 m	2	YES
	802.11g		54.0	54.0	~30 m	10	YES
	802.11n		200.0	200.0	~50 m	40	NO
2G Mobile Data	GSM	GPRS Class 10	.0856	.0428	~16 mi	.014	YES
		EDGE Type 2	.4736	.4736	~16 mi	.034	YES
		EDGE Evolution	1.8944	.9472	~16 mi	-	NO
	cdmaOne	IS-95	.1152	.1152	~16 mi	.0144	YES
3G Mobile Data	UMTS W-CDMA	HSDPA	14.4	.3840	~18 mi	1-2	YES
		HSUPA	14.4	5.760	-	-	NO
		HSPA+	42.0	11.5	-	-	NO
	CDMA 2000	RTT 1x	.3072	.1536	~18 mi	.125	YES
		EV-DO Rev 0	2.458	.1536	~18 mi	.75	YES
		EV-DO Rev A	3.1	1.8	-	-	NO
		EV-DO Rev B	4.9	1.8	-	-	NO

Technology	Standard		Max Downlink (Mbps)	Max Uplink (Mbps)	Range	Typical Downlink (Mbps)	Current Wide Deployment
4G Mobile Data	WiMAX	802.16	70.0	70.0	~4 mi	>10	NO
	HIPERMAN		56.9	56.9	~4 mi	-	NO
	WiBro		50	50	~900 m	-	NO
	iBurst	802.20	64	64	3 – 12 km	-	NO
	UMTS	LTE	>100	>50	-	-	NO
Satellite	Low Earth Orbit		.0386	.0386	Global	-	YES
	Geostationary		.492	.492	Global	-	YES

In Table 2, the theoretical maximum uplink and downlink bit rates are shown; however, it is unlikely to achieve these data rates in practice. Many factors such as interference, operating environment (indoor vs. outdoor), network overhead and surrounding structures can affect each technology differently, and will cause often data rates to be significantly lower than the theoretical limits. Also, some technologies such as the IEEE 802.11 standards change the modulation schemes used to code the data based on the amount of power that is received, which will result in lower data rates. The column labeled "Range" provides the maximum range possible to receive data at approximately 25% of the given typical rate.

The IEEE 802.11 standards are designed for local area networks (LAN), and are characterized by high data rates, and relatively short ranges. The 2G and 3G mobile data networks are aimed at wide area networks (WAN), and generally have lower data rates, but greater ranges than the 802.11 standards. Unlike the WiFi technologies, the mobile data networks are normally deployed in a cellular topology with a cell tower providing service to all users within the radius of its transmission range. Currently, 4G technologies are being developed, but have yet to be widely deployed. The high data rates of these upcoming technologies will most certainly make them useful in future telemedicine efforts. Satellite technology provides global coverage; however, these channels are generally characterized by high latency and low bandwidths that can render them unsuitable for certain applications. One wireless technology that is not shown in Table 2 is packet radios. Data or packet radios can vary in throughput and range based on a number of factors such as the frequency range of operation, the amount of power transmitted and the modulation techniques used to code the data. Wireless Personal Area Network (WPAN) technologies such as *Bluetooth* and *ZigBee* were left off the list because they have very short ranges (<10 m) and are not particularly good options for most telemedicine applications.

Delivery Options

The last dimension in this taxonomy of telemedicine is *Delivery Options*. Delivery options refer to the type of application that must be run over the telecommunication channel to effectively transmit patient data. The most common types of information exchanged in telemedicine applications include audio, still images, video, and text. The type of medical procedure coupled with the information needed by the base station will dictate exactly which types of data will be necessary. Each data option will have different bandwidth and compression requirements for successful transmission to a base station. For the majority of telemedicine applications, the delivery options fall into one of two categories: store-and-forward or real-time. Store-and-forward systems allow the base station to download data that has been pre-stored on the remote device while real-time options allow the base station to interact with the remote system in real-time. Table 3 gives examples of different store-and-forward and real-time delivery options that are widely used in telemedicine.

Table 3: Example Delivery Options [34]

	Store and Forward	**Real-Time**
Data	• Email • Web Pages • Pre-stored text • Pre-stored image	• Instant messaging • Chat room
Audio	• Voicemail • Pre-stored audio clips	• Telephone • Live two-way audio stream
Video	• Pre-stored video clips	• Live video stream • Videoconferencing

Live two-way voice transmission is a common data type that may be necessary in some telemedicine applications. While some organizations may already have a method for voice communication in place such as radios or cell phones, others will have to rely on the portable medical unit for voice transmission. The bandwidth requirements for live voice transmission vary depending on the coding algorithm used to digitize the speech. High quality voice channels require upwards of 40 kbps while the absolute minimum bandwidth needed for lower quality calls is around 8 kbps including overhead [14].

Real-time one-way video transmission will also be a necessity in a wide variety of telemedicine applications. There are multiple factors that will dictate the bandwidth requirements necessary to send live video. One major factor is the amount of compression that can be tolerated without compromising the integrity of the data. If the source video undergoes too much compression, the quality at the receiving end may be too low make an accurate diagnosis or analysis of the video and therefore rendered useless. Other factors, such as the resolution and frame rate, also influence the required minimum bandwidth of the channel. For these reasons, it is difficult to estimate the bandwidth requirements for streaming video without knowledge of the precise telemedicine application.

Store-and -forward systems will have much more lenient bandwidth requirements than real-time systems. In store-and-forward applications, data is saved locally on the portable medical unit and made available to be downloaded by a remote host. This may be useful in applications where information must be sent to a base station for long-term storage or for review by a more highly skilled physician. The static files generally take the form of patient information files, still images, audio recordings, or video clips. The reconfigurable ultrasound system, for which this project is developed, is capable of saving and transmitting all of the aforementioned data types as well as transmitting real-time voice and video.

1.3 Portable Ultrasound

Until recently, ultrasound imaging only took place in clinical settings such as hospitals or rural health centers. The recent development of mobile imaging systems has lead to a number of emerging applications of ultrasound in telemedicine. Ultrasound imaging has proved to be very useful in pre-hospital settings where the patient is in the process of being transported to a hospital environment; often an emergency department. For example, injuries such as bleeding within the abdomen or a pneumothorax condition (collapsed lung) can be identified by using pre-hospital ultrasound imaging [4]. Knowledge of these types of conditions will often change the course of treatment taken by medical personnel, increasing the chances of successful recovery of the patient.

Routine medical care can also benefit from mobile ultrasound systems outside of a clinical environment. For example, certain high-risk pregnancies requiring regular ultrasound examinations could be performed away from a larger hospital, in smaller clinics or even at the home of the pregnant woman. Also, many rural areas that are not located near facilities equipped with ultrasound equipment. Patients that live in these areas must often travel great distances for routine care and could greatly benefit from being examined with a mobile ultrasound imaging system.

Up until 1999, ultrasound technology was not possible in telemedicine applications because most of the systems were cart-base and intended for clinical or hospital settings. Figure 5 shows a typical cart-based ultrasound system used in hospitals today.

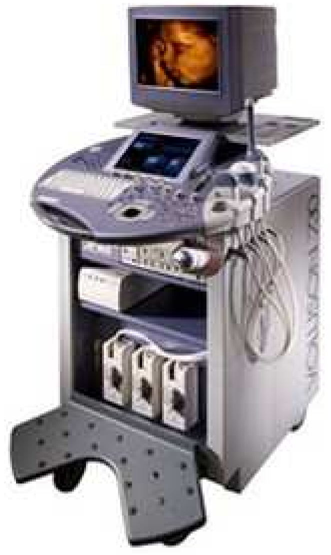


Figure 5: GE Voluson 730 Cart Based Ultrasound System [10]

In 1999, the company *SonoSite* released the first portable ultrasound system called the *SonoSite 180,* which is shown in Figure 6. In 2001, Teratech released the first PC based portable ultrasound system, which was the Terason 2000. Since then, a number of different manufacturers have released portable ultrasound units in different sizes and form factors, the most notable being GE [4]. These new portable ultrasound devices allowed for the ultrasound sonographer to bring the equipment to the patient rather that bringing the patient to the ultrasound system. This new technology opened the door for ultrasound in telemedicine; however, most of the portable machines were not built to withstand harsh outdoor use and exposure to elements such as rain and dust. Additionally, the battery life of most of these portable units was not long enough to be used for more than one or two hours at a time. For these reasons, a research effort began at WPI to develop a portable ultrasound unit that could be used in the harshest of telemedicine applications.

Figure 6: SonoSite 180 Plus Portable Ultrasound System [10]

1.4 WPI's Reconfigurable Ultrasound Systems

In an on-going research effort, a number of design attempts have been made to develop a mobile ultrasound system that can be used in environments that may lack stable power and/or communications infrastructures. To date, three generations of mobile ultrasound designs have been completed, and the fourth generation of the design is currently under development. All three of the prototypes use a commercial of-the-shelf (COTS) ultrasound imaging system from a medical ultrasound company called *Teratech*.

Teratech's imaging systems, called the *Terason 2000* and *Terason 3000,* consists of imaging software, a range of ultrasound transducers, and a front end, containing beam forming and gain control circuitry [2]. Terason's imaging software is compatible with Windows XP computing platforms, and is generally intended to be used on a laptop. The combination of an ultrasound transducer and front-end electronics is called a *SmartProbe.* Figure 7 shows a Terason 2000 SmartProbe with a phased array transducer. The free end of the SmartProbe is connected to a power module that interfaces with the computing platform. Different types of ultrasound transducers can be used with the front-end electronics to support a variety of examination types. Both the first and second generation designs used the Terason 2000, while the third generation uses the updated Terason 3000 system.

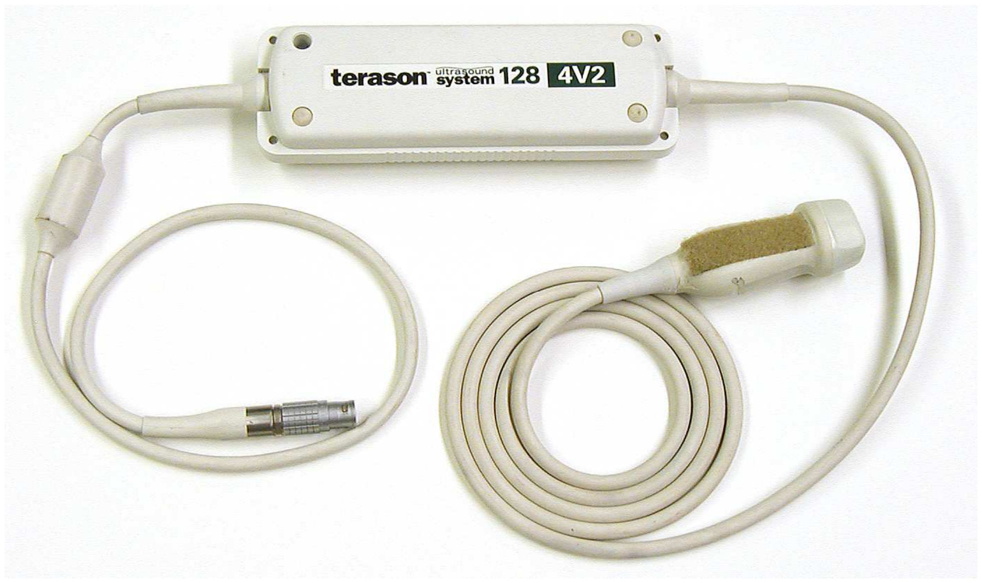


Figure 7: Terason 2000 SmartProbe [4]

The first generation prototype used a standard laptop to run the Terason 2000 software. The laptop was housed in a backpack along with all other peripherals. Commands could be issued to the system through the use of voice recognition software, and the ultrasound images were viewed through a head-mounted display (HMD) [11]. The second generation design replaced the laptop with an embedded computer in a 3.5'' form factor, and was housed in a custom designed metal enclosure. The embedded computer along with peripheral devices were powered using a COTS power supply, and two rechargeable Li-Ion batteries were used as the power source [12]. The first and second generation ultrasound systems can be seen in Figure 8 and Figure 9, respectively.

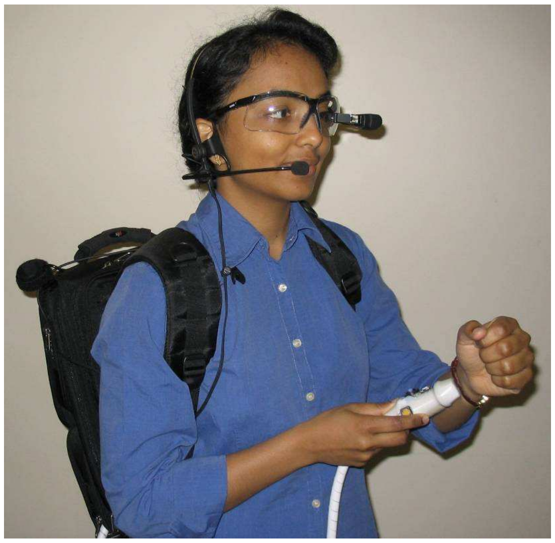


Figure 8: 1st Generation Ultrasound System [11]

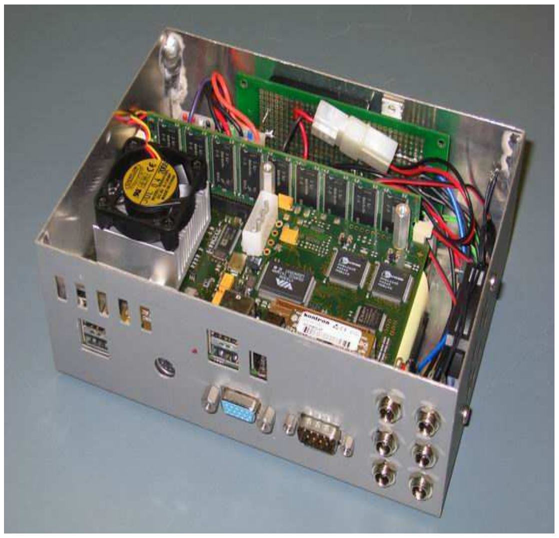


Figure 9: 2nd Generation Ultrasound System (Embedded Computer Only) [12]

This section will now go on to give a brief design overview of the third generation prototype. Each of the wireless options tested for this thesis are intended for use with the third or fourth generation systems, and all of the wireless hardware necessary for remote data transfer can be integrated into the third generation design as is.

1.4.1 3rd Generation Design Overview

Like the second generation prototype, the third generation design uses an embedded computer to run Terason's ultrasound software. The embedded computer is housed in a new custom enclosure that is more sleek and ruggedized than that of the metal enclosure seen in Figure 9. This new enclosure is able to operate in the presence of moisture and dust while also providing effective cooling for operation in high ambient temperatures. The exterior of the enclosure is made out of a material called Delrin, which is a strong plastic generally used as a substitute for nonferrous metals, such as aluminum, tin, zinc or brass. Inside the enclosure, there are two compartments separated by a heatsink. The bottom compartment is hermetically sealed to keep out moisture and dust particles, and contains the essential system components. The upper compartment contains two fans and is necessary to cool the sealed compartment [4]. The exterior of the third generation enclosure can be seen in Figure 10.

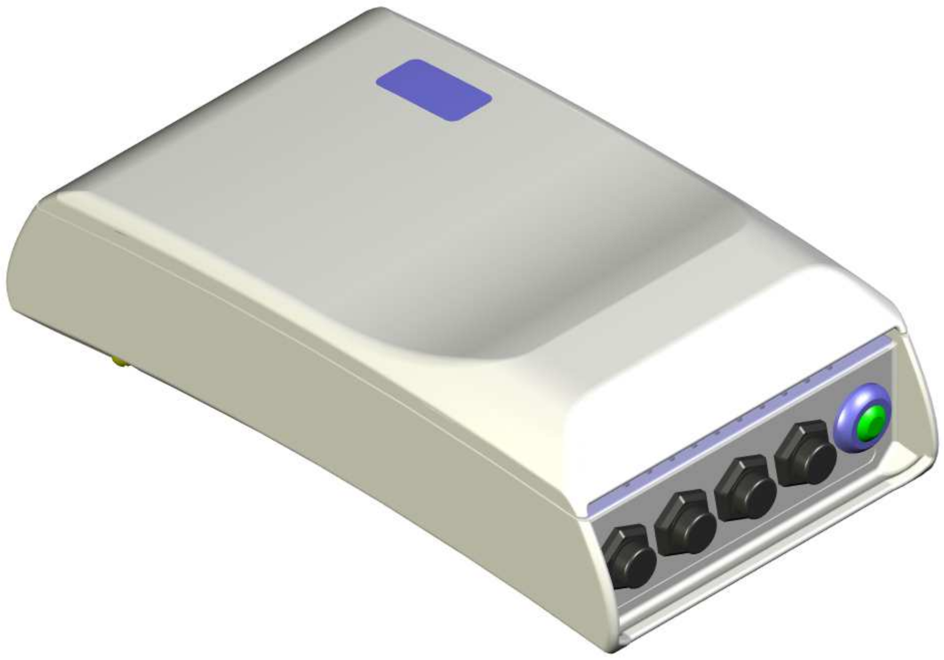


Figure 10: Ultrasound System Enclosure [4]

Located on the outside of the enclosure are custom locking connectors necessary for interfacing with peripheral devices. Each female socket on the enclosure has a unique male connector counterpart ensuring devices cannot be inserted into the wrong socket. When a device is attached to a socket, it must be locked using a twisting motion so that components won't become disconnected if an unexpected strain is exerted on the connector. The major system components of the third generation design are:

- Embedded Computing Platform
- Ultrasound Transducer
- Two Li-Ion rechargeable batteries
- Head-mounted display
- Microphone
- Mouse

Figure 11 shows a block diagram of the third generation design. The embedded computing platform is located inside the custom enclosure, while all other components are attached via the locking connectors.

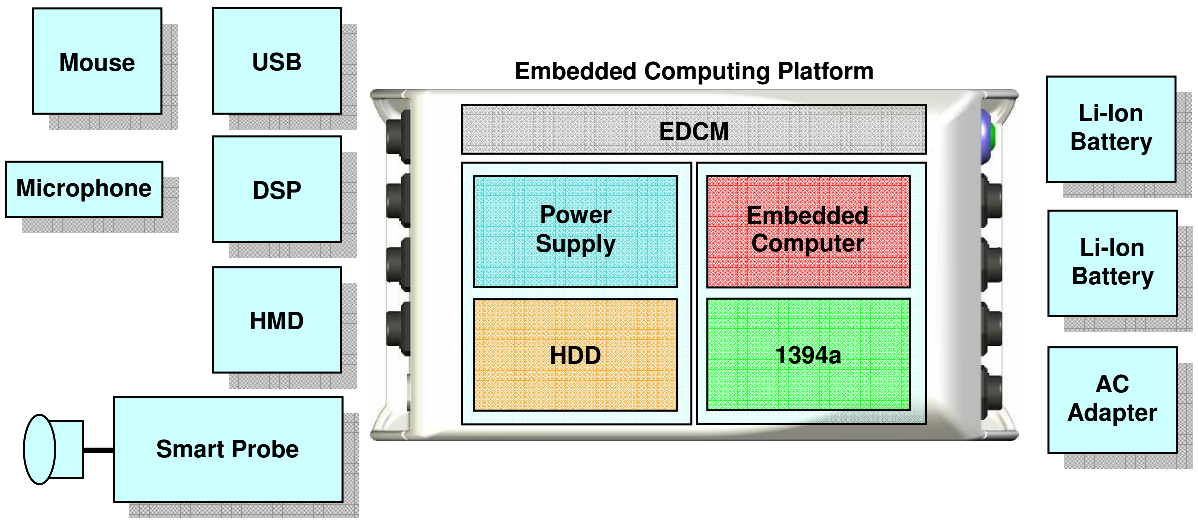
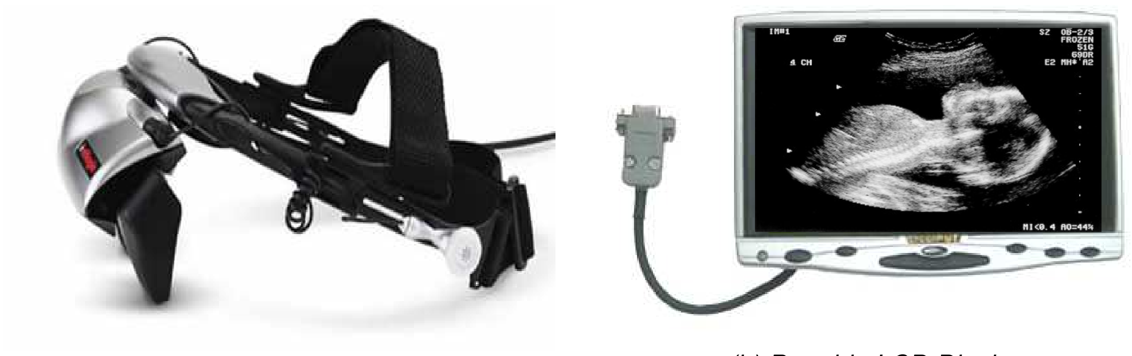


Figure 11: 3rd Generation Reconfigurable Ultrasound System Block Diagram [4]

The embedded computing platform consists of the embedded computer, a custom power supply, a hard disk drive (HDD), an IEEE 1394a (*FireWire*) interface, and an External DC Module (EDCM). Each of these components are located inside the hermetically sealed compartment of the enclosure. To run Terason's software, the Windows XP operating system was installed on the embedded computer, and a 20 GB HDD was added for data storage. Terason hardware (front end and transducer) requires a 6-pin FireWire port to be used by a computer. Since the embedded computer did not have an on-board FireWire interface, an additional board was added to support this feature. The EDCM is a power supply that is necessary to provide the Terason 3000 with a number of required voltages. One end of the EDCM connects to the IEEE 1394a interface while the other end is attached to a Terason 3000 SmartProbe. The last component housed within the hermetically sealed chamber is a custom built power supply designed by Philip Cordeiro as part of his thesis project [4]. All of the system components are powered through this supply, and two Li-Ion batteries are used as a power source. On two fully charged batteries, the ultrasound system can run in full operation mode for upwards of eight hours.

The remaining components of the ultrasound system are located on the outside of the enclosure, and are attached to the system via the aforementioned custom locking connectors. The ultrasound image is viewed either through a head-mounted display (HMD) or a portable LCD monitor. A standard visual graphics array (VGA) interface carries the video signal from the embedded computer to the portable display option, and the resolution of the output is 800x600 pixels. An example of an HMD and portable monitor can be seen Figure 12.



(a) eMagin HMD [13] *(b) Portable LCD Display*

Figure 12: Display Options for Mobile Ultrasound System

There are two options for issuing commands to the ultrasound system. The first is through the use of a trackball mouse which can be seen in Figure 13. The idea behind the trackball mouse is that it can be used with one hand and does not require a flat surface to operate. This is useful because the system operator will require at least one hand to utilize the ultrasound transducer, and rarely will there be an adequate surface for the operation of a mouse in actual field usage.

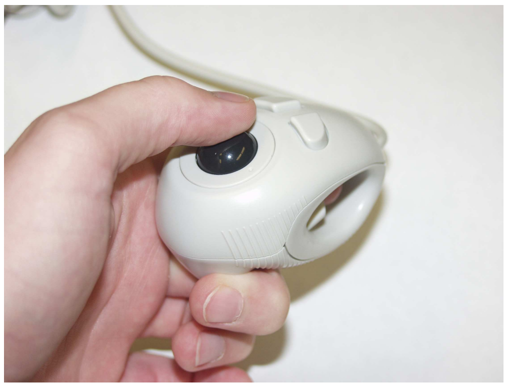


Figure 13: Trackball Mouse [4]

The second option for executing commands on the system is through the use of speech recognition software. A microphone can be connected to the system through one of the locking connectors to carry an audio signal from the operator. The third generation design uses what is called a speaker-independent Automated Speech Recognition (ASR) engine which means training is not required before each individual user, but only a restricted number of phrases will be recognized. All of the recognizable phrases are contained in a grammar file on the system. Only those phrases contained in the grammar file can be recognized by the ASR engine, and when a phrase is successfully recognized, a pre-programmed command execution will be performed. Full control of the system can be had through speech recognition; from changing exam modes, to entering and saving patient information to system shutdown. The third generation system uses the ASR engine *VoCon 3200* from a company called *ScanSoft (*now *Nuance)*.

1.4.2 Reconfigurable Design

As previously described, the third generation design is referred to as a reconfigurable ultrasound system because its configuration can be adapted depending on the specific application it is being used for. The two most common arrangements are a vest and a bag configuration. The handheld bag configuration, which can be seen in Figure 14, houses the system enclosure, Li-Ion batteries, and SmartProbe in the main compartment of the bag. The remaining components such as the mouse, microphone, and HMD can be held in the side pockets of the bag for easy access.

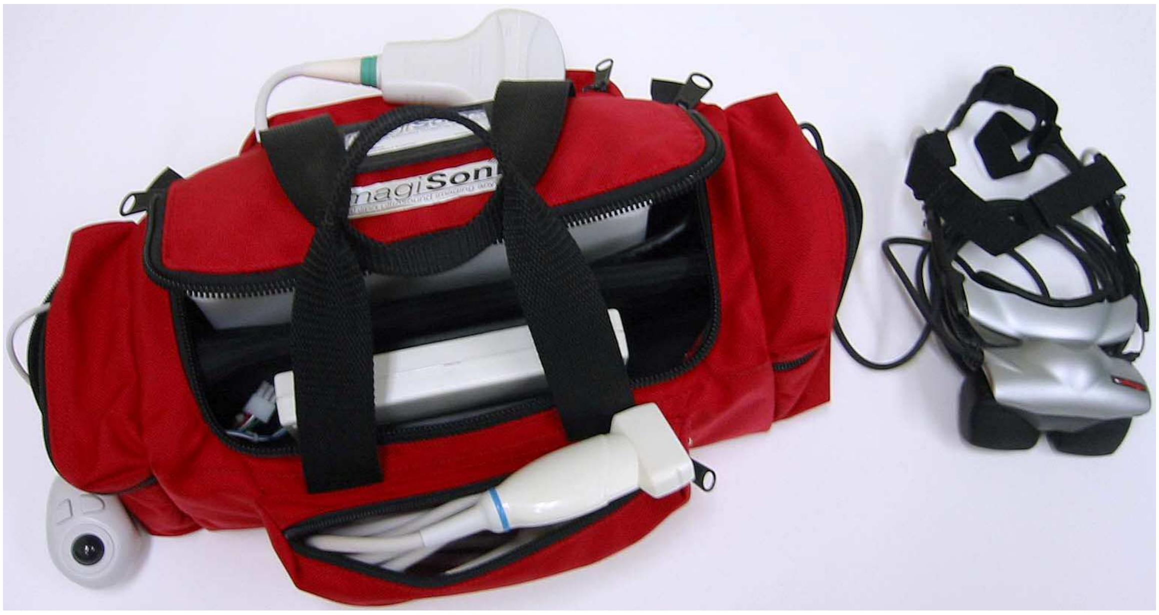


Figure 14: Handheld Bag Configuration with Head-Mounted Display

An alternative to the bag configuration is a vest configuration. In this arrangement, all of the system modules are contained in separate compartments of a wearable vest. Figure 15 shows how each of the components can be arranged in the vest, and Figure 16 displays the reconfigurable system in use. The vest shown in these images is a photographer's vest designed by a company called Domke. It has been modified by a tailor to include openings within the vest to run the cables that interconnect the system.

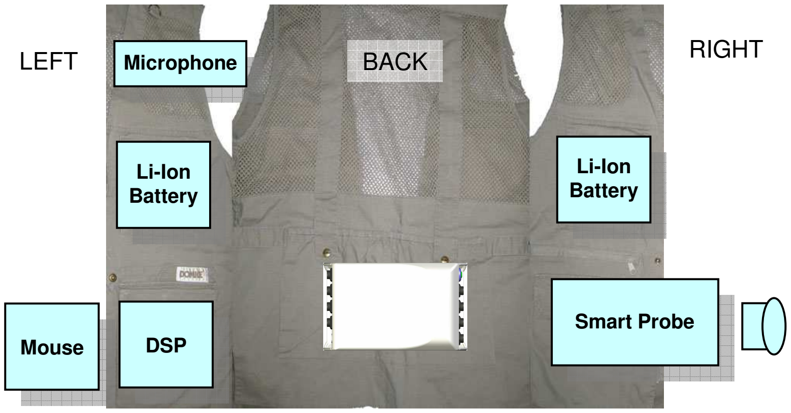


Figure 15: Vest Component Layout [4]

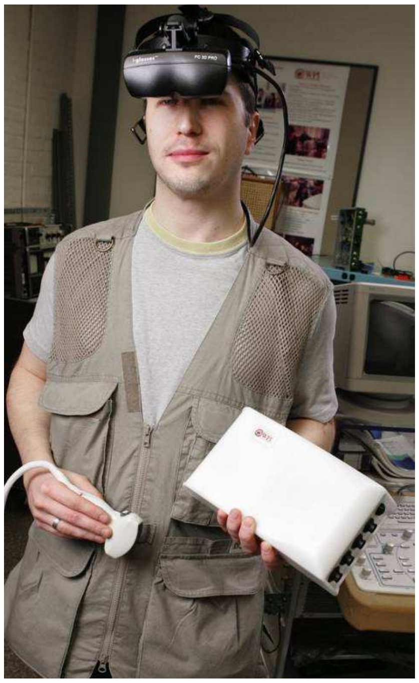
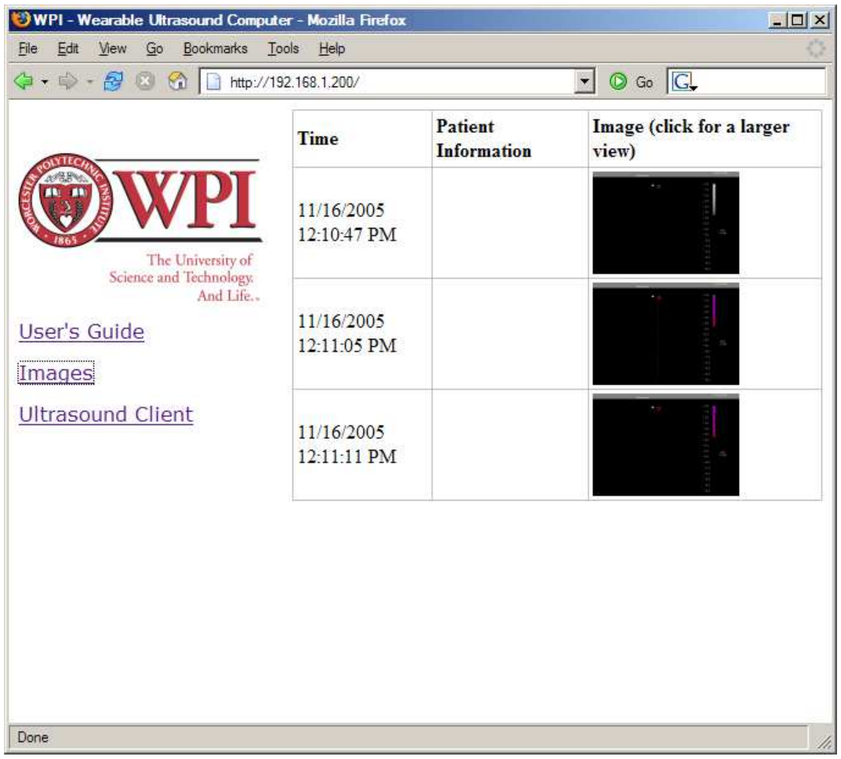


Figure 16: Wearable Vest Configuration [4]

<u>1.4.3 Data Capabilities</u>

WPI's mobile ultrasound system is capable locally storing patient information, still images, audio recordings, and video clips for download by a remote host as well as streaming live voice and video data. To retrieve locally stored files from the portable system, a web server supplied by the Microsoft Windows operating system has been deployed on the unit. As it is currently configured, remote users can access the web server through the use of an ad-hoc (peer-to-peer) network to obtain the files saved on the system. Figure 17 shows the interface through which a remote user can download saved images from the portable ultrasound system. The web server can be accessed through any standard web browser.


WPI - Wearable Ultrasound Computer - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://192.168.1.200/
Go
WPI
The University of Science and Technology. And Life.
User's Guide
Images
Ultrasound Client
Done

Time	Patient Information	Image (click for a larger view)
11/16/2005 12:10:47 PM		
11/16/2005 12:11:05 PM		
11/16/2005 12:11:11 PM		

Figure 17: Saved Images on Web Server [4]

When downloading static files from the ultrasound system, the Transmission Control Protocol (TCP) is the transport protocol that is used. In contrast, the User-Datagram Protocol (UDP) is used when streaming live voice or video information. Chapter 2 will discuss each of these protocols in greater detail and explain why they are used in different applications. The following table shows the format and typical file sizes for static files saved on the ultrasound system. Since there are many different factors that determine data rates for streaming voice and video applications, it is difficult to determine the exact bandwidth requirements for each application. Chapter 6 will go into much greater detail regarding real-time data transfer.

Table 4: Static File Sizes and Formats on 3rd Generation System

File Type	File Format	Length	File Size
Patient Information	SQL	-	1 KB
Image	PNG	-	15 KB
Audio/Voice	WAV (GSM 6.10)	30 sec	~50 KB
Video	AVI	10 sec	~50 MB

1.5 Thesis Contributions

This section lists the contributions that this thesis makes to the overall body of knowledge. Before the contributions are presented, it should be emphasized that three different wireless telecommunication technologies were evaluated in this project; however, this thesis is not meant to make a direct comparison of the three different technologies. Instead, it is meant to determine the opportunities that are possible with each wireless option and provide guidance as to which option may be most efficient for a given telemedicine application. The following list describes the major contributions of the research work completed in this thesis:

- Performance metrics for an isolated 802.11g ad-hoc network
    - Throughput as a function of SNR (UDP and TCP)
    - Latency as a function of SNR (forward and reverse)
    - UDP Throughput as a function of packet size
    - Latency as a function of packet size (forward and reverse)
    - Packet loss percentage at channel capacity
    - Jitter behavior at 25%, 50%, 75% and full channel capacity
- Performance metrics for AT&T's 3G wireless data network that employs HSDPA on the downlink and UMTS on the uplink
    - Throughput as a function of SNR (UDP and TCP)
    - Latency as a function of SNR (forward and reverse)
    - UDP Throughput as a function of packet size
    - Latency as a function of packet size (forward and reverse)
    - Packet loss percentage at channel capacity
    - Jitter behavior at 25%, 50%, 75% and full channel capacity
- Performance metrics for Inmarsat's BGAN satellite network
    - Throughput as a function of SNR (UDP and TCP)
    - Latency as a function of SNR (forward and reverse)
    - UDP Throughput as a function of packet size

    - Latency as a function of packet size (forward and reverse)
    - Packet loss percentage at channel capacity
    - Jitter behavior at 25%, 50%, 75% and full channel capacity
- Information regarding real-time image streaming using H.264 SVC video compression over an isolated 802.11g ad-hoc network
    - Jitter behavior
    - Histogram of packet sizes used in data stream
    - Packet loss percentage
    - Maximum capabilities using echocardiograph ultrasound scans – VGA resolution / 30 fps
- Information regarding real-time image streaming using H.264 SVC video compression over AT&T's 3G wireless data network
    - Jitter behavior
    - Histogram of packet sizes used in data stream
    - Packet loss percentage
    - Maximum capabilities using echocardiograph ultrasound scans – QVGA resolution / 10 fps
- Information regarding real-time image streaming using H.264 SVC video compression over a network emulator meant to simulate Inmarsat's BGAN satellite network
    - Jitter behavior
    - Histogram of packet sizes used in data stream
    - Packet loss percentage
    - Maximum capabilities using echocardiograph ultrasound scans – QVGA resolution / 15 fps
- Feedback from physicians regarding diagnostic value of ultrasound video streams over the three different wireless telecommunication technologies

As would be expected, some of this information is currently available in open literature. For example, Xiao and Rosdahl examined the throughput and delay limits of 802.11 in [61]. Additionally, throughput and delay performance of UMTS, which is a 3G network, was examined in [62]. However, most of the available information is based on theoretical limits based on the implementations of the interface. Empirical performance data was more difficult to find, especially for 3G and satellite networks. Additionally, the literature based on measured data is unique to the specific hardware and chipsets used during testing. For this reason, this thesis provides empirical data for hardware that can currently be implemented into WPI's reconfigurable ultrasound system.

1.6 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 details the methodology used to measure various channel characteristics of the wireless links. It goes on to discuss all of the metrics that were obtained during testing, and why each one is important in different remote data applications. Lastly, the development of a custom client/server application used to measure each of the performance metrics is presented. Issues such as one-way delay, jitter and the timing accuracy of the measurements are also presented in this chapter.

Chapter 3 begins with an overview of the IEEE 802.11 protocol including a detailed discussion of the PHY layer and MAC layer implementations described by the standards. Next, a testing protocol used to obtain the performance metrics for 802.11g is given followed by the testing results and analysis.

Chapter 4 discusses the evolution of the *Global System for Mobile Communication* (GSM) family of wireless data standards from common 2G to 3G standards. It goes on to discuss a testing protocol which was carried out on AT&T's *BroadbandConnect Network* which is a 3G wireless data network. Finally, a presentation of the performance test results is given.

Chapter 5 describes a set of performance tests carried out over a satellite network. The chapter begins with an overview of a satellite network from a company called *Inmarsat*. It goes on to discuss the details of the testing protocol and concludes with the results and analysis made during testing.

Chapter 6 describes a second phase of the testing in which real-time ultrasound video streams were transmitted over the various wireless links. The chapter starts with an overview of the technology used by WPI's mobile ultrasound system to stream live ultrasound image. It goes on to discuss the testing protocol and ends with a presentation of the testing results. Finally, Chapter 7, the last section of the thesis, presents conclusions and recommendations for future work.

2 Measuring Channel Characteristics

An initial phase of testing was performed to measure the basic channel properties of the wireless communication links studied in this thesis. When measuring channel characteristics, it is important to determine exactly how data is transmitted over each link. This chapter will describe the different protocols used by the reconfigurable ultrasound system to send and receive data. An explanation of why specific protocols are used in certain applications will be given. The chapter will continue with a discussion regarding the specific metrics that were measured during each test as well as the methodology used to obtain the measurements. Lastly, it will describe some of the pitfalls of the measurement system and how these problems were resolved.

2.1 Transport Protocols

The *Open System Interconnection* (OSI) model, shown in Figure 18, defines a networking framework for implementing protocols in seven layers [17]. When sending data across a network, control is passed from one layer to the next, starting at the application layer and proceeding all the way down to the physical layer. The opposite is true when receiving data, where control is passed up the OSI hierarchy starting at the physical layer.

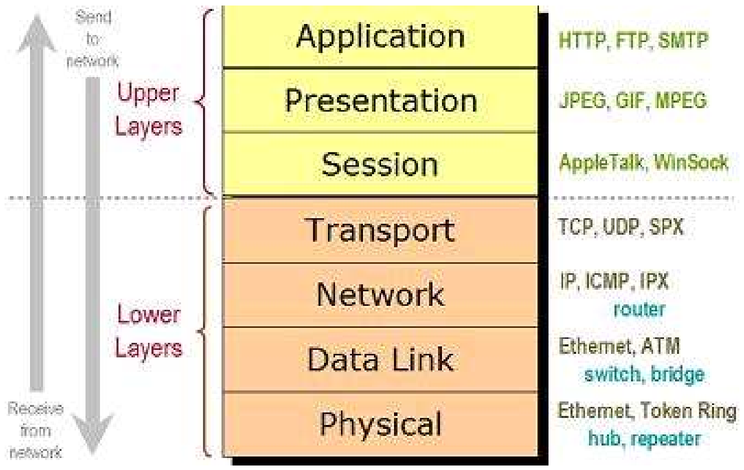


Figure 18: OSI Network Model [16]

In the OSI network model, each of the communication options examined in this thesis reside in the bottom two layers of the model. Each wireless option provides support for both the physical and data link layers of the OSI model making it possible to connect to an IP (Internet Protocol) network. The physical layer is responsible for transmitting a raw bit stream across a medium at the electrical and mechanical level, while the data link layer codes and decodes raw bits into frames for network transmission [17].

The current generation of the reconfigurable ultrasound system requires an IP network (Layer 3) for remote data transfer. Once on an IP network, the system can use different higher layer protocols, depending on the type of data that is being sent. The *Transmission Control Protocol* (TCP) is used when downloading pre-stored files from the mobile unit, while the *User Datagram Protocol* (UDP) is used when streaming live video or voice information. This section will describe each of the two transport protocols and the reason for their use.

2.1.1 TCP

TCP was specifically developed for the reliable end-to-end delivery of a byte stream over an unreliable network [17]. Data at the network layer, which is an IP network in this case, is broken up into units called *packets*. The IP is responsible for the actual routing and delivery of packets between two network endpoints; however, it makes no guarantees regarding the successful delivery or ordering of packets. TCP provides a level of reliability to IP networks by guaranteeing the successful transmission of packets in the correct order. In addition to reliability, TCP provides congestion and flow control to provide efficient data transfer on diverse network topologies that may have different bandwidths, delays, and packet sizes. This protocol was designed to dynamically adapt to different network conditions and to be resistant to many kinds of failures [17]. Figure 19 shows both the network and transport layers of the OSI model. It also shows how

IP data is inherently unreliable at the network layer because there are no built in error control mechanisms. This figure shows that data at the transport layer is reliable due to TCP error control utilities.

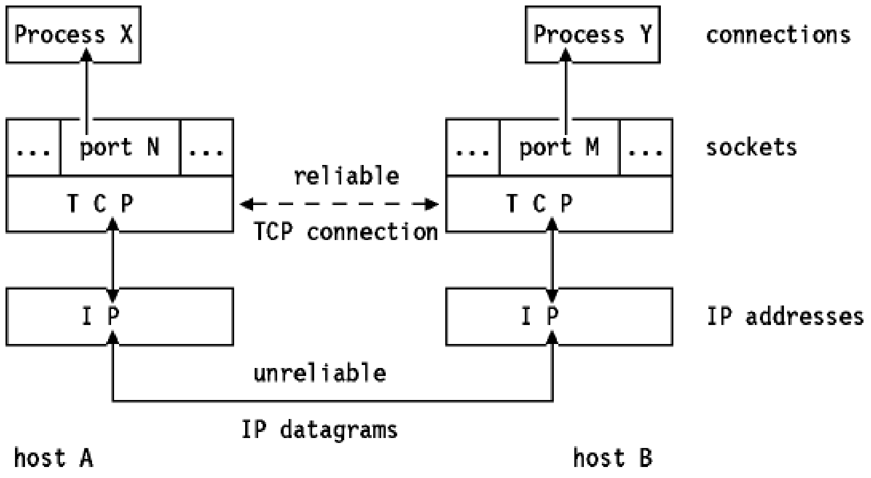


Figure 19: Reliable TCP Connection [18]

TCP entities exchange data in the form of TCP segments, although it is not uncommon to refer to TCP segments as packets. Each segment begins with a fixed 20 byte TCP header followed by a variable amount of data. Each segment must fit into the Maximum Transmission Unit (MTU) of the network it is on, which is generally 1500 bytes or the maximum size of an Ethernet payload. The IP also adds a 20 byte header to the packet which results in a maximum TCP segment size (MSS) of 1480 bytes, and a maximum TCP payload of 1460 bytes [17]. The TCP header can be seen in Figure 20.

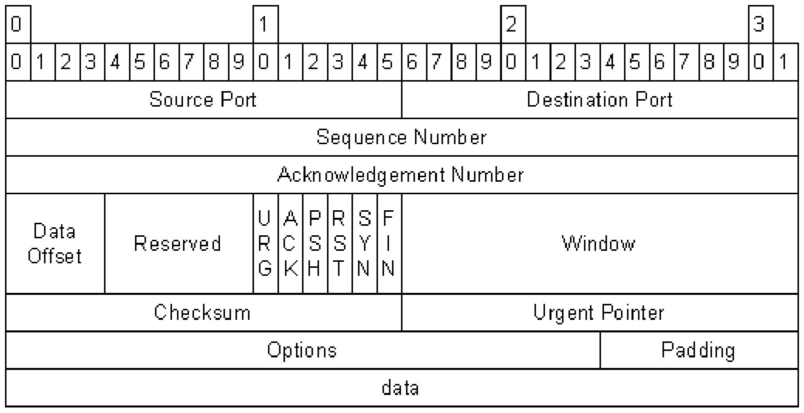


Figure 20: TCP Header (20 Bytes) [19]

TCP is a connection oriented protocol, meaning both client and server must agree to establish a connection before any data can be exchanged. Before a connection can be established, both the client and server applications must create a network endpoint called a socket. Each socket consists of an IP address along with a 16-bit number unique to that host called a port. All TCP connections are full duplex and point-to-point, meaning the connection has exactly two endpoints, and data can travel in both directions simultaneously. TCP connections are established by means of a three-way handshake. Initially, the client sends a SYN (SYN flag set to 1 in header) packet to the server. The server then replies back with a SYN+ACK segment, and finally the client sends an ACK back to the server. The TCP connection is now established, and data can be exchanged [17].

TCP uses what is called a “sliding window” protocol for data transmission. The sliding window protocol ensures the reliable delivery of data in the correct order as well as provides a mechanism for flow control between the sender and receiver. Each TCP segment contains a sequence number, which is the number of the first octet (or byte) in the segment. When the connection is set up, the client and server agree to an initial window size, which dictates how much data can be transmitted by the sender before receiving an acknowledgement (ACK) packet. The size of the window can vary throughout the duration of the connection, and is normally based on the amount of space that is available in the receiver’s incoming buffer. The receiver periodically sends ACK packets back to the sender, which acknowledges the successful reception of octets up to a given sequence number, and adjusts the window size limiting the amount of data that the sender can transmit before receiving another ACK [20]. Figure 21 gives an example of the sliding window scheme. In this example, *SN* refers to the sequence number which corresponds to the octet number of the first byte in the segment. *AN* refers to an acknowledgement number which means the receiver acknowledges the successful delivery of octets up to the sequence number *AN* –

1. Lastly, *W* is the size of the window which tells the sender that *W* more bytes can be sent before receiving another ACK. It should be noted that the segment size in the following example is 200 bytes.

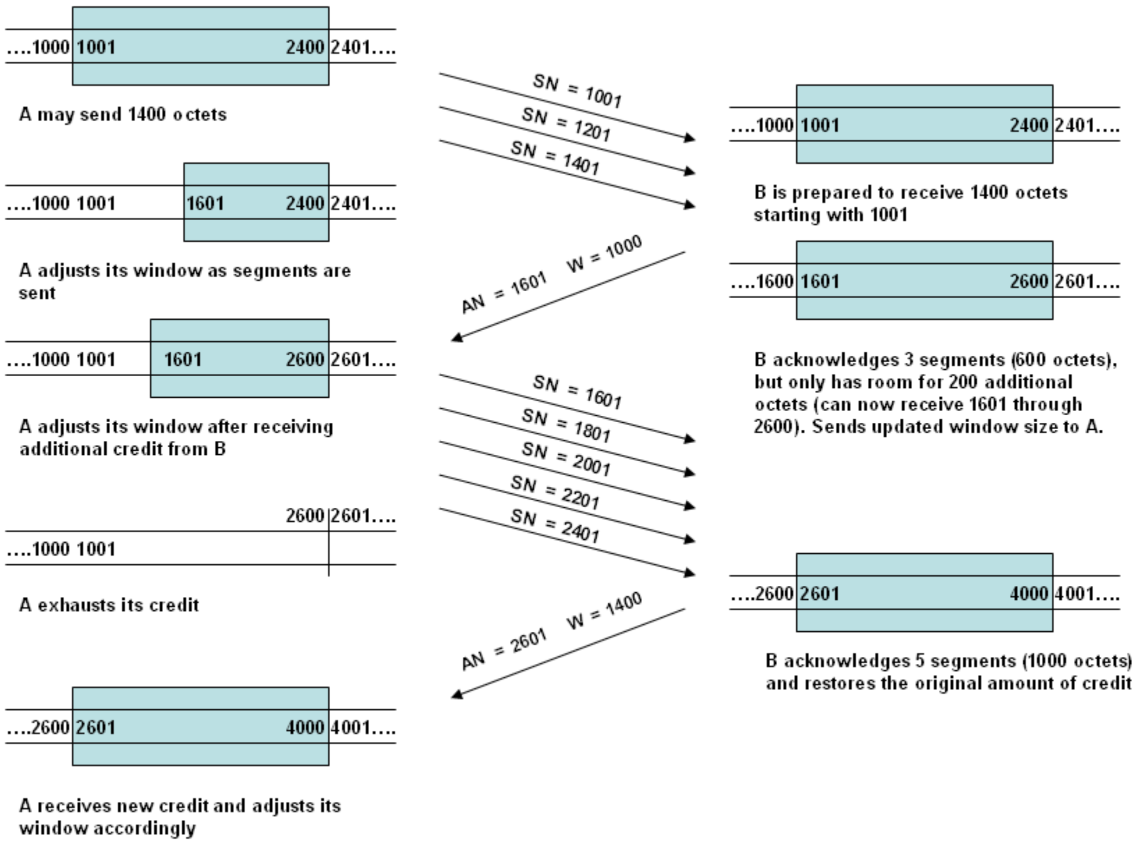

....1000 1001 2400 2401....
A may send 1400 octets
SN = 1001
SN = 1201
SN = 1401
....1000 1001 2400 2401....
B is prepared to receive 1400 octets starting with 1001
....1000 1001 1601 2400 2401....
A adjusts its window as segments are sent
AN = 1601 W = 1000
....1600 1601 2600 2601....
B acknowledges 3 segments (600 octets), but only has room for 200 additional octets (can now receive 1601 through 2600). Sends updated window size to A.
....1000 1001 1601 2600 2601....
A adjusts its window after receiving additional credit from B
SN = 1601
SN = 1801
SN = 2001
SN = 2201
SN = 2401
2600 2601....
....1000 1001
A exhausts its credit
....2600 2601 4000 4001....
AN = 2601 W = 1400
B acknowledges 5 segments (1000 octets) and restores the original amount of credit
....2600 2601 4000 4001....
A receives new credit and adjusts its window accordingly

Figure 21: Example of TCP Sliding Window Scheme [20]

To ensure the successful delivery of all segments, unacknowledged segments must be retransmitted. To do this, a timer is associated with each segment as it is sent, and if the timer expires before the segment is acknowledged, it must be retransmitted. The setting of this timer will greatly affect the performance of TCP. If the timer setting is too small, there will be many unnecessary segment retransmissions resulting in wasted bandwidth. If the timer setting is too high, there will be a long delay when handling lost segments. Depending on the specific TCP implementation, different methods can be used to determine the time value of the retransmission timer (RTO). The RTO is based on the *round trip time* (RTT) of the link, and the two most common methods for estimating the expected RTT for a given segment are *Average Round-Trip Time* (ARTT) and *Smoothed Round-Trip Time* (SRTT). ARTT keeps a running average of the RTT of each segment while SRTT exponentially averages old RTTs, putting more weight on the most recent samples. The SRTT and ARTT algorithms can be seen below. The variable, α in the SRTT algorithm is a smoothing coefficient. Typical TCP implementations use a value of 7/8 for α [20].

$$ARTT(K+1) = \frac{1}{K+1} \times \sum_{i=1}^{K+1} RTT(i) \quad (1.1)$$

$$SRTT(K+1) = \alpha \times SRTT(K) + (1-\alpha) \times RTT(K+1) \quad (1.2)$$

The last major feature of TCP is congestion control. Congestion control ensures that the sender will not flood the network with more data than it can carry even though the receiver may have enough buffer space to handle it. Most TCP implementations use what is called “slow start with congestion avoidance” for congestion control. After the connection is set up, the initially window size will allow for the sender to send one segment. The window size will continue to grow exponentially assuming all of the transmitted segments as successfully acknowledged. This algorithm is called “slow start.” Once the window size grows too large and a timeout occurs, a threshold on the window size will be set at the last successful transmission. The slow start algorithm will start again, but this time, the window size will increase by one rather than exponentially after it reaches the threshold [20]. Refer to Figure 22 for an example of this algorithm. In the figure, the Y-axis has a value of *cwnd* (congestion window) in units of TCP segments. The X-axis is in units of round trip times. For example, the congestion window is initially one segment. After that segment is acknowledged, it grows to two then to four and so on for each subsequent round trip time.

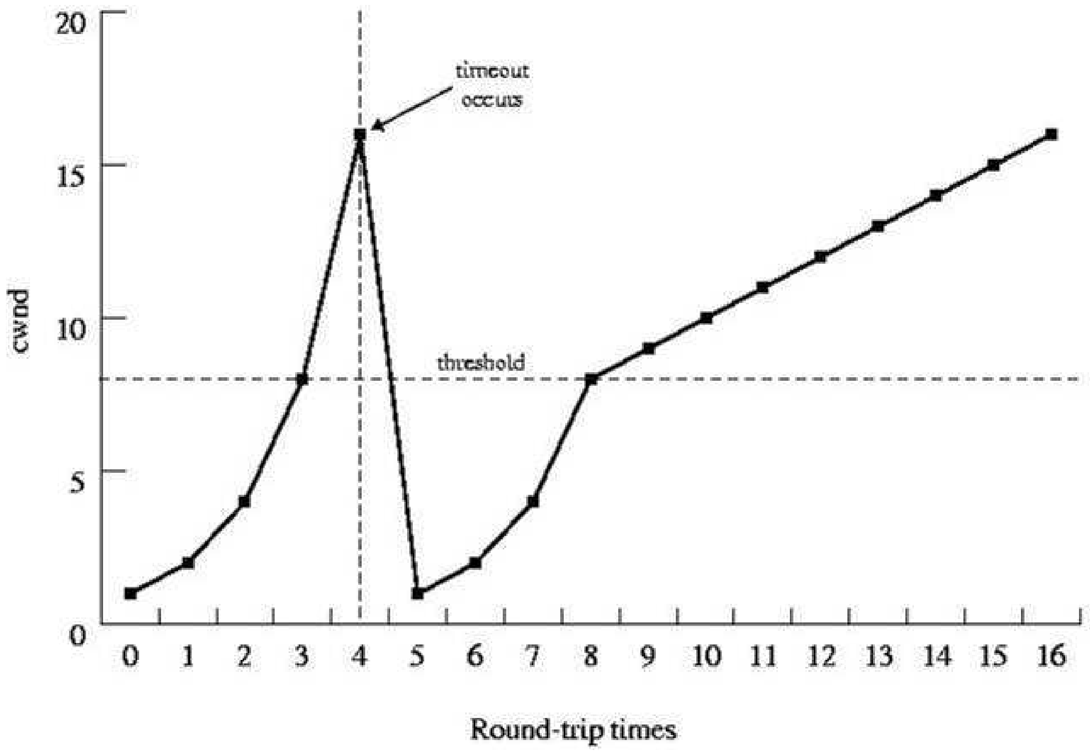


Figure 22: An Example of Slow Start with Congestion Avoidance (cwnd - Congestion Window) [20]

As previously mentioned, the reconfigurable ultrasound system uses TCP when transmitting pre-stored files containing information such as patient information, still images, audio recordings and video clips. TCP is used for file transfers because it is necessary to send the data free of errors or else the files may become corrupt and unusable. Also, it is evident that the flow control and congestion control features of TCP will never allow the protocol to fully utilize all of the channel capacity of the link. By throttling itself, some amount of bandwidth is bound to be wasted.

One problem with using the TCP protocol for data transmission from the portable ultrasound unit is that TCP was optimized for use on wired networks, and can be very inefficient when used on wireless networks. The main issue is the congestion control algorithm. When run on wired networks, TCP assumes that segments timeout due to network congestion rather than packet loss. When this happens, TCP slows down to try to alleviate congestion as shown above.

Wireless links are much less reliable than wired networks, and packets are lost much more frequently. The congestion control approach of TCP is not efficient for a high percentage of packet loss [17]. There are experimental implementations of TCP that have been shown to exhibit better performance on wireless links when compared to most standard TCP implementations. To adjust TCP settings, and implement a new version of TCP, registry values within the operating system must be altered. These experimental TCP versions have not yet been implemented on WPI's reconfigurable ultrasound system.

2.1.2 UDP

The User Datagram Protocol (UDP) is another transport level protocol that rides on top of the IP network layer in the OSI network model. UDP is a connectionless protocol meaning it does not require the set up of a virtual circuit and each transmitted datagram is routed independently. In contrast to TCP, UDP does not offer flow control, congestion control, error detection or retransmission of dropped packets [17]. Instead, it relies on higher level protocols to deal with these issues. UDP simply sends and receives datagrams with no concern if they reach the intended destination.

UDP is basically a wrapper for IP. It adds a small, 8 byte header to IP packets to build a UDP segment or datagram. Like TCP, UDP uses the concept of a socket ,which consists of an IP address and port number. The main feature that UDP adds to the IP protocol is demultiplexing multiple processes using ports. The UDP header can be seen in Figure 23.

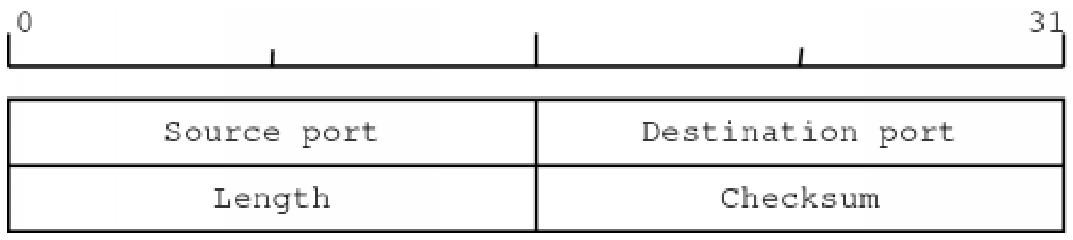


Figure 23: UDP Header (8 Bytes) [21]

UDP was designed for time sensitive applications that can tolerate the occasional dropped packet. This makes it perfect for applications such as streaming video, real-time gaming and voice over IP (VoIP). As with many streaming applications, WPI's reconfigurable ultrasound system uses UDP when sending live video or voice information. More information regarding the upper level protocols used on top of UDP when sending voice or video will be provided in Chapter 6: Live Image Stream Testing.

2.2 Testing Metrics

In the first phase of testing, basic information regarding the channel characteristics of each telecommunication option was measured. This information is essential to determine which types of data transfer applications are possible on each channel, and what kind of performance can be expected in each case. This section will discuss the specific metrics that were measured and recorded for each channel as well as explain how data transmission is affected by each property.

2.2.1 Throughput

The first metric that is of obvious importance is the throughput or capacity of the channel. Throughput is defined as the amount of digital data per unit time that can be delivered over a physical link, and it is generally measured in bits per second (bps) [22]. Due to contrasting definitions, this thesis will refer to throughput as the total amount of data transmitted over time including overhead. The term "goodput" will refer to the quantity of meaningful data (no overhead) sent using a given protocol.

Some of the main factors that affect throughput include packet loss, flow control algorithms, and network congestion. As described in the previous section, TCP implements flow and congestion control algorithms that limit the amount of data that is sent. For this reason, TCP throughput must be measured separately from the raw channel throughput with no flow control algorithms. Packet loss will obviously decrease throughput because all of the transmitted data is not successfully received. Packet loss can be caused by network congestion as well as bit errors due to the transmission medium. Another factor influencing the performance of wireless networks is received signal strength. Lower signal-to-noise ratio (SNR) values can lead to higher packet loss, and in some protocols, coding schemes are changed relative to the signal strength.

The two most important factors to look at when measuring throughput are the average throughput and the minimum throughput. Consider the example data shown in Figure 24. This channel exhibits a high average throughput (43.01 Mbps) but poor minimum throughput (1.66 Mbps). The average throughput is useful when estimating how long it will take to transmit a pre-stored file of a given size, but is not helpful in determining if the link can support a data stream that requires a consistent minimum data rate. The minimum throughput will determine if a certain data stream can be sent across the channel. If the minimum throughput of the channel routinely falls below the minimum required data rate of the stream, then data will inevitably be lost.

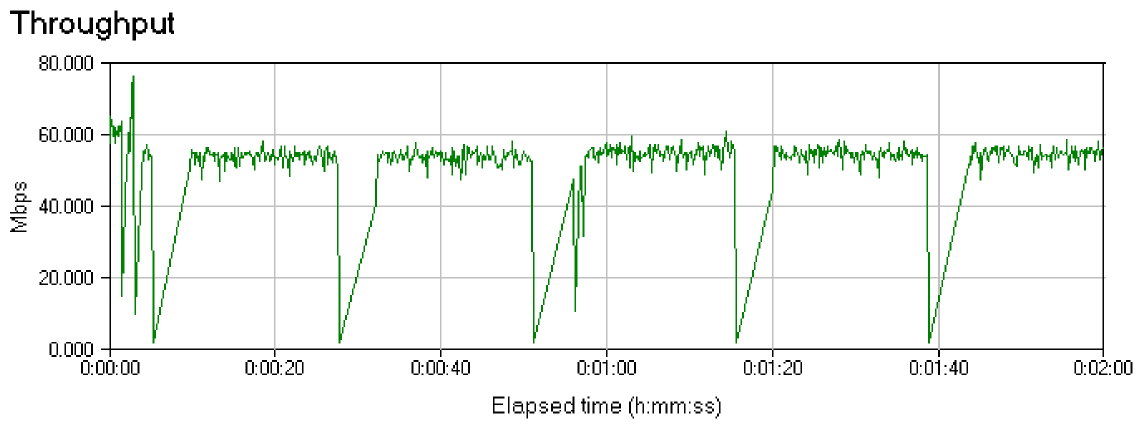


Figure 24: Example of High Average / Low Minimum Throughput [23]

2.2.2 Latency

The next metric that was measured for each channel was latency or delay. For this thesis, latency was measured as the total amount of time it takes from the start of transmitting a packet of data to when the entire packet has been received by the receiving station. Figure 25 shows a physical depiction of how the latency of a single packet is measured over a network. Two types of delay measurements were performed for each channel. Round-trip delay refers to the total delay to and from another network endpoint while one-way delays correspond to the latency of either the forward or reverse paths. Most delay measurements are on the order of milliseconds and different techniques were implemented to measure round-trip and one-way delays. The latency of a single packet is independent of the transport protocol being used to send packet.

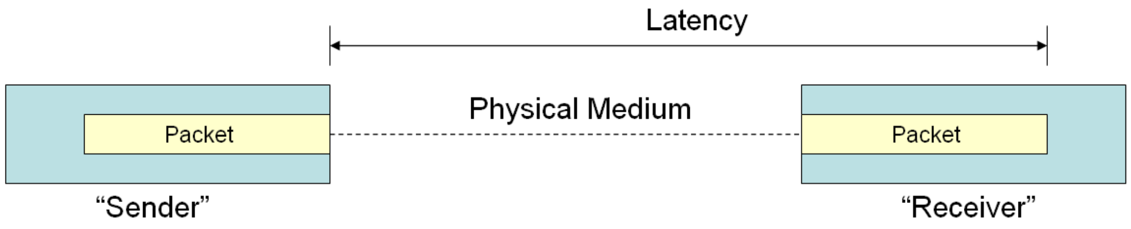


Figure 25: Measuring Packet Latency over a Network

Factors that contribute to network delay include packet size, channel capacity, distance, and collision avoidance algorithms of different network protocols. The packet size and channel capacity limit how fast data can be placed on the transmission medium. The distance between the two network endpoints will add some small propagation delay necessary for the analog signal to travel from the sender to the receiver. Lastly, some protocols have collision avoidance algorithms that induce a delay before a data is even placed on the transmission medium. On multi-hop networks such as the internet, additional delays due to queuing and processing delays at each router are introduced.

Awareness of network latency is important in applications such as two-way voice communications. Large delays in either direction will cause confusion between the speakers by not knowing when to speak and when to listen. In two-way voice applications, it is desirable to have delays below 150 ms in each direction [24]. Delay is not as much of a factor in one-way streaming applications because there is no interaction between the two network endpoints.

2.2.3 Jitter

As defined by RFC 4689, jitter is the difference between the forwarding delay of two consecutive packets belonging to the same data stream [48]. For example, if two consecutive packets (A and B) are sent through a network [49]:

- Packet A takes 18 ms to traverse the network
- Packet B takes 15 ms to traverse the network
- Jitter = 15 – 18 = -3 ms

To calculate jitter, four parameters are required: (i) the transmit time of A, (ii) the receive time of A, (iii) the transmit time of B, and (iv) the receive time of B. If these four parameters are known, then jitter can be calculated according to Figure 26. It should be noted that the clocks on both of the network endpoints need not be synchronized to measure jitter. Any offset between the clocks is eliminated in the jitter calculations.

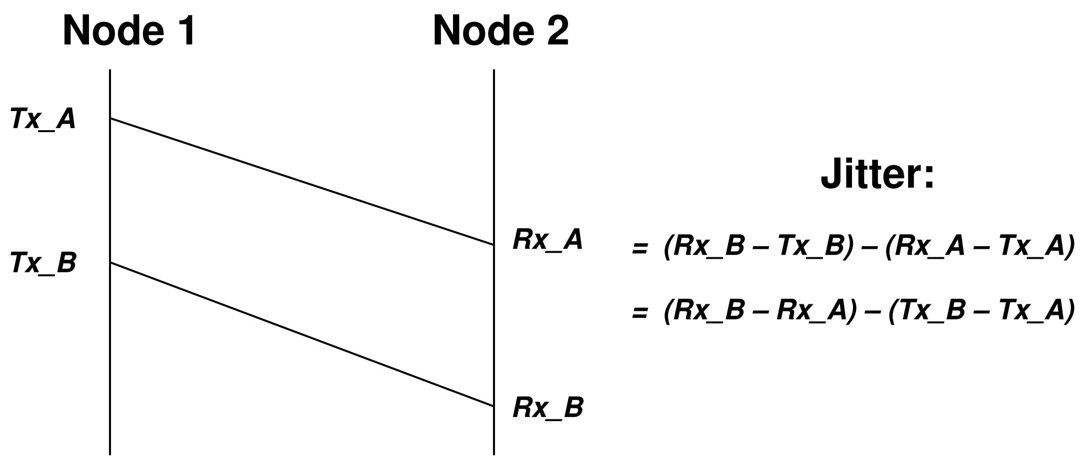

Node 1
Node 2
Tx_A
Tx_B
Rx_A
Rx_B
Jitter:
= (Rx_B – Tx_B) – (Rx_A – Tx_A)
= (Rx_B – Rx_A) – (Tx_B – Tx_A)

Figure 26: Calculating Jitter

If the measured jitter is a positive value, then that means the second packet took longer than the first to traverse the network. This is known as *spreading*. On the contrary, *clustering* is when the second packet traverses the network faster than the first which leads to a negative jitter value. An example of both scenarios can be seen in Figure 27.

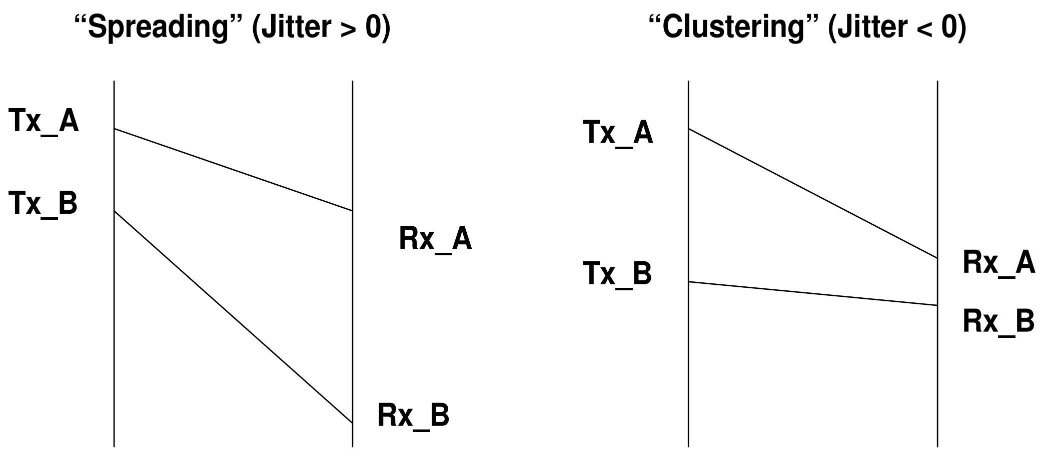


Figure 27: Examples of Jitter "Spreading" and "Clustering"

The main cause of delay jitter is intermediate network devices such as routers and switches. Buffering and queuing algorithms as well as network architectures all contribute to the overall jitter of a link. The variation in packet delay is compounded by each device through which the packets traverse. Jitter can also vary with traffic characteristics such as burst distribution and packet size [49].

Variation in packet delays has a significant effect on the quality of streaming voice and video applications. Most applications are designed to tolerate a certain amount of jitter by buffering incoming data in a jitter buffer; however the buffering adds to the overall latency of the link as seen by the user applications. Excessive jitter will lead to degradation in service quality by causing the jitter buffer to overflow or become empty. This tends to lead to dropouts or clicks in an audio stream or a choppy display in a video stream. The amount of tolerable jitter depends on the specific application, and it is typically less than 50 ms for most video and voice services [49].

2.2.4 Packet Loss

The last metric that was measured during testing was packet loss. Packet loss can be caused by a number of factors, including signal degradation over the network medium, oversaturated network links, faulty networking hardware, routing errors and packet collisions. Packet loss has different effects depending on the transport level protocol in use. For example, packet loss in TCP applications will greatly reduce throughput as packets will have to be retransmitted; however all of the data should still be successfully transferred due to the error recovery utilities of the protocol. On the other hand, packet loss in UDP streaming session will result in degradation in image or voice quality depending on the application.

2.3 Network Emulation

In order to simulate specific network conditions for testing, a network emulator was used. A network emulator is essentially a “network in a box”, where various network conditions such as bandwidth, latency, jitter and packet loss can be specified. Other computers or endpoints can then be connected to the emulator, and send packets to each other over the virtual network. A software program called *NistNET* from the National Institute of Technology and Standards (NIST) [57] was used as the emulator software, and it was installed on a system running Linux *Slackware v 2.6*. NistNET can emulate both symmetric and asymmetric links. For both the forward and reverse links, the bandwidth, latency, jitter, and packet loss percentage can be specified.

The network emulator helped ensure that the methods used to measure the channel characteristics of the various wireless options were reliable. The measurement software, which is described in the following section, was tested on the emulator, and predefined network conditions were measured to ensure the accuracy of the measurement tools. The emulator also helped to determine exactly what types of network conditions will begin to adversely affect image quality during video transmission from the ultrasound system. Chapter 7 describes a testing protocol used to observe streaming video run over the network emulator under various network conditions. Figure 28 shows how tests can be configured to run over the virtual network of the emulator using two network endpoints.

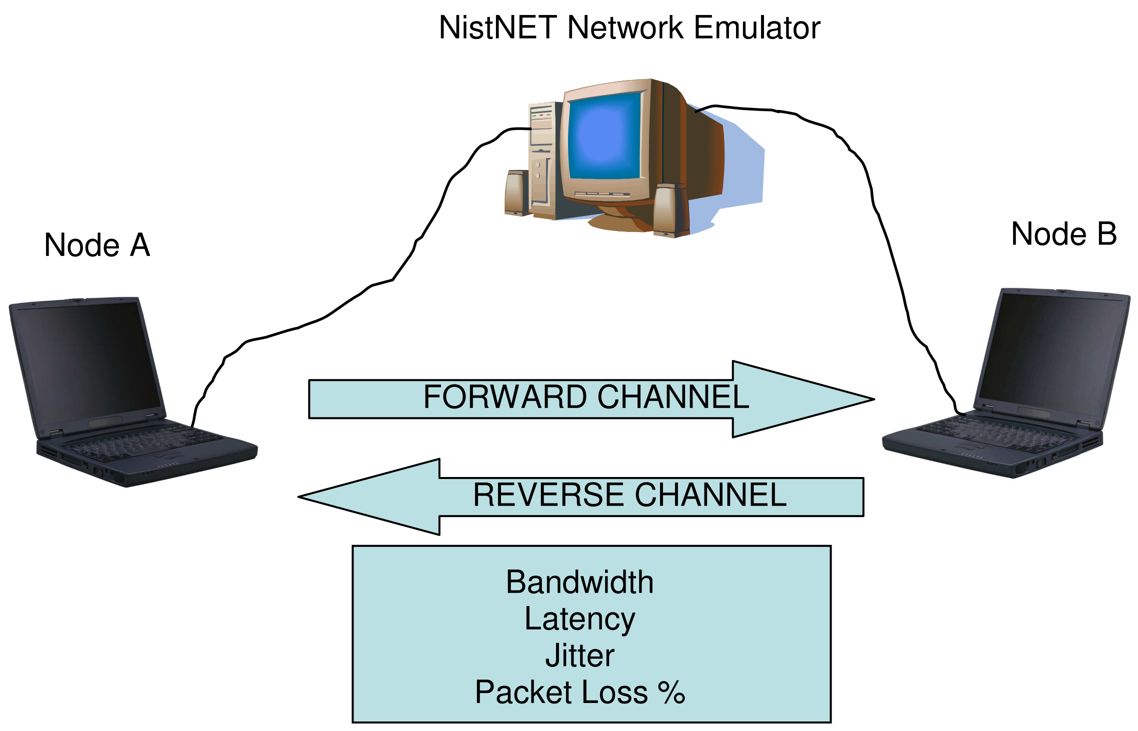


Figure 28: Network Emulator Configuration

2.4 Java Measurement Toolbox

To measure the characteristics of each of the wireless channels, a custom toolbox written in the Java programming language was developed. The toolbox uses UDP datagrams to measure each of the metrics mentioned in Section 2.2. In order to use the toolbox, control over two endpoints on an IP network is required. One endpoint must run a server application while another endpoint runs a client application. Both the server and client applications were written in the Java SE 6 programming language using the *NetBeans 5.5.1* development environment [25]. The remainder of this section will explain the overall configuration of the software as well as the individual functions of the toolbox.

The reason UDP was chosen as the transport protocol for this toolbox, as opposed to TCP, is because additional delay due to connection set up and acknowledgement packets would lead to inaccurate calculations regarding raw channel capacity and latency. Instead, fast packet transfer was chosen over reliability; however, the issue of dropped packets would need to be dealt with. Additionally, the third generation ultrasound system uses UDP when streaming live voice and video data, further leading to the choice of this protocol.

2.4.1 Library Package

The entire performance evaluation toolbox consists of three separate packages: the client, the server, and the library, which is used by both the client and server. To run either the client or server package from a network endpoint, the library package (called *ProjLib*) must be included with either the client or server package to function properly. The library contains all of the client and server side code for each of the measurement tools, while the client and server packages simply open instances of the classes defined in the library. In total, there are thirteen class definitions in the library, which will be explained in greater detail in the upcoming sections. The class definitions included in the library are:

- BWClient.java / BWServer.java (Bandwidth Tests)
- PSBWClient.jave / PSBWServer.java (Bandwidth vs. Packet Size Tests)
- DelayClient.java / DelayServer.java (Delay Tests)
- PSDelayClient.java / PSDelayServer.java (Delay vs. Packet Size Tests)
- DiscoveryServiceClient.java / DiscoverServiceServer.java (Discovery Service)
- Message.java (Messages)
- HRTimer.java (High Resolution Timer)
- Device.java (Internet Device)

The first ten classes define a separate thread that is started by the either client or server packages. These threads carry out the actual measurement tests and record the data. The Message class defines a message structure that is used throughout each of the tests while the Device class defines a structure describes internet compatible devices or network adapters capable of sending and receiving IP data. Lastly, the HRTimer class defines a high resolution timer used to calculate various performance metrics such as throughput and latency.

<u>*2.4.2 Client Package*</u>

The client package, called *ProjClient*, contains two files including ProjClient.java and DeviceView.java. Each of these files defines a graphical user interface (GUI) available in the NetBeans development environment. Initially when the client package is executed, an instance of the ProjClient.java class is created, which initializes the GUI illustrated in Figure 29.

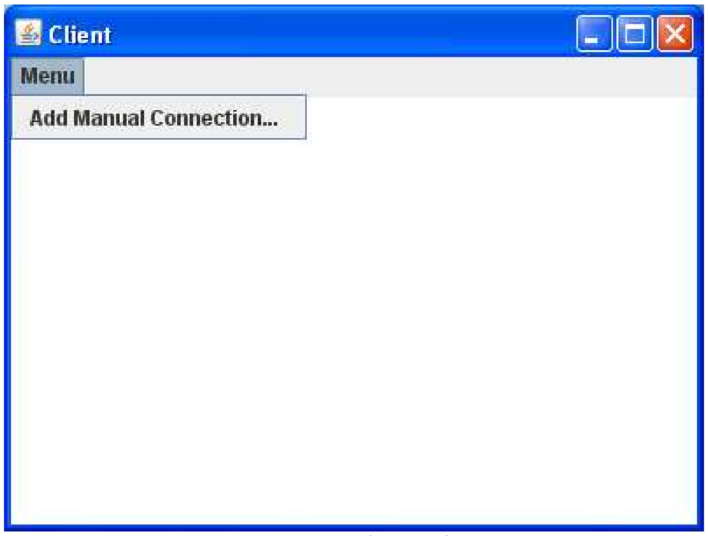


Figure 29: Client GUI

While this GUI is displayed, an instance of DiscoveryServiceClient.java (defined in *ProjLib*) is created. Essentially, the Discovery Service client searches for any nodes in the network that are running the Discovery Service server by sending out a broadcast request, *i.e.* flooding. The broadcast request will be heard by all nodes on the subnet, and if any of the endpoints are running the Discovery Service server, they will reply to the client. If tests are to be carried out between two nodes that are not on the same subnet, the “Add Manual Connection” option must be chosen in the GUI menu. This option will open a dialog box that allows the user to input an IP address that is assumed to be running the server side code. It is necessary to add the connection manually for nodes outside of the subnet because the broadcast requests sent by the Discovery Service client can only be heard inside the subnet. The Discovery Service will be explained in greater detail later in this section.

Once the Discovery Service client finds a node running the server side code, it will close the previous GUI, and create and instance of the DeviceView.java class, which initializes the GUI shown in Figure 30. The Device View GUI shows the IP address of the node that is running the server side code along with the host name of the system. Once this GUI has been initialized, the measurement tools can be run by clicking on one of the buttons at the bottom of the window.

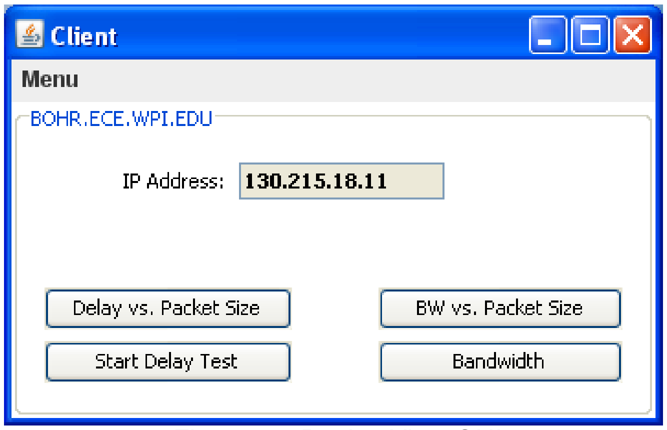


Figure 30: Device View GUI

2.4.3 Server Package

The server package, named *ProjServer*, only contains one file, and is much simpler than the client package. The only file in the server package is Main.java and it is used to initialize the server side code for each of the individual measurement tools. Although the server does display a GUI window when run, it is simply a blank window with no active menus or buttons. The GUI, shown in Figure 31, serves only to show that the server components are running.

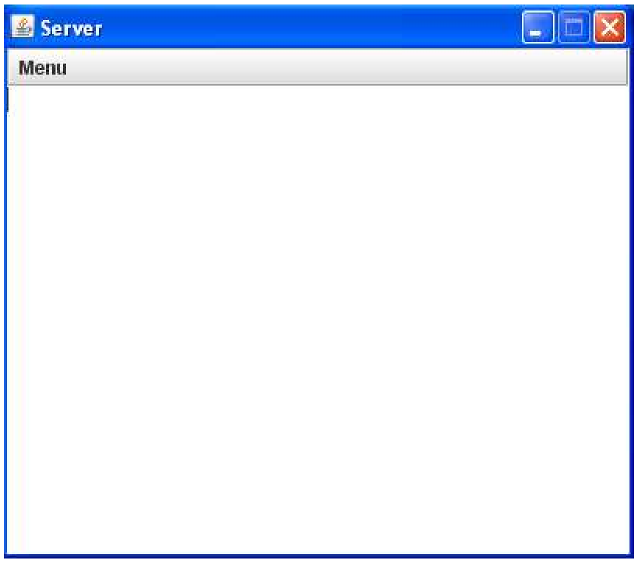


Figure 31: Server GUI

When the server package is started, an instance of each of the server side threads is started and configured to run on a specific port. Table 5 shows which port each tool is run on. Essentially, when an instance of a server side tool is created, it sits there and listens on its port for a request from a client to start a specific test. For example, if a user wishes to run a bandwidth test, the test is initiated by pressing the “Bandwidth” button on the Device View GUI (Figure 30), which will send a request to the server, which is listening on the specified port. Once the server receives the request, the test will begin.

Table 5: Port Assignments for Server Side Threads

Tool	Port
DiscoveryServiceServer.java	10011
PSDelayServer.java	10013
DelayServer.java	10015
BWServer.java	10019
PSBWServer.java	10020

2.4.4 Messages

For this project, a custom message class was created which defines the format of the messages used in each of the measurement tools. The message class, called Message.java, is defined the in the *ProjLib* package. Each message has three fields including a sequence number, a payload length, and lastly a variable sized payload. Figure 32 shows the structure of the messages used in this application.

BYTES							
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Sequence Number				**Payload Length**			
Payload (Variable Size) **:** **:**							

Figure 32: Java Toolbox Message Structure

The sequence number field is used during tests to determine if packets are lost or received out of order. The payload size field specifies the size of the payload while the payload itself contains the actual data to be transferred in the message. The Message.java class also contains a number of functions that operate on the message structure. These functions allow the user to manipulate the fields of a message as well as extract the different fields upon reception of a message.

Lastly, it should be noted that the maximum payload size for a message to fit into one UDP datagram is 1464 bytes. The maximum size allowable for a UDP datagram is limited by the Maximum Transmission Unit (MTU) of the Ethernet protocol which is 1500. The overhead required to send a UDP datagram is 28 bytes including a 20 byte IP header and an 8 byte UDP header. After the sequence number field (4 bytes) and the payload length field (4 bytes) are added, that leaves 1464 bytes available for the message payload. If the message payload exceeds 1464 bytes, the message will be fragmented into two separate UDP datagrams.

2.4.5 Output Data

All of the results from the tests are output to a text file after the test has been finished. While the test is running, the software writes the data that is to be captured to a buffer, and then flushes the buffer to a text file periodically during the test. In all of the tests, the measurements are computed on the server side node; therefore the output files are also generated and stored on the server system. In cases where computations or measurements are made on the client system, the data is sent in a message to the server for logging. The text files are formatted in such a way that it is easy to organize and plot the data. Normally, the text files are imported into MATLAB where a custom MATLAB script is used to manipulate and plot the data depending on the type of test that was run. The names of the output files include the type of test run as well as a timestamp so they are all unique and easily organized.

2.4.6 Timing

By default, the best time resolution that can be achieved through the Java SE 6 API is based directly on the underlying operating system (OS) that is running the application. This is because the method used to obtain the system time (`System.currentTimeMillis()`) is only as accurate as the system clock of the operating system. For example, the `System.currentTimeMillis()` method can achieve 1 ms resolution on a Linux OS while Windows 98 suffers from 50 ms resolution. On the Microsoft Windows XP OS, used on the portable ultrasound system, the system clock is updated about 64 times per second, or once every 15.625 ms which is the best time resolution that could be achieved using the Java API. When trying to determine delay and throughput characteristics of a channel, this poor time resolution can lead unrealistic and inaccurate measurements. Figure 33 shows round trip delay times measured using Window's timestamps. The test was run on the network emulator with a forward and reverse delay of 15 ms, and a 2 ms jitter in each direction. The measured values tend to be one of three values (16, 32 or 48) due to the behavior of the system timer on Windows XP.

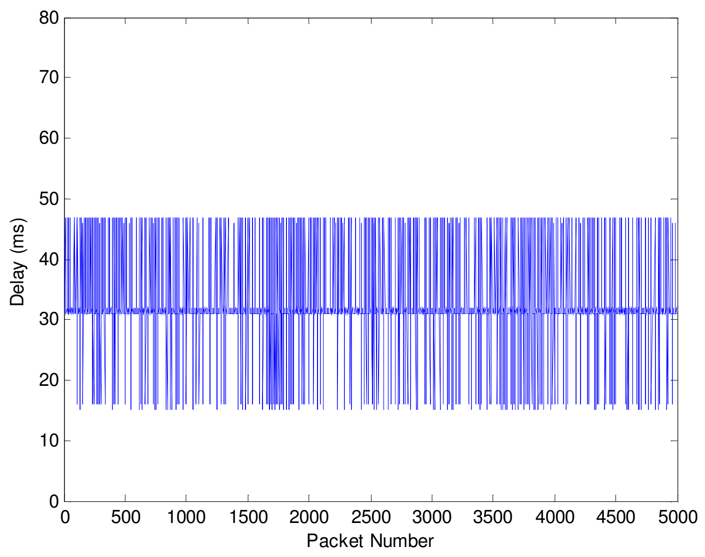


Figure 33: Delay Test Using Window's Timestamps

For better measurement accuracy, a new method for obtaining time information was developed. Initially, a custom timer using the Java API was written. This custom timer created a new thread that would go to sleep for 1 ms, and then update a counter by one in a continuous loop. The value of the counter could then be accessed at will and used as a timestamp. Although this method appeared to have better resolution than the system clock, thread scheduling in Java lacks the guarantees necessary to make this a reliable solution. For example, sometimes Java would suspend the timer thread for one reason or another which caused the clock to randomly stop.

To overcome this problem, a non-Java solution was used. For a high precision timer, a simple Java Native Interface (JNI) was used to access a Win32 method, which returns precision timing information. The JNI allows the Java application to call a method written in the C programming language. The C method `getTime()` wraps a simple Win32 system call `QueryPerformanceCounter()` which returns the number of CPU clock cycles since power up. The counter value can then be divided by the CPU clock frequency to determine the amount of time that has elapsed since power up. This counter is updated at a much higher frequency than the system clock, and therefore provides much more precise time information (about 1 ms resolution). Figure 34 displays a simplified model of the Java Native Interface.

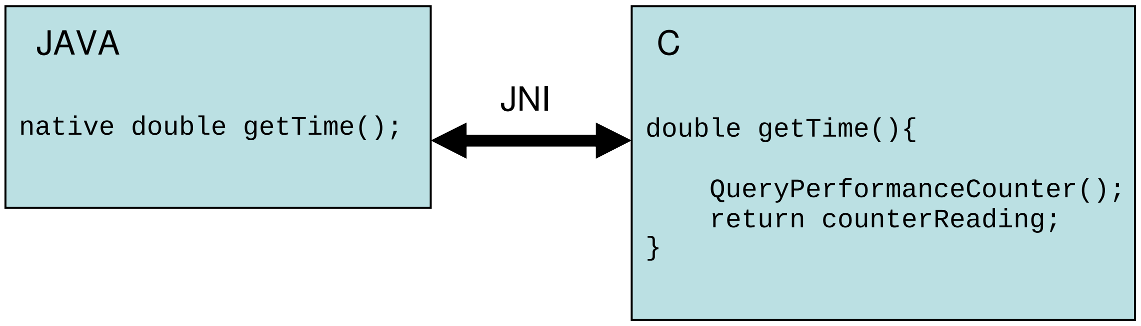


Figure 34: Java Native Interface for Precision Timer

A timer library was written in Java using the `getTime()` method to obtain time information. The library contains methods to stop, start, and reset the timer. There is also a method that returns the amount of time that has elapsed since the timer was started which provides the “timestamps” used in all of the measurement tools. Although these timestamps are not the true wall clock time, they are sufficient for bandwidth and delay calculations because only the difference between two timestamps is necessary. Figure 35 shows round trip delay times measured using the high precision timer on the same channel as those measured in Figure 33. It is evident the there is a great improvement in the precision when compared to the results obtained using Window’s timestamps.

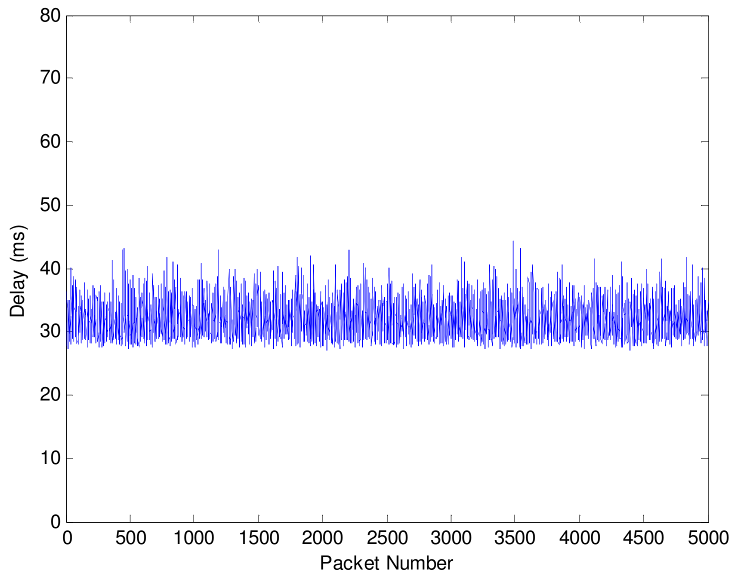


Figure 35: Delay Test with High Resolution Timer

2.4.7 Performance Evaluation Toolbox

The main functions of the performance evaluation toolbox include bandwidth measurements, latency measurements, and a Discovery Service that automatically creates a connection to any nodes within a subnet that are running server side code. This section will give a more detailed description of each of the measurement tools included in the software. As mentioned earlier, each measurement tool requires both client and server side code. The server side code is initialized when the server package is run, where it listens on a specific port for a request to start a test. The client side code is started once one of the buttons on the Device View window is selected, and a request to start a test is sent to the server. Lastly, all of these classes are defined in the ProjLib package.

Discovery Service

The Discovery Service is used by a client to locate and connect to any nodes on the subnet that are running the server side code. The Discovery Service server thread is started when the server package (*ProjServer*) is run, and it is configured to listen on port 10011 by default. To listen on a specific port, a datagram socket had to be created and bound to the desired port. This was done by calling the `DatagramSocket`() constructor from the Java SE 6 API [15]. Once a datagram socket has been open and bound to a port, datagrams can be sent and received from that port. Once the server is started, the application sits and waits for a request message to be received.

The Discovery Service client thread is automatically started when the client package (*ProjClient*) is run and the Client GUI (Figure 29) is displayed. Like the server, the Discovery Service client must also create a new datagram socket and bind to a port so it can send and receive datagrams. Once it has set up its socket, the client periodically (once every thirty seconds) broadcasts request messages that contains nothing more than the string, “ISULTRASOUND?”. It does this by sending the request message to the IP address, “255.255.255.255.” This message will be heard by all nodes on the subnet.

While the Discovery Service server is listening on its port, it will discard all datagrams that do not match the string, “ISULTRASOUND?”. Once it receives a valid request from a Discovery Service client, it will copy the client’s address and port, and reply back with a datagram that contains the string, “OK”. After the client receives a response from the server, it will close the Client GUI (Figure 29) and open a Device View GUI (Figure 30) that will now allow any of the measurement tests to be run. The Device View window will contain the IP address of the node it has found using the Discovery Service along with its hostname. Figure 36 displays a flow chart of the Discovery Service client. The server simply replies back if it receives the broadcasted request message.

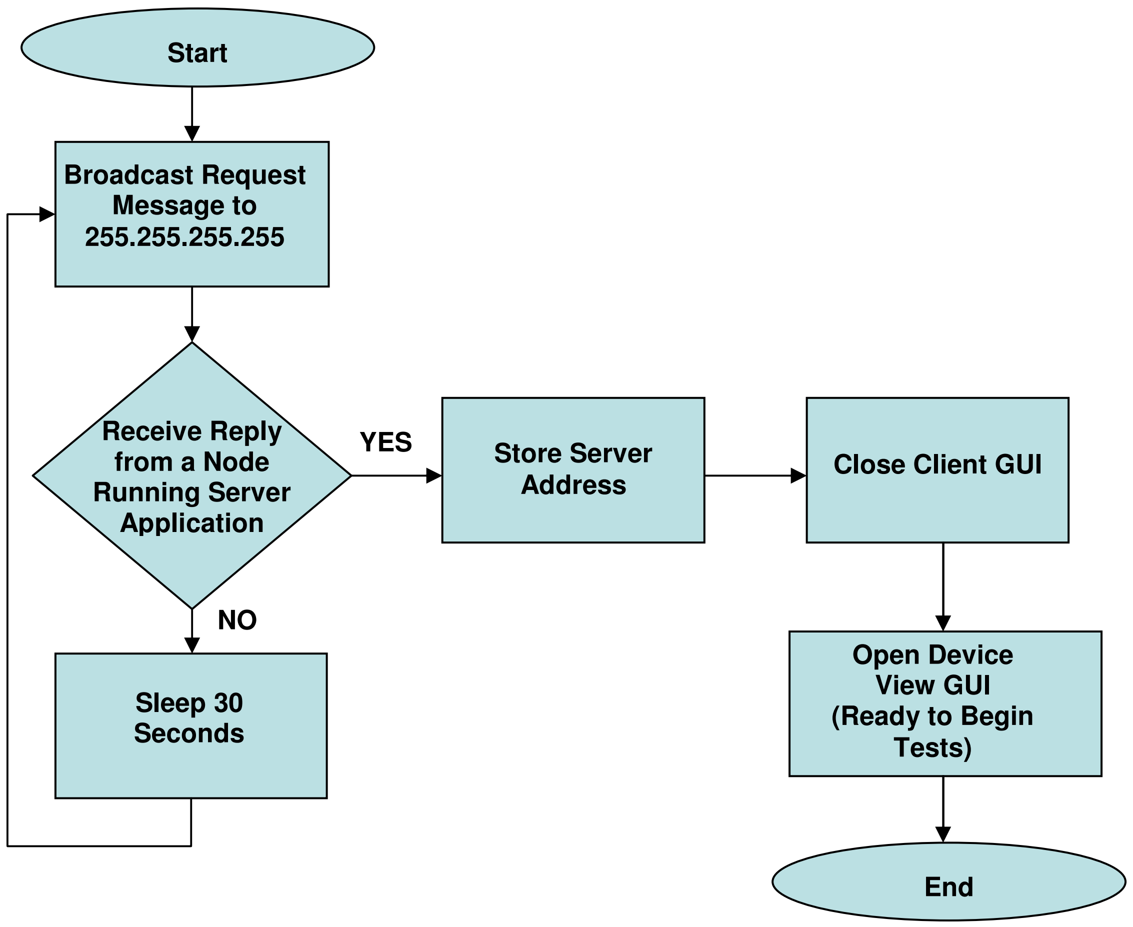

Start
Broadcast Request Message to 255.255.255.255
Receive Reply from a Node Running Server Application
YES
Store Server Address
Close Client GUI
NO
Sleep 30 Seconds
Open Device View GUI (Ready to Begin Tests)
End

Figure 36: Discovery Service Flow Chart (Client Only)

The Discovery Service client also contains a function that allows nodes to be searched for manually. This can be done by selecting the "Add Manual Connection" option in the Device View Menu and entering the IP address of the endpoint running server side code. If tests are to be run between two hosts that are not on the same subnet, this option must be chosen. All that this does is force the Discovery Service client to send a request message to a specific IP address rather than to the broadcast address because the broadcast message will not be heard outside of the subnet.

Delay

Calculating and recording the latency between two endpoints is one of the major functions of the toolbox. The first delay measurement tool defined in DelayServer.java and DelayClient.java, sends a set number of packets, all of which are the same size, and measures the latency of these packets.

As with the Discovery Service server, the Delay Server (DelayServer.java) opens a datagram socket and binds to a port when the server package is started. A client can begin a delay test by clicking the “Start Delay Test” option on the Device View GUI. When that option is selected, an instance of the Delay Client thread, defined in DelayClient.java, will be started on the client node.

After an instance of DelayClient.java is initiated, a request message in the form of the string, “DELAY_START?” will be sent to the server, and the test will be started. The server will then create a message of a predefined size, record a timestamp (server_ts_s), and send the message to the client. Upon reception, the client system will immediately record a timestamp (client_ts_r). The client node will then form a messages structure of its own, and insert the reception timestamp into the payload of the message. Lastly, the client will take one more timestamp (client_ts_s), insert it into the message payload and send it back to the server. After the server receives the message, it will record a final timestamp (server_ts_r).

The server system will then calculate the forward delay (client_ts_r – server_ts_s), the reverse delay (server_ts_r – client_ts_s) and the round trip delay ([server_ts_r – server_ts_s] – [client_ts_s - client_ts_r]) which are all in milliseconds. It will then write these values to a buffer that will later be written to a text file. The process will then be started again and continued for a specified number of packets. The following figure depicts how a test is run.

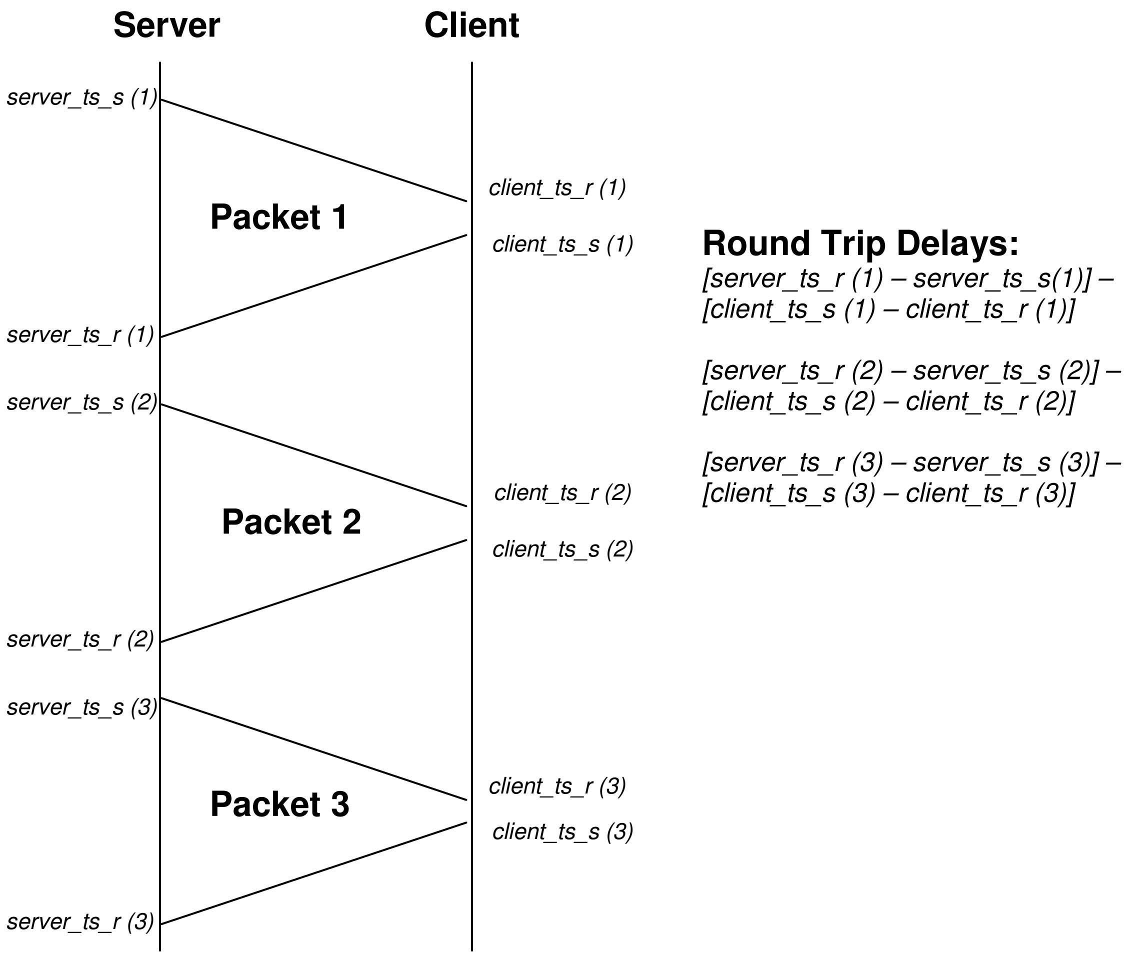
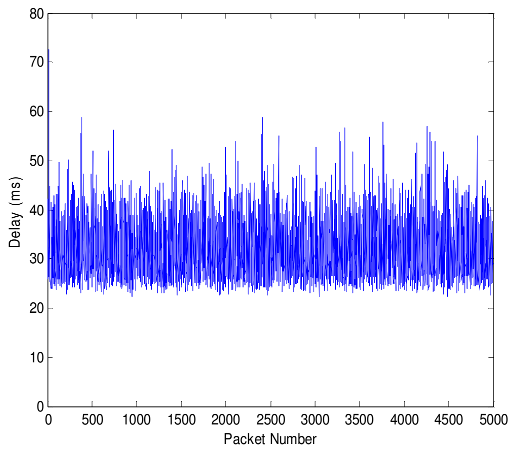


Figure 37: Delay Test Configuration

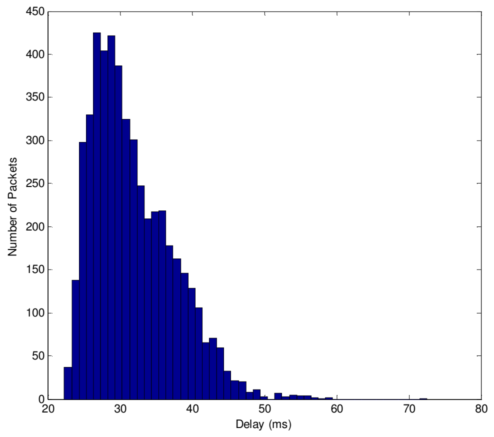
The number of round trip packets used in a single test run must be defined before the test is run. Also, the size of each packet is predefined, and does not change during the test run. The packet size must be in the range between 64 bytes and 1500 bytes, or the MTU. After the preset number of packets have been sent, the server will send a message containing the string, “LAST_MESSAGE”. Upon reception of this message, the client will reply back and close its socket. The server will close its socket, write the buffered delay values to a file, and re-open a socket to wait for another test.

As mentioned before, this measurement tool uses UDP as the transport protocol during testing, and this can result in lost packets. To account for this, the server uses a timeout option when receiving packets. After it sends a packet to the client, it calls a function that sits and waits to receive the reply message; however it will timeout if it does not receive the reply in a given amount of time. If this happens, either the forward packet or the reverse packet is assumed to be lost, and the server will send a new packet with a new timestamp. The timeout value must be significantly greater than the average round trip time of the link so as not to discard living packets with a larger than normal delay. For example, a timeout of 800 ms was used when testing an 802.11 link while it had to be increased to 2 seconds when testing a 3G connection. The number of dropped packets are also recorded and written to the output file.

Figure 38 shows test results from a sample delay test which was run on the network emulator. The emulator was arbitrarily configured to have a forward delay of 15 ms, a forward jitter of 6 ms, a reverse delay of 15 ms, a reverse jitter of 2 ms, and a total packet loss of 2% (1% in each direction). Figure 38 (a) shows the round trip delay times as a function of time. In total, 5000 round trip packets were used to obtain the measurements. Figure 38 (b) shows the distribution of the round trip delays. Analysis of the data showed a mean round trip time of 31.80 ms, a forward jitter of 5.77 ms, a reverse jitter of 1.94 ms, and a packet loss of 1.8% (90 out of 5000).



(a) Round Trip Delay vs. Time



(b) Delay Distribution

Figure 38: Sample Delay Tests Run on Network Emulator

Delay vs. Packet Size

The second delay tool, defined in PSDelayServer.java and PSDelayClient.java, examines how channel latency is affected by packet size. The PSDelay tool operates in a similar fashion as the standard Delay tool, only it periodically increases the size of the packets it sends. Like the Delay Server, the PSDelay Server starts its thread when the server package (ProjServer) is run. The thread opens a datagram socket on port 10013, and waits for a request from a client.

The PSDelay Client thread is started when the “Delay vs. Packet Size” option is selected on the Device View GUI. After the thread is started, a request message containing the string “PSD_START?” is sent to the server who responds to the client with the string “PSD_OK”. The test is now started, and it runs just as the standard delay test described in the previous section; the only difference being the size of the probe packets is periodically increased. The size of the initial set of packets, the number of increments made to the packet size, and the number of packets to send at each increment can be altered before a test is run. Typically, the first set of packets is around 100 bytes, and they are increased in increments of 50 bytes up to around 1500 bytes where it reaches the MTU. A set number of packets are sent at each interval and can be averaged to observe the effect of packet size on delay. This tool is helpful on lower bandwidth links (< 1 Mbps) to see how the delay is affected by the packet size. Figure 39 shows a sample delay vs. packet size test run on the network emulator. In this test, both the forward and reverse channels had a delay of 100 ms and a bandwidth of 16 kbps. The low bandwidth of the link is the primary reason that the delay increases as the packets grow in size. For this test, 25 packets were sent for each packet size, and the delays were averaged to obtain the curve shown in Figure 39.

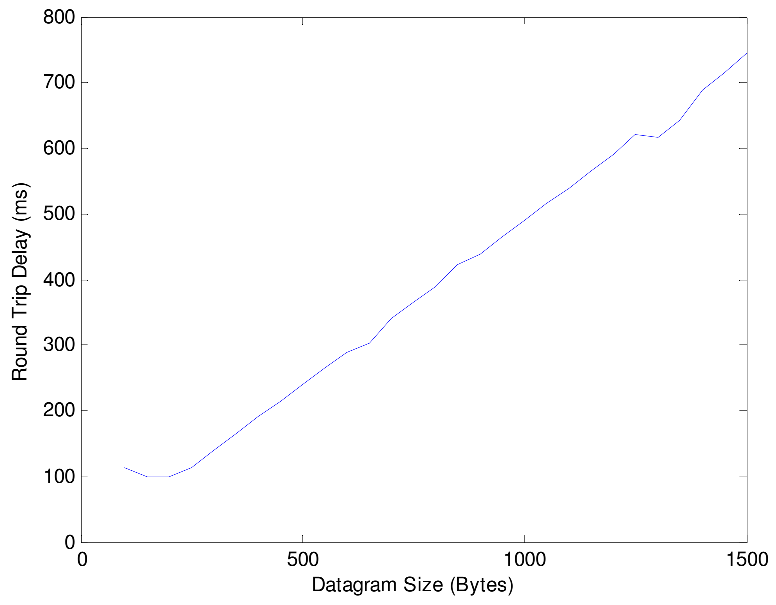


Figure 39: Sample Delay vs. Packet Size Test Generated from Network Emulation

Channel Capacity

The measurement toolbox also contains functions to measure the capacity of the wireless channel. The first tool uses UDP packets to measure the raw channel capacity or bandwidth of the link. Similar to the delay tools, the bandwidth server (BWServer.java) is started when the server package is started, and a test begins when the client (BWClient.java) sends a request message ("BW_START?") to the server.

The test parameters must be adjusted depending on the type of link that is being measured. On a single-hop channel, such as an ad-hoc 802.11 network, bandwidth tests are run as follows. Once a test is started, the server takes a timestamp (ts_start), and sends set number of packets to the client. The size of the packets can be altered, but are typically around the MTU or 1522 bytes. The server sends the packets as fast as the channel will allow it to. After all of the packets have been sent, another timestamp is taken (ts_end). The channel capacity can then be obtained by dividing the total amount of data sent (num_packets x 1522 bytes) by the total time it took to send the data (ts_end – ts_start). The same method is then applied in reverse where the client sends a number of packets to the server, and determines the reverse bandwidth by dividing the total amount of data sent by the amount of time it took to send it. Lastly, the number of packets lost during each test are also monitored and recorded. Figure 40 shows a sample throughput test run on the network emulator. The emulator was set to have a bandwidth of 10 Mbps, and the results show the measured throughput over a thirty second time span. It appears as if the measured bandwidth is slightly higher than the expected bandwidth which could be attributed to minor inaccuracies in the network emulation software.

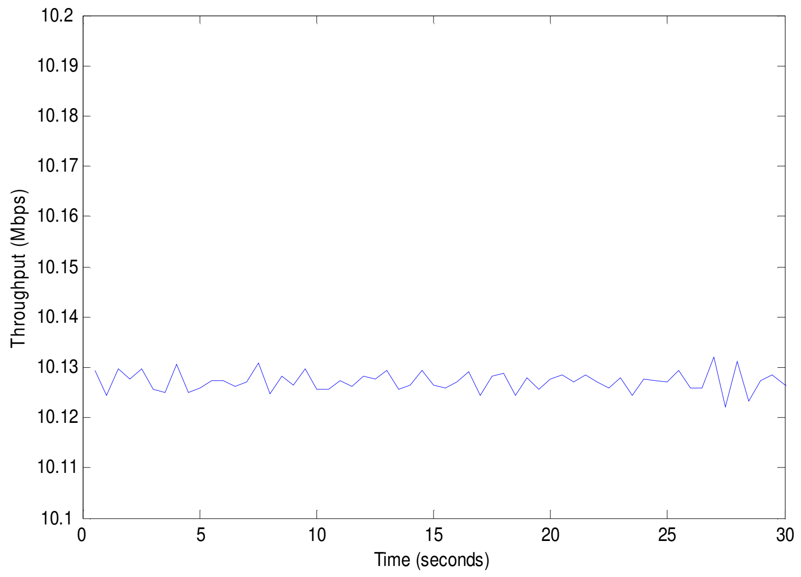


Figure 40: Sample Throughput Test

Determining the raw channel capacity on multi-hop channels requires a different methodology. Figure 41 shows a hypothetical test configuration for a multi-hop channel. This configuration is common for wireless links such as satellite and cellular networks where data is routed from the wireless medium to the internet. It can be seen that the client system is mobile, and is connected to the internet via a wireless link. The server on the other hand has internet connectivity through a traditional wired cable such as a 100 Mbps Ethernet interface.

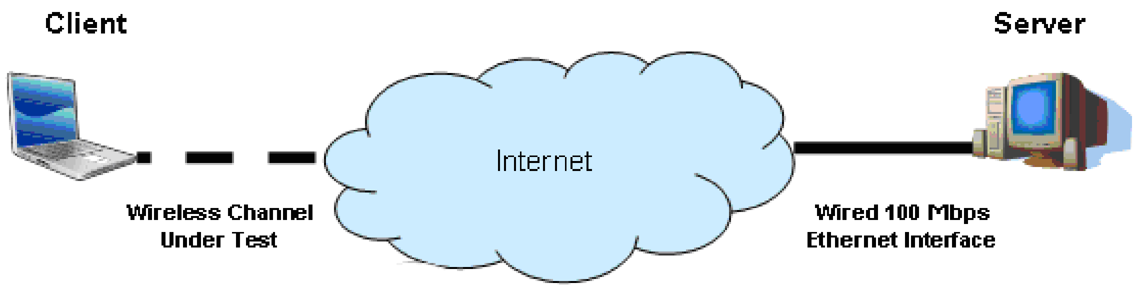

Client
Server
Internet
Wireless Channel Under Test
Wired 100 Mbps Ethernet Interface

Figure 41: Hypothetical Multi-hop Test Configuration

In a multi-hop setup, the forward bandwidth (client to server) is determined in the same manner as the single-hop configuration. The client transmits packets as quickly as the channel will allow, and the bandwidth can then be computed. Going in the reverse direction (server to client) requires a little more work. If the same methodology was used, the server would put packets onto the wired medium as fast as it would allow, which has a much higher channel capacity than the “bottleneck” in the wireless link. Instead of throttling itself, the server would flood the channel with packets resulting in a large percentage of dropped packets at the receiving end.

To determine the reverse channel capacity, the server initially sends packets at a predefined data rate that will likely be less than the maximum bandwidth of the wireless link. After the completion of the first test, the data rate will then be slightly increased. The number of data rate intervals that are tested along with the data rate at each interval must be configured before the test. The forward channel capacity can then be determined by examining the number of packets that are dropped at each interval. When the number of dropped packets begins to dramatically increase, it is due to sending data at a higher rate than the maximum supported bandwidth of the wireless link. Figure 42 shows the results of a sample test run on a 3G cellular network. Five intervals in the range between 850 and 2000 Kbps were tested, and 2000 packets were sent at each interval. The number of dropped packets was then plotted as a function of data rate. From this data, it is evident that the channel capacity of the reverse link is slightly above 1200 Kbps.

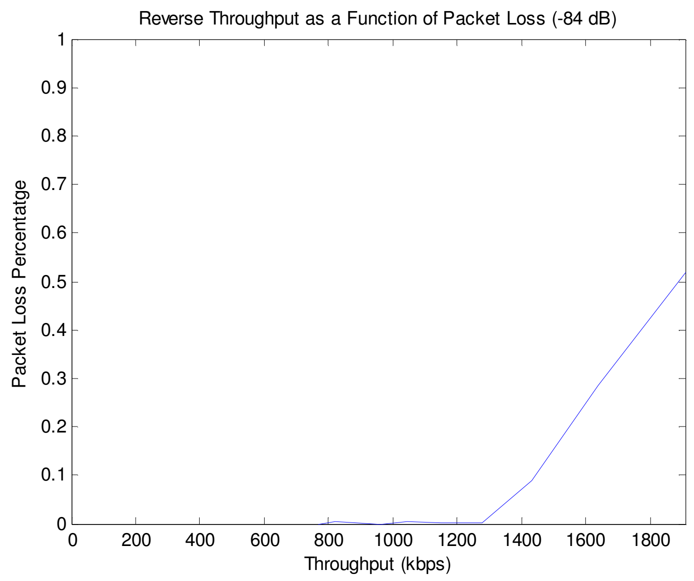


Figure 42: Sample Reverse Bandwidth Test on 3G Network (-84 dBm)

<u>Channel Capacity vs. Packet Size</u>

The last function of the Java application is to measure channel capacity as a function of packet size. This tool only applies to the reverse flow of traffic (client to sever) as that will be the primary direction of data for the portable ultrasound unit. Like each of the other tools, the server thread (PSBWServer.java) begins when the server package (*ProjServer*) is started, and a test begins when the client (PSBWClient.java) sends a request message ("PSBW_START?") to the server. When a test is started, 200 byte packets are sent for a given period of time, and the maximum data rate for that packet size can be determined by dividing the total amount of data sent by the time it took to send the data. Tests are carried out in this manner increasing packet sizes by 200 bytes, up to 1400 bytes packets. As in the previous tests, dropped packets are counted and recorded. Figure 43 shows a sample bandwidth vs. packet size test run on a 3G cellular network with packets ranging between 200 and 1400 bytes. The results from this specific test indicate that the UDP throughput is more or less independent of the size of the packets being transmitted.

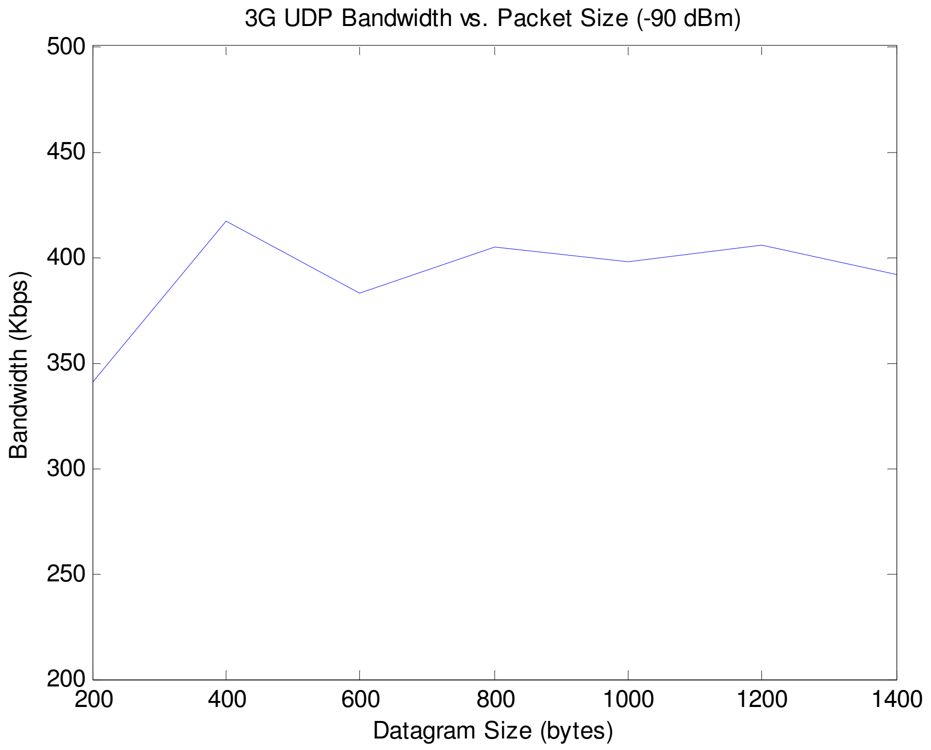


Figure 43: Sample Throughput vs. Packet Size Test on 3G Network

<u>*2.4.8 Measuring Jitter*</u>

There are a number of existing methods to measure jitter across a network; however most techniques contain fundamental flaws that can lead to inaccurate jitter analysis. For example, one popular method called the *Inter-arrival Method* [49] transmits packets at a known constant interval and measures the inter-arrival times at the receiving end. By transmitting packets at constant intervals, two of the four parameters (Tx_A and Tx_B) necessary to calculate jitter are known. The inter-arrival times at the receiving end can then be used to produce a jitter profile of the link. The main problem with this method is that lost or out-of-order packets are not accounted for. Dropped packets will lead to very high jitter values that corrupt the accuracy of the measurements. Also, packets have to be sent at perfectly equal intervals to maintain accurate calculations. Another common method is to capture all of the received packets and perform jitter calculations after a test has been completed. This method is not suited for long tests, and it becomes tedious going through each packet to retrieve the necessary timestamps.

To provide a set of industry standard definitions, the Metro Ethernet Forum (MEF) released the MEF 10 specification in 2004 [49]. This specification contains a section that describes the proper way to measure jitter while taking into account lost or corrupt packets, and this is the technique that was implemented to measure jitter in this project. Figure 44 illustrates how jitter is calculated at a network endpoint according to the MEF 10 specification.

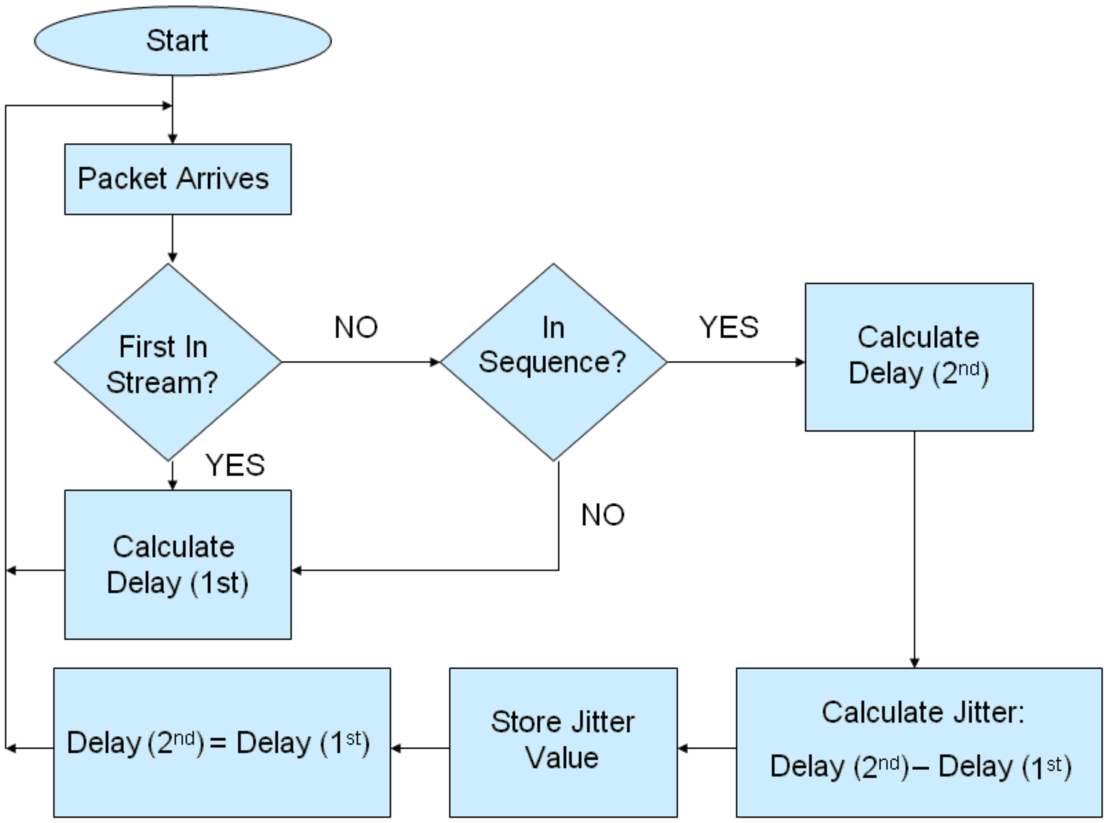
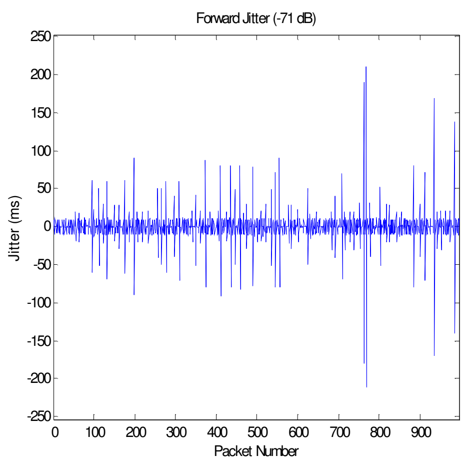


Figure 44: Jitter Measurement Flow Chart [49]

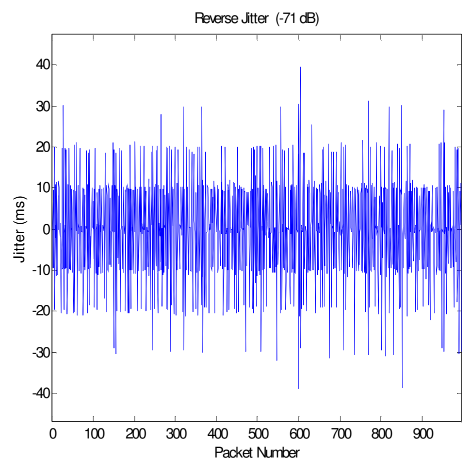
When a packet arrives, a check is performed to determine if it is the first packet in the stream. If it is, then the latency is measured by subtracting the transmit time which was inserted into the message from the receive time on the local system. This value is then stored. If it is not the first packet in the stream, then a check is performed to determine if the message is in sequence using the sequence number field in the message structure. If it is in sequence, then it means no packets were dropped. The latency of this packet is then measured. The jitter value for this packet pair is calculated and stored, and the delay for the most recent packet takes the place of the previous packet. If a packet is received out of sequence, then either a packet has been dropped or received out of order. If this is the case, then the jitter value for the corresponding packet pair is discarded and the algorithm is restarted.

One of the main advantages of using this jitter measurement system is that packets do not have to be transmitted at a constant interval. This enables jitter to be measured at different data rates and packet sizes which isn’t possible using the Inter-arrival Method. Additionally, jitter is measured in real time giving a clear view of the behavior of the latency of the channel.

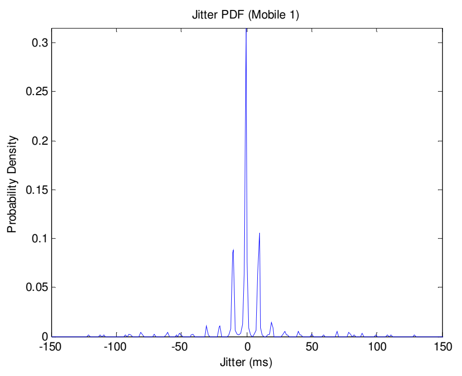
This algorithm is implemented on each of the previously described functions of the toolbox. By doing this, jitter analysis under different traffic loads and packet sizes can be made. Figure 45 (a) and (b) show jitter measurements made on a 3G network. The graph on the left shows the measured forward jitter while the one on the right shows the reverse jitter. In addition to the raw jitter measurement, the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) can be plotted to observe the distribution of the jitter. Figure 45 (c) through (f) show these plots. It should be noted that the absolute value of the raw jitter measurements are used when plotting the CDF as it is easier to interpret the distribution this way.



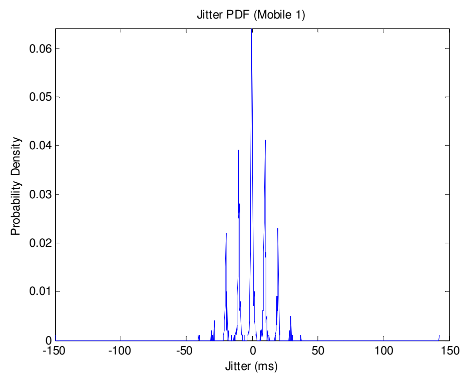
(a) Measured Jitter on Forward Link



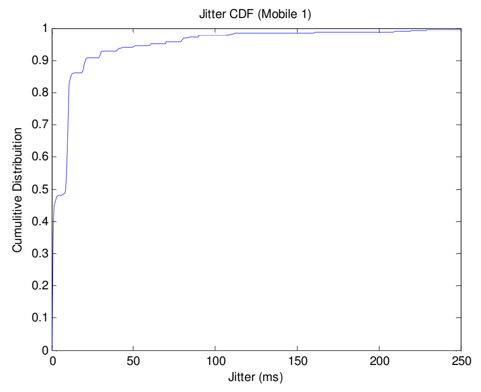
(a) Measured Jitter on Reverse Link



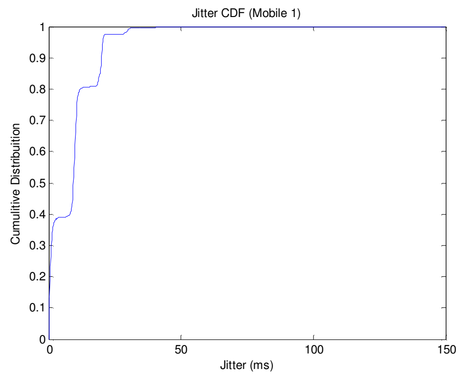
(c) Forward Jitter PDF



(d) Reverse Jitter PDF



(e) Forward Jitter CDF



(f) Reverse Jitter CDF

Figure 45: Sample Jitter Results on Uplink and Downlink on 3G Network (-71 dBm)

2.5 TCP Measurements

The only metric that is unattainable through the use of the Java measurement toolbox is TCP bandwidth. To measure TCP bandwidth, an open source tool developed by the National Laboratory for Applied Network Research (NLANR) called *Iperf* was used. Iperf was designed to optimize TCP connections by tweaking different parameters such as the window size and segment length. It has additional utilities to measure some UDP characteristics; however, those tools were not used in this project [26].

The Iperf package comes precompiled, and must be run from a command line utility such as MS-DOS. Like the Java program, one network endpoint must run a server application, and another must run a client application to perform a test. Iperf provides the ability to change the maximum TCP window size as well as the MSS (maximum segment size) for each test. When a test is started, Iperf sends data from the client to the server using TCP. An option is available to run a test in the opposite direction (server to client) after the initial test has finished. During each test, the data rate is continuously monitored, and the test results are exported to a text file on the server system after a test has been completed. Figure 46 shows the results of a thirty second TCP bandwidth test on an 802.11g link with a received SNR of 39 dB, which is a relatively strong signal. The link had an average TCP bandwidth of 19.03 Mbps for the thirty second duration of the test.

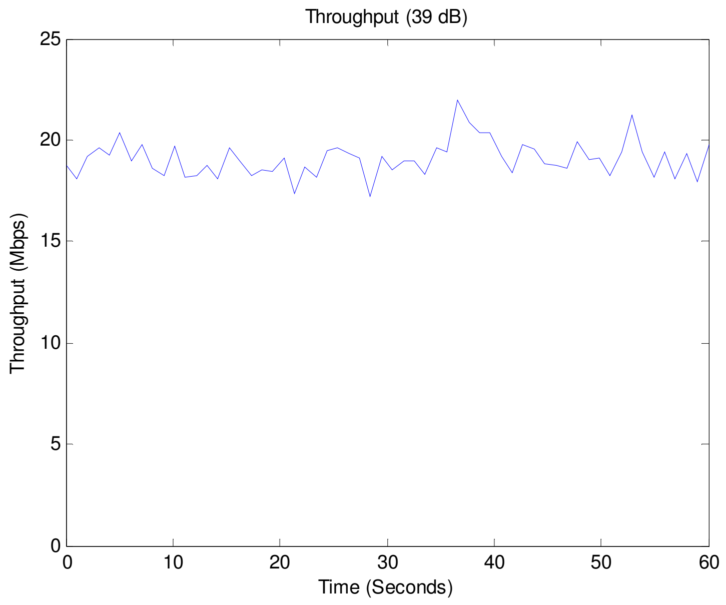


Figure 46: Example TCP Bandwidth Test on an 802.11g Link

Although Iperf allows users to tweak the TCP parameters for each test, it should be noted that all tests were done with default Windows XP TCP settings. Different operating systems possess different TCP implementations, and because the portable ultrasound unit runs Windows XP, it was decided to use the default TCP parameters provided by the operating system. By default, Windows XP uses a MSS of 1460 bytes and a maximum window size of 8KB. It uses the Smoothed Round Trip Time algorithm, as given in (2), to calculate the expected round trip time of a segment. Lastly, it uses the slow start with congestion control technique described in Section 2.1.1 after setting up new connections and on timeouts. The only difference is that XP has an initial window size of two segments rather than one as shown in the example (Figure 21) [27].

2.6 Measuring One-Way Delay

When compared to measuring round-trip delays, calculating one-way delays is a much more difficult task. A number of methods exist for one-way delay estimation; however, most are just approximations that make some fairly liberal assumptions about the channel while others do not provide enough precision for reliable analysis of the link. This section will discuss some existing one-way delay measurement options and the problems associated with each one. It will conclude with an explanation of the technique that was employed in this thesis to obtain one-way delay measurements.

Measuring round-trip latency is simple because all of the time measurements are computed on the same system. The main problem when trying to measure one-way delay is that the system clocks on the two network endpoints are generally not synchronized, and even if they are, they will most likely drift apart over time. This is true even when using the timer from the Java toolbox because it would be near impossible to start both timers at the exact same moment. Figure 47 shows how forward and reverse delays could be calculated if both the client and server clocks were precisely synchronized and there was no drift between the clocks.

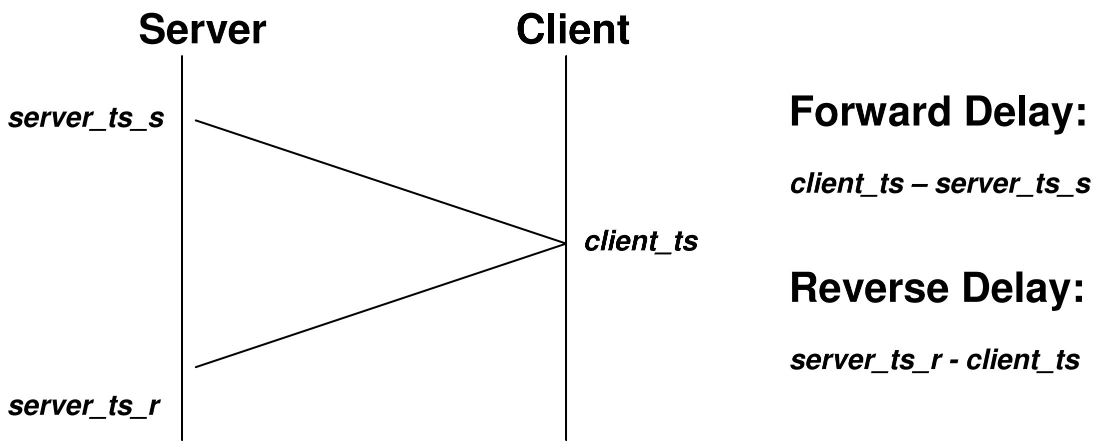


Figure 47: One-Way Delay Diagram

Figure 48 shows actual test results from the measurement method shown above. These results were obtained on an 802.11g wireless link, and the test was run for a duration of two minutes. If taken literally, the results show that the link has a reverse delay of around 62 ms that slowly decreases over time. It is evident that this is impossible given that the round-trip delay has a mean of 1.56 ms and is pretty much constant over time. By reexamining the one-way measurements, it can be concluded that the two systems had a clock offset of around 62 ms when the test started (y-intercept), and they are drifting together at a rate of about 2 ms/min; however this information provides no useful insight into the forward and reverse latencies of the channel.

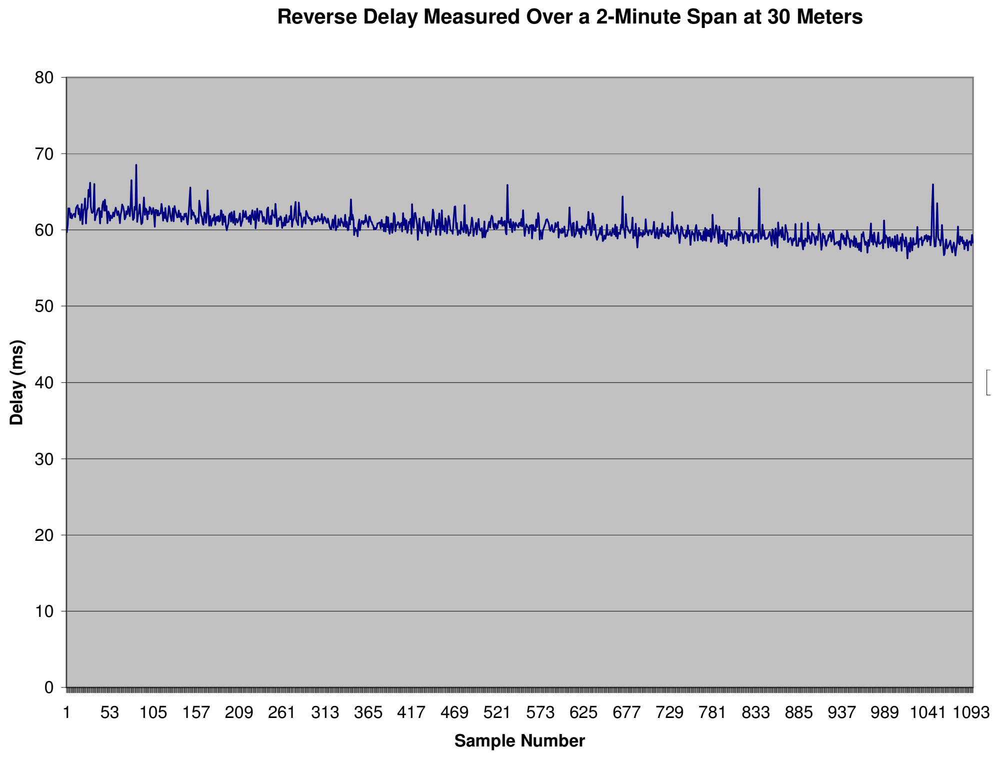


Figure 48: Example One-Way Delay Data

One approach to measuring one-way delay is to try to remove the offset between the two systems by precisely synchronizing both of the clocks. One method to do this is to use Network Time Protocol (NTP) servers on the internet for synchronization. The NTP was developed to synchronize system clocks to the time of some NTP server; however it does not provide enough precision for accurate one-way delay measurements. NTP assumes symmetric links when synchronizing nodes, and therefore introduces errors. The uncertainty of NTP synchronization is between 10 and 100 ms which is too great for measuring one-way delay [28]. Additionally, a continuous internet connection is necessary to communicate with the NTP server, and this will not always be possible during testing.

Another option is to use Global Positioning System (GPS) receivers on each node for clock synchronization. Although this method can be more accurate than using NTP servers, many factors contribute to the limited accuracy that can be attained using GPS. First, an appropriate GPS receiver must be chosen. For time synchronization applications, a PPS (pulse per second) signal from the GPS receiver is necessary which is not available on all receivers. The PPS signal is sent by the GPS receiver once per second, and it carries the time information that it has received from the GPS satellites. The accuracy of the GPS receiver itself is not overly important because the majority of the error will be from the operating system on the machine being synchronized. The best receivers have nanosecond accuracy while the lower end receivers are still accurate to around 50 microseconds.

Some delay tests were performed using GPS clock synchronization on both of the test computers. The GPS receiver that was used was a Garmin 18 LVC receiver, and the configuration used to update the system clock of a computer can be seen in Figure 49. In addition to the GPS receiver, each system also needed special software that was able to take the PPS signal from the GPS receiver and reset the system clock. The software that was used for this purpose was the *Totally Accurate System Clock (TAC32)* from CNS Systems, INC [30].

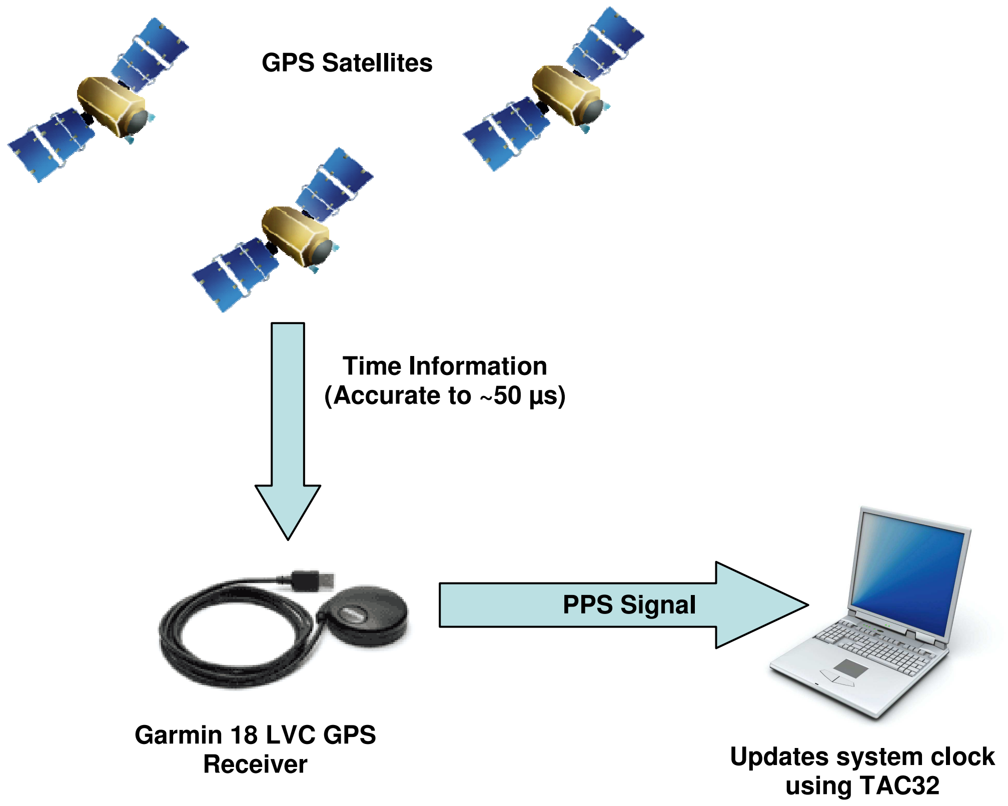


Figure 49: GPS Clock Synchronization using Garmin 18 LVC Receiver

After running some tests, it was apparent that the GPS receivers would not provide the precision necessary for accurate delay measurements. The problem was not the accuracy of the GPS receiver, rather it was that Windows XP is not a real-time operating system and therefore does not make any scheduling guarantees. This means that even though the GPS receiver sent an accurate PPS signal precisely once every second, the test system added error to the time measurement because it did not update its clock precisely upon reception of the PPS signal due to the manner in which Windows schedules processes for execution. The tests showed that each of the two test systems were synchronized to within plus or minus 30 ms of the GPS time resulting in a maximum clock offset of 60 ms between the two systems. This offset is unacceptable for accurate one-way delay measurements. Better accuracy is possible on real-time operating systems where the scheduling of resetting the system time can be guaranteed; however these alternatives were not tested.

Additional algorithms that attempt to approximate the offset were investigated; however, most of the techniques made some assumptions about the link that are not always true. For example, one method that was examined assumed that the offset remained constant for the duration of the test (no drift) and that the reverse and forward paths were identical. In practical applications, these assumptions cannot be guaranteed, and preliminary tests using such an algorithm produced unusable data.

After evaluating these alternative methods, an original technique that bypasses the problem of clock offset was implemented. The new method was set up so that the same computer took all of the time measurements, removing the problem of offset and drift when using two different computers. The new method is intended for measurements on cellular and satellite networks because these links have asymmetrical bandwidths which could possibly lead to asymmetric delays. Latency on point-to-point links such as 802.11 is generally symmetric and will not use the following method for one-way delay measurement. The new measurement system is configured as shown in Figure 50.

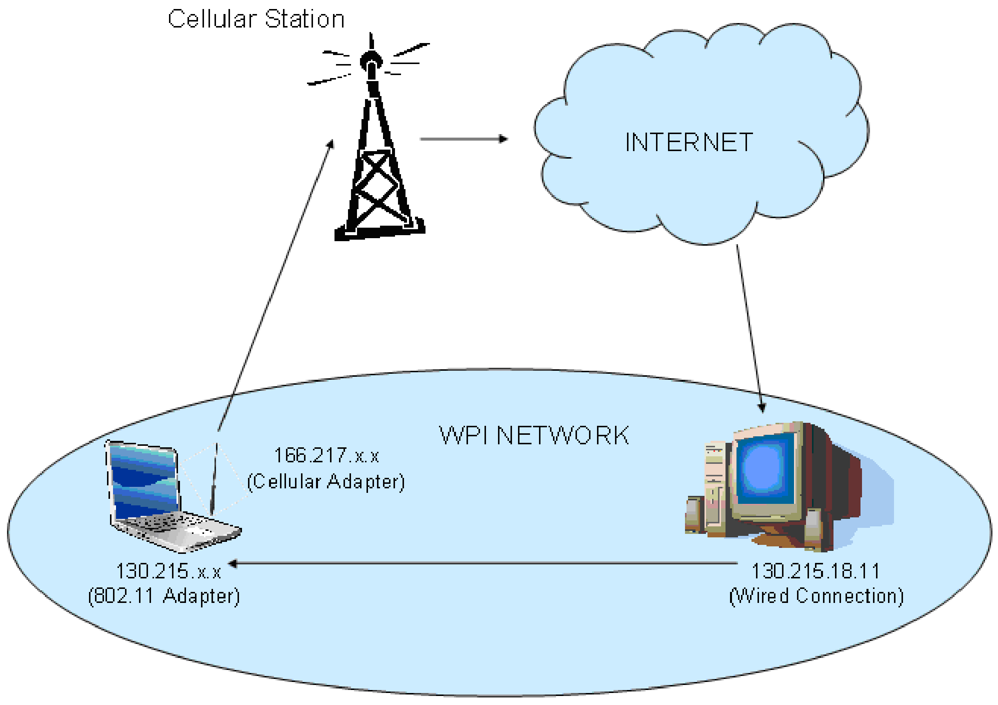


Figure 50: One-Way Delay Measurement System

This example shows the testing of a 3G cellular network; however the same configuration could be used for satellite networks as well. Using this setup, an initial timestamp is taken before a packet is sent (ts_s). The packet is then sent from the laptop using the 3G cellular network adapter. Next, the packet travels wirelessly to the cellular station where it is routed through the internet to the destination computer which resides on the WPI LAN. Rather than sending the packet back through the cellular network, it returns the packet to the sending system through the WPI network. Another timestamp is then taken (ts_r), and the total travel time of the packet can then be determined by taking ts_r – ts_s.

SInce the forward delay through the cellular network will have a much higher latency than the reverse delay through the LAN, the total travel time of the system is quite close to the forward delay of the link. For higher accuracy, half of the round-trip time of the LAN can be subtracted from the initial delay measurements for a closer approximation to the true forward delay. For example, consider the following data obtained on a 3G cellular link using the one-way delay measurement method just described.

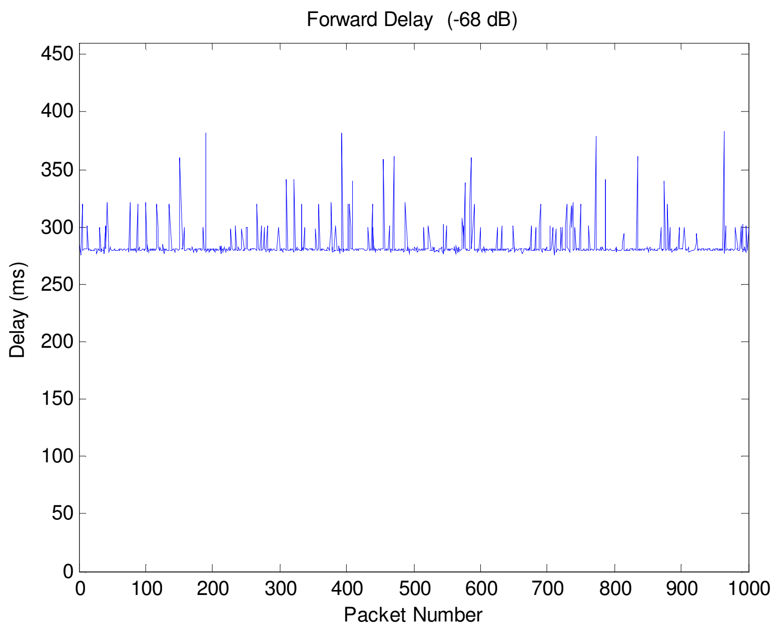


Figure 51: Example One-Way Delay Test on 3G Network (-95 dBm)

This data has a mean of 298.8 ms and a standard deviation of 13.8 ms. The round-trip time of packets sent both ways through the LAN was measured to have a mean of 5.2 ms and a standard deviation of 1.8 ms. If these distributions are assumed to be normal (N~(μ,σ^2)) with a mean μ of and a standard deviation of σ, then true forward delay and reverse delays can be derived. If the reverse trip through the LAN is assumed to be half of this round-trip time, then it would have a mean of 2.6 ms and a standard deviation of 1.27 ms. Therefore, the an approximation of the true forward delay can be obtained by subtracting half of the LAN round-trip time from original measurements resulting in a forward delay with a mean of 296.2 ms and a standard deviation of 13.74 ms. The previous results were derived from the calculations shown in Table 6.

Table 6: Sample One-way Delay Calculations

Measured Data	*A* ~ N(298.8, 190.44)
RTT Through LAN	*RTL* ~ N(5.2, 3.24)
Reverse Trip Through LAN (½ *RTL)*	*RL* ~ N(2.6, 1.62)
Forward Delay (*A – RL)*	*FD* ~ N(296.2, 188.82)

If both the forward delay and round-trip time of the link are known, then the reverse delay distribution can also be incurred. Figure 52 shows the round-trip delay distribution of packets sent both ways through the 3G network.

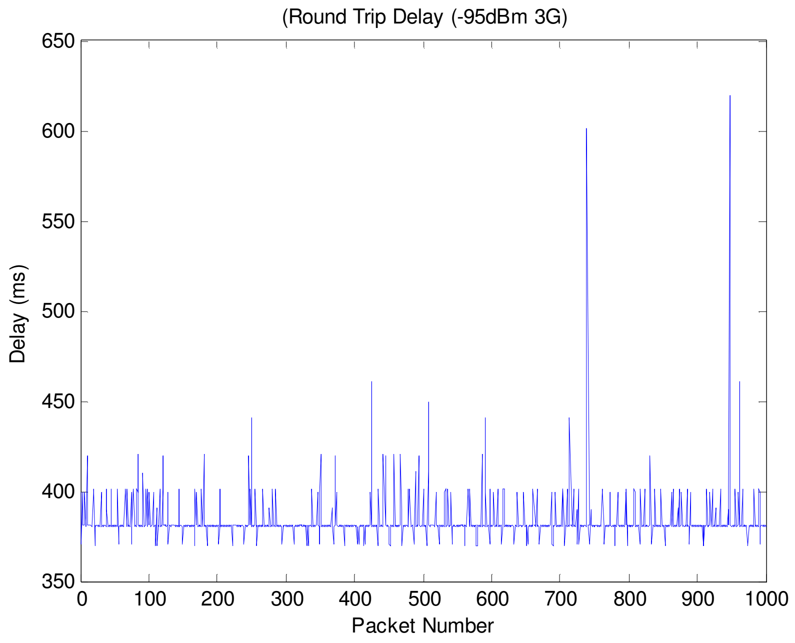


Figure 52: Round Trip Delay of 3G Network

This data has a mean of 383.6 ms and a standard deviation of 14.8 ms. The reverse link latency can be approximated by subtracting the forward delay from the round-trip delay. This results in a reverse link with a mean delay of 87.4 ms and a standard deviation of 5.5 ms. These calculations can be seen in Table 7.

Table 7: Approximation of Reverse Delay

Forward Delay	*FD* ~ N(296.2, 188.82)
RTT Through 3G Network	*RT3* ~ N(383.6, 219.04)
Reverse Delay (*RT3 – FD*)	*RL* ~ N(87.4, 30.22)

3 Wireless Communication via IEEE 802.11g

The first wireless technology that was analyzed for this thesis was based on the 802.11 standards from the *Institute of Electrical and Electronics Engineers* (IEEE). The 802.11 standards denote a set of regulations intended for deployment in wireless local area networks (WLAN), which are generally characterized by high data rates (> 1 Mbps) over relatively short distances (< 100 m). Although the range of 802.11 is relatively small, there are still possibilities for the technology in telemedicine. One area it could be used is in a military field hospital, where treatment may occur in close proximity to a base station. Another possibility is to use 802.11 as a gateway to some other means of communication. For example, consider the scenario depicted in Figure 53 where 802.11 technology could be used to send data to a nearby helicopter or ambulance which already has some built-in means of communication with a base station or hospital.

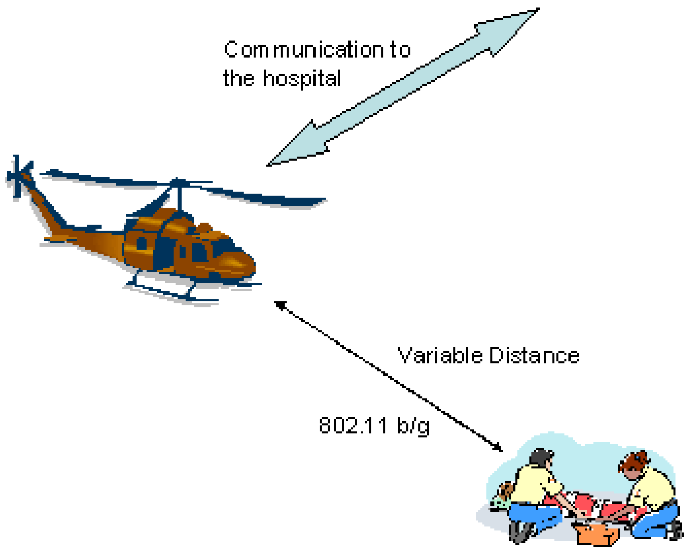


Figure 53: 802.11 Used as a Gateway

The original 802.11 standard was completed in 1997 [43]; however, recent enhancements to the original standard have resulted in multiple versions of 802.11; most notably 802.11a, 802.11b, and 802.11g. Table 8 shows some basic information on each of these standards. This thesis focuses primarily on 802.11g as it is the newest standard that possesses significant advantages over 802.11b and 802.11a.

Table 8: 802.11 Standards [43]

	802.11b	**802.11a**	**802.11g**
Date of Approval	July 1999	July 1999	June 2003
Max. Data Rate	11 Mbps	54 Mbps	54 Mbps
Modulation	CCK	OFDM	CCK and OFDM
Data Rates	1,2,5.5,11 Mbps	6, 9, 12, 18, 24, 36, 48, 54 Mbps	CCK: 1, 2, 5.5, 11 OFDM: 6, 9, 12, 18, 24, 36, 48, 54 Mbps
Frequency Range	2.4–2.497 GHz	5.15–5.35 GHz 5.425–5.675 GHz 5.725–5.875 GHz	2.4–2.497 GHz

802.11n, which uses multiple input and output antennas (MIMO), has shown major improvements over previous versions of 802.11 in regards to data rate and range; however, it is estimated that the technology will not be widely available until 2009 [44]. The next section will provide a brief background on 802.11 technology to help explain why certain channel properties behave as they do. The following section contains the protocol used to test the various channel properties outlined in Section 2.2. The chapter concludes with testing results and analysis.

3.1 Background

Each of the 802.11 specifications supports multiple data rates to allow the highest possible communication speed while trying to minimize the number of communication errors. Each 802.11 standard defines both a Physical Layer (PHY) and Medium Access Control (MAC) implementation. The PHY implementation, which is the bottom layer of the OSI model from Figure 18, provides various coding schemes to support multiple data rates for communication. The MAC layer, which is a sublayer of the Data Link Layer (Layer 2) of the OSI model, controls access to the wireless channel by dictating which node is permitted to transmit at what time [17]. Figure 54 displays the

802.11 protocol stack, which resides in the bottom two layers of the OSI model. The 802.11 standard as a whole allows for seamless connection to an IP network.

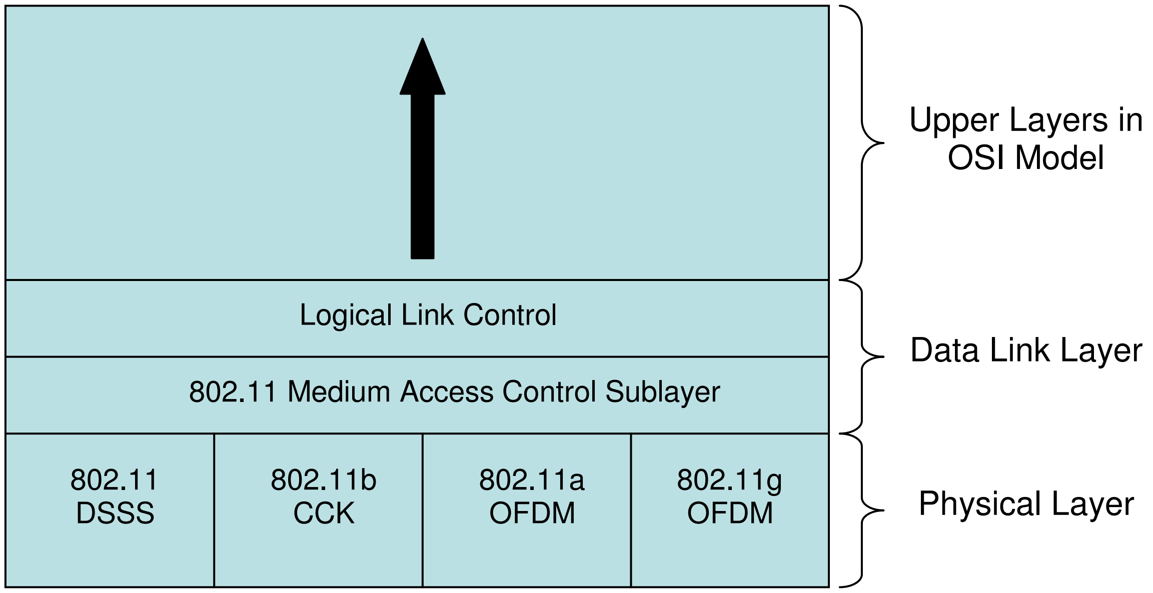


Figure 54: 802.11 Protocol Stack [17]

The three most common WLAN standards, 802.11a, 802.11b and 802.11g, all share a common MAC layer, but all have different PHY implementations. The PHY implementations use different modulation techniques to code data, which allows for a number of different transmission rates. The original 802.11 PHY specification, completed in 1997, supports only two data rates; 1 or 2 Mbps. The modulation method used for these two data rates is called Direct Sequence Spread Spectrum (DSSS), and data is transmitted at 1 Mbaud (million symbols per second). To achieve data rates of 1 or 2 Mbps, data is coded using either Binary Phase Shift Keying (BPSK) which encodes one bit per symbol, or Quadrature Phase Shift Keying (QPSK) which encodes two bits per symbol [17].

In 1999, 802.11b expanded upon the original specification by adding two additional data rates of 5.5 and 11 Mbps. These data rates are achieved by using a modulation technique called Complementary Code Keying (CCK). CCK is a variation of M-ary Orthogonal Keying Modulation that uses complex symbol structures called Walsh/Hadamard codes. The symbols are coded using QPSK, and they are transmitted at 1.375 Mbaud. Using the Walsh/Hadamard codes, data is encoded at 4 or 8 bits per baud to achieve data rates of 5.5 and 11 Mbps respectively [45].

The last modulation scheme used by the 802.11 standards is Orthogonal Frequency Division Multiplexing (OFDM). OFDM, which is utilized by both 802.11a and 802.11g, uses multiple closely spaced orthogonal sub-carriers to carry data. The sub-carriers typically overlap in frequency, but can be efficiently separated using a Fast Fourier Transform (FFT) algorithm. Each sub-carrier uses a conventional modulation scheme (BPSK, QPSK etc.) to code data [44]. Table 9 shows the different modulation schemes used by the 802.11 standards, and the corresponding data rate for each method. The data rate is computed by multiplying the number of symbols sent per second with the number of bits encoded in each symbol. For OFDM, there is also a coding rate involved in the computation which specifies the rate at which symbols are coded.

Table 9: 802.11 Modulation Schemes and Data Rates [40] [17] [45]

802.11 Specification	**Modulation Scheme**	**MBaud**	**Coding Technique**	**Bits/Symbol**	**Coding Rate**	**Data Rate (Mbps)**
Original 802.11	DSSS	1	BPSK	1	-	1
		1	QPSK	2	-	2
802.11b	CCK	1.375	QPSK + Walsh/Hadamard Codes	4	-	5.5
		1.375	QPSK + Walsh/Hadamard Codes	8	-	11
802.11g 802.11a	OFDM	12	BPSK	1	1/2	6
		12	BPSK	1	3/4	9
		12	QPSK	2	1/2	12
		12	QPSK	2	3/4	18
		12	16QAM	4	9/16	27
		12	16QAM	4	3/4	36
		12	64QAM	6	3/4	54

Each of the coding techniques used by 802.11 differs in the number of elements in their signal set. For example, BPSK has two elements in its signal set which allows it to encode only one bit per symbol ($2^1 = 2$). On the other hand, 64QAM, which has 64 elements in its signal set, is able to encode 6 bits per symbol ($2^6 = 64$). QPSK has 4 elements in its signal set while 16QAM has 16, corresponding to 2 and 4 bits per symbol respectively. As the number of bits per symbol is increased, the higher the signal to noise ratio (SNR) must be at the receiver to maintain a constant error rate. This is because the more elements in the signal set, the greater chance the receiver will make an error demodulating the signal if there is added noise [40]. Figure 55 shows the symbol error rates for the coding techniques used by the 802.11 standards. The vertical axis shows the symbol error rate (10^x), while the horizontal axis shows the signal-to-noise ratio necessary to achieve this error rate. The curves shown in Figure 55 are theoretical, and assume a steady-signal channel in the presence of Additive White Gaussian Noise (AWGN).

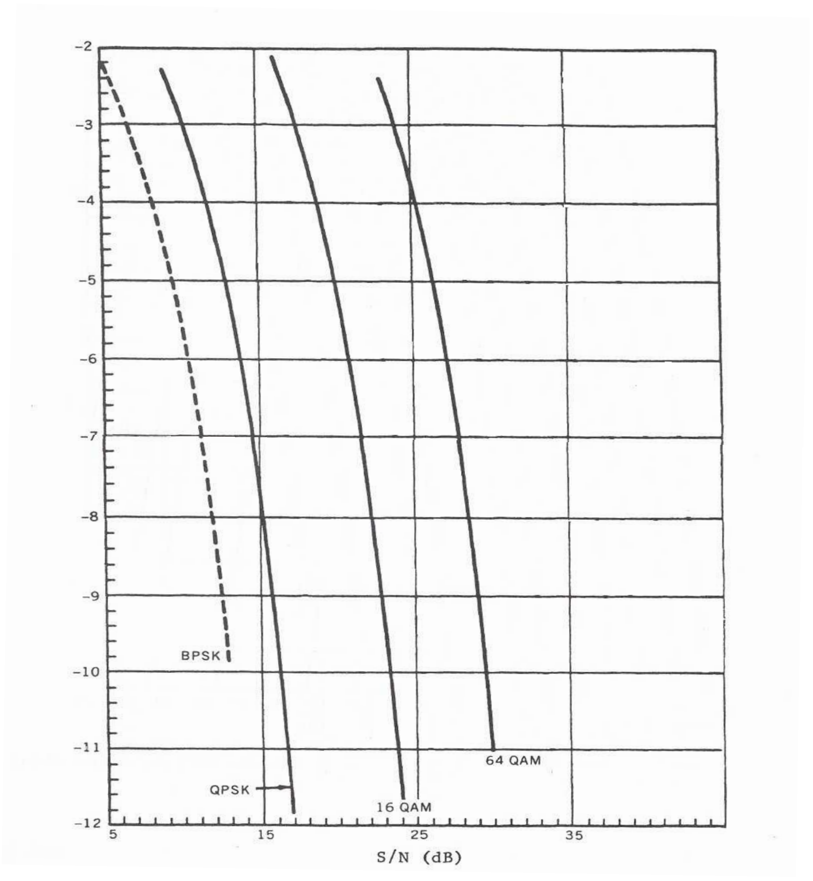


Figure 55: Bit Error Rates for Selected PSK and QAM Modulation Methods [40]

Because the higher 802.11 data rates require a higher SNR to maintain a constant error rate, the 802.11 standards use algorithms that dynamically change the transmission rate based on the channel conditions [45]. As a result, 802.11 data rates become a function of the signal strength at the receiver, which can be influenced by many factors including distance, channel interference, and physical obstructions. Figure 56 shows typical 802.11 data rates as a function of distance, where the channel is assumed to have no interference and a clear line of sight between the sender and receiver.

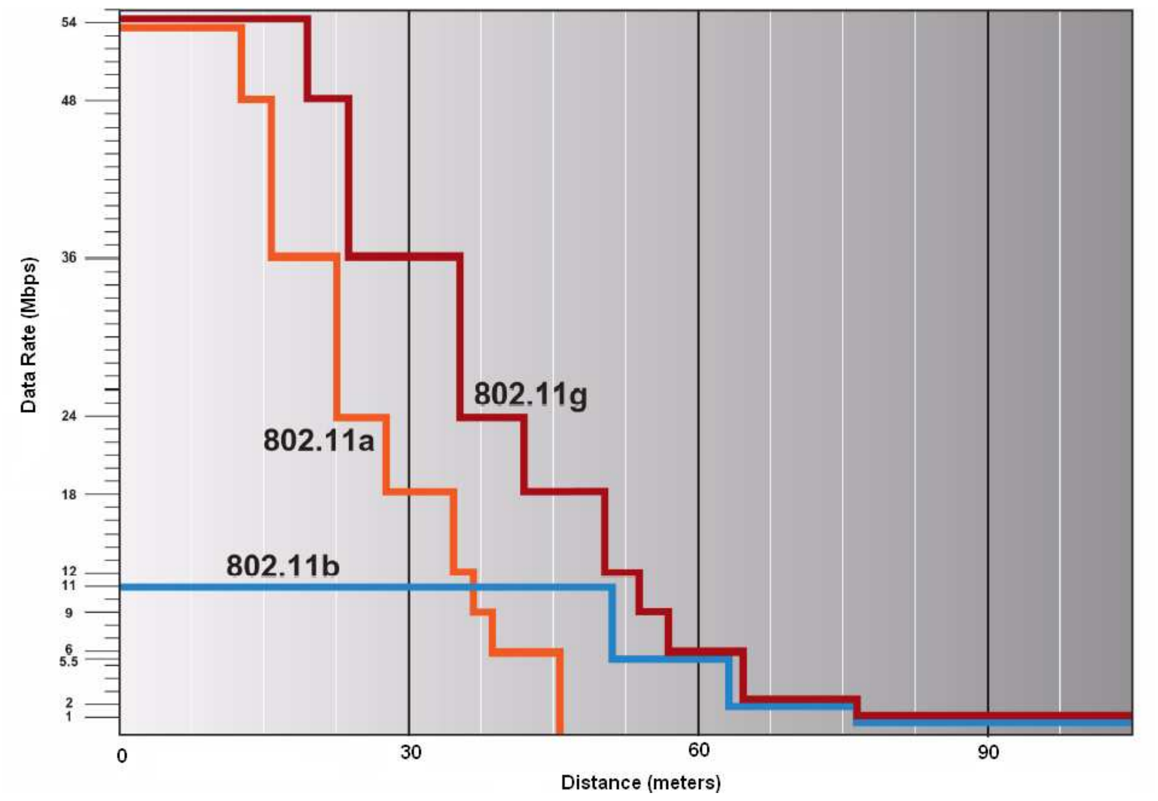


Figure 56: Approximate 802.11 Data Rates as a Function of Distance [43]

It is evident from Figure 56 that 802.11a does not have the same range as 802.11b and g. This is because 802.11b and 802.11g operate in the unlicensed 2.4 GHz spectrum while 802.11a operates in 5 GHz spectrum. Since 5 GHz radio waves do not propagate as well as 2.4 GHz, the range of 802.11a is shorter than that of 802.11b and 802.11g [43]. Also, the reason 802.11g operates at the lower 802.11b data rates at distances greater than 50 m is because most 802.11g devices are made for backwards compatibility with 802.11b. Consequently, most 802.11g devices implement original 802.11, 802.11b as well as the additional high-speed OFDM data rates [43]. Lastly, it should be noted that 802.11 channels are only half duplex, meaning that nodes cannot transmit and receive at the same time on a single frequency [17]. Because the 802.11 MAC protocol uses acknowledgment frames (ACK) to verify the successful reception of data frames, achievable data rates are a little less than half of the values seen in Figure 56 [45]. The 802.11 MAC implementation will be explained later in this section.

Figure 57 shows the approximate physical ranges of the 802.11 standards. 802.11 b and g can operate at distances upwards of 100 meters at the lowest data rates while 802.11a only works up to ranges of around 50 meters. Of course, surrounding obstructions (walls, geologic structures etc.), transmission power and external antennas can all affect the range of 802.11. The values given in Figure 57 assume an outdoor environment with a single wall contributing to the loss.

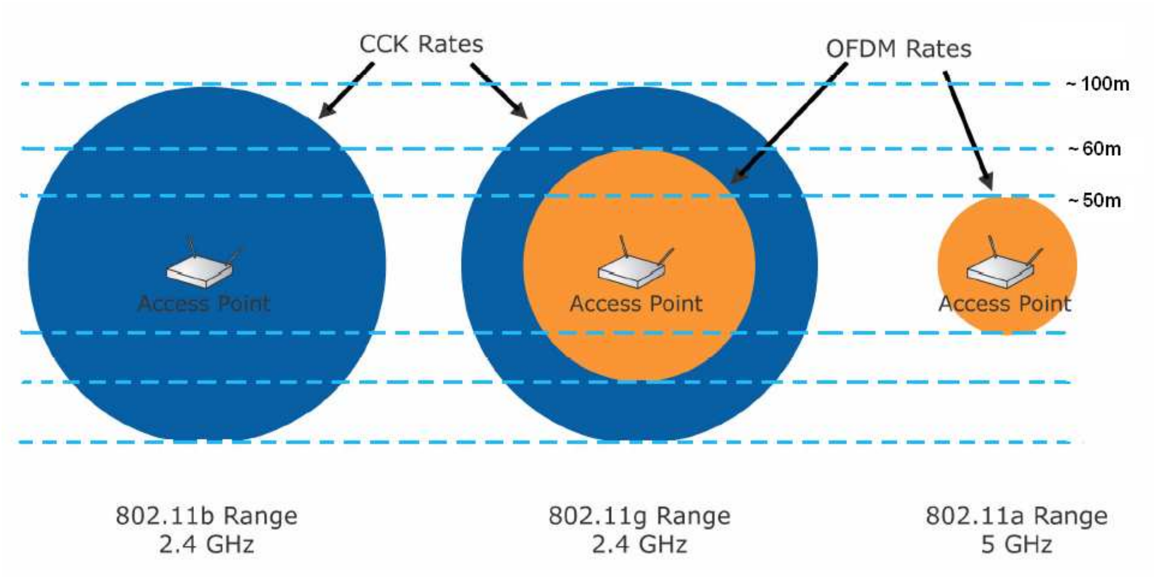


Figure 57: 802.11 Physical Range (Orange Represents OFDM Data Rates; Blue Represents CCK Data Rates) [43]

In addition to the PHY implementation, the 802.11 standards define a MAC protocol, which controls access to the wireless medium. The MAC protocol, which is common to all of the 802.11 versions, uses what is called a *Distributed Coordination Function* (DCF) which is based on the *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) protocol [46]. The 802.11 DCF mechanism contributes to the overall delay of the channel and can limit the throughput between two network endpoints. The 802.11 DCF operates as follows.

Before a wireless station can initiate a frame transmission, it senses the wireless medium to determine if it is busy. If it determines that the medium is idle for longer than a period called the *Distributed Inter-Frame Space* (DIFS), then it can transmit its frame immediately. If it is determined that the medium is busy, then it must wait until the medium becomes idle, and then defer for an additional DIFS interval. If the medium remains idle, then the MAC begins a backoff procedure by selecting a random backoff count. The random backoff procedure is described in detail below. While the medium stays idle, the backoff counter is decremented by one for each transmission slot time. After the counter reaches zero, the station can transmit the frame. For each successful frame reception, the receiver sends an acknowledgment frame (ACK) after a *Short Inter-Frame Space* (SIFS) interval to verify reception. If an ACK is not received by the sender within an ACK timeout period, another random backoff procedure is executed, and the frame is then retransmitted [45]. Figure 58 displays this basic access method of the 802.11 DCF mechanism.

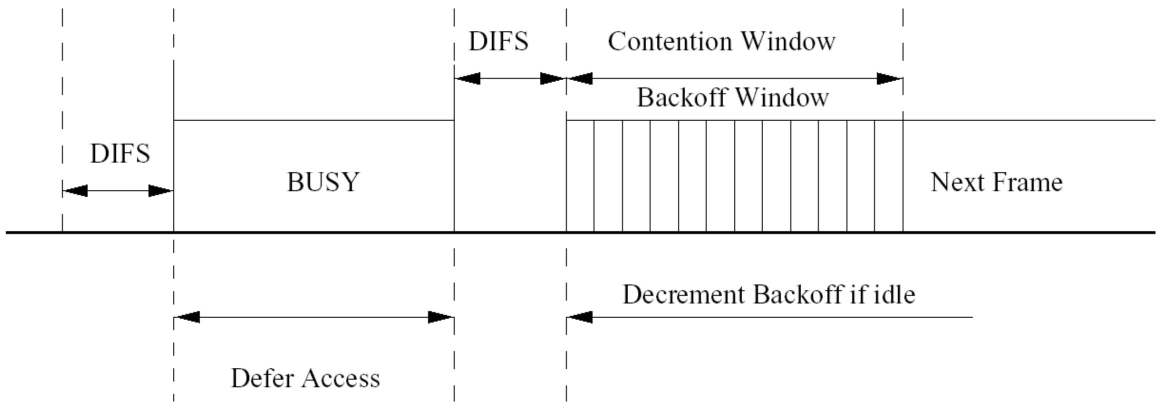


Figure 58: 802.11 DCF Basic Access Method [46]

The random backoff count is selected as a random integer drawn from a uniform distribution over the interval [0,CW], where CW represents a *contention window* expressed in units of transmission slot times. Initially, the size of CW is set to CW_{MIN}, and it is doubled each time a transmission attempt fails. CW will continue to increase for each unsuccessful transmission until it reaches the value of CW_{MAX}. CW will remain at CW_{MAX} until it is reset. CW is reset to CW_{MIN} after a successful transmission or after reaching the maximum retry limit. After the maximum retry limit is reached, the frame is ultimately discarded [45]. Figure 59 shows the behavior of CW.

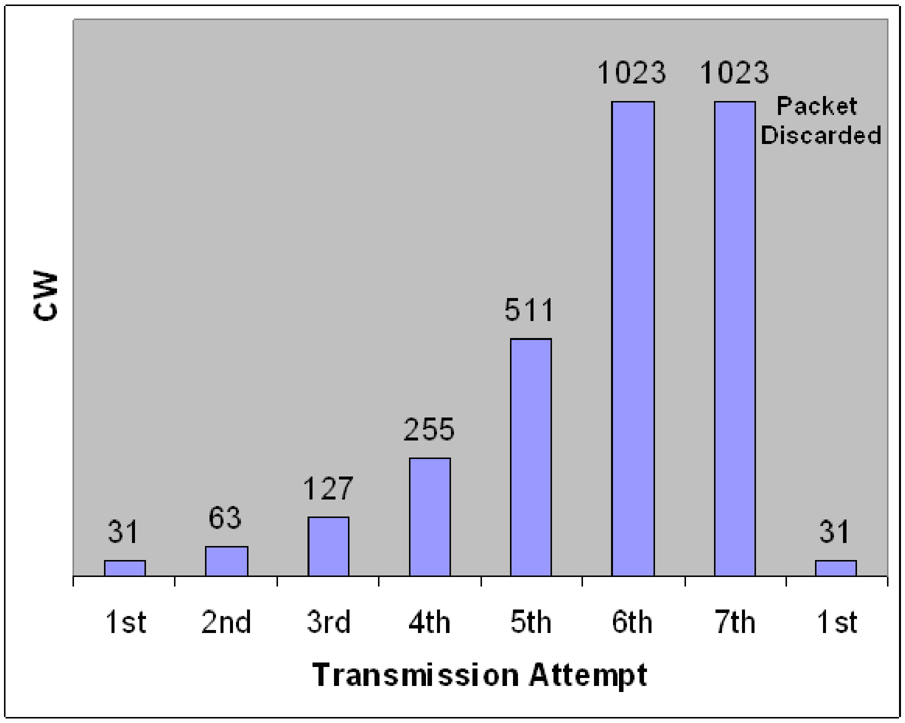


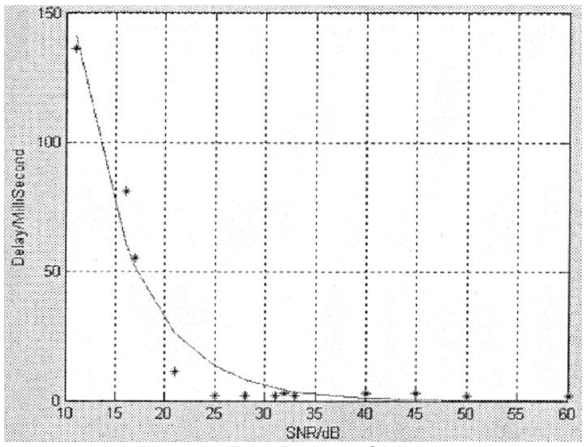
Figure 59: Backoff Windows for IEEE 802.11 [46]

Because 802.11b and g are interoperable, they use the same DCF parameters. Table 10 shows the typical MAC parameters used in most 802.11b and g device implementations.

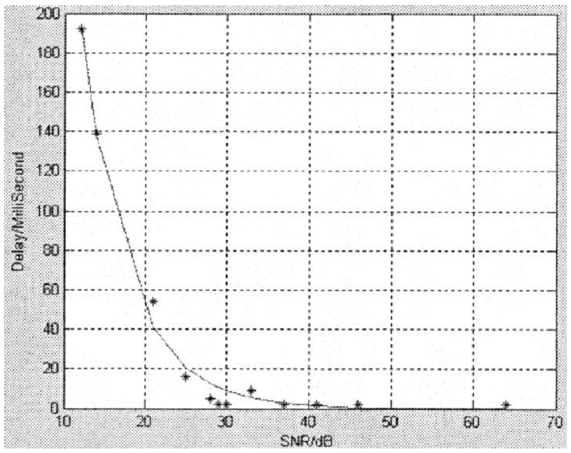
Table 10: 802.11b/g MAC Parameters [45]

Parameter	**Value**
Slot Time	20 μsec
DIFS	50 μsec
SIFS	10 μsec
CW_{MIN}	31
CW_{MAX}	1023
Max Retry Limit	6

It can be seen from Figure 59 how the latency of an 802.11 channel can be negatively affected by the MAC implementation. In networks that have many nodes competing to transmit frames or that have high error rates, the overall delay increases exponentially as channel conditions become worse. Figure 60 shows empirical delay results made by Wang and Refai [47]. These results were obtained on an 802.11g network isolated from all other 802.11 networks and free of any channel interference. The tests were run with only two active nodes on the network and round trip delay times were measured. Two tests were conducted, each using a different manufacturer's chipset. The vertical axis represents the round trip delay in milliseconds and the horizontal axis corresponds to the signal-to-noise ratio in dB.



(a) Round Trip Delay (Chipset A)



(b) Round Trip Delay (Chipset B)

Figure 60: Empirical 802.11g Results [47]

3.2 Testing Protocol

As discussed in the previous section, the performance characteristics of the 802.11 standards depend heavily on the signal strength at the receiving station. For this reason, network performance tests were conducted at various received signal strengths to observe changes in performance relative to signal quality. Although most 802.11 compatible devices operate in a similar fashion, the exact performance metrics will vary from chipset to chipset. The chipset that was tested in this project was the *Realtek RTL8187,* which is utilized in an adapter manufactured by a company called *Alfa Network*. An image of the 802.11 network adapter can be seen in Figure 61.

Figure 61: Alfa USB 802.11b/g Network Adapter with RP-SMA Antenna Jack [52]

The *Alfa USB 802.11b/g Network Adapter* was chosen for a couple of reasons. First, the adapter has a USB interface which makes integration into the current ultrasound design easy as there are USB ports available for external devices. Second, this adapter has an accessible RP-SMA jack, which can be used to connect an external antenna for increased transmission range. A couple of other commercially available adapters with both of these features were examined; however, the adapter by Alfa Network had the best performance in preliminary tests.

To gather 802.11 performance metrics, two laptops were connected through an 802.11g ad-hoc wireless network. Both laptops used the Alfa USB 802.11b/g Network Adapter, and there were no other nodes on the network other than the two laptops. All of the testing was conducted in an outdoor environment in the confines of Institute Park located in Worcester, MA. Institute Park is a relatively flat and open area with a few scattered trees distributed throughout the park. The reason Institute Park was chosen as the testing location was to ensure that there was always a clear Line Of Sight (LOS) between the client and server systems during all tests. Each of the laptops was elevated approximately two feet off the surface of the ground during testing. Also, there was not much activity in the 802.11g frequency range (2.4 GHz) in the park. By isolating the ad-hoc test network from all other 802.11 networks, performance degradation due to inter-band interference was avoided. Lastly, signal degradation due to walls and other objects as well as excessive multipath effects inherent in indoor environments were removed by testing outdoors.

To test the performance of the 802.11 adapter, one of the laptops ran the server side code of the channel measurement application while the other ran the client application. The server system was placed in a fixed location where it remained throughout all of the tests. The client system was moved to various locations throughout the park, each a different distance from the server system. Prior to running a set of performance tests, the signal strength at both the server and client systems was monitored and recorded. Care was taken to ensure that both the client and server systems had approximately the same received signal strength before running a test. This would allow the assumption that the channel was symmetrical at that point in time because each of the two adapters should perform the same given the same signal quality. A symmetrical channel should have the same channel properties, such as throughput and latency, on both the uplink and downlink. After the signal strength at both nodes was verified, a set of performance tests would was run using the channel measurement application described in Section 2.4. In total 20 different points were tested. Table 11 shows the parameters that were used throughout the 802.11 tests.

Table 11: Testing Parameters for 802.11g Measurements

Test	**Test Length**	**Packet Size(s) (Bytes)**
Delay	• 1000 Round Trip Packets	• 1500
Delay vs. Packet Size	• 50 Round Trip Packets at each interval • 29 total intervals	• 100 to 1500 in intervals of 50 bytes
Throughput	• 60 seconds in forward direction	• 1400
Throughput vs. Packet Size	• 5000 packets at each interval • 7 intervals	• 200 to 1400 in intervals of 200 bytes

To track the received signal strength, a free program called *NetStumbler* was used. NetStumbler measures and records the SNR of any 802.11 signals detected by the wireless adapter. The results from NetStumbler could be saved to a text file for each individual data point. Before a performance test was run, data from NetStumbler was observed for approximately one minute to ensure there were no large variations in SNR at the receiver. Figure 62 shows the NetStumbler GUI tracking the signal of another node on the network.

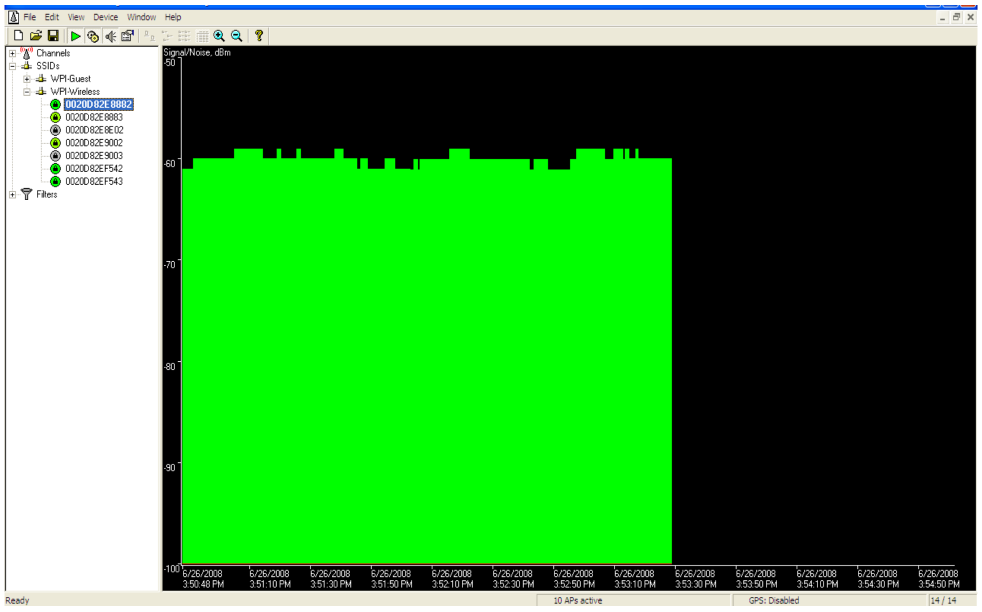
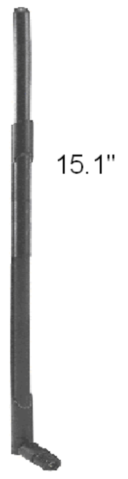
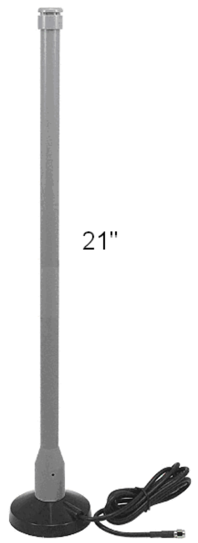


Figure 62: NetStumbler GUI Tracking 802.11 SNR

In addition to the performance tests, the performance of external antennas was examined. Figure 63 shows the two external 802.11 antennas that were tested as part of this project. Both antennas are made by a company called *HyperGain* and both have an RP-SMA interface that is compatible with the jack on the 802.11 adapter. The idea behind testing the antennas was to determine how far the transmission range could be extended by simply adding an external antenna to the client. To test the antenna performance, the server system, which is meant to replicate the base station or gateway, was equipped with the HG2408RD "Rubber Duck" Antenna shown in Figure 63(a). This would not be an unreasonable requirement for a base station as they are generally located in a stationary location or on a fixed piece of hardware such as a helicopter or ambulance. Tests were run with the client system equipped with either the HG2408RD "Rubber Duck" Antenna, shown in Figure 63(a), the RE09U "Range Doubler" Antenna shown in Figure 63(b), or no external antenna at all. The client was initially place 10 meters from the server, and SNR measurements were taken at the server using NetStumbler. The client was then moved to a distance of 20 meters, and the measurements were repeated. The range between the client and server was increased in increments of 10 meters all the way to a distance of 200 meters. The SNR measurements taken at the server could then be matched up with the performance tests to determine what type of performance could be expected at a given distance using a particular antenna.



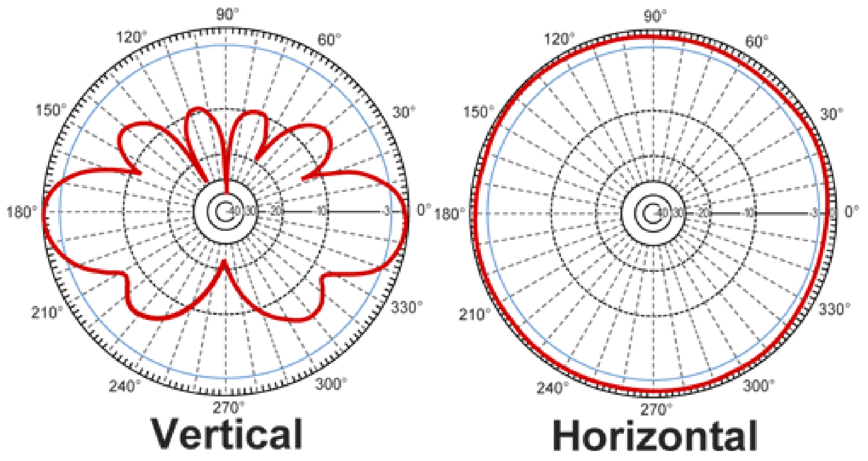
(a) HyperGain HG2408RD "Rubber Duck" Antenna



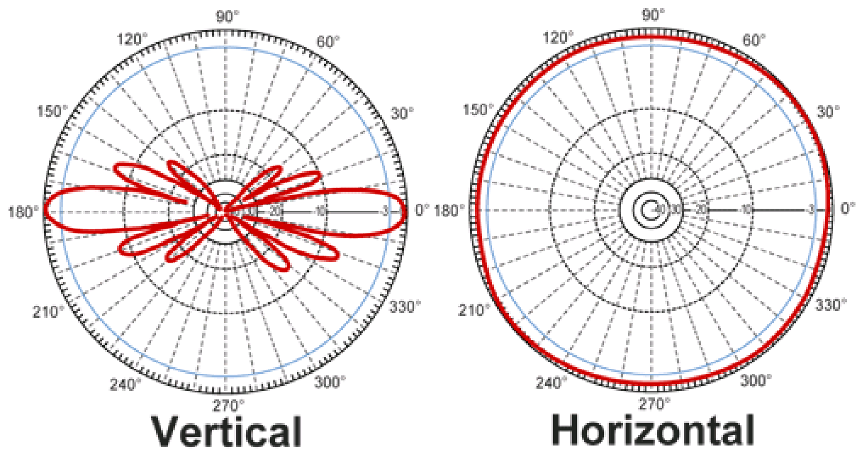
(b) HyperGain RE09U "Range Doubler" Antenna

Figure 63: External 802.11g Antennas with RP-SMA Interface [51]

The HG2408RD "Rubber Duck" Antenna has a gain factor of 9 dBi and is 15.1" in length. It has a tilt and swivel RP-SMA interface at its base that can be connected directly to a female RP-SMA jack. The REO9U "Range Doubler" uses a 4 foot coax cable to interface with the RP-SMA jack. This antenna is 21" tall and has a gain of 8.5 dBi. The REO9U has a magnetic base for easy mounting on a vehicle or other surface, and it advertises twice the range of "rubber duck" antennas. Both of the antennas from Figure 63 are omnidirectional in the horizontal domain; however their gain patterns in the vertical plane differ. Figure 64 shows the gain patterns of each antenna.



(a) HG2408RD "Rubber Duck" Gain Pattern



(b) RE09U "Range Doubler" Gain Pattern

Figure 64: 802.11g Antenna Gain Patterns [51]

3.3 Testing Results and Analysis

Because 802.11 devices uses link adaptation algorithms to adjust coding techniques based on signal quality, the performance of the protocol depends heavily on the signal strength at the receiving node. This behavior was confirmed after viewing the empirical data gathered during the 802.11 performance tests. This section will discuss the relevant findings made during 802.11 testing. The results from each individual data point can be seen in Appendix A.

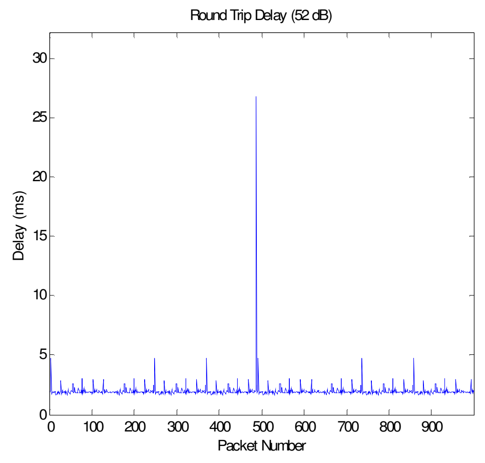
3.3.1 Latency

The latency on a single-hop, 802.11g channel is almost negligible when compared to the latencies exhibited on 3G and satellite networks. As described in Section 3.1, the main source of latency on 802.11 networks is due to the MAC protocol. As the signal-to-noise ratio is decreased and additional retransmissions are required from the MAC protocol, the contention window is increased exponentially causing the delays to increase exponentially as well. For this reason, a lot of the data that was recorded during testing can be fit to an exponential decay function as shown in (3.1). To fit this function to the measured delay data, the *EzyFit Toolbox* utility was used along with MATLAB to find the best fit as well as plot the curve. The EzyFit utility uses the FMINSEARCH function in MATLAB which finds the best fit by performing an unconstrained nonlinear minimization of the sum of squared residuals with respect to the various parameters [58]. The same technique was used to fit curves to jitter and packet loss data as well.

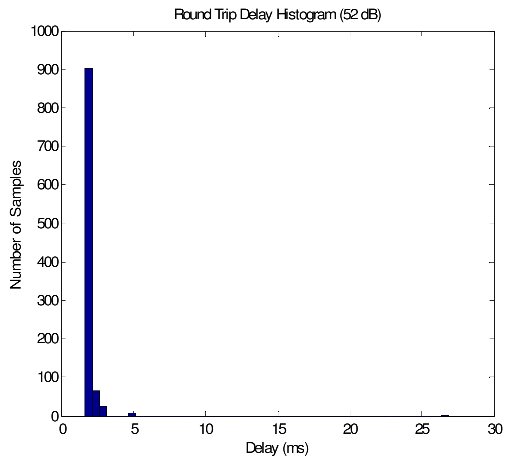
$$F(SNR) = A \bullet \exp(-B \bullet SNR) + C \tag{3.1}$$

Figure 65 shows the results from a delay test run over the 802.11g ad-hoc network with a signal-to-noise ratio of approximately 53 dB at both the sending and receiving nodes. Similar to the 3G tests, the round trip latency is given as a function of time (Figure 65(a)), from which a histogram could be plotted to show the distribution of the delays (Figure 65(b)). In addition to the delay data, one-way jitter and packet loss were recorded as well. Figure 65(c) shows the forward jitter as a function of time measured during the test. Because it was ensured that the signal-to-noise was approximately equal at both the sender and receiver prior to each test, both the delay and jitter characteristics can be assumed to be equal in each direction. For this reason, only the forward jitter was plotted (Figure 65(c)); however if there was more than a 25% difference in average jitter in the reverse direction when compared to the forward direction, the results were thrown out. The PDF and CDF of the jitter can be seen in Figure 65(d) and

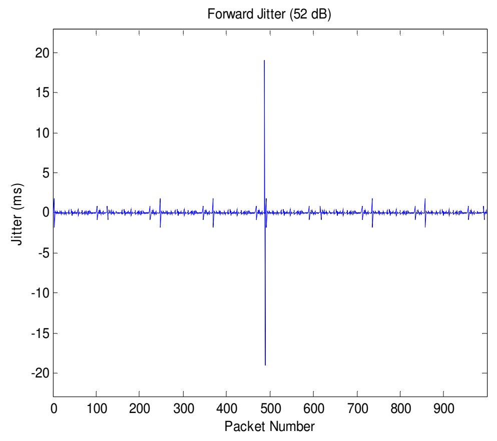
Figure 65(e) respectively. The results from each of the twenty tested data points can be seen in Appendix A.



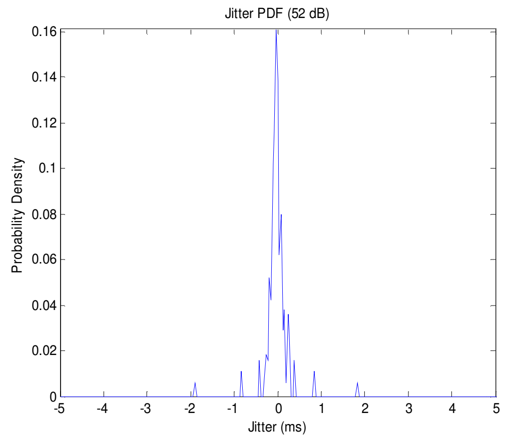
(a) Round Trip Delay vs. Time



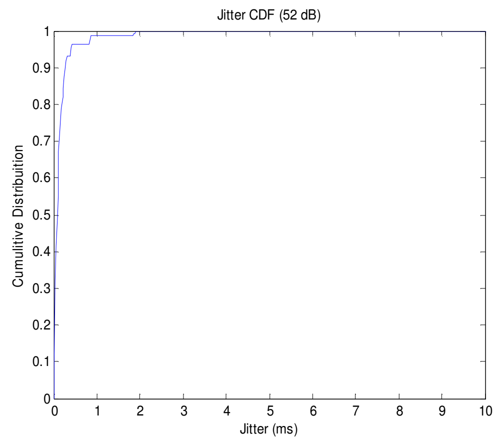
(b) Round Trip Delay Histogram



(c) Jitter vs. Time



(d) Jitter PDF



(e) Jitter CDF

Max Delay = 26.8 ms
Min Delay = 1.6 ms
Mean Delay = 2.0 ms
Round Trip Standard Deviation = ms
Packet Loss (Forward) = 0/1000
Packet Loss (Reverse) = 0/1000
Mean Forward Jitter = 0.2 ms
Forward Jitter (95%) = 0.4 ms
Forward Jitter (99%) = 1.9 ms

Figure 65: Results from an 802.11g Delay Test with a SNR of 52 dB

Figure 66 shows how the average round trip delay on an 802.11g ad-hoc network behaves as a function of signal-to-noise ratio. For example, the data point at 52 dB corresponds to the mean of the data shown in Figure 65(a). An exponential decay function was fit to the data which can also be seen in Figure 66. The equation for this curve is given in (3.2).

$$RTD(SNR) = 6212.2 \bullet \exp(-0.333 \bullet SNR) + 4.395 \text{ [ms]} \tag{3.2}$$

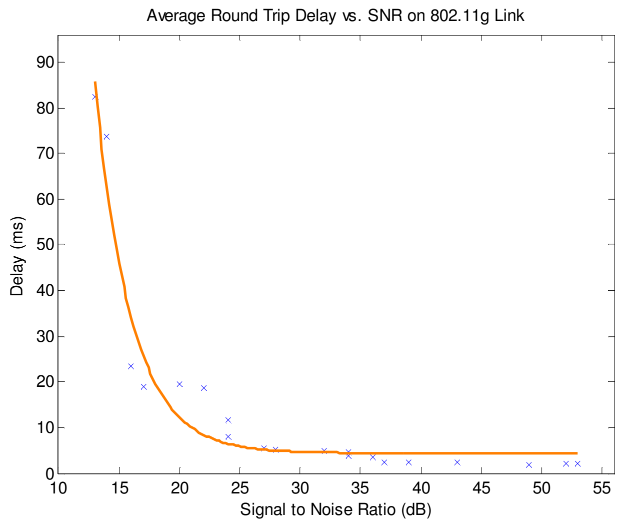


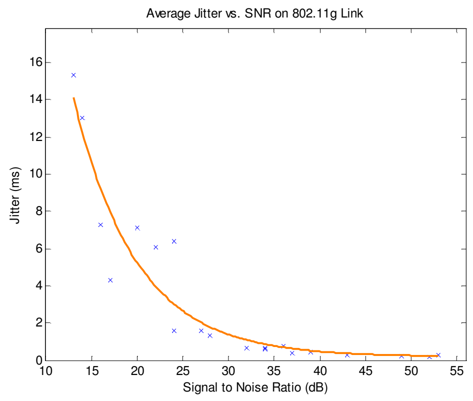
Figure 66: Average Round Trip Delay vs. SNR on an Ad-hoc 802.11g Network

The jitter characteristics of the 802.11g channel can be seen in Figure 67. This figure shows the mean jitter (Figure 67(a)), 95% jitter threshold (Figure 67(b)) and 99% jitter threshold (Figure 67(c)) as a function of signal-to-noise ratio. As with the 3G tests, the 95% and 99% jitter thresholds correspond to the points where at least 95% and 99% of the jitter magnitudes are between zero and the threshold value. All three of these plots also appear to decay exponentially and were fit to curves. The equations for each of the lines of fit can be seen in (3.3), (3.4) and (3.5).

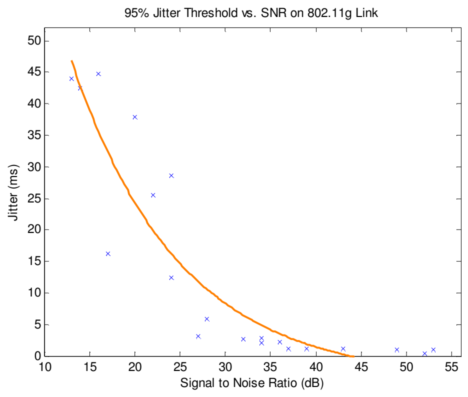
$$MJ(SNR) = 91.3 \bullet \exp(-0.145 \bullet SNR) + 0.194 \text{ [ms]} \tag{3.3}$$

$$95JT(SNR) = 151.2 \bullet \exp(-0.084 \bullet SNR) - 3.842 \text{ [ms]} \tag{3.4}$$

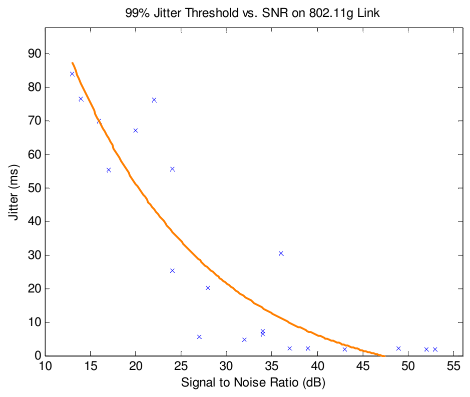
$$99JT(SNR) = 230.9 \bullet \exp(-0.066 \bullet SNR) - 10.390 \text{ [ms]} \tag{3.5}$$



(a) Mean Jitter



(b) 95% Jitter Threshold



(c) 99% Jitter Threshold

Figure 67: Jitter Behavior vs. SNR on an Ad-hoc 802.1g Network

The last metric that was measured during the delay tests was packet loss. Figure 68 shows the average packet loss as a function of signal-to-noise ratio when sending a single packet at a time. Like the jitter and latency, packet loss also decreases exponentially with SNR. Using MATLAB, an exponential decay function was fit to the data. The equation for this curve can be seen in (3.6).

$$PL(SNR) = 143.9 \bullet \exp(-0.289 \bullet SNR) + 0.038 \text{ [\%]} \tag{3.6}$$

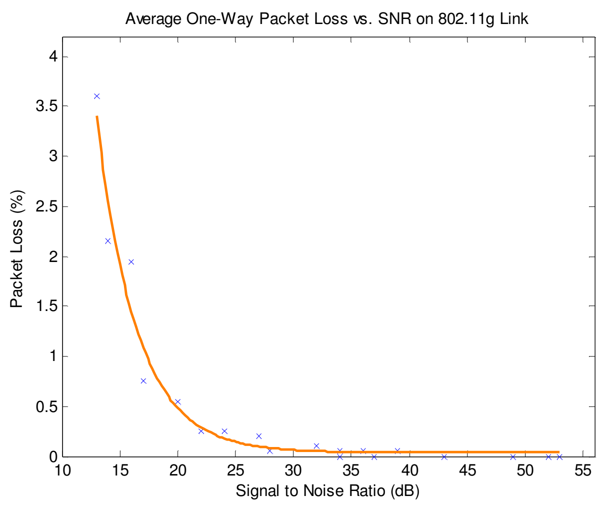
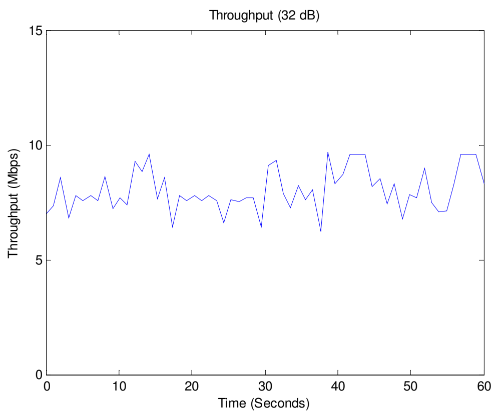


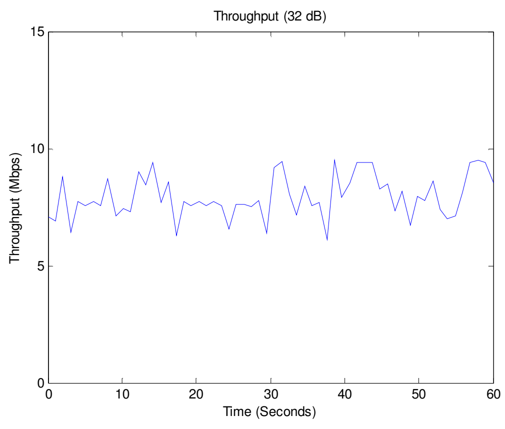
Figure 68: Average One-Way Packet Loss vs. SNR on an Ad-hoc 802.11g Network

3.3.2 Throughput

Twenty throughput tests were run at varying signal strengths to observe how signal quality affects data speeds on an 802.11g ad-hoc network. Like the latency tests, the results show that 802.11g throughput is greatly affected by the signal-to-noise ratio as different modulation schemes are used to adjust the data based on signal quality. Figure 69 shows the throughput results obtained during a test conducted with a signal strength of 32 dB. Figure 69(a) shows the throughput as seen by the client (sender) while Figure 69(b) shows the receiving throughput. Throughout most of the tests, the client and server reports closely resemble one another. Lastly, the percentage of dropped packets is monitored and recorded during these tests.



(a) Client Report

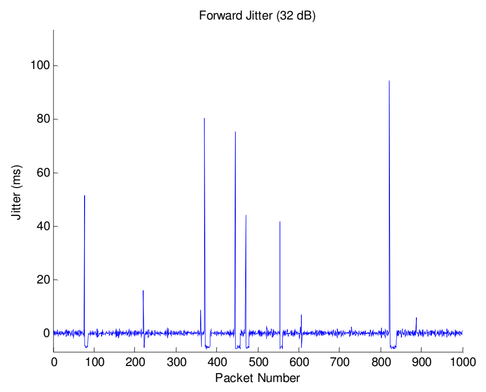


(b) Server Report

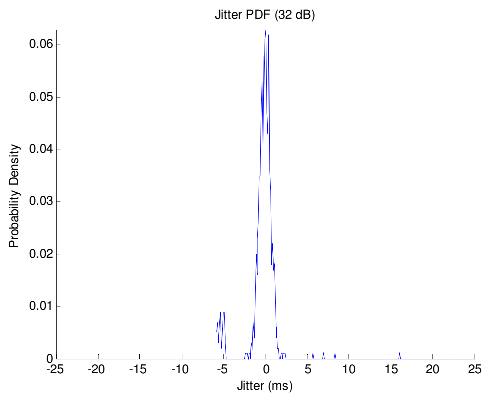
Max BW = 9.51 Mbps
Min BW = 6.13 Mbps
Mean BW = 7.95 Mbps
Drop Percentage = 0.3 %

Figure 69: Throughput Results with a Signal-to-Noise Ratio of 32 dB

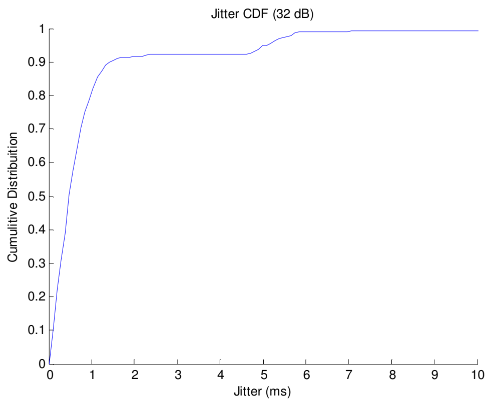
In addition to the throughput calculations shown in Figure 69, jitter calculations were also performed. Figure 70 shows the jitter behavior observed with a corresponding signal-to-noise ratio of 32 dB. During each test, data was sent at approximately 25%, 50%, 75% and 100% of the capacity of the channel at each data point. Figure 70 shows the jitter plotted as a function of time as well as the PDF and CDF of the recorded jitter values. In general, the jitter behavior did not vary wildly when data was sent at different fractions of the channel capacity. As can be seen in the jitter PDFs (Figure 70(b), (e), (h) and (j)), there is a limit on how small (large negative) the jitter can be. This is due to data being sent at a constant rate at the sending node which puts a limit on the minimum jitter values. As the data rate is increased, this limit grows (smaller negative) to compensate for the high data rate. This in turn tends to lead to smaller average jitters as data rates are increased, but it is not always the case.



(a) Jitter vs. Time (25%)

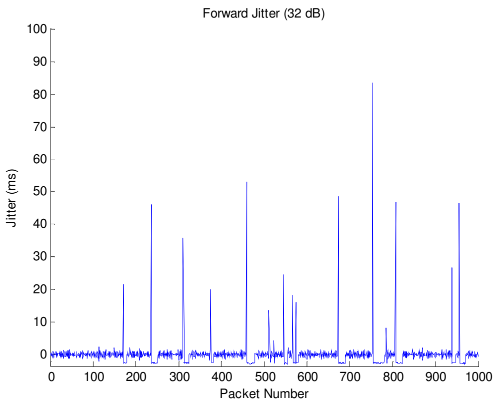


(b) Jitter PDF (25%)

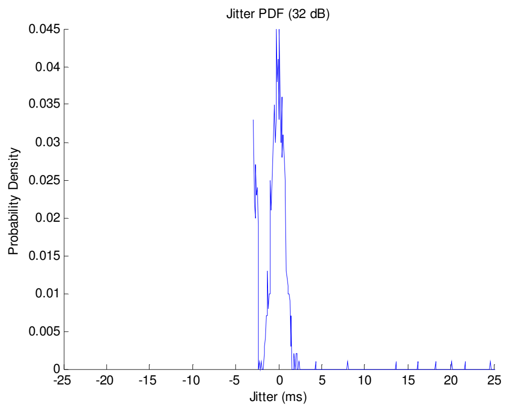


(c) Jitter CDF (25%)

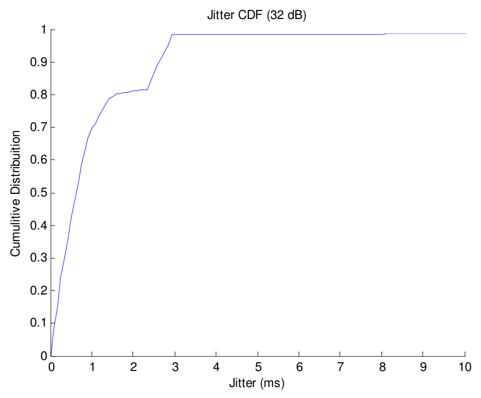
Throughput = 2.38 Mbps
Mean Forward Jitter = 1.25 ms
Forward Jitter (95%) = 5.28 ms
Forward Jitter (99%) = 5.93 ms



(d) Jitter vs. Time (50%)

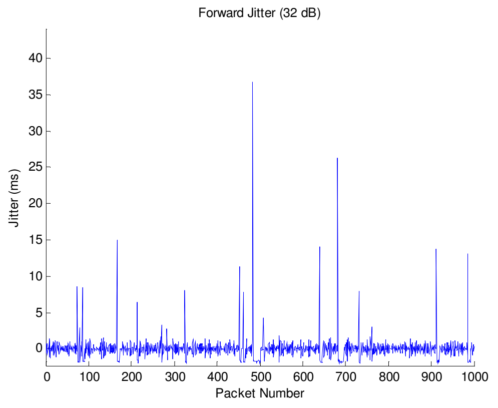


(e)Jitter PDF (50%)

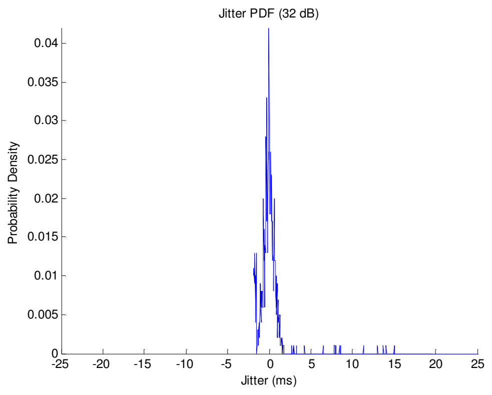


(f) Jitter CDF (50%)

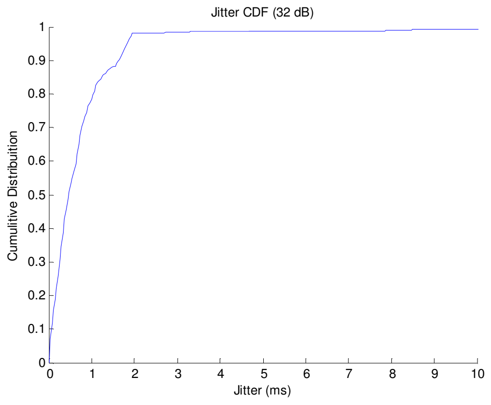
Throughput = 5.12 Mbps
Mean Forward Jitter = 1.42 ms
Forward Jitter (95%) = 2.92 ms
Forward Jitter (99%) = 20.3 ms



(g) Jitter vs. Time (75%)

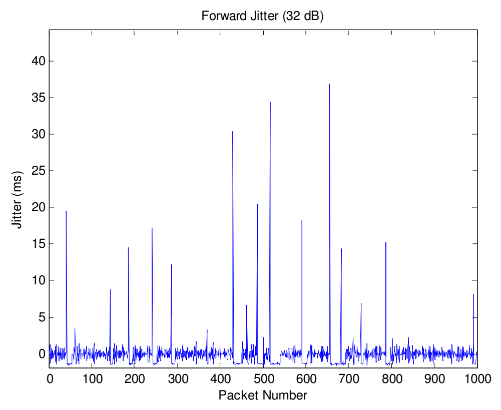


(h)Jitter PDF (75%)

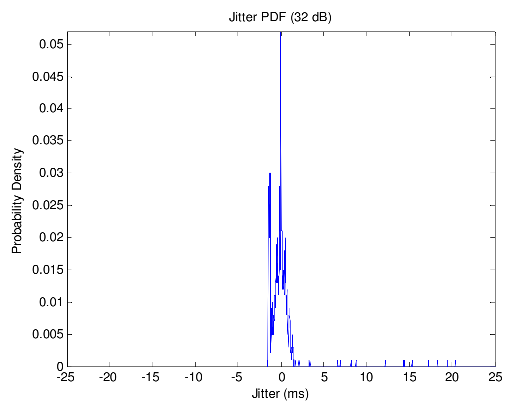


(i) Jitter CDF (75%)

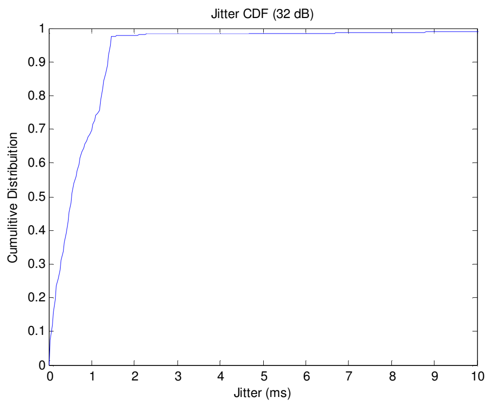
Throughput = 7.34 Mbps
Mean Forward Jitter = 0.79 ms
Forward Jitter (95%) = 1.88 ms
Forward Jitter (99%) = 7.94 ms



(j) Jitter vs. Time (100%)



(k) Jitter PDF (100%)



(l) Jitter CDF (100%)

Throughput = 9.51 Mbps
Mean Forward Jitter = 0.90 ms
Forward Jitter (95%) = 1.51 ms
Forward Jitter (99%) = 12.3 ms

Figure 70: Jitter Results with a Signal-to-Noise Ratio of 17 dB

Figure 71 shows the average 802.11g throughput as a function of signal-to-noise ratio. It is obvious from this that the achievable throughput on an 802.11g link is heavily dependent on SNR. By inspection, it was found that the measured data appeared to have the same shape as the CDF of a normal distribution. For this reason, it was decided to model this data as an altered version of the error function (*erf()*). To find best line of fit, a number of different curves were plotted in the same form as seen in (3.7). The mean square error was then calculated between the curve approximation and the discrete measured data. The curve that was chosen as the best line of fit was the one with the minimum mean square error.

$$BW(SNR) = (A/2) \bullet (erf((SNR - B)/C) + 1) \text{ [Mbps]} \quad (3.7)$$

In this case, A is used to scale the function vertically, while B is used to shift it, and C is used to stretch it horizontally. MATLAB was used to plot the curve seen in Figure 71. To find the best line of fit, a number of curves were plotted and the mean square error was then calculated for the measured data points. After a number of trials, the line that contained the minimum mean squared error was (3.8).

$$BW(SNR) = 21.9 \bullet (1/2) \bullet (erf((SNR - 32.5)/11.5) + 1) \text{ [Mbps]} \qquad (3.8)$$

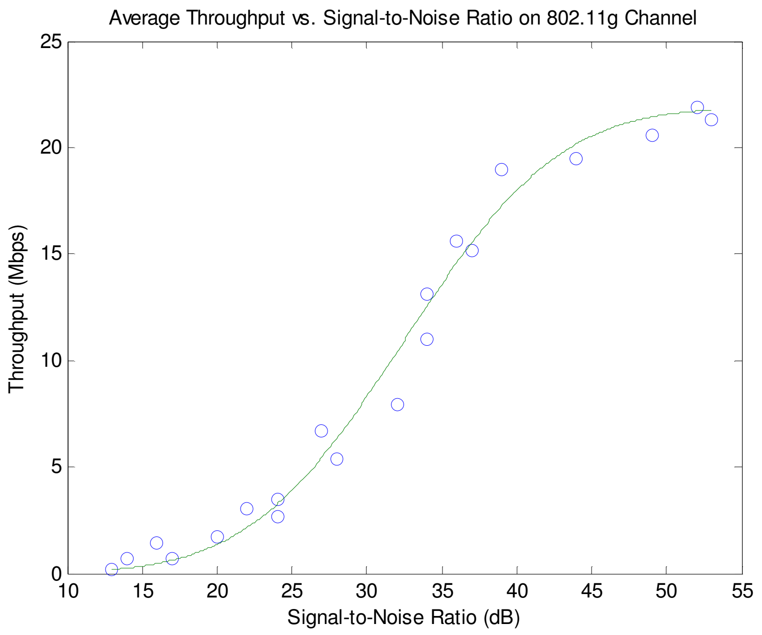


Figure 71: Average UDP Throughput vs. SNR on 802.11g Channel

TCP throughput tests were also run using the Iperf software described in Section 2.5. For each test, data was sent for a period of thirty seconds using TCP and the resultant throughput was calculated. The TCP throughput seen on 802.11 closely resembles the UDP throughput. Figure 72 shows the average TCP throughput as a function of signal-to-noise ratio. The equation for the line of fit also seen in Figure 72 can be found in (3.9).

$$BW(SNR) = 21.3 \bullet (1/2) \bullet (erf((SNR - 32.5)/11.9) + 1) \text{ [Mbps]} \qquad (3.9)$$

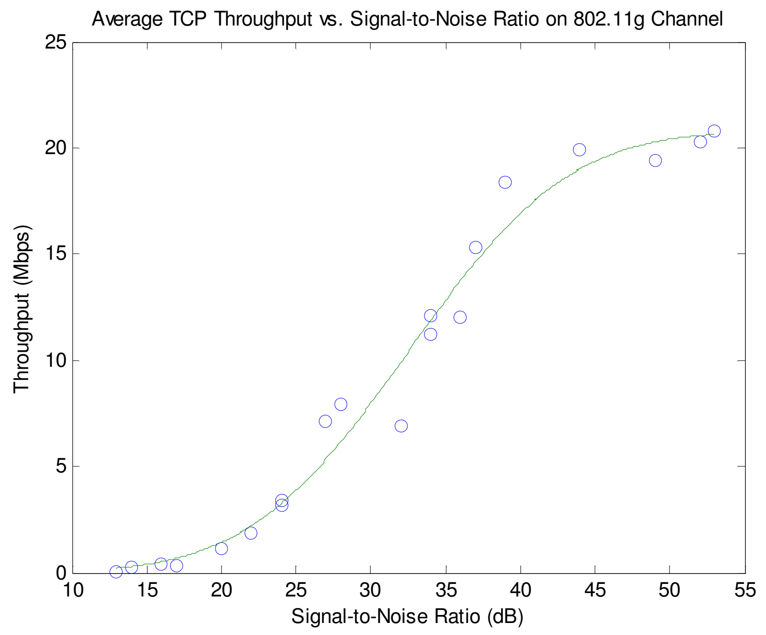


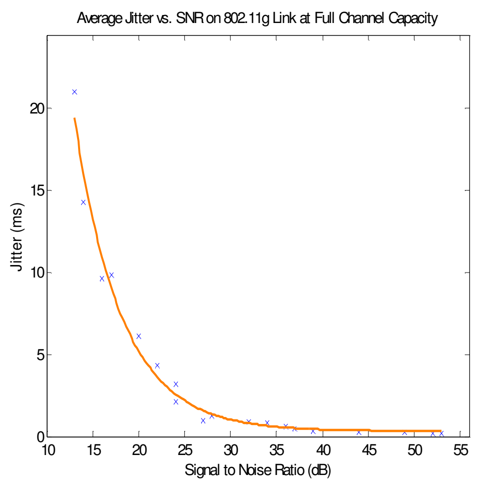
Figure 72: Average TCP Throughput vs. SNR on 802.11g Channel

As mentioned before, jitter was measured during each of the throughput tests at four different data rates. Throughout each of the tests, there were not any glaring differences in the jitter behavior at varying channel capacities. For this reason, only the jitter results at channel capacity are plotted in this section. The complete jitter results measured at 25%, 50%, and 75% channel capacity along with the results from each throughput test can be viewed in Appendix A. Figure 73 shows the jitter behavior vs. SNR at full channel utilization. The equations for the exponential decay curve fits can be seen in (3.10), (3.11), and (3.12).

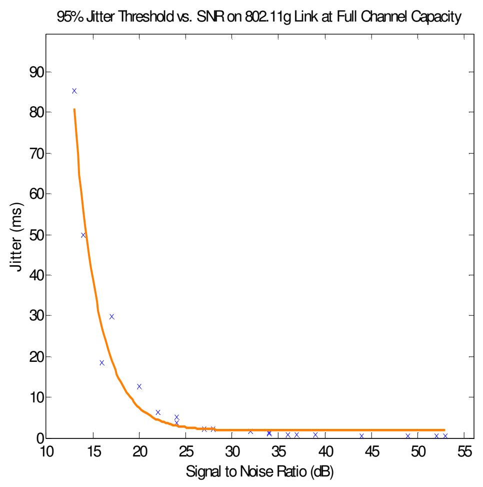
$$MJ(SNR) = 238.88 \bullet \exp(-0.194 \bullet SNR) + 0.321 \text{ [ms]} \tag{3.10}$$

$$95JT(SNR) = 10625 \bullet \exp(-0.377 \bullet SNR) + 1.770 \text{ [ms]} \tag{3.11}$$

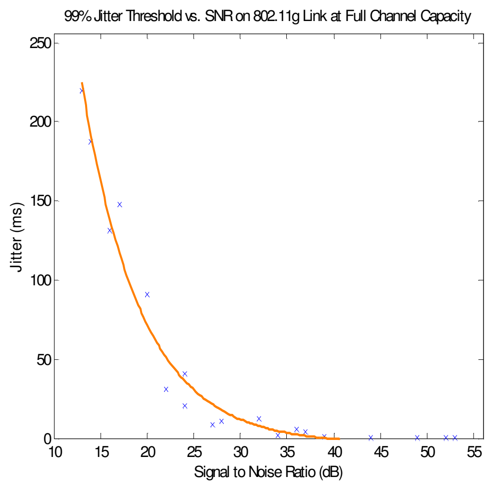
$$99JT(SNR) = 1782 \bullet \exp(-0.158 \bullet SNR) - 3.667 \text{ [ms]} \tag{3.12}$$



(a) Average Jitter



(b) 95% Jitter Threshold



(c) 99% Jitter Threshold

Figure 73: Jitter Behavior vs. SNR on 802.11g Link at Full Channel Capacity

The last metric that was recorded during the throughput tests was packet loss. Like the jitter characteristics, the percentage of lost packet regresses exponentially as the signal-to-noise ratio is increased. Figure 74 shows the packet loss percentage as a function of SNR as well as the exponential decay line of fit which can be seen in (3.13). Similar to the latency test, packet loss decreases exponentially with increasing SNR at data rates close to the channel capacity; however, greater packet loss is seen as more of the available channel is utilized.

$$PL(SNR) = 155.18 \bullet \exp(-0.250 \bullet SNR) + 0.279 \ [\%] \tag{3.13}$$

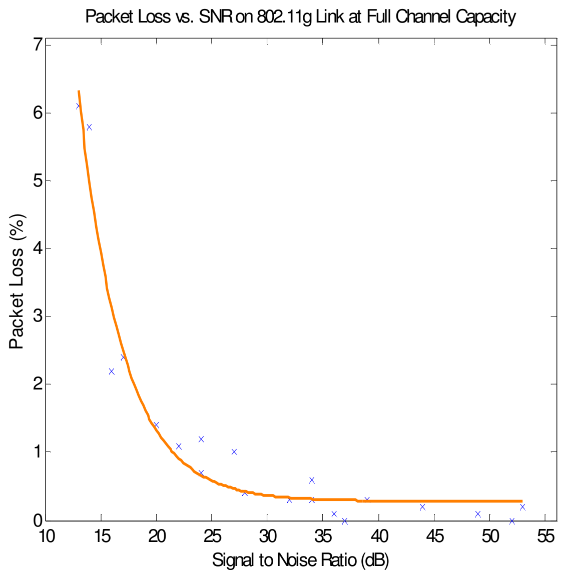
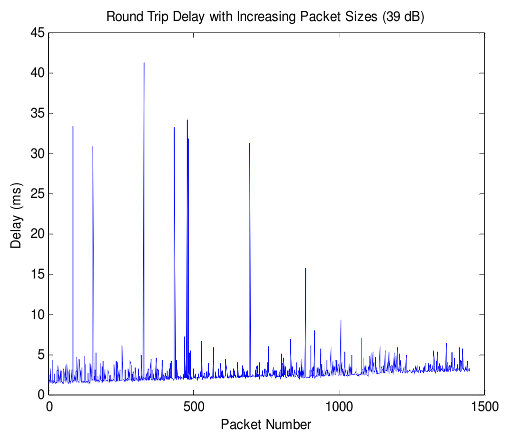


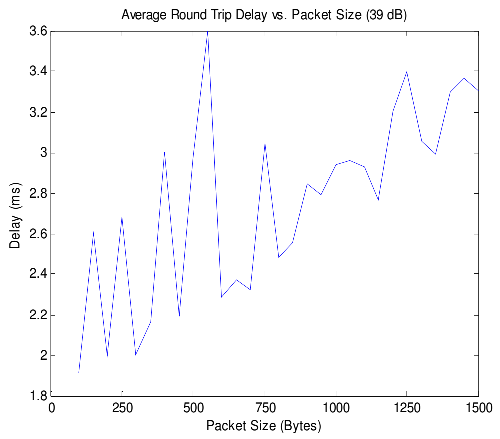
Figure 74: Packet Loss vs. SNR on 802.11g Link at Full Channel Capacity

<u>3.3.3 Latency vs. Packet Size</u>

802.11g latency was also measured as a function of packet size. Throughout most of the tests, there appeared to be an increase in latency as the size of the packets was increased. This increase in delay was very small though (<5 ms) and would be negligible in the scope of streaming media applications. Figure 75 shows the results from a delay vs. packet size test conducted with a signal-to-noise ratio of 39 dB. Figure 75(a) shows the measured round trip delay as a function of time. From this figure, it is easy to see the upwards trend of the delay as the test progresses and the size of the packets are increased. It is also clear that the increase due to increasing packet size is very small (~2 ms), and would have little to no impact on streaming media applications. Figure 75(b) shows the average round trip delay as a function of packet size. This curve was produced by averaging the values from Figure 75(a) in intervals of fifty as the packet size was increased every fifty packets during the test. Although an upwards trend can still be seen in this graph, the presence of spikes due to one or two packets with a longer than normal delay make it a little difficult to observe.



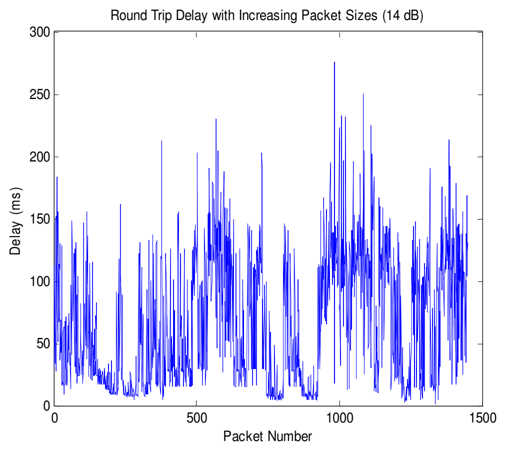
(a) Round Trip Delay vs. Test Packet Number



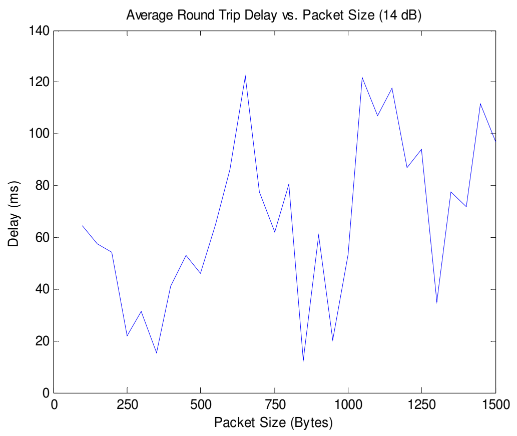
(b) Round Trip Delay vs. Packet Size

Figure 75: Results from an 802.11g Delay vs. Packet Size Test with a SNR of 39 dB

As the signal-to-noise ratio worsens, the clear upwards trend in latencies disappears as there are more significant factors contributing to the overall delay. Link adaptation algorithms along with the 802.11 MAC protocol at poor signal strengths cause a large variation in round trip delay that masks any effects due to increasing packet size. Figure 76 shows a delay vs. packet size test run with a SNR of 14 dB. It is difficult to see any effects that packet size may have on the overall delay in this instance. The results from the rest of the delay vs. packet size tests can be seen in Appendix A.



(a) Round Trip Delay vs. Test Packet Number

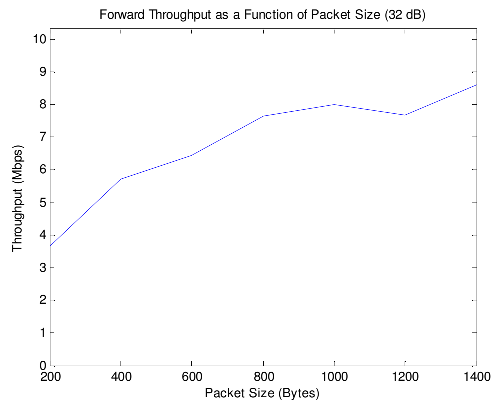


(b) Round Trip Delay vs. Packet Size

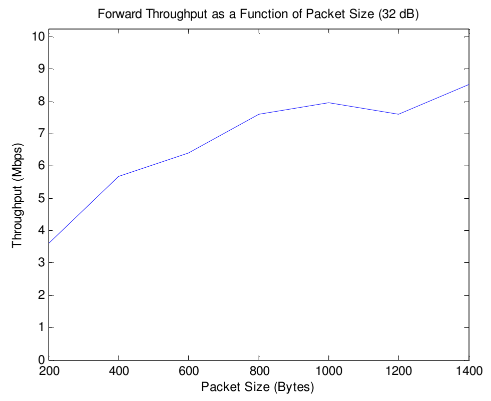
Figure 76: Results from an 802.11g Delay vs. Packet Size Test with a SNR of 14 dB

3.3.4 Throughput vs. Packet Size

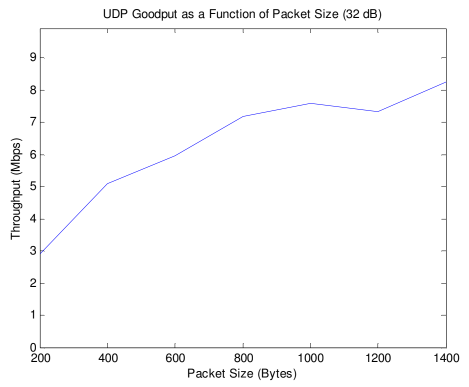
The last set of tests that were run over the 802.11g ad-hoc network were throughput vs. packet size tests. Figure 77 shows the results from a throughput vs. packet size test run with a signal-to-noise ratio of 32 dB present at both the sending and receiving nodes. Figure 77(a), (b) and (c) show the sending throughput, receiving throughput, and UDP goodput, respectively. Generally, increases in data rates were seen as the size of the packets was increased. The sending and receiving throughputs were almost identical throughout most of the tests while the performance gains seen in UDP goodput were magnified as packet sizes grew due to less overhead. Packet loss as well as jitter information was also measured and recorded during each test. This data can be viewed in Figure 77(d) through (g).



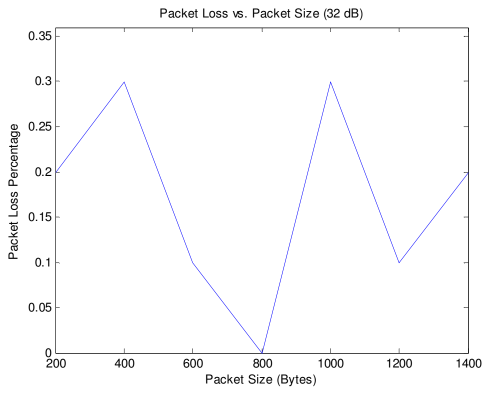
(a) Throughput vs. Packet Size (Client Report)



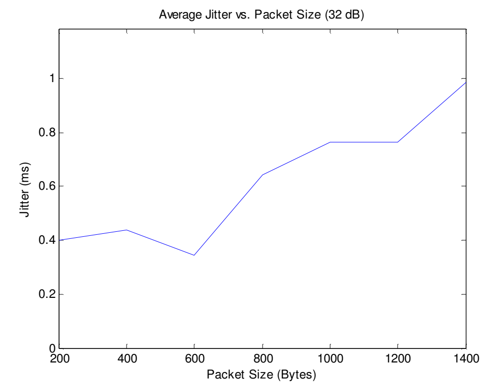
(b) Throughput vs. Packet Size (Server Report)



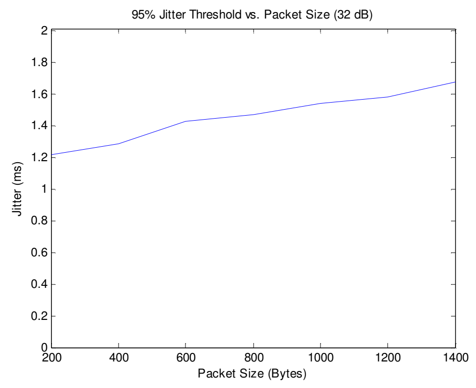
(c) UDP Goodput vs. Packet Size



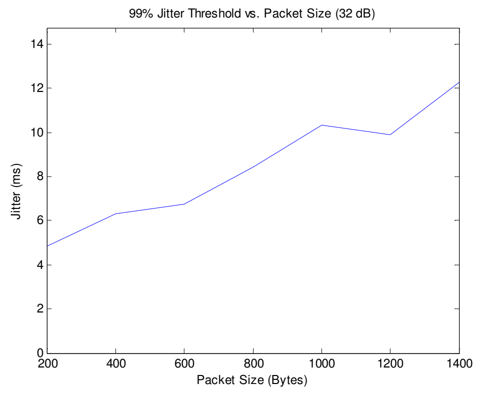
(d) Packet Loss vs. Packet Size



(e) Average Jitter vs. Packet Size



(f) 95% Jitter Threshold vs. Packet Size



(g) 99% Jitter Threshold vs. Packet Size

Figure 77: Results from a Throughput vs. Packet Size Test with a SNR of 32 dB

Because there were large discrepancies in performance at different signal strengths, most of the data gathered in the throughput vs. packet size tests needs to be viewed in three dimensions. Figure 78 shows a 3D plot of throughput vs. packet size at varying signal strengths. In general, throughput increased as the signal-to-noise ratio increased; however a greater increase in throughput was seen using larger packets when compared to smaller packets. The increase in throughput followed the same general shape as the altered error function given in (3.7), but a higher maximum throughput was achievable only by increasing the size of the datagrams.

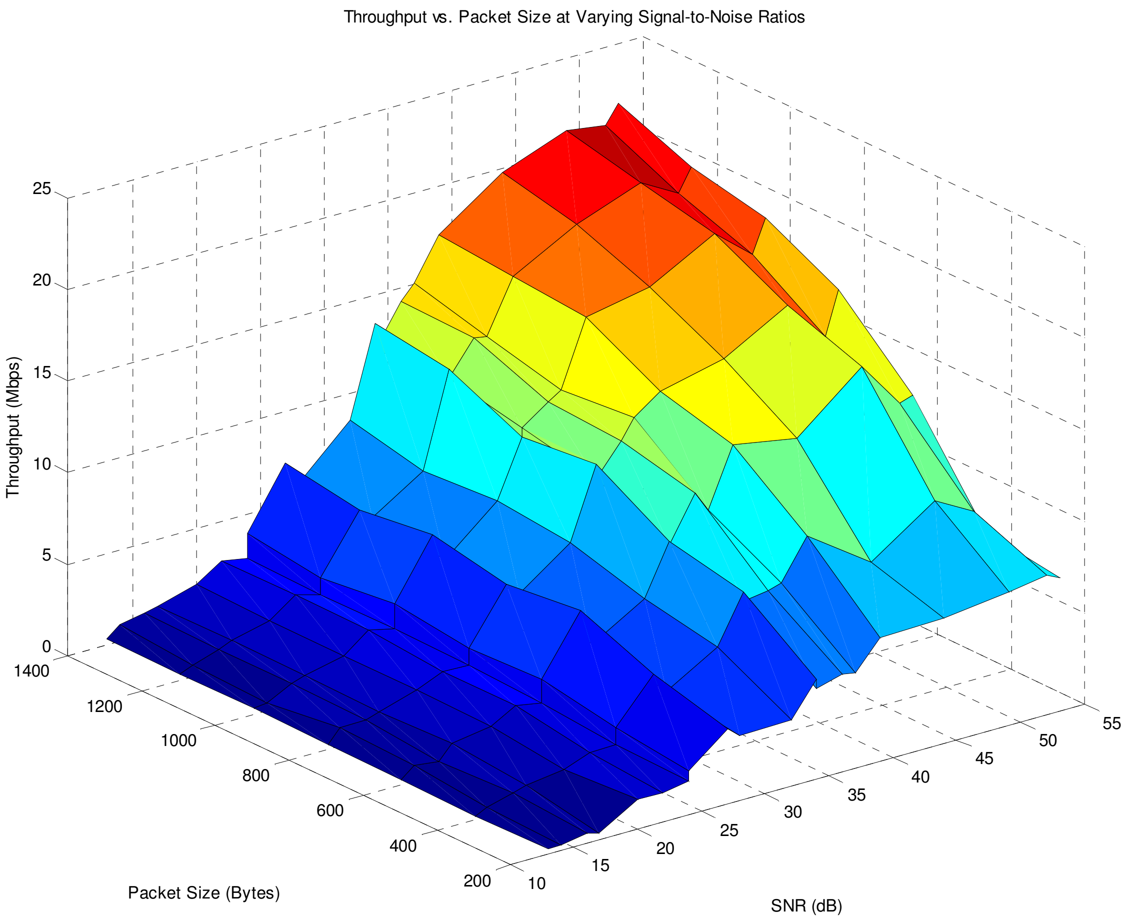


Figure 78: Throughput vs. Packet Size at Varying Signal-to-Noise Ratios on 802.11g Channel

Figure 79 shows the average jitter vs. packet size at varying signal-to-noise ratios. These test results show that jitter increases as signal-to-noise ratio decreases; however the increase in jitter is more apparent when using larger sized packets. The 95% and 99% jitter thresholds have similar behavior.

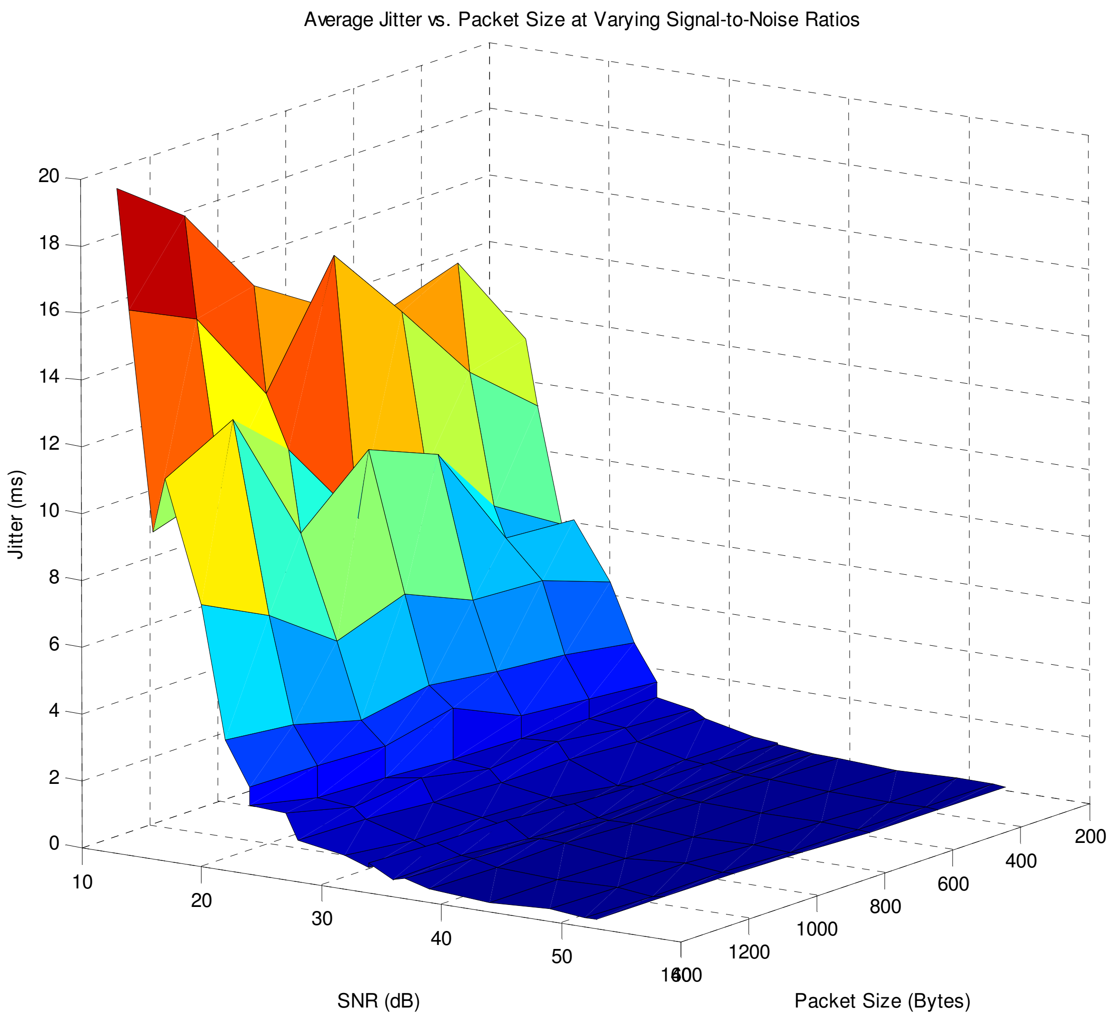


Figure 79: Average Jitter vs. Packet Size at Varying Signal-to-Noise Ratios on 802.11g Channel

The last metric that was measured during the throughput vs. packet size tests was packet loss. Like jitter, packet loss increases as the signal-to-noise ratio worsens; however, the increase in packet loss is more pronounced when using larger packets compared to smaller packets. Figure 80 shows packet loss as a function of packet size at varying signal strengths on an 802.11g ad-hoc network.

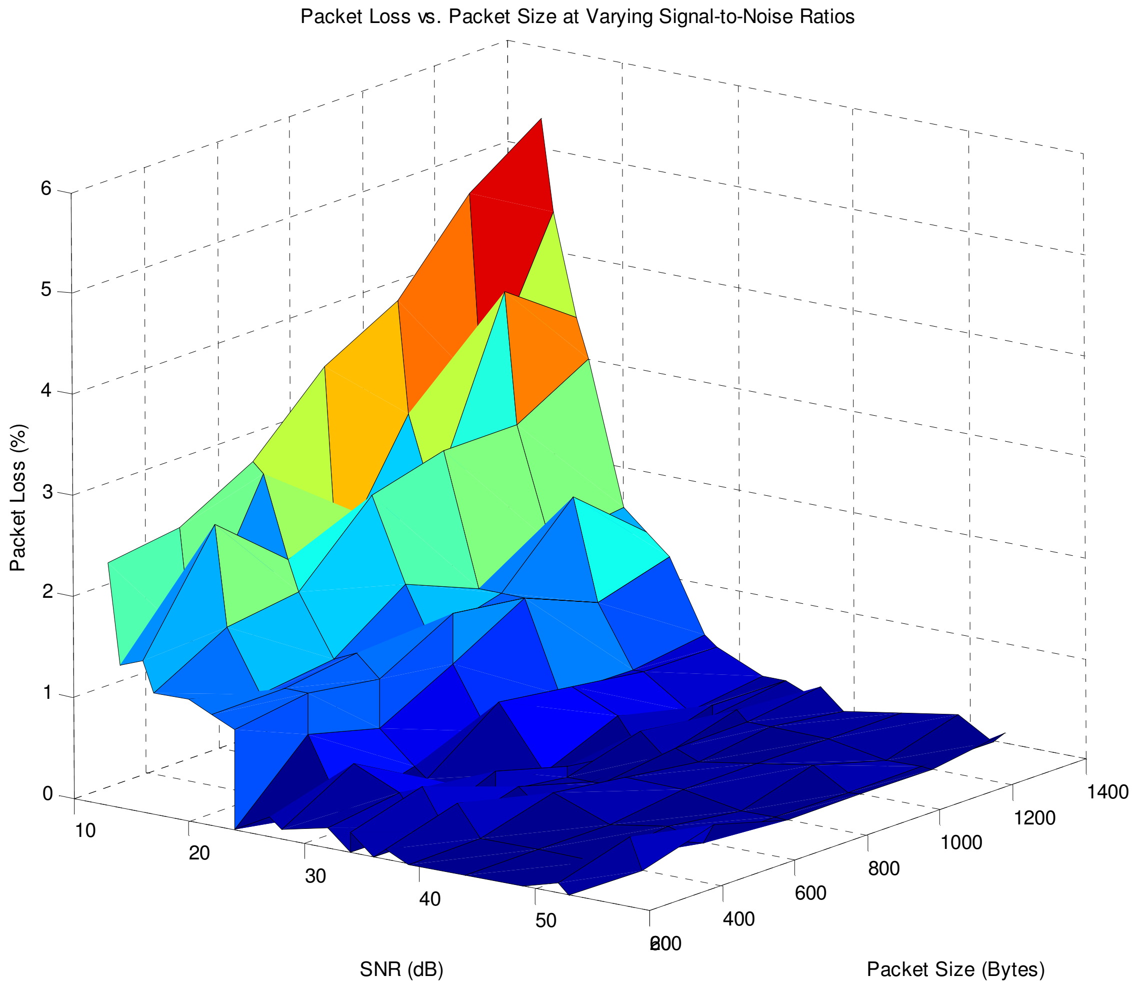


Figure 80: Packet Loss vs. Packet Size at Varying Signal-to-Noise Ratios on 802.11g Channel

<u>3.3.5 Antenna Performance</u>

In addition to the performance testing on the 802.11g channel, the transmission range of the protocol was also examined. As mentioned in Section 3.2, two external antennas were tested in addition to the natural range of the adapter. For each test, the server (receiving station) was equipped with a HyperGain "Rubber Duck" Antenna (Figure 63(a)) which is a reasonable expectation to have present at a base station, hospital or other gateway device. To test the transmission range, the client was initially placed 10 meters from the server, and the SNR was tracked at the server for a period of one minute using the NetStumbler software. This process was carried out using just Alfa USB 802.11b/g Network Adapter with no antenna, the HyperGain "Rubber Duck" Antenna (Figure 63(a)), and the

HyperGain “Range Doubler” Antenna (Figure 63(b)) on the client node (sending station). The client was then moved to a distance of 20 meters, and the measurements were repeated. The range between the client and server was increased in increments of 10 meters all the way to a distance of 200 meters. Figure 81 shows how the SNR measurements were configured along with the hardware and software present at both the client and server nodes.

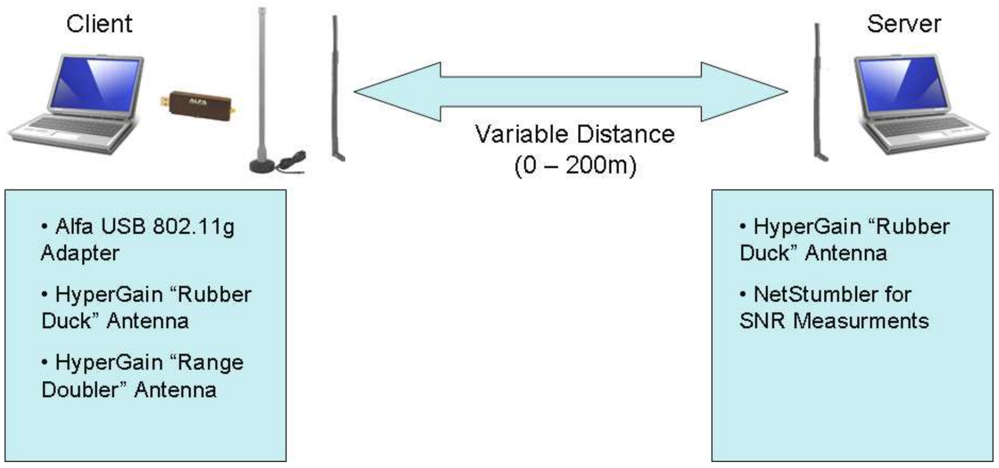

Client
Server
Variable Distance
(0 – 200m)
• Alfa USB 802.11g Adapter
• HyperGain “Rubber Duck” Antenna
• HyperGain “Range Doubler” Antenna
• HyperGain “Rubber Duck” Antenna
• NetStumbler for SNR Measurments

Figure 81: Measuring SNR on 802.11g Link

The SNR measurements taken at the server could then be matched up with the performance tests to determine what type of performance could be expected at a given distance using a particular antenna. Figure 82 shows the results of the SNR measurements taken at the server node as a function of distance.

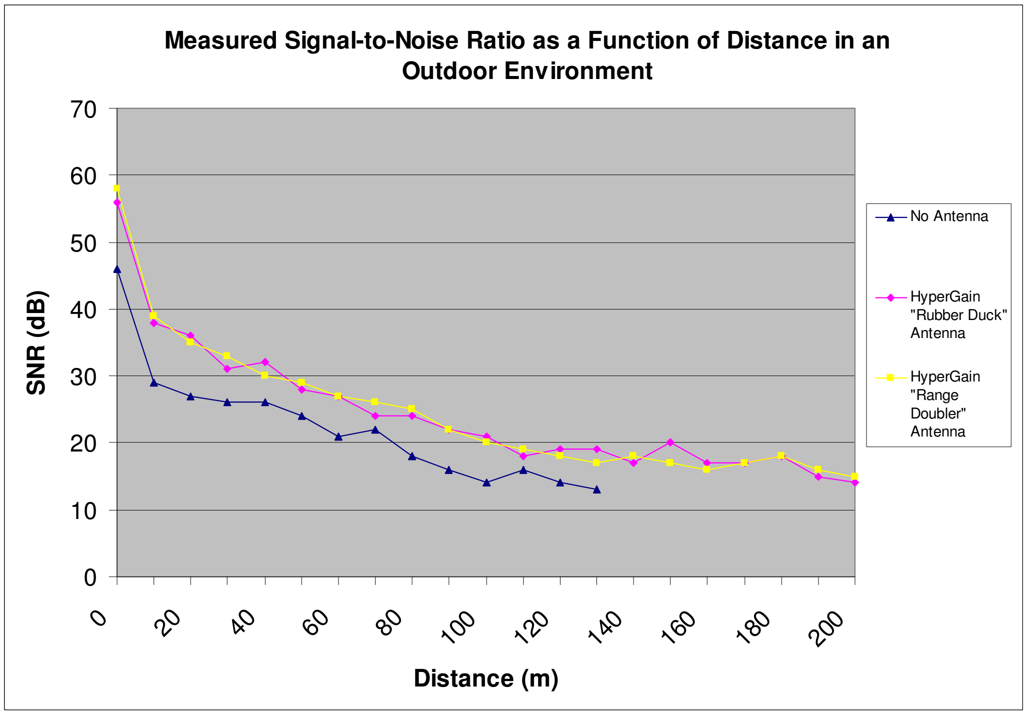


Figure 82: Measured SNR as a Function of Distance with Client using Various External 802.11 Antennas

The results show that both the HyperGain “Rubber Duck” Antenna and the HyperGain “Range Doubler” Antenna had similar performance in an outdoor environment with a clear line of sight. Both of these antennas showed a continuous improvement of 5 to 10 dB in SNR when compared to the case where no antenna was used. Based on the performance tests, it appears that reliable transmission using 802.11g can be made up to distances of around 150 meters when using an external antenna and 100 meters when using no antenna at the remote station. It should be noted that these results are unique to the *Realtek RTL8187* chipset, and could vary if a different 802.11 chipset were used.

3.3.6 Conclusion

After viewing the results from 802.11g testing, it is fairly obvious that signal-to-noise ratio has a significant effect on both the throughput and latency of the system. When using 802.11g at an average to above average SNR (>20 dB), most all streaming media applications should have sufficient bandwidth, latency, jitter and packet loss to operate without problems. As the signal-to-noise ratio drops below 20 dB or so, the available bandwidth may drop below the necessary data rate needed to stream video or voice information. Additionally, packet loss and jitter start to increase rapidly which further complicates streaming media performance. Lastly, it was shown that by utilizing an external antenna on the remote system, the range of acceptable SNR for media applications can be extended by a factor between 1.5 and 2 over the case where no external antenna is used. Table 12 shows a summary of the delay and throughput information discussed in this chapter.

Table 12: Summary of 802.11g Testing Results

Delay	Mean Round Trip Delay	$RTD(SNR) = 6212.2 \bullet \exp(-0.333 \bullet SNR) + 4.395$ [ms]
	Mean One-way Delay	$OWD \approx (1/2) \bullet RTD(SNR)$ [ms]
	Mean One-way Packet Loss	$PL(SNR) = 143.9 \bullet \exp(-0.289 \bullet SNR) + 0.038$ [%]
	Mean Jitter	$MJ(SNR) = 91.3 \bullet \exp(-0.145 \bullet SNR) + 0.194$ [ms]
	95% Forward Jitter Threshold	$95JT(SNR) = 151.2 \bullet \exp(-0.084 \bullet SNR) - 3.842$ [ms]
	99% Forward Jitter Threshold	$99JT(SNR) = 230.9 \bullet \exp(-0.066 \bullet SNR) - 10.390$ [ms]
Throughput	Mean Throughput	$BW(SNR) = 21.9 \bullet (1/2) \bullet (erf((SNR - 32.5)/11.5) + 1)$ [Mbps]
	Mean Packet Loss	$PL(SNR) = 155.18 \bullet \exp(-0.250 \bullet SNR) + 0.279$ [%]
	Mean Jitter	$MJ(SNR) = 238.88 \bullet \exp(-0.194 \bullet SNR) + 0.321$ [ms]
	95% Forward Jitter Threshold (100% Channel Capacity)	$95JT(SNR) = 10625 \bullet \exp(-0.377 \bullet SNR) + 1.770$ [ms]
	99% Forward Jitter Threshold (100% Channel Capacity)	$99JT(SNR) = 1782 \bullet \exp(-0.158 \bullet SNR) - 3.667$ [ms]

4 Wireless Communication via 3G Cellular Broadband

In recent years, enhancements to existing cellular networks have created many opportunities for high-speed wireless data applications. Currently available 3G networks can provide average downlink speeds of around 1 Mbps and average uplink speeds of around 350 kbps. This chapter will discuss wireless data transfer on a commercially available GSM (Global System for Mobile Communication) network. The following section will provide some background information on the history and evolution of the GSM family of wireless data standards. Section 4.2 will detail a protocol used to test the performance of the cellular data network. The last two sections of the chapter will provide the results and analysis of the data obtained during performance testing.

4.1 Background

This section will provide a brief description of the historical and technological evolution of the GSM family of wireless data standards. GSM is the most popular standard for cellular voice service in the world. The *GSM Association* estimates that 82% of the global market uses the GSM standard [50]. The GSM family of data services includes GPRS (General Packet Radio Service), EDGE (Enhanced Data Rates for GSM Evolution), WCDMA (Wideband Code Division Multiple Access), and HSDPA (High Speed Downlink Packet Access). GPRS is classified as a 2G standard, while EDGE is 2.5G and WCDMA/HSDPA are 3G technologies. With each successive data standard, throughput is increased and latency is decreased paving the way for more robust network applications.

The technical approaches have changed from the 2G to 3G standards. Both GPRS and EDGE use TDMA (time division multiple access) to transmit data while most of today's 3G technologies utilize CDMA (code division multiple access). Table 13 shows many of the currently available wireless technologies, and the technical approaches used by each one.

Table 13: Summary of Different Wireless Technological Approaches [50]

Approach	Technologies
TDMA	GSM, GPRS, EDGE
CDMA	CDMA2000 1xRTT, CDMA2000 EVDO, WCDMA, HSDPA, 802.11b
OFDM	WiMAX, 802.11a, 802.11g

4.1.1 GPRS

GPRS, the world's most pervasive wireless data service, is a packet-based network solution which supports connectivity to IP networks. As can be seen in Figure 83, GPRS is essentially the addition of a packet-data infrastructure to a GSM network. The main components necessary for the addition of GPRS on a GSM network are the Serving GPRS Support Node (SGSN) and the Gateway GPRS Support Node (GGSN). Packet based traffic is forwarded from the base station controller to the SGSN which is responsible for authenticating and tracking mobile stations in its area. The SGSN performs switching operations for IP traffic just as the mobile switching center does for voice traffic. The SGSN then forwards the packet data to the GGSN which is a gateway to external networks such as the internet. The GGSN is responsible for dynamically assigning IP to mobile stations [50].

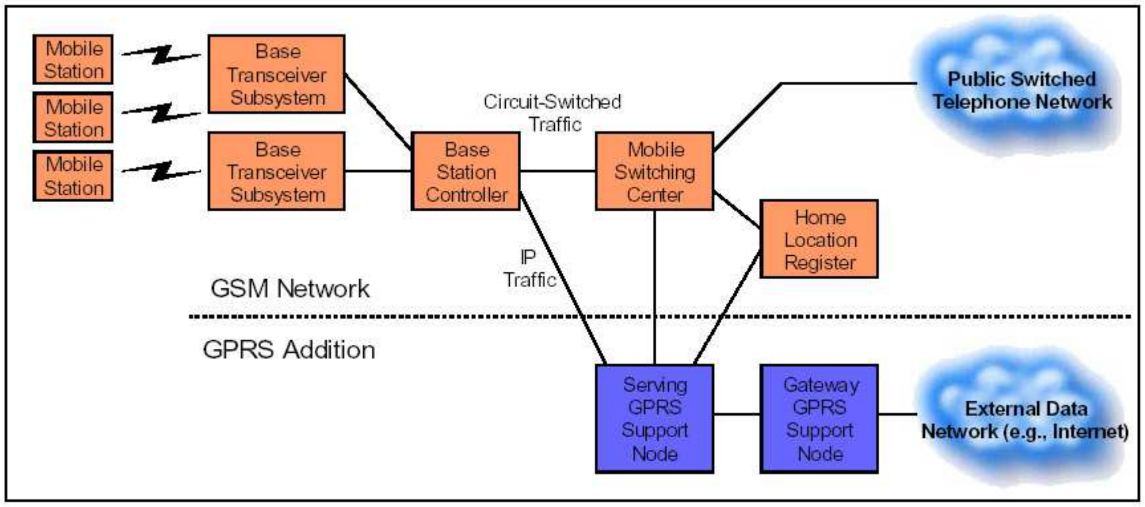

Mobile Station
Mobile Station
Mobile Station
Base Transceiver Subsystem
Base Transceiver Subsystem
Base Station Controller
Circuit-Switched Traffic
Mobile Switching Center
Public Switched Telephone Network
Home Location Register
IP Traffic
GSM Network
GPRS Addition
Serving GPRS Support Node
Gateway GPRS Support Node
External Data Network (e.g., Internet)

Figure 83: GPRS Network Infrastructure [50]

Each GSM radio channel is 200 KHz wide and is divided into eight timeslots that in total last 4.6 ms. The network then dynamically assigns different functions to each timeslot such as the broadcast control channel, circuit switched functions (e.g. voice calls or circuit-switched data calls), the packet broadcast control channel, and packet data channels. By dynamically assigning the functions of the timeslots, network efficiency is improved. An example of the timeslot structure can be seen in Figure 84. GPRS uses one of four modulation and coding schemes to encode data in each timeslot. The first two coding schemes (referred to as MCS-1 and MCS-2) can support user data rates of up to 10 kbps in a single timeslot. Up to four timeslots can be assigned to a single user on the downlink resulting in throughputs of around 40 kbps. MCS-3 and MCS-4 can support data rates approaching 20 kbps within a single timeslot [50]. Table 14, on page 120, gives more information regarding the coding schemes used by GPRS. The modulation and coding scheme used depends on the deployment of the GPRS network as well as the user device used to access the network.

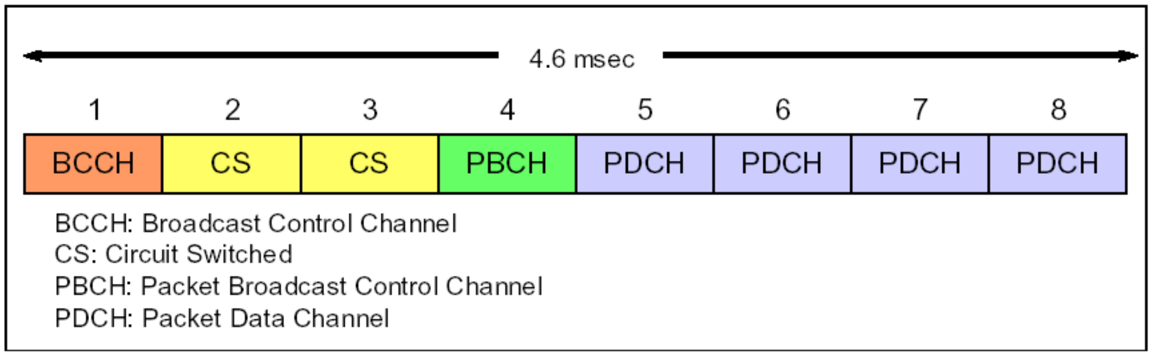

4.6 msec
1
2
3
4
5
6
7
8
BCCH
CS
CS
PBCH
PDCH
PDCH
PDCH
PDCH
BCCH: Broadcast Control Channel
CS: Circuit Switched
PBCH: Packet Broadcast Control Channel
PDCH: Packet Data Channel

Figure 84: Example of GSM / GPRS Time Slot Structure [50]

4.1.2 EDGE

EDGE, also referred to as EGPRS (Enhanced GPRS), provides significant performance enhancements to GPRS networks without the need to change hardware. It does so by enhancing the radio interface while allowing the hardware (SGSN, GGSN etc.) and timeslot structure to remain intact. Only software upgrades are needed to update a GPRS network to EDGE. EDGE increases typical user data rates by a factor of around three over GPRS and also reduces round-trip latency. Increased data rates are achieved by improving the spectral efficiency over those of GPRS [50]. Figure 85 shows the performance increase of EDGE over GPRS in terms of throughput per timeslot as a function of C/I (carrier-to-interference) ratio, which is equivalent to signal-to-noise ratio. This graph assumes a 50% network load. If the traffic load on the network were to be increased, the curves would begin to shift slightly to the right, meaning a higher C/I ratio would be necessary for the same throughput. Conversely, if the traffic were decreased, the curves would shift to the left.

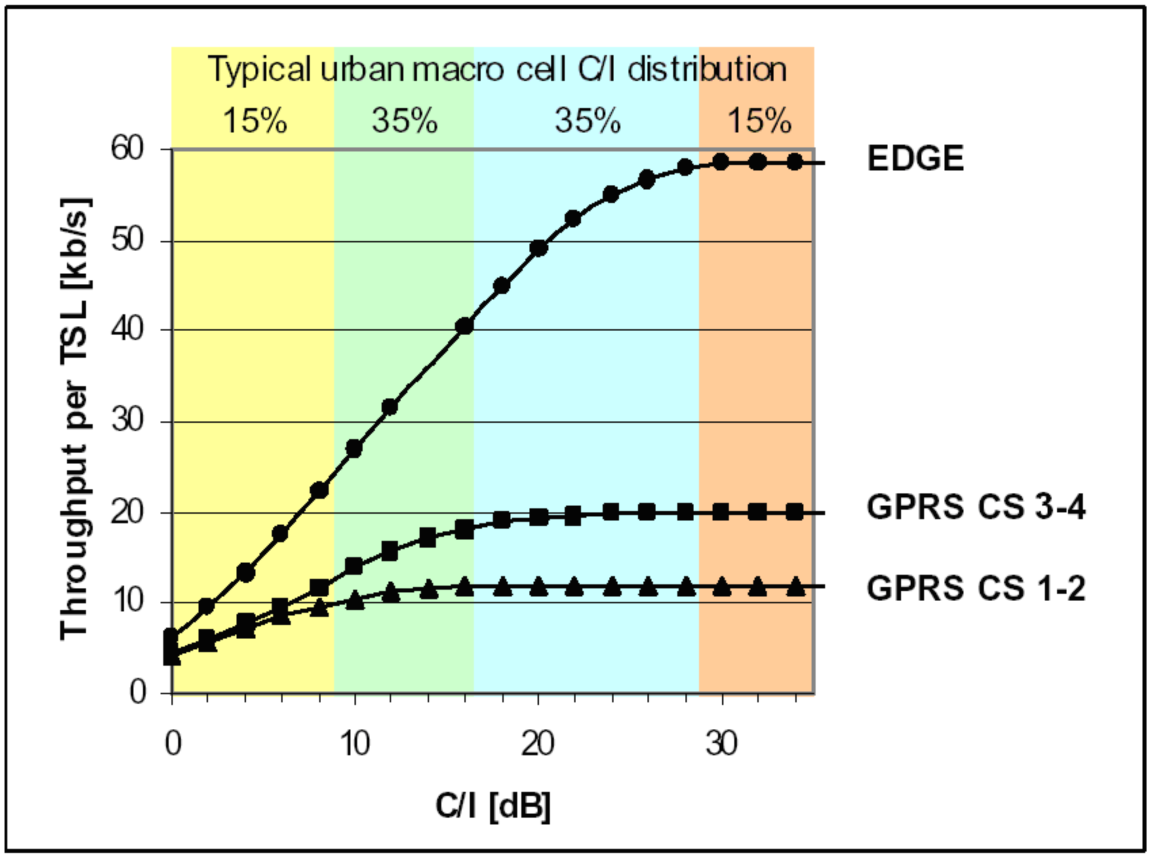


Figure 85: EDGE Performance Increase over GPRS at 50% Network Load (C/I = Carrier-to-Interference Ratio) [50]

EDGE applies a number of advanced techniques to the radio link to improve spectral efficiency, and consequently data rates. The first technique is the addition of a new modulation technique called Orthogonal Phase Shift Keying (8-

PSK) which allows three bits to be encoded in each transmitted symbol. In contrast, GPRS uses Gaussian Minimum Shift Keying (GMSK) which only encodes one bit per symbol. EDGE is backwards compatible with GPRS so it too uses GMSK under poor radio conditions or on GPRS networks [50]. Table 14 shows the different modulation schemes used by EDGE and GPRS as well as the maximum theoretical throughput per timeslot.

Table 14: GPRS and EDGE Modulation Schemes [50]

Technology	Modulation Scheme and Coding	Modulation	Throughput per Time Slot (kbps)
GPRS, EDGE	MCS-1	GMSK	8.8
	MCS-2	GMSK	11.2
	MCS-3	GMSK	14.8
	MCS-4	GMSK	17.6
EDGE	MCS-5	8-PSK	22.4
	MCS-6	8-PSK	29.6
	MCS-7	8-PSK	44.8
	MCS-8	8-PSK	54.4
	MCS-9	8-PSK	59.2

The second technique used to increase data rates is called *link adaptation* where the network can automatically choose the modulation and coding scheme as well as adjust the number of bits used for error control. The network will choose a lower data rate and use more bits for error control under poor radio conditions to ensure successful data delivery, and high data rates under good radio condition to boost throughput. The last noteworthy technique employed by EDGE is called *incremental redundancy*. Using this technique, if blocks of data are received in error, the blocks are immediately retransmitted using a different coding scheme, significantly increasing the likelihood for a successful transmission. The theoretical peak network capacity under good radio conditions for EDGE is 476.3 kbps, however actual user throughput is typically between 100 and 130 kbps [50].

4.1.3 UMTS – WCDMA

WCDMA (Wideband Code Division Multiple Access), sometimes referred to as UMTS (Universal Mobile Telecommunication System), is a true 3G technology that utilizes wideband CDMA technology. UMTS generally refers to the complete radio system including all the hardware components while WCDMA refers to the radio interface technology used to encode and transmit data. Unlike EDGE, UMTS networks require additional hardware components to be added to GSM networks for interoperability. The greatest advantage of WCDMA technology over GSM is the wideband nature of the spectrum, which allows it to translate the available spectrum into high data rates [50].

WCDMA uses direct-sequence spread spectrum technology to transmit data to different users on the same physical channel by associating a code with each individual user. Whether for voice or data, WCDMA systems can alter the capacity of each user channel every 10 ms. To increase the capacity of a channel, the amount of spreading must be reduced meaning that shorter codes must be used. To reduce the capacity of a channel, the spreading factor must be increased. For example, voice channels typically use a spreading factor of 128 or 256, while a high-speed (384 kbps) data channel uses a spreading factor of 8. The maximum theoretical rate for WCDMA is just over 2 Mbps, obtained by combining three physical channels, each with a capacity of 768 kbps. Actual user throughputs are typically between 220 and 320 kbps [50].

4.1.4 HSDPA

HSDPA (High Speed Downlink Packet Access), which is also uses CDMA, achieves its performance increase over WCDMA by using techniques similar to those that improve EDGE performance past GPRS. Like EDGE, HSDPA uses *higher order modulation* as well as *fast link adaptation*. While WCDMA uses only QPSK modulation, HSDPA utilizes both QPSK and 16 QAM to achieve higher data rates. This is because QPSK only encodes two bits per symbol while 16 QAM encodes four bits in every symbol. Fast link adaptation refers to the practice of using different levels of error coding depending on the current conditions of the radio channel. For example, a three-quarter coding rate means that three-quarters of the bits are data bits, while one-quarter of the bits are error correcting bits [50]. Table 15 shows the different modulation and coding schemes employed by WCDMA and HSDPA. The variable number of codes used will be explained shortly.

Table 15: WCDMA and HSDPA Modulation Schemes [50]

Technology	**Modulation**	**Coding Rate**	**Throughput with 5 Codes**	**Throughput with 10 Codes**	**Throughput with 15 Codes**
WCDMA, HSDPA	QPSK	1/4	600 kbps	1.2 Mbps	1.8 Mbps
		2/4	1.2 Mbps	2.4 Mbps	3.6 Mbps
		3/4	1.8 Mbps	3.6 Mbps	5.4 Mbps
HSDPA	16 QAM	2/4	2.4 Mbps	4.8 Mbps	7.2 Mbps
		3/4	3.6 Mbps	7.2 Mbps	10.7 Mbps
		4/4	4.8 Mbps	9.6 Mbps	14.4 Mbps

Like WCDMA, HSDPA also assigns a number of codes within a single 5 MHz physical channel to create individual user channels. A single user can possess more than one code within the channel to provide increased data rates. The individual codes within the physical channel are referred to as *High Speed – Downlink Shared Channels* (HS-DSCH). In Figure 86, as an example, four users are sharing a single physical channel that is divided using fifteen different codes resulting in fifteen HS-DSCH. The total number of codes used in HSDPA can either be five, ten or fifteen. HSDPA will automatically adjust how users are assigned to the HS-DSCH depending on the demands of each user and resource availability. It should also be noted that HSDPA can adjust channel assignments every 2 ms which is significantly faster than the 10 ms of WCDMA. This results in higher data rates and lower overall latencies [50].

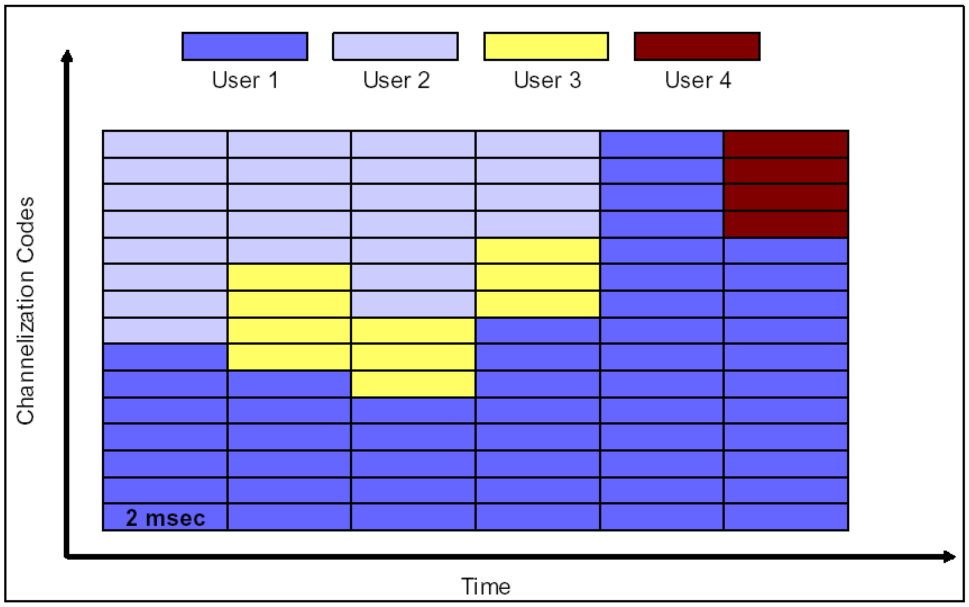


Figure 86: Example of HSDPA Downlink Shared Channel with Four Users [50]

The last two advanced transmission techniques used by HSDPA include *Fast Scheduling with User Diversity* and *Fast Hybrid Automatic Repeat Request* (Fast Hybrid ARQ). Fast scheduling with user diversity means that the users with the best instantaneous signal quality will be given more of the available channels than users with poor signal quality. Because signal quality varies somewhat randomly, most users can be serviced under optimum radio conditions improving the overall efficiency of the network. WCDMA schedules users in a round-robin fashion, which has proven to be somewhat inefficient in field tests. Fast Hybrid ARQ refers to the process of combining repeated data transmissions with prior transmissions to increase the chance of successful delivery. Using this process, it is possible to receive the same block of data with errors on two separate retransmissions, and still be able to successfully decode the data. Using the previously described approaches, HSDPA networks typically produce average download speeds in excess of 1 Mbps [50]. The enhancements in radio interface technology also produce lower latencies when measured from the cellular base station to the end user device. Figure 87 shows typical one-way delays from the cellular base station to the mobile device. This graph does not include latency incurred from external networks such as the internet.

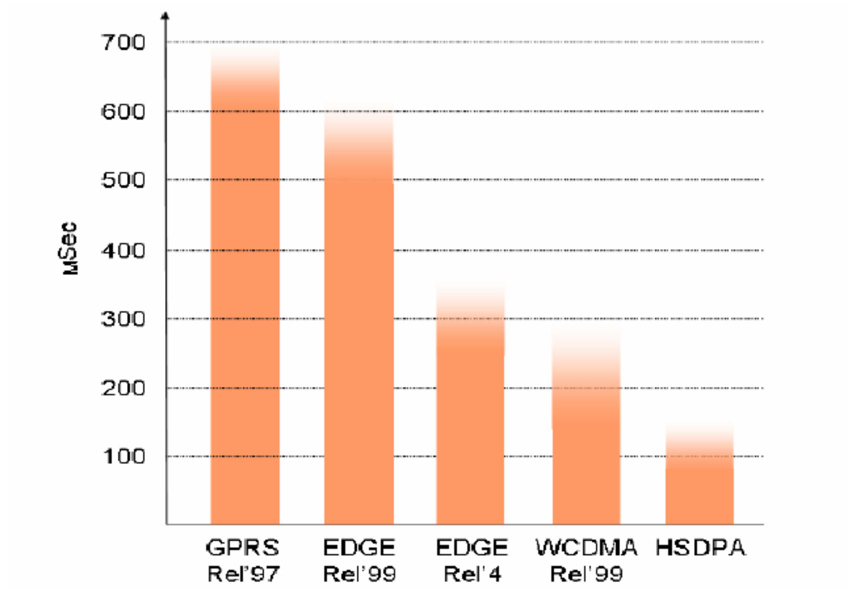


Figure 87: Typical Round-Trip Latencies of Wireless Data Technologies [50]

As HSDPA is used to optimize downlink performance, HSUPA (High Speed Uplink Packet Access) can be used to optimize uplink capacity. Because most network applications require a higher downlink speed than uplink speed, most vendors dedicate more of the available spectrum for the downlink. As a result, most wireless data networks are asymmetrical with a higher downlink capacity than uplink capacity. HSUPA, sometimes called E-DCH (Enhanced Dedicated Channel), helps to balance the capacity of network as well as improve uplink latency, which is beneficial to many applications such as VoIP. HSUPA uses many of the same techniques employed by HSDPA such as Fast Hybrid ARQ, fast scheduling, and a transmission time interval of 2 ms to achieve higher data rates [50]. It should be noted that not all HSDPA networks use HSUPA on the uplink, and HSUPA does not require HSDPA on the downlink. Also, the end user device must be HSUPA compatible to achieve the performance advantages. For streaming media applications from the mobile ultrasound system, the asymmetric nature of 3G channels is a disadvantage as the majority of data must be sent on the uplink.

4.2 Testing Protocol

The most prevalent HSDPA network that is commercially available in the Central Massachusetts area is the *BroadbandConnect Network* from *AT&T* (formerly *Cingular Wireless*). To perform testing on this network, a network subscription was purchased along with a *Sierra Wireless AirCard 875U* which is necessary for a PC to gain access to the network. AT&T's 3G network utilizes HSDPA on the downlink, and standard UMTS on the, uplink resulting in peak download speeds of 3.6 Mbps and peak upload speeds of 400 kbps. Typical download speeds are generally between 600 and 1400 kbps while typical upload speeds are between 220 and 320 kbps. In some areas, AT&T has deployed HSUPA on the uplink; however, this is not the case in the Worcester area. In locations where AT&T has deployed HSUPA, the peak uplink capacity is 2.0 Mbps, and typical uplink speeds are between 500 and 800 kbps. Figure 88 shows the coverage of the AT&T's BroadbandConnect Network as of June, 2008. The blue region shows the 3G coverage, while the orange region refers to areas that only support EDGE.

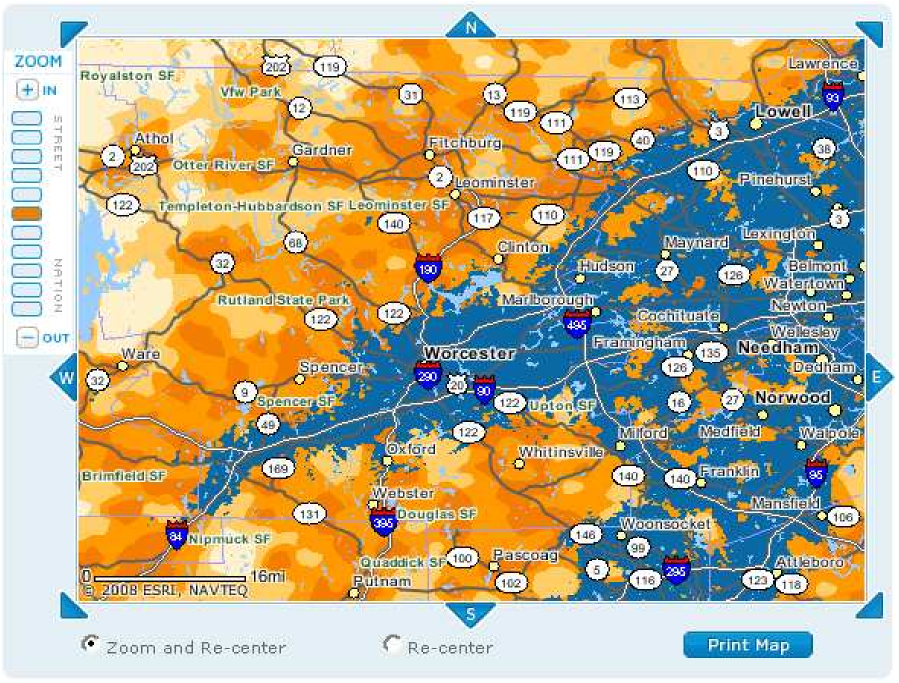


Figure 88: Broadband Connect Coverage in Central Massachusetts (June, 2003) [53]

The Sierra Wireless AirCard 875U, shown in Figure 89, interfaces with a computer via a USB connection. This makes it compatible with the third generation ultrasound system as there are available USB ports for external devices. The AirCard provides access to HSDPA, UMTS, EDGE, and GPRS networks, and supports a maximum download speed of 3.6 Mbps and a maximum upload speed of 400 kbps. It automatically connects to the highest quality network possible, and will default to a lower quality network only if necessary. For example, if a user on the AT&T BroadbandConnect Network moves out of the 3G coverage area, the AirCard will automatically connect to the EDGE network if possible.

Figure 89: Sierra Wireless AirCard 875U

The *AT&T Communication Manager* is a software package provided by AT&T used to manage connections to the network. The Communication Manager allows users to connect and disconnect to any AT&T networks that are within the range of the receiver. It automatically defaults to the highest quality network available; however, users can connect to a lower quality network (such as EDGE) using the Communication Manager. Additionally, the Communication Manager indicates the received signal strength at the receiver. Figure 90 shows the Communication Manager GUI, and the received signal strength indicator can be seen on the right side of the GUI.

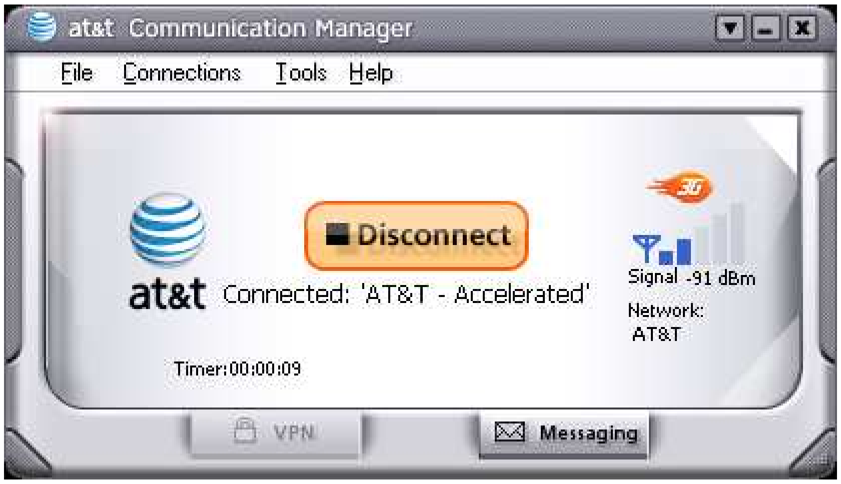

at&t Communication Manager
File Connections Tools Help
Disconnect
at&t
Connected: 'AT&T - Accelerated'
3G
Signal -91 dBm
Network:
AT&T
Timer:00:00:09
VPN
Messaging

Figure 90: AT&T Communication Manager

During preliminary performance tests, it was discovered that by default, AT&T blocks incoming connections to the remote system connected to the internet via a 3G connection. This meant that external clients with standard wired internet connectivity could not connect to the image server on the remote system nor could performance tests be conducted. To overcome this problem, a special service called *I2Gold* had to be purchased from AT&T. The I2Gold service provided a public static IP address that was accessible from any system with internet connectivity.

Testing of the 3G network was conducted as follows. A laptop computer was connected to the HSDPA network via the Sierra Wireless AirCard. The laptop, which will also be referred to as the client, ran the client application of the channel measurement toolbox. A desktop system located in the Atwater Kent building on the WPI campus in Worcester, MA was used to run the server application. The server computer was connected to the internet via a wired Ethernet connection through WPI's ECE network. Figure 91 shows a diagram of how the tests were configured on AT&T's 3G network.

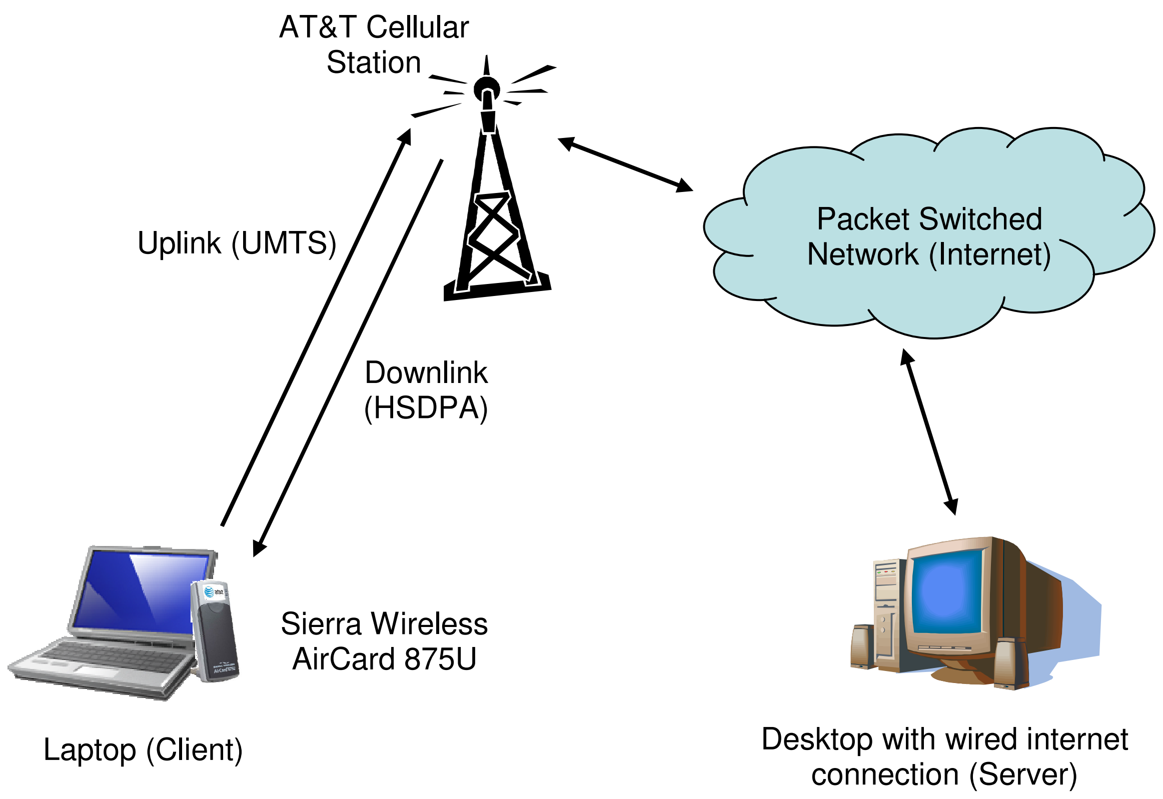

AT&T Cellular Station
Packet Switched Network (Internet)
Uplink (UMTS)
Downlink (HSDPA)
Sierra Wireless AirCard 875U
Laptop (Client)
Desktop with wired internet connection (Server)

Figure 91: Test Setup on AT&T BroadBand Connect Network

The laptop was moved to various indoor and outdoor locations to carry out measurements at different received signal strengths. Before a test was run, the received signal strength was recorded based from the AT&T Communication Manager GUI, and one of the channel measurement tests was run. If the signal strength changed in the middle of a test, the test was subsequently stopped and restarted using the new signal strength. Signal strengths ranged from -66 dBm (Excellent) to -96 dBm (Poor). If the signal strength dropped further below -96 dBm, the receiver would stop using the HSDPA network and default to the EDGE network. For the most part, the received signal strength stayed relatively stable if the laptop was not moved during a test. In total, thirteen data points were measured for each of the measurement tools available in the channel measurement application. Table 16 shows the parameters that were used for each of the 3G tests that were all described in Section 2.4.

Table 16: Test Parameters for 3G Measurements

Test	Test Length	Packet Size(s) (Bytes)
Delay	• 1000 Round Trip Packets	• 1500
Delay vs. Packet Size	• 50 Round Trip Packets at each interval • 29 total intervals	• 100 to 1500 in intervals of 50 bytes
Throughput	• 60 seconds in forward direction • 10 seconds at each interval in reverse direction • 10 intervals in reverse direction	• 1400
Throughput vs. Packet Size	• 1000 packets at each interval • 7 intervals	• 200 to 1400 in intervals of 200 bytes

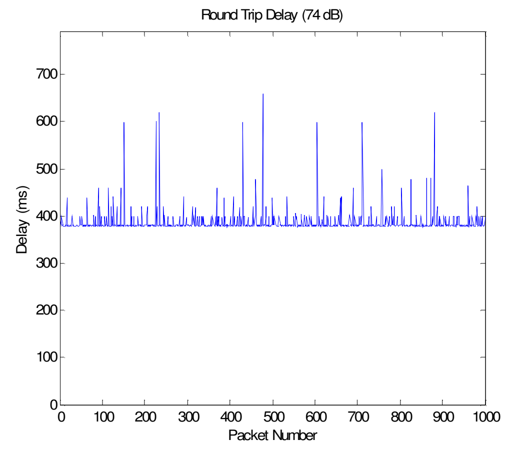
In addition to the twelve points measured at varying signal strengths, two sets of tests were conducted in a mobile environment to observe any differences in performance. To obtain these measurements, tests were conducted in a moving vehicle, being sure to stay within the coverage range of the 3G network. The received signal strength was not recorded during these tests as it was constantly changing.

4.3 Testing Results and Analysis

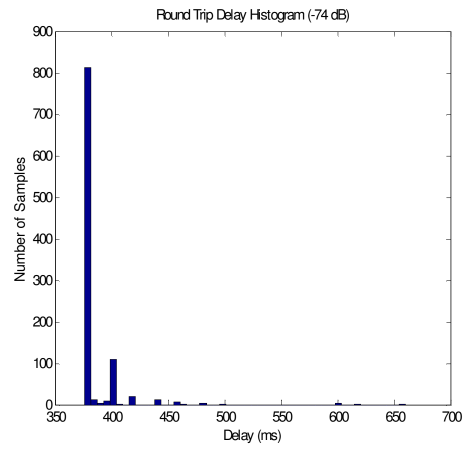
After sorting and plotting the results from the 3G cellular tests, it appears as though most of the channel properties are not greatly affected by the signal strength present at the receiver. This is good news from a telemedicine perspective because users can generally expect consistent throughput and latency as long as they are within the 3G coverage area. Users need not worry about unstable network performance due to varying received signal strength. The following sections will discuss the results from the 3G testing. Complete results for every measured data point are contained in Appendix B. As with the 802.11 tests, the term “forward” refers to the uplink (mobile to server) while the term “reverse” refers to the downlink (server to mobile).

4.3.1 Latency

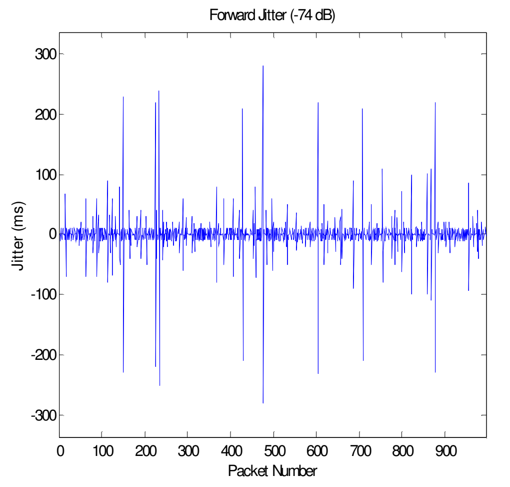
In total, twelve latency tests were run over the 3G network at varying received signal strengths. For each test, 1,000 round trip packets were used to measure round trip latency, forward packet loss, reverse packet loss, forward jitter and reverse jitter. Each of the test packets was 1500 bytes in size, and each test took approximately six to seven minutes to complete. Figure 92 shows the complete results from a single delay test conducted with a received signal strength of -74 dBm. These results closely resemble the results from the other delay tests, which can be found in Appendix B. It should also be noted that the delay test functions by sending only one packet at a time, which does not utilize the full capacity of the channel. For this reason, the packet loss measurements and jitter measurements differ from the results of the throughput tests where the full capacity of the channel is utilized during the test.



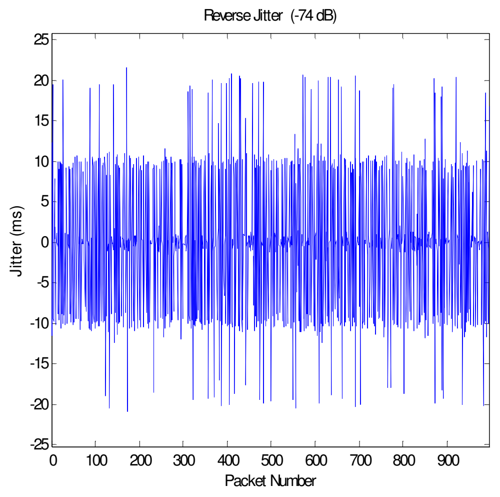
(a) Round Trip Delay vs. Time



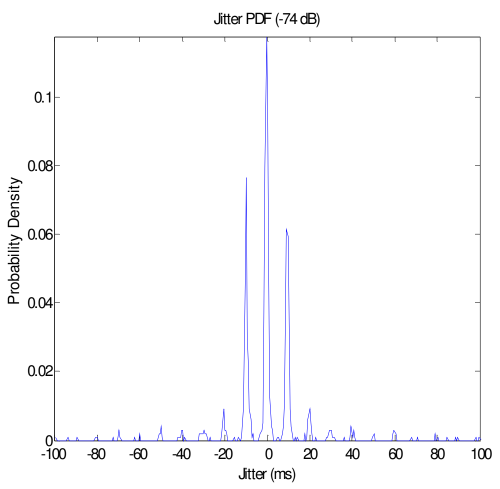
(b) Round Trip Delay Histogram



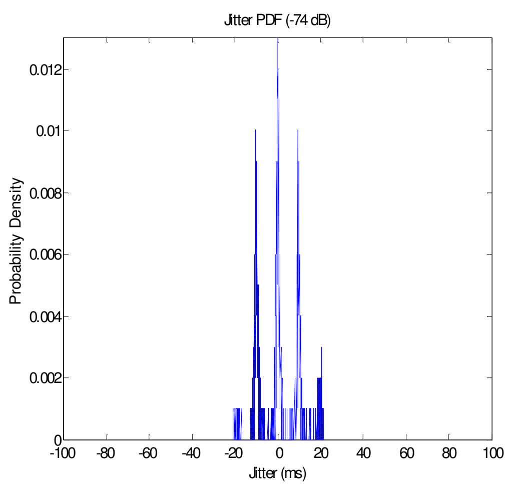
(c) Forward Jitter vs. Time



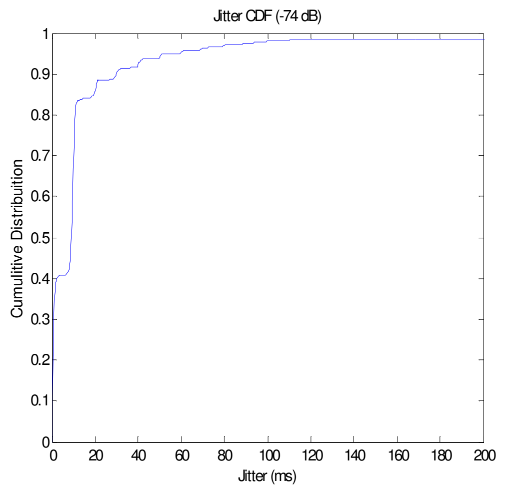
(d) Reverse Jitter vs. Time



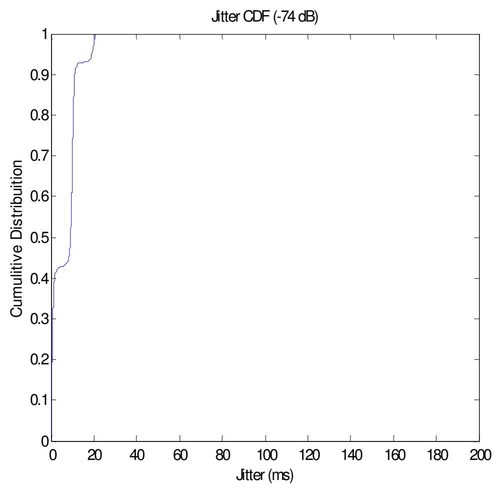
(e) Forward Jitter PDF



(f) Reverse Jitter PDF



(g) Forward Jitter CDF



(h) Reverse Jitter CDF

Max Delay = 660.1 ms
Min Delay = 375.9 ms
Mean Delay = 386.7 ms
Round Trip Standard Deviation = 24.7 ms
Packet Loss (Forward) = 5/1000
Packet Loss (Reverse) = 6/1000
Mean Forward Jitter = 14.2 ms
Forward Jitter (95%) = 60.1 ms
Forward Jitter (99%) = 220.4 ms
Mean Reverse Jitter = 6.6 ms
Reverse Jitter (95%) = 19.1 ms
Reverse Jitter (99%) = 20.4 ms

Figure 92: Results from 3G Delay Test with a Received Signal Strength of -74 dBm

As seen in Figure 92, each test measures the round trip delay, forward jitter and reverse jitter as a function of time. Additionally, packet loss in each direction is recorded during each test. Using these results, a histogram of the round trip delays (Figure 92(b)) could be plotted to better observe the distribution of the delays. Also, the Probability Density Function (PDF) and Cumulative Distribution Function (CDF) of the jitter data was plotted to gain a better understanding of the true behavior of the jitter in each direction. Figure 92(e) and (f) show the PDFs while Figure 92(g) and (h) display the CDFs of both the forward and reverse jitters. It should be noted that the magnitude (absolute value) of the jitter was used when plotting the CDF as well as for finding the average jitter in each direction. A summary of the plotted results can be found at the bottom of Figure 92. The values next to “Forward Jitter (95%)” and “Forward Jitter (99%)” correspond to the points at which 95% and 99% of the forward jitter magnitudes fall between zero and that value.

After looking at the results from each of the delay tests, it appears that packet latency is not affected by the received strength of the signal. Figure 93 shows the mean round trip delay measured at each data point tested. For example, the point at -74 dBm corresponds to the mean of the data plotted in Figure 92(a). The results show that there remains a fairly consistent round trip latency regardless of received signal strength. The overall mean round trip delay from all twelve tests came out to 391.7 ms.

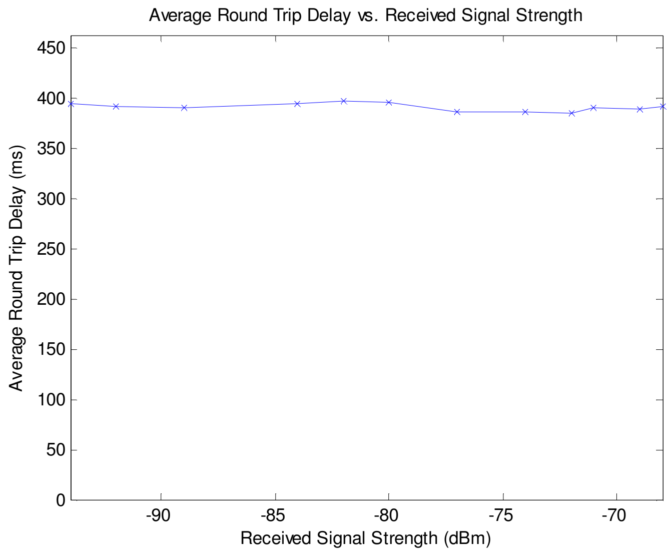
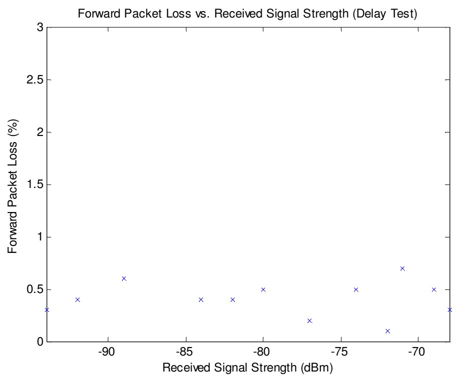
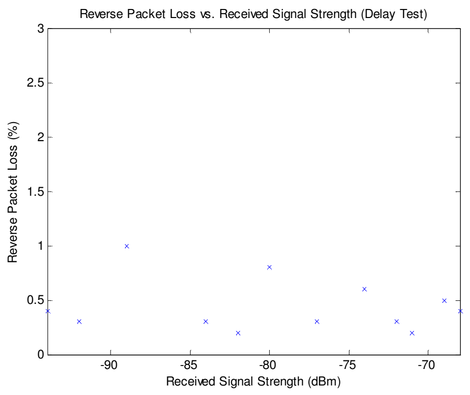


Figure 93: Round Trip Latency vs. Received Signal Strength on a 3G Network

The packet loss in each direction also appeared to remain fairly consistent throughout the each of the delay tests. Figure 94 shows both forward and reverse packet loss as a function of received signal strength. Although the packet loss does vary slightly from point to point, there is no clear trend that shows packet loss being directly affected by received signal strength. The mean forward packet loss for all of the tests was 0.41 % while the mean reverse packet loss was 0.44 %.



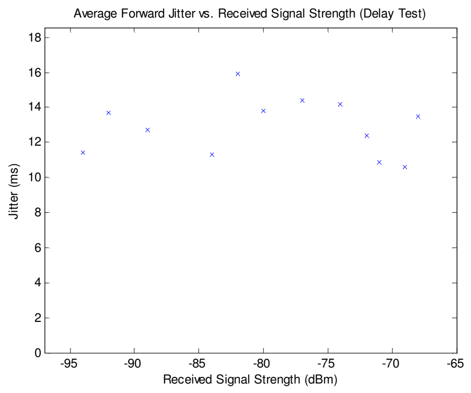
(a) Forward Packet Loss



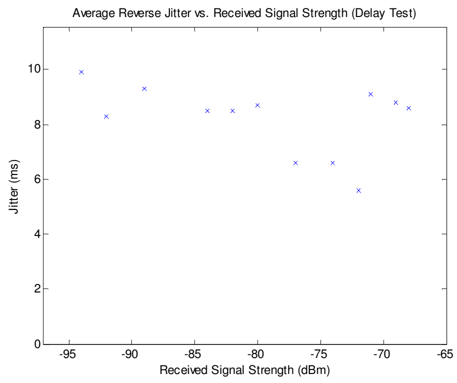
(b) Reverse Packet Loss

Figure 94: Packet Loss as a Function of Received Signal Strength on a 3G Network

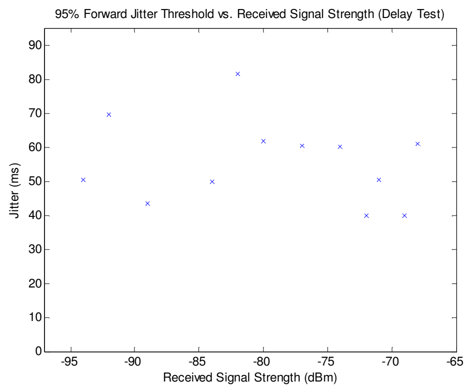
Like the round trip delay and packet loss, both the forward and reverse jitter do not appear to be affected in any deterministic way by varying received signal strengths. Figure 95 displays the jitter characteristics of both the uplink and downlink observed during the delay tests. Figure 95 (a) and (b) show the average magnitude of the jitter in each direction as a function of signal strength. The average forward jitter for all twelve tests was 12.9 ms while the average reverse jitter was 8.2 ms. The values in Figure 95 (c) and (d) correspond to the jitter threshold in which 95% of all measured values are less than. The mean 95% threshold for all twelve tests was 55.7 ms on the uplink and 20.2 ms on the downlink. Figure 95(e) and (f) show the 99% jitter thresholds in each direction as a function of received signal strength. The uplink had and average 99% jitter threshold of 184.1 ms while the downlink had an average of 29.3 ms over all twelve delay tests. Again, all of these measurements were made at a channel utilization that was significantly less than that of the channel capacity. The throughput tests presented later in this chapter will show that the jitter varies as data is sent at higher rates.



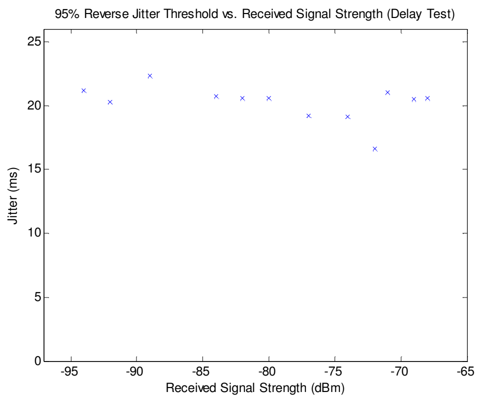
(a) Average Forward Jitter vs. Received Signal Strength



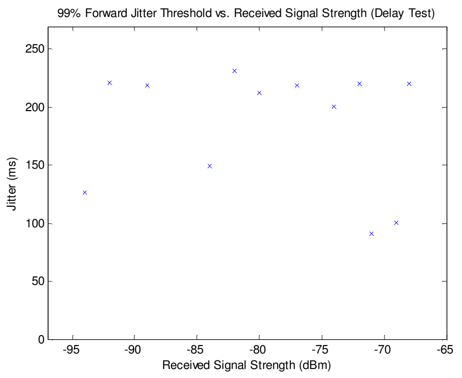
(b) Average Reverse Jitter vs. Received Signal Strength



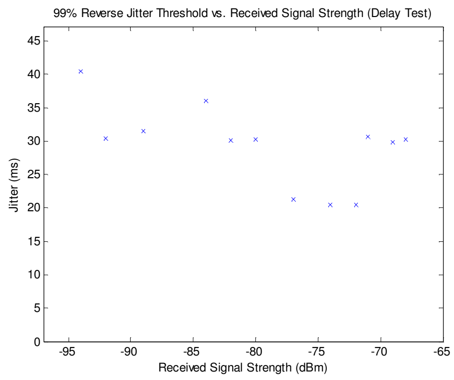
(c) 95% Forward Jitter Threshold vs. Received Signal Strength



(d) 95% Reverse Jitter Threshold vs. Received Signal Strength



(e) 99% Forward Jitter Threshold vs. Received Signal Strength



(f) 99% Reverse Jitter Threshold vs. Received Signal Strength

Figure 95: Jitter Characteristics as a Function of Received Signal Strength on a 3G Network

In addition to the round trip delay tests, one-way delay tests were performed as detailed in Section 2.6. Packets were sent in the forward direction via the 3G connection; however they were returned to the client through the WPI LAN which was a much lower latency link. This would result in measurements that were very close to the true latency of the uplink. Figure 96 shows the average measured one-way delays as a function of packet size. The parameters used in these tests were identical to those used in the round trip delay tests which used one thousand 1500 byte packets per test. Like the round trip delay tests, variation in the received signal strength had no obvious effects on packet latency.

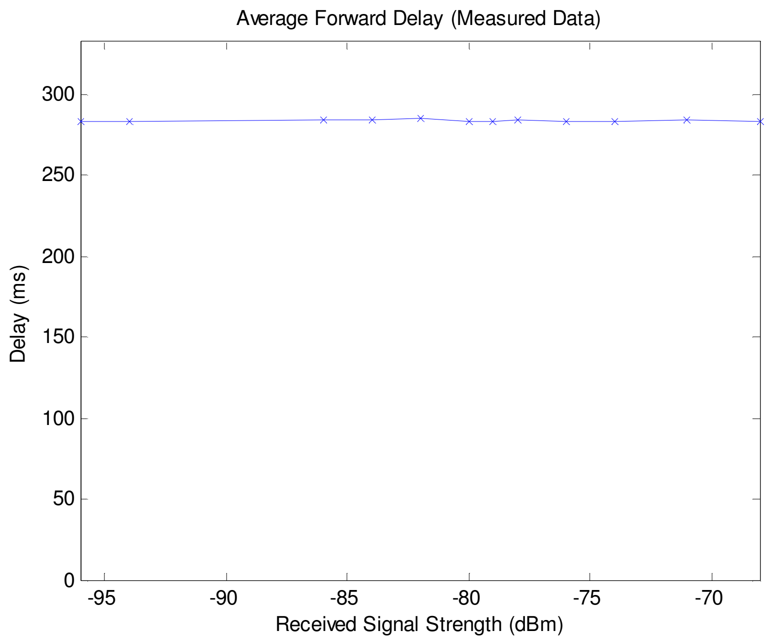


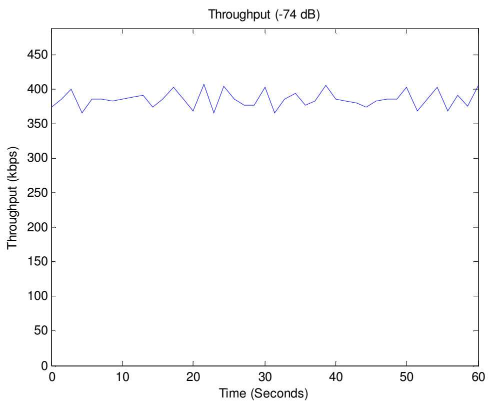
Figure 96: Measured Latency Data from One-Way Delay Tests vs. Received Signal Strength

The measured latency data had a mean of 283.5 ms, with an average jitter of 11.2 ms in the forward direction and an average jitter of 1.9 ms on the reverse link through the WPI network. Repeated round trip delay tests over the WPI network showed an average latency of 4.1 ms with .7 ms jitter in the forward direction and 2.1 ms jitter in the reverse direction. If the reverse latency through the WPI LAN is estimated at half of the round trip latency, or around 2 ms, then the true forward delay of the 3G network can be more closely approximated by subtracting this from the measured data. Therefore, the estimated average latency of the uplink is 281.5 ms with an average jitter around 12 ms as measured in both the one-way and round trip delay tests. To approximate the mean reverse delay, the average forward delay is subtracted from the average round trip delay, resulting in an estimated reverse delay of 103.7 ms. The jitter in the reverse direction is around 8.2 ms as measured in the round trip tests. These delay results are consistent with the expected results discussed in Section 4.1, and more specifically Figure 87.

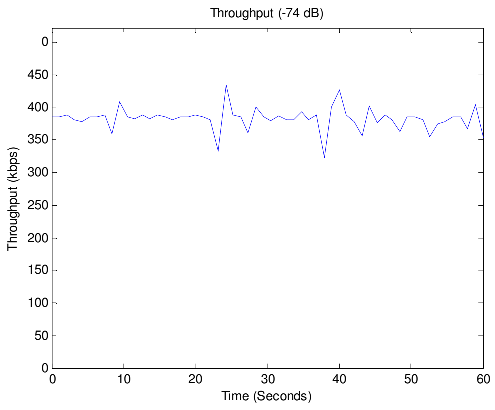
4.3.2 Throughput

As with the latency tests, twelve throughput tests were run using the channel measurement toolbox presented in Section 2.4. To test the bandwidth of the uplink, a stream of data was transmitted from the client for a period of sixty seconds. Each packet in the data stream was 1400 bytes in size. Packets were sent as fast as the channel would permit, and the throughput was measured at both the sending and receiving node. Factors such as packet loss and packet buffering sometimes lead to different throughputs measured at the receiver than at the sender.

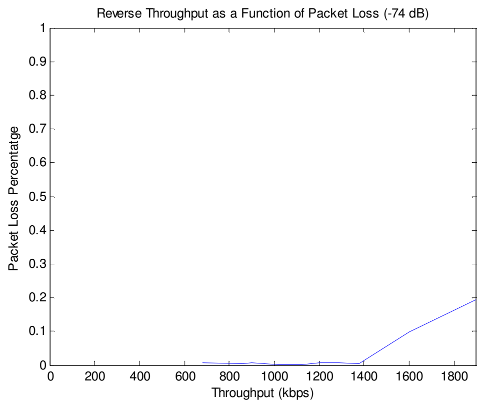
To measure the downlink channel capacity, data was sent at a set rate for a period of ten seconds, and packet loss was recorded. The sending rate was then increased to test the next point. In total, ten points were tested and packet loss was measured for each data rate. The initial data rate was set at 650 kbps and increased in increments of 125 kbps to the final data rate of 1900 kbps. The downlink bandwidth could then be determined by finding where excessive packet loss began due to sending at a higher rate than the channel capacity. This technique avoided using control packets to throttle the sender which could result in inaccurate throughput measurements. An example of this test can be seen in Figure 97 (c).



(a) Forward Throughput – Client Report



(b) Forward Throughput – Server Report

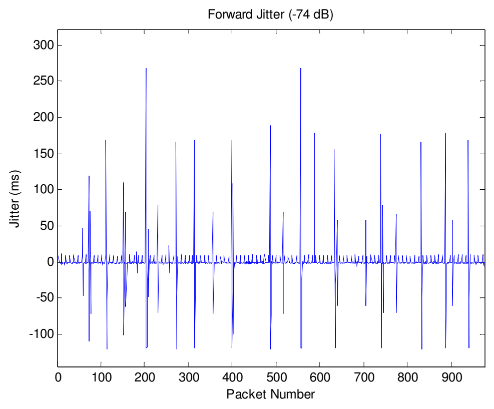


(c) Reverse Throughput vs. Packet Loss

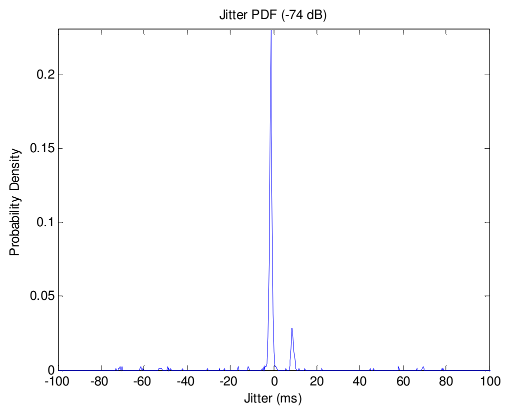
Max BW = 435.5 kbps
Min BW = 323.8 kbps
Mean BW = 382.8 kbps
Packets Sent = 1980
Packets Received = 1912
Drop Percentage = 3.4%

Figure 97: Throughput Results from 3G Throughput Test with a Received Signal Strength of -74 dBm

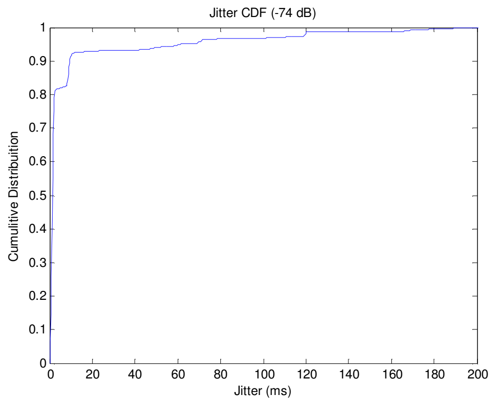
Lastly, the forward jitter was measured at different traffic loads to observe how jitter is affected at different data rates. To do this, data was sent at approximately 25%, 50% and 75% of the maximum channel capacity, and the jitter was recorded over time. Additionally, the PDF and CDF of the jitter measurements were plotted for each data point. Figure 98 shows the complete results from a throughput test run with a received signal strength of -74 dBm. These results were similar to those of the rest of the data points shown in Appendix B.



(a) Forward Jitter (25% Channel Load)



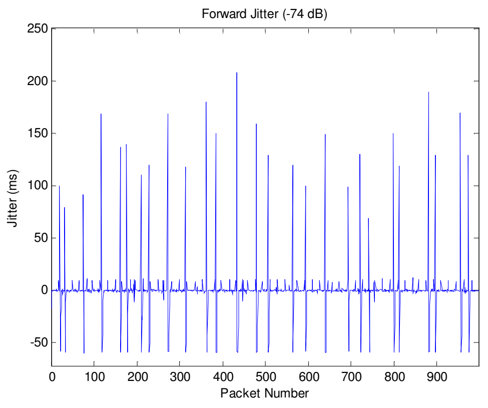
(b) Forward Jitter PDF (25% Channel Load)



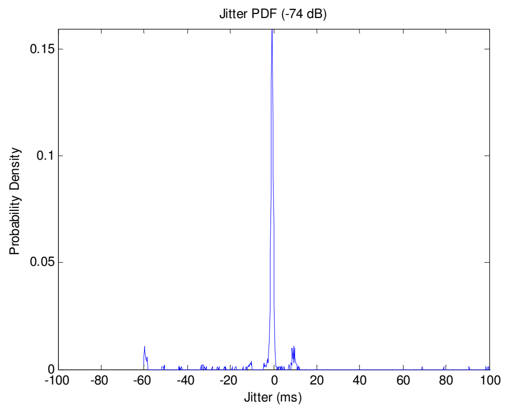
(c) Forward Jitter CDF (25% Channel Load))

25% Channel Capacity

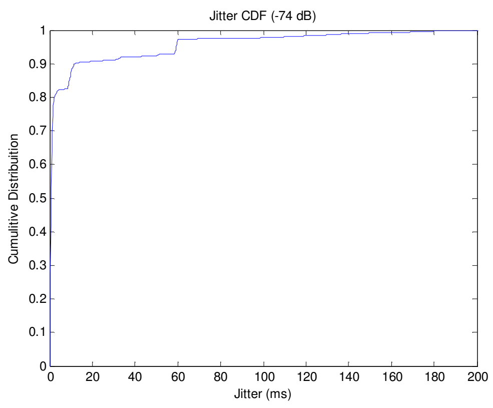
Throughput = 95.9 kbps
Mean Forward Jitter = 8.9 ms
Forward Jitter (95%) = 61.6 ms
Forward Jitter (99%) = 168.7 ms



(d) Forward Jitter (50% Channel Load)



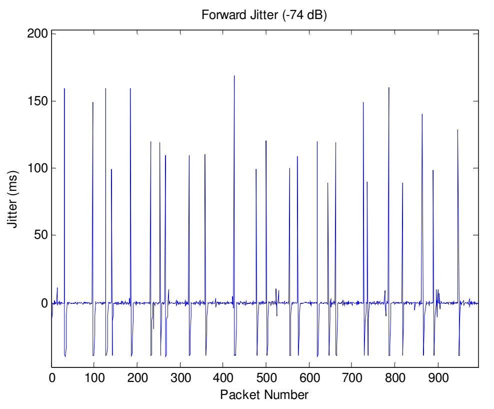
(e) Forward Jitter PDF (50% Channel Load)



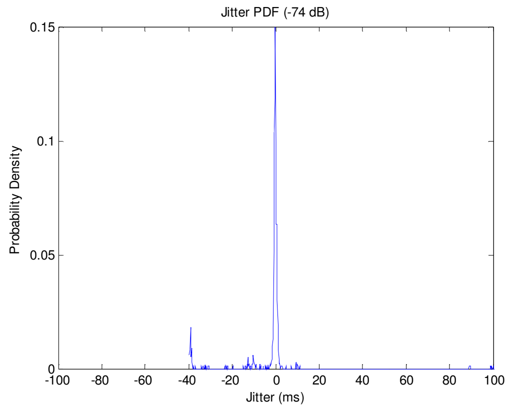
(f) Forward Jitter CDF (50% Channel Load))

50% Channel Capacity

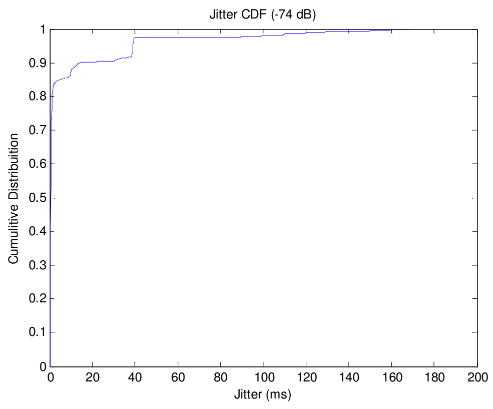
Throughput = 189.1 kbps
Mean Forward Jitter = 8.5 ms
Forward Jitter (95%) = 59.6 ms
Forward Jitter (99%) = 149.6 ms



(g) Forward Jitter (75% Channel Load)



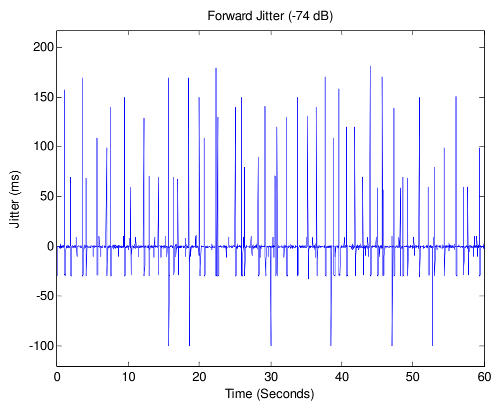
(h) Forward Jitter PDF (75% Channel Load)



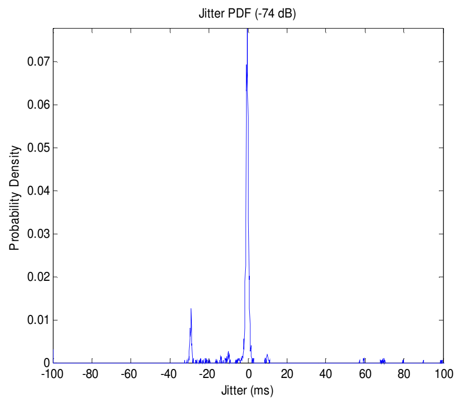
(i) Forward Jitter CDF (75% Channel Load))

75% Channel Capacity

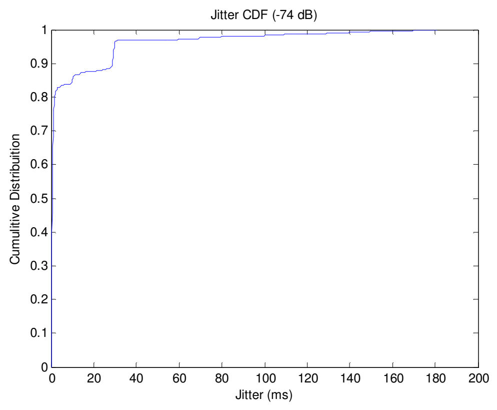
Throughput = 286.0 kbps
Mean Forward Jitter = 6.8 ms
Forward Jitter (95%) = 39.2 ms
Forward Jitter (99%) = 120.8 ms



(j) Forward Jitter (100% Channel Load)



(k) Forward Jitter PDF (100% Channel Load)



(l) Forward Jitter CDF (100% Channel Load))

Full Channel Capacity

Throughput = 382.8 kbps
Mean Forward Jitter = 7.1 ms
Forward Jitter (95%) = 29.8 ms
Forward Jitter (99%) = 138.7 ms

Figure 98: Jitter Results from 3G Throughput Test with a Received Signal Strength of -74 dBm

Like the latency tests, channel throughput does not appear to vary at different signal strengths. Figure 99 shows the average forward throughput at each data point as a function of received signal strength. These values correspond to the measurements taken at the receiver (base station) because the receiving throughput is much more important to streaming media applications than the sending throughput. The average forward throughput throughout all of the tests was 380.2 kbps. Some tests experienced bursts as high as 476 kbps while the minimum throughput seen during any of the tests was 295 kbps.

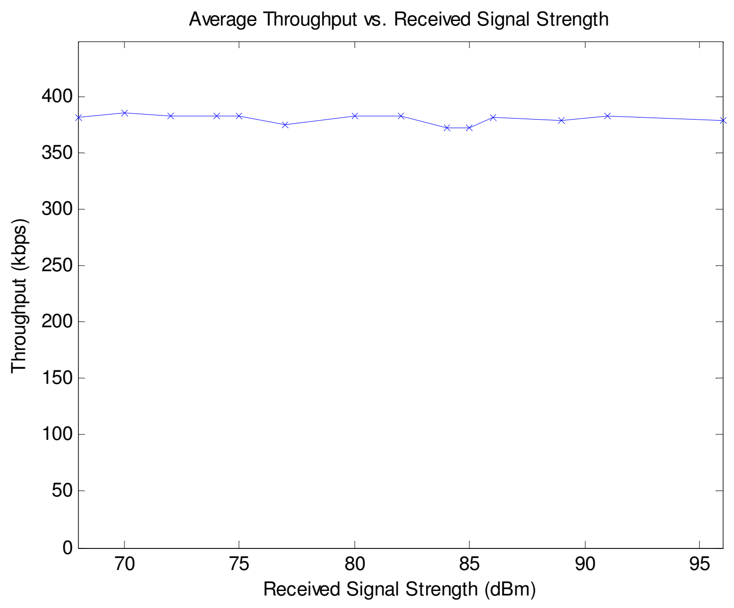


Figure 99: Average Forward Throughput vs. Received Signal Strength on 3G Network

The reverse channel capacity was determined by finding the data rate at which excessive packet loss started in the reverse direction (see Figure 97 (c)). For each test, the last data rate at which data could be sent with less than a 5% packet loss percentage was recorded. These values were then averaged to come up with an approximate channel capacity of the downlink. The results showed the downlink had an approximate throughput of 1361 kbps.

During each test, packet loss was measured in the forward direction at channel capacity. Figure 100 shows the packet loss percentage as a function of received signal strength for each of the tests. Most tests had a packet loss of around 3.5% while a few test had drop percentages of near 7%. The overall packet loss for all of the throughput tests was 4.0% in the forward direction. This is significantly higher than the .41% seen during the delay tests which shows that packet loss increases as the data rate increased. Again, the packet loss percentage did not appear to be directly affected by the received signal strength.

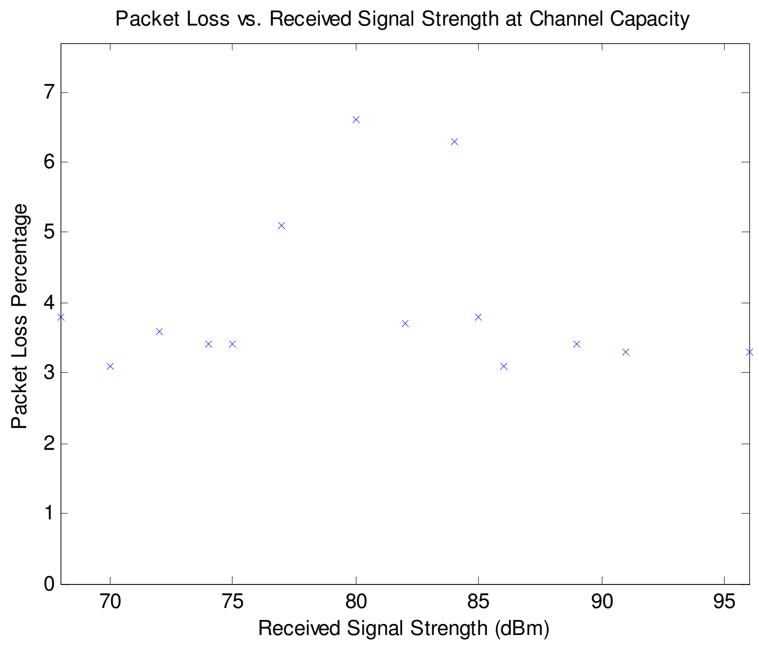
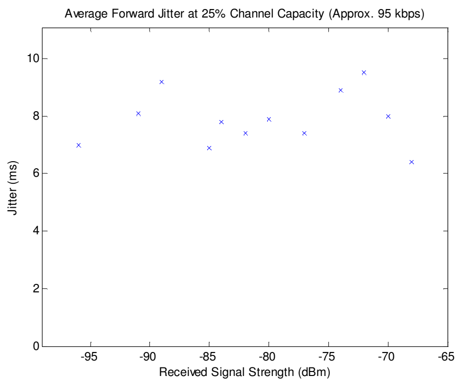
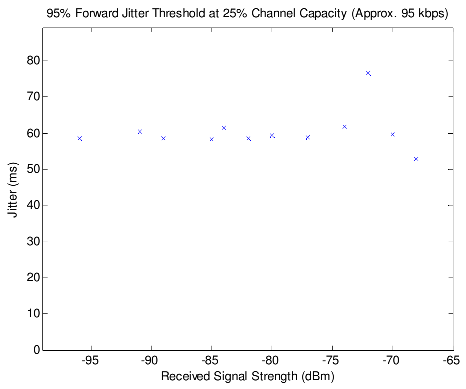


Figure 100: Packet Loss vs. Received Signal Strength on 3G Network at Channel Capacity (1500 byte packets)

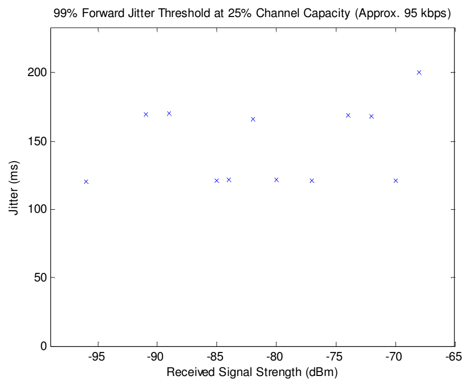
The last metric that was measured during these tests was forward jitter. Figure 101 shows the mean, 95% threshold (95% of jitter values less than this threshold) and 99% threshold for the forward jitter at different traffic loads. Following the plots, Table 17 provides a summary of the data shown in Figure 101.



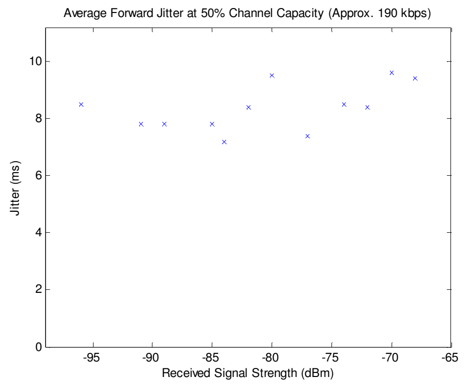
(a) Average Forward Jitter vs. Received Signal Strength (25% Channel Capacity)



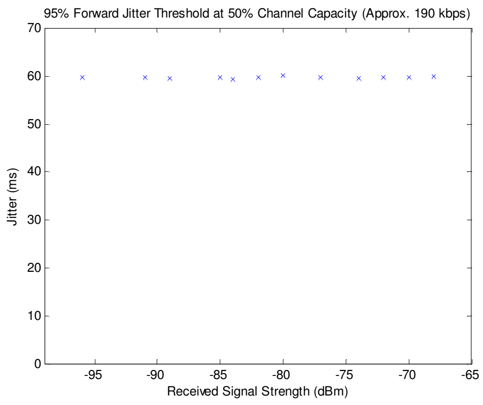
(b) 95% Forward Jitter Threshold vs. Received Signal Strength (25% Channel Capacity)



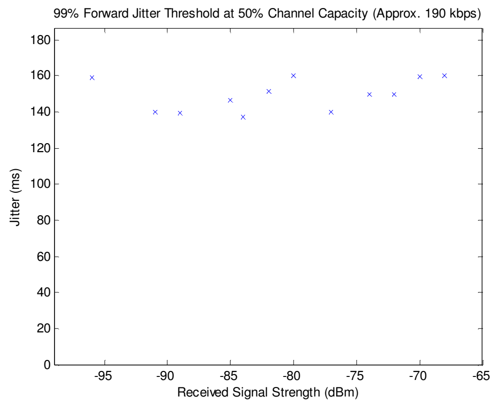
(c) 99% Forward Jitter Threshold vs. Received Signal Strength (25% Channel Capacity)



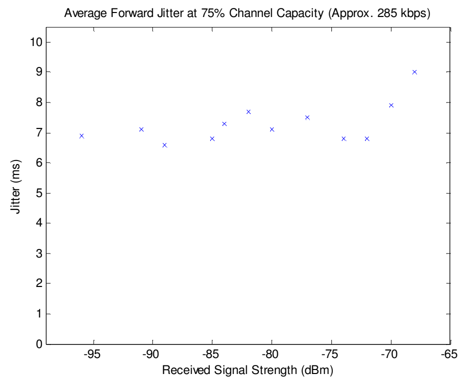
(d) Average Forward Jitter vs. Received Signal Strength (50% Channel Capacity)



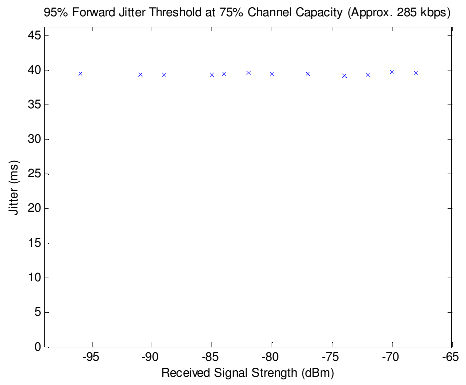
(e) 95% Forward Jitter Threshold vs. Received Signal Strength (50% Channel Capacity)



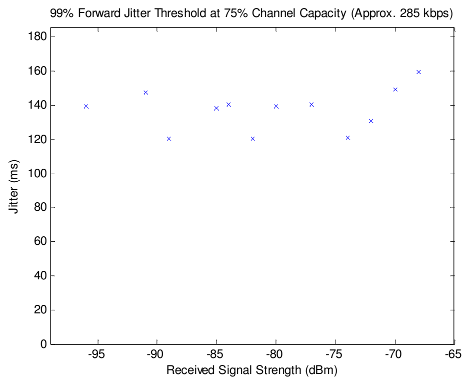
(f) 99% Forward Jitter Threshold vs. Received Signal Strength (50% Channel Capacity)



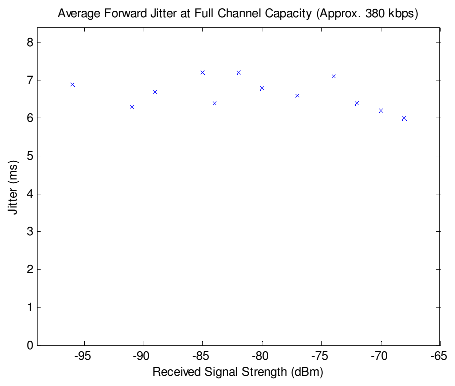
(g) Average Forward Jitter vs. Received Signal Strength (75% Channel Capacity)



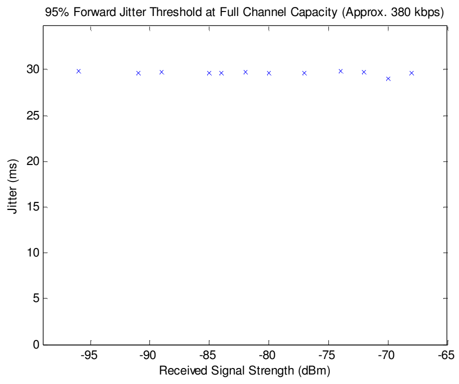
(h) 95% Forward Jitter Threshold vs. Received Signal Strength (75% Channel Capacity)



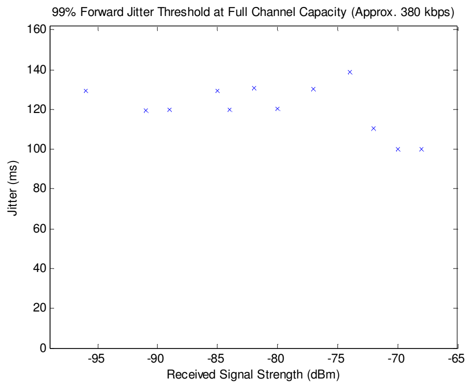
(i) 99% Forward Jitter Threshold vs. Received Signal Strength (75% Channel Capacity)



(j) Average Forward Jitter vs. Received Signal Strength (100% Channel Capacity)



(k) 95% Forward Jitter Threshold vs. Received Signal Strength (100% Channel Capacity)



(l) 99% Forward Jitter Threshold vs. Received Signal Strength (100% Channel Capacity)

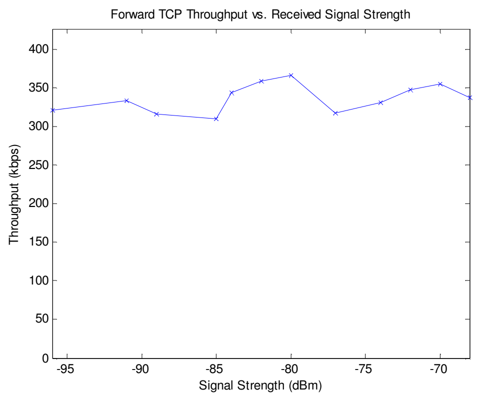
Figure 101: Jitter Characteristics vs. Packet Size at Varying Channel Loads

Table 17: Summary of Jitter Characteristics at Different Traffic Loads

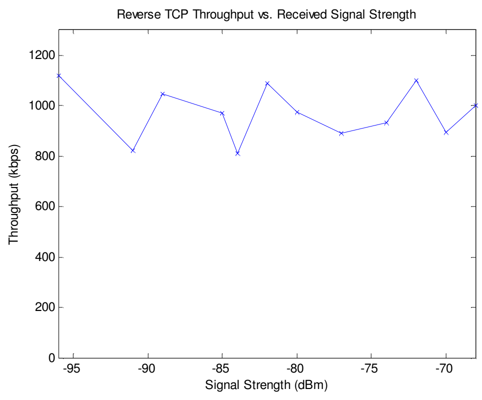
Channel Utilization (%)	Average Jitter (ms)	95% Jitter Threshold (ms)	99% Jitter Threshold (ms)
25	7.9	60.4	147.4
50	8.3	59.7	149.4
75	7.3	39.5	137.3
100	6.7	29.6	120.8

As seen in Figure 101 and Table 17, the average jitter does not vary significantly across the different traffic loads. Instead, the average jitter generally stays between 6 and 9 ms which is less than the 12.9 ms jitter observed during the delay tests. On the other hand, the 95% and 99% jitter thresholds actually decrease as the data rate is increased. One possible reason for this is that the data streams sent over the wireless link used a fixed time interval between each packet. To increase the data rate, the time interval between successive packets needed to be decreased. For example, when sending data at 50% of the channel capacity, 1446 byte packets needed to be sent about every 60 ms to achieve the data rate of 190 kbps. The time interval between packets needed to be decreased to around 30 ms to send data at full channel capacity. As discussed in Section 2.4.8, the time interval between sent packets essentially puts a limit how small (large negative value) the jitter can be. Therefore, if there was a large amount of spreading between two consecutive packets (positive jitter), then there would be a number of negative jitter values equal to the sending interval to compensate for the spreading. This is evident in Figure 98 (h) and Figure 98 (n) where there are a large number of jitter values at -60 ms and -30 ms respectively. This behavior explains why the average jitter as well as the 95% and 99% jitter thresholds appear to decrease has the data transmission rate is increased.

In addition to UDP tests, TCP tests were run using the *iperf* application described in Section 2.5. Figure 102 shows both the forward and reverse TCP throughput measured as a function of received signal strength. For each test, TCP data was sent using iperf for a period of thirty seconds. As expected, the TCP throughput experienced on a 3G network is less than what can be achieved using UDP. Throughout all of the tests, the average forward TCP throughput was 336.8 kbps while the average reverse throughput was 971.4 kbps.



(a) Forward TCP Throughput

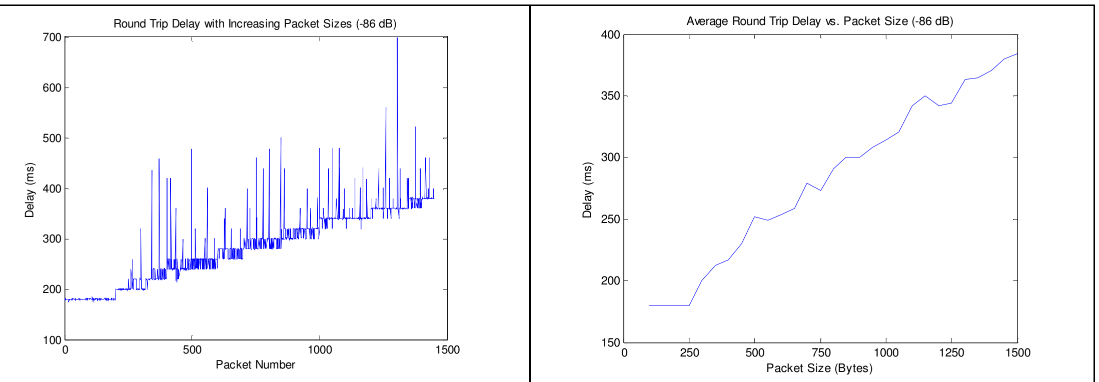


(b) Reverse TCP Throughput

Figure 102: TCP Throughput vs. Received Signal Strength on 3G Network

4.3.3 Latency vs. Packet Size

Twelve tests that measured latency as a function of packet size were conducted at varying signal strengths. During each test, fifty round trip packets were sent at each size interval to measure round trip delay. The first interval used 100 byte packets, and for each successive interval, the packet size was increased by 50 bytes. In total, twenty-nine intervals were used resulting in 1500 byte packets at the final interval. In addition to round trip delays, jitter and packet loss was measured in each direction. Figure 103 shows the complete results from a latency vs. packet size test conducted with a received signal strength of -86 dBm. The “Delay vs. Sample Number” graph shows the individual delays of each round trip packet while the “Delay vs. Packet Size” graph shows the average round trip delay of the fifty packets at each interval. Each of these tests took around six or seven minutes to complete.



(a) Delay vs. Sample Number

(a) Delay vs. Packet Size

Figure 103: Results from a Latency vs. Packet Size Test with a Received Signal Strength of -86 dBm.

After viewing the results from the latency vs. packet size tests, there were no obvious performance differences at the various received signal strengths. The results from each individual data point can be seen in Appendix B. To gain a better insight into the latency behavior of the channel, the results from all thirteen tests were combined and viewed independent of signal strength. Figure 104 shows the average round trip delay times at each interval for all thirteen of the tests. The straight line corresponds to the first-order line of fit of the measured data, for which the equation is:

$$RTD(PS) = 0.154 \bullet PS + 158.1 \text{ [ms]} \tag{4.1}$$

where *RTD* refers to the round trip delay in milliseconds, and *PS* refers to the size of the packet in bytes.

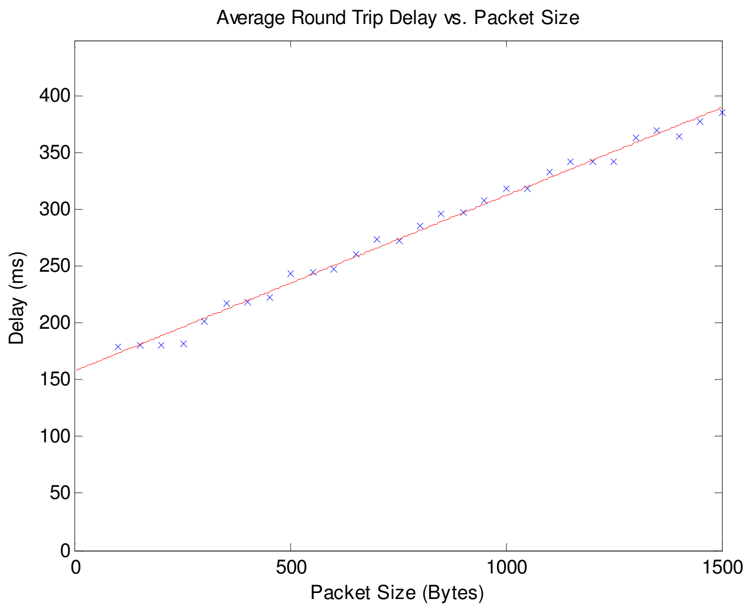


Figure 104: Average Round Trip Delay vs. Packet Size on 3G Network

As with the standard delay tests, one-way delay tests were also conducted as a function of packet size. Figure 105 shows the average measured data as a function of packet size for twelve separate tests run at varying signal strengths. It should be noted that the measured latency data in Figure 105 includes reverse delay through the WPI LAN. An estimation for the true forward delay can be found on the next page. Equation 6 gives the equation for the measured data where *MFD* refers to measured forward delay in milliseconds and *PS* corresponds to the size of the packet in bytes.

$$MFD(PS) = 0.129 \bullet PS + 89.0 \text{ [ms]} \tag{4.2}$$

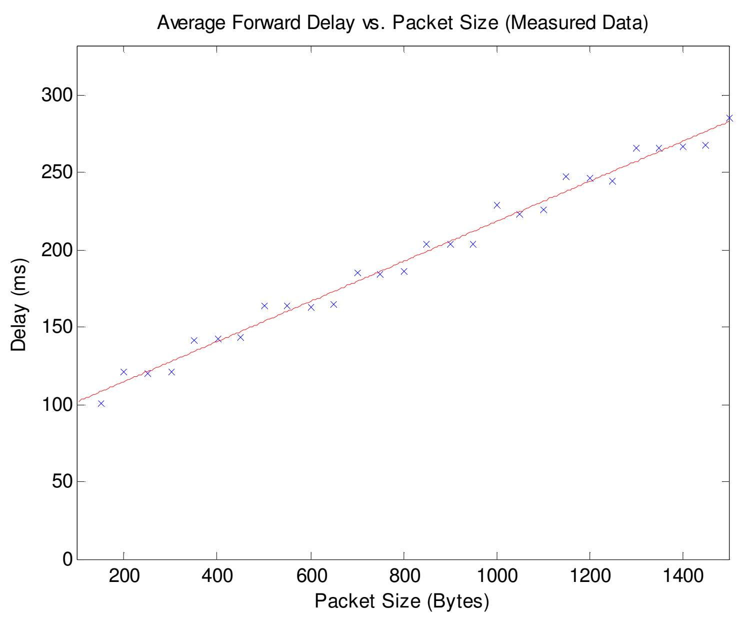


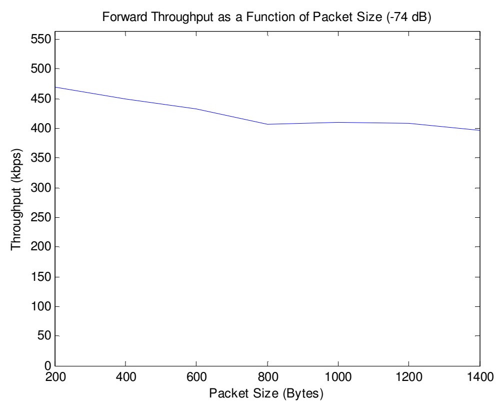
Figure 105: Average Forward Delay vs. Packet Size on 3 G Network

This measured data can be used to derive expressions to estimate both the forward and reverse latency as a function of packet size. A number of packet size vs. latency tests were run over the WPI network, and there seemed to be no difference in round trip latency from packets between 100 and 1500 bytes. The average round trip latency for these tests remained around 4 ms as was the case with the standard delay tests run over the WPI LAN. If the reverse link through the WPI LAN is approximated at 2 ms, then 4.3 can be used to estimate the forward delay as a function of packet size on the 3G network. The reverse delay was derived by subtracting the forward delay expression from the round trip delay expression given in 4.1. This results in 4.4 which can be used to approximate the reverse delay as a function of packet size. By comparing the slopes of both of these expressions, it is evident that increasing the packet size has a greater impact on the uplink than it does on the downlink.

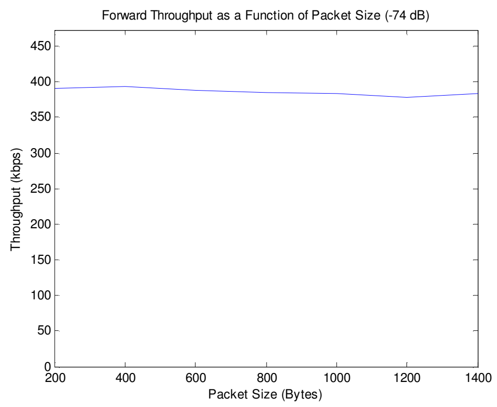
$$FD(PS) = 0.129 \bullet PS + 87.0 \text{ [ms]} \quad (4.3)$$
$$RD(PS) = 0.025 \bullet PS + 71.1 \text{ [ms]} \quad (4.4)$$

4.3.4 Throughput vs. Packet Size

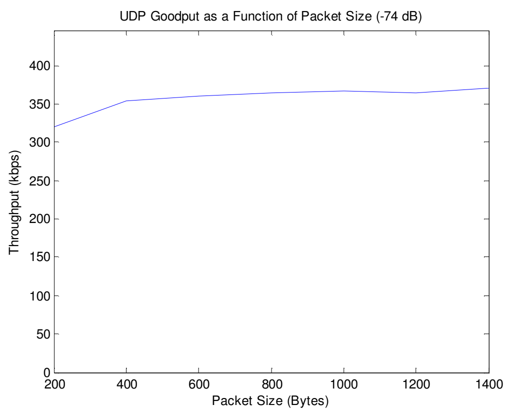
The last set of tests that were conducted on the 3G cellular network measured the throughput of the uplink as a function of packet size. As with each of the other tests, twelve total tests were conducted, and there seemed to be little to no performance difference across the various received signal strengths. For each test, 1000 packets were sent at each size interval, and the throughput was measured at both the client and server nodes. There were seven total packet size intervals, ranging from 200 to 1400 bytes in increments of 200 bytes. The sending throughput, receiving throughput, UDP goodput, packet loss and jitter were measured during each test. Figure 106 shows a complete set of results from a throughput vs. packet size test conducted with a received signal strength of -74 dBm. The remaining data points can be viewed in Appendix B.



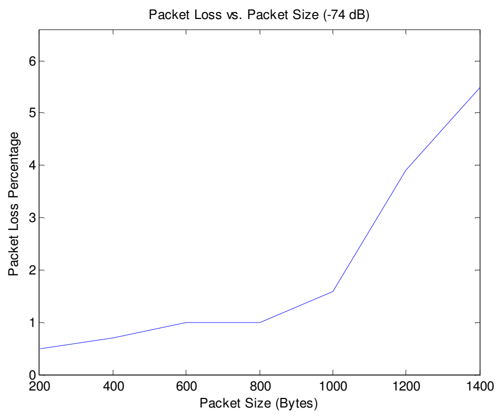
(a) Throughput vs. Packet Size (Client Report)



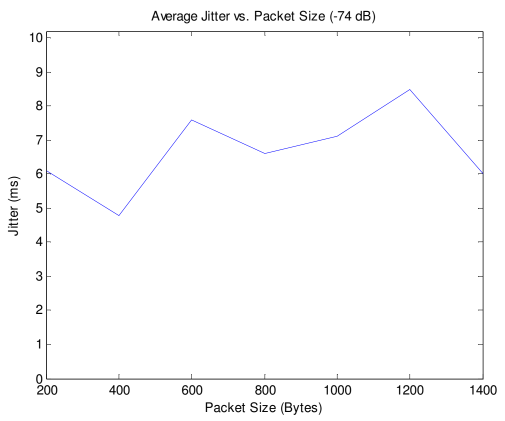
(b) Throughput vs. Packet Size (Server Report)



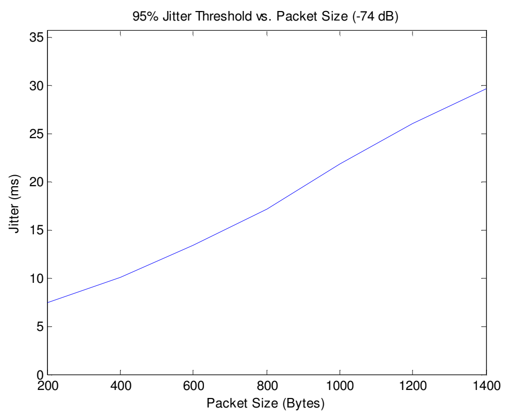
(c) UDP Goodput vs. Packet Size (Server Report)



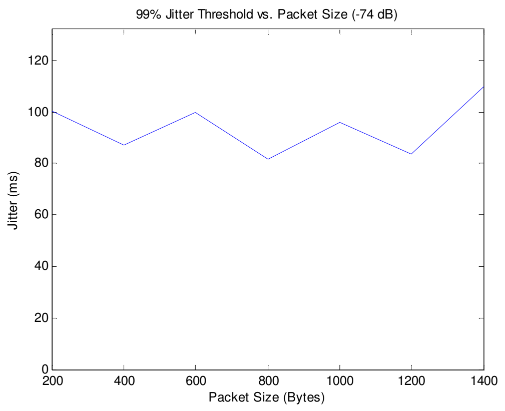
(d) Packet Loss vs Packet Size



(e) Average Forward Jitter vs. Packet Size (100% Channel Load)



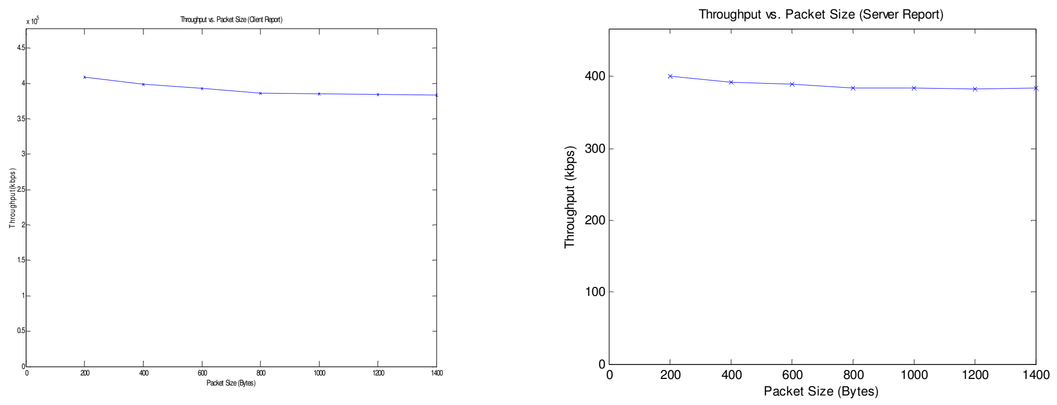
(f) 95% Forward Jitter Threshold vs. Packet Size (100% Channel Load)



(g) 99% Forward Jitter Threshold vs. Packet Size (100% Channel Load)

Figure 106: Results from 3G Throughput vs. Packet Size Test with a Received Signal Strength of -74 dBm

Like the other performance tests, these results showed no significant differences among the various signal strengths. Therefore, the test results were viewed independent of received signal strength to get a better idea of how throughput is affected by packet size on a 3G network. Figure 107 shows the average forward throughput vs. packet size over all twelve of the tests measured by both the client (sender) and server (receiver). Both of these curves show slightly higher raw data rates using smaller sized packets; however, the additional overhead necessary to send smaller packets negates this slight increase in data rate.



(a) Client Measurement *(b) Server Measurement*

Figure 107: Average Forward Throughput vs. Packet Size on 3G Network

Figure 108 shows the average UDP goodput or the useful data without the 46 bytes of UDP and IP header information. This curve was produced by measurements made at the server node. The results show that although smaller packets may slightly increase the raw data rate on the channel, it is still beneficial to use larger packets because the additional overhead required when sending smaller packets reduces the amount of useful data that can be sent. For example, when sending 200 byte UDP datagrams, 18.7 % (46/246) of the data is overhead. On the other hand, 1400 byte datagrams only require 3.2 % (46/1446) overhead resulting in more useful data sent over time.

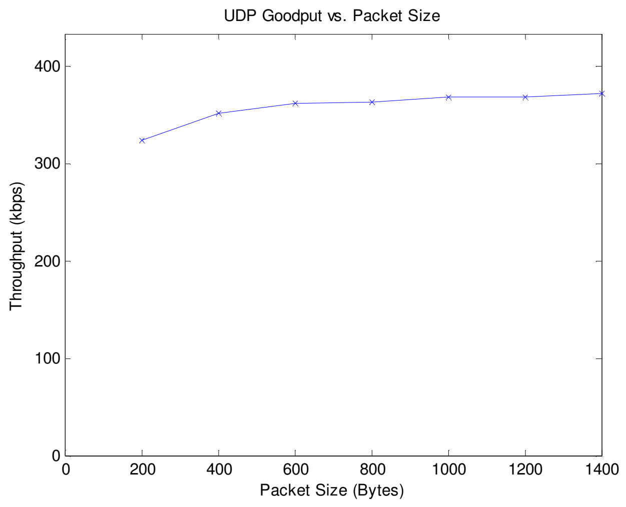


Figure 108: Average Forward UDP Goodput vs. Packet Size on 3G Network (Server Measurement)

Figure 109 shows the average packet loss percentage on the uplink as a function of packet size. Throughout all of the tests, the packet loss was generally less than 2% for packets less than or equal to 1000 bytes. Packet loss then seemed to jump between 3% and 4% for packet sizes of 1200 and 1400 bytes. These results are consistent with the results from the standard throughput tests with packet loss near 4% for 1400 byte packets at channel capacity.

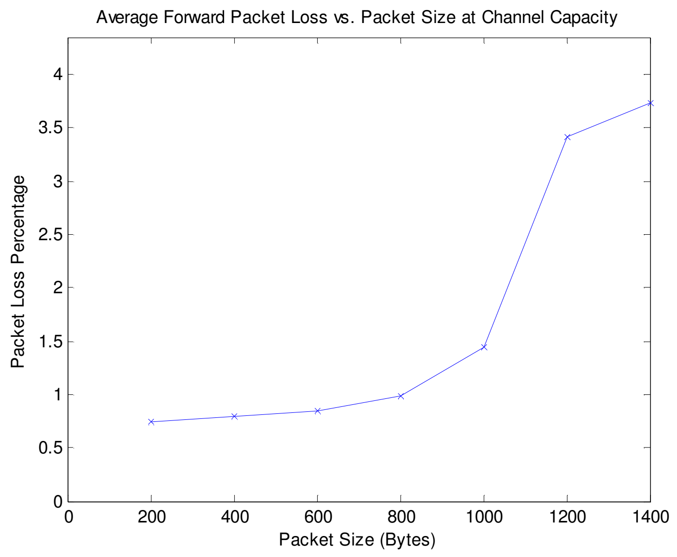
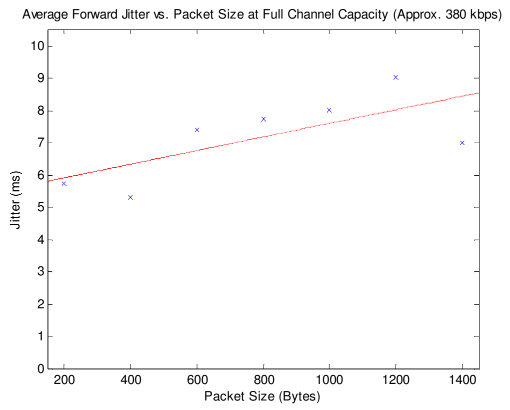


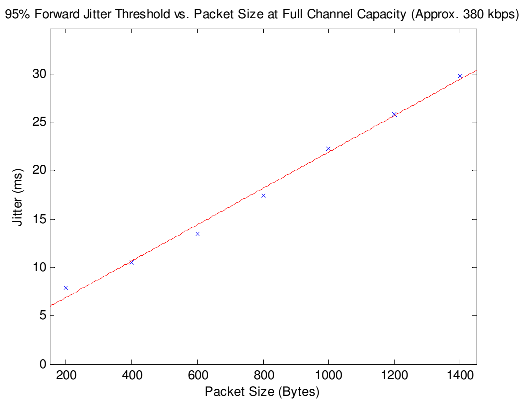
Figure 109: Forward Packet Loss vs. Packet Size on 3G Network at Channel Capacity

The last metric that was measured during the throughput vs. packet size tests was the forward jitter. Figure 110 shows the plots of the average forward jitter, the 95% threshold, and the 99% threshold as a function of packet size. The equations for the first order approximations of the data shown in Figure 110 are shown below. These results suggest the jitter in the forward direction worsens as the packet size is increased.

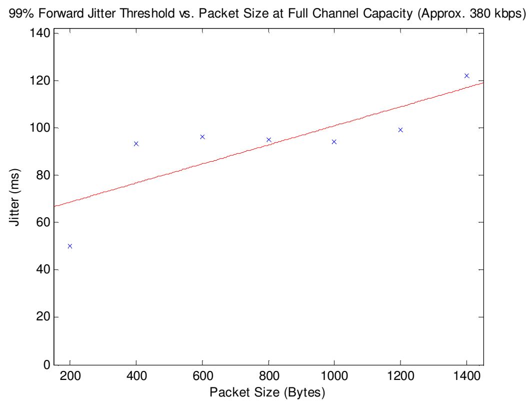
$$FJ(PS) = 0.002 \bullet PS + 5.5 \text{ [ms]} \qquad (4.5)$$
$$FJ95(PS) = 0.018 \bullet PS + 3.1 \text{ [ms]} \qquad (4.6)$$
$$FJ99(PS) = 0.04 \bullet PS + 60.7 \text{ [ms]} \qquad (4.6)$$



(a) Average Forward Jitter vs. Packet Size



(b) 95% Jitter Threshold vs. Packet Size



(c) 99% Jitter Threshold vs. Packet Size

Figure 110: Forward Jitter Behavior Using Different Packet Sizes at Channel Capacity

4.3.5 Mobile Tests

Because many telemedicine applications do not take place in a static location, two sets of performance tests were run in a mobile environment. These tests were intended to observe any performance differences between static positions as opposed to a moving system. The mobile tests took place in a moving car where the received signal strength was constantly changing, and care was taken to ensure the vehicle was always within the 3G coverage area.

The results from the mobile tests were consistent with the results from static tests. Table 18 shows the main results from the mobile tests. Complete results can be seen in Appendix B. These results show that there are no major performance differences on the network between stationary and mobile systems.

Table 18: Summary of Test Results from Mobile Tests Run on 3G Network

Average Round Trip Delay	387.4 ms
Average Forward Throughput	378.9 kbps
Average Reverse Throughput	1624 kbps
Forward Packet Loss at Channel Capacity	3.6 %
Forward Jitter at Channel Capacity (1400 byte packets)	7.6 ms

4.3.6 Conclusion

After viewing all of the results from the performance testing on the 3G network, it is apparent that the performance characteristics of the channel are not significantly affected by the strength of the received signal. For telemedicine applications, this is a positive characteristic because users do not need to worry about varying network performance depending on location or mobility. Table 19 summarizes the test results discussed in this section.

Table 19: Summary of 3G Testing Results

Delay	Mean Round Trip Delay	391.7 ms
	Mean Forward Delay	281.5 ms
	Mean Reverse Delay	110.2 ms
	Mean Forward Packet Loss	0.41 %
	Mean Reverse Packet Loss	0.44 %
	Mean Forward Jitter	12.9 ms
	95% Forward Jitter Threshold	55.7 ms
	99% Forward Jitter Threshold	184.1 ms
	Mean Reverse Jitter	8.2 ms
	95% Reverse Jitter Threshold	20.2 ms
	99% Reverse Jitter Threshold	29.3 ms
Throughput	Mean Forward Throughput	380.3 kbps
	Mean Reverse Throughput	1361 kbps
	Mean Forward Packet Loss	4.0 %
	Mean Forward Jitter	6.7 ms
	95% Forward Jitter Threshold (100% Channel Capacity)	29.6 ms
	99% Forward Jitter Threshold (100% Channel Capacity)	120.8 ms
Delay vs. Packet Size	Mean Round Trip Delay	$RTD(PS) = 0.154 \bullet PS + 158.1$ [ms]
	Mean Forward Delay	$FD(PS) = 0.129 \bullet PS + 87.0$ [ms]
	Mean Reverse Delay	$RD(PS) = 0.025 \bullet PS + 71.1$ [ms]
Throughput vs. Packet Size	Mean Forward Jitter	$FJ(PS) = 0.002 \bullet PS + 5.5$ [ms]
	95% Forward Jitter Threshold (100% Channel Capacity)	$FJ95(PS) = 0.018 \bullet PS + 3.1$ [ms]
	99% Forward Jitter Threshold (100% Channel Capacity)	$FJ99(PS) = 0.04 \bullet PS + 60.7$ [ms]

5 Wireless Communication via Satellite Systems

The last wireless communication option that was tested as part of this project was a satellite network owned and operated by a company called *Inmarsat*. Testing was conducted on Inmarsat's BGAN network, which consists of three geostationary earth orbit (GEO) satellites, as well as an extensive terrestrial network. Like most other satellite systems, the BGAN network is characterized by high latencies (close to 1 second each way) and medium data rates (maximum of 492 kbps). This section will give explain Inmarsat's BGAN network as well as outline a protocol used to test the performance of the network. It will go on to present the results and analysis of the performance testing.

5.1 Background

Inmarsat's BGAN network consists of three geostationary earth orbit (GEO) satellites to provide near global coverage. GEO satellites are located directly above the Earth's equator (0˚ Latitude), and appear motionless in the sky as the rotational period of the satellites are equal to that of the Earth's. For a satellite to be geostationary, it must be at an altitude equal to the altitude of geosynchronous orbit which is 35786 km. Due to the high altitude, only a small number of satellites are needed to cover the entire planet. Other satellite systems such as low earth orbit (LEO) and medium earth orbit (MEO) satellites need many more satellites for global coverage. Also, because these satellites do not appear motionless relative to the Earth's surface, handoff protocols must be implemented to deal with coverage handoffs amongst the satellite network. This is not necessary in GEO systems. One disadvantage of geostationary earth orbit satellites is the long latencies encountered when exchanging data over the wireless interface. It takes approximately 125 ms for a signal to propagate from the earth's surface to the satellite in orbit. Considering a round trip packet must make four space hops, a minimum of around one-half second is necessary just for propagation delay. Figure 111 shows the coverage map of the BGAN network as of 2008.

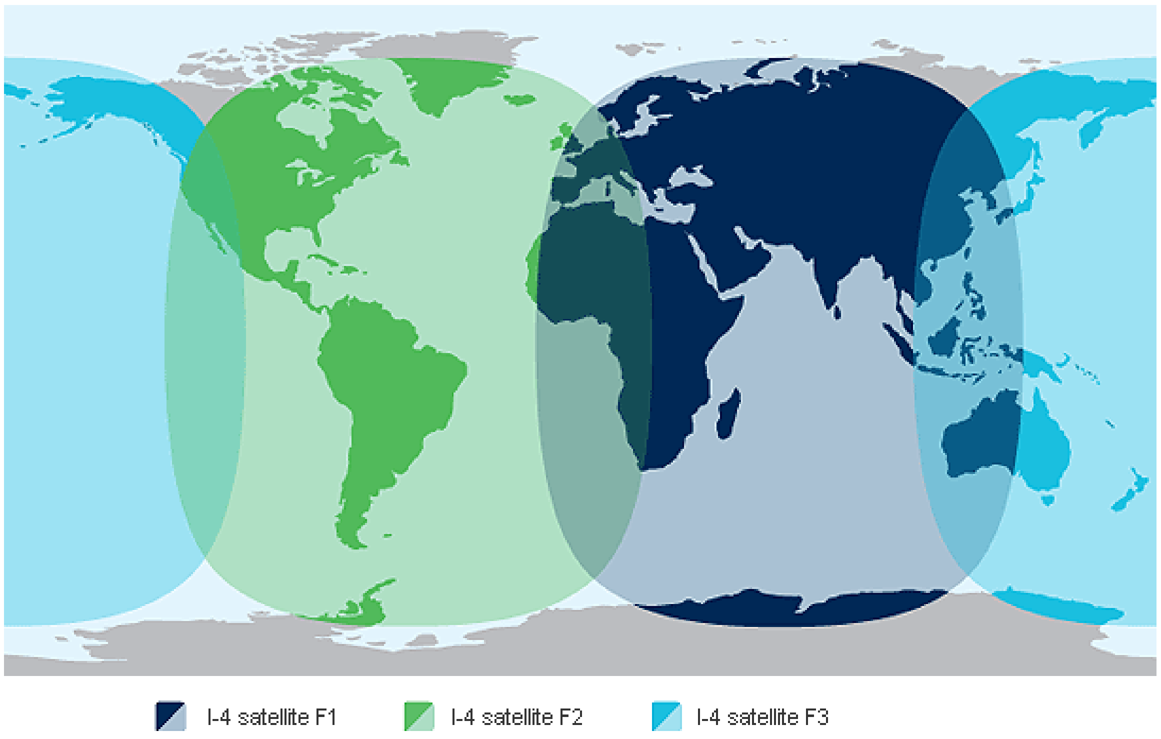


Figure 111: BGAN Coverage Map

To interface with Inmarsat's BGAN network, a BGAN compatible terminal is necessary. Before a user can connect with the network, the BGAN terminal must be properly aligned to obtain the best possible signal strength. Once connected to the network, Inmarsat offers two separate classes of service for IP data. The first is a standard IP connection that sends IP data on a "best effort" basis. The network supports a maximum bandwidth of 492 kbps; however, typical data rates depend on a number of other factors such as signal strength, inter-band interference and network utilization. When using standard IP data connections, users are billed relative to the amount of data that they send and receive over the network. Inmarsat customers can expect to pay around $7 per MB of data either sent or received. All of the performance tests conducted over the BGAN network used a standard IP connection.

Inmarsat also offers a streaming IP service where there are Quality of Service (QoS) guarantees for time-sensitive traffic. This means that throughout the duration of the connection, the network guarantees a minimum configurable data rate. Inmarsat offers streaming IP services at 32, 64, 128 and 256 kbps on both the uplink (forward) and downlink (reverse). Users of the streaming IP service are billed according to how long the connection remains open. A typical cost for a one-minute streaming session at 256 kbps is approximately $20.

The last type of service offered by Inmarsat is a circuit switched network which can be used for standard telephone and ISDN services. Circuit switched services are billed on a per minute basis for the duration of the connection. The basic configuration of Inmarsat's BGAN network can be seen in Figure 112.

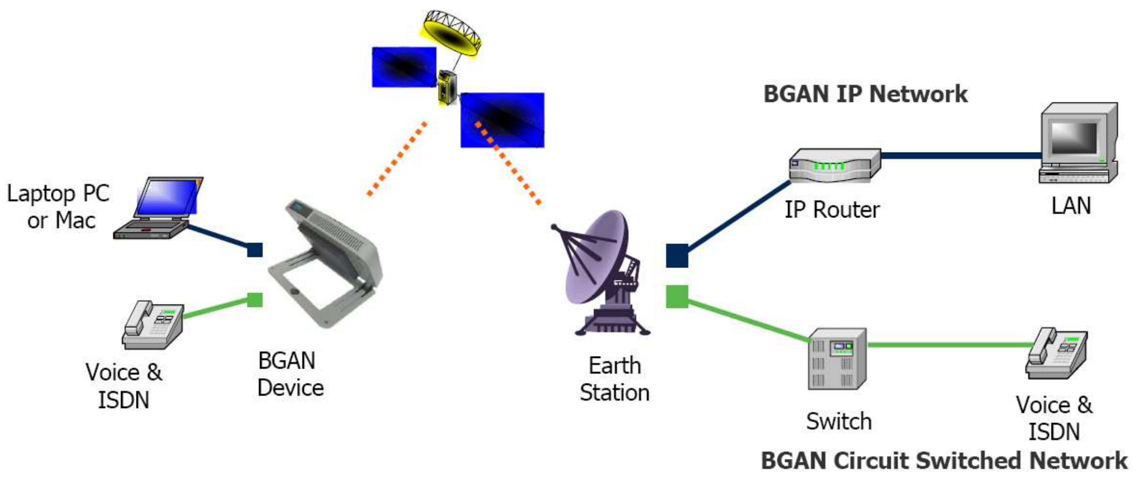


Figure 112: BGAN Network Configuration

Because Inmarsat does not release its proprietary information to the public, the specific implementation details of the air interface between the BGAN terminal and the satellite are not available. It is known that data is modulated using 16 QAM in both the forward and reverse directions while turbo coding is used for error correction. The coding techniques implemented by Inmarsat allow for data rates as high as 492 kbps. One of the main disadvantages of BGAN is the high network latency. Typical delays of between 800 ms and 1100 ms can be expected in each direction which can cause problems for certain types of network services.

5.2 Testing Protocol

In order to perform testing on Inmarsat's BGAN network, a BGAN terminal was rented from a company named *Outfitter Satellite*. The BGAN terminal that was rented was the *Hughes 9201,* which is the top of the line BGAN unit currently available. While not all BGAN terminals support the maximum data rate achievable on the BGAN network, the Hughes 9201 supports data rates up to 492 kbps on both the uplink and the downlink. The Hughes 9201 terminal is 27.5 cm x 37.5 cm x 5.0 cm in size and interfaces with a laptop via an Ethernet, USB or 802.11b connection. Figure 113 shows the Hughes 9201 BGAN terminal used for testing in this project.

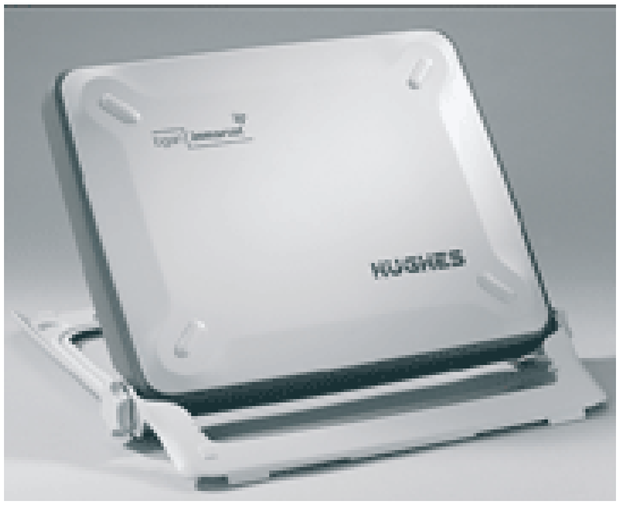
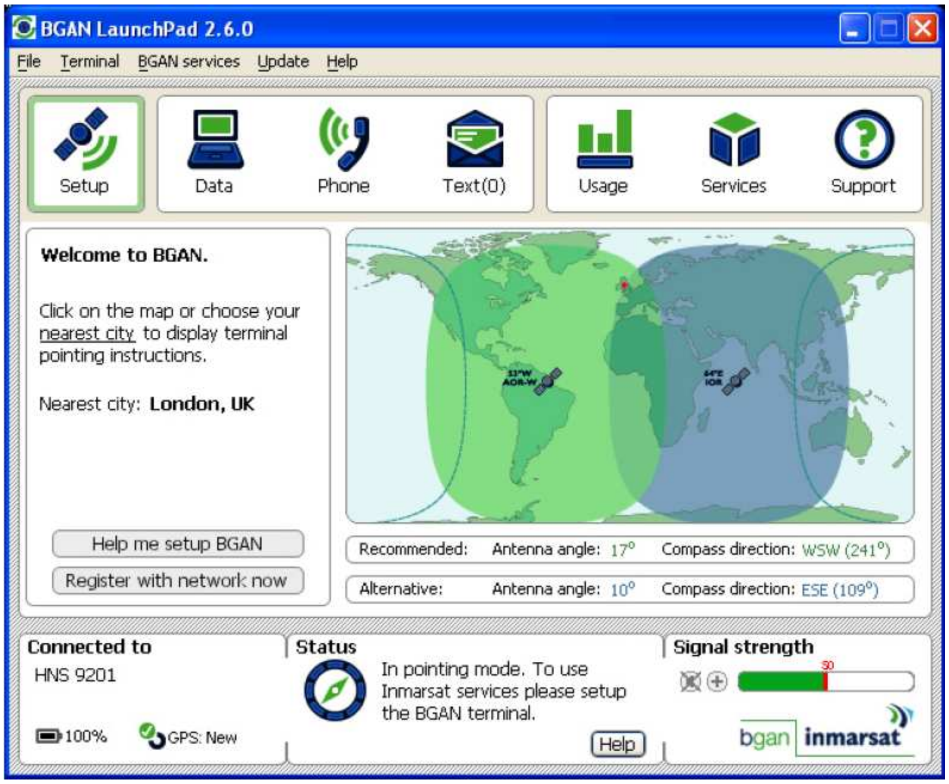


Figure 113: Hughes 9201 BGAN Terminal

Before a BGAN connection can be set up, the terminal must be properly aligned with a satellite to achieve the best possible signal quality. Inmarsat provides a software package called *BGAN LaunchPad* to align the satellite terminal as well as create and manage network connections. Additionally, the BGAN Lauchpad constantly monitors the received signal strength. Figure 114 shows the BGAN Launchpad GUI and the signal strength indicator can be seen on the bottom right corner of the GUI.



Welcome to BGAN.

Click on the map or choose your nearest city to display terminal pointing instructions.

Nearest city: London, UK

Recommended: Antenna angle: 17° Compass direction: WSW (241°)

Alternative: Antenna angle: 10° Compass direction: ESE (109°)

Connected to: HNS 9201

100% GPS: New

Status: In pointing mode. To use Inmarsat services please setup the BGAN terminal.

Signal strength: 50

Figure 114: BGAN LaunchPad GUI

To conduct performance testing over the BGAN network, a laptop was connected to the internet using a Hughes 9201 BGAN terminal. The laptop interfaced with the satellite terminal via a USB connection, and the BGAN terminal was configured to use Inmarsat's standard IP service, which operates on a "best effort" basis. The laptop was used to run the client side code of the channel measurement application detailed in Section 2.4. A desktop computer located in the Atwater Kent Laboratories building of WPI was used to run the server side application of the channel measurement toolbox. The desktop had a standard wired Ethernet connection to WPI's network. In order to make a connection to a satellite, the laptop had to be outdoors to obtain a line of sight with the satellite. The laptop (client) was brought to various outdoor locations to try to obtain different signal strengths from the satellite. Once a connection was made with the BGAN network, channel performance tests were performed according to the protocols outlined in Section 2.4. Figure 115 shows the general test setup for performance tests run over the BGAN network.

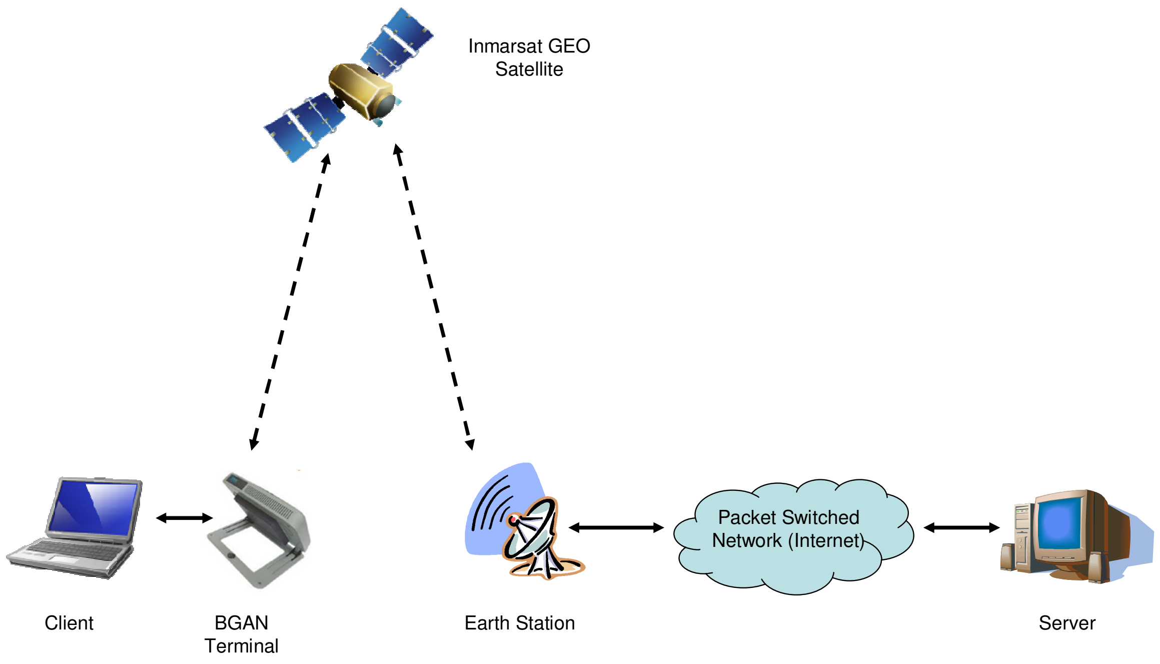


Figure 115: BGAN Test Setup

Because BGAN users are billed according to how much data is used, the performance tests were slightly altered relative to the 3G tests in attempts to reduce the overall amount of data used. Table 20 shows the test parameters used for each of the channel measurement tests presented in Section 2.4. In total, five sets of performance tests were conducted, each with the client in a different location. For the first three performance tests, the client was stationed in various locations in Worcester, MA. In an effort to run tests with a different signal strength than those encountered in Worcester, MA, additional tests were conducted in Bedford, NH. The following section contains the results and analysis of the performance test run over the BGAN satellite network.

Table 20: Testing Parameters for BGAN Satellite Measurements

Test	Test Length	Packet Size(s) (Bytes)
Delay	• 500 Round Trip Packets	• 1500
Delay vs. Packet Size	• 50 Round Trip Packets at each interval • 14 total intervals	• 100 to 1500 in intervals of 100 bytes
Throughput	• 30 seconds in forward direction	• 1400
Throughput vs. Packet Size	• 500 packets at each interval • 7 intervals	• 200 to 1400 in intervals of 200 bytes

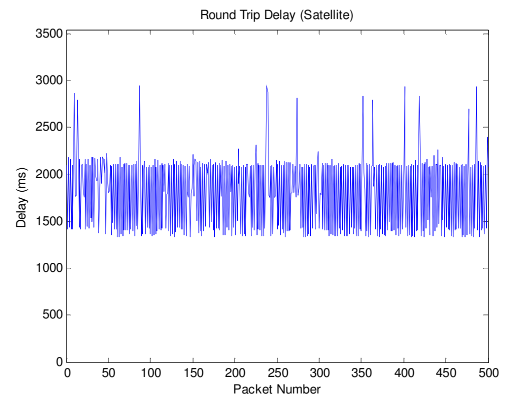
5.3 Testing Results and Analysis

Due to constraints on the amount of data that was available for testing, five sets of performance tests were conducted over the BGAN satellite network. Initially, the plan was to test the network performance at a number of different signal strengths; however once testing began, it was apparent that the signal strength was more or less consistent in our geographic region (MA and NH). The first three tests were conducted in Worcester, MA at various locations around the campus of WPI. The signal-to-noise ratios experienced during these tests were 51, 52 and 52 dB. In an effort to test diverse signal strengths, the final two tests were conducted in Bedford, NH; however, the same approximate signal-to-noise ratio was present in this location as well. The SNRs experienced during the two tests in NH were 54 and 54 dB. All five of the tests were conducted on different days in different locations.

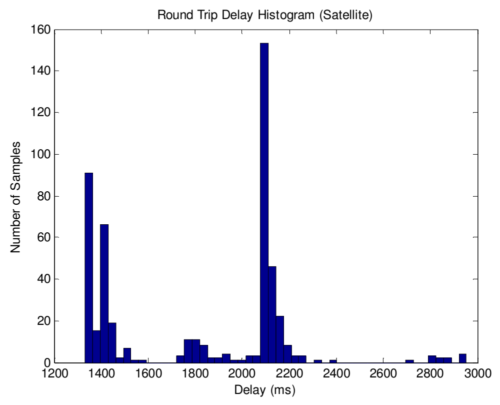
After viewing the results from the performance tests, there were no glaring differences throughout this small range of signal-to-noise ratios. Similar to the 3G tests, the results were fairly consistent from test to test, and the SNR did not seem to affect the performance of the system. For this reason, the data presented in this section will be viewed independent of signal strength. Instead, the data can be looked at as a whole, and for signal-to-noise ratios of 50 to 55 dB, typical BGAN performance can be concluded.

5.3.1 Latency

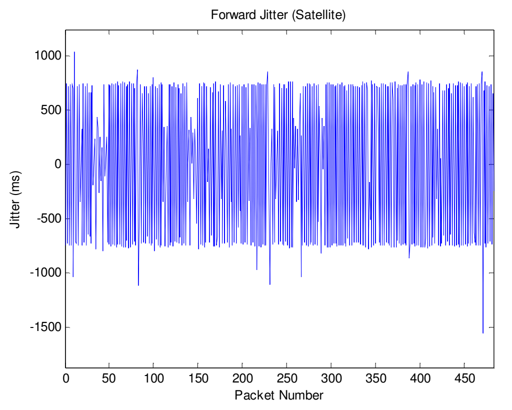
The most notable characteristic of the satellite network was the extremely high latency relative to most other types of networks. Round trip delay times over the BGAN network were routinely between 1.5 and 2 seconds. Figure 116 shows the complete results from a latency tests run on the BGAN network with a signal-to-noise ratio of 51 dB. The results from the remaining satellite delay tests, all of which are similar to those in Figure 116, can be found in Appendix C.



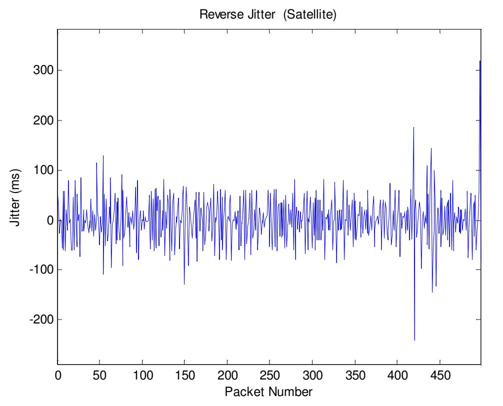
(a) Round Trip Delay vs. Time



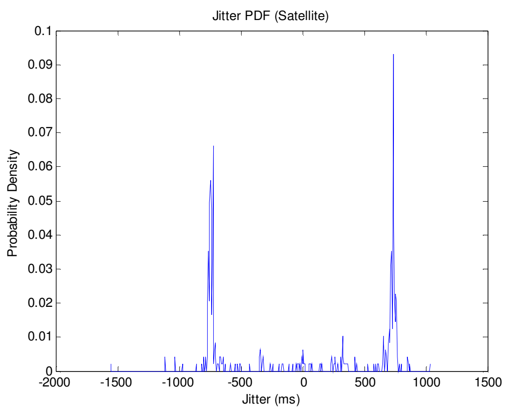
(b) Round Trip Delay Histogram



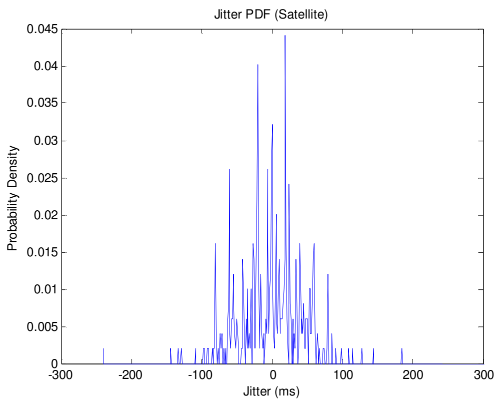
(c) Forward Jitter vs. Time



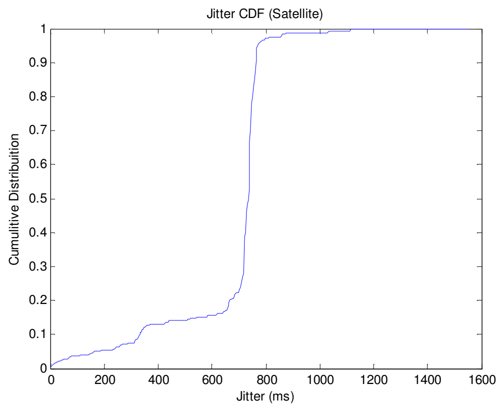
(d) Reverse Jitter vs. Time



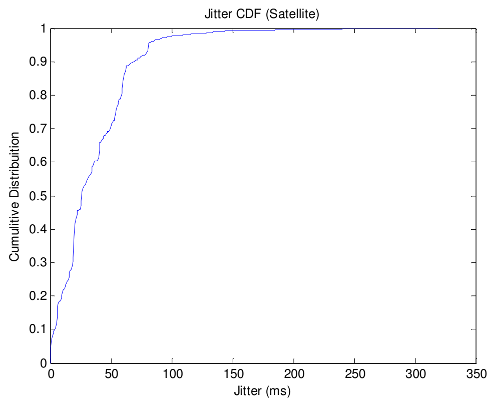
(e) Forward Jitter PDF



(f) Reverse Jitter PDF



(g) Forward Jitter CDF



(h) Reverse Jitter CDF

Max Delay = 2953.9 ms
Min Delay = 1332.7 ms
Mean Delay = 1814.0 ms
RT Standard Deviation = 381.5 ms
Packet Loss (Forward) = 15/500
Packet Loss (Reverse) = 1/500
Mean Forward Jitter = 665.6 ms
Forward Jitter (95%) = 775.8 ms
Forward Jitter (99%) = 1036.5 ms
Mean Reverse Jitter = 35.1 ms
Reverse Jitter (95%) = 81.4 ms
Reverse Jitter (99%) = 144.9 ms

Figure 116: Results from a Satellite Latency Test with a SNR of 51 dB

Figure 116(a) shows the round trip delay of each packet sent during the test while Figure 116(b) shows a histogram of the round trip delay data. The jitter behavior exhibited during the delay tests can be seen in Figure 116(c) through (h). These figures show the jitter vs. time, the PDF and the CDF of both the forward and reverse links.

This data shows that the satellite link behaves rather oddly during the delay tests. The round trip delays seem to alternate between 1.4 seconds and 2.1 seconds throughout the duration of the test. After looking at the jitter data, it appears as though this behavior can be attributed to the uplink where the jitter has two spikes at approximately +/- 700 ms. Although the cause for this behavior is not known, it was consistent throughout all of the delay tests run on the BGAN network. Also, as will be presented in the upcoming sections, the forward jitter is drastically reduced in circumstances where more than a single packet is being sent at a time. This is good news for streaming media applications as 700 ms is quite excessive for jitter.

As previously mentioned, the data from the other four delay tests was similar to that of Figure 116. In addition to the standard round trip delay tests, one-way delay tests were conducted as described in Section 2.6. During the one-way delay tests, the uplink experienced the same jitter behavior that was experienced during the round trip tests. This caused the average forward delay to be greater than that of the reverse link. The average forward delay for the satellite network was 1120 ms. The resulting reverse delay, which was approximated by subtracting the forward delay from the round trip delay, was 718 ms. Table 21 shows the complete results from the delay tests on the satellite network along with the average values for all five of the tests. Again, the results did not seem to be impacted by the SNR at the BGAN terminal.

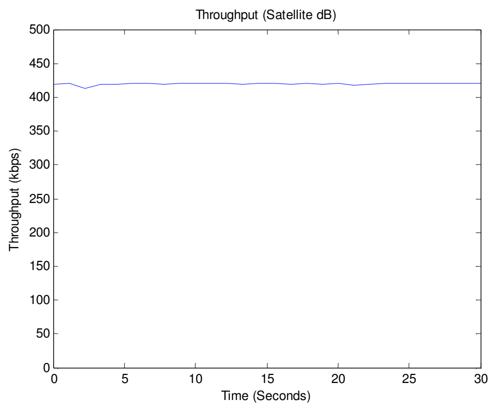
Table 21: Complete Results from Delay Tests on BGAN Satellite Network

	Test Number (SNR dB)					**AVG**
	1 (51 dB)	**2 (52 dB)**	**3 (52 dB)**	**4 (54 dB)**	**5 (54 dB)**	
Average Round Trip Delay (ms)	1814	1873	1840	1819	1848	1838.6
Average Forward Delay (Measured) (ms)	1070	1174	1132	1104	1121	1120.3
Average Reverse Delay (Derived) (ms)	745	698	707	714	727	718.3
Packet Loss Forward (%)	3	0	0.4	0.4	1	0.96
Packet Loss Reverse (%)	0.2	0.4	0	0.4	0.6	0.32

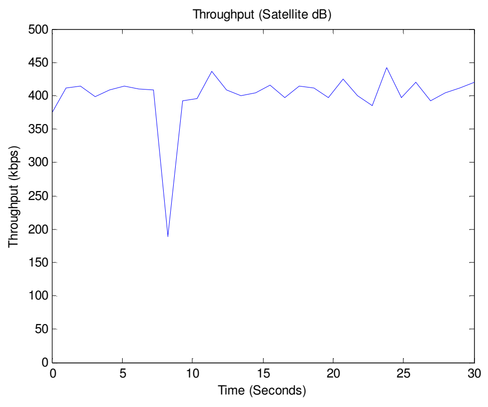
Forward Jitter (ms)	**Average**	665	686	661	700	638	670.14
	95% Threshold	776	791	787	796	777	785.4
	99% Threshold	1037	1750	2049	2051	908	1558.9
Reverse Jitter (ms)	**Average**	35	51	44	41	54	44.96
	95% Threshold	81	130	98	63	110	96.42
	99% Threshold	145	226	162	98	194	164.68

5.3.2 Throughput

Like the delay tests, the throughput tests did not experience clear differences at different signal to noise ratios. The results from these tests showed that the throughput on the uplink of the satellite network was actually higher than that for the 3G cellular network. The overall average forward throughput on the uplink was 407.2 kbps throughout all five of the tests. Figure 117 shows the results from a throughput test run over the BGAN network with a signal-to-noise ratio of 54 dB. The results from the remaining four tests can be found in Appendix C. It should be noted that the reverse throughput was not measured during these tests in an effort to reduce the amount of data necessary for each test.

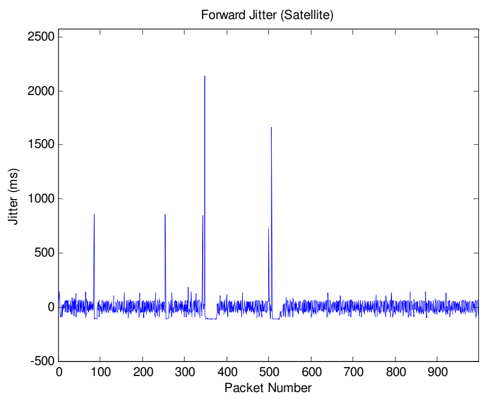


(a) Forward Throughput (Client Report)

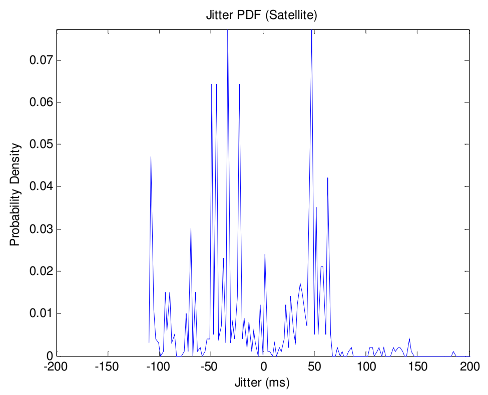


(b) Forward Throughput (Server Report)

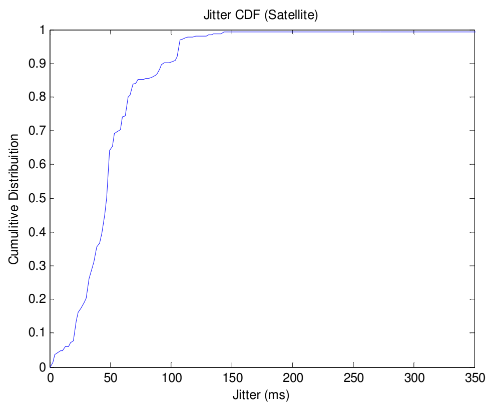
Max BW = 442.6 kbps
Min BW = 190.0 kbps
Mean BW = 400.2 kbps
Packets Sent = 1059
Packets Received = 1030
Drop Percentage = 2.74%



(c) Forward Jitter (25% Channel Load)

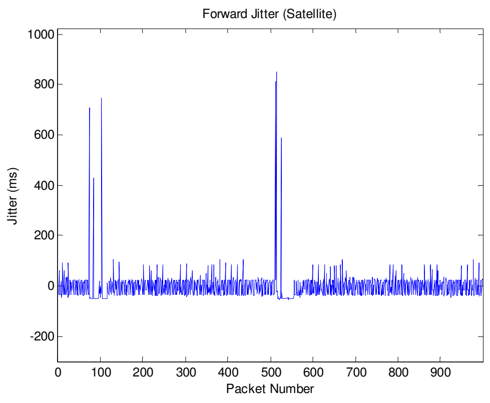


(d) Forward Jitter PDF (25% Channel Load)

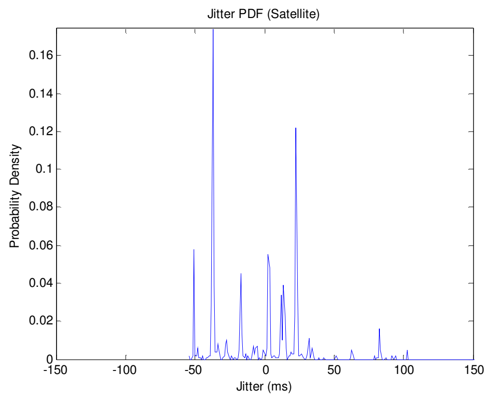


(e) Forward Jitter CDF (25% Channel Load)

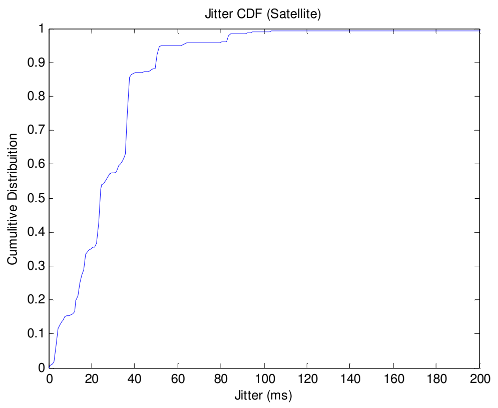
Throughput = 101.4 kbps
Mean Forward Jitter = 56.8 ms
Forward Jitter (95%) = 109.3 ms
Forward Jitter (99%) = 145.8 ms



(f) Forward Jitter (50% Channel Load)

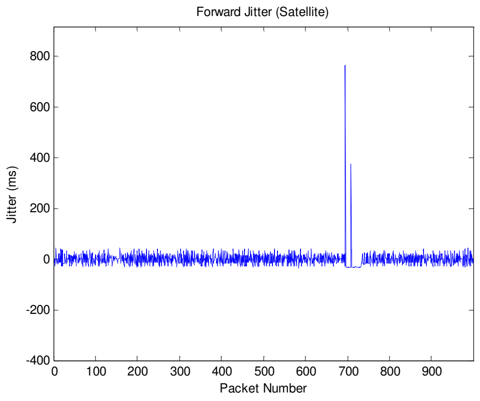


(g) Forward Jitter PDF (50% Channel Load)

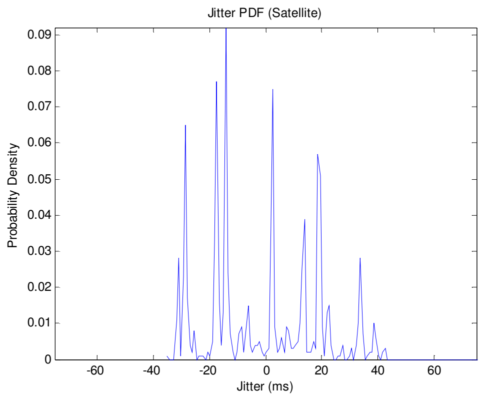


(h) Forward Jitter CDF (50% Channel Load)

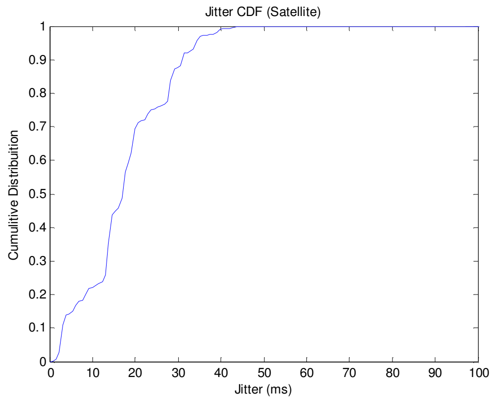
Throughput = 200.3 kbps
Mean Forward Jitter = 31.3 ms
Forward Jitter (95%) = 63.2 ms
Forward Jitter (99%) = 104.1 ms



(i) Forward Jitter (75% Channel Load)

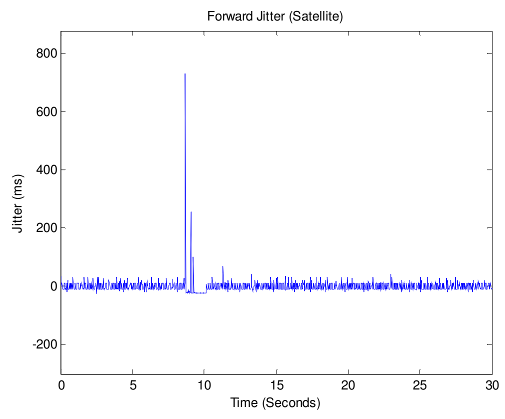


(j) Forward Jitter PDF (75% Channel Load)

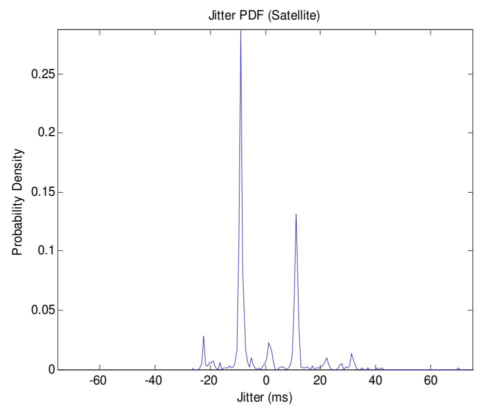


(k) Forward Jitter CDF (75% Channel Load)

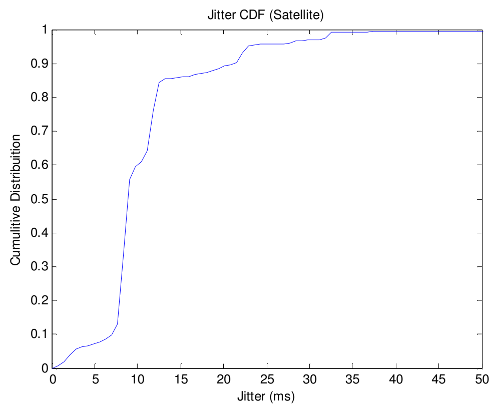
Throughput = 302.1 kbps
Mean Forward Jitter = 18.5 ms
Forward Jitter (95%) = 35.2 ms
Forward Jitter (99%) = 41.3 ms



(l) Forward Jitter (100% Channel Load)



(m) Forward Jitter PDF (100% Channel Load)



(n) Forward Jitter CDF (100% Channel Load)

Throughput = 400.2 kbps
Mean Forward Jitter = 12.3 ms
Forward Jitter (95%) = 23.6 ms
Forward Jitter (99%) = 33.7 ms

Figure 117: Results from a Satellite Throughput Test with a SNR of 54 dB

Figure 117(a) shows the sending throughput measured from the client system connected via the satellite terminal. Figure 117(b) shows the throughput measured at the receiver as a function of time. It is apparent from these figures that the capacity of the satellite link varies over time because as data is sent at a constant rate, the data rate at the receiver tends to fluctuate. There is also a fairly large drop in the receiving bandwidth which is evident in Figure 117(b). This behavior was not uncommon during the throughput tests and could pose a problem for real-time video applications if the bandwidth routinely falls below the minimum data rate necessary for the video bit stream. The remaining plots ((c) through (n)) in Figure 117 display the forward jitter vs. time, the jitter PDF and the jitter CDF for channel capacities of 25%, 50%, 75% and 100%. All of the jitter characteristics including average jitter, 95% threshold and 99% threshold seem to improve as the sending data rate is increased.

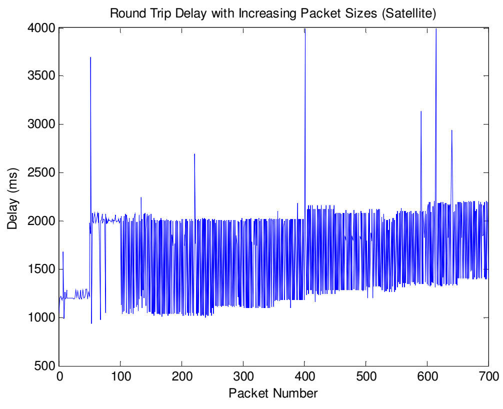
In addition to UDP tests, the TCP throughput was measured in each direction. Iperf was used to transmit data using TCP for thirty seconds in each direction and the resulting throughput was measured and recorded. Due to the high latency on the link, Windows XP's implementation of TCP would not be sufficient in providing acceptable TCP performance. To improve TCP data rates, Inmarsat's *TCP Accelerator* software was used which optimized Window's TCP parameters for a high latency wireless link. TCP Accelerator increases the maximum TCP window size, implements delay based congestion control, and employs a fast start algorithm that is useful in exchanging small amounts of data over a high latency link. Table 22 shows the complete results from the throughput tests run over the BGAN network. Also included are the averaged results over all five of the tests.

Table 22: Complete Results from Throughput Tests on BGAN Satellite Network

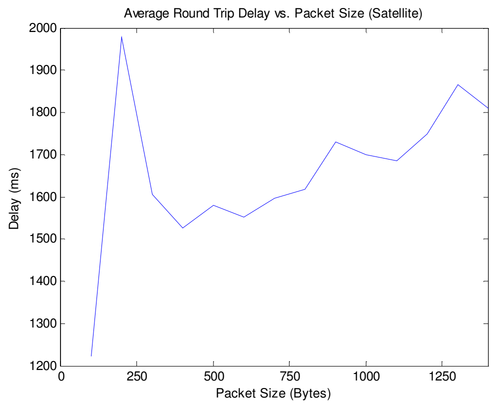
		Test Number (SNR dB)					**AVG**
		1 (51 dB)	**2 (52 dB)**	**3 (52 dB)**	**4 (54 dB)**	**5 (54 dB)**	
Average Forward Throughput (kbps)		402.4	401.9	407.3	400.2	419.4	406.24
Minimum Forward Throughput (kbps)		233.2	162.7	387.1	190.0	361.7	266.9
TCP Throughput (Forward) (kbps)		273.2	185.1	228.4	244.7	248.2	235.9
TCP Throughput (Reverse) (kbps)		293.5	248.2	300.2	264.3	256.2	272.5
Packet Loss (%)		2.75	3.13	2.83	2.74	2.82	2.854
25% Channel Capacity (ms)	**Average Jitter**	56.3	51.6	64.7	56.8	44.4	54.76
	95% Threshold	109.7	109.5	110.2	109.3	101.6	108.06
	99% Threshold	222.8	147.8	199	145.8	134.9	170.06
50% Channel Capacity (ms)	**Average Jitter**	31.6	29	38.3	31.3	33.6	32.76
	95% Threshold	65.5	64	68	63.2	58.6	63.86
	99% Threshold	106.5	109.4	90.7	104.1	97.7	101.68
75% Channel Capacity (ms)	**Average Jitter**	18.9	20.3	19.8	18.5	20.8	19.66
	95% Threshold	35.9	35.9	35	35.2	35.3	35.46
	99% Threshold	44.3	56.2	43.2	41.3	43.2	45.64
100% Channel Capacity (ms)	**Average Jitter**	11.4	15.1	10.7	12.3	13.9	12.68
	95% Threshold	23.6	27.4	23.4	23.6	25.5	24.7
	99% Threshold	32.9	70.4	31.9	33.7	42.4	42.26

<u>5.3.3 Latency vs. Packet Size</u>

Like on 3G and 802.11 networks, latency was examined as a function of packet size on the BGAN network. Figure 118 shows the results from one such test conducted with a signal-to-noise ratio of 54 dB. These results are typical of the other four latency vs. packet size tests run on the BGAN network. The results show that after the packet size surpasses 200 bytes, the same behavior that was exhibited in the standard delay tests becomes apparent once again. The round trip delay times seem to alternate back and forth between two values (~1.2 seconds and ~2 seconds). As the packet size continues to grow, the minimum round trip delay also increases. The maximum round trip delay increases as well; however, it is not as evident as the increase seen in the minimum delay.



(a) Delay vs. Sample Number



(b) Delay vs. Packet Size

Figure 118: Results from a Latency vs. Packet Size Test on a Satellite Link with a SNR of 54 dB

To observe how the average round trip delay is affected by packet size, the results from all five tests were averaged. Figure 119 shows the average round trip delay as a function of packet size ranging from 300 to 1400 bytes. Because the delays experienced using packets less than 300 bytes were erratic and inconsistent, these values were omitted. Equation (5.1) contains the linear fit for the data shown in Figure 119 where RTD is the round trip delay in milliseconds and PS is the packet size in bytes.

$$RTD(PS) = 0.226 \bullet PS + 1503.5 \text{ [ms], for } 300 \leq PS \leq 1400 \quad (5.1)$$

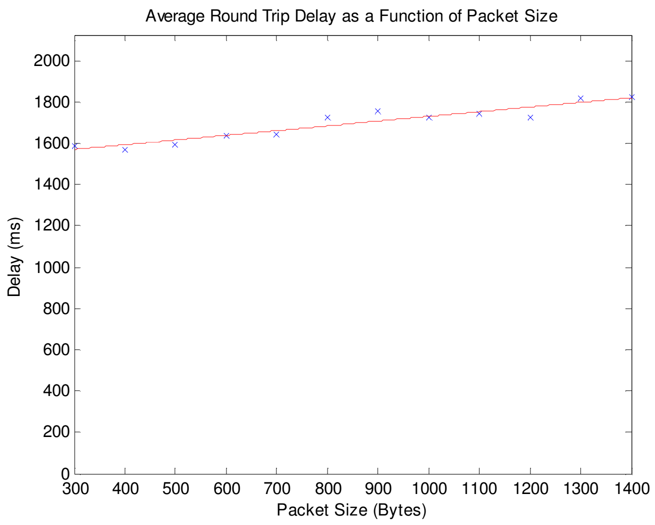
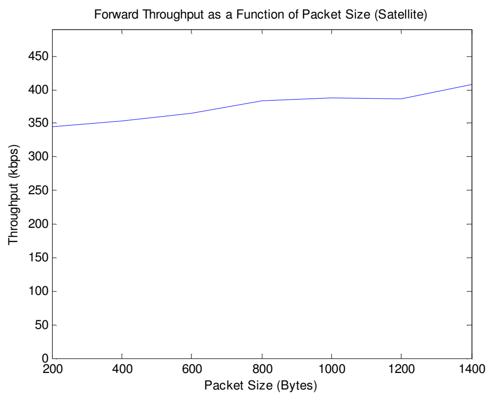


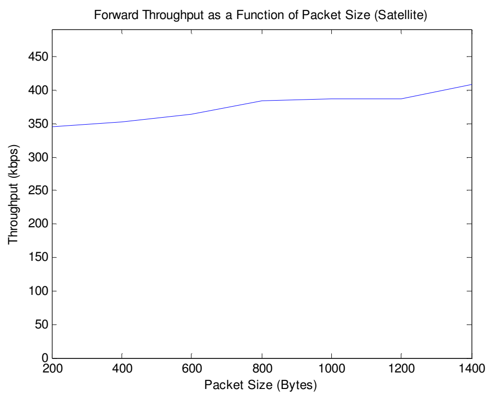
Figure 119: Average Round Trip Delay as a Function of Packet Size on a Satellite Link

5.3.4 Throughput vs. Packet Size

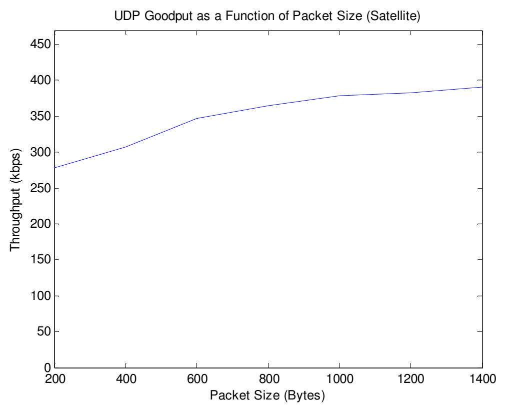
The last type of tests run over the satellite network was throughput vs. packet size tests. Figure 120 shows the results from a throughput vs. packet size test conducted with a signal-to-noise ratio of 54 dB. For each test, the raw forward throughput ((a) and (b)), the UDP goodput (c), the forward packet loss (d) and the forward jitter behavior ((e) through (g)) was measured and recorded.



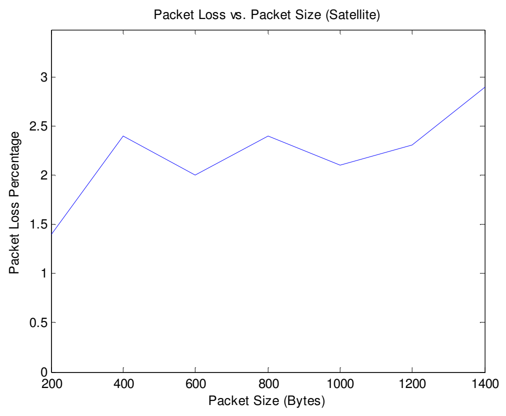
(a) Forward Throughput vs. Packet Size (Client Report)



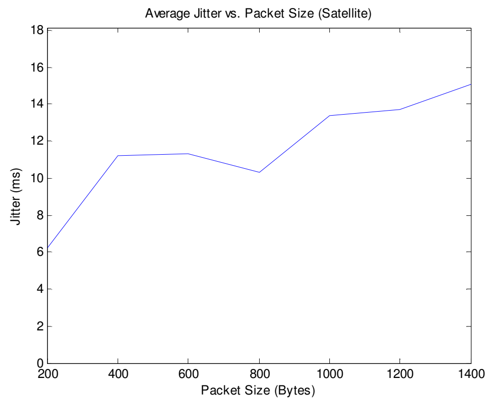
(b) Forward Throughput vs. Packet Size (Server Report)



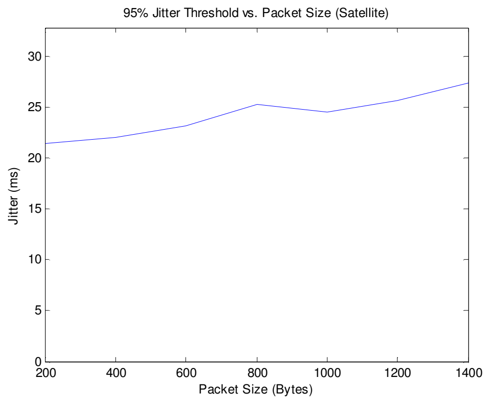
(c) UDP Goodput vs. Packet Size (Server Report)



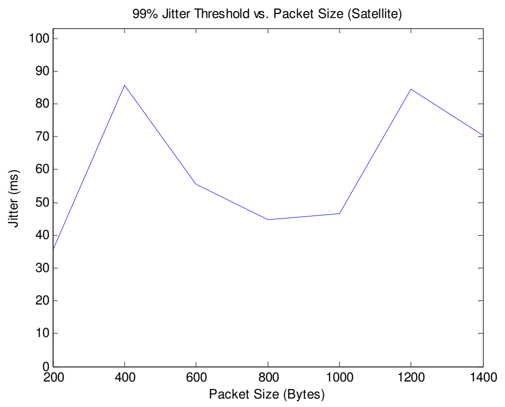
(d) Forward Packet Loss vs. Packet Size



(e) Average Forward Jitter vs. Packet Size (100% Channel Capacity)



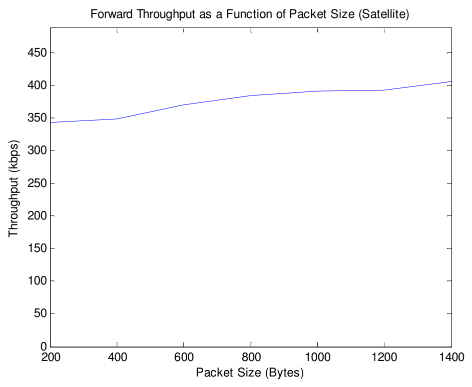
(f) 95% Forward Jitter Threshold vs. Packet Size (100% Channel Capacity)



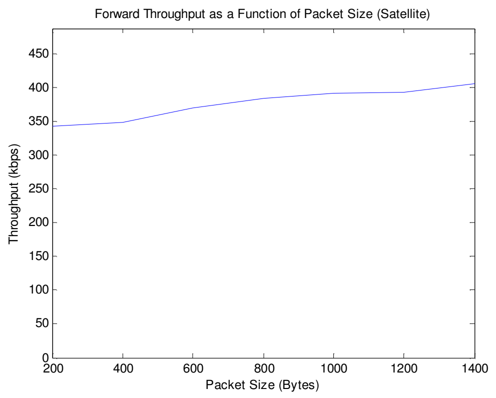
(g) 99% Forward Jitter Threshold vs. Packet Size (100% Channel Capacity)

Figure 120: Results from a Throughput vs. Packet Size Test on a Satellite Link with a SNR of 54 dB

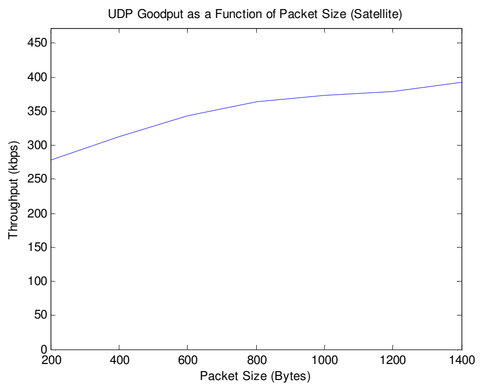
The average throughput results from all five of the tests can be seen in Figure 121. During these tests, a slight improvement in the overall throughput was seen as packet sizes were increased which is evident in Figure 121(a) and (b). This behavior was magnified when looking and the usable data that could be transmitted once UDP overhead was removed (Figure 121(c)).



(a) Forward Throughput vs. Packet Size (Client Report)



(b) Forward Throughput vs. Packet Size (Server Report)



(c) UDP Goodput vs. Packet Size (Server Report)

Figure 121: Average Results from Five Throughput vs. Packet Size Tests on Satellite Links

Figure 122 shows the average packet loss as a function of packet size. As expected, packet loss increases as packets become larger due to the higher probability of an error in a larger packet. Equation (5.2) gives the linear fit for the packet loss as a function of packet size.

$$PL(PS) = 0.00084 \bullet PS + 1.54 \ \ [\%] \qquad (5.2)$$

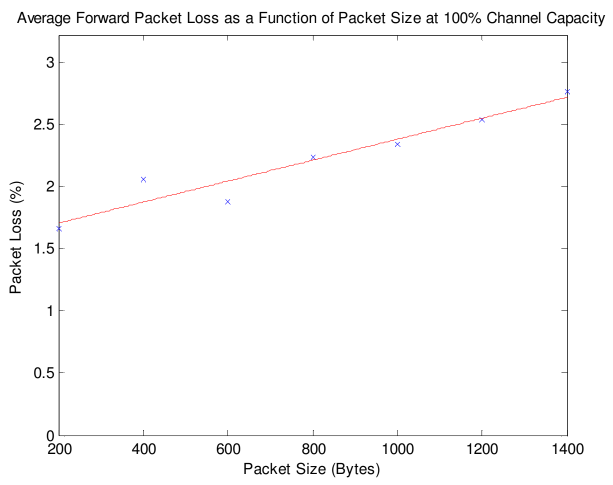


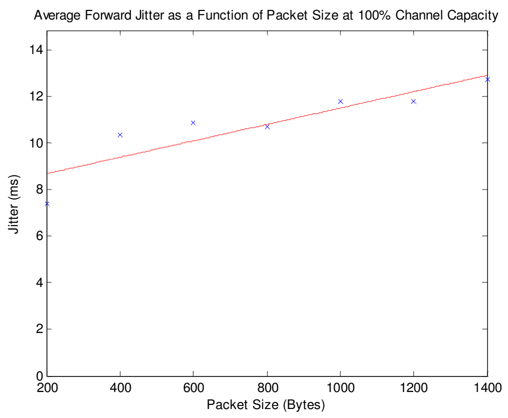
Figure 122: Average Forward Packet Loss vs. Packet Size at 100% Channel Capacity

The last metric that was recorded during these tests was the forward jitter. The results show that the jitter characteristics seem to become slightly worse as packet sizes grow. The jitter characteristics of the uplink can be seen in Figure 123 with the average jitter in (a), the average 95% jitter threshold in (b) and the average 99% jitter threshold in (c). The linear lines of fit for this data can be seen in (5.3), (5.4), and (5.5) respectively.

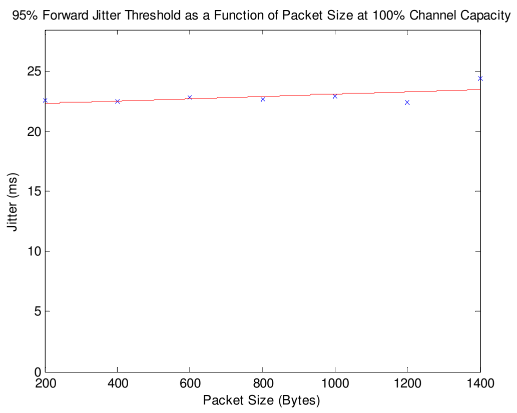
$$FJ(PS) = 0.0035 \bullet PS + 7.98 \text{ [ms]} \tag{5.3}$$

$$FJ95(PS) = 0.0010 \bullet PS + 22.17 \text{ [ms]} \tag{5.4}$$

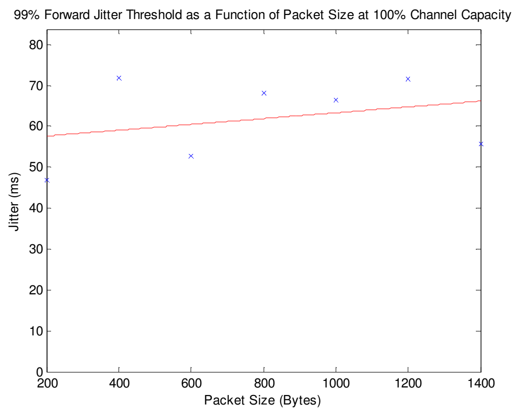
$$FJ99(PS) = 0.0071 \bullet PS + 56.22 \text{ [ms]} \tag{5.5}$$



(a) Average Forward Jitter vs. Packet Size (100% Channel Capacity)



(b) 95% Forward Jitter Threshold vs. Packet Size (100% Channel Capacity)



(c) 99% Forward Jitter Threshold vs. Packet Size (100% Channel Capacity)

Figure 123: Forward Jitter Behavior Using Different Packet Sizes at Channel Capacity on Satellite Link

5.3.5 Conclusion

Performance testing over Inmarsat's BGAN satellite network produced some very useful information when contemplating streaming media applications over the link. The main difference between the BGAN network and other terrestrial networks is the high latency which could cause problems for certain voice or video applications. The throughput of the uplink, which was actually higher than that of the 3G network, was better than initially expected. Table 23 contains a summary of the data gathered during the performance testing. Again, it should be noted that all tests were conducted with a signal-to-noise ratio between 50 and 55 dB at the satellite terminal. The performance of the system outside of this SNR range is not known.

Table 23: Summary of BGAN Satellite Testing Results

Delay	Mean Round Trip Delay	1839 ms
	Mean Forward Delay	1120 ms
	Mean Reverse Delay	718 ms
	Mean Forward Packet Loss	0.96 %
	Mean Reverse Packet Loss	0.32 %
	Mean Forward Jitter	670ms
	95% Forward Jitter Threshold	785 ms
	99% Forward Jitter Threshold	1559 ms
	Mean Reverse Jitter	45 ms
	95% Reverse Jitter Threshold	96 ms
	99% Reverse Jitter Threshold	165 ms
Throughput	Mean Forward Throughput	406.2 kbps
	Mean Forward TCP Throughput	235.9 kbps
	Mean Reverse TCP Throughput	272.5 kbps
	Mean Forward Packet Loss	2.854 %
	Mean Forward Jitter (100% Channel Capacity)	12.6 ms
	95% Forward Jitter Threshold (100% Channel Capacity)	24.7 ms
	99% Forward Jitter Threshold (100% Channel Capacity)	42.2 ms
Delay vs. Packet Size	Mean Round Trip Delay	$RTD(PS) = 0.226 \bullet PS + 1503.5$ [ms], for $300 \le PS \le 1400$
Throughput vs. Packet Size	Mean Forward Packet Loss	$PL(PS) = 0.00084 \bullet PS + 1.54$ [ms]
	Mean Forward Jitter	$FJ(PS) = 0.0035 \bullet PS + 7.98$ [ms]
	95% Forward Jitter Threshold (100% Channel Capacity)	$FJ95(PS) = 0.0010 \bullet PS + 22.17$ [ms]
	99% Forward Jitter Threshold (100% Channel Capacity)	$FJ99(PS) = 0.0071 \bullet PS + 56.22$ [ms]

6 Live Ultrasound Image Stream Testing

In addition to conducting performance testing on the various wireless channels, live image stream testing was performed to examine exactly how the channel properties affect a live image stream. This chapter will first discuss the technology used by the current generation of the ultrasound system to stream live video. It will then discuss a methodology used to transmit and record a live ultrasound image stream on the various wireless links. Finally, a discussion of the results and implications of sending ultrasound video over the different wireless channels will be presented.

6.1 Image Streaming Utilities

The current generation (Gen 3) of the mobile ultrasound system has utilized software from a company called *Layered Media Inc.* (now *Vidyo*) to encode and transmit live ultrasound video. Layered Media uses the H.264 Scalable Video Codec (SVC) to encode and decode individual ultrasound frames and implements the Real-time Transport Protocol (RTP) to transmit the image data. The next generation of the mobile ultrasound system (Gen 4) will not employ Layered Media's utilities to transmit live video; rather a custom application will be developed to achieve this functionality. Although Layered Media's software will no longer be used, both the H.264 SVC standard and RTP will be implemented in this new custom video transmission application. For this reason, it was decided that the utilities provided by Layered Media would be useful in examining the performance of H.264 SVC and RTP in terms of real-time video transmission over wireless links. This section will provide an overview of Layered Media and the technologies used to process and transmit ultrasound images as a live video stream.

6.1.1 Layered Media and H.264 SVC

Layered Media Inc. was the first company to apply the H.264 Scalable Video Codec (SVC) standard to IP video conferencing. The H.264 SVC is an extension to the H.264/AVC standard and allows for “scalable” video streaming. The H.264 SVC standard has the ability to encode a high quality video source at multiple temporal and spatial resolutions. By doing this, a high-quality video bit stream can be broken up into multiple subset bit streams that can be individually decoded to produce a lower quality video than the source. The more of the subset bit streams or layers that can be decoded, the higher the quality of the received video will be [56]. If all of the layers are decoded, then the original high quality video can be reproduced.

Most traditional video transmission systems are not scalable in the sense that they either work or don’t work depending on the data rate of the bit stream and availability of network bandwidth. The H.264 SVC allows systems to adapt to different network conditions and computing power by providing “graceful degradation.” In conventional video conferencing methods, insufficient bandwidth will most likely lead to dropped frames and corrupted images rather than a smooth video with degraded resolution and/or frame rate. Figure 124 shows how H.264 SVC compares to conventional coding techniques [56].

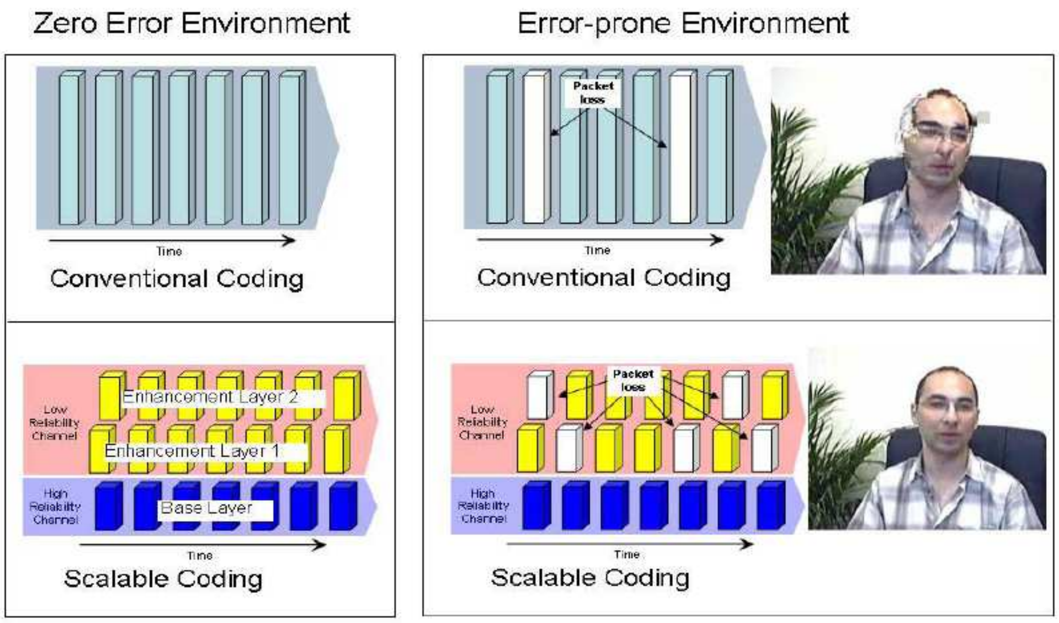


Figure 124: Conventional Coding vs. Scalable Coding [56]

Video conferencing applications can implement the H.264 SVC standard differently depending on the demands and intended use of the specific application. Layered Media's video conferencing software used in this project encodes the original source video into two layers or subset bit streams. If both layers are decoded at the receiving network node, the resulting video will have VGA (640x480) quality. If network resources do no allow for both layers to be decoded, then a single layer will be decoded which will produce a QVGA (320x240) quality video output. In addition to different image resolutions, the receiver can decode images at different frame rates as well. The receiver can decode the video at frame rates ranging from 10 to 30 frames per second (fps), in increments of 5 fps. Obviously, the receiver cannot decode images faster than they are being sent. For example, if the original source video was recorded at only 15 fps, then the maximum rate at which it can be decoded is 15 fps.

6.1.2 Real-time Transport Protocol (RTP)

In order to stream live ultrasound video to a network endpoint, Layered Media utilizes the Real-time Transport Protocol (RTP). RTP, which is defined by the

Internet Engineering Task Force (IETF) in RFC 3550, provides “end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services [55].” This section will briefly describe the Real-time Transport Protocol as it pertains to live data transfer on the mobile ultrasound system.

When referring to the OSI network model discussed in Section 2.1 and shown in Figure 18, RTP resides at the Application Layer or uppermost level of the model. RTP makes use of UDP for the actual transfer of packets between network endpoints. In addition to RTP, the Real-time Control Protocol (RTCP) is used as the control channel, which manages the data transfer between network nodes as well as provides statistics regarding the active session. Like RTP, RTCP also uses UDP datagrams to send information.

Each RTP connection uses two sequential UDP ports on a given network endpoint. For example, an RTP connection on port 1000 will use port 1000 for RTP data transfer and port 1001 for RTCP control signals. Figure 125 shows a theoretical RTP connection on UDP port 1000 over an 802.11 physical layer connection. The dashed lines refer to virtual connections that are achieved using lower level protocols not shown in the figure.

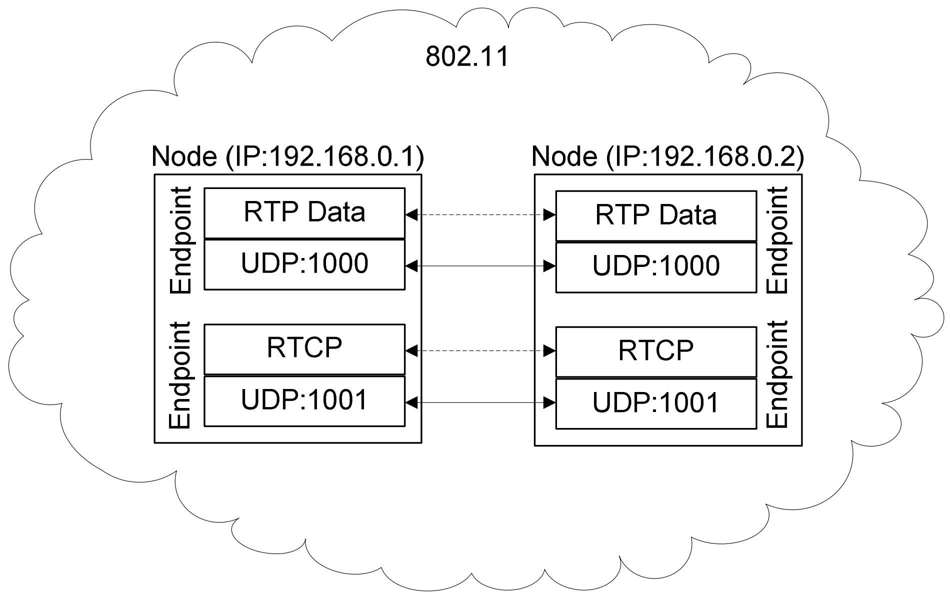


Figure 125: Example RTP Connection over 802.11 [4]

An RTP session contains two or more participants, which use RTP to send and receive information. If a network endpoint only receives data, then it is labeled as a receiver while an endpoint that both sends and receives data is labeled as a sender. Each participant is identified by a unique 32 bit identifier called a Synchronization Source (SSRC). Every RTP data transfer includes a packet header containing the following relevant information [55]:

- RTP Version
- Payload Type
- Sequence Number
- RTP Timestamp
- SSRC

Each RTP packet transferred by a session participant includes all of the above information. The RTP version is the version of the RTP protocol being used. The payload type identifies what type of information is in the packet, such as audio or video data. The sequence number is incremented by one for every packet that is sent, allowing for the calculation of dropped and/or out-of-order packets. The RTP timestamp indicates the time at which the packet was sent while the SSRC identifies the participant that sent this packet.

RTCP is used as the control channel for RTP. The main function of RTCP are session initiation, session tear-down, periodically reporting statistical data to all of the session participants, and gathering information on session participants. The statistical data takes the form of sender or receiver reports, depending on the type of participant, and is periodically sent by every participant in the session. The main purpose of the statistical data is to provide information regarding the quality of data distribution to all of the active participants. This statistical information can be used by RTP applications for functions such as flow control algorithms, adaptive coding algorithms, or to diagnose network problems. Receiver reports are periodically sent by receiver participants to each sender that the receiver has received data from. Each receiver report contains the following information [55]:

- Sender SSRC
- Fraction of packets lost
- Total number of lost packets
- Interarrival jitter
- Timestamp relating to the last sender report receiver from this sender
- Relative delay since receiving the last sender report receiver from this sender

The sender SSRC identifies which sender in this session this report pertains to. The fraction of packets lost gives the fraction of dropped RTP packets from this particular sender. The total number of lost packets is the cumulative number of lost packets from this sender. Interarrival jitter is an estimate of the absolute jitter between these two endpoints. The timestamp is the absolute time that a sender report was received from this sender while the relative delay indicates the relative delay since the last time a sender report was received. The other type of reports sent in RTCP is sender reports. Sender reports are only sent by sender participants and contain the same information as a receiver report, but also includes the following information pertaining to the sender [55]:

- Network Time Protocol (NTP) Timestamp
- RTP Timestamp
- Sender’s packet count
- Sender’s octet count

The NTP timestamp defines the current wallclock or absolute time according to the sender. The RTP timestamp is the same as defined in an RTP data transfer. The sender’s packet count is the cumulative number of RTP data packets sent by this sender while sender’s octet count is the cumulative number of bytes sent in the payloads of each RTP data packet.

6.2 Testing Protocol

This section will present a methodology used to transmit and capture live ultrasound video to observe real-time image quality over the various wireless communication channels. It will discuss the details of the utilities provided by Layered Media necessary to encode, decode, send, and receive live ultrasound video. It will also explain how the received image stream was captured and information about network performance was gathered while live video was being transmitted.

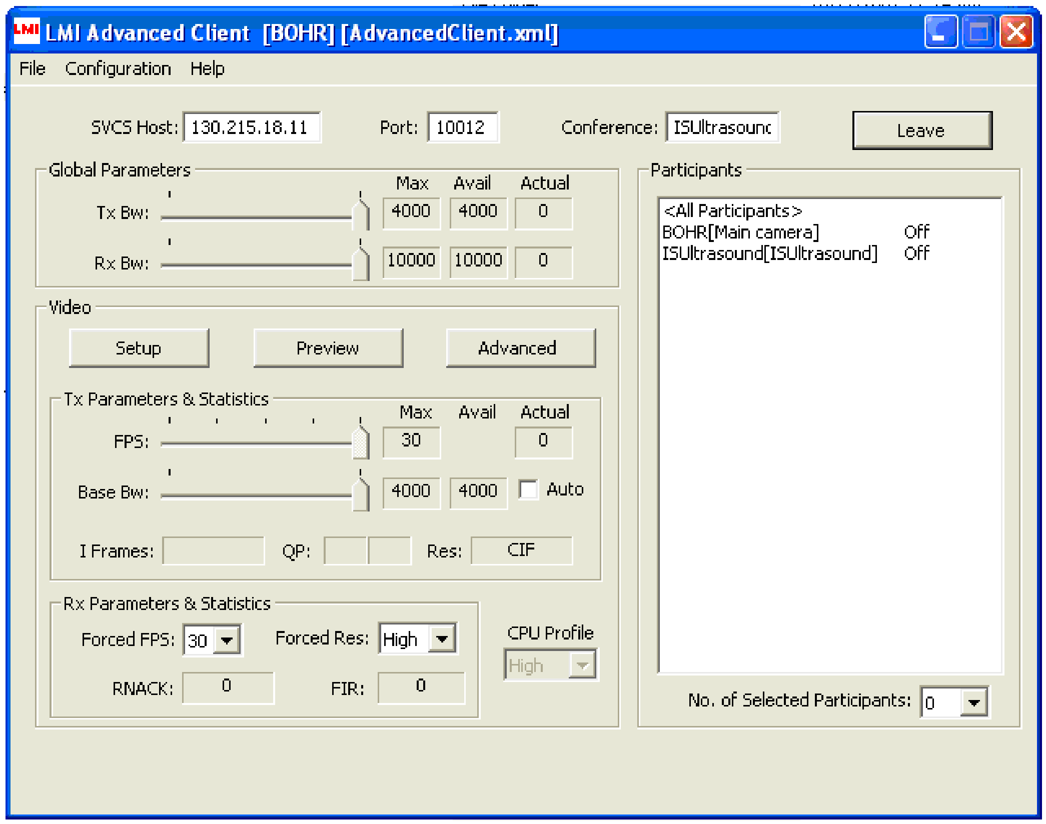
6.2.1 Transmitting

When testing live video over wireless links, an evaluation program provided by Layered Media called *frameclient* was used. Although the name is a little misleading, the frameclient application actually acts as a server in this setup as it is used to encode and transmit individual ultrasound images. The frameclient program functions by taking a directory of individual bitmap images, converting them from a RGB format to a YUV 4:2:0 format, and streaming the images at a specified frame rate. To create a directory of individual bitmap images from a source AVI file, a MATLAB script was needed to break up the AVI file into individual frames, and convert them to 640x480 bitmap images with a color depth of eight bits. This results in a directory of individual images that are about 300 KB each. The rate at which the frames were transmitted could be changed by altering the configuration parameters of the frameclient application.

The frameclient program uses Layered Media's facilities to actually stream the images, meaning it compresses the YUV images using the H.264 SVC standard and sends two subset bit streams at varying resolution. In the current configuration, the frameclient sends the base layer at QVGA (320x240) quality. This is the minimum amount of information necessary to view the transmitted image stream. It also sends an additional enhancement layer that allows the receiving end to decode and view the images at full VGA (640x480) resolution. The base layer is given a higher priority than the enhancement layers; so on a low bandwidth channel, Layered Media will attempt to send an uncorrupted QVGA signal (base layer) before adding the enhancement layers. The receiving end has the ability to choose if it wants to decode both layers or just the base layer.

6.2.2 Receiving

Layered Media also provides a client program called *Advanced Client* that was used to decode and view the transmitted ultrasound video. The client program allows the receiver to specify the resolution and frame rate at which to receive the transmitted images. The three options for specifying the resolution of the images are “Low,” ”High”, and “Auto.” When receiving at a “Low” resolution, only the base layer of the transmitted images are decoded and displayed. In the case of the frameclient program, the base layer is QVGA (320x240). When the “High” resolution option is chosen, the client programs decodes and displays the base layer along with the enhancement layer to provide better resolution. The enhancement layer of a frameclient transmission allows the loop to be viewed at VGA (640x480) quality. If the resolution option is set to “Auto,” then the client program chooses which resolution to decode the image stream at based on the availability of resources such as bandwidth or computing power. Figure 126 shows the configuration window of the Advanced Client program. The reception settings can be seen at the bottom under the heading, “Rx Parameters & Statistics.”


LMI Advanced Client [BOHR] [AdvancedClient.xml]
File
Configuration
Help
Leave
Global Parameters
Video
Setup
Preview
Advanced
Tx Parameters & Statistics
Rx Parameters & Statistics
Participants

SVCS Host: 130.215.18.11 Port: 10012 Conference: ISUltrasounc

Global Parameters

	Max	Avail	Actual
Tx Bw:	4000	4000	0
Rx Bw:	10000	10000	0

Tx Parameters & Statistics

	Max	Avail	Actual
FPS:	30		0
Base Bw:	4000	4000	Auto

I Frames: QP: Res: CIF

Rx Parameters & Statistics

Forced FPS: 30 Forced Res: High

RNACK: 0 FIR: 0

CPU Profile: High

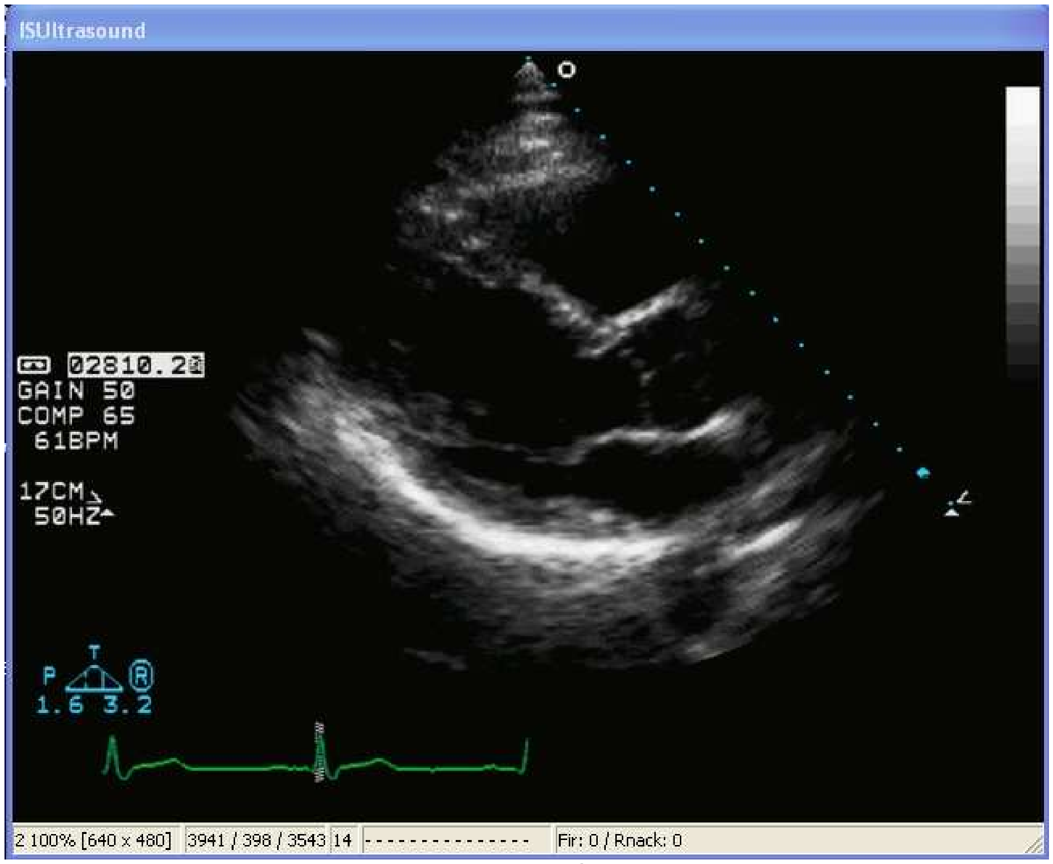
Participants

<All Participants>	
BOHR[Main camera]	Off
ISUltrasound[ISUltrasound]	Off

No. of Selected Participants: 0

Figure 126: LMI Advanced Client Configuration Window

The client program can also limit the rate at which it receives frames. It can receive frames from 10 to 30 frames per second (fps) in increments of 5 fps. The “Forced FPS” option does not alter the transmission rate of the frames. For example, if the receiver is receiving at 10 fps and the sender is sending at 30 fps, the receiver will only see one out of three frames rather than every frame at one-third of the actual speed. Also, it is fairly intuitive that the receiver cannot receive frames faster than they are transmitted. The following figure shows the display window of the *Advanced Client* program. The parameters at the bottom of the window show relevant reception statistics such as how many layers are being decoded, the resolution, the reception bit rate, and the frame rate.



2 100% [640 x 480] 3941 / 398 / 3543 14 --------------- Fir: 0 / Rnack: 0

Figure 127: LMI Advanced Client Image Viewer

<u>*6.2.3 Recording*</u>

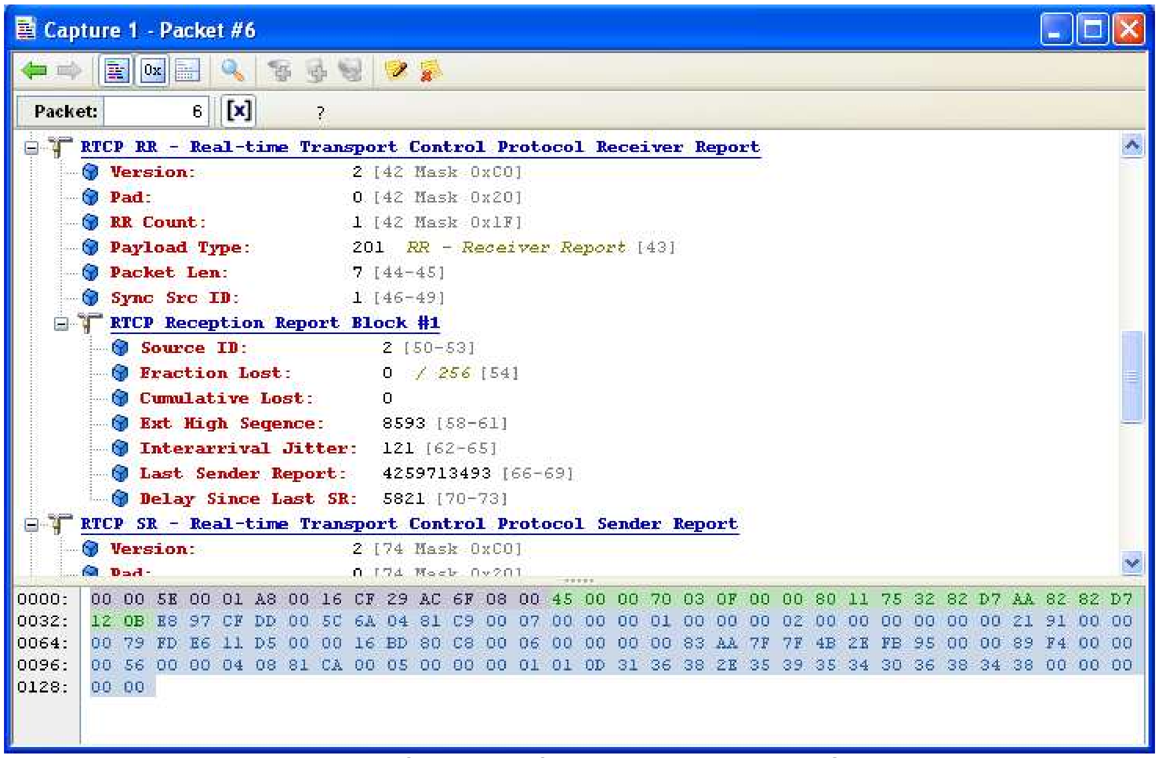
Unfortunately, the Advanced Client program provided by Layered Media is not capable of saving or recording the received image loop to the hard drive. In order to compare the received image stream to the transmitted stream, a screen recording application called *Camtasia* was be used. *Camtasia* has the ability to record a given area of the screen using different frame rates, pixel dimensions, and compression techniques. Screen recording can be very demanding on the CPU depending on the settings during the capture. The result of over-using system resources during a screen capture is either a “choppy” recording that does not closely resemble the actual received loop, or degradation in the performance of the client program resulting in dropped frames at the receiver. The main factors that determine the load placed on the CPU and memory include:

- Color depth (16 bit vs. 32 bit)
- Encoder/Compressor
- CPU priority of the application
- Size of screen recording
- Recorded Frame Rate

After doing some preliminary testing using various combinations of the above settings, one combination continuously provided an acceptable balance between image quality and performance (speed). It was decided to use Camtasia's proprietary video codec called Tech-Smith Screen Capture Codec (TSSCC) to compress the recorded portion of the screen in real time. The TSSCC encoder is a lossless image codec written specifically for screen capturing applications. Other options included using a MPEG-4 part II compressor (DivX) or no compression at all; however, these options did not provide the same balance of quality and performance as the TSSCC compressor. The DivX encoder resulted in a fairly smooth recording at the expense of image quality, while trying to record uncompressed frames resulted in a choppy recording that did not resemble the actual received loop. Lastly, it should be noted that all recordings were made at 30 fps to ensure that the image quality of the recorded clip was not compromised due to undersampling.

6.2.4 Network Statistics

In addition to recording the image stream, it would be helpful to know exactly what type of network conditions correspond to different received video qualities. To do this, a packet capturing program called *WildPackets Omnipeek* was used to capture the RTP and RTCP packets of the video stream. As previously mentioned, RTP periodically sends out control packets called RTCP packets that contain information about either the sender or receiver throughout the streaming session. Included in the RTCP packets is some important statistics such as the fraction of packets lost, interarrival jitter, and total number of packets. WildPackets Omnipeek can be configured to capture only RTP and RTCP packets so these network statistics can later be extracted from the individual packets. Information such as packet loss, jitter, data rate and packet size can be gathered from these packet captures. Also, Omnipeek makes analyzing a single packet much easier than looking at it in binary or hex form by breaking up the individual fields of the packet into a human-readable form. Figure 128 shows an RTCP packet captured by Wildpackets Omnipeek.



```
RTCP RR - Real-time Transport Control Protocol Receiver Report
    Version:              2 [42 Mask 0xC0]
    Pad:                  0 [42 Mask 0x20]
    RR Count:             1 [42 Mask 0x1F]
    Payload Type:         201  RR - Receiver Report [43]
    Packet Len:           7 [44-45]
    Sync Src ID:          1 [46-49]
  RTCP Reception Report Block #1
      Source ID:             2 [50-53]
      Fraction Lost:         0  / 256 [54]
      Cumulative Lost:       0
      Ext High Seqence:      8593 [58-61]
      Interarrival Jitter:   121 [62-65]
      Last Sender Report:    4259713493 [66-69]
      Delay Since Last SR:   5821 [70-73]
RTCP SR - Real-time Transport Control Protocol Sender Report
    Version:              2 [74 Mask 0xC0]

0000:  00 00 5E 00 01 A8 00 16 CF 29 AC 6F 08 00 45 00 00 70 03 0F 00 00 80 11 75 32 82 D7 AA 82 82 D7
0032:  12 0B E8 97 CF DD 00 5C 6A 04 81 C9 00 07 00 00 00 01 00 00 00 02 00 00 00 00 00 00 21 91 00 00
0064:  00 79 FD E6 11 D5 00 00 16 BD 80 C8 00 06 00 00 00 00 83 AA 7F 7F 4B 2E FB 95 00 00 89 F4 00 00
0096:  00 56 00 00 04 08 81 CA 00 05 00 00 00 01 01 0D 31 36 38 2E 35 39 35 34 30 36 38 34 38 00 00 00
0128:  00 00
```

Figure 128: RTCP Packet Capture by WildPackets Omnipeek

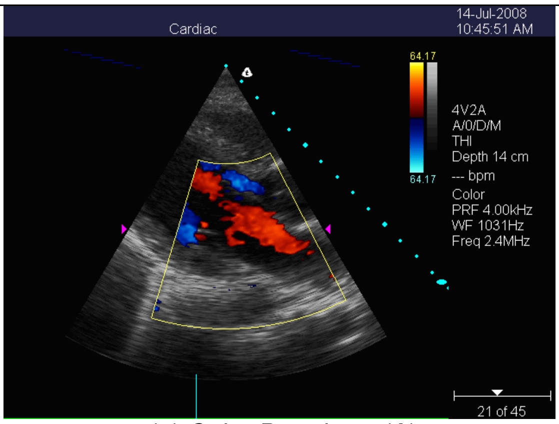
6.2.5 Ultrasound Image Loops

Four separate ultrasound videos were used during image stream testing. Two of the videos were recorded in Color Doppler mode at 15 fps while the other two were standard black and white ultrasound scans recorded at 30 fps. Table 24 gives a description of the four ultrasound loops that were used during testing. Also, for the remainder of this document, the individual image loops will be referred to by the label given in Table 24. For example, “A” corresponds to the Color Doppler 1 image loop.

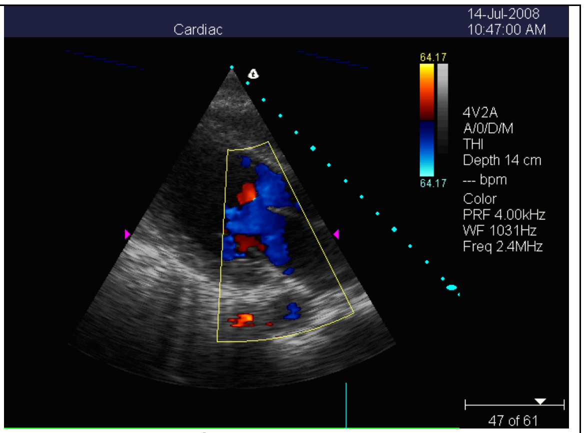
Table 24: Ultrasound Loops used for Image Stream Testing

Label	Scan Type / Loop Name	Frame Rate	Resolution	Scan Type
A	Color Doppler 1	15 fps	VGA	Echocardiograph
B	Color Doppler 2	15 fps	VGA	Echocardiograph
C	Black and White Scan 1	30 fps	VGA	Echocardiograph
D	Black and White Scan 2	30 fps	VGA	Echocardiograph

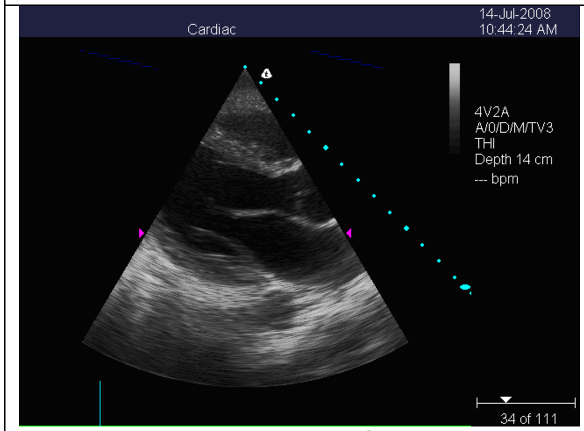
As can be seen from the table, all four of the videos are echocardiographs which are ultrasound scans of the heart. Because a beating heart is continuously moving, these four scans contain a lot of motion compared to other types of ultrasound scans. The reason high-motion videos were chosen for testing was because it would be easier to observe image degradation in loops that have a lot of motion compared to loops that are nearly static, i.e. a lot of correlation from frame to frame. Also, using Color Doppler scans along with black and white scans would reveal any differences in the behavior of the H.264 SVC when streaming different types of ultrasound scans. For example, the different scan types may require different data rates to stream video of the same quality and frame rate. Figure 129 shows a single frame from each one of the ultrasound test loops.



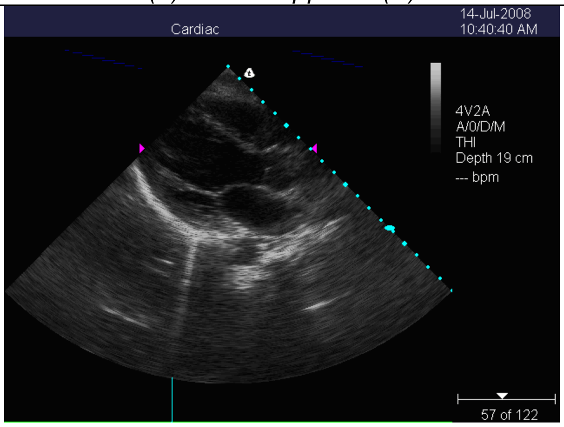
(a) *Color Doppler 1 (A)*



(b) Color Doppler 2 (B)



(c) Black and White Scan 1 (A)



(d) Black and White Scan 2 (D)

Figure 129: Single Frame of Ultrasound Loops Used for Testing

During testing, the test loops are not always presented at the same frame rate as the original source loops. This is done for three reasons. First, it was desired to carry out some of the tests at frame rates lower than 15 fps, and ultrasound source videos at this frame rate could not be obtained. Second, comparisons between Color Doppler scans and a black and white scans at the same frame rate and resolution could only be made if the transmission frame rate was altered. Lastly, Layered Media's software enables the receiver to decode an image stream at a lower frame rate than it is being sent at and this functionality needed to be tested as well.

In the upcoming sections that describe the testing configurations, the following should be kept in mind. If an image stream is being sent at a frame rate lower than the original source video, then the received image stream will appear slower than the original video. This is because all of the frames are sent, but just at a slower rate. For example, if ultrasound loop C is transmitted at 10 fps and received at 10 fps, then it will play back at 1/3rd the speed of the original loop which is at 30 fps. On the other hand, if a loop is sent at a higher frame rate than it is received at, the playback speed will be the same as the original, but not all of the frames are received. For example if loop C is transmitted at 30 fps and decoded at 10 fps, then the speed of the received video will appear the same as the original video; however only one out of every three frames will be decoded and displayed.

6.2.6 Test Setup

To analyze live image streams over the various wireless channels, tests will be set up as follows. Two laptops will be necessary to carry out the tests. The receiving laptop is a ThinkPad Lenovo T61 running Windows XP. It has an Intel dual-core processor with 4 GB of memory. During preliminary tests, this computer has exhibited sufficient performance necessary for decoding and displaying the image stream while simultaneously recording the loop. The laptop that will act as the sender during the tests will be an Acer Travelmate TM3260. Table 25 contains the specifications for the two test laptops.

Table 25: System Specs for Image Stream Testing

Model	Application	Processor	Memory	OS
Acer Travelmate TM3260	Sender	Intel Core Duo T2450 (2 GHz)	2 GB	Windows XP
ThinkPad Lenovo T61	Receiver	Intel Core 2 Duo T8300 (2.4GHz)	4 GB	Windows XP

Figure 130 shows the general setup of the tests and the software necessary on both computers. The sender requires the *frameclient* program for streaming the individual ultrasound images. It uses *Layered Media's* libraries to encode the image stream using H.264 SVC and transmits the images using RTP as previously described. The receiving computer will need to be running three separate programs; *Advanced Client* to decode and display the images, *Camtasia* to record and save the video, and *Wildpackets Omnipeek* to capture RTCP packets for network statistics.

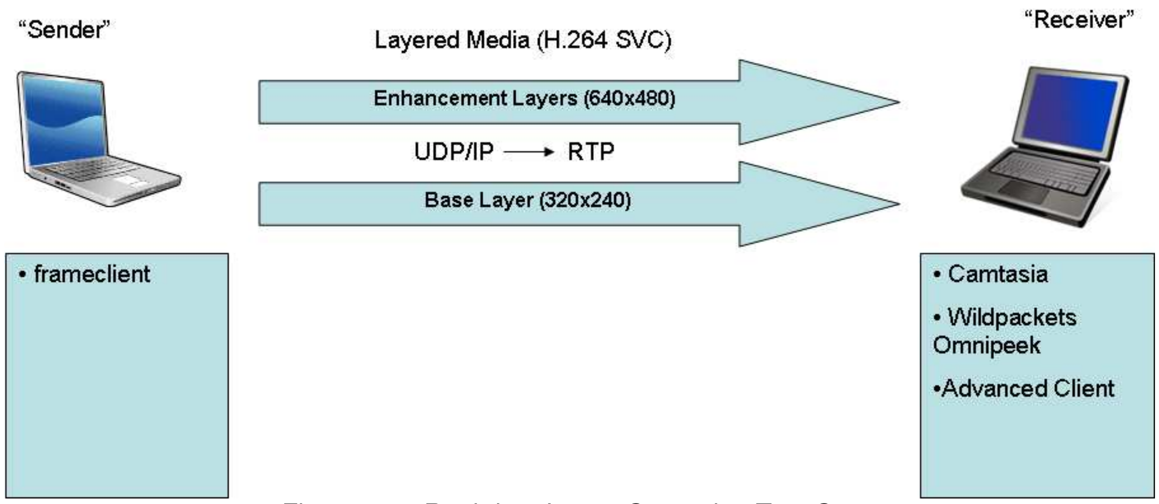


Figure 130: Real-time Image Streaming Test Setup

In addition to recording the received image stream during each test, network statistics were also captured. By capturing RTCP packets during video transmission, information about the networks condition as well as jitter information could be gathered. The following information was gathered for each image stream test:

- Packet Jitter
- Packet Sizes
- Packet Loss
- Data Rate

Different protocols had to be used for the different wireless links, as network conditions restricted certain types of tests on some links. For example, the bandwidth availability on both 3G and satellite networks made streaming VGA quality video impossible. The following sections will outline how tests were conducted over the various wireless channels.

6.2.7 802.11

During preliminary tests, it was determined that an ultrasound image stream could be transmitted at 30 fps at VGA quality using H.264 SVC, given an available minimum data rate of somewhere between 1.5 and 2 Mbps. Based on the performance tests, the data rates supported by the 802.11g standard should easily be able to sustain VGA quality streaming at 30 fps given a high enough SNR. The first set of 802.11 video streaming tests was conducted at a SNR of approximately 35 dB. Table 26 shows the different combinations of streaming scenarios that were conducted over an 802.11g link with a signal-to-noise ratio of around 35 dB.

Table 26: Image Stream Tests for 802.11 Links with a High SNR (~35 dB)

Source Video	Tx (fps)	Rx (fps)	Resolution
Color Doppler 1 & Color Doppler 2	15	15	VGA
Black and White Scan 1 and Black and White Scan 2	30	15	VGA
		30	QVGA
		30	VGA

From the above table it can be seen that eight individual ultrasound recordings were made for each round of tests. In total, two rounds of testing were conducted for 802.11 at this SNR range. Although the data rates supported by 802.11 should easily be able to handle VGA quality video at 30 fps, additional test combinations were added to obtain a better understanding of the behavior of H.264 SVC. For example, it may be beneficial to observe how the data rate or packet size distribution changes when the resolution of a 30 fps video stream is reduced from VGA to QVGA.

Two additional rounds of testing were conducted over 802.11, but this time at a lower signal-to-noise ratio. Based on the 802.11 performance tests, data rates at a SNR of around 20 dB fluctuated around the minimum required data rate for H.264 SVC to stream VGA quality video. This SNR was chose to observe the behavior an H.264 SVC video stream under these conditions. The testing combinations conducted during these two additional rounds of testing can be seen in Table 27.

Table 27: Image Stream Tests for 802.11 Links with a Poor SNR (~20 dB)

Source Video	Tx (fps)	Rx (fps)	Resolution
Color Doppler 1 & Color Doppler 2	15	15	QVGA
		15	VGA
Black and White Scan 1 and Black and White Scan 2	15	15	QVGA
		15	VGA
	30	15	VGA
		30	QVGA

The above combinations were chosen to determine which scenarios would provide the best results with data rates fluctuating around the minimum required data rate of an H.264 SVC video stream. For example, would it be better to stream at 30 fps and decode at 30 fps QVGA or to stream at 30 fps and decode at 15 fps VGA.

6.2.8 3G

From the data rates observed during 3G performance testing, it was obvious that AT&T's HSDPA network would be unable to successfully stream VGA quality video. For this reason, only QVGA videos were used during 3G image stream testing. Table 28 shows the various test combinations that were used during these tests. For each round of tests, twenty ultrasound clips were recorded. Like the 802.11 tests, two full rounds of tests were conducted.

Table 28: Image Stream Tests for 3G and Satellite Links

Source Video	Tx (fps)	Rx (fps)	Resolution
Color Doppler 1 & Color Doppler 2	7	7	QVGA
	10	10	QVGA
	15	10	QVGA
		15	QVGA
Black and White Scan 1 and Black and White Scan 2	7	7	QVGA
	10	10	QVGA
	15	10	QVGA
		15	QVGA
	30	10	QVGA
		15	QVGA

The above combinations were chosen to determine the maximum frame rate that could be supported on a 3G link at QVGA quality. Additionally, it could be reveal any differences in transmitting and receiving at the same frame rate as opposed to sending at a higher frame rate and decoding at a lower frame rate. For example, the test scenarios shown in Table 28 would show any differences between transmitting and receiving at 10 fps as opposed to sending at 15 fps and decoding at 10 fps.

6.2.9 Satellite

Unfortunately, Layered Media's Advanced Client software did not function over Inmarsat's BGAN network. Due to the latency on the link (1.5 to 2 seconds round trip), a connection from the client to the image server could not be made. When contacted, Layered Media was unable to fix the problem with their software. Though the testing software did not work on the link, it does not mean that H.264 SVC can't be used to stream video over a satellite network; only that the software that we were using was unable to create and keep a connection to the image server.

Instead, it was decided to simulate the BGAN network using the network emulator to test image streaming. The only difference that was made was to lower the one way delay of the network from around one second down to 100 ms. This allowed the Advanced Client software to keep a connection with the server. The rest of the emulator settings, which can be seen in Table 29, were taken from the results of the satellite performance tests. The same tests that were run on the 3G network (Table 28) were run on the satellite network emulator as well.

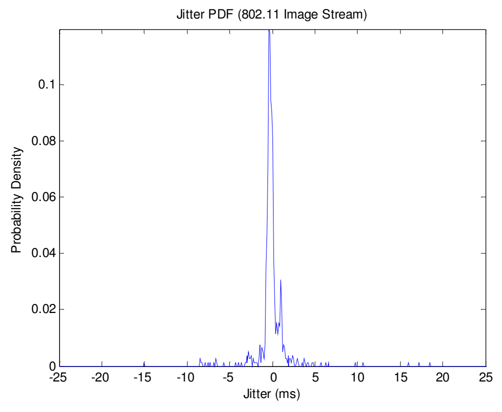
Table 29: Network Emulator Settings for Both the Uplink and Downlink for Satellite Image Streaming

Parameter	**Value**
Throughput	390 kbps
Delay	100 ms
Jitter	13.5 ms
Packet Loss	2.5%

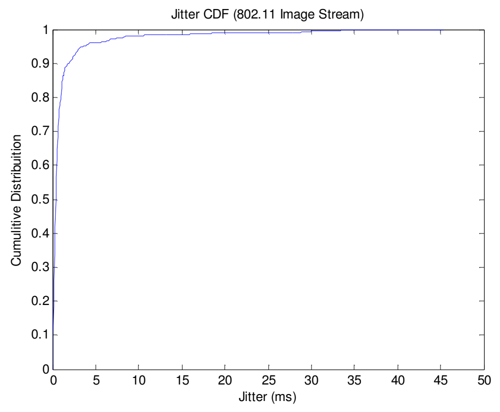
6.3 Testing Results

As outlined in the previous section, a number of metrics were gathered in addition to screen recording of the received image stream. All of the screen recordings made during testing can be found on the DVD accompanying this document. Appendix G contains the 802.11 recordings, Appendix H has the 3G recording, and finally, Appendix I includes the recordings from the satellite emulator. Figure 131 shows the complete results of one such test. This figure contains the testing results from an image stream transmitted over an 802.11 link with a SNR of 20 dB. The image stream that was used was the “Black and White Scan 1” described in the previous section. The stream was transmitted at 15 fps and decoded at 15 fps in QVGA resolution.

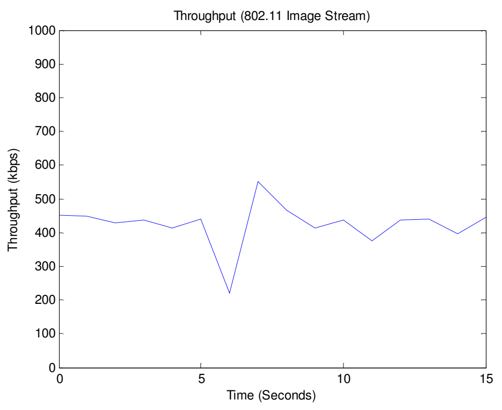
802.11 (20 dB)	**B&W Scan 1**	**Tx - 15 fps**	**Rx – 15 fps**	**QVGA**



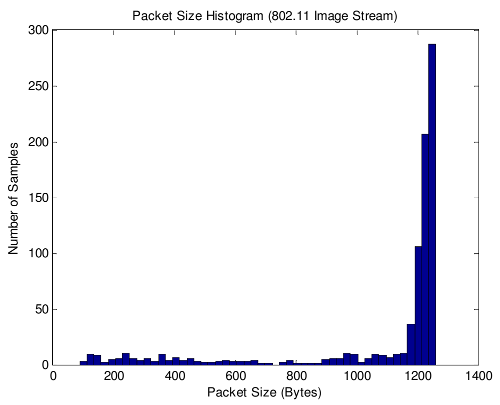
(a) Jitter PDF



(b) Jitter CDF



(c) Image Stream Data Rate vs. Time



(d) Histogram of Image Stream Packet Sizes

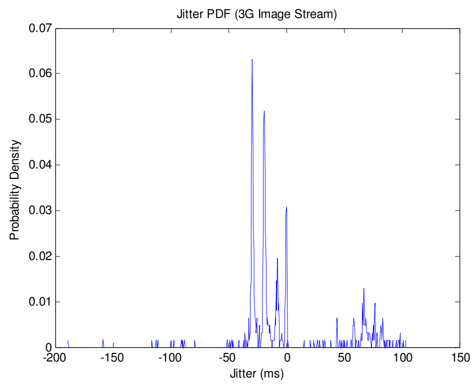
Average Data Rate = 426 kbps
Max Data Rate = 552 kbps
Min Data Rate = 221 kbps
Packet Loss = 2.4%
Average Jitter = 1.2 ms
95% Jitter = 3.6 ms
99% Jitter = 28.6 ms
Average Packet Size = 1082 Bytes
Packet Size Standard Deviation = 314 Bytes

Figure 131: Test Results for a Black and White Ultrasound Image Stream Transmitted over an 802.11 Link with an SNR of 20 dB

For each image stream, the jitter was measured throughout the duration of the test. Figure 131 (a) and (b) show the jitter PDF and CDF, respectively. The data rate of the image stream can be seen in Figure 131 (c) as a function of time. It should be noted that this value corresponds to the data rate of the useful video information and does not include network overhead. Assuming an average packet size of approximately 1100 bytes and 40 bytes of overhead per packet (IP - 20 bytes; UDP – 8 bytes; RTP – 12 bytes), the actual data rate over the network is approximately 1.036 times or 3.6% greater than those given in Figure 131 as well as all other image streaming results. Lastly, a histogram of the packet sizes that made up the video stream is shown in Figure 131 (d). A summary of the results is given at the bottom of the figure. The above information was recorded and plotted for each test, and the complete results for all of the image stream tests can be found in the appendices of this document. The 802.11, 3G and satellite results can be found in Appendix D, Appendix E, and Appendix F respectively. To provide a comparison of the different types of results obtained during tests, and complete set of results from a 3G image test can be seen in Figure 132. The differences between this 3G test and the 802.11 test from Figure 131 are quite clear, especially in the jitter behavior. The remainder of this section will summarize the results of the image stream tests over the various wireless channels.

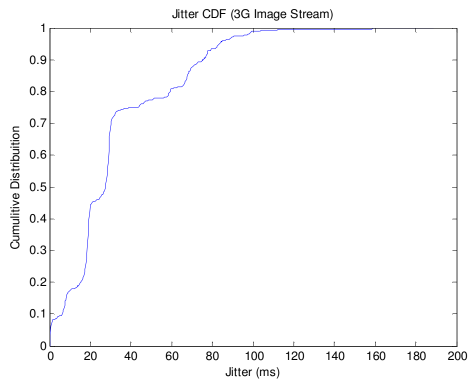
3G	Color Doppler 2	Tx - 10 fps	Rx – 10 fps	QVGA

Jitter PDF



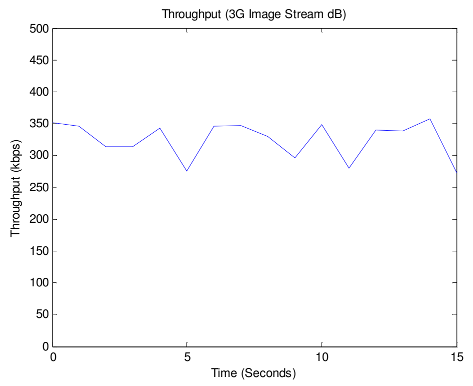
(a) Jitter PDF

Jitter CDF



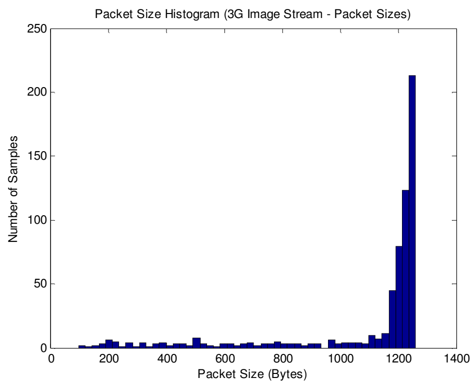
(b) Jitter CDF

Throughput



(c) Image Stream Data Rate vs. Time

Packet Size PDF



(d) Histogram of Image Stream Packet Sizes

Average Data Rate = 325 kbps
Max Data Rate = 357 kbps
Min Data Rate = 273 kbps
Packet Loss = 3.3%
Average Jitter = 32.5 ms
95% Jitter = 83.9 ms
99% Jitter = 102.2 ms
Average Packet Size = 1095 Bytes
Packet Size Std Deviation = 283 Bytes

Figure 132: Test Results for a Color Doppler Ultrasound Image Stream Transmitted over a 3G Link with an SNR of -75 dBm

6.3.1 802.11

Two sets of tests were conducted on an 802.11 channel with a signal-to-noise ratio of approximately 35 dB. Based on the performance testing, this SNR should provide sufficient bandwidth to support the data rates necessary to transmit VGA quality video at 30 fps using H.264 SVC. After viewing the results from the image stream testing, this was in fact the case. Table 31 shows the complete testing results for the 802.11 image stream tests with a SNR of 35 dB. The top four rows of the table describe the type of test that was run including the source image stream, the transmission frame rate, the reception frame rate, and the resolution. For example, the first test which is shown in the third column of Table 31 used the "Color Doppler 1" stream as the video source, and transmitted and received the stream at 15 fps in VGA quality. The leftmost column shows the type of measurement that is being presented. Lastly, the second column shows with test (Test 1 or Test 2) corresponds with the data and finally provides an average value for both of the tests.

In addition to the metrics shown at the bottom of Figure 131, a row titled "Received Image Stream Quality" was added to the bottom of the table. This field gives a qualitative grade to the image stream in attempts to describe the overall quality of the received video. Table 30 provides a description of the different grades used to indicate the quality of the image stream. An "A" corresponds to uncorrupted video that has a smooth playback and no indications of image degradation. A "B" means that there are some segments of the video where image degradation is noticeable, however the overall quality of the video is still pretty good, and the majority of the video is absent of image degradation. A "C" corresponds to significant degradation in the image stream which is present in the majority of the video. A "D" refers to cases where video streaming is not possible, and continuous image playback cannot occur. It should be noted that in no way do these grades reflect the clinical value of the various clips. For example, an image stream at 7 fps and QVGA resolution may receive a grade of

"A"; however that does not mean that this image stream will be useful in a clinical ultrasound setting. This issue will be explored in the following section titled "Physician Feedback."

Table 30: Legend for "Received Image Stream Quality" Field

A	No apparent degradation in image quality. Smooth video playback with no pauses, speedups or dropped frames.
B	Presence of some sort of degradation in image quality such as pauses, speedups or dropped frames. Percentage of smooth video playback greatly outweighs degraded image segments.
C	Significant presence of degradation in image quality. Percentage of degraded image segments approximately equal to or greater than smooth video playback.
D	Image streaming not possible. Frozen video or no video at all.

Table 31: Complete Test Results for 802.11 Image Stream Tests with a SNR of 35 dB

Image Stream		**A**	**B**	**C**	**C**	**C**	**D**	**D**	**D**
Tx FPS		**15**	**15**	**30**	**30**	**30**	**30**	**30**	**30**
Rx FPS		**15**	**15**	**15**	**30**	**30**	**15**	**30**	**30**
Resolution		**VGA**	**VGA**	**VGA**	**VGA**	**QVGA**	**VGA**	**VGA**	**QVGA**
Data Rate (kbps)	Test 1	1447	1580	1293	1337	634	1321	1355	637
	Test 2	1469	1567	1311	1312	631	1327	1328	634
	AVG	**1458**	**1574**	**1302**	**1325**	**633**	**1324**	**1342**	**636**
Packet Loss (%)	Test 1	0.1	0.1	0	0.1	0.2	0.1	0	0
	Test 2	0	0	0.1	0	0.2	0.3	0	0.1
	AVG	**0.05**	**0.05**	**0.05**	**0.05**	**0.2**	**0.2**	**0**	**0.05**
Average Jitter (ms)	Test 1	0.23	0.31	0.35	0.47	0.59	0.33	0.39	0.52
	Test 2	0.28	0.31	0.38	0.63	0.56	0.31	0.39	1.31
	AVG	**0.26**	**0.31**	**0.37**	**0.55**	**0.58**	**0.32**	**0.39**	**0.92**
95% Jitter Threshold (ms)	Test 1	0.71	0.76	1.24	1.4	1.35	0.93	1.07	1.44
	Test 2	0.83	0.9	0.99	1.53	1.33	0.99	1.34	2.34
	AVG	**0.77**	**0.83**	**1.12**	**1.47**	**1.34**	**0.96**	**1.21**	**1.89**
99% Jitter Threshold (ms)	Test 1	1.92	2.18	4.18	7.27	18.09	4.79	6.17	13.6
	Test 2	2.21	2.17	4.37	8.77	16.2	2.41	4.5	29.41
	AVG	**2.07**	**2.18**	**4.28**	**8.02**	**17.15**	**3.60**	**5.34**	**21.51**
Average Packet Size (Bytes)	Test 1	1161	1163	1146	1101	1020	1169	1115	1062
	Test 2	1157	1162	1139	1101	1016	1163	1116	1057
	AVG	**1159.0**	**1162.5**	**1142.5**	**1101.0**	**1018.0**	**1166.0**	**1115.5**	**1059.5**
Packet Size Std Deviation (Bytes)	Test 1	224	224	262	312	366	224	289	307
	Test 2	232	225	275	312	369	236	290	323
	AVG	**228**	**224.5**	**268.5**	**312**	**367.5**	**230**	**289.5**	**315**
Received Image Stream Quality	Test 1	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**
	Test 2	**A**	**A**	**A**	**A**	**A**	**A**	**A**	**A**

From Table 31, it is obvious that 802.11 at a relatively high SNR is more than capable of streaming VGA quality video at 30 fps using H.264 SVC. In all cases, the received video appeared uncorrupted and the image stream played back smoothly. The results also show that the Color Doppler streams (A and B) require more bandwidth than do the black and white scans (C and D). To transmit the Color Doppler scans at 15 fps in VGA resolution, the average data rates were 1458 kbps for A and 1574 kbps for B. To transmit at the same rate and resolution, the black and white scans only required 1302 kbps and 1324 kbps respectively. The packet loss and jitter characteristics were all consistent with those observed during the performance testing on 802.11 at similar signal-to-noise ratios.

Additional streaming tests were conducted over 802.11; this time at a lower SNR. Table 32 shows the complete results from two sets of streaming tests run over 802.11 with a signal-to-noise ratio of approximately 20 dB. It was decided to run tests at a SNR of around 20 dB because the performance tests showed that the bandwidth was unstable in this range of SNRs and it would be beneficial to observe the consequences bandwidth fluctuations close to the required minimum data rate of the image stream.

Table 32: Complete Test Results for 802.11 Image Stream Tests with a SNR of 20 dB

Image Stream		**A**	**A**	**B**	**B**	**C**	**C**	**C**	**C**	**D**	**D**	**D**	**D**
Tx FPS		**15**	**15**	**15**	**15**	**15**	**15**	**30**	**30**	**15**	**15**	**30**	**30**
Rx FPS		**15**	**15**	**15**	**15**	**15**	**15**	**15**	**30**	**15**	**15**	**15**	**30**
Resolution		**QVGA**	**VGA**	**QVGA**	**VGA**	**QVGA**	**VGA**	**VGA**	**QVGA**	**QVGA**	**VGA**	**VGA**	**QVGA**
Data Rate (kbps)	Test 1	576	1404	500	1348	426	1605	1302	631	441	1404	1989	638
	Test 2	533	1444	488	1516	427	1293	1300	628	443	1627	1341	641
	AVG	**555**	**1424**	**494**	**1432**	**427**	**1449**	**1301**	**630**	**442**	**1516**	**1665**	**640**
Packet Loss (%)	Test 1	1.7	4.5	2.2	4.9	2.4	3.5	2.1	2	1.2	3	2.9	1.9
	Test 2	4.3	1.4	2	1.9	0.5	6	2	1.7	2.6	1.6	1.6	0.8
	AVG	**3.0**	**3.0**	**2.1**	**3.4**	**1.5**	**4.8**	**2.1**	**1.9**	**1.9**	**2.3**	**2.3**	**1.4**
Average Jitter (ms)	Test 1	4.6	2.7	3.9	2.8	1.2	8.3	2.7	6.22	2.2	1.9	1.2	3
	Test 2	6.2	2	6.9	3.1	3	2.5	4.1	1.5	2.1	4.4	2.2	1
	AVG	**5.4**	**2.4**	**5.4**	**3.0**	**2.1**	**5.4**	**3.4**	**3.9**	**2.2**	**3.2**	**1.7**	**2.0**
95% Jitter Threshold (ms)	Test 1	31.5	9.4	12.1	9	3.6	35.5	9.9	28.9	8.5	4.9	3.2	20.5
	Test 2	25.9	4.8	50.3	12.9	10.1	8.9	16.9	7.1	6.6	19.8	7.9	2.2
	AVG	**28.7**	**7.1**	**31.2**	**11.0**	**6.9**	**22.2**	**13.4**	**18.0**	**7.6**	**12.4**	**5.6**	**11.4**
99% Jitter Threshold (ms)	Test 1	65.9	45.7	63.9	43.1	28.6	64.7	34.2	76.5	22.9	26.3	14	36.4
	Test 2	65.9	33.9	69.4	39.7	62.6	40.4	51.1	23.8	29.5	53.2	33.2	15.7
	AVG	**65.9**	**39.8**	**66.7**	**41.4**	**45.6**	**52.6**	**42.7**	**50.2**	**26.2**	**39.8**	**23.6**	**26.1**
Average Packet Size (ms)	Test 1	1102	1161	1091	1164	1082	1164	1144	1026	1096	1175	1162	1047
	Test 2	1101	1161	1094	1165	1080	1165	1138	1026	1095	1174	1165	1049
	AVG	**1102**	**1161**	**1093**	**1165**	**1081**	**1165**	**1141**	**1026**	**1096**	**1175**	**1164**	**1048**
Packet Size Std Deviation (ms)	Test 1	278	224	291	219	314	225	266	365	306	218	232	326
	Test 2	277	221	289	218	314	224	276	363	307	219	231	325
	AVG	**278**	**223**	**290**	**219**	**314**	**225**	**271**	**364**	**307**	**219**	**232**	**326**
Received Image Stream Quality	Test 1	**B**	**C**	**B**	**B**	**B**	**B**	**B**	**A**	**A**	**C**	**B**	**A**
	Test 2	**B**	**B**	**B**	**B**	**A**	**C**	**B**	**B**	**B**	**B**	**B**	**A**

In general, the received image streams for these tests where characterized by long periods smooth video playback with short pauses or freezes sporadically mixed in. These pauses were due to instances where a sequence of packets was dropped or when the bandwidth of the channel temporarily dropped below the minimum data rate required by the image stream. The packet loss results show that the packet loss for these tests was significantly higher than it was for the tests run at a SNR of 35 dB. These packet loss results along with the jitter behavior are consistent with the results from the 802.11 performance tests for signal-to-noise ratios around 20 dB. For the most part, these image streams could still be valuable in live ultrasound applications; however, periodic degradation in image quality may cause problems from time to time.

6.3.2 3G

In total, two sets of twenty tests were run over AT&T's 3G cellular network as described in Section 6.2.6. The complete results from these tests can be seen in Table 33. For these tests, video could only be streamed in QVGA resolution due to the bandwidth limitation of the 3G network. One of the tests (Test 1) was conducted with a received signal strength of -75 dBm while the other (Test 2) had a signal strength of -85 dBm. Like the performance tests, no significant differences were observed due to variations in signal strength.

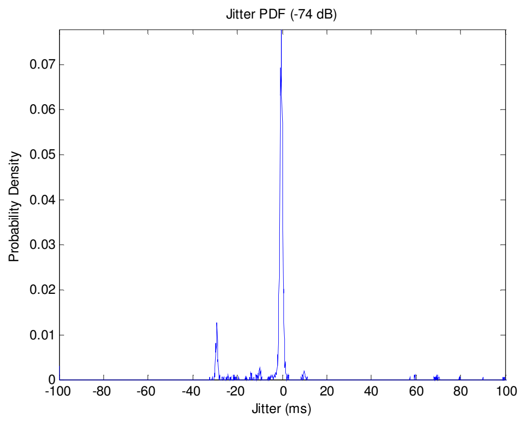
Table 33: Complete Test Results for 3G Image Stream Tests

Image Stream		**A**	**A**	**A**	**A**	**B**	**B**	**B**	**B**	**C**	**C**	**C**	**C**	**C**	**C**	**D**	**D**	**D**	**D**	**D**	**D**
Tx FPS		**7**	**10**	**15**	**15**	**7**	**10**	**15**	**15**	**7**	**10**	**15**	**15**	**30**	**30**	**7**	**10**	**15**	**15**	**30**	**30**
Rx FPS		**7**	**10**	**10**	**15**	**7**	**10**	**10**	**15**	**7**	**10**	**10**	**15**	**10**	**15**	**7**	**10**	**10**	**15**	**10**	**15**
Resolution		**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**	**QVGA**
Data Rate (kbps)	Test 1	287	348	344	337	255	325	345	343	214	277	337	337	233	347	226	295	350	349	285	345
	Test 2	289	350	345	343	249	325	346	341	217	281	334	343	233	341	224	288	346	342	271	345
	AVG	**288**	**349**	**345**	**340**	**252**	**325**	**346**	**342**	**216**	**279**	**336**	**340**	**233**	**344**	**225**	**292**	**348**	**346**	**278**	**345**
Packet Loss (%)	Test 1	1.4	2.8	1.7	1.6	1.5	3.3	2.7	1	2.2	1.6	1.6	2	2	1.8	1.1	2	0.9	2.4	1.5	1.9
	Test 2	2	1.6	2.7	2.1	1.8	1.9	1.3	2.4	1.7	2.2	1.5	1.2	1.8	2.1	2	2.1	2.2	1.7	1.3	1.6
	AVG	**1.7**	**2.2**	**2.2**	**1.9**	**1.7**	**2.6**	**2.0**	**1.7**	**2.0**	**1.9**	**1.6**	**1.6**	**1.9**	**2.0**	**1.6**	**2.1**	**1.6**	**2.1**	**1.4**	**1.8**
Average Jitter (ms)	Test 1	34.9	35.1	42	41.6	33.8	32.5	38.4	41.6	33.4	31.7	31.7	38.5	30.3	39.3	33.7	32.3	34.4	41.4	34	41.4
	Test 2	36.8	37.7	41.5	40.7	35	32.5	37.9	40.9	35.7	32.1	30.9	38.5	29.8	38.9	33.8	31	33.6	40.7	33.7	41
	AVG	**35.9**	**36.4**	**41.8**	**41.2**	**34.4**	**32.5**	**38.2**	**41.3**	**34.6**	**31.9**	**31.3**	**38.5**	**30.1**	**39.1**	**33.8**	**31.7**	**34.0**	**41.1**	**33.9**	**41.2**
95% Jitter Threshold (ms)	Test 1	103.2	97.3	100.6	91.1	96.9	83.9	112.2	93.5	91.7	78.4	97.8	99.3	79.6	73.7	97.3	79.5	119.2	86.1	92.6	94.9
	Test 2	109.7	109.9	99.6	95.5	102	82.4	104.5	99.8	99.8	78.1	98.2	99.3	79.6	77.3	90.2	81.6	113.9	92.1	92.6	98.7
	AVG	**106.5**	**103.6**	**100.1**	**93.3**	**99.5**	**83.2**	**108.4**	**96.7**	**95.8**	**78.3**	**98.0**	**99.3**	**79.6**	**75.5**	**93.8**	**80.6**	**116.6**	**89.1**	**92.6**	**96.8**
99% Jitter Threshold (ms)	Test 1	125.9	255.2	604.9	617.9	127.5	102.2	511.5	614.9	118.7	100.2	122.3	549.3	108.6	606.1	126.9	98.9	218.8	608.6	122.2	599.7
	Test 2	135.5	529.1	601.7	620	138.3	109.3	515.2	611.5	130	105.4	1221.4	549.3	109.2	611.6	125.9	103.6	221.2	603.9	124.3	602.9
	AVG	**130.7**	**392.2**	**603.3**	**619.0**	**132.9**	**105.8**	**513.4**	**613.2**	**124.4**	**102.8**	**671.9**	**549.3**	**108.9**	**608.9**	**126.4**	**101.3**	**220.0**	**606.3**	**123.3**	**601.3**
Average Packet Size (Bytes)	Test 1	1107	1099	1099	1099	1098	1095	1093	1117	1070	1079	1075	1082	1056	1032	1106	1117	1112	1117	1098	1080
	Test 2	1106	1108	1111	1113	1090	1090	1092	1111	1091	1080	1087	1077	1058	1049	1099	1083	1106	1113	1113	1106
	AVG	**1107**	**1104**	**1105**	**1106**	**1094**	**1093**	**1093**	**1114**	**1081**	**1080**	**1081**	**1080**	**1057**	**1041**	**1103**	**1100**	**1109**	**1115**	**1106**	**1093**
Packet Size Std Deviation (Bytes)	Test 1	269	282	283	288	283	283	302	277	323	316	324	313	341	361	288	277	282	272	296	326
	Test 2	270	270	272	256	290	288	292	270	306	310	311	320	336	336	306	322	298	283	271	288
	AVG	**270**	**276**	**278**	**272**	**287**	**286**	**297**	**274**	**315**	**313**	**318**	**317**	**339**	**349**	**297**	**300**	**290**	**278**	**284**	**307**
Received Image Stream Quality	Test 1	B	B	D	D	A	A	C	D	A	A	A	C	A	B	B	A	A	D	B	C
	Test 2	A	A	C	D	A	A	C	D	A	A	A	B	B	C	A	A	B	D	B	C

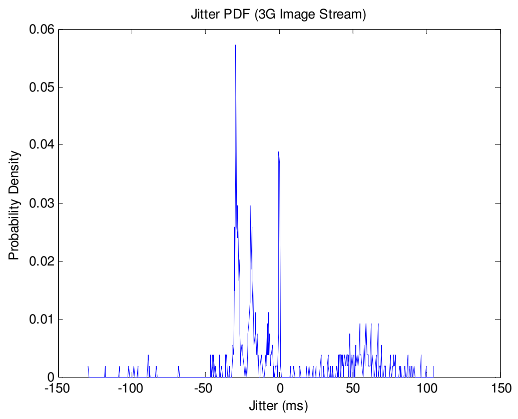
The results from the 3G streaming tests show that the capacity of AT&T's HSDPA network is very close to the minimum threshold necessary to transmit and receive QVGA quality video at 15 fps. During both sets of tests, all four of the image streams could be transmitted and received at acceptable video quality (grade "A" or "B") at 7 fps as well as 10 fps. Additionally, both of the black and white scans (C and D) could be streamed with acceptable image quality while being transmitted at both 15 fps and 30 fps while being decoded at 10 fps. This was not the case for the Color Doppler streams (A and B) as there was excessive degradation once the transmission frame rate exceeded 10 fps. The image quality for the black and white scans significantly declined when the client attempted to decode at greater than 10 fps.

Three conclusions can be drawn from these results. The first, which was already observed during the 802.11 tests, is that the Color Doppler streams require more bandwidth (roughly 10% more) than do the black and white scans to stream video at the same resolution and frame rate. This is evident by the fact that the black and white scans have lower average data rates than the Color Doppler scans. Also, C and D could be successfully transmitted at 15 fps and decoded at 10 fps while A and B could not. Next, the results show that there is a difference between transmitting at 15 fps and decoding at 10 fps as opposed to transmitting and receiving at 10 fps. This can be concluded because both of the Color Doppler scans do not have a problem transmitting and receiving at 10 fps; however, once the transmission frame rate is increased to 15 fps, the received image stream is significantly degraded even though the client attempts to decode it at 10 fps. Lastly, it appears as if the maximum data rate that can be supported by the uplink of the 3G channel is approximately 350 kbps (~363 kbps including overhead). This comes pretty close to the average throughput of the 3G network observed during the performance tests which was around 380 kbps.

The last interesting characteristic observed during 3G image stream testing was the jitter behavior. For the 802.11 tests, the jitter behavior of the image stream matched up with those seen when testing the bandwidth during the performance testing. This was not the case here. The average forward jitter observed during performance testing was around 7 ms while it jumped to around 33 ms during image stream testing. Figure 133 shows a jitter PDF taken during a bandwidth test as well as one taken during a streaming test. During an image stream, it appears as if a large positive jitter value (~ 50 ms) is recorded followed by multiple negative jitter values (~ -20 ms) to compensate for the spreading. Although it is not entirely known what causes this jitter behavior, it is believed that because there is not sufficient network capacity to send the data required by the image stream, the MAC layer protocol of the 3G hardware must continuously wait until the medium is free before it can send a packet. This would result in the large positive values. Once the medium is free, it may then be able to send multiple packets at a time which will result in multiple negative jitter values. The exact cause of this behavior could not be determined without knowing the specific implementation details of the MAC layer protocol employed by the 3G hardware.



(a) Jitter PDF from Performance Testing at Channel Capacity



(b) Jitter PDF During Image Stream Testing (Black and White Scan 1 / 10 fps / QVGA)

Figure 133: Comparison of Forward Jitter between Performance Testing and Image Stream Testing

6.3.3 Satellite

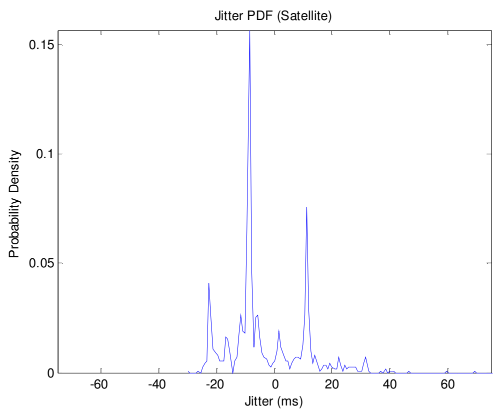
As explained in Section 6.2.6, the last image streaming test was conducted on the network emulator which was configured to replicate the behavior of Inmarsat's BGAN network. Because the results from the first round of tests on the emulator were all similar, only one round of image streaming tests was conducted. The results from these tests can be seen in Table 34.

Table 34: Complete Test Results for Satellite Image Stream Tests Run on Network Emulator

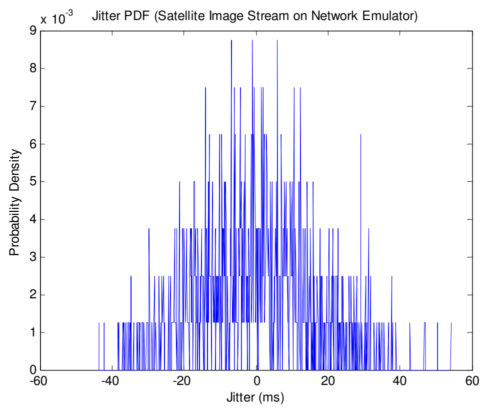
Image Stream	A	A	A	A	B	B	B	B	C	C	C	C	C	C	D	D	D	D	D	D
Tx FPS	7	10	15	15	7	10	15	15	7	10	15	15	30	30	7	10	15	15	30	30
Rx FPS	7	10	10	15	7	10	10	15	7	10	10	15	10	15	7	10	10	15	10	15
Resolution	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA	QVGA
Data Rate (kbps)	325	370	368	370	283	352	369	368	246	318	369	368	330	368	260	332	368	370	359	369
Packet Loss (%)	2.4	2.4	2.2	2.5	2.5	2.5	2.6	2.3	2.5	2.4	2.5	378	2.5	2.4	2.3	2.4	2.2	2.5	2.4	2.4
Average Jitter (ms)	12.6	12.9	13.3	12.6	13.1	12.9	13.2	12.8	13.6	13.2	13.1	360	13	13.3	13.5	12.5	12.7	12.9	12.9	13.1
95% Jitter Threshold (ms)	31.2	31	33.4	33.2	33.1	31.5	33.8	31.1	32.3	32.4	31.9	31.5	32.4	32.6	33.6	28.9	31.8	30.1	32.4	31.6
99% Jitter Threshold (ms)	41.1	39.5	44.1	43.1	42.5	39	43.4	40.9	43.2	41.7	42.1	43.7	42.7	43.8	42.9	36	41.2	39.1	42.5	41.3
Average Packet Size (Bytes)	1112	1104	1108	1031	1103	1106	1105	1093	1106	1095	1073	1081	1083	1025	1108	1112	1097	1096	1144	1073
Packet Size Std Deviation (Bytes)	272	274	265	374	281	273	271	289	292	303	326	313	331	375	298	295	302	305	237	318
Received Image Stream Quality	A	A	A	B	A	A	A	B	A	A	A	B	A	A	A	A	A	B	A	B

The results from these tests were very consistent throughout each of the tests. In all cases, a video stream with acceptable image quality (grade of “A” or “B”) could be transmitted between the client and server. The maximum average data rate achieved by any of the streams was 370 kbps excluding overhead. Therefore, it can be concluded that a consistent minimum bandwidth of around 390 kbps is sufficient to stream both Color Doppler and black and white ultrasound scans at 15 fps at QVGA quality.

Although the tests run over the satellite emulator came out well, it is difficult to determine how well the network emulator actually mimicked the true behavior the satellite channel. One obvious difference is that the overall round trip latency was reduced from around 2 seconds to 200 ms. As explained in Section 6.2.6, this was necessary in order to get Layered Media’s software to function properly. Another difference was that the bandwidth of the BGAN network varies with time as can be seen in Figure 117 (b). During the performance tests, the bandwidth routinely dropped for periods of a few seconds, which would inevitably lead to dropped frames if an image stream were being sent over the network. The network emulator does not have the ability to vary its bandwidth with time; rather it just keeps a consistent maximum bandwidth that cannot be exceeded. Lastly, even though the average jitter for the satellite network and the network emulator were the same (~13 ms), the distribution differs. The emulator injects a jitter to the packets with more or less a normal distribution while the true jitter behavior appeared much more random during the performance tests over the BGAN network. Figure 134 compares the jitter PDFs of both scenarios. Without actually conducting streaming tests over the BGAN network, it cannot be concluded how these factors would affect a live image stream.



(a) *True Jitter on BGAN Network Measured During Performance Testing*



(b) *Jitter used by Network Emulator During Image Stream Testing*

Figure 134: Comparison of True BGAN Jitter and Jitter Used By the Network Emulator

6.3.4 Conclusions

In addition to the screen recordings of the various received image streams, valuable information was gathered during this phase of testing. It was determined that the SNR of 802.11, which dictates the bandwidth of the channel, also affects an H.264 SVC image stream. Sufficiently high enough SNRs (35 dB) can stream video at VGA quality at 30 fps with no problems. 802.11 at lower SNRs can also produce usable image streams; however, periodic pauses in the video should be expected. Also, image streaming over AT&T's 3G network can be done at 10 fps at VGA quality using the H.264 SVC. Once the transmission rate exceeds 15 fps, degradation in the video quality can be expected. Lastly, the testing on the network emulator showed that a consistent minimum bandwidth of around 390 kbps is sufficient to stream ultrasound video at 15 fps in VGA quality using H.264 SVC.

Information on the packet sizes used in H.264 SVC video streams was also gathered during testing. The average packet size throughout all of the tests was around 1100 bytes with 90% of the packets falling between 1000 and 1260 bytes. It can also be concluded from the tests that the compression ratio using H.264 SVC was greater for the black and white scans than in was for the Color Doppler streams. This is supported by the fact that in virtually all cases, the Color Doppler videos required a higher average data rate than the black and white videos to transmit at the same frame rate and resolution. This occurred despite the fact that the individual frames of the AVI files were identical in size between the different scan types. Next, it was shown in the 3G streaming tests transmitting at 15 fps and decoding at 10 fps requires more bandwidth than does transmitting and receiving at 15 fps. Lastly, from the packet loss it can be concluded that the percentage of packets lost does not directly correspond to the quality of the received image stream. For example, the 802.11 tests conducted at 20 dB SNR produced much better image quality than did the 3G tests. Even though the more packets were lost on the 802.11 tests, which did lead to some dropped frames, this scenario provided better results than the 3G network, which had a small amount of packet loss but also lower bandwidth availability.

Finally, it should be emphasized that the ultrasound scans used during the image stream testing (echocardiographs) had a large amount motion compared to most other types of ultrasound scans. A less dynamic ultrasound scan, such as an obstetric sonograph, would typically have more correlation from frame to frame. This in turn would lead to a higher compression ratio using the H.264 SVC, producing lower overall data rates for such scans. This means that the image quality of other types of ultrasound scans may actually be better than were the echocardiographs due to the lower data rate requirements for less dynamic scan types.

6.4 Physician Evaluation

To determine the diagnostic value of the transmitted ultrasound streams, the screen recordings of the received image streams were given to # physicians for evaluation. The physicians first viewed the original AVI file containing the source ultrasound video. They were then given the various screen recordings that corresponded to that particular source video. After viewing both videos, the physicians were asked to give the transmitted ultrasound stream a score indicative of its image quality and diagnostic value. The scoring system that they were asked to used can be seen in Table 35.

Table 35: Scoring System for Physician Evaluation

Grade	Description
A	Received image stream is indistinguishable from the source video. Full diagnostic information is retained.
B	Received image stream is close to original, but some degradation is present. Full diagnostic information is retained.
C	Noticeable degradation present in received image stream. Most of the diagnostic information is retained.
D	Significant degradation in received image stream. Little to no diagnostic information is retained.

****

Currently in the process of having doctors at UMASS Memorial Medical Center view and score the recorded ultrasound clips. This section will be completed once their evaluation is completed.

****

6.5 Voice Streaming Considerations

In many instances, two-way voice communication will be necessary between the remote ultrasound operator and personnel at the base station. If a separate infrastructure or device is not already in place, voice communication may be done over the wireless link using IP packets. Although live voice testing was not conducted as part of this project, this section will discuss some considerations that must be examined when streaming real-time voice over IP networks (VoIP).

There are many methods that can be used to transmit real-time speech over IP networks. The main characteristics that dictate the network requirements of VoIP applications are the codec used to code and decode the voice data, the frame period, and the network protocols used to send and receive data. The two most common protocols used in VoIP applications are RTP and Session Initiation Protocol (SIP). Because the RTP has already been explained, this section will assume RTP is the protocol used to deliver speech frames. VoIP applications that use SIP share many similarities with those that use RTP; however some of the minor details may differ.

One of the main problems with sending voice frames over IP networks is the amount of overhead that is needed. Each RTP packet contains 40 bytes or 320 bits of overhead incurred from the IP (20 bytes), UDP (8 bytes) and RTP (12 bytes) headers. Because most applications use small frame sizes, sometimes the overhead can be as high as 200%. Typical VoIP systems use packets that are large enough to hold 20 to 30 ms of voice data resulting in transmission rates of between thirty and fifty packets per second. If fifty packets are sent per second, then approximately 16 kbps will be necessary just for protocol overhead (320 x 50) [14]. Table 36 shows some commonly used speech codec and their corresponding bit rates. It goes on to show what a typical frame period may be for each individual codec along with the resulting bandwidth required on an IP network assuming packets are sent using IP/UDP/RTP protocols.

Table 36: Common Speech Codecs and IP Bandwidth (Assuming RTP) [14]

Codec	**Codec Bit Rate (kbps)**	**Typical Frame Period (ms)**	**IP Bandwidth (kbps)**
G.711	64	20	80
G.723.1	5.6	30	16.27
	6.4	30	17.07
G.726	32	20	48
G.728	16	30	26.67
G.729(A)	8	20	24
GSM 6.10	5.6	20	21.6
	13	20	29

There are techniques that can be employed to reduce the IP bandwidth necessary for VoIP applications. One fairly obvious way to reduce protocol overhead would be to use larger packets. This would result in a smaller number of packets being sent per second which would reduce the percentage of bandwidth needed for overhead data. Although using larger packets may reduce IP bandwidth requirements, it can cause problems in real-time voice applications. Larger packets generally have longer delay times, increased jitter and a higher tendency for packet loss which all negatively impact VoIP systems.

Another technique used to lower the IP bandwidth requirements of VoIP applications is to compress the protocol headers. For example, one compression technique called cRTP (Compressed Real-time Transport Protocol) can compress the 40 bytes of IP/UDP/RTP headers down to 4 bytes. This significantly reduces the overhead as well as the IP bandwidth necessary to stream live voice data. In order to use RTP compression algorithms, both of the network endpoints need to be preconfigured to work properly. Also, some of the error detection and correction properties of the network protocols are lost when the headers are compressed. Using RTP compression algorithms can lower the required bandwidth close to that of the actual bit rate of the codec being used [54]. In general, a reliable VoIP application will need a minimum of about 8 to 10 kbps in each direction to successfully stream live voice data.

Even if there is sufficient bandwidth available to stream voice frames, the delay and jitter properties of the network can introduce problems. Excessive one-way delays in two-way voice applications can cause confusion between the speakers as to who should speak when. The International Telecommunication Union (ITU) considers network delay for voice applications in Recommendation G.114. This recommendation defines three bands of one-way delay as shown in Table 37 [24]. It should be noted that network delay is not the only source of one way delay in live voice systems. Additional factors such as coding/decoding delay, queuing delay, de-jitter buffering delay all contribute to the overall end-to-end delay as well.

Table 37: ITU One-way Delay Specifications [24]

Range in milliseconds	Description
0 – 150	Acceptable for most user applications.
150 – 400	Acceptable provided that administrators are aware of the transmission time and the impact it has on the transmission quality of user applications.
400+	Unacceptable for general network planning purposes. However, it is recognized that in some exceptional cases this limit is exceeded.

The last major factor that must be taken into account is network jitter. To remove variation in delay so that the audio output is played at a fixed rate, a de-jitter buffer is needed. Making the buffer too small will result in buffer overflows and discarded packets leading to gaps in the voice playback. If the buffer is too large, unnecessary delay is added to the system which can introduce problems. There are a few different techniques to determine the appropriate size of the de-jitter buffer that are commonly implemented in VoIP applications. One technique is to use a fixed size buffer that is equal to the mean jitter in the network. Another uses a buffer equal to the size of the nominal one way delay to remove delay variation. The last common method is to use an adaptive buffer that is dynamically increased when high jitter values are experienced and decreased when the variation in delay is low [24]. Although all of these methods have been effective in different circumstances, no single method will work for every type of network. VoIP applications must be tested live over real networks to determine if the de-jitter buffer is too small or large.

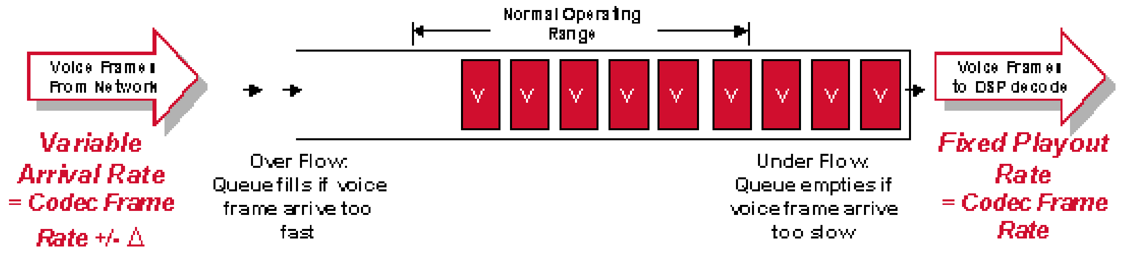

Normal Operating Range
Voice Frames From Network
Variable Arrival Rate = Codec Frame Rate +/- Δ
V
Over Flow: Queue fills if voice frame arrive too fast
Under Flow: Queue empties if voice frame arrive too slow
Voice Frames to DSP decode
Fixed Playout Rate = Codec Frame Rate

Figure 135: De-Jitter Buffer [24]

7 Conclusions and Recommendations

The goal of this project was to examine a number of different wireless communication options as candidates for possible integration into a mobile ultrasound system for use in remote data transmission applications. The wireless technologies that were researched included 802.11g, 3G cellular broadband, and Inmarsat's BGAN satellite network. To determine possible remote data applications for which each communication option may be useful, two phases of testing were conducted.

During the first phase, the general characteristics of the wireless channel were gathered. A client and server software application was written to measure and record various channel properties, such as the channel capacity (throughput), latency, packet loss and jitter. This information was essential to determine the capabilities of each of the wireless technologies. For network applications that are not real-time, such as downloading a static image or video, the information gathered during this phase of testing was helpful in predicting how long it would take to download a file of a specific size. It will also be useful for future network application developers to understand the dynamics of the link for which they are writing an application.

In the second phase of testing, the wireless links were tested for possible use in real-time network applications. During these tests, live ultrasound image streams were transmitted over the various links, and screen recordings were made for each of the received video streams. Additional data such as jitter, data rate and packet loss was also recorded. These tests helped determine if real-time image streaming using H.264 SVC was possible on the link, and if so, what type of resolution and frame rate it could support.

The first wireless option that was tested was the 802.11g standard. 802.11g is characterized by high data rates (> 2 Mbps) at a relatively short transmission range (< 100 m). The performance of 802.11g is heavily dependent on the signal-to-noise ratio present at the receiving node as adaptive data rate control algorithms adjust the transmission rate based on the received signal quality. For this reason, most of the data measured during the performance testing is in some way a function of signal-to-noise ratio. The live image stream tests showed that 802.11g with a sufficiently high enough SNR could easily support VGA quality video streams at 30 fps using H.264 SVC compression. As expected, as the SNR dropped to around 20 dB or so, degradation in the video stream began to appear due to fluctuating data rates and an increase packet loss. These 802.11g tests are specific to a given 802.11 chipset (Realtek RTL8187); however, similar performance should be expected among various 802.11 adapters. Finally, it was demonstrated that the use of an external 802.11 antenna could extend the range of acceptable SNRs for real-time media applications (> 20 dB) by a factor of around 2 over the case where no antenna is used. Table 38 summarizes the results obtained during 802.11 testing in this project.

Table 38: Summary of 802.11g Results

Channel Characteristic		**Value / Description**
Mean Throughput	Up/ Down	$BW(SNR) = 21.9 \bullet (1/2) \bullet (erf((SNR - 32.5)/11.5) + 1)$ [Mbps]
Mean TCP Throughput	Up/ Down	$BW(SNR) = 21.3 \bullet (1/2) \bullet (erf((SNR - 32.5)/11.9) + 1)$ [Mbps]
Mean Packet Loss	Up/ Down	$PL(SNR) = 143.9 \bullet \exp(-0.289 \bullet SNR) + 0.038$ [%]
Mean Delay	Up/ Down	$RTD(SNR) = 6212.2 \bullet \exp(-0.333 \bullet SNR) + 4.395$ [ms] $OWD \approx (1/2) \bullet RTD(SNR)$ [ms]
Mean Jitter (at Full Channel Capacity)	Up/ Down	$MJ(SNR) = 238.88 \bullet \exp(-0.194 \bullet SNR) + 0.321$ [ms]
Is the link symmetric?		Yes - if both the sender and receiver have the same received SNR
Does signal strength affect performance?		Yes - Significantly
Is Throughput affected by packet size?		Yes – smaller packets lead to lower data rates
Is Latency affected by packet size?		Not significantly
Transmission Range		Good SNRs can be achieved up to 100 meters using external antennas in an open environment. Performance degrades as range is extended.
Maximum image streaming capabilities		30 fps / VGA resolution

Restrictions on image streaming	Lower data rates cause by low SNRs can cause frames to be dropped and corruption to the received image stream
Cost	Cheap. 802.11g adapters: ~$60. Unlimited data usage.

The next wireless technology that was researched was AT&T's 3G HSDPA network. One advantage of 3G over 802.11 is the distance between the mobile ultrasound unit and base station is not a factor as long as the remote system is located within the coverage area of a 3G network. Performance tests showed the 3G network had a fairly consistent bandwidth around 380 kbps on the uplink and 1300 kbps on the downlink. These values did not vary significantly even when the received signal strength was changed or the remote system was placed in a mobile environment. For telemedicine applications, this is a positive characteristic because users do not need to be concerned about varying network performance based on location or mobility.

However, the latency across the 3G network is significantly higher than 802.11 as well as most other physical layer options. During tests, round trip times of close to 400 ms were routinely experienced. The real-time image streaming tests run over the 3G network showed that the capacity of the network is right around the threshold of the data rate necessary for H.264 SVC needs to transmit a QVGA quality video at 15 fps. For the most part, the network could handle QVGA at 10 fps but image quality started to breakdown at 15 fps. When the network capacity was increased by 10 to 20 kbps on the network emulator for satellite testing, QVGA resolution at 15 fps was possible. Table 39 provides a summary of the results gathered during testing of AT&T's 3G network.

Table 39: Summary of AT&T's 3G Network Results

Channel Characteristic		**Value / Description**
Mean Throughput	Up	380.3 kbps
	Down	1361 kbps
Mean TCP Throughput	Up	336.8 kbps
	Down	971.4 kbps
Mean Packet Loss	Up	0-4% depending on channel utilization
	Down	
Mean Delay	Up	281.5 ms
	Down	110.2 ms

Mean forward jitter at full channel capacity	6.7 ms
Is the link symmetric?	No
Does signal strength affect performance?	No
Is Throughput affected by packet size?	Not significantly
Is Latency affected by packet size?	Yes – smaller packets have lower latency
Transmission Range	Depends on coverage of network. AT&T currently available in most metropolitan areas.
Maximum image streaming capabilities	10 fps / QVGA resolution
Restrictions on image streaming	Depending on the dynamics of the ultrasound scan being transmitted, 15 fps at QVGA resolution may be possible.
Cost	Fair. USB modem for 3G network: ~$300. Monthly data plan for unlimited data usage: ~$80

The last wireless option that was tested was Inmarsat's BGAN satellite network. Due to limits on the amount of data that could be used on the satellite networks, five sets of performance tests were conducted on the network. During these tests, the packet latency that was exhibited was very high compared to most other transmission media. Round trips times between 1.5 and 2 seconds were typical over the network. The throughput of the uplink experienced during testing was slightly higher than that seen on the 3G network. The average throughput of the uplink was around 400 kbps; however during many of the tests, there were periods of time where the bandwidth would suddenly drop much lower than this value. This is a bad characteristic as far as streaming media applications go, as sudden drops in bandwidth will inevitably lead to dropped packets and/or frames.

Unfortunately, real-time video testing could not be carried out over the network because there were problems keeping a connection to the server due to the high network latency. Instead, a network emulator was configured to simulate the BGAN network, and image stream tests were conducted. Image streams at 15 fps at QVGA quality could successfully be transmitted; however it is difficult to determine how well the emulator actually mimics the true behavior of the satellite network. The main advantage that the BGAN network has over the other wireless technologies is that users have near global coverage meaning the distance between the remote system and the base station in insignificant. One of the disadvantages of was the high cost of using the system. At around $7/MB, users can expect to pay $21 per minute of streaming video at 400 kbps. Table 40 contains a summary of the results obtained during testing over the BGAN network.

Table 40: Summary of Inmarsat BGAN Results

Channel Characteristic		**Value / Description**
Mean Throughput	Up	406.2 kbps
	Down	-
Mean TCP Throughput	Up	235.9 kbps
	Down	272.5 kbps
Mean Packet Loss	Up	0-3% depending on channel utilization
	Down	
Mean Delay	Up	1120.3 ms
	Down	718.3 ms
Mean forward jitter at full channel capacity		12.68 ms
Is the link symmetric?		No
Does signal strength affect performance?		No (only tested SNRs between 50 and 55 dB)
Is Throughput affected by packet size?		Yes – larger packets exhibited a slightly higher throughput
Is Latency affected by packet size?		Yes – smaller packets had slightly lower latency
Transmission Range		Global
Maximum image streaming capabilities		15 fps / QVGA (based on network emulator tests)
Restrictions on image streaming		Although the network appears to have sufficient bandwidth to stream at 15 fps / QVGA, the performance is unknown on a true satellite link (used emulator)
Cost		Expensive. BGAN terminal: ~$3500 to purchase, ~$10/day to rent. Data: ~$7/MB

Although a lot of useful information was obtained during the two testing phases of this project, there are still some areas where future work may want to expand upon. One obvious task that would be very useful is to find or create an application that is able to use H.264 SVC over a satellite link. At the time of this project, the only accessible software able to stream video using H.264 SVC were Layered Media's frameclient (server) and Advanced Client (client) programs. Using this software, a connection could not be made between the client and server to create a video stream. It is presumed that the high latency of the link is to blame; however this problem was never resolved. It would most definitely be beneficial to either create a custom application (currently in progress) or find another application capable of keeping a connection and transmitting a live image stream over a satellite network. This would show the true behavior of an H.264 image stream over a satellite network.

Another issue to keep in mind is that AT&T is currently making enhancements to its 3G network which should significantly improve its performance. During discussions with AT&T, they plan on matching the performance of the uplink to the current performance of the downlink by 2009. This would increase the average data rate from around 380 kbps to around 1300 kbps which would definitely allow for a higher quality image stream to be transmitted from the mobile ultrasound system. In the following year (2010), they plan on increasing the data rates of both the uplink and downlink to somewhere around 5 Mbps. This would enable a host of network applications that are not currently possible on the 3G network such as simultaneous two-way voice and video. Although the network is not currently capable of such applications, the network should be retested once the upgrades are made.

Another useful task would be to conduct additional image stream tests with other types of ultrasound scans than echocardiographs. As previously described, the echocardiographs have a high amount of motion relative to other types of ultrasound scans. Scan types where the transducer is moved slowly over the body surface, such as an abdominal scan, contain a higher amount of correlation from frame to frame and should have a higher compression ratio using H.264 SVC. It would be beneficial to examine the minimum required data rates for different types of ultrasound scans using H.264 at various frame rates and resolutions.

Another recommendation for the real-time image stream testing is to find or create a more standardized method of classifying the image quality of the received video stream. In this project, a qualitative assessment of the image quality was made. Quantitative metrics such as packet loss and jitter were also obtained; however, it was difficult to create a clear correlation between the qualitative and quantitative date. Additionally, because the screen recordings of the received image stream were compressed by Camtasia, comparison on a frame-by-frame basis was not possible as the source video and received image stream were in totally different formats and frame rates. In going forward, some sort of standardized comparison method should be used when contrasting the quality of the received video stream to the source ultrasound scan.

Lastly, since the commencement of this project, a number of new wireless technologies have begun to emerge. IEEE 802.16 (*WiMAX*) appears as though it could be very useful in a number of telemedicine applications. 802.16.d can deliver data at up to 75 megabits per second over a range of 70 km between fixed points while the mobile version of WiMax (802.16.e) can provide 15 Mbps over a 4 km radius [59]. Also, IEEE 802.22 is a working group aimed at creating standards for *Wireless Regional Area Networks* (WRAN). The PHY layer implementation for this standard could provide data rates up to 19 Mbps at distances up to 30 km [60]. One last wireless option that could possibly be used for remote data transmission on the ultrasound system is data radios. Data radios can provide IP connectivity over a greater distance than the 802.11 standard, but normally at lower data rates. It could be worthwhile to examine the performance of some of these additional wireless technologies and evaluate the possibility of using them for remote data applications.

In closing, this project has provided an in-depth analysis of three different wireless technologies and elucidated how effectively they could be incorporated into a mobile ultrasound system for remote data applications. The information gathered during testing revealed the abilities and limitations of the different technologies. This information was helpful in determining what the system is currently capable of in terms of real-time video applications and will be valuable in the future for those trying to develop network applications for telemedicine procedures.

References

[1]	Wikipedia. “Medical Ultrasonography.” http://en.wikipedia.org/wiki/Medical_ultrasonography. 2008.
[2]	Terason, “Terason Ultrasound System User Guide”, June, 2005.
[3]	Douglas S. Richards, http://www.obgyn.ufl.edu/ultrasound/1ObtainImage/1Equipment/2frequency.html, 2003.
[4]	Cordeiro, Phil. “Design of a Wearable Ultrasound System.” WPI. August, 2006.
[5]	http://www.aium.org/aboutAIUM/timeline/1950.asp
[6]	http://zoot.radiology.wisc.edu/~block/Med_Gallery/us_baby.html
[7]	http://www.medison.ru/uzi/eho404.htm
[8]	http://www.ultrasoundjobsstat.com/4.html
[9]	Kyriacou, E. et al. “Multi-purpose HealthCare Telemedicine Systems with Mobile Communication Link Support.” BioMedical Engineering OnLine. March, 2003. http://www.biomedical-engineering-online.com/content/pdf/1475-925X-2-7.pdf
[10]	Sentinel Imaging. http://www.sentinelultrasound.com/, 2008
[11]	Dalys Sebastian, “Development of a Field-Deployable Voice-Controlled Ultrasound Scanner System”, WPI, 2004.
[12]	Carsten Poulsen, “Design of 2nd Generation Wearable Ultrasound System”, WPI, Worcester, MA, Jul. 2004.
[13]	eMagin, http://www.3dvisor.com/products.php, 2006.
[14]	http://www.erlang.com/bandwidth.html
[15]	Sun Microsystems. “Java Platform SE 6 API Specification.” 2008. http://java.sun.com/javase/6/docs/api/
[16]	http://compnetworking.about.com/library/graphics/basics_osimodel.jpg

[17]	Tanenbaum, Andrew. “Computer Networks (4th Ed.)” Prentice Hall PTR. 2002.
[18]	http://www.ssfnet.org/Exchange/tcp/tcpTutorialNotes.html
[19]	Ahonen and Koskelainen. “Transport Control Protocol.” University of Helsinki. 1998. http://www.netlab.hut.fi/opetus/s38130/s98/tcp/TCP.htm
[20]	Gannon, Thomas. “Chapter 12 - TCP Traffic Control.” EE 506 Lecture. WPI. 2007.
[21]	http://research.edm.luc.ac.be/jori/thesis/onlinethesis/images/chapter_2/fig_udp_header.pn g
[22]	Wikipedia. “Throughput.” http://en.wikipedia.org/wiki/Throughput. 2008.
[23]	http://www.ixiacom.com/library/test_plans/display?skey=ixchariot_for_wireless
[24]	Cisco. “Understanding Delay in Packet Voice Networks: Document ID: 5125” 2007. http://www.cisco.com/warp/public/788/voip/delay-details.html
[25]	http://download.netbeans.org/netbeans/6.0/final/
[26]	http://dast.nlanr.net/Projects/Iperf/
[27]	Microsoft. “Microsoft Windows Server 2003 TCP/IP Implementation Details.” 2003. http://technet2.microsoft.com/windowsserver/en/library/823ca085-8b46-4870-a83e-8032637a87c81033.mspx?mfr=true
[28]	http://www.ntp.org/documentation.html
[29]	Garmin. “GPS 18 Technical Specification.” 2005. http://www8.garmin.com/manuals/425_TechnicalSpecification.pdf
[30]	http://www.cnssys.com/Tac32/
[31]	McLaren, Paul. “Telemedicine and Telecare: what can it offer mental health services?” Advances in Psychiatric Treatment vol. 9 pp 54-61. 2003.
[32]	ECG. “Telemedicine in the Ambulatory Setting: Trends Opportunities and Challenges.” ECG. 2007.

[33]	Burkle, Frederick and Garshnek, Victoria. “Telemedicine Applied to Disaster Medicine and Humanitarian Response: History and Future.” Proceedings of the 32nd Hawaii International Conference on System Sciences – 1999.
[34]	Chatterjee, Samir and Tulu, Bengisu. “A Taxonomy of Telemedicien Efforts with respect to Applications, Infrastructure, Delivery Tools, Type of Setting and Purpose.” Proceedings of the 38th Hawaii International Conference on System Sciences – 1999.
[35]	Chu, Yuechun and Ganz, Aura. “Mobile Telemedicine System Using 3G Wireless Networks.” Business Briefing: US Healthcare Strategies. 2005.
[36]	Wikipedia. “Comparison of wireless data standards.” http://en.wikipedia.org/wiki/Comparison_of_wireless_data_standards. 2008
[37]	Lozano, Fernando. “Introducing NetBeans C/C++ Pack.” http://www.netbeans.org/community/magazine/html/03/c++/. 2008.
[38]	Committee on Evaluating Clinical Applications of Telemedicine. “Telemedicine: A Guide to Assessing Telecommunications in Health Care.” Washington, D.C.: National Academy Press, 1996.
[39]	Wikipedia. “List of device bandwidths.” http://en.wikipedia.org/wiki/List_of_device_bandwidths. 2008.
[40]	Pahlavan, Kaveh and Levesque, Allen. “Wireless Information Networks.” Wiley-Interscience. 2005.
[41]	Inmarsat. “BGAN: Globabl voice and broadband data” 2008. http://www.inmarsat.com/Downloads/English/BGAN/Collateral/bgan_overview_brochure_EN.pdf?language=EN&textonly=False
[42]	Globalstar. “GSP-1700 Satellite Phone System.” 2008. http://common.globalstar.com/doc/common/en/products/gsp1700_brochure_gusa.pdf
[43]	Broadcom. “IEEE 802.11g: The New Mainstream Wireless LAN Standard.” Irvine, CA. July, 2 2003.
[44]	Wikipedia. “IEEE 802.11n.” http://en.wikipedia.org/wiki/802.11n. 2008.

[45]	Pavon, Javier and Choi, Sunghyum. “Link Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement.” IEEE. 2003.
[46]	Pham, Peter. “Comprehensive Analysis of the IEEE 802.11.” Mobile Networks and Applications 10, 691-703. 2005.
[47]	Wang, Tianlin and Refai, Hazem. “The Development of an Empirical Delay Model for IEEE 802.11b/g Based on SNR Measurements.” 2005 International Conference on Wireless Networks, Communications and Mobile Computing.
[48]	Poretsky, S. et. Al. “RFC 4689 – Terminology for Benchmarking Network-layer Traffic Control Mechanisms.” Network Working Group. 2006.
[49]	Spirent Communications. “Measuring Jitter Accurately.” Spirent Communications White Paper. May, 2007.
[50]	Rysavy, Peter. “Data Capabilities: GPRS to HSDPA and Beyond.” Rysavy Research. September, 2006.
[51]	Hyperlink Technologies. http://www.hyperlinktech.com/web/antennas. 2006.
[52]	Data-Alliance. http://www.data-alliance.net/servlet/the-75/USB-Wireless-802.11g-Adapter/Detail. 2008.
[53]	AT&T. http://www.wireless.att.com/businesscenter/plans/connections-coverage.jsp?WT.svl=title. 2008
[54]	Sze, H.P. et al. “A Multiplexing Scheme for H.323 Voice-Over-IP Applications.” IEEE Journal on Selected Areas in Communications. Vol. 20, No. 7. September 2002.
[55]	IETF RFC3550, “RTP: A Transport Protocol for Real-Time Applications”, 2003.
[56]	Wainhouse Research. “A Ready Market: Indroducing H.264 SVC.” March 2006.
[57]	National Institute of Standards and Technology. “NistNET.” 2007. http://snad.ncsl.nist.gov/nistnet/

[58]	Moisey, Frederic. “ezyfit. A Free Curve Fitting Tool for Matlab.” Lab. FAST - University Paris Sud. March, 2008.
[59]	Ma, Liangshan and Jia Dongyan. “The Competition and Cooperation of WiMAX, WLAN and 3G.” Beijing Consulting and Design Institute of P&T, P.R. China. 2005
[60]	Cordeiro, C et. al. “IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios.” IEEE Journal of Communications, Vol 1, No 1. April 2006.
[61]	Xiao, Yang and Rosdahl, Jon. “Throughput and Delay Limits of IEEE 802.11.” IEEE Communications Letters, Vol. 6, No. 8. August 2002.
[62]	Borgonovo, Flaminio et. al. “Packet Service in UMTS: Delay-Throughput Performance of the Downlink Shared Channel.” Computer Network Vol. 38, Issue 1. January 2002.

Appendix A – IEEE 802.11g Performance Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from 802.11g performance testing. The following files can be found in this appendix:

80211_Throughput.doc	802.11g throughput test results
80211_Delay.doc	802.11g delay test results
80211_Throughput_vs_PS.doc	802.11g throughput vs. packet size results
80211_Delay_vs_PS.doc	802.11g delay vs. packet size results

Appendix B – 3G Performance Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from 3G performance testing. The following files can be found in this appendix:

3G_Throughput.doc	3G throughput test results
3G _Delay.doc	3G delay test results
3G_Throughput_vs_PS.doc	3G throughput vs. packet size results
3G_Delay_vs_PS.doc	3G delay vs. packet size results

Appendix C – Satellite Performance Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from satellite performance testing. The following files can be found in this appendix:

Satellite_Throughput.doc	Satellite throughput test results
Satellite _Delay.doc	Satellite delay test results
Satellite _Throughput_vs_PS.doc	Satellite throughput vs. packet size results
Satellite _Delay_vs_PS.doc	Satellite delay vs. packet size results

Appendix D – IEEE 802.11g Image Stream Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from 802.11g image stream testing. The following files can be found in this appendix:

80211_35_Test1.doc	Test1: 802.11 image stream test results with an SNR of 35 dB
80211_35_Test2.doc	Test2: 802.11 image stream test results with an SNR of 35 dB
80211_20_Test1.doc	Test1: 802.11 image stream test results with an SNR of 20 dB
80211_20_Test2.doc	Test2: 802.11 image stream test results with an SNR of 20 dB

Appendix E – 3G Image Stream Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from 3G image stream testing. The following files can be found in this appendix:

3G_75_Test1.doc	Test1: 3G image stream test results with an received signal strength of -75 dBm
3G_85_Test2.doc	Test1: 3G image stream test results with an received signal strength of -85 dBm

Appendix F – Satellite Image Stream Test Results

This appendix is contained on the DVD that accompanies this thesis. It contains the complete results from satellite image stream testing. The following files can be found in this appendix:

Sat_NE_Test1.doc	Test1: Satellite image stream test results conducted on the network emulator

Appendix G – IEEE 802.11g Image Stream Recordings

This appendix is contained on the DVD that accompanies this thesis. It contains all of the screen recordings that were made during 802.11 image stream testing. The names of the AVI files take the following form:

(Tx Frame Rate)_(Ultrasound Loop)_(Rx Frame Rate)_(Resolution).avi

Appendix H – 3G Image Stream Recordings

This appendix is contained on the DVD that accompanies this thesis. It contains all of the screen recordings that were made during 3G image stream testing. The names of the AVI files take the following form:

(Tx Frame Rate)_(Ultrasound Loop)_(Rx Frame Rate)_(Resolution).avi

Appendix I – Satellite Image Stream Recordings

This appendix is contained on the DVD that accompanies this thesis. It contains all of the screen recordings that were made during satellite image stream testing on the network emulator. The names of the AVI files take the following form:

(Tx Frame Rate)_(Ultrasound Loop)_(Rx Frame Rate)_(Resolution).avi