University of Southern Queensland
Faculty of Engineering & Surveying

Issues with Wireless 802.11 Networks

A dissertation submitted by

Benjamin Gray

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Electrical & Electronic Engineering

Submitted: October, 2006

Abstract

Wireless networking is a relatively new technology that is rapidly replacing conventional wired network infrastructure. Whilst wireless technology has dramatically increased the portability of computer networks, it has come with a performance cost. There are inherent problems due to the nature of wireless transmission and to the imperfect implementation of the Transmission Control Protocol for non-wired networks.

For a great portion of the development of the TCP protocol, wired networks were the de facto networking standard. As a result, the TCP protocol's algorithms and parameters have been optimised to suit wired networks and the implementation for wireless networks is far from perfect. A considerable improvement in performance may be possible if the parameters / algorithms of the TCP protocol are modified in a way that better suits the nature of wireless networks.

A specific problem of 802.11 wireless networks is its ability to handle real-time voice applications. Wireless networks have a shallow capacity for concurrent 'Voice over IP (VoIP)' sessions. In the presence of TCP traffic sources, the performance of VoIP becomes unacceptable. This project has its focus on investigating the causes of issues with VoIP over wireless 802.11 networks, and improving the performance by means of a prioritised packet scheduling algorithm, and changes to certain networking parameters. By setting the RTS / CTS threshold to a greater value than the VoIP packet length, the call capacity of 802.11 access points was approximately tripled.

Using a prioritised packet scheduling algorithm, TCP traffic was able to coexist with VoIP calls on the same access point up to the call capacity whilst maintaining QoS contracts for all VoIP connections. The improved performance of VoIP in the presence of TCP was not at the expense of TCP throughput, which was unnaffected by the prioritised packet scheduling algorithm.

University of Southern Queensland
Faculty of Engineering and Surveying

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof R Smith
Dean
Faculty of Engineering and Surveying

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged. I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Benjamin Gray 001222646

Acknowledgments

This project was made possible by through the support and enthusiasm of various people.

I'd like to thank my project supervisor Dr. John Leis, for taking on another project student when his plate was already full (I hope I didn't make myself a burden). Your guidance, direction and words of wisdom over the course of this project helped and influenced my work a great extent.

Gratitude needs to be expressed to a few of my friends, Greg, Matt, Rudolphe, and Sherwin for listening to my incessant ravings about wireless networking. Your suggestions and food for thought helped me develop the skills I needed to do this work.

Finally, I'd like to thank my girlfriend Johana for all her support and patience during the course of the project. Your encouragement and kind words motivated me throughout the year and helped me polish the final product.

Benjamin Gray

University of Southern Queensland
October 2006

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

A wireless 802.11 network is essentially a group of computers that can communicate without wires. Through the transmission of radio frequency signals, wireless 802.11 networks have essentially the same functionality of a wired *Local Area Network (LAN)*. Wireless 802.11 networks have enjoyed growing popularity, especially in the last few years. Nowadays, wireless 802.11 devices are encountered almost everywhere. It is likely that there is an 802.11 access point in your workplace providing you with mobile access to corporate data. Wireless access points are commonly found in cafes, restaurants, airports, hotels and universities, serving the internet in what is called a *hot spot*. With the rapid emergence of portable devices such as *Personal Digital Assistants (PDAs)*, the popularity of 802.11 wireless networks can only increase.

Despite the obvious popularity of 802.11 wireless networks, there are serious issues supporting real-time applications that are sensitive to packet loss and delay. An exceptional example of issues with wireless 802.11 networks is its failure to support TCP and real-time traffic such as *Voice over Internet Protocol (VoIP)* simultaneously. Voice over Internet Protocol is the routing of voice conversations over IP-based packet switched networks. Voice over IP is becoming an important internet application, and is likely to replace the *Public Switched Telephone Network (PSTN)* in due course. VoIP calls are usually free or inexpensive, and can provide near toll quality on a broadband internet connection. Commercial VoIP applications such as Skype^{TM} are already in place to take advantage of this growing industry. The emergence of wireless 802.11 networks has opened up new possibilities for cheap mobile communication. However, the call capacity of 802.11 access points is quite low and heavy TCP traffic cripples VoIP. For the bulk of this project, the focus will be on the performance of VoIP over wireless due to its popularity, and performance problems.

1.2 Issues with 802.11 Performance & Current Research

Wireless 802.11 networking infrastructure is rapidly replacing conventional wired 'ethernet' networks on a global scale. With the increasing popularity of VoIP as an alternative to PSTN telephony, solutions to issues with VoIP on wireless networks are being saught. The current research into this area has not offered the scalable solutions to the problem at hand. From the research completed in this project, current solutions to the problem included

- An implementation of token ring for 802.11
- Modifying the 802.11 protocol
- Using commercial VoIP access points such as Meru^{TM} access points

Although implementing a token ring system for 802.11 could work when the number of wireless nodes is low, the technique does not appear to be scalable. In this solution, a 'token' is passed by the access point to all associated nodes sequencially (Mustafa Ergen, 2003). Upon receiving the token, the node may transmit for a given time period or choose to forfeit the token. Major scalability issues of this technique may be become evident with tens of nodes, where a significant amount of time is wasted in passing the token to nodes that do not wish to transmit. It also ignores problems that could occur if the token were'dropped'

Modification to the 802.11 protocol is the perfect solution to the *Quality of Service (QoS)* issues in VoIP, and has already been accomplished by the IEEE. A new ammendment ('e') to the protocol was given final approval late 2005, and devices supporting this standard should become mainstream. Apart from already being done by the IEEE, modifying the 802.11 MAC protocol removes compatibility with the 802.11 standard (except for ammendments made by the IEEE), which leads to compatibility issues. The author chose not to make modifications at the MAC layer for these reasons and because significant performance increases may be had without changing the way things work at the MAC level.

Commercialised AP's have been designed to solve this very problem. Some commercial solutions to the VoIP over wireless problem have shown impressive performance and advanced features such as handoff capability and call synchronisation. Meru's enterprise access point was found to be an excellent solution for VoIP on 802.11 networks, at a price. Although being a viable solution to the QoS problem for VoIP, it did not cater for existing wireless infrastructure.

1.3 Performance Improvement and Compatibility

By changing various parameters of wireless 802.11 networks and the implementation of a prioritised scheduler, significant improvements in access point call capacity and the QoS can be realised. By setting the value of the *Requeset to Send / Clear to Send (RTS / CTS)* theshold to a value above the size in bytes of the VoIP packet size, the call capacity of 802.11 access points can be approximately tripled. Setting the *Maximum Transmissible Unit (MTU)* to a lower value than the default of 1500 bytes (as on most access points) facilitated better flow for smaller packets. An MTU of 960 marginally improved the consistency of the VoIP packet arrival delay. Protection measures of 802.11g to allow backwards compatibility with 802.11b networks were found to be a major hinderence. It is definitely advisable to enable the non-protected mode for 802.11g networks. To solve the problem of VoIP not being able to coexist with existing network traffic (particularly TCP traffic sources), a prioritised scheduler was developed as a network component for the NS-2 network simulator. Large performance increases were realised with no modifications to the 802.11 protocol in simulation.

1.4 Viability of Solutions

From research into the area of issues with 802.11 performance, it became appearant that other solutions had been offered in the past. However, none of the solutions offer a low cost solution on existing hardware for any 802.11 network. People who could benefit from the outcomes of this project are those with considerable 802.11 infrastructure in

place, and a desire to route voice conversations over the network to better utilise the network resource. The cost to implement a majority of the performance enhancements is close to zero, with the possible exception of the implementation of a prioritised packet scheduler. In order to implement this on an access point, an individual would require access to the firmware sources and software development skills in C/C++. Provided this is at hand, the priorised packet scheduler is an efficient and cost effective solution to the performance problems with VoIP over 802.11 networks.

1.5 Objectives

The main objectives of this project were to:

1. Establish that issues with wireless 802.11 networks exist and determine leads for further studies
2. Experimentally verify the existence of issues with performance via network simulation
3. Identify problem areas and possible solutions
4. Modify networking components for the ns-2 network simulator to achieve better network performance
5. Critically evaluate the effect of these alterations on performance via simulation
6. Propose improvements to increase the performance of wireless networks
7. Implement the improvements / adaptations on a wireless network testbed

1.6 Overview

A majority of the work in this project was done using the NS-2 network simulator to simulate and analyse the way VoIP packets traverse the communications medium. Via

analysing the impact of changing key network parameters on VoIP call capacity and QoS, it was possible to determine solutions to the problem at hand.

The second chapter was written to give the reader the neccessary backround information to appreciate the work undertaken in this project. A review of current research in the area is given, including opinons of experts found in journal articles, texts, other dissertations and PhD theses.

The main aim of this project was to research performance issues that exist in 802.11 networks with motivation to design and test enhancements to the way wireless networks operate. Research was undertaken into the 802.11 protocol, the TCP/IP protocol and Voice over IP technology, and serious performance limitations were found. Chapter three covers the main performance issues inherent to wireless 802.11 networks.

The purpose of simulating Vo802.11 was to reproduce performance issues in wireless 802.11 networks via simulation. This would serve to further verify the existance of performance issues, and expose the mechanisms for said issues. Chapter four is concerned with the efficiency and effectiveness of various wireless networking arrangements to carry VoIP, analysed using the NS-2 network simulator.

After verifying the existance of performance issues identified in chapter three, improvements to the way wireless networks were identified & implemented in the form of a *prioritised packet scheduler*. In chapter five, the methods for improvement were discussed and the expected performance benefits are analysed. A detailed discussion of the implementation of performance improvements to wireless 802.11 networks follows, with simulation results to demonstrate the impact of said improvements.

Chapter six begins by summarising the important findings of this project with a critical analysis of the level of achievement had. Recommendations for performance improvements for VoIP over 802.11 networks is despensed, and the chapter concludes with a statement of the shortcomings of this project and leads for further work.

Chapter 2

Literature Review

2.1 Wireless Networks

Wireless networking is a technology that allows computers to communicate via the transmission of radio frequency signals. The technology has been around in some form or another for over thirty years. The University of Hawii networked four computers on separate islands without wires in 1970 (dubbed ALOHANET) (Thurwatcher, 2002, pg 98), which some consider the birth of wireless networking. Despite existing for over three decades, wireless technology has only really been mainstream for the last 12 years and is considered maturing technology.

Wireless networks are rapidly replacing conventional wired networks for business, educational and home computing on a global scale. People are enjoying cost benefits of wireless networks which don't require any cable to be laid and concealed, and are accessible inside a radius of 45-90 metres (Ohrtman, 2004, pg 239-40). In the same way that mobile phones gave people the portability and flexibility to 'talk anywhere, anytime', wireless technology is set to revolutionise the way we use computers (Wong, 2005, pg 1-6).

Many wireless standards exist for wireless network interfaces, but the most recent and widespread are the 802.11x standards designed by the IEEE (Ohrtman, 2004, pg 1-3, 6). Wireless 802.11 [a, b, g] and the upcoming 'n' networks will be considered in this project. It should be noted that the IEEE consider there to be only one standard, which is 802.11 with the different letters on the end denoting an amendments to the protocol.

2.1.1 802.11 Networks

In 1997, the 802.11 protocol was accepted by the Institute of Electrical & Electronic Engineers and has become the standard for wireless networking. In its original form, the protocol allowed wireless data transfers at 1-2 Mbps and used

a 2.4 GHz carrier signal (Gast, 2002, pg 6). The first standard featured frequency-hopping and direct-sequence modulation techniques. In 1999, ammendments (a) & (b) were accepted. The specifications for 802.11a required the carrier frequency to operate at 5 GHz and increased the bandwidth available to 54 Mbps (IEEE, 1999). At the same time, the specifications for 802.11b were finalised and allowed transmission rates of up to 11 Mbps to be achieved at the same carrier frequency as the original specification (2.4 GHz).

Although the (a) & (b) ammendments were finalised together in 1999, technical limitations delayed the arrival of products conforming to the 802.11a specification, and far more 802.11b products have been purchased by consumers (Prasad, 2005, pg 35). Late 2002, the approval of the 802.11g draft saw yet another standard come to light. Whilst the standard is fully backwards compatible with 802.11b networks, it offers the same bandwidth of 802.11a networks (54 Mbps). It also has the advantage of operating at the same frequency of 802.11b networks (2.4 GHz) and is set to replace them entirely (Ohrtman, 2003, pg 20).

2.1.2 Types of 802.11 Wireless Networks

There exists two subclassifications of wireless 802.11 networks, namely Ad Hoc and Infrastructure networks. Both types consists of nodes with wireless network interface cards and infrastructure networks have one or many 'access points' (AP's). Each arrangements has advantages and disadvantages, the suitability of the network arrangement to its application must be given careful consideration. A detailed description of the two wireless 802.11 network types follows.

2.1.3 Ad Hoc Networks

In ad-hoc networking, an AP is not used to regulate the transmission of data over the network. Individual stations communicate with each other directly, and are not able to transmit to stations that are out of range. Ad-hoc networking

is the default networking topology for wireless devices. In the absence of an AP, compatible 802.11 stations will rapidly form a usable ad-hoc network. If an AP becomes available, the base stations will reconfigure themselves to direct all communications to the access point (Thurwatcher, 2002, pg 504).

Ad-hoc networks have the advantages of being easier and cheaper to set up and administrate. Unfortunately, ad-hoc networks have serious drawbacks. Each station may only transmit to stations that are within its own transmission range. This seriously limits the scalability of ad-hoc networks. It becomes neccessary to use an infrastructure type wireless network for more complex networking arrangements.

2.1.4 Infrastructure Networks

Infrastructure networks are characterised by the presence of an access point. All wireless traffic from stations is directed to access points for these networks. The access points serve to:

1. Forward communications to other wireless nodes that are in or out of range of the originating base station
2. Provide a gateway to other wired / wireless networks
3. To manage the association of stations, so all nodes can 'see' and communicate with each other

Infrastructure wireless networks offer practically infinite scalability. Multiple access points can provide wireless network coverage over large areas in a 'Lilly Pad', or 'Hot-Spot' type arrangement (Thurwatcher, 2002, 504-10).

2.1.5 Problems Inherent to Wireless Transmission

Wireless networks have several performance issues due to the nature of wireless transmissions. When compared to wired networks, wireless networks have lower data rates and throughputs, higher packet loss rates and less reliable connectivity (Thurwatcher, 2002, pg 539).

2.1.6 Packet Loss / Corruption / Latency

As stated by Ohrtman (2003, pg 177) in any IP network, a percentage of the packets potentially may be dropped or delayed. This is especially true in times of congestion. If a base station receives packets more quickly than it can forward them to the appropriate recipient, packets are queued for transmission. When the capacity of the queue is exceeded, packets must be 'dropped'. Packet loss is undesirable, and can cause poor performance in some networking applications. An example of an application that is sensitive to packet loss is VoIP (Ohrtman, 2004, pg 159).

Packet corruption on wireless networks usually results from interference by other stations or electronic devices. As a medium, air is much less reliable and predictable for transmission of signals than wires. Stations sometimes interfere with the transmission of other stations due to bad timing (Gast, 2002, pg 26). Interference may also be caused by transmissions containing no intelligence, such as that of a microwave ovens (which operate on the same band as 802.11 b & g networks) or due to the operation of power tools.

2.1.7 Carrier Sense Multiple Access / Collision Avoidance

Considerable measures are taken by 802.11 networks to avoid collisions at the expense of additional overhead. The primary Quality of Service (QoS) mechanism in Voice over IP is the *Carrier Sense Multiple Access / Collision Avoidance*

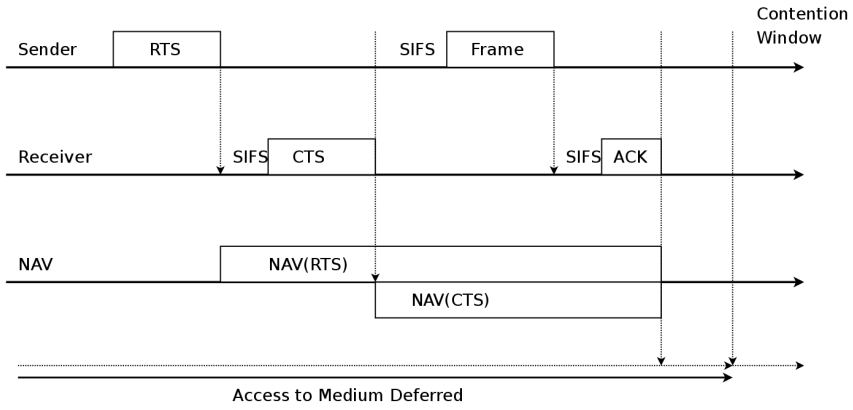

Figure 2.1: Carrier-Sensing Multiple Access / Collision Avoidance

protocol (CSMA/CA). The protocol aims to avoid collisions by forcing stations to wait a random interval after the channel is clear before attempting transmission of data (Thurwatcher, 2002, 102-8).

2.1.8 Hidden Node Problem

The 802.11 standard includes CSMA / CA media access protocol as the basic mechanism to avoid collisions with other stations on the same channel. Usually, all stations sharing a given channel are within transmitting range of each other but this is not always the case. It is possible for two nodes to be placed on opposite sides of an access point at the far reaches of the access point's range. This is commonly referred to as the hidden node problem, and is depicted in figure 2.2.

Since stations A and B cannot detect collisions with traffic from the other, they assume the communciations medium is free and transmit frames. When both stations begin transmitting to the access point at the same time, a collision occurs, and the nodes back off a random delay before reattempting transmission. A collision may be detected in this case by the absence of an 802.11 ACK returned by the access point. This problem is overcome by the RTS / CTS mechism introduced with 802.11b.

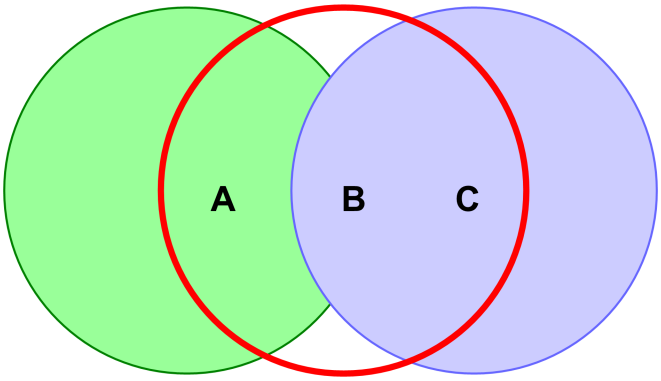

Figure 2.2: Illustration of Hidden Node Problem

The RTS / CTS mechanism of 802.11 provides virtual carrier sensing to prevent the hidden node problem. A station that wishes to transmit a packet must first transmit a short control packet called Request To Send (RTS) which includes the source, destination and the duration of the following transaction. If the communications medium is free, the access point responds with a CTS control packet. Because in infrastructure networks all nodes should be within range of the access point, all nodes should hear this CTS. Both the RTS and CTS packets include the duration that the medium is requested for, allowing all stations to set their *Network Allocation Vectors (NAVs)* to avoid transmitting during another nodes access slot.

Unfortunately, the RTS / CTS mechanism adds a considerable amount of overhead in order to solve the hidden node problem. Greater network performance may be had by moving hidden nodes closer to the access point until they come into range of the other associated nodes, or increasing the transmitting power of far stations. Of course, this solution may not be practical in some situations.

2.1.9 Exposed Node Problem

In wireless networks, the exposed node problem when a node is unnecessarily prevented from transmitting packets to other nodes due to a neighboring transmitter. This is a consequence of the carrier sense mechanism of 802.11 networks.

The solution to the exposed node problem is the utilise the IEEE 802.11 RTS / CTS mechanism, at the expense of some additional overhead. When a node hears an RTS from a neighboring node, but not the corresponding CTS, that node can deduce by itself that it is an exposed node, and it may transmit to neighboring nodes without causing collisions.

2.1.10 Quality of Service

In packet-switched networks and computer networking, the traffic engineering term *Quality of Service (QoS)* refers to the probability of the telecommunication network meeting a given traffic contract. The term is usually used to refer to the probability of a packet successfully traversing the link between two network nodes within a nominal latency period. In the field of telephony, the term takes on a different meaning . In this context, QoS is used to describe user satisfaction of all imperfections affecting a telephone conversation. This definition includes the human in the assessment and demands an appropriate subjective weighting of diverse defects such as noise and tones on the circuit, loudness levels, noticeable echos, etc. In this project, the former definition is used almost exclusively. Artificial *Mean Opinion Scores (MOS)* have be used where it is neccessary to give the reader a reference point for likely user satisfaction levels.

2.2 Voice Over Internet Protocol

Voice over IP is the routing of voice conversations over the Internet or through any other IP-based network (Wright, 2001, pg 69). The voice data flows over a general-purpose packet-switched network, instead of dedicated legacy circuit-switched telephony transmission lines. VoIP technology allows voice conversations to be conducted over conventional Ethernet networks, and more recently over wireless networks (Comer, 2000, pg 539-47). VoIP is handled by the IP protocol, which has the advantage of compatibility with existing network infrastructure. Voice

2.2.1 Corporate Intranet Communication

VoIP is becoming a viable 'telephone alternative'. It is possible to use VoIP systems to make a voice calls to people in the same way that telephones are used today. On corporate intranets and hospitals, VoIP systems are often utilised to facilitate cheap mobile voice communication within the building. In this configuration, mobile 802.11 devices use the wireless networking infrastructure to carry voice calls. A mobile 802.11 device could take the form of a laptop with a microphone and VoIP software installed, or a deditacted Vo802.11 handset.

2.2.2 Gateway to Public Switched Telephone Networks

VoIP gateways allow routing of VoIP calls in much the same way as switched telephone networks. In the same way that a 'gateway' in the network sense is used to connect local area networks to the internet, VoIP gateways provide users with the ability to make VoIP calls to telephones connected to the PSTN. VoIP gateways connect the public telephone network with a computer network and performs the necessary actions and conversations to make the call possible. To make a call to somebody, you would call the gateway and specify the destination for the call. The call woud then be set up and if the other end is available, the conversation can start.

2.2.3 Application to Wide Area Networks

When using VoIP over a Local Area Network (LAN), there is usually plenty of bandwidth available and the delay between sending and receiving is usually very low. Here, VoIP can often be used without problems. However, when a *Wide Area Network (WAN)* is used (ie, the internet) problems can arise. One problem is the delay: while the delay on a LAN is usually very low, on a WAN this is not necessarily true. If the delay becomes too large, the conversation will not be very pleasant. Another problem is the quality of the speech signals. When

certain routes get too heavily loaded, packets on the WAN will be lost. These lost packets cause interruptions in the speech signal. In turn, these interruptions, when large enough, can also disturb the conversation. To alleviate the load, a lot of VoIP programs use lossy compression techniques. Lossy (as opposed to lossless) audio compression achieves excellent data compression ratios by removing some of the signal's information. With such zealous compression techniques, telephone quality audio is not achieved.

2.2.4 Signalling Protocol

Signalling protocols handle the connection and disconnection of voice calls on a packet switched network. Common signalling protocols for VoIP include H.323 and SIP. H.323 was the first signalling protocol to be used for VoIP, and is the signalling protocol recommended by the International Telocommunication Union (ITU-T) (Ohrtman, 2004, page 28). For these reasons, the H.323 signaling protocol only will be considered in this project.

The H.323 signalling protocol is comprised of several other subprotocols. For example, the H.225.0 protocol is used for registration, admission, status, call signaling, and control whilst the H.245 protocol is used for media description control and other uses. Essentially, the process of the H.323 signalling protocol is:

1. Send an invitation to talk to the intended recipient
2. Listen for the response (busy, ringing, or none)
3. Wait for the other party to acknowledge the invitation
4. A voice session is initiated
5. Either party terminate the call
6. The other party acknowledges the call termination

2.2.5 Encoder / Decoder Algorithms

In order for voice to be transmitted over a packet switched network it must first be digitised. The process of converting an analogue signal into a digital representation consists of two stages: quantisation and compression. This task is performed by an algorithm called a codec (Ohrtman, 2004, page 36). Upon reception at the other end, the encoded analogue signal may be reconstructed by decompressing the voice data contained in the VoIP packet. The codec family G.72x has gained the most prevalence, and thus will be the main codec family considered in this scope of this project.

2.2.6 Bandwidth Requirements of VoIP

Depending on the codec used to encode the VoIP data stream, bandwidth requirements vary considerably. A voice over IP connection typically requires less than 10 KB/s of actual data through put, the G.723.1 codec requires only one kilobyte per second of actual data throughput (Wright, 2001, pg 79). Unfortunately, due to the overhead required to transmit each VoIP packet the actual bandwidth required is significantly higher. Each VoIP voice packet must transmit a Real-Time Transport Protocol (RTP), a User Datagram Protocol, and a IP datagrams (Comer, 2000, 542-5). In total, these three datagrams add 40 bytes of overhead to each VoIP packet transmitted. This overhead adds 16 kbps to the bandwidth required, assuming a VoIP packet rate of 50 Hz.

2.2.7 Jitter Buffer

When packets are sent over switched networks, they are not guaranteed to arrive in the same order as they were sent at the destination (or at all) (Gast, 2002, pg 32). Generally, this is not a problem as TCP/IP applications simply reorder packets that were received in the wrong sequence and wait for packets that didn't arrive to be retransmitted successfully. However, for voice over IP it is desirable

for packets to arrive in the right order for obvious reasons. If voice over IP packets were received and played in a different order than recorded, the result would be unintelligible gibberish. Voice over IP applications provide a jitter buffer to alleviate the problem. A jitter buffer allows packets to arrive in any order and at any speed whilst the VoIP application pulls packets off the buffer at regular intervals (Wright, 2001, pg 31-36).

2.2.8 Performance Issues with Vo802.11

At this point in time, the general consensus of the industry is that the quality of service of Vo802.11 is inadequate (Ohrtman, 2004, pg 143). Voice over IP is very sensitive to packet loss and high latency, both difficult things to avoid in wireless data transmissions in general. These problems are compounded by the gross inefficiency of VoIP packet transmission due to excessive overhead and the excessive delays caused by the CSMA/CA protocol (Ohrtman, 2004, pg 144). The IEEE is working on an ammendment to the protocol, namely 802.11e in order to address the QoS issues with Vo802.11 (Ohrtman, 2003, pg 20). However, this ammendment will require the purchase of new hardware, supporting the 802.11e standard. Significant improvement should be possible by other avenues such as through prioritised queuing (Ohrtman, 2004).

2.3 The UDP Protocol

The *User Datagram Protocol (UDP)* is a transport layer protocol defined for use with the IP network layer protocol. It provides a best-effort datagram service to a destination system. UDP transactions are connectionless, and unresponsive. That is to say, it is not neccessary to set up a connection with a server running a UDP service, a user datagram is simply constructed, transmitted and forgotten. There is no TCP equivalent of an acknowledge control packet for UDP. The purpose of UDP was to avoid the added overhead utilised by TCP where reliability

of the packet stream is not important, and / or where the packet contents are delay sensitive. The service provided by UDP is unreliable, that is to say, it provides no guarantees for delivery and no protection from packet duplication or arrival out of sequence.

The UDP header is extremely simple. It of four two byte fields, which makes it eight bytes in length. These fields are:

1. Source Port: UDP packets from a client use this as a *Service Access Point (SAP)* to indicate the session on the local client that originated the packet.
2. Destination Port: UDP packets from a client use this to indicate the service required from the remote server.
3. UDP length: The size of the UDP header plus the payload data in bytes.
4. UDP Checksum: The UDP checksum attempts to detect alteration in the packet (corruption) due to various noise mechanisms between the transmitting node and the receiving node. A checksum cannot absolutly verify that the end to end data has not been corrupted, but when the check sum computes it is statistically unlikely that the packets contents are corrupt. This field can be ignored by zeroing the bits of the UDP checksum.

2.4 Network Simulation

In computer network research, *network simulation* is a technique where a program simulates the behavior of a network. The program performs this simulation either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back network parameters from a production network.

Using this input, the behavior of the network and the various applications and services it supports can be observed in a simulation. ¡reference¿ Various attributes

of the environment can also be modified in a controlled manner to asses these behaviors under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as *network emulation*.

2.4.1 The NS-2 Simulator

The NS-2 simulator is a powerful *open-source* network simulator capable of simulating complex wired and / or wirless network infrastructure.Development of ns has been was supported by DARPA through the VINT project and by numerous other organisations including Xerox PARC, the University of California, Berkeley and the University of Southern California. Currently ns development is support through *Defense Advanced Research Projects Agency (DARPA)* in collaboration with other researchers including AT&T's center for internet research.

NS-2 has become a popular tool in both industrial and academic settings due to its extensibility and plentiful online documentation. The NS-2 simulator supports a plethora of popular network protocols, and offers simulation capabilities for both wired and wireless networks. The simulation results of the NS-2 network simulator are believed to be accurate indications of actual network performance. Whilst developers of the simulation package state that accurate results are not gauranteed, studies have shown that the accuracy of NS-2 is comparable to commercial network simulators (such as the OPNET Modeller).

> *"While we have considerable confidence in ns, ns is not a polished and finished product, but the result of an on-going effort of research and development. In particular, bugs in the software are still being discovered and corrected. Users of ns are responsible for verifying for themselves that their simulations are not invalidated by bugs. We are working to*

> *help the user with this by significantly expanding and automating the validation tests and demos."* —**John Heidemann 2000**, Ns Developer

The core of the NS-2 network simulator is written in c++, providing a robust framework to base simulations upon. This part of the simulator performs the functions of various network protocols and emulation of physical links between nodes. In order to allow users to rapidly produce and alter network simulations, simulation scripts are written in *Object-Oriented Tool Command Language*, often abbreviated to *oTcl*. NS-2 provides oTcl interfaces to user controllable paramaters such as the speed of a communications link, or the length of a queue on a router. The oTcl language is far less 'structured' and syntactically complex than c++, allows ns users to ignore a great deal of the underlying complexity and produce simulation scripts that read well.

2.4.2 Network Animator

Nam is a Tcl/TK based animation tool capable of interpreting network simulation traces produced by the NS-2 network simulator. In this project, nam was used as a tool to visualise data flow through various network topologies. It allows manipulation of the topology layout and displays packet level animation. Various data inspection tools provide the user access to statistics such as channel packet loss and average end to end delay at any point in the simulation.

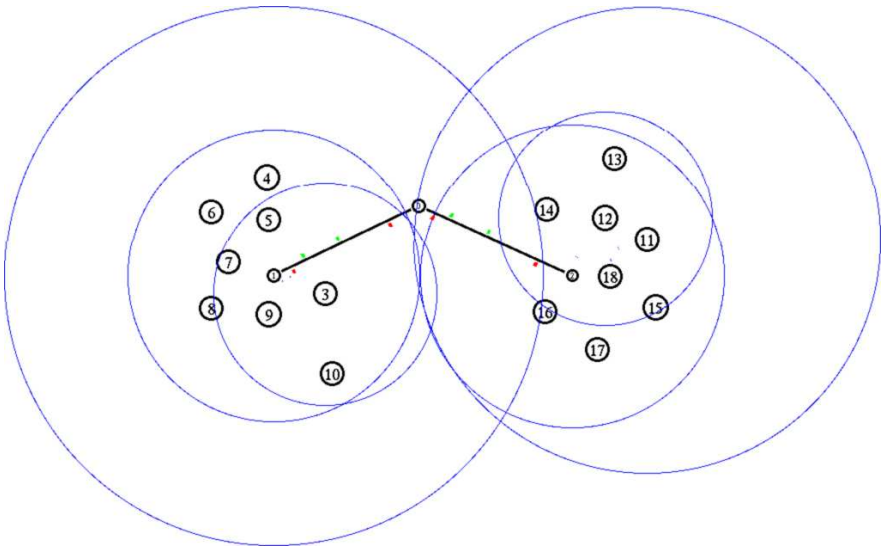

Figure 2.3: Network Animator Depicting Wired & Wired Topology

Chapter 3

Methodology

3.1 Introduction

In order to achieve each of the objectives of this project, several seperate tasks were defined from the list of objectives. In this chapter, the author identifies the methods used to achieve the objectives and justifies their selection. The objectives of the project were to:

1. Establish that issues with wireless 802.11 networks exist and determine leads for further studies
2. Experimentally verify the existence of issues with performance via network simulation
3. Identify problem areas and possible solutions
4. Modify networking components for the ns2 network simulator to achieve better network performance
5. Critically evaluate the effect of these alterations on performance via simulation
6. Propose improvements to increase the performance of wireless networks
7. Implement the improvements / adaptations on a wireless network testbed

The methods used to achieve each objective are described in the sections to follow.

3.2 Investigation of Performance Issues

The purpose of investigating performance issues that exist with wireless 802.11 networks was to:

- Justify the need for research in this area
- Determine the extent of performace problems

- Find and incorporate existing relevant research into this project

In order to establish that issues with wireless 802.11 networks exist, the author found relevant existing research by reading various publications on the topic (as referenced in the literature review). It was also useful to complement this by studying the *transport control protocol / internet protocol (TCP/IP)* in detail (RFC 1180, 791, 793), which led to a better understanding of the inner workings IP based networks. General networking theory was revisited, and network simulation theory was explored. Since wireless networking devices communicate using the 802.11 protocol, information on the protocol was gathered to give the author an insight into the operation of 802.11 networks. Relevant ammendments to the 802.11 protocol were also studied in detail, with particular attention to wireless 802.11g networks.

3.3 Theoretical Capacity Analysis

After issues with the performance of 802.11 wireless networks were established, some theoretical capacity calculations were undertaken to determine the major causes of performance problems. The most significant performance issue found with wireless networks was its poor capacity for voice calls. In order to validate the research on performance issues in wireless networks, theoretical calculations were performed (as reported in chapter XX). The efficiency of VoIP at the IP layer and theoretical call capacity of access points were determined. This was done by utilising information gathered on VoIP, TCP/IP and the 802.11 protocol.

3.4 Reproducing Issues in Simulation

Reproducing performance issues in wireless 802.11 networks in simulation served to further verify the existance of performance issues. By simulation, the efficiency and effectiveness of various wireless networking arrangements to carry VoIP were

analysed. A constant bit rate (CBR) source operating intermittently provided a reasonably accurate approximation to a voice over IP application on a network. Network simulation, as opposed to actual network testing was chosen to expose performance problems that are explained in the sections to follow.

3.4.1 Complex Topologies Possible in Simulation

Network simulation allows a user to simulate complex networking arrangements with equipment that he / she does not possess or cannot afford. Through simulation, a person can analyse a situations that are impractical to physically reproduce, such a distributed denial of service (DDOS).

3.4.2 Simulation Accuracy

The simulation results of the ns-2 network simulator are believed to be accurate indications of actual network performance. Whilst developers of the simulation package state that accurate results are not gauranteed, studies have shown that the accuracy of ns-2 is comparable to commercial network simulators (such as the OPNET Modeller).

3.4.3 Access to More Information

In simulation, it is possible to access far more information than available in actual testing. This is due to the fact that it is possible to log *any* network event on a network simulator, some of which may be impossible to know on a networking test bed. For instance, a packet analyser could tell you that a certain packet was transmitted and not acknowleged by the recipient, but it cannot tell the user where the packet was dropped or the reasons for it. In network simulation using the ns-2 network simulator and **nam**, dropped packets may be visualised and interrogated to determine the time and reason they were dropped.

3.4.4 More Flexible & Extensible

It is possible to rapidly change the simulation parameters in ns-2 simulations by editing the simulation scripts. Parameters such as link speed, queue length and node position can be changed in a few seconds, as opposed to minutes or hours with a networking test bed *when it is even possible*. Implementing and debugging new network protocols can be accomplished by writing / modifying network components for the ns-2 network simulator in C++ recompiling the simulator.

3.5 Modification of Network Protocols

After the initial simulations exposed performance issues with the 802.11 protocol, the root causes were identified. From this information it was possible to devise methods increase the performance of VoIP on wireless networks without hardware alterations or modification to the 802.11 protocol. These methods included changing parameters of wireless 802.11 access points such as the RTS / CTS threshold, and the MTU size as well as modifying network components of the NS-2 network simulator. Alterations to network components to be more conductive to VoIP traffic included writing a prioritised packet scheduler for the NS-2 network simulator, and a VoIP agent. No modifications were made on the hardware level, or to the 802.11 protocol to ensure that:

1. Improvements could be implemented on any 802.11 device / network
2. Backwards compatibility with existing 802.11 networks was maintained
3. Performance improvements could be implemented 'out of house' (by end user)

3.6 Evaluation of Performance

A set of benchmarks are needed to measure and report the performance of a packet switched network. Several performance criteria were chosen to convey information about network performance. These included:

1. Percentage packet loss
2. Average end-to-end delay
3. Round trip time & Latency
4. Artificial Mean Opinion Scores (VoIP specific)
5. Call capacity (VoIP specific)
6. Jitter

Percentage packet loss, average end-to-end delay and round trip time are standard measures of network efficiency. Artificial Mean opinion scores were chosen to evaluate the perceived voice quality of the simulated VoIP sessions because actual voice data is not transmitted in simulation. Artificial MOS's have had a high correlation with actual MOS's in studies, and have the advantage of not being avoiding human subjectivity. Call capacity is a measure of the number of simultaneous VoIP sessions an Access Point (AP) can support whilst retaining a good MOS. It is an estimate of the maximum number of voice calls an access point can handle without significant voice quality degradation. The term 'Jitter' is used to refer to describe the variation in transit delay of individual packets which leads to annoying voice artifacts in voice conversations.

3.7 Performance Improvements

From initial simulations, possible improvements to the way wireless networks handle voice traffic were idenitified, and modifications to ns-2 simulation scripts

and networking components were made to implement them. After each change to parameters / network protocols, the network performance was re-evaluated and recorded. It should be noted that it was not possible to test all possible improvements due to the sheer complexity of the ns-2 simulator and the network protocols themselves.

3.8 Prediction of Access Point Capacity

The access point calculation for different 802.11 networks were found by determining the minimum framing interval for a VoIP packet, and inverting this value to get the maximum number of VoIP frames transmitted per second.

$$i_{frame} = DIFS + DATA/datarate + SIFS + 802.11ACK \tag{3.1}$$

$$packetrate = \frac{1}{i_{frame}} \tag{3.2}$$

$$call\ capacity = \frac{packetrate}{2 \times VoIP_{framerate}} \tag{3.3}$$

Chapter 4

Performance Issues with Vo802.11

4.1 Introduction

Voice over IP is the routing of voice conversations over IP-based networks. The voice data flows over a general-purpose packet-switched network, instead of the dedicated public switched telephone network. This has enabled voice conversations to be carried over conventional Ethernet networks, and more recently over wireless networks.

Unlike telephone switching networks, the medium for VoIP is not dedicated for voice traffic. Packet switched networks are typically used for data rather than voice, and are geared towards high data throughput rather as a result. This chapter aims to analyse the performance of Vo802.11 networks and determine areas for improvement.

4.2 Capacity analysis of 802.11 Access Points

From a simple efficiency calculation, the poor suitability of 802.11 networks for VoIP becomes obvious. Modern 802.11 access points has a bandwidth of 54Mb/s (in 'a' or 'g' modes). The typical bandwidth requirement of a bi-directional voice call is under 20 kb/s[1] (considering only the voice payload), depending on the codec used. A list of common codecs and their bandwidth utilisation is provided in table 4.1.

In principle, 802.11 access points have enough raw bandwidth to carry thousand of simultaneous connections VoIP sessions. The theoretical call capacity for an 802.11 access point operating at common data rates is shown in table 4.2

The reality is, an 802.11g access point operating at 54 Mb/s can only handle about 18 simultaneous connections using the G.729 codec (as opposed to a theoretical capacity of over 3000 simultaneous connections) in protected mode. This equates

[1] In telephony, a kilobit per second is 1,000 bits per second, as opposed to the 2^{10} (1024) bits per second that is more common in computing.

Codec	Voice Payload (Bytes)	Sample Interval (ms)	Bitrate (Kb/s)
G.711	160	20	64
G.729	20	20	8
G.723.1a	24	30	6.3
G.723.1b	20	30	5.3
G.726a	80	20	32
G.726b	60	20	24
G.728	60	30	16

Table 4.1: Common VoIP Codec Bandwidth Requirements

BW	G.711	G.729	G.723.1a	G.723.1b	G.726.a	G.726.b	G.728
54 Mb/s	432	3456	4388	5216	864	1152	1728
48 Mb/s	384	3072	3900	4636	768	1024	1536
36 Mb/s	288	2304	2925	3477	576	768	1152
24 Mb/s	192	1536	1950	2318	384	512	768
18 Mb/s	144	1152	1462	1738	288	384	576
12 Mb/s	96	768	975	1159	192	256	384
11 Mb/s	88	704	893	1062	176	234	352

Table 4.2: Theoretical Call Capacities of 802.11 Access Points at Common Data Rates

to an transport efficiency of VoIP just over 0.5 %. From this standpoint, it obvious the implementation of VoIP over wireless is grossly inefficent.

4.3 Efficiency Limitations at Protocol Level

VoIP packets must be small to give good quality of service over a network because the voice sample size sets the lower bound for the transit delay. Unfortunately, the small transmission size of VoIP packets make it inherently inefficient. The necessary overhead for transmission of a UDP packet is 40 bytes including the IP, UDP and RTP headers. With this considered, the payload seems remarkably small. This causes the efficiency at the IP layer to typically be lower than 60%. Table 4.3 shows the efficiency of the most common VoIP codecs.

Codec	**Payload (Bytes)**	**Net Bitrate (Kb/s)**	**% Efficiency**
G.711	160	80	80.0
G.729	20	24	33.4
G.723.1a	24	17.4	37.5
G.723.1b	20	16.3	33.4
G.726a	80	48	66.7
G.726b	60	40	60.0
G.728	60	27.2	60.0

Table 4.3: Efficiency of Common VoIP Codecs over Wireless 802.11 Networks

4.4 Queing Problems with Small Packets

Another problem due to the small size of VoIP packets is they have a tendency to get stuck behind larger packets in network queues. In order to maximise the efficiency and throughput, TCP traffic utilise larger packet sizes. These are usually close to the maximum transmissible unit, which defaults to 1500 on most APs. These large packets consume large amounts of bandwidth, and tend to be

transmitted in rapid succession. Since the transmission time for a TCP packet may be several times longer than the transmission time for a VoIP packet, this causes TCP traffic to produce delays in VoIP packet transmission..

4.5 CSMA / CA Delays

The *Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA)* protocol helps prevent collisions by sensing when other devices are transmitting on the network. A station wishing to transmit must first sense the channel for the duration of a *distributed interframe space (DIFS)* plus the *backoff*[1] to check for activity on the channel. If the channel is sensed "idle" then the station is permitted to transmit. If the channel is sensed as "busy" the station defers transmission. Figure 4.1 depicts two stations communicating via the CSMA/CA protocol.

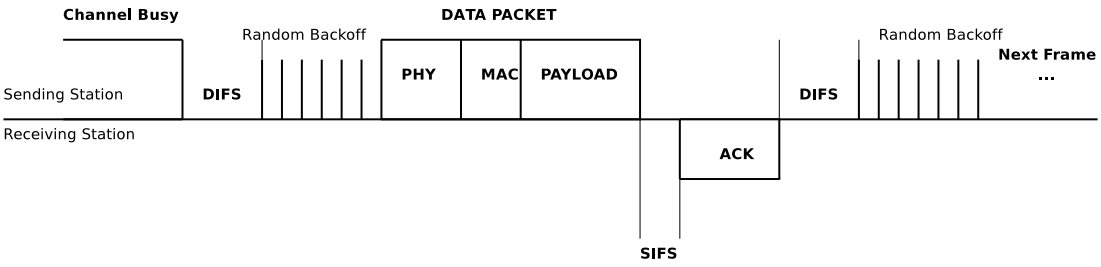

Figure 4.1: Carrier Sense Multiple Access with Collision Avoidance Protocol

The CSMA/CA protocol attempts to increase the throughput on 802.11 networks by reducing the number of packet collisions. In doing so, it introduces additional overhead for every packet transmitted. The protocol does very well to prevent collisions on congested networks, but delays introduced to VoIP packets are quite large. The *Distributed Coordination Function (DCF)* paramaters for 802.11 networks define timing aspects of the CSMA/CA protocol. These parameters are listed in table 4.4.

[1]The *backoff* is a random number multiple of *slots* ranging from 0 to the *contention window (CW)*

Parameter	802.11a	802.11b	802.11g
DIFS	34 μs	50 μs	28 μs
SIFS	16 μs	10 μs	10 μs
Slot Time	9 μs	20 μs	10 μs
CW_{MIN}	15	31	15
CW_{MAX}	1023	1023	1023
Max Rate	54 Mb/s	11 Mb/s[1]	54 Mb/s

Table 4.4: Distributed Coordination Function Parameters for 802.11[a, b & g] Networks

4.6 Overhead Due to RTS / CTS Mechanism

The *Request to Send, Clear to Send (RTS/CTS)* mechanism is required to avoid the hidden node problem. When a node has data to send, it initiates the process by sending an RTS frame. The destination node replies with a CTS frame if the channel was idle. The amount of time the node should wait before trying to get access to the medium is included in both the RTS and the CTS frame. Any node that 'overhears' the CTS frame sets their *Network Allocation Vector (NAV)* and will not attempt to access the channel until the NAV has expired. This virtual carrier sensing mechanism solves the hidden node problem, however, it comes at the expense of additional network overhead. Figure 4.2 depicts a RTS/CTS transaction.

Voice over IP packets are small, and take little time to transmit. It is impractical to use the RTS/CTS protocol in conjunction with medium VoIP traffic due to the delays caused by RTS/CTS frames. In the time it takes to send a RTS and receive a positive acknowledge, at least two VoIP packets could have been transmitted. This has the effect of reducing the network capacity for VoIP sessions by two thirds. It is possible to set an RTS/CTS threshold between 0 and 2347 bytes, which has the effect of allowing packets smaller than the threshold to bypass the RTS/CTS mechanism. If the threshold is set higher than this value, the RTS/CTS mechanism is disabled (which is the default setting of most access

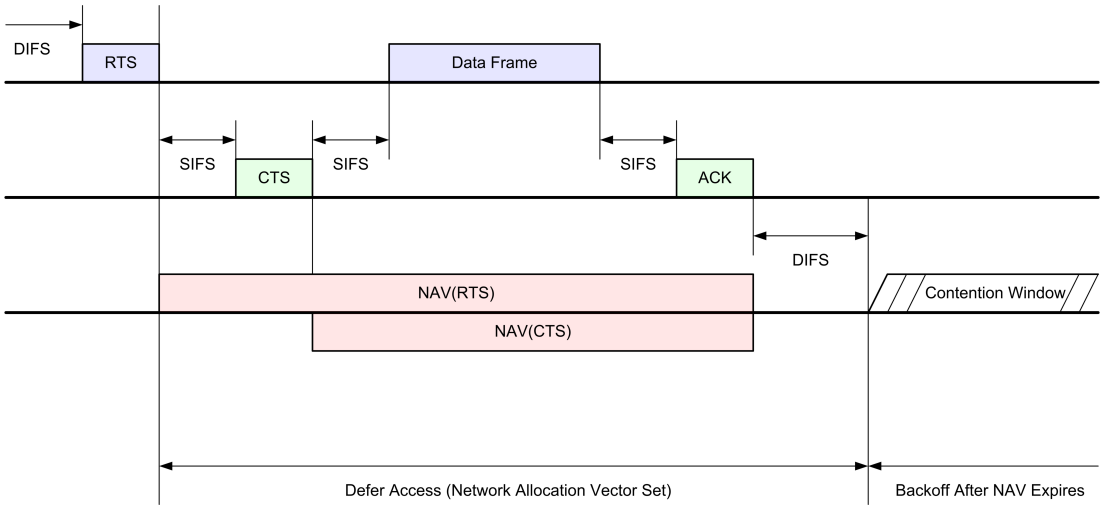

Figure 4.2: Request to Send, Clear to Send Transaction

points).

4.7 Association Issues

It takes time for wireless nodes to associate / deassociate from an access point. With 802.11, it is not possible to be associated with two access points at the same time. When users conducting a voice call using VoIP move between access points with wireless 802.11, their call drops out. Roaming is not possible at the moment, one possible solution is to allow association at more than one access point, and implement an improved hand-off mechanism. Association issues are outside the scope of this project because they are part of the 802.11 protocol. It is discussed here only for completeness.

4.8 Security Protocols

Mention that implementing security causes decreased network capacity

4.9 Inability to Coexist with TCP Traffic

Chapter 5

Network Simulation

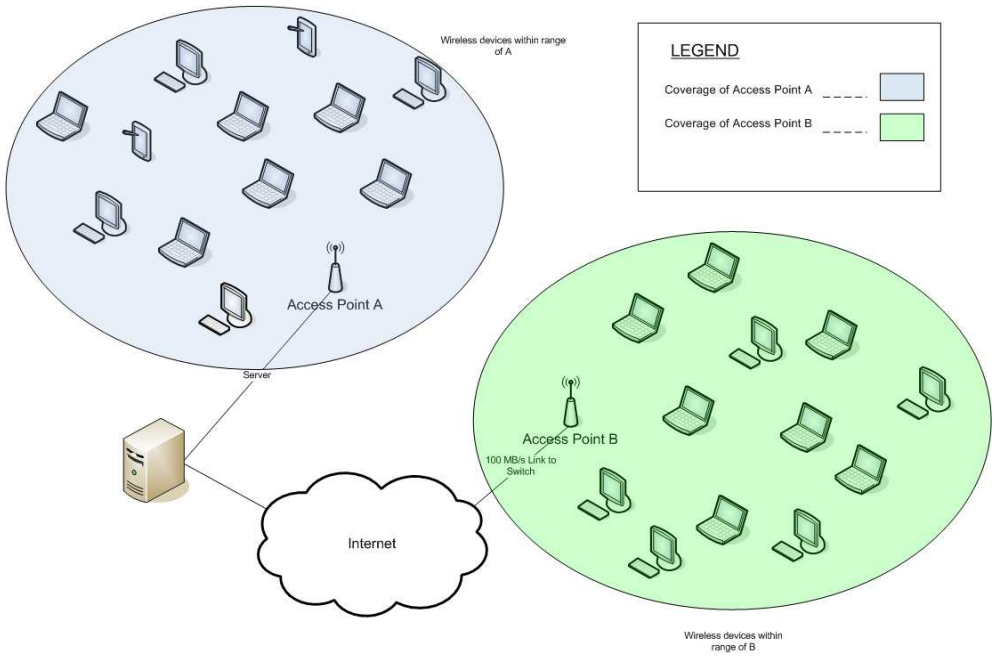

Figure 5.1: Wireless Networks Bridged via the Internet

5.1 Introduction

The aim of simulating Vo802.11 was to reproduce performance issues in wireless 802.11 networks via simulation. This would serve to further verify the existance of performance issues, and expose the mechanisms for said issues. The efficiency and effectiveness of various wireless networking arrangements to carry VoIP were analysed using the ns-2 network simulator[1].

5.2 Wireless Network Topology

The main network configuration was two wireless networks separated via a low latency broadband internet link. This was approximated in simulation by creating two wireless / wired hybrid nodes (access points) connected via a T1 speed link. A T1 link is a full duplex link with a maximum throughput 1.544 Mb/s. The latency of the link was arbitrarily chosen to be 50 ms, which is a relatively low latency connection on the internet. It should be noted that the end to end latency of this link is not as important

[1]Please refer to Appendix B for the tcl simulation scripts

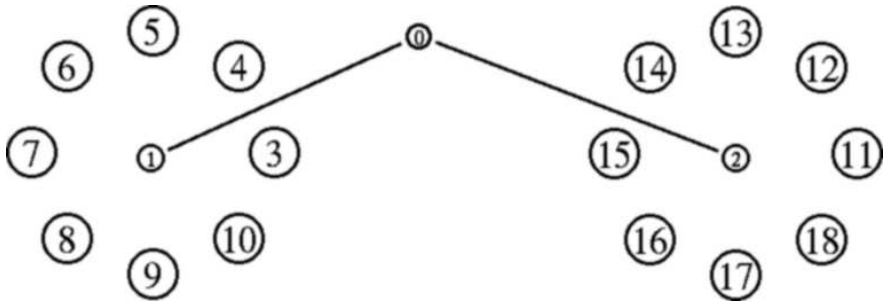

Figure 5.2: Network Topology as Illustrated with the Network Animator

as the variation of end to end delay (jitter), and varies greatly between internet nodes. The wireless simulation test plan defined in the methodology was followed for both networking scenarios. To briefly restate the important details:

1. Each access point supports a varying number of mobile wireless nodes
2. The two wireless networks are not within transmission range of each other
3. Both networks operate in *Infrastructure Mode* at all times
4. All mobile nodes belonging to an access point are within range of all other mobile nodes associated with said access point
5. Security protocols (WEP & WPA) have been disabled
6. Mobile nodes remain stationary over the simulation period (specifically, no random movement)

In order to model VoIP sessions in simulation, *Constant Bit Rate (CBR)* sources were attached to *User Datagram Protocol (UDP)* transport agents for each node pair conducting VoIP sessions. The CBR sources operated intermittently, sending payloads of 'voice data' to the UDP agents at the frame rate of the VoIP codec modelled. In actual fact, VoIP packets are not plain UDP packets, they use the *Real Time Protocol (RTP)*. However, a CBR agent generating UDP packets approximate VoIP traffic quite well because RTP and UDP are both unresponsive protocols[1].

[1] Unresponsive traffic types do not sense the network communications medium or retransmit dropped / corrupted packets. For this reason, it is appropriate to substitute a UDP agent for a RTP agent in network simulation without affecting the results.

5.3 Wireless Network Scenarios

Two main wireless networking scenarios are included in this report. In the first scenario, the wireless networks are dedicated to voice traffic. When no other traffic sources are competing for bandwidth, the call capacity of the topology can be calculated. However, it is not often the case that a wireless network is reserved exclusively for voice caller access. Most wireless networks in place are used to connect computers on a network to share data (as opposed to carrying voice calls). Network simulation was used to determine how well VoIP coexists with external traffic sources.

5.3.1 Network Dedicated for Voice Traffic

To briefly recap the methodology for this scenario, wireless 802.11 a, b & g networks are simulated at various data rates for each of the common VoIP codecs listed in chapter one. The simulation process was as follows:

- Simulation begins with no VoIP traffic
- Each 10 seconds, an additional node pair begin a VoIP session
- Once a node pair begin a VoIP session, it is maintained until the end of simulation

After simulation, the packet loss, jitter and average end to end delay were found using the unix *awk* utility. Awk provides text file processing facilities (it is not a word processor), and can perform calculations on data in text files. The results obtained from simulation were tabulated, and performance issues analysed. The capacity of the network for VoIP sessions was also recorded.

5.3.2 Presence of Transmission Control Protocol Traffic

The simulation procedure for this scenario was identical to the previous, with the exception of the presence of a TCP traffic source. The TCP traffic source starts soon after the beginning of the simulation and before the first VoIP call is established. As

with the previous scenario, a new VoIP session is started each 10 seconds until any of the VoIP sessions has a packet loss of over 1%. When this occured, no more nodes were added to the simulation.

5.4 Initial Simulation Results

5.4.1 Network Dedicated for Voice Traffic

The network dedicated for VoIP traffic had performance in simulation that was consistent with the expected results. It can be seen from the results that network data rate has only a marginal effect on VoIP capacity. Specifically, doubling the data rate does not even come close to doubling the number simultanious voice calls supported.

The of 802.11a access points to carry VoIP on dedicated networks was found to be quite good, even at low data rates. Compared to the 802.11b protocol, the 802.11a has a much tighter distributed coordination function.

Rate (Mb/s)	**Ideal Capacity**	**Predicted Capacity**	**Simulation Capacity**	**Difference** (%)
6	384	49.0	43	14.0
9	576	59.5	49	21.4
12	768	66.6	56	18.9
18	1152	75.8	63	19.0
24	1536	81.3	69	17.8
36	2304	87.7	75	17.0
48	3072	91.3	76	18.5
54	3456	92.6	76	20.0

Table 5.1: Concurrent Call Capacity for 802.11a per Access Point with G.729

Access points using the 802.11b standard are somewhat less efficient at carrying voice calls, especially with the older 192μs physical preamble. The minimum transaction

time of 802.11b networks is much longer 802.11a, causing gross inefficiency for small packet traffic like VoIP. Newer 802.11b devices can reliably synchronise with the carrier in less than half the time than in the original 802.11b standard ($96\mu s$). Large performance increase may be had by using the shorter preamble. In simulation, devices using the shorter preamble were able to support approximately 35% more concurrent VoIP sessions.

Rate (Mb/s)	**Ideal Capacity**	**Predicted Capacity**	**Simulation Capacity**	**Difference** (%)
1	62.5	13.5	12	12.5
2	125.0	19.9	17	17.1
5.5	343.8	28.5	26	9.6
11	687.5	32.6	29	12.4
22	1375.0	35.1	32	9.7

Table 5.2: Concurrent Call Capacity for 802.11b per Access Point (Short Preamble)

Rate (Mb/s)	**Ideal Capacity**	**Predicted Capacity**	**Simulation Capacity**	**Difference** (%)
1	62.5	10.7	9	18.9
2	125.0	14.4	13	10.1
5.5	343.8	18.4	15	22.7
11	687.5	20.1	18	11.7
22	1375.0	21.0	18	16.7

Table 5.3: Concurrent Call Capacity for 802.11b per Access Point (Long Preamble)

The capacity of 802.11g without RTS/CTS or CTS to self protection is *identical* to 802.11a. At first glance, this seems unusual as 802.11a and 802.11g use different timing values. Although the DIFS for 802.11g is shorterthan 802.11a by $6\mu s$, the required $6\mu s$ extension after the data frame cancels any performance benefit. When this is considered, it is no surprise that the performance calulated and seen in simulation is identical for the two networks.

Backwards compatibility is useful in situations where interoperability with existing

Rate (Mb/s)	Ideal Capacity	Predicted Capacity	Simulation Capacity	Difference (%)
6	384	49.0	43	14.0
9	576	59.5	49	21.4
12	768	66.6	56	18.9
18	1152	75.8	63	19.0
24	1536	81.3	69	17.8
36	2304	87.7	75	17.0
48	3072	91.3	76	18.5
54	3456	92.6	76	20.0

Table 5.4: Concurrent Call Capacity for 802.11g per Access Point with G.729 (Non - Protected)

infrastructure is required. The 802.11g standard was designed to yield performance increases for next generation wireless equipment whilst retaining backwards compatibility with the older 802.11b standard. It could do so because 802.11g operates in the same part of the radio frequency spectrum as 802.11b.

However, this backwards compatibility comes with a serious performance cost. When an 802.11b device associates with an 802.11g access point, the slower DCF timings of 802.11b are used. Additinally, CTS-to-self protection enables 'g' devices to communicate at higher speeds without disturbing 'b' devices but with larger overheads than 'g' only networks. Performance of VoIP on combined 'g' & 'b' networks is far less than ideal.

5.4.2 Presence of Transmission Control Protocol Traffic

It is rarely the case that a wireless network is reserved specifically for voice traffic. Normally, wireless networks are used for data traffic. In simulation, the addition of TCP traffic sources ***severely interfered*** with VoIP traffic. With the addition of a single TCP source, the quality of service metric cannot be met even for a few VoIP sessions. Whilst the packet loss criteria can be met for low numbers of VoIP connections,

Rate (Mb/s)	Ideal Capacity	Predicted Capacity	Simulation Capacity	Difference (%)
6	384	22.9	20	14.5
9	576	24.9	20	24.5
12	768	26.1	22	18.6
18	1152	27.4	23	19.1
24	1536	28.1	25	12.4
36	2304	28.8	25	15.2
48	3072	29.2	26	12.3
54	3456	29.3	26	12.7

Table 5.5: Concurrent Call Capacity for 802.11g per Access Point with G.729 (Protected, Long Preamble)

Rate (Mb/s)	Ideal Capacity	Predicted Capacity	Simulation Capacity	Difference (%)
6	384	29.3	27	8.5
9	576	32.8	29	13.1
12	768	34.8	31	12.3
18	1152	37.2	35	6.3
24	1536	38.5	35	10.0
36	2304	39.8	37	7.6
48	3072	40.6	38	6.8
54	3456	40.8	38	7.3

Table 5.6: Concurrent Call Capacity for 802.11g per Access Point with G.729 (Protected, Short Preamble)

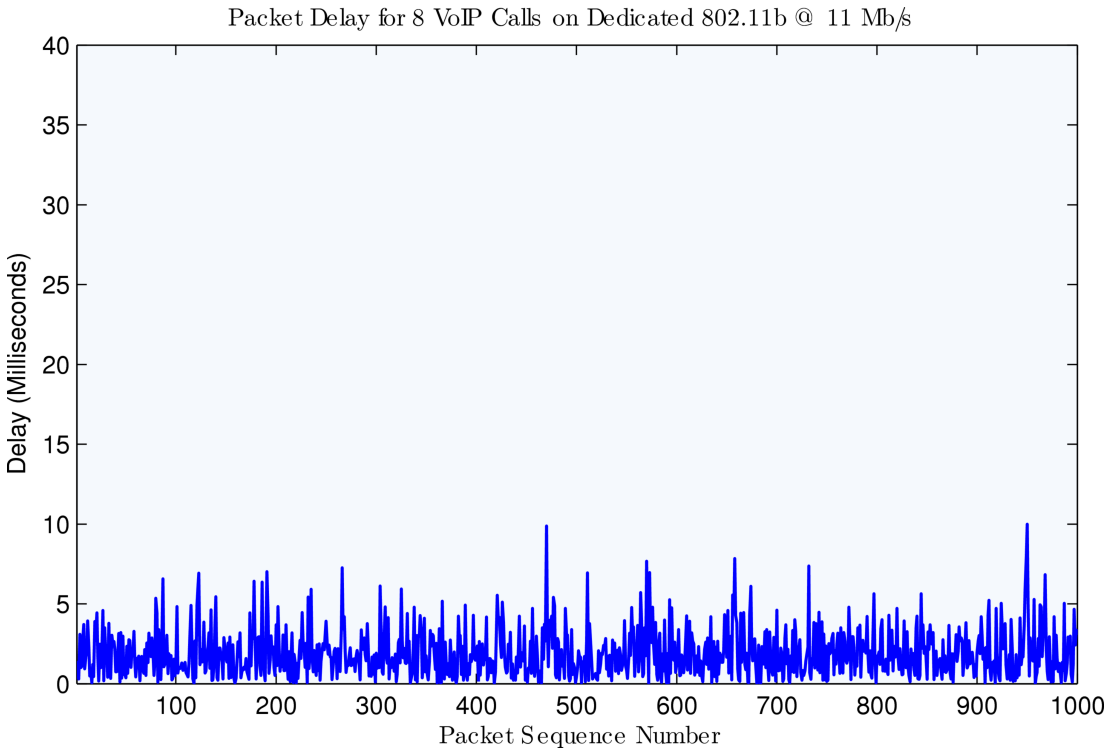

Figure 5.3: Packet Delay for 8 VoIP sessions without a TCP Traffic Source

it is more difficult to make guarantees of low jitter.

Due to the unresponsive nature of UDP, VoIP traffic is transmitted at the constant bit rate defined by the codec used. When a TCP traffic source appears on the network, it begins increasing its rate of transmission exponentially until packet loss occurs. At this point, it has found a (*presumably*) suitable value for its contention window, and reduces its transmission rate. However, from time to time, the TCP traffic source will attempt to increase its transmission rate again, which results in packets loss.

VoIP clients do not have to compete with TCP traffic for bandwidth as such, however, delay insensitive TCP data is often transmitted ahead of VoIP traffic causing delays. Figure 5.4 and 5.3 show the packet delay versus sequence number for a wireless network with (figure 5.3) and without (figure 5.4) TCP traffic. Upon comparing figure 5.4 to figure 5.3, the impact of TCP traffic on VoIP can easily be seen.

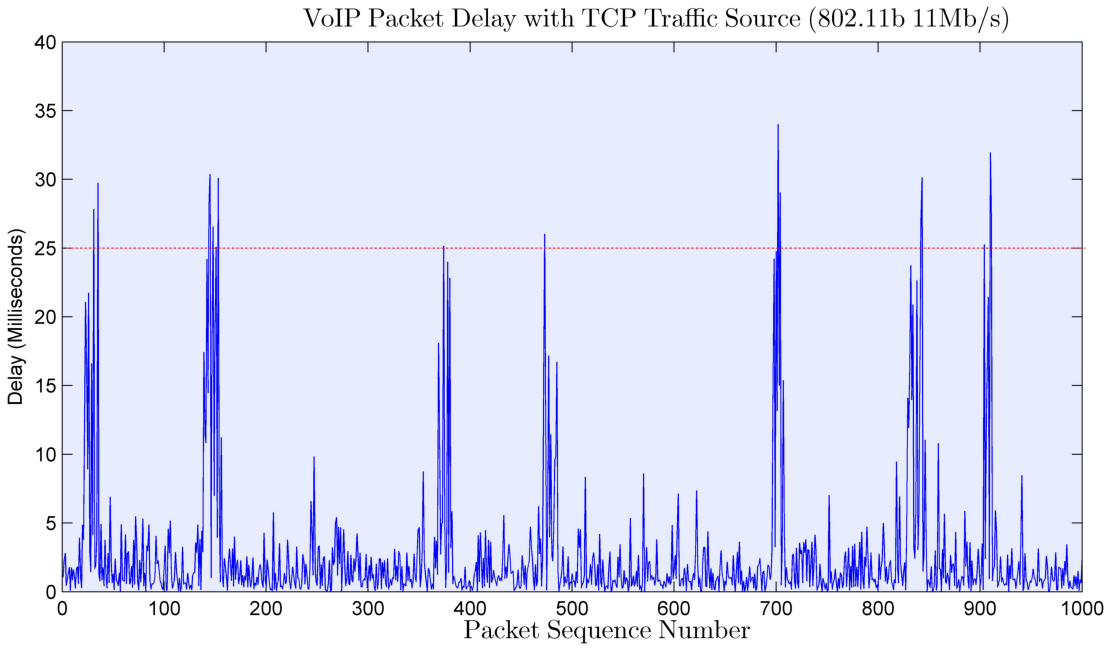

Figure 5.4: Packet Delay for 8 VoIP sessions with a TCP Traffic Source

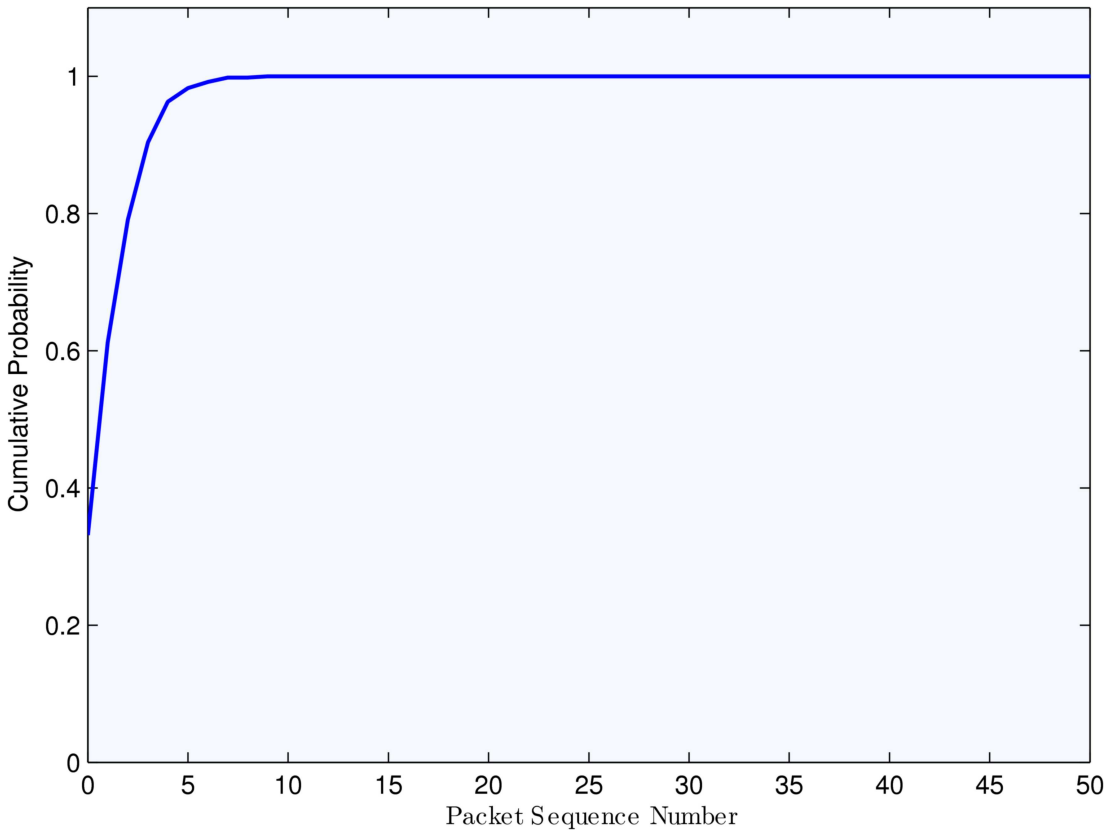

Figure 5.5: Cumulative Distribution Function of Packet Delay (without a TCP Traffic Source)

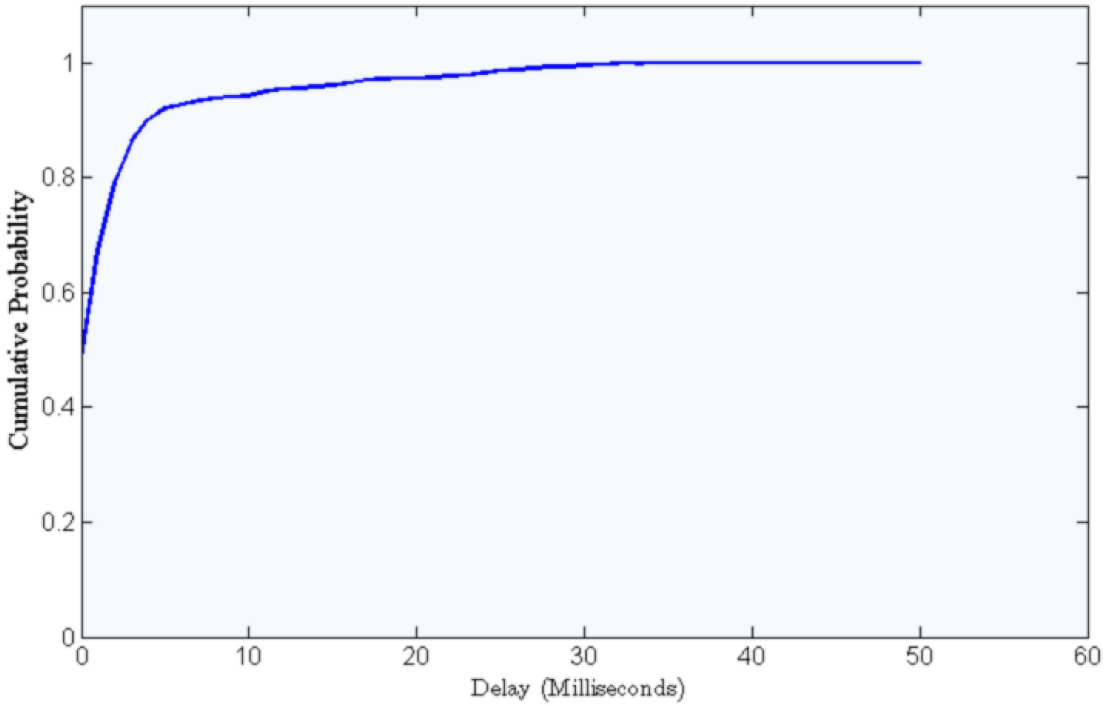

Figure 5.6: Cumulative Distribution Function of Packet Delay (with a TCP Traffic Source)

5.5 Outcomes of Initial Simulation

Network simulation of 802.11 wireless networks used for VoIP has served several purposes. An objective was to obtain simulation results that correlate well with the predicted capacity calculations. This objective was achieved, with simulation results generally within 20% of the predicted values. The simulation results were always below what the theoretical calculations predicted. A likely cause of the discreptancy is the capacity calculations not taking collisions and packet corruption into account. The fact that simulation results were within 20% of the predicted performance supports the premise that the ns-2 simulator can produce accurate results. In doing this, the existance of performance issues in wireless 802.11 networks was also confirmed.

Through simulation, insight into the mechanisms of 802.11 networks was gained and leads for performance improvement determined. These performance improvements are discussed and implemented in the prioritised packet scheduler discussed in the next chapter.

Chapter 6

The Prioritised Packet Scheduler

6.1 Introduction

In light of the performance issues identified in chapter, improvements to the way wireless networks have been identified & implemented in the form of a *prioritised packet scheduler*. In this chapter, the methods for improvement are discussed and the expected performance benefits are analysed. A detailed discussion of the implementation of performance improvements to wireless 802.11 networks follows, with simulation results to demonstrate the impact of said improvements.

6.2 Performance Improvements for Vo802.11

6.2.1 RTS / CTS Threshold Setting

The purpose of the *Request To Send / Clear To Send (RTS / CTS)* mechanism was to solve the hidden and exposed node problems. As noted in simulation, the RTS / CTS mechanism of the CSMA / CA protocol increases the network overhead significantly. When RTS / CTS is disabled, the call capacity of 802.11 networks is greatly increased. In simulation, the call capacity was tripled by setting the RTS / CTS threshold above the VoIP packet length. This paramater can be easily adjusted on most access points, and requires no hardware / software changes.

6.2.2 Maximum Transmission Unit Size

On packet switched networks, the *Maximum Transmission Unit (MTU)* is the largest packet size that can be transmitted. If a larger packet than the MTU needs to be sent, the packet is divided into *fragments* for transmission, and recombined at the receiver. The significance of the MTU in traffic engineering is that network packet size sets the lower bound for queueing delays. By setting the MTU to a low value, it is possible to give smaller packets a greater possibility of being transmitted after a smaller delay. It also has the advantage of reducing the size of retransmissions due to packet corruption.

Care must be taken not to set an MTU too low, as the network overhead increases with the number of packets transmitted. Changing the MTU alters the performance of VoIP sessions in the presence of TCP traffic, but has little effect otherwise. By itself, changing the MTU didn't have a great impact on performance. However, when used in conjunction with a prioritised scheduler, the performance increases were significant.

6.2.3 Priority Queuing for VoIP Packets

Real-time communications, like VoIP, are sensitive to end-to-end delay and variation in packet arrival times. They require a steady, reliable stream of packets to provide acceptable quality. End-to-end delay is the time it takes a packet to traverse the communications medium and arrive at its destination. The delay budget for reasonable two-way conversations is approximately 150 milliseconds. When the end-to-end delay exceeds this threshold, callers have a tendency to talk over one other. This compromises the real-time aspect of the conversation, and can be a frustrating experience for the callers.

An audio artifact known as *jitter* can be heard when packets arrive early, late, or out of sequence. Jitter is the result of variation having occurred in the time between packets transmitted and packets received. Excessive jitter causes the users to experience quality degradation during a call. Jitter creates different audio artifacts, depending on its severity and the original audio signal. A VoIP audio stream of a person speaking with a moderate amount of jitter can sounds almost as if the person were talking underwater. This audio artifact sounds similar to very low bitrate *MPEG Layer III (mp3)* encoded audio.

A way to compensate for excessive jitter is to increase the size of the jitter buffer. The jitter buffer for VoIP is responsible for reassembling packet streams. When there is an issue with packets, such as arriving out of sequence or too late, the buffer will try compensate by replaying the last packet received, filling the slot with white 'comfort' noise. However, the jitter buffer needs to be as small as possible to to reduce the delay between the audio being recorded by the sender and being played at the receivers end.

To this end, it is better to transmit VoIP packets as soon as possible, so the jitter buffer can be kept small and good voice quality is perceived by users.

A priority queuing mechanism favouring realtime traffic is neccessary to provide satisactory QoS in the presence of other traffic sources (especially TCP). In the experiments conducted as part of this project, a prioritised queue for VoIP packets allowed the access point call capacity to approach the dedicated network call capacity whilst maintaining QoS contracts.

6.2.4 Use of Multicast Packets

Most computer network applications use unicast packets exclusively for transporting information between points. Some examples of such network applications might web browsers using the *Hyper-text Transfer Protocol (HTTP)* protocol, the *Network File System (NFS)* protocol, and the interaction between an email client and a *Post Office Protocol (POP)* server.

There are cases where the transmission of information using multicast packets, however, is more practical. A multicast packet is the packet switched network equivalent of broadcasting. In this sense, a single packet can be transmitted to multiple destinations. This could be a single subnet, or an entire network. There are two particularly useful aspects about multicast packets on a wireless network.

1. Multicast packets can be transmitted to all nodes connected to an access point simultaneously, as opposed to sending individual packets to associated nodes
2. There is no *Acknowledge (ACK)* at the 802.11 protocol level for these packets, which greatly reduces the overheads of transmission

The use of multicasting conserves the bandwidth of a network because only the transmission of a single packet is necessary rather than sending packets individually addressed to each node. This is especially important for 802.11 networks as they have

limited bandwith compared to conventional ethernet LANs. By removing the requirement of an 802.11 ACK for VoIP packets, **the effective voice call capacity practically doubles**. In simulation, the performance increase measured was close to 100% when using multicast packets.

Multicasting is supported in the 802.11 standards as part of its asynchronous services. An 802.11 client associated with an access point can request a multicast delivery by sending multicast packets in 802.11 unicast data frames directed to the access point. Provided the access point receives this frame correctly, it responds with an 802.11 ACK frame. If the client sending the frame doesn't receive an acknowledgement, it will retransmit the frame after a delay. After receiving the unicast data frame from the client, the access point transmits the data (that the originating client wants to multicast) as a multicast frame. Each of the destination stations can receive the frame, but do not respond with 802.11 ACKs. It would be impractical to implement this, as nodes may join / leave the network silently and the acknowledge packets would take much time and bandwidth. As a result, multicasting doesn't ensure a complete, reliable flow of data but this is a non-issue for VoIP packets. It is better for a VoIP packet to be dropped than arrive outside the jitter window.

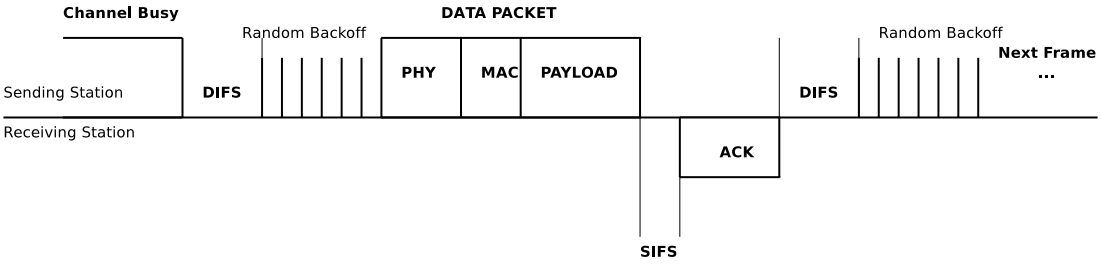

Figure 6.1: The Unicast Packet Transaction on 802.11 Wireless Networks

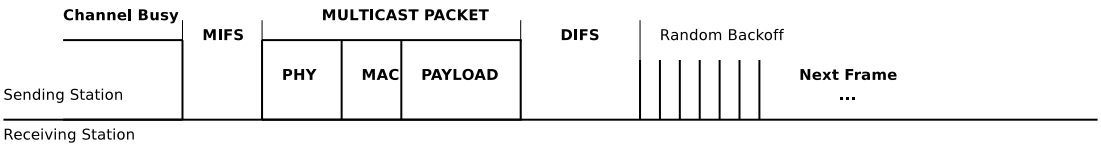

Figure 6.2: Multicast Packet Transaction on 802.11 Wireless Networks

In order to provide improved performance by exploiting multicast packets, it is neccessary to make some configuration changes at the access point and the wireless clients.

On most access points, multicast packets are transmitted at the lowest supported speed by default. For 802.11a, this is 6 Mbits/s whereas 802.11b/g the basic rate is used 1 Mbit/s. When the lowest speeds are used for multicasting VoIP packets, it all but defeats the purpose of performance gains. However, the multicast transmission rate is easily changable on most access points, and can be set to the maximum data rate the wireless network device supports. Provided all nodes have excellent signal reception, the maximum rate can be selected and large performance increases are obtained.

Another issue with multicasting to consider is that multicast frames may experience lower quality of service. With 802.11 networks, multicast packets are delayed when one or more of the wireless clients are using the 802.11 power save mode. If a wireless node that has enabled power saving is associated to an access point, that access point buffers all multicast frames and sends them immediately following the next *Delivery Traffic Indication Message (DTIM)*. This problem is solved by turning power saving off on all wireless clients. When power saving mode is disabled, access points transmit multicast frames as soon as possible instead of waiting for the next DTIM.

6.2.5 Choice of VoIP Codec

Different VoIP codecs affect the performance of VoIP over 802.11 networks in different ways. The choice of VoIP Codec is dependent upon the characteristics of the network involved. Since the major limitation for voice call capacity of 802.11 access points is the minimum framing interval, the codec packet rate plays a larger role in determining performance than the payload size. Common VoIP codecs have a payload size between 20 and 160 bytes, and take a small fraction of the framing interval to be transmitted. For example, a G.729 VoIP packet (60 bytes) can be transmitted with 2 symbols (216 bits per symbol) at the maximum symbol rate of 250, 000 symbols per second[1]. The transmission time is calculated as follows:

$$Transmission\ Time \quad = \quad \frac{Symbols}{Rate}$$

[1]Only valid for 802.11a/g access points, 802.11b networks utilise smaller 8 bit symbols

$$= \frac{3}{250k}$$
$$= 12\ \mu s$$

When compared to the framing interval of 455 μs for 802.11g networks in protected mode, the transmission time of 12 μs seems remarkably small. Of course, it is better to use VoIP codecs that use lower VoIP payloads to minimise bandwidth requirements. However, due to the large symbol size of 802.11a/g networks, there is no difference in the transmission time for packets between 54-81 bytes[2].

The most important consideration when choosing a VoIP codec for implementation on a wireless network is the codec framing interval. For small framing intervals, the end-to-end delay is minimised at the expense of total call capacity. This is due to the transmitting station buffering one frame of call audio before transmission. Of course, for low framing intervals, greater packet rates are required for each VoIP stream (hence, lower call capacity). Long framing intervals (or combining frames) reduce the framing overhead required for transmission of each VoIP stream. However, the end-to-end delay is increased by the length of the framing interval.

In order to maximise the call capacity and QoS of VoIP sessions via choice of codec, it is neccessary to know the characteristics of the wireless network. If the end-to-end delay is high ($>$ 60 ms), it is better to trade some call capacity for lower end-to-end delay using a VoIP codec with a small framing interval, such as G.729 (20 ms). On the other hand, if the network had low end-to-end latency ($<$ 60 ms) extra call capacity could be obtained by using a codec with a larger framing interval, such as G.723.1 (30 ms).

[2]The symbol size is 216 bits for 802.11a/g, which is 27 bytes. It is not possible to transmit a fraction of a symbol, therefore three symbols are required for packets between 54-81 bytes in length

6.2.6 Type of Wireless Network

From the 802.11 networks considered, 802.11a & 802.11g networks provide the best performance. The call capacities of access points belonging to these two networks are identical when 802.11g is used in 'g-only' mode without CTS-to-self protection. By enabling support for 802.11b devices, the call capacity of 802.11g access points is greatly reduced. Between 55-75% of the call capacity is lost when an 802.11b device associates with an 802.11g access point, depending on whether short or long 802.11b preambles are used.

Legacy 802.11b devices are inferior to 802.11a & 802.11g due to their long minimum framing intervals and relatively low data rates (maximum of 11 Mbits/s according to the 802.11b standards). Far greater performance can obtained using 802.11a/g networks when the network is required to carry VoIP calls.

6.3 Prioritised Packet Scheduler

A prioritised scheduler was implemented using the NS-2 network simulator to test the performance of VoIP sessions on wireless networks with other traffic sources (particularly TCP). This was accomplished by writing network components for NS-2 to favour VoIP traffic over all other traffic sources. The main components required to be written were:

- A Voice Over IP Agent
- The addition of a VoIP packet type for NS-2 (multicasting enabled by default)
- Prioritised queueing for the VoIP packet type

After these networking components were written, some changes to configuration files and sources were required in order to implement them. The most significant of these changes were adding the priority queue and the VoIP agent components to the ns-2.29 makefile, and adding the packet type PT_VOIP to the definitions in packet.h. After

testing and debugging the new network components with the GNU *Data Display Debugger (DDD)*, the wireless simulations were repeated using the prioritised scheduling algorithm.

6.3.1 Voice Over IP Agent

In the initial simulations, voice over IP traffic was approximated by attaching a constant bit rate source generating the required number of packets per second to a UDP agent. Whilst this was an accurate model for the behaviour of VoIP traffic in simulation, it did not allow differentiation between UDP traffic and VoIP traffic types. Any attempt to give VoIP packets priority in this model would apply the same priority to other, non-voip UDP packets.

In order to solve this problem, a VoIP agent for the NS-2 simulator was written in C++, using the UDP agent (udp.cc) as a base. Another requirement for the improvement of VoIP performance was the use of multicast packets for VoIP packets. This was incorporated into the new VoIP agent written for NS-2. The major differences between the base UDP agent and the derived VoIP agent were:

- A new packet type (PF_VOIP) is defined to enable NS-2 networking components to differentiate between VoIP & all other traffic
- Multicasting is enabled by default for the VoIP agent to disable the 802.11 ACK at the receiving end
- The default packet size is set to 60 bytes (G.729 VoIP Codec)
- A default packet rate of 50 packets per second is used (G.729 VoIP Codec)

Modification of the base UDP agent provided in the NS-2 simulator to create a VoIP agent set the framework in place for a priority based scheduler for VoIP packets.

6.3.2 Implementation of A Priority Queue for VoIP Packets

In order to provide the required QoS for voice over IP packets in simulation, a prioritised packet scheduler was written in C++. Via network simulation, it was found that *delay insensitive* TCP traffic interferes with VoIP connections by forcing full queues and delaying VoIP packets. TCP sources only reduce their transmission rate when packet loss is encountered, and periodically attempt to exponentially increase their thoughput which forces VoIP packets to be dropped from queues. In order to increase the fairness for treatment of VoIP packets, a prioritised scheduling algorithm was designed.

NS-2 has a base queue class called 'Queue' that provides generic *First In First Out (FIFO)* queue and supporting functions. A lot of the neccessary functions for a network queue are provide by the derived 'PacketQueue' class. Member functions are provided in this class to access and manipulate the queue, such as dropping and transmitting packets. Through the use of C++ inheritance, the VoIP priority scheduler derived this basic functionaliy from the 'Queue' class included in the NS-2 sources.

```
/*
 * A priority based queue for increasing the call capacity of 802.11 access pts
 */

// Inherit the 'Queue' class, and create a packet queue
class VoipPri : public Queue {
  public:
        VoipPri() {
                q_ = new PacketQueue;
                pq_ = q_;                                                    10
                bind_bool("summarystats_", &summarystats);
                bind_bool("queue_in_bytes_", &qib_); // boolean: q in bytes?
                bind("mean_pktsize_", &mean_pktsize_);
        }
        ~VoipPri() {
                delete q_;
        }
}
```

The priority sheduling algorithm works by ordering packets in the queue based on their packet type, and the time they were received. Each time a new packet is received, it is queued immediately if there is enough room on the queue to accomodate the packet. In the event that there is not enough space on the queue, the packet type is checked. If the packet is a low priority (non-voip), it is discarded immediately.

However, if the received packet is a critical packet (VoIP) the algorithm looks through the queue for a low priority packet to remove to make room for the received packet. If low priority packets exist on the queue, one is dropped to make room for the critical packet. After this, the critical packet is enqueued at the tail, and each non-critical packet on the queue is removed in turn (oldest first), and re-queued at the tail after the critical packet. However, if no low priority packets exist on the queue, the received critical packet is dropped. This can occur when the call capacity of the access point is exceeded, and is impossible to avoid. The process described above is depicted in the form of a software flow chart in figure 6.3.

6.4 Simulation Results

The networking scenario simulated to test the call capacity of 802.11 access points in the presence of TCP traffic was the same as the simulation in the previous chapter, with two exceptions:

1. The VoIP agent written for NS-2 was sustituted for the UDP transport agent used in the simulation scripts.
2. A priority queue was used to give VoIP traffic preference

In this scenario, the TCP traffic source begins soon after the start of the simulation. A new VoIP session is started each 10 seconds until any of the VoIP sessions has an average packet loss exceeding 1%. When this occured, no more nodes were added. After simulation, the packet loss, jitter and average end to end delay were found using the unix *awk* utility. The results obtained from simulation were tabulated, and performance

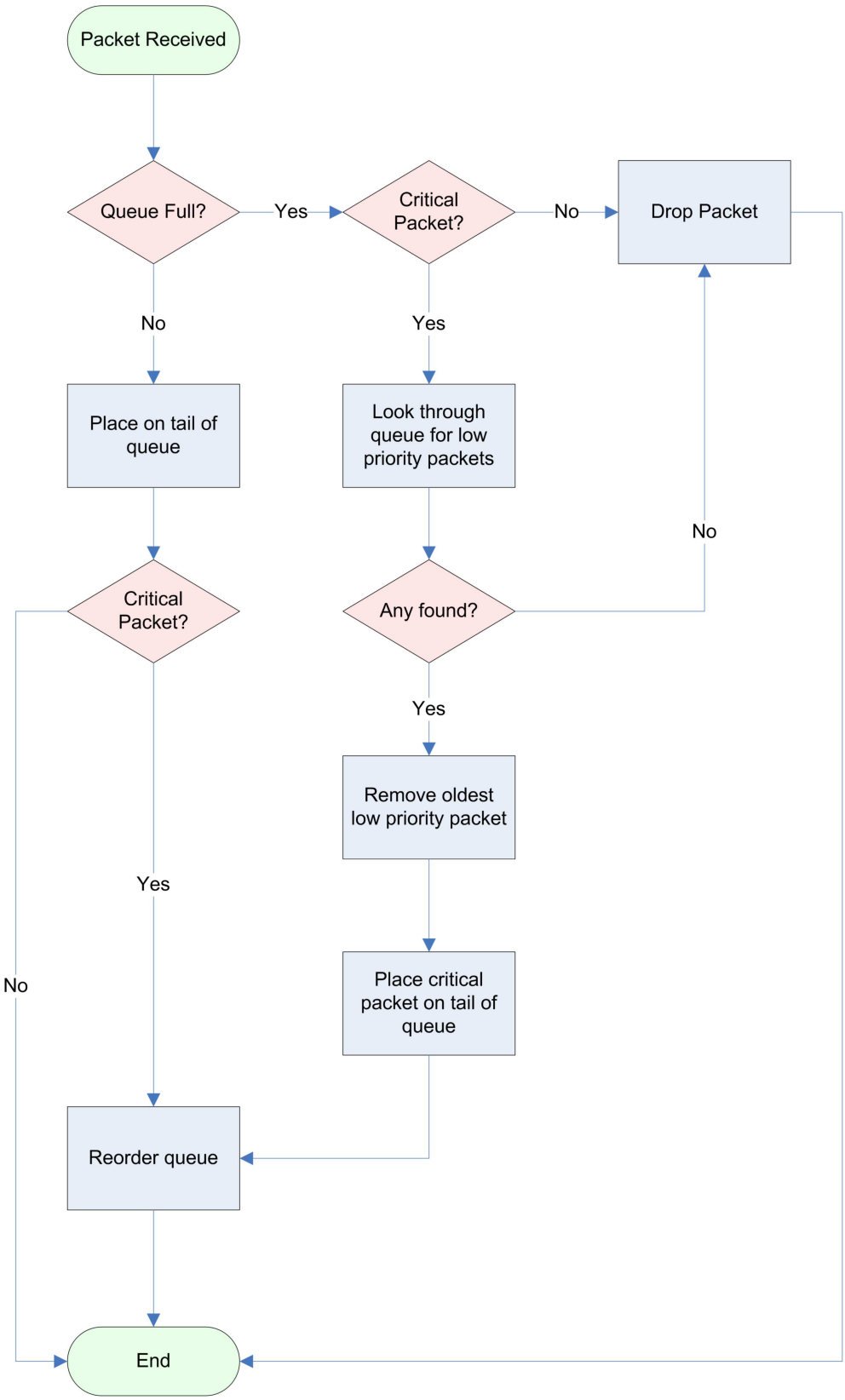

Figure 6.3: Priority Queue Flow Chart

issues analysed. The capacity at which both traffic contractual constraints (less than 1% loss & less than 1% of packets delayed by 25 ms was recorded as the call capacity for that particular networking configuration.

6.4.1 VoIP Coexisting with TCP Traffic

The network simulation results for all 802.11 acess point types indicated that the addition of a priority queue greatly improves the VoIP call capacity in the presence of TCP traffic sources. With the priority queue for VoIP packets in place, the call capacity of 802.11 access points reached or came close to that of the dedicated network scenario described in the previous chapter. It should however, be noted that as the call capacity was approached, the TCP throughput declined to almost zero. As can be seen in figure 6.4, there appears to be a linear tradeoff between TCP traffic and the number of simultaneous VoIP connections.

Rate (Mb/s)	**Predicted Capacity**	**Without TCP Capacity**	**With TCP Capacity**	**Difference** (%)
6	49.0	43	42	2.3
9	59.5	49	49	0.0
12	66.6	56	56	0.0
18	75.8	63	62	1.6
24	81.3	69	69	0.0
36	87.7	75	75	0.0
48	91.3	76	75	1.3
54	92.6	76	75	1.3

Table 6.1: Call Capacity for 802.11a Access Points using Priority Queuing

6.4.2 Multicasting Technique

By removing the requirement of an 802.11 ACK for VoIP packets, the framing interval is significantly reduced. In turn, this allows 802.11 access points to transmit VoIP packets more rapidly, and avoid retransmissions caused by packet corruption. In simulation, the

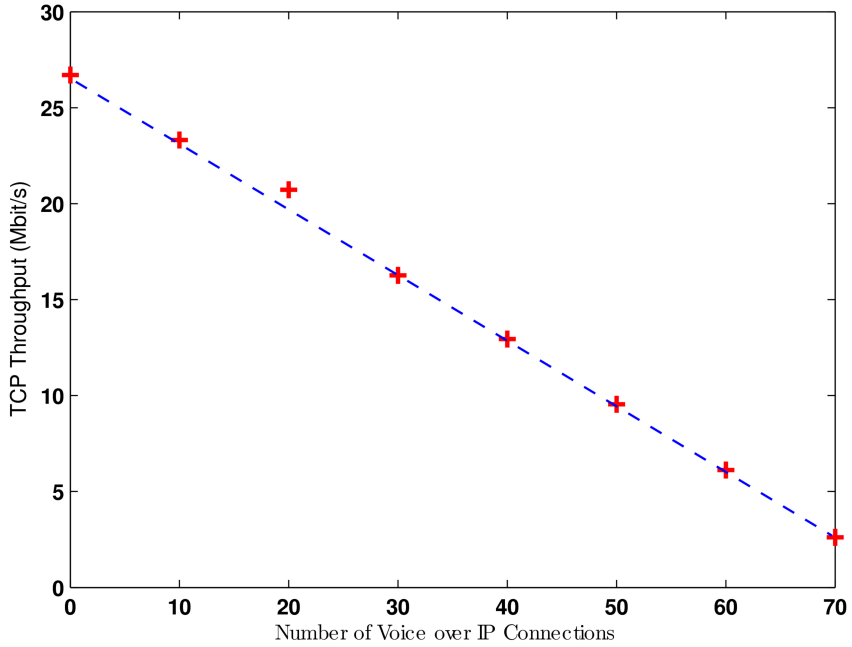

Figure 6.4: TCP Throughput versus Number of VoIP Connections for 802.11a with Prioritised Scheduler

Rate (Mb/s)	Predicted Capacity	Without TCP Capacity	With TCP Capacity	Difference (%)
1	13.5	12	12	0.0
2	19.9	17	17	0.0
5.5	28.5	26	26	0.0
11	32.6	29	29	0.0
22	35.1	32	32	0.0

Table 6.2: Call Capacity for 802.11b Access Points with Priority Queuing (Short Preamble)

Rate (Mb/s)	Predicted Capacity	Without TCP Capacity	With TCP Capacity	Difference (%)
1	10.7	9	9	0.0
2	14.4	13	13	0.0
5.5	18.4	15	15	0.0
11	20.1	18	18	0.0
22	21.0	18	18	0.0

Table 6.3: Call Capacity for 802.11b Access Points with Priority Queuing (Long Preamble)

Rate (Mb/s)	Predicted Capacity	Without TCP Capacity	With TCP Capacity	Difference (%)
6	49.0	43	42	2.3
9	59.5	49	49	0.0
12	66.6	56	56	0.0
18	75.8	63	62	1.6
24	81.3	69	69	0.0
36	87.7	75	75	0.0
48	91.3	76	75	1.3
54	92.6	76	75	1.3

Table 6.4: Call Capacity for 802.11g Access Points with Priority Queuing (Non - Protected)

Rate (Mb/s)	Predicted Capacity	Without TCP Capacity	With TCP Capacity	Difference (%)
6	22.9	20	20	0.0
9	24.9	20	20	0.0
12	26.1	22	22	0.0
18	27.4	23	23	0.0
24	28.1	25	25	0.0
36	28.8	25	25	0.0
48	29.2	26	26	0.0
54	29.3	26	26	0.0

Table 6.5: Call Capacity for 802.11g per Access Point with Priority Queuing (Protected, Long Preamble)

Rate (Mb/s)	Predicted Capacity	Without TCP Capacity	With TCP Capacity	Difference (%)
6	29.3	27	27	0.0
9	32.8	29	29	0.0
12	34.8	31	31	0.0
18	37.2	35	35	0.0
24	38.5	35	35	0.0
36	39.8	37	37	0.0
48	40.6	38	38	0.0
54	40.8	38	38	0.0

Table 6.6: Call Capacity for 802.11g Access Points with Priority Queuing (Protected, Short Preamble)

performance increase measured when using multicast packets varied with the network type. For 802.11b networks, an 802.11 ACK takes 203 μs to transmit, which is almost half the the time required for a single VoIP transaction (499 μs at 11 Mb/s using the G.729 VoIP Codec). Less impressive performance increases are had for 802.11a/g networks due to tighter timing specifications. An 802.11a ACK only takes 24 μs, which is roughly a quarter of the time required for a single VoIP transaction (106 μs at 54 Mb/s using the G.729 VoIP Codec). For 802.11g networks, the 802.11 ACK takes slightly longer, and greater peformance increases are realised as a consequence. The time required for a VoIP transaction on a non-protected 802.11g network is 76 μs with multicast packets, as opposed to 106 μs using unicast packets (an improvement of 28.3%).

Rate (Mb/s)	**Predicted Capacity**	**Simulation Capacity**	**Difference** (%)
6	56.2	52	7.5
9	70.4	65	7.7
12	80.6	76	5.7
18	90.5	82	9.4
24	103.1	94	8.8
36	113.6	102	10.2
48	119.8	105	12.4
54	121.9	106	13.0

Table 6.7: Call Capacity for 802.11a Access Points using Priority Queuing & Multicasting

6.4.3 Discussion of Simulation Results

A main objective of the prioritised packet scheduler was to increase the performance of VoIP in the presence of external traffic sources. By using the prioritised packet scheduler, this objective has been met. In the presence of a single TCP traffic source, the prioritised packet scheduler allowed the full 'dedicated network' capacity whilst maintaining QoS contracts in most cases. In the cases that the full 'dedicated network' capacity was not obtained, the call capacity was at least 97% of the dedicated capacity.

Rate (Mb/s)	Predicted Capacity	Simulation Capacity	Difference (%)
6	58.1	53	8.7
9	73.5	67	8.8
12	84.7	77	9.1
18	100.0	93	7.0
24	109.9	98	10.8
36	122.0	108	11.5
48	129.0	114	11.6
54	131.6	115	12.6

Table 6.8: Call Capacity for 802.11g Access Points using Priority Queuing & Multicasting

Rate (Mb/s)	Predicted Capacity	Simulation Capacity	Difference (%)
1	15.7	14	10.8
2	25.3	23	9.1
5.5	41.1	35	14.8
11	50.1	42	16.2
22	56.2	43	23.5

Table 6.9: Call Capacity for 802.11b Access Points with Priority Queuing & Multicasting (Short Preamble)

Rate (Mb/s)	Predicted Capacity	Simulation Capacity	Difference (%)
1	13.7	12	12.4
2	20.3	18	11.3
5.5	29.5	25	15.25
11	33.8	27	20.1
22	36.5	29	20.5

Table 6.10: Call Capacity for 802.11b Access Points with Priority Queuing & Multicasting (Long Preamble)

Without the prioritised packet scheduler, 802.11 networks cannot maintain the QoS contracts for VoIP consistently for even one VoIP session contending with TCP Traffic[1].

With or without the presence of external traffic sources, the use of multicast packets for VoIP been shown to significantly increase the call capacity of 802.11 access points. Access point call capacity increases between 25 and 40 percent were found to be possible in simulation by exploiting multicast packets. In an infrastructure based network, the only downside to using multicast packets for VoIP traffic is a slight increase in complexity on the VoIP client side. Each and every VoIP client associated with an access point will receive all VoIP packets, some of which will be intended for other nodes.

The solution to this problem is to modify the VoIP client slightly to identify the intended recipient of VoIP packets. This recipient information could be stamped in unused bits of the RTP / IP / UDP headers, or added to the UDP data segment of VoIP packets. Neither of these solutions incur additional overhead for any of the common VoIP codecs considered for 802.11a or 802.11g networks as a result of the large symbol size (216 bits). This is a result of none of the VoIP codecs considered in this project having packet sizes that are even multiples of 216 bits (meaning the last symbol of a transmitted VoIP packet is padded with zeros). For 802.11b networks, an additional byte in the UDP datagram would cause negligible additional overhead (between 0.7 and 8 μs, depending on the data rate).

The most effective method of increasing access point capacity for 'dedicated networks' was also the simplest. By setting the RTS / CTS threshold above the VoIP packet length, the VoIP call capacity was approximately tripled in simulation. Although the RTS / CTS mechanism is required to solve the hidden node problem, it has arguable benefits in a typical office situation (Kaixin Xu, 2002). This is due to the effectiveness

[1] Packet loss for VoIP in the presence of TCP traffic is bursty. Although a single connection could maintain a packet loss below 1% overall, the packet loss spiked well above this periodically. A voice over IP conversation conducted in such a scenario would exhibit audio degradation that would result in an intelligible voice stream with 'annoying' audio artifacts

of carrier sensing mechanism of wireless devices. A wireless device can sense a carrier signal at distances where the actual transmitted information is unintelligible. For these reasons, the author believes better performance in an office environment can be had by:

1. Disabling the RTS / CTS mechanism by setting the threshold to 2347
2. Ensure that neighbouring wireless networks occupy a different channel
3. Increasing the carrier sense sensitivity to appropriate levels

A hidden node on a network without RTS / CTS causes problems due to packet collisions, and should be avoided wherever possible. Increasing the number of, or strategically placing access points in the area reduces the likelyhood of the hidden node problem occuring, albeit at additional financial expence. A greater number of access points increases the network performance in general if enough devices utilise them.

Perhaps the best feature of the prioritised packet scheduler is that the improvement of VoIP performance in the presence of TCP traffic is not at the expense of TCP throughput. That is to say, when an 802.11 access point is supporting a certain number of VoIP calls and has a TCP traffic source, the TCP throughput is the same with or without the prioritised packet scheduler. This is an important result, as it infers that the performance issues of VoIP on 802.11 networks can be largely solved by using the prioritised packet scheduler. However, by solving one problem, the prioritised packet scheduler has created another.

As the number of VoIP calls per access point increases to the call capacity of the access point, the TCP throughput drops to zero. This may or may not be desirable behaviour, depending on what the main function of the wireless network is. Another problem is that when the access point's call capacity is exceeded, packet losses begin to surge. In simulation, increasing the number of VoIP sessions just one above the call capacity found in simulation led to downstream VoIP packet losses of over 5% for 802.11(a/b/g) networks.

A solution to this issue might be to limit the number of concurrent VoIP sessions at the AP. This could be accomplished by recording each VoIP stream's source and destination addresses to keep track of the number of VoIP sessions in existance. When another VoIP session attempt begins that would cause the network to become overloaded, packets from that session could be discarded, perhaps until another VoIP session ends. By effectively limiting the number of VoIP sessions on the wireless network, it is possible maintain VoIP performance and reserve some bandwidth for other types of traffic.

Chapter 7

Conclusions And Further Work

7.1 Introduction

The main aim of this project was to research performance issues that exist in 802.11 networks with motivation to design and test enhancements to the way wireless networks operate. Research was undertaken into the 802.11 protocol, the TCP/IP protocol and Voice over IP technology, and serious performance bottlenecks were found. Through extensive network simulation and an implementation of a prioritised packet scheduler, the major issue of VoIP not coexisting with TCP traffic over wireless 802.11 networks has been solved. This chapter is intended to highlight the major discoveries made in the course of this project, along with proposed solutions to issues in 802.11 performance and a critical analysis of the robustness of these solutions. The chapter closes by mentioning certain shortcomings of the project undertaken, and makes mention of further work to be undertaken in this area.

7.2 Establishment of Issues with 802.11 Networks

Several performance issues with 802.11 networks were found as a result of extensive research. Many papers have been written on the performance issues due to the imperfect implementation of the TCP protocol for wireless, and the difficulties faced in securing a shared medium. The main issues with wireless 802.11 performance were found to be:

- A complete lack of QoS facilities existing for 802.11
- The inability for VoIP and TCP traffic to coexist
- Security issues, especially due to the weakness of WEP

Issues with security in 802.11 networks has been rectified for the most part, and were not the main concern of this project. However, the inability for VoIP and TCP traffic to coexist and the lack of QoS facilities for 802.11 networks were real challenges with no complete solution. From research, it was found that the IEEE devoted a task group to solve VoIP issues by introducing QoS mechanisms via the 'e' ammendment (approved

late 2005). However, mainstream 802.11 devices are yet to fully adopt the new 'e' ammendment.

7.3 Current Research

Numerous methods and attempts have been made to solve the issue of the poor support for real-time services and the inability for such services to coexist with other traffic sources in 802.11 networks. Some efforts included:

- An implementation of token ring for 802.11
- Modifying the 802.11 protocol
- Using commercial VoIP access points such as Meru^{TM} access points

Implementing a token ring protocol for 802.11 is a reasonable solution to the problems VoIP has coexisting with external traffic sources where the number of wireless nodes is low. However, the token must be passed between the access point and individual nodes on the network in order for them to take control of the communications medium and transmit their data. Upon closer inspection, this solution had major scalability issues due to the overhead in passing the token to nodes that do not wish to transmit, and possible delays if the token is 'dropped'

Modification to the 802.11 protocol is the ideal solution to the QoS issues explored in this report, which is exactly what the IEEE have proposed with their 'e' ammendment to the protocol. Modifying the 802.11 MAC protocol however, removes compatibility with the 802.11 standard (except for ammendments made by the IEEE), which leads to compatibility issues. The author chose not to make modifications at the MAC layer for these reasons and because significant performance increases may be had without changing the way things work at the MAC level.

Commercialised AP's especially designed for VoIP over wireless had impressive specifications, and advanced features such as handoff capability and call synchronisation.

Meru's enterprise access point was found to be an excellent solution for VoIP on 802.11 networks. Unfortunately, Meru access points are quite expensive which limits their use primarily for business users. Although being a viable solution to the QoS problem for VoIP, it did not cater for existing wireless infrastructure.

Other researchers suggested dedicating networks to VoIP in order to have good QoS for voice calls and maintain a efficient data network. Of course, this doesn't actually solve the original problem. In fact, it sidesteps the problem at hand by suggesting the use of financial resources to add more network infrastructure. This solution does not attempt to effectively utilise existing wireless networking infrastructure that may be in place.

7.4 Verifying Issues via Simulation

Using the NS-2 network simulator, the existance of performance issues in 802.11 networks was verified. The 'dedicated' voice capacities for various 802.11 configurations were tested, with the simulation results agreeing with theoretical capacity calculations to a large extent. The simulations did not produce the exact results calculated, with discreptancies being worse for the older 802.11b networks. Generally simulation results were within 12% of calculated capacity, but differed by as much as 20%. Likely causes of these discreptancies were suggested to be:

- Limitations of the network model used in NS-2
- Errors due to bugs in the NS-2 simulator's wireless implementations
- Collisions and packet loss due to corruption, which was not accounted for in the access point capacity calculations

Despite the inconsistencies between the theoretical capacity calcuations of 802.11 access point call capacity and simulated call capacity, the results were perceived to be adequate reflections of the performance of actual 802.11 networks. The actual call capacities of

802.11 access points found in simulation was deemed less important than the percentage increases obtained via changes to the way 802.11 wireless networks handle VoIP traffic.

7.5 Determining Solutions to 802.11 Issues

From network simulation, various issues with the performance of VoIP on 802.11 networks were exposed, validating the performance issues discovered via research. The major issues were the low VoIP call capacity of 802.11 access points, (especially when compared to the theoretical capacity), and the intolerance of other traffic sources by VoIP. Wireless 802.11a/g access points have enough raw bandwidth to support (in principle) thousands of VoIP connections according to capacity calcuations made in this project. However, a stock 802.11a/g access point can only support about 90 simultaneous VoIP connections (in the absence of external traffic sources) out of the box.

The QoS contractual constrains of <1% packet loss and <1% of packets arriving with less than 25 ms variation in delay for VoIP sessions was not possible in simulation with the addition of a TCP traffic source. Although the packet loss could be kept below <1% on average, packet loss 'surged' to several times this threshold occasionally as the TCP traffic source attempted to increase its output.

After numerous network simulations and extensive research into 802.11 networks, several configuration changes were discovered that could significantly increase the performance of VoIP sessions over wireless 802.11 networks. Configurational factors that influenced the performance of VoIP over wireless were:

- Decreasing MTU to facilitate the flow of smaller packets
- Correct setting of RTS-CTS threshold
- Choice of VoIP codec depending on network
- Choice of wireless network type (802.11g being the clear winner)

However, in order to enable VoIP to coexist with other traffic sources, network components of the NS-2 network simulator had to be modified. A prioritised packet scheduler was developed to meet this need, complete with multicasting packets to reduce overhead due to the (802.11 ACK)

7.6 Development of a Prioritised Packet Scheduler

By developing new network components for the NS-2 network simulator, it was possible to implement a priority queuing mechanism for VoIP packets on an 802.11 network. This was accomplished by creating VoIP traffic type, derived from the UDP class in the NS-2 sources, and writing a queue that favours the flow of this traffic. The ability to use multicast packets was also incorporated into the prioritised packet scheduler to enable larger peformance increases.

In simulation, the prioritised packet scheduler allowed VoIP to coexist with TCP traffic up to the call capacity limit of the access point. An attractive feature of the prioritised packet scheduler was that the performance gains for VoIP traffic was not at the expense of TCP throughput. That is to say, with 10 VoIP connections and a TCP traffic source, the TCP throughput is identical whether the prioritised packet scheduler is used or not (so long as the number of VoIP connections does not meet or exceed the call capacity of the access point found in simulation).

Multicasting provided significant capacity improvements just by itself. The increase in call capacity realised by exploiting multicast packets varied according to the network type used. Performance increases measured in simulation were between 25-40 % by using muticast packets. Although to take advantage of multicast packets for VoIP on a wireless network in this fashion some added complexity at the VoIP client side is required, the call capacity gained by are well worthwhile.

Setting the RTS-CTS threshold above the VoIP packet length has been found to approximately triple the access point VoIP call capacity in simulation. In the presence of

TCP traffic, it was found to be best to disable the RTS / CTS mechanism entirely by setting the RTS / CTS threshold to 2347 bytes. The only downside to disabling the RTS / CTS mechanism on 802.11 wireless devices is that the network becomes susceptible to the hidden and exposed node problems. In most cases, the hidden / exposed node problems don't become an issue due to carrier sensing and adjacent networks being set to different channels.

7.7 Critical Evaluation of Solutions

Although the prioritised packet scheduler allowed VoIP sessions to coexist with other traffic sources, the behaviour it exibits when nearing the access point call capacity could be undesirable in some circumstances. The prioritised packet scheduler promotes VoIP packets, perhaps a little to zealously. In the previous chapter, it was found that the TCP throughput tapered off linearly with the number of VoIP sessions on an access point using the prioritised packet scheduler. Another important result was the effect of adding an extra VoIP session to an access point that was already at its call capacity was to cause the VoIP packet loss to surge periodically at about 5%. Both of these problems can be substancially solved by limiting the VoIP capacity at the access point to a value somewhat below the call capacity.

Another downfall of the prioritised packet scheduler is its requirement for a specially modified VoIP client in order to take advantage of multicast packets. The modifications required for the VoIP client are quite small, and could be completed by an accomplished C/C++ programmer in a matter of hours if the sources for the client were available (and they may not be available). It was not neccessary to do this to test the prioritised packet scheduler in network simulation, so it is possible that other challenges exist in the test bed implementation.

Disabling the RTS / CTS mechanism causes the wireless network to become susceptible to the hidden and exposed node problems. In the situations that this occurs, the effectiveness of the 802.11 access points to carry voice traffic could be reduced due to collisions on the network. In most cases, the hidden / exposed node problems

don't become an issue due to carrier sensing mechanism of 802.11 devices and physical proximity of nodes to an access point. The exposed node problem can be removed by ensuring adjacent networks are set up to use different channels.

7.8 Proposed Improvements for 802.11 Networks

All things considered, the overall recommendations to increase the peformance of VoIP with wireless 802.11 networks are:

1. Disable RTS / CTS, or at least set it higher than VoIP packets if the majority of wireless nodes are in close proximity to the access point to approximately triple the VoIP call capacity per access point. This effectively triples the VoIP call capacity of 802.11 access points, albeit with a small risk of the hidden / exposed node problems occuring

2. Use a VoIP codec that is suited to the individual wireless network. If the end-to-end delay is high ($>$ 60 ms), it is better to trade some call capacity for lower end-to-end delay using a VoIP codec with a small framing interval, such as G.729 (20 ms). On the other hand, if the network had low end-to-end latency ($<$ 60 ms) extra call capacity could be obtained by using a codec with a larger framing interval, such as G.723.1 (30 ms).

3. Set the MTU to a smaller value than the default 1500 bytes which provides the best balance between VoIP performance and TCP throughput. An MTU of 960 marginally improved the consistency of the VoIP packet arrival.

4. Use a prioritised packet scheduler where VoIP traffic must coexist with TCP traffic

5. Exploit the lower framing interval of 802.11 multicast packets for VoIP to increase the call capacity of access points between 25-40 %

6. If it is possible, better performance can be had using 802.11g compatible networking equipment exclusively, and in non-protected mode. Whilst the performance of 802.11a and 802.11 g is identical for unicast packets, 802.11g has a slightly

shorter framing interval for multicast packets and is the clear winner for VoIP with the prioitised packet scheduler solution.

7.9 Evaluation of Performance on a Wireless Networking Test Bed

Unfortunately, time did not permit the completion of the final objective in this project. As a result, the prioritised packet scheduler has not yet been trialled on a wireless networking testbed. It is possible that testing the prioritised packet scheduler on a wireless networking test bed may have brought other issues to light, or shown a performance impact different to that found in simulation.

Regardless, the NS-2 network simulator is an internationally renouned for its accuracy and performance as a simulation tool, which infers that success in simulation is very likely to translate to success in practice in this case. The author considers that the network simulations carried out gave good indications of the performance gains that can be expected from a full implementation of the prioritised packet scheduler on a wireless networking test bed.

7.10 Shortcomings & Further Work

Several areas of research in this area were neglected due to time constraints. Further research in the area of issues with 802.11 wireless network performance could be to explore other hurdles wireless networks face in more detail. Specific examples might include:

- In wireless 802.11 networks, low TCP throughput relative to the connection speed is a problem, according to Gast (2002). Apparently, the TCP throughput for 802.11 wireless networks is slightly more than half the bandwidth. There is obviously room for improvement in a system that is less than 50% efficient.

- Current TCP alorithms are not entirely suited to for wireless networks, notably the contention window algorithm. Wireless 802.11 networks have a higher bit error rate than ethernets the TCP protocol was designed for. As a result, TCP does not achieve optimal performance on a wireless network. Via modification to TCP parameters / algorithms, it should be possible to tailor the TCP protocol to its application on wireless networks and receive performance improvements as a result.

- An access point capacity analysis for *Variable BitRate (VBR)* VoIP codecs was one of the things missing from this project. VBR codecs get their name from their ability to change the transmitting bit rate on the fly, notably transmitting at very low bitrates in periods where the caller is not speaking. Despite the lower bandwidth requirements of VoIP, its suitability on 802.11 networks is uncertain due to its intermittent nature of transmission.

- The possibility using *Cross-Layer Coupled Coding* to improve the performance of VoIP on congested 802.11 networks requires investigation. Cross-Layer coupled coding allows communication between the network layer and the VoIP compression codec. The VoIP codec can get information about current network conditions such as packet loss or delay, and request bandwidth based on these factors.

Dispite the shortcomings and limitations of this project, the majority of the objectives were met to a high degree. Through this project, several performance issues in 802.11 wireless networks have been solved with substantial progress towards a physical impementation of the prioritised packet scheduler. The findings and analyses conducted in this project would provide a good basis for further research in the area of wireless 802.11 networks and voice over the internet protocol.

References

Marco Gruteser Ashish Jain. Benefits of packet aggregation in ad-hoc wireless network. Technical report, University of Colorado at Boulder, 2003.

IEEE Standards Board. Wireless lan medium access control and physical layer specifications. Technical report, IEEE, 1999.

Douglas E. Comer. *Internetworking with TCP/IP*. Prentice Hall, 2000.

Fouad A. Tobagi David P. Hole. Capacity of an ieee 802.11b wireless lan supporting voip. Technical report, Stanford University, Stanford, California, 2004.

John Fitzpatrick. SCTP based Handover Mechanism for VoIP. Technical report, University College Dublin, Ireland, 2004.

Matthew S. Gast. *802.11 Wireless Networks - The Definitive Guide*. O'Reilly & Associates, Inc, 2002.

Andrei Gurtov. Modeling wireless links for transport protocols. Technical report, University of Helsinki, 2003.

Sri Harsha. Capacity Estimation of Combined VoIP and TCP File Transfers. Technical report, Indian Institute of Science, Bangalore, 2005.

Wendi Beth Heinzelman. Application-specific protocol architectures for wireless networks. Technical report, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2000.

IEEE. ANSI/IEEE 802.11 Standard. Technical report, Institute of Electrical & Electronic Engineers, 1999.

Mario Gerla Kaixin Xu. Effectiveness of RTS / CTS handshake in ieee 802.11 based ad hoc networks. Technical report, UCLA Computer Science Department, 2002.

Stefan Mangold. IEEE 802.11e Wireless LAN for Quality of Service. Technical report, Aachen University of Technology, Germany, 2005.

Matthew Mathis. Forward acknowledgment: Refining TCP congestion control. Technical report, Pittsburgh Supercomputing Center, 2002.

Duke Lee Mustafa Ergen. Wireless token ring protocol-performance comparison with 802.11. Technical report, University of California, Berkley, 2003.

Frank Ohrtman. *Voice Over 802.11.* Artech House, 2004.

Frank Ohrtman. *Wi-Fi Handbook.* McGraw-Hill, 2003.

Neeli Prasad. *802.11 WLANs and IP Networking : Security, QoS, and Mobility.* Artech House, 2005.

Charles N. Thurwatcher. *Wireless Networking.* Prentice Hall, 2002.

Daniel Wong. *Wireless Internet Telecommunications.* Artech House, 2005.

David J. Wright. *Voice over Packet Networks.* J.W. Wiley, 2001.

Kaixin Xu. Effectiveness of RTS/CTS handshake in IEEE 802.11 based networks. Technical report, UCLA Computer Science Department, 2003.

Appendix A

Project Specification

Appendix B

NS-2 Networking Components

B.1 Prioritised Scheduling Queue for Voip

```
#include "voip-priqueue.h"

static class VoipPriClass : public TclClass {
 public:
        VoipPriClass() : TclClass("Queue/VoipPri") {}
        TclObject* create(int, const char*const*) {
                return (new VoipPri);
        }
} class_voip_pri;

void VoipPri::reset()
{
        Queue::reset();
}

int
VoipPri::command(int argc, const char*const* argv)
{
        if (argc==2) {
                if (strcmp(argv[1], "printstats") == 0) {
                        print_summarystats();
                        return (TCL_OK);
                }
                if (strcmp(argv[1], "shrink-queue") == 0) {
                        shrink_queue();
                        return (TCL_OK);
                }
        }
        if (argc == 3) {
                if (!strcmp(argv[1], "packetqueue-attach")) {
                        delete q_;
                        if (!(q_ = (PacketQueue*) TclObject::lookup(argv[2])))
                                return (TCL_ERROR);
                        else {
                                pq_ = q_;
                                return (TCL_OK);
                        }
                }
```

```
        }
        return Queue::command(argc, argv);
}

/*
 * Priority queueing
 */
void VoipPri::enque(Packet* p)
{
        // If we care about statistics, update them
        if (summarystats) {
                Queue::updateStats(qib_?q_->byteLength():q_->length());
        }

        // Update queue information
        int qlimBytes = qlim_ * mean_pktsize_;
        if ((!qib_ && (q_->length() + 1) >= qlim_) ||
        (qib_ && (q_->byteLength() + hdr_cmn::access(p)->size()) >= qlimBytes)){

                // If the queue would overflow if we added this packet,
                // check to see what type of packet we have...
                struct hdr_cmn *ch = HDR_CMN(p);

                // ...if it isn't a udp packet, just get rid of it...
                if (ch->ptype() != PT_VOIP)
                {
                        drop(p);
                }

                // ...It's a VoIP packet, look for something else to kill...
                else {

                        int i = 0;
                        bool found_lp = false;

                        while (i < q_->length() && !found_lp)
                        {
                        Packet *pp = q_->lookup(i);
                        struct hdr_cmn *ch = HDR_CMN(pp);
```

```
                    // ... if we find one, kill it...
                if (ch->ptype() != PT_VOIP)
                {
                    q_->remove(pp);
                    drop(pp);
                    q_->enque(p);
                    found_lp = true;
                }
                }

                // If we tried our best, but could not find any low priority packets
                // to get rid of to make room, we must discard the newest packet
                if (!found_lp) drop(p);
        }
}

// The queue is not full, & this is a voip packet, enque with priority
else
{
        q_->enque(p);

        // Sort queue, VoIP packets in front of TCP packets.
        struct hdr_cmn *ch = HDR_CMN(p);
        if (ch->ptype() == PT_VOIP)
        {
                // search the queue, looking for the first non-critical packet
                int i=0;
                while (i < q_->length())
                {
                    Packet* pp_ = q_->lookup(i);
                    struct hdr_cmn *ch_ = HDR_CMN(pp_);

                    // if we find one, pull it off & put at the end of the line
                    if (ch_->ptype() != PT_VOIP)
                    {
                                Packet oldpack = *pp_;
                                q_->remove(pp_);
                                q_->enque(pp_);
                    }
                    i++;
```

```
                }
        }
    }
}

//AG if queue size changes, we drop excessive packets...
void VoipPri::shrink_queue()
{
    int qlimBytes = qlim_ * mean_pktsize_;
        if (debug_)
                printf("shrink-queue: time %5.2f qlen %d, qlim %d\n",
                        Scheduler::instance().clock(),
                        q_->length(), qlim_);
    while ((!qib_ && q_->length() > qlim_) ||
        (qib_ && q_->byteLength() > qlimBytes))
    {
                Packet *pp = q_->tail();
        q_->remove(pp);
        drop(pp);
    }
}

Packet* VoipPri::deque()
{
        if (summarystats && &Scheduler::instance() != NULL) {
                Queue::updateStats(qib_?q_->byteLength():q_->length());
        }
        return q_->deque();
}

void VoipPri::print_summarystats()
{
                //double now = Scheduler::instance().clock();
        printf("True average queue: %5.3f", true_ave_);
        if (qib_)
                printf(" (in bytes)");
        printf(" time: %5.3f\n", total_time_);
}
```

```
#ifndef ns_voip_priq_h
#define ns_voip_priq_h

#include <string.h>
#include "queue.h"
#include "config.h"

/*
 * A priority based queue for increasing the call capacity of 802.11 access pts
 */
class VoipPri : public Queue {
  public:
        VoipPri() {
                q_ = new PacketQueue;
                pq_ = q_;
                bind_bool("summarystats_", &summarystats);
                bind_bool("queue_in_bytes_", &qib_); // boolean: q in bytes?
                bind("mean_pktsize_", &mean_pktsize_);
        }
        ~VoipPri() {
                delete q_;
        }
  protected:
        void reset();
        int command(int argc, const char*const* argv);
        void enque(Packet*);
        Packet* deque();
        void shrink_queue(); // To shrink queue and drop excessive packets.

        PacketQueue *q_;      /* underlying FIFO queue */
        int summarystats;
        void print_summarystats();
        int qib_;             /* bool: queue measured in bytes */
        int mean_pktsize_;    /* configured mean packet size in bytes */
};

#endif
```

B.2 Voice Over Internet Protocol Agent

```
/*
 *      VoIP Agent, built from a modified ns-2.30 UDP agent, written to
 *      differentiate VoIP traffic from regular UDP traffic
 *
 *
 *
 */



#include "voip.h"
#include "rtp.h"
#include "random.h"
#include "address.h"
#include "ip.h"



static class VoIPAgentClass : public TclClass {
public:
        VoIPAgentClass() : TclClass("Agent/VoIP") {}
        TclObject* create(int, const char*const*) {
                return (new VoIPAgent());
        }
} class_VoIP_agent;

VoIPAgent::VoIPAgent() : Agent(PT_VOIP), seqno_(-1)
{
        bind("packetSize_", &size_);
}

VoIPAgent::VoIPAgent(packet_t type) : Agent(type)
{
        bind("packetSize_", &size_);
}

// put in timestamp and sequence number, even though VoIP doesn't usually
// have one.
void VoIPAgent::sendmsg(int nbytes, AppData* data, const char* flags)
{
```

```
Packet *p;
int n;

assert (size_ > 0);

n = nbytes / size_;

if (nbytes == -1) {
        printf("Error: sendmsg() for VoIP should not be -1\n");
        return;
}

// If they are sending data, then it must fit within a single packet.
if (data && nbytes > size_) {
        printf("Error: data greater than maximum VoIP packet size\n");
        return;
}

double local_time = Scheduler::instance().clock();
while (n-- > 0) {
        p = allocpkt();
        hdr_cmn::access(p)->size() = size_;
        hdr_rtp* rh = hdr_rtp::access(p);
        rh->flags() = 0;
        rh->seqno() = ++seqno_;
        hdr_cmn::access(p)->timestamp() =
            (u_int32_t)(SAMPLERATE*local_time);

        // Send all voip packets to broadcast address to avoid 802.11 ack
        hdr_ip* iph = hdr_ip::access(p);
        iph->dst_ = IP_BROADCAST;

        if (flags && (0 ==strcmp(flags, "NEW_BURST")))
                rh->flags() |= RTP_M;
        p->setdata(data);
        target_->recv(p);
}
n = nbytes % size_;
if (n > 0) {
        p = allocpkt();
```

```
                hdr_cmn::access(p)->size() = n;
                hdr_rtp* rh = hdr_rtp::access(p);
                rh->flags() = 0;
                rh->seqno() = ++seqno_;
                hdr_cmn::access(p)->timestamp() =
                    (u_int32_t)(SAMPLERATE*local_time);
                // add "beginning of talkspurt" labels (tcl/ex/test-rcvr.tcl)
                if (flags && (0 == strcmp(flags, "NEW_BURST")))
                        rh->flags() |= RTP_M;
                p->setdata(data);
                target_->recv(p);
        }
        idle();
}
void VoIPAgent::recv(Packet* pkt, Handler*)
{
        if (app_ ) {
                // If an application is attached, pass the data to the app
                hdr_cmn* h = hdr_cmn::access(pkt);
                app_->process_data(h->size(), pkt->userdata());
        } else if (pkt->userdata() && pkt->userdata()->type() == PACKET_DATA) {
                // otherwise if it's just PacketData, pass it to Tcl
                //
                // Note that a Tcl procedure Agent/VoIP recv {from data}
                // needs to be defined. For example,
                //
                // Agent/VoIP instproc recv {from data} {puts data}

                PacketData* data = (PacketData*)pkt->userdata();

                hdr_ip* iph = hdr_ip::access(pkt);
                Tcl& tcl = Tcl::instance();
                tcl.evalf("%s process_data %d {%s}", name(),
                        iph->src_.addr_ >> Address::instance().NodeShift_[1],
                        data->data());
        }
        Packet::free(pkt);
}
```

```
int VoIPAgent::command(int argc, const char*const* argv)
{                                                                    120
        if (argc == 4) {
                if (strcmp(argv[1], "send") == 0) {
                        PacketData* data = new PacketData(1 + strlen(argv[3]));
                        strcpy((char*)data->data(), argv[3]);
                        sendmsg(atoi(argv[2]), data);
                        return (TCL_OK);
                }
        } else if (argc == 5) {
                if (strcmp(argv[1], "sendmsg") == 0) {
                        PacketData* data = new PacketData(1 + strlen(argv[3]));   130
                        strcpy((char*)data->data(), argv[3]);
                        sendmsg(atoi(argv[2]), data, argv[4]);
                        return (TCL_OK);
                }
        }
        return (Agent::command(argc, argv));
}
```

```
#ifndef ns_voipagent_h
#define ns_voipagent_h

#include "agent.h"
#include "trafgen.h"
#include "packet.h"

#define SAMPLERATE 8000
#define RTP_M 0x0080 // marker for significant events

class VoIPAgent : public Agent {
public:
        VoIPAgent();
        VoIPAgent(packet_t);
        virtual void sendmsg(int nbytes, const char *flags = 0)
        {
                sendmsg(nbytes, NULL, flags);
        }
        virtual void sendmsg(int nbytes, AppData* data, const char *flags = 0);
        virtual void recv(Packet* pkt, Handler*);
        virtual int command(int argc, const char*const* argv);
protected:
        int seqno_;
};

#endif
```

Appendix C

Networking Simulation Scripts

C.1 Initial Simulation Script

```
# File:         initial.tcl
# Author:       Benjamin Gray, except small sections taken from wired-and-wireless
# Purpose: Simulation of wired-cum-wireless topology to determine
#                       802.11 access point capacities
#
# ======================================================
# Define options
# ======================================================

set opt(chan)  Channel/WirelessChannel    ;# channel type                    10
set opt(prop)  Propagation/TwoRayGround ;# radio-propagation model
set opt(netif) Phy/WirelessPhy            ;# network interface type
set opt(mac)   Mac/802_11                 ;# MAC type
set opt(ifq)   Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)    LL                         ;# link layer type
set opt(ant)   Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50                ;# max packet in ifq
set opt(nn)             200               ;# number of mobilenodes
set opt(adhocRouting) DSDV                ;# routing protocol
                                                                           20
set opt(cp)    ""                         ;# cp file not used
set opt(sc)    ""                         ;# node movement file.

set opt(x)     1000                       ;# x coordinate of topology
set opt(y)     1000                       ;# y coordinate of topology
set opt(seed)  0.0                        ;# random seed
set opt(stop)  1000                       ;# time to stop simulation

set opt(ftp1-start)     10.0
set opt(voip-start)             20.0                                       30

set num_wired_nodes  1
set pi 3.141592                           ;# Used to place mobile nodes in a circle
                                                    ;# surrounding the access point

# ======================================================
#                         MAC Parameters
#             NB: These were modified for different access points
```

```
# ================================================
                                                                    40
Mac/802_11 set SlotTime_      0.000020      ;# 20us
Mac/802_11 set SIFS_          0.000010      ;# 10us
Mac/802_11 set PreambleLength_ 144          ;# Long preamble
Mac/802_11 set PLCPHeaderLength_ 48         ;# 48 bits
Mac/802_11 set PLCPDataRate_ 1.0e6          ;# 1Mbps
Mac/802_11 set dataRate_      54.0e6        ;# 22Mbps
Mac/802_11 set basicRate_     1.0e6         ;# 1Mbps
Mac/802_11 set RTSThreshold_ 512            ;# No RTS for VoIP

# ================================================                  50

# check for boundary parameters and random seed
if { $opt(x) == 0 || $opt(y) == 0 } {
        puts "No X-Y boundary values given for wireless topology\n"
}
if {$opt(seed) > 0} {
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)
}
                                                                    60
# create simulator instance
set ns_  [new Simulator]

# ================================================
#                        Colour Definitions (For coloured traces)
# ================================================
# define color index
$ns_ color 0 blue
$ns_ color 1 red
$ns_ color 5 chocolate                                              70
$ns_ color 3 brown
$ns_ color 4 tan
$ns_ color 2 gold
$ns_ color 6 black

# ================================================
# Hierachial Routing (ns-2.29 doesn't directly support access points)
# ================================================
```

```
$ns_ node-config -addressType hierarchical

AddrParams set domain_num_ 3    ;# number of domains
lappend cluster_num 2 1 1       ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 2 1     ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain


# ==================================================
#                  Set up simulation traces
# ==================================================

set tracefd [open simout.tr w]
set namtrace [open simout.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo  [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
#   2 for HA and FA
create-god [expr $opt(nn) + 2]

# ==================================================
#                  Creating wired nodes
# ==================================================

set temp {0.0.0 0.1.0}          ;# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
}


# ==================================================
```

```
#                    Configure for Access Points
# ================================================== 120

$ns_ node-config -mobileIP ON \
                 -adhocRouting $opt(adhocRouting) \
                 -llType $opt(ll) \
                 -macType $opt(mac) \
                 -ifqType $opt(ifq) \
                 -ifqLen $opt(ifqlen) \
                 -antType $opt(ant) \
                 -propType $opt(prop) \
                 -phyType $opt(netif) \ 130
                 -channelType $opt(chan) \
                          -topoInstance $topo \
                 -wiredRouting ON \
                          -agentTrace ON \
                 -routerTrace OFF \
                 -macTrace OFF

# ==================================================
#                                Create Home Agent and Foreign Agent
# ================================================== 140

set HA [$ns_ node 1.0.0]
set FA [$ns_ node 2.0.0]
$HA random-motion 0
$FA random-motion 0

# Position (fixed) for base-station nodes (HA & FA).
$HA set X_ 1.000000000000
$HA set Y_ 2.000000000000
$HA set Z_ 0.000000000000 150

$FA set X_ 650.000000000000
$FA set Y_ 600.000000000000
$FA set Z_ 0.000000000000

# ==================================================
#                    Create mobile nodes & associate with access points
# ==================================================
```

```
$ns_ node-config -wiredRouting OFF

for { set i 0 } { $i < [expr $opt(nn) / 2] } { incr i } {
        set MH($i) [$ns_ node 1.0.[expr $i + 1]]
        set node_($i) $MH($i)
        # Pretty traces
        $node_($i) color "1"

        set HAaddress [AddrParams addr2id [$HA node-addr]]
        [$MH($i) set regagent_] set home_agent_ $HAaddress

        # Arrange the mobile nodes in circles around their parent AP
        $MH($i) set Z_ 0.00
        $MH($i) set Y_ [expr 100 + 50 * sin($i * 4 * $pi / $opt(nn))]
        $MH($i) set X_ [expr 100 + 50 * cos($i * 4 * $pi / $opt(nn))]
}

for { set i [expr $opt(nn) / 2] } { $i < $opt(nn) } { incr i } {
        set MH($i) [$ns_ node 2.0.[expr $i + 1]]
        set node_($i) $MH($i)
        set FAaddress [AddrParams addr2id [$FA node-addr]]
        [$MH($i) set regagent_] set home_agent_ $FAaddress

        $MH($i) set Z_ 0.00
        $MH($i) set Y_ [expr 600 + 50 * sin($i * 4 * $pi / $opt(nn))]
        $MH($i) set X_ [expr 650 + 50 * cos($i * 4 * $pi / $opt(nn))]
}

# create links between wired and BaseStation nodes
#$ns_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail
$ns_ duplex-link $HA $W(0) 1.5Mb 20ms DropTail
$ns_ duplex-link $FA $W(0) 1.5Mb 20ms DropTail

$ns_ duplex-link-op $W(0) $HA queuePos 0.5
$ns_ duplex-link-op $FA $W(0) queuePos 0.5

# Setup TCP connections between a wired node and the MobileHost
# if we are simulating with TCP
```

```
#set tcp1 [new Agent/TCP]
#$tcp1 set class_ 1
#set sink1 [new Agent/TCPSink]
#$ns_ attach-agent $MH(12) $tcp1
#$ns_ attach-agent $MH(0) $sink1
#$ns_ connect $tcp1 $sink1
#set ftp1 [new Application/FTP]
#$ftp1 attach-agent $tcp1
#$ns_ at $opt(ftp1-start) "$ftp1 start"

# Set up voip sessions, increasing by 1 every 10 seconds
for {set i 0} {$i < [expr $opt(nn) / 10]} {incr i} {
        set voipsrc1($i) [new Agent/UDP]
        set voipsnk1($i) [new Agent/LossMonitor]
        $ns_ attach-agent $MH($i) $voipsrc1($i)
        $ns_ attach-agent $MH([expr $i + $opt(nn) / 10]) $voipsnk1($i)
        $ns_ connect $voipsrc1($i) $voipsnk1($i)

        set voipsrc2($i) [new Agent/UDP]
        set voipsnk2($i) [new Agent/LossMonitor]
        $ns_ attach-agent $MH([expr $i + $opt(nn) / 10]) $voipsrc2($i)
        $ns_ attach-agent $MH($i) $voipsnk2($i)
        $ns_ connect $voipsrc2($i) $voipsnk2($i)

        set cbr1($i) [new Application/Traffic/CBR]
        set cbr2($i) [new Application/Traffic/CBR]

        # colour traffic
        $voipsrc1($i) set class_ 2
        $voipsrc2($i) set class_ 3
        $ns_ color 1 orange
        $ns_ color 2 green
        $ns_ color 3 red

        $cbr1($i) set packetSize_ 60            ;# Codec dependent
        $cbr1($i) set interval_ 0.02            ;# Codec dependant
        $cbr1($i) attach-agent $voipsrc1($i)
        $ns_ at [expr $opt(voip-start) + $i * 1.005250] "$cbr1($i) start"

        $cbr2($i) set packetSize_ 60
```

```
        $cbr2($i) set interval_ 0.02
        $cbr2($i) attach-agent $voipsrc2($i)
        $ns_ at [expr $opt(voip-start) + $i * 1.005250 + 0.005] "$cbr2($i) start"

}

# Define initial node position in nam

for {set i 0} {$i < $opt(nn)} {incr i} {
    # 20 defines the node size in nam
    $ns_ initial_node_pos $node_($i) 20
}

# Tell all nodes when the siulation ends
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).0 "$node_($i) reset";
}
$ns_ at $opt(stop).0 "$HA reset";
$ns_ at $opt(stop).0 "$FA reset";

$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"
proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

# some useful headers for tracefile
puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp \
       $opt(adhocRouting)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."
$ns_ run
```

C.2 Simulation Script with Prioritised Scheduler

```
# File:         withvoip-pq.tcl
# Author:       Benjamin Gray, extensively modified from wired-and-wireless
# Purpose: Simulation of wired-cum-wireless with priority queue to determine
#                       the voice call capacity of 802.11 access point capacities
#
# ======================================================
# Define options
# ======================================================

set opt(chan) Channel/WirelessChannel    ;# channel type                          10
set opt(prop) Propagation/TwoRayGround ;# radio-propagation model
set opt(netif) Phy/WirelessPhy           ;# network interface type
set opt(mac)  Mac/802_11                 ;# MAC type
set opt(ifq)  Queue/DropTail/PriQueue    ;# interface queue type
set opt(ll)   LL                         ;# link layer type
set opt(ant)  Antenna/OmniAntenna        ;# antenna model
set opt(ifqlen)         50               ;# max packet in ifq
set opt(nn)             200              ;# number of mobilenodes
set opt(adhocRouting) DSDV               ;# routing protocol
                                                                                20

set opt(cp)   ""                         ;# cp file not used
set opt(sc)   ""                         ;# node movement file.

set opt(x)    1000                       ;# x coordinate of topology
set opt(y)    1000                       ;# y coordinate of topology
set opt(seed) 0.0                        ;# random seed
set opt(stop) 1000                       ;# time to stop simulation

set opt(ftp1-start)    10.0
set opt(voip-start)            20.0                                             30

set num_wired_nodes 1
set pi 3.141592                          ;# Used to place mobile nodes in a circle
                                                 ;# surrounding the access point

# ======================================================
#                           MAC Parameters
#             NB: These were modified for different access points
```

```
# ===================================================
Mac/802_11 set SlotTime_        0.000020        ;# 20us
Mac/802_11 set SIFS_            0.000010        ;# 10us
Mac/802_11 set PreambleLength_  144             ;# Long preamble
Mac/802_11 set PLCPHeaderLength_ 48             ;# 48 bits
Mac/802_11 set PLCPDataRate_    1.0e6           ;# 1Mbps
Mac/802_11 set dataRate_        54.0e6          ;# 54Mbps
Mac/802_11 set basicRate_       1.0e6           ;# 1Mbps
Mac/802_11 set RTSThreshold_    3000            ;# No RTS for VoIP

# ===================================================

# check for boundary parameters and random seed
if { $opt(x) == 0 || $opt(y) == 0 } {
        puts "No X-Y boundary values given for wireless topology\n"
}
if {$opt(seed) > 0} {
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)
}

# create simulator instance
set ns_   [new Simulator]

# ===================================================
#                      Colour Definitions (For coloured traces)
# ===================================================
# define color index
$ns_ color 0 blue
$ns_ color 1 red
$ns_ color 5 chocolate
$ns_ color 3 brown
$ns_ color 4 tan
$ns_ color 2 gold
$ns_ color 6 black

# ===================================================
# Hierachial Routing (ns-2.29 doesn't directly support access points)
# ===================================================
```

```
$ns_ node−config −addressType hierarchical

AddrParams set domain_num_ 3    ;# number of domains
lappend cluster_num 2 1 1       ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 2 1     ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain


# ======================================================
#                          Set up simulation traces
# ======================================================

set tracefd [open simout.tr w]
set namtrace [open simout.nam w]
$ns_ trace−all $tracefd
$ns_ namtrace−all−wireless $namtrace $opt(x) $opt(y)

# Create topography object
set topo  [new Topography]

# define topology
$topo load_flatgrid $opt(x) $opt(y)

# create God
#   2 for HA and FA
create−god [expr $opt(nn) + 2]

# ======================================================
#                          Creating wired nodes
# ======================================================

set temp {0.0.0 0.1.0}          ;# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
}


# ======================================================
```

```
#                    Configure for Access Points
# ================================================== 120

$ns_ node-config -mobileIP ON \
                -adhocRouting $opt(adhocRouting) \
                -llType $opt(ll) \
                -macType $opt(mac) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(ifqlen) \
                -antType $opt(ant) \
                -propType $opt(prop) \
                -phyType $opt(netif) \ 130
                -channelType $opt(chan) \
                        -topoInstance $topo \
                -wiredRouting ON \
                        -agentTrace ON \
                -routerTrace OFF \
                -macTrace OFF

# ==================================================
#                              Create Home Agent and Foreign Agent
# ================================================== 140

set HA [$ns_ node 1.0.0]
set FA [$ns_ node 2.0.0]
$HA random-motion 0
$FA random-motion 0

# Position (fixed) for base-station nodes (HA & FA).
$HA set X_ 1.000000000000
$HA set Y_ 2.000000000000
$HA set Z_ 0.000000000000 150

$FA set X_ 650.000000000000
$FA set Y_ 600.000000000000
$FA set Z_ 0.000000000000

# ==================================================
#                    Create mobile nodes & associate with access points
# ==================================================
```

```
$ns_ node-config -wiredRouting OFF

for { set i 0 } { $i < [expr $opt(nn) / 2] } { incr i } {
        set MH($i) [$ns_ node 1.0.[expr $i + 1]]
        set node_($i) $MH($i)
        # Pretty traces
        $node_($i) color "1"

        set HAaddress [AddrParams addr2id [$HA node-addr]]
        [$MH($i) set regagent_] set home_agent_ $HAaddress

        # Arrange the mobile nodes in circles around their parent AP
        $MH($i) set Z_ 0.00
        $MH($i) set Y_ [expr 100 + 50 * sin($i * 4 * $pi / $opt(nn))]
        $MH($i) set X_ [expr 100 + 50 * cos($i * 4 * $pi / $opt(nn))]
}

for { set i [expr $opt(nn) / 2] } { $i < $opt(nn) } { incr i } {
        set MH($i) [$ns_ node 2.0.[expr $i + 1]]
        set node_($i) $MH($i)
        set FAaddress [AddrParams addr2id [$FA node-addr]]
        [$MH($i) set regagent_] set home_agent_ $FAaddress

        $MH($i) set Z_ 0.00
        $MH($i) set Y_ [expr 600 + 50 * sin($i * 4 * $pi / $opt(nn))]
        $MH($i) set X_ [expr 650 + 50 * cos($i * 4 * $pi / $opt(nn))]
}

# create links between wired and BaseStation nodes
#$ns_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail
$ns_ duplex-link $HA $W(0) 1.5Mb 20ms VoipPri
$ns_ duplex-link $FA $W(0) 1.5Mb 20ms VoipPri

$ns_ duplex-link-op $W(0) $HA queuePos 0.5
$ns_ duplex-link-op $FA $W(0) queuePos 0.5

# Setup TCP connections between a wired node and the MobileHost
# if we are simulating with TCP
```

```
#set tcp1 [new Agent/TCP]
#$tcp1 set class_ 1
#set sink1 [new Agent/TCPSink]
#$ns_ attach-agent $MH(12) $tcp1
#$ns_ attach-agent $MH(0) $sink1
#$ns_ connect $tcp1 $sink1
#set ftp1 [new Application/FTP]
#$ftp1 attach-agent $tcp1
#$ns_ at $opt(ftp1-start) "$ftp1 start"

# Set up voip sessions, increasing by 1 every 10 seconds
for {set i 0} {$i < [expr $opt(nn) / 10]} {incr i} {
        set voipsrc1($i) [new Agent/VOIP]
        set voipsnk1($i) [new Agent/LossMonitor]
        $ns_ attach-agent $MH($i) $voipsrc1($i)
        $ns_ attach-agent $MH([expr $i + $opt(nn) / 10]) $voipsnk1($i)
        $ns_ connect $voipsrc1($i) $voipsnk1($i)

        set voipsrc2($i) [new Agent/VOIP]
        set voipsnk2($i) [new Agent/LossMonitor]
        $ns_ attach-agent $MH([expr $i + $opt(nn) / 10]) $voipsrc2($i)
        $ns_ attach-agent $MH($i) $voipsnk2($i)
        $ns_ connect $voipsrc2($i) $voipsnk2($i)

        set cbr1($i) [new Application/Traffic/CBR]
        set cbr2($i) [new Application/Traffic/CBR]

        # colour traffic
        $voipsrc1($i) set class_ 2
        $voipsrc2($i) set class_ 3
        $ns_ color 1 orange
        $ns_ color 2 green
        $ns_ color 3 red

        $cbr1($i) set packetSize_ 60            ;# Codec dependent
        $cbr1($i) set interval_ 0.02            ;# Codec dependant
        $cbr1($i) attach-agent $voipsrc1($i)
        $ns_ at [expr $opt(voip-start) + $i * 1.005250] "$cbr1($i) start"

        $cbr2($i) set packetSize_ 60
```

```
        $cbr2($i) set interval_ 0.02
        $cbr2($i) attach-agent $voipsrc2($i)
        $ns_ at [expr $opt(voip-start) + $i * 1.005250 + 0.005] "$cbr2($i) start"

}

# Define initial node position in nam

for {set i 0} {$i < $opt(nn)} {incr i} {
    # 20 defines the node size in nam
    $ns_ initial_node_pos $node_($i) 20
}

# Tell all nodes when the siulation ends
for {set i 0} {$i < $opt(nn) } {incr i} {
    $ns_ at $opt(stop).0 "$node_($i) reset";
}
$ns_ at $opt(stop).0 "$HA reset";
$ns_ at $opt(stop).0 "$FA reset";

$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "stop"
proc stop {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
}

# some useful headers for tracefile
puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp \
       $opt(adhocRouting)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"

puts "Starting Simulation..."
$ns_ run
```