Network Analysis of Nintendo DS Traffic

A Major Qualifying Project Report:
submitted to the faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

Matthew K Brennan

Robert M Shaw III

Nate Gershaneck

Date: March 02, 2006

Approved:

1. networks
2. handheld gaming
3. wireless

Mark Claypool

Abstract

This paper studies the characteristics and behaviors of wireless networks utilized by the Nintendo DS during multiplayer gaming. The nature of DS wireless multiplayer is explored. By setting up a PC to sniff wireless traffic, we were able to perform carefully designed experiments that would allow for a comparative analysis of two, three, and four players. The resulting data was used to discuss scalability and network architecture.

ii List of Illustrations

iii List of Tables

iv Table of Contents

1 Introduction

With the limitations of wired networking, companies and consumers alike are looking for better ways to get connected and stay connected as they move around. Wireless networking has seeped into many different markets. Home owners now opt for wireless routers instead of threading Ethernet cables throughout their homes. Some cities, like Philadelphia, plan to implement city-wide wireless access (Reuters 2005). The networks utilizing the IEEE 802.11 wireless LAN standard have been integrating themselves into society for several years now. The next big step for 802.11 to take is into the gaming world.

Some of the most popular of the current generation of handheld gaming systems are the Nintendo DS and the Sony PSP. Both Nintendo and Sony heavily market the included wireless capabilities. The capability to connect wirelessly makes multiplayer gaming an easier and less burdensome experience than it has been in the past. Previously, multiplayer on handheld systems like the original Gameboy was only achievable via a link cable. The link cable is a wire that can tangibly connect a maximum of two systems. In the new generation of handheld gaming, devices can connect with other like devices (no cross-platform connecting) just by being in range (30-100 ft.); there is no need for networking the devices with a link cable. In addition, more than two players can connect via wireless networking. Finally, there is also the prospect that a player can sit in the middle of a busy area and join games hosted by strangers, since wireless multiplayer games advertise their presence to other systems in the area. This feature allows players to easily find other gamers in the area and start up a game with little to no hassle.

From a player's standpoint, wireless sounds fantastic. Many business opportunities are being created in the gaming industry and consumers are clearly excited about the technologies being used in the DS and PSP. Although there are many questions about the business and social aspects of wireless functionality in the DS and PSP, we were fueled by technical questions.

What protocols did Nintendo employ? We were interested to determine whether the Nintendo DS adhered to the IEEE 802.11 standard. The Nintendo DS might utilize the more widely accepted variants 802.11a, 802.11b, or 802.11g, but it is also possible that the DS could exploit the newer variants such as 802.11b+ or 802.11e. There is also the possibility that Nintendo strayed from the IEEE standards entirely and developed a proprietary network design.

What sort of architecture would we see? A further point of curiosity lied within the architecture of the multiplayer games. The server/client architecture, where one system hosts for the benefit of the client, has been the typical architecture exploited by multiplayer games. However, the ad-hoc aspect of wireless networking might encourage a non-centralized architecture to be adopted by the Nintendo DS.

Does the architecture shift when more devices are communicating? Another topic we considered was whether or not the DS would alternate from one type of architecture to another depending on the number of systems involved. One possible benefit of doing so could be to avoid high bandwidth consumption, if one type of architecture scales better. Another possible benefit would be to exploit the collective power, either processing power, broadcasting power, or even transmission range of the DSes involved by switching to whichever architecture provides the best infrastructure for the given scenario.

How scalable are the networks? One of the biggest concerns in multiplayer game development is scalability. In order to allow many players to play together, the game needs to scale well from 2 players to 4 players to 5+ players. Since the Nintendo DS is one of the first gaming systems to include wireless functionality, it may set a lot of precedents and trends for future systems. Therefore, understanding the scalability of the games is important. Will adding a third player double the data rate, or will we see a more efficient algorithm?

What general network characteristics are shared across games and across player counts? In order to offer any viable conclusions about the general behavior of the Nintendo DS, it is important to understand what traits are common to most of the scenarios we can test. If every game has completely unique network behavior, the only conclusions we can draw would be game specific. Overall, we expect to see characteristics shared between the games, and even across the player counts. Scalability, bandwidth usage, architecture, and network phases are all possible traits that may be shared.

In order to offer answers to these questions, first we must understand what inherent troubles occur with wireless networking. One dilemma that occurs in wireless is turbulence. Turbulence, in terms of a network, refers to the size and distribution of packets over time. In general, the larger the variation is over time for both the size and distribution of packets, the higher the turbulence is within the network. Previous work (Claypool 2005) has analyzed games from several different genres and their network traffic behavior was characterized to show the overall similarities and differences. It was shown that the PSP games had a larger variance in bit-rates compared to the DS offerings while the overall bit-rates for the DS were higher. DS games were also found to have similar network behavior when compared to each other while PSP games “[varied] considerably in bitrate, frame size, frame frequency, and fraction of broadcast traffic.” By revealing the widely varying traffic patterns of games from various genres, the groundwork for which to build knowledge of wireless multiplayer gaming is formed.

The groundwork thus far consists only of our knowledge of two player wireless network traffic. What changes can we see in the network behavior when more players are added? How does bandwidth scale as more players are added? The architecture behind the wireless communications would reveal much in terms of the scalability of network traffic and help develop network infrastructure with the effects of wireless hand-held gaming systems in mind.

This study looks into 802.11 WLAN traffic generated by multiple (2+) Nintendo DSes. Our study was conducted across several different games in order to develop a more generalized sense of DS network behavior and architecture. We gathered data that allowed us to examine the trends and patterns between different games and different player counts. This data also allowed us to define specific phases in the network communications as well as understand the behavior of the specific frames. After examining the data, we were able to develop conclusions that discuss shared characteristics between test cases, scalability, and network architecture. We will discuss models and analyses that will be both interesting on their own but also applicable to future work done in this field of research.

We need to first talk about the background information relevant to our research in section 2. In the background section, we will explain some of the technologies we have worked with, as well as other work done in this field of study. Afterwards, we will outline the methodology we used to gather useful and reliable data to study in section 3. This section includes the test cases included in our study, and also the experimentation procedures. In section 4, we will present our data and give detailed analyses of it. We will then take the observations that we make and state conclusions based on our data in section 5. In the same section, we will finish this paper by detailing several paths that seem viable for future work in this area of study.

2 Background

With the advent of wireless portable hand-held game consoles, the amount of data being transferred through the air around us is increasing. The Nintendo Dual Screen (DS) and Sony Playstation Portable (PSP) both utilize wireless transmissions to communicate with other units that may be within proximity. Previous research into the wireless traffic of home networks and these game consoles reveals certain problems that need to be addressed in the near future in order to better utilize the bandwidth provided by the 802.11 protocol.

Several games from various genres have already had their network traffic behavior characterized (Claypool 2005), revealing their underlying architectures for communication. In most cases, the communications analyzed were limited to two devices at once, ignoring possible situations of three of more players participating in a game. The effect of more than two players on the network was left unexplored, creating a gap in our knowledge of how the wireless bandwidth is utilized when incorporating additional players to a network.

2.1 Nintendo DS

The Nintendo Dual Screen (DS) is capable of communicating with other DSes that are within proximity via a proprietary protocol known as the "Nintendo Low Latency Protocol." (Nintendo of America Inc.) While this protocol is said not to be 802.11b, it is still operating in the same 2.4 GHz bandwidth that 802.11 does, therefore possibly affecting any 802.11 transmissions that might be sent. The DS does not contain a TCP/IP stack but it is still capable of communicating with a wireless router. The typical data rate for transmissions is either 1 Mbps or 2 Mbps.

The DS has the capability of allowing cartridge sharing. Cartridge sharing is a mechanism by which one person who owns a copy of a DS game can share the game with other DS owners who do not have the game themselves. This sharing is achieved by allowing players without the game to download the necessary game data from the person with the game cartridge. This feature allowed us to easily run multiplayer tests with only a single copy of each game.

Currently, Nintendo has plans of allowing DSes to connect to each other not just in an ad-hoc mode, a mode not requiring a base station to communicate between devices, but also through the Internet itself via wireless hotspots and home routers (Bramwell 2003). Since the DS does not have a TCP/IP stack, games must program in their own stack in order to work across the Internet. Most of the early releases for the system did not program in a TCP/IP stack so they are only capable of multiplayer gaming in ad-hoc mode. Nintendo is also releasing a wireless USB adapter that can be plugged into a home computer if the owner does not own a wireless router.

The baseband/media access control processor of the wireless module is a Mitsumi MM3155 (Yomogita 2005). It is accompanied by an RF transceiver IC from RF Micro Devices, an intermediate frequency (IF) surface acoustic wave (SAW) filter and a radio frequency (RF) bandpass filter. It is currently unknown if the module is capable of RTS/CTS (Request to Send / Clear to Send) which would allow for better, uninterrupted transmissions between devices under certain network conditions.

2.2 802.11

802.11 is an IEEE standard governing wireless networks. The operating frequency band of 802.11 communication is either 2.4 GHz or 5 GHz depending on the specific

802.11 standard being used. There are several variations of the 802.11 standard being used at the moment. 802.11a, 802.11b, and 802.11g are all variations that you can find included in market-available products. Because 802.11 traffic utilizes radio waves as its communication medium, there are many interesting problems that arise in wireless networks, such as the ‘anomaly of 802.11’.

2.2.1 Anomaly of 802.11

802.11b is capable of transmitting data up to 11 Mbps but will drop the bit rate down to 5.5, 2, or 1 Mbps if it detects unsuccessful frame transmissions. While this is ideal for the computer, allowing it to keep successfully transmitting with reduced frame loss, it has been shown that the wireless network as a whole will suffer if there is more than one computer connected to a wireless access point (Heusse et al 2003). When one computer begins to communicate with the wireless router at a lower bit-rate than all the other computers, it begins to affect the throughput of all the other computers, forcing them to wait on the slow system to transmit since all hosts have an equal probability of channel access. The reason for this observed anomaly is the use of the CSMA/CA channel access method; CSMA/CA “guarantees an equal long term channel access probability to all hosts.” So when one host locks up the channel with a low bit-rate, the other systems have no choice but to wait for the slow transmissions.

2.2.2 LLC

Logical Link Control, or LLC, is a term used under the IEEE 802 networking standards to denote a portion of the data link layer (Wikipedia 802.11). Considered an upper sub-layer, LLC operates on top of the MAC protocol. LLC is primarily concerned with identifying the specific IP protocol that should be used at the source and destination for processing a network frame. It can also be used to provide flow control, or detection and retransmission of dropped packets, although this functionality is optional.

During the course of analyzing Nintendo DS wireless network traffic, a significant portion of the packets were reported by Ethereal as being LLC. Our observations of the nature of the DS communications led us to consider the possibility, however, that the packets are detected as LLC, but are actually something else. Since the Nintendo DS does not utilize standard 802.11 wireless protocols for much of its traffic, we consider it possible that it is a coincidence that some of the packets appear to be LLC.

2.2.3 SNA

System Network Architecture, or SNA, is a proprietary networking architecture owned by IBM and developed in the 1970s (Wikipedia SNA). Although SNA is still in use in certain specialized areas of business and government networks, one of the primary pieces of networking hardware for use with SNA networks is no longer being produced. In addition, IBM is expected to drop support for SNA sometime in the near future. Under SNA, link control is not managed by network applications themselves, but is instead dealt with by a Network Control Program. This has several effects on the ease of implementing networking capabilities in an application and on the process of debugging networking errors.

In our tests, we found that the Nintendo DS occasionally sends packets which are reported by Ethereal as SNA protocol packets. However, usage of SNA does not seem to logically fit with the rest of the transmissions sent by the DS. These SNA packets are reported as malformed, with very unusual information regarding the sending and receiving devices. Since Nintendo claims to be using a proprietary wireless network protocol, and SNA is a proprietary architecture for which they would have had to pay royalties to IBM, we consider it almost certain that the packets detected as SNA serve a different purpose. It is far more likely that certain packet structures in the Nintendo Low-Latency Protocol share similar headers to SNA packets, and trigger Ethereal to recognize them as such.

2.2.4 Wireless Sniffing

Wireless sniffing presents some unique problems that do not arise when sniffing on wired networks. The techniques used by (Jardosh et al 2005) provide much information on these potential problems that can be encountered when capturing wireless traffic. The first problem is a result of the positioning of the devices and the sniffer. The only data that can be analyzed is the data that the sniffer successfully captures. If the sniffer is too far from one of the devices, a sizable amount of data can potentially be missed in the capture. Therefore, the location of the sniffer cannot be haphazardly chosen, but must be carefully considered before any testing can begin.

The second problem that occurs in wireless sniffing is the issue of uncaptured frames. When we have two or more devices communicating wirelessly, the sniffer has to keep up with the traffic. If the devices being monitored are sending more information than the sniffer's hardware is capable of capturing, then there will be a notable percentage of lost frames. A second contribution to the inability to capture all frames is congestion in wireless networks. With numerous signals inhabiting the same air space, it is possible for signals to interfere with one another, resulting in garbled and unreadable frames.

When sniffing wireless networks, it is important to consider the previously mentioned problems. If a large enough percentage of the overall data is lost due to one reason or another, the final results of the study can be inaccurate and erroneous. In order to test our configuration, we first preformed a capture under conditions of heavy traffic. No packets were reported to be dropped by our system. This result suggested that neither problem asserted itself in a significant way during the course of this study.

3 Methodology

Our primary focus was to measure the behavior of 802.11 wireless network traffic between multiple Nintendo DSes. Within this focus are several different paths of study that we intended to explore. Some of our work echoes the work done by (Claypool 2005), but takes a different approach by involving upwards of three Nintendo DSes. This alteration is a fundamental difference, even though the goals and methodology remained noticeably similar. By exploring scenarios with varying numbers of players, we hoped to better understand the nature of multiplayer gaming with the Nintendo DS. Previous papers did not investigate scalability or network architecture.

3.1 Goals

Although the physical hardware was of interest to us, our attention was centered more on the implementation of networking. We found ourselves intrigued by the topics of scalability, network architecture, bandwidth usage, and network characteristics. The subject of Nintendo DS networking lent itself to a multitude of questions.

First, our goal was to derive a network model of a typical DS multiplayer game. Although previous work (Claypool 2005) talks about common network phases seen in 2 player games, we wanted to explore 3 and 4 player cases to see if the phases remained the same. Robust network models would help us begin to flesh out our understanding of what the DSes were doing in regard to their communication. As we developed these models, we looked to define distinct phases of the game session. (Claypool 2005) showed there were several phases mentioned such as the download phase, play phase, etc. We wanted to verify and annotate these phases for the games we were working with as well as for different numbers of players. We were hoping to see common phenomena between different games and different scenarios.

Next, we wanted to delve deeper into the data and look at the specific communication from each device. Our goal here was to determine what sort of architecture the DS uses for multiplayer gaming. By understanding whether sever/client or peer-to-peer architecture was used, we would be able to make some assumptions about network behavior. We would also be able to make better sense of the system-specific data we gathered.

Our third goal flowed naturally out of looking at the outputs for each device. After understanding the communications between two DSes, we wanted to increase the player count to see what effects it would have on the network. We had two topics of interest in this area of study. The first was to begin to answer the questions we had about how well the DS networks scaled. The second topic of interest was to compare the third DS to the original two. We were curious to see what characteristics were shared between all three systems, as well as the characteristics that we only common to two of the three.

The overall motivation for our project was to attempt to answer each of the questions stated in the introduction (section 1), and also to express a sound understanding of DS wireless networking. We wanted to build up an expertise of not only DS networks but also of 802.11 networks and wireless communication in general.

3.1.1 Early Goals

The original concept for our research was to examine network behavior under varying wireless connection qualities. During the course of our analysis of standard game play captures and host to client communication, our focus shifted towards network behavior under varied quantities of players. Our first step was to perform some of the tests used in 'On the 802.11 Turbulence of Nintendo DS and Sony PSP Hand-held Network Games' and apply them to three Nintendo DSes. These tests included two-player captures of Pictochat and Super Mario 64 DS. While performing these tests, we sought to explore two separate and distinct scenarios: the network traffic when only one device has a poor connection and the network traffic when all devices share an equally poor connection.

A large factor on the way the network traffic performs is how hosting works. We spent time looking into the communication between the host device and the other devices. What happens when the host has a poor connection, while the other two sit literally next to each other? Will the host drop a connection entirely in order to preserve the quality of the game play for the others connected?

Lastly, we wanted to answer several questions about the nature of the 802.11 network that the Nintendo DS uses. Currently, we do not have any information indicating whether the DS uses RTS/CTS as a method of collision avoidance. It is known that the planned Nintendo Gameboy Advance wireless adapter will probably use TDMA (time division multiple access), which uses time division multiplexing in order to support multiple simultaneous channels on the same frequency. We do not know whether or not this technology is implemented on the Nintendo DS.

3.2 Preparation

We used multiple Nintendo DS systems and a selection of games from different categories, with the idea being that different types of game applications might exhibit different network behavior. The test cases included different games and different player counts. All of our testing took place in the Fossil Lab (FL B17) in Fuller Laboratories on the WPI campus. This lab has a coating of a copper-based paint which was intended to isolate the room from outside wireless interference. After testing, we found that the room was not fully isolated, but was still less flooded with wireless traffic than other possible test sites. A nearby computer system inside the lab was equipped with an 802.11 wireless NIC with sniffing capability. Figure 3.2.1 shows the positioning used for most of our testing, where node 1 denotes the sniffing system and nodes A, B, and C are the DSes. Our system was placed in the corner of the lab where there seemed to be the most shielding from wireless traffic. The remainder of the room consisted of PCs without wireless capabilities. Due to their lack of impact on our study,

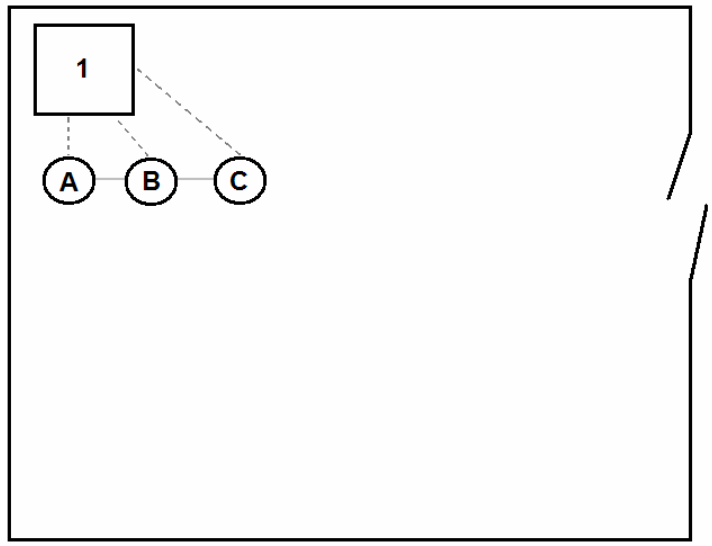


Figure 3.2.1 Wireless network configuration in Fossil Lab

they are not pictured in the diagram. Wireless network activity was monitored and logged by the Ethereal program running on the wireless sniffer for later analysis.

3.2.1 Hardware

The hardware used was a 735 MHz Pentium Celeron processor with 256 MB of RAM. The wireless Ethernet card was manufactured by Linksys and contained an ISL38xx chip that was capable of capturing packets in a promiscuous mode. Although sometimes the system processing seemed delayed, no packets were dropped during testing as a result of buffer overflows. We used four different Nintendo DSes throughout our research.

3.2.2 Software

The operating system used was SuSE 9.2, a Linux distribution known for good driver support and usability. In order to use the wireless card to capture packets, we used the drivers from the Prism54 project to setup the card in promiscuous mode. Ethereal [v 0.10.12] and its text-based version, Tethereal [v 0.10.12], were used to capture packets and save them for later analysis. The games used for testing and analysis were: Pictochat, Super Mario 64 DS, Advance Wars: Dual Strike, and GoldenEye: Rogue Agent. We also developed a piece of software to perform calculations on the raw captured packet information.

We gathered data for four games: Pictochat, GoldenEye: Rogue Agent, Super Mario 64 DS, and Advance Wars: Dual Strike. We chose these games based on the varied game play elements in each. Pictochat, by Nintendo, is the chat program that comes pre-installed on the DS, an application that should show intermittent burst of activity. GoldenEye: Rogue Agent, by Electronic Arts, is a First-Person Shooter, which will usually contain a steady flow of data with brief pauses after a player's death. Super Mario 64 DS, by Nintendo, is an Action-Platformer that experiences constant player movement and brief intermittent interactions between players. Advance Wars: Dual Strike, by Nintendo, is both a Turn-Based and Real-Time Strategy game but only the Real-Time game is available for multiplayer mode; the game play involves constant player movement and actions.

3.3 Experimentation

The foundation for any reliable research is reliable data. In order to generate reproducible data, it is important to develop test cases and procedures that give us controlled and reproducible results. In the case of our research, we needed to construct test cases that would allow us compare behaviors of different games. We also needed to create test cases that would give us the ability to compare scenarios involving varying numbers of players. Finally, in order to make these data sets comparable, we needed a standardized data collection system.

3.3.1 Test Cases

Our focus was primarily on the traffic of different Nintendo DS games with varying numbers of players. For each game, the number of players participating varied from two to four. We also looked at some of the games with periods of no game activity during actual game play mixed with a brief burst of high game activity.

In total we took 28 captures. Each capture spanned a complete game session, from connection to game play to disconnect. The captures, numbers, and conditions are shown in Table 3.3.1.

Game	Player Count	Captures Taken	Special Conditions
Advance Wars: Dual Strike	2	3	
		1	No cartridge during play
	3	1	
	4	3	
Super Mario 64 DS	2	3	
	3	3	
		1	No capture filter
	4	2	
Pictochat	2	1	
		1	Varying connection quality
	3	1	
	4	1	Controlled activity variation
GoldenEye: Rogue Agent	2	1	
	3	2	
		1	No activity during play
		1	Controlled activity variation
	4	2	

Table 3.3.1 Test Cases

Each game has a different amount of captures depending on the viability of the data and special conditions we wanted to test. Several captures had channel switching or in-game phenomena that resulted in misleading data and were therefore excluded from deeper analyses. Advance Wars: Dual Strike and GoldenEye: Rogue Agent both experienced channel switching during many captures. The channel switching only produced a completely unusable capture in one of the Advance Wars: Dual Strike 2 player captures.

In the case of Advance Wars: Dual Strike, we preformed a capture in which the player with the game cartridge shared the game with two other DSes, then did not participate in the actual game. Early in our testing, we did a capture of Super Mario 64 DS without any Ethereal capture filters. The idea behind this test was to check to see if there was any additional useful information that was being filtered out. We concluded that our filtered did not exclude any useful information. Pictochat had two special captures. In “varying connection quality”, one DS was stationary while another moved around the lab in order to create differing levels of connection quality. In “controlled activity variation”, we refrained from sending messages with the exception of specific times. The goal was to verify transmission behavior in Pictochat. GoldenEye: Rogue agent had two complimenting special cases. In the first, we started a game but did not actively play. Each player merely stood motionless at their spawn points. In the second special capture, we echoed the same behavior, except for a minute where we tried to generate as much activity as possible. These captures will be explored in more depth later on.

3.3.2 Procedure

We needed a standardized method for capturing the network activity and converting it to a format the can be easily analyzed and graphed. First, the DS containing the cartridge for the game under analysis was booted up and sent into multiplayer mode. We refer to this DS as the host. This terminology is not meant to imply a host/client network architecture, only that one DS has the game cartridge. After the host began transmitting broadcast packets to announce the game it was hosting, we alternated our sniffer between channels 1, 6, and 11 to find which wireless channel the game was being hosted on. Typically, we found that the channel the broadcast was sent on was the same channel used through most or all of the rest of the game session. Next, we began capturing and logging packets. Then the other DSes started a search for available multiplayer games. These DSes are referred to as clients. As with our usage of ‘host,’ the usage of ‘client’ is not intended to imply network architecture. The clients selected the host’s game and began to download it from the host. The players then did all necessary pre-game setup, ranging from character to map selection. When everyone had completed this setup, the host started the actual game, which was then played through by the players. After game play was done, the clients would disconnect from the host and the capture would be stopped.

Each capture was saved using a standard naming convention consisting of the games name, the date the capture was made, the quality of the link between the DSes and the iteration number. The captures then needed to be run through Tethereal, the command-line program that outputs Ethereal captures to text. Since the resulting files contained every packet captured, they were usually too large to work with when we wanted to analyze the data. We used a parser written in C++ that could quickly perform the necessary calculations on a capture after is was processed by Tethereal. This parser would output the important statistics in a comma-separated values file using a specified time interval value. The information in these files could then be easily examined and graphed using Microsoft Excel. We graphed numerous values for each capture, including breakdowns based on both the entire capture as well as subsections thereof.

4 Data and Analysis

In order to develop an understanding of the behaviors and patterns of the wireless networks generated by the Nintendo DS, we need to interpret the large amounts of data we gathered throughout our research. Through various analyses of our data, we can decipher many common patterns and traits across different scenarios.

In order to better understand our data, there were several different analyses we were looking to use. The first was to observe the characteristics of the network traffic and communication for entire captures. Such an examination would give us insight into what phases are common to all DS games. We also wanted to compare behavior during play for each game and determine what likenesses were shared.

Since our focus is on the changes in the network behavior as we add or subtract players from the game, we needed to look at communications from each specific device. In addition to looking at the overall bandwidth, we broke down the traffic based on the source and the destination. By performing this analysis, we hoped to gain insight into the network architecture and scalability of the DS networks. When breaking down the bandwidth, we also wanted to look at the characteristics of the specific frames.

4.1 Data flows

During the course of our study of the wireless behavior of the Nintendo DS, we came to understand that the various interactions between different machines could be divided into several categories. The communications between DSes do not conform to conventional wireless transmission characteristics. The DSes utilize MAC addresses that are not specific to the device hardware for any of the systems under examination. We theorize that these addresses correspond to virtual MACs that are not dropped by the DS hardware. Although there are a small number of packets that do not use these virtual addresses, they make up an insignificant portion of the data. The remaining packets make possible all the communication necessary for gameplay. Figure 4.1.1 is an excerpt of a wireless packet capture and demonstrates the various types of packets that make up what we came to term as *DS data flows*. Our captures were formatted to display the sequence number first, then the time index, the source and destination addresses, the protocol, and finally the frame size.

```
11702  86.972644 00:09:bf:71:9e:07 -> 03:09:bf:00:00:00 LLC 540
11703  86.972861 00:09:bf:66:b7:4b -> 03:09:bf:00:00:10 IP 38
11704  86.973224 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IP 38
11705  86.979577 00:09:bf:71:9e:07 -> 03:09:bf:00:00:00 LLC 540
11706  86.979747 00:09:bf:66:b7:4b -> 03:09:bf:00:00:10 IP 38
11707  86.980159 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IP 38
11708  86.986160 00:09:bf:66:b7:4b -> 03:09:bf:00:00:10 IP 38
11709  86.986532 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IP 38
11710  86.986863 00:09:bf:71:9e:07 -> 03:09:bf:00:00:03 LLC 28
11711  86.992631 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IP 38
```

Figure 4.1.1 Sample Packet Capture

First, there were what appear to be standard IEEE 802.11 broadcast packets, an example of which is given in Figure 4.1.2. Sent to all systems on the wireless channel, these packets do not seem to differ from those sent from a conventional 802.11 device. These packets were only observed emanating from the DS hosting a multiplayer game; never from one of the other DSes that joined such a game.

```
6   0.000203 00:09:bf:71:9e:07 -> Broadcast    IEEE 802.11 188
```

Figure 4.1.2 Broadcast Packet

Next, there was what we termed the *host data flow*. A packet on this flow is shown in Figure 4.1.3. The DS that initiated a multiplayer game was the only device that ever sent packets on this flow. This flow was characterized by being sent to the MAC address 03:09:bf:00:00:00 and was classified as either a generic IP packet or as LLC by

Ethereal. Our belief is that this functions as a DS-specific broadcast protocol, since only one packet of this type is sent, rather than one to each client. The MAC address is also one of the virtual MAC addresses discussed earlier.

```
27541 138.250253 00:09:bf:71:9e:07 -> 03:09:bf:00:00:00 LLC 172
```

Figure 4.1.3 Host Data Packet

The game initiator was also solely responsible for a second flow, this one being what we called the *host feedback*. Figure 4.1.4 shows a typical packet of this sort. Similar in several ways to the host data flow, this flow was also sent to a MAC for which there was no DS, in this case 03:09:bf:00:00:03. The protocol detected for this packet was LLC, but as discussed earlier we do not think this is an accurate descriptor for these packets. Our hypothesis for this flow is that it is used for responses to transmissions from clients, in effect acting as no more than a system for sending acknowledgements.

```
27565 138.451869 00:09:bf:71:9e:07 -> 03:09:bf:00:00:03 LLC 28
```

Figure 4.1.4 Host Feedback Packet

The clients only were directly detected as being responsible for one type of flow, several examples of which can be seen in Figure 4.1.5. We called this the *client data flow*. Each DS acting as a client would be responsible for sending a flow of this type, while the host never sends this type of packet. These packets were among those detected as malformed SNA packets, as mentioned in section 2, but they are also sometimes classified as IP, IPX or LLC. They are sent from one of the clients to the (again nonexistent) MAC 03:09:bf:00:00:10. This MAC address was always the destination used by this flow, regardless of the game or hosting DS. We determined that this flow utilizes the same quasi-broadcasting characteristics as the flows sent by the host.

```
27549 138.317573 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 LLC 30
27550 138.333348 00:09:bf:66:b7:4b -> 03:09:bf:00:00:10 IPX 60
27551 138.333704 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IPX 60
27552 138.350603 00:09:bf:66:b7:4b -> 03:09:bf:00:00:10 LLC 30
27553 138.350968 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 LLC 30
27554 138.367416 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 IPX 60
27555 138.384472 00:09:bf:3b:97:84 -> 03:09:bf:00:00:10 LLC 30
```

Figure 4.1.5 Client Data Packets

The last type of flow may not be a flow at all. A packet of this sort is shown in Figure 4.1.6, although the exact sources and destinations in these packets are not always the same. Certain packets were detected, again as malformed SNA frames, but had completely incomprehensible source and destination information. For example, the source and destination might both be read as '0000,' which is neither an actual device nor even a valid form of expressing a MAC address. The actual MAC addresses involved appeared in a different place within the packet. Since they were malformed for the detected protocol type, this output was difficult to interpret. We do not know whether these packets make up a flow all their own, or whether they are actually parts of other flows and are not detected as such because of how they are sent. In fact, due to the extremely erratic network behavior of these sorts of packets, we have no idea what purpose they serve. We do know, although, that these packets are apart of the DS traffic. Ethereal did not filter out these packets even though we configured Ethereal to filter out any packets that did not emanate from any of the MAC addresses specific to our DSes. When we looked at the information included in these packets, there was still a reference to the address of the DS that sent it. Furthermore, the occurrence of SNA packets in our captures seemed reliable and constant. This behavior further implies that the SNA traffic is correlated to the DSes.

```
16185   95.588905          0100 -> 0200          SNA 34
```

Figure 4.1.6 Malformed SNA Packet

4.2 Phases

Our bandwidth analyses made clear that usage of the wireless network by the DS is not uniform throughout the entirety of a gaming session. Intuitively we would assume that there would be points of high activity and points of low or no activity. Previous research had divided gaming sessions into a series of distinct phases, and we adopted a similar scheme for our own data presentation.

Our data depicted four discrete phases of network activity present in most of the games. There was an *initial phase* of broadcast packets in which the host is announcing the game that they have available. Once a client chooses to download the game, the host then begins to send the game data to the client. The *download phase* always has the highest bandwidth for each of the games. After the download phase is the *setup phase*, where the host and players choose from the options available. This phase generally exhibited extremely low bandwidth usage. The final phase is the *play phase*, in which the host and clients play out the game. This phase usually has a fairly constant bandwidth usage with occasionally large spikes of activity on some games. In the graphs of our captures included below, it should be noted that the capture is cut off at 180 seconds. The data not included consists only of additional play phase information that demonstrates minimal variation from that which is shown. The figures shown in this section clearly demonstrate the breakdown of the captures into these phases.

The first game examined was Super Mario 64 DS with two players (Figure 4.2.1). The beginning of the graph shows no activity; this is before the initial phase begins at time 5 seconds.

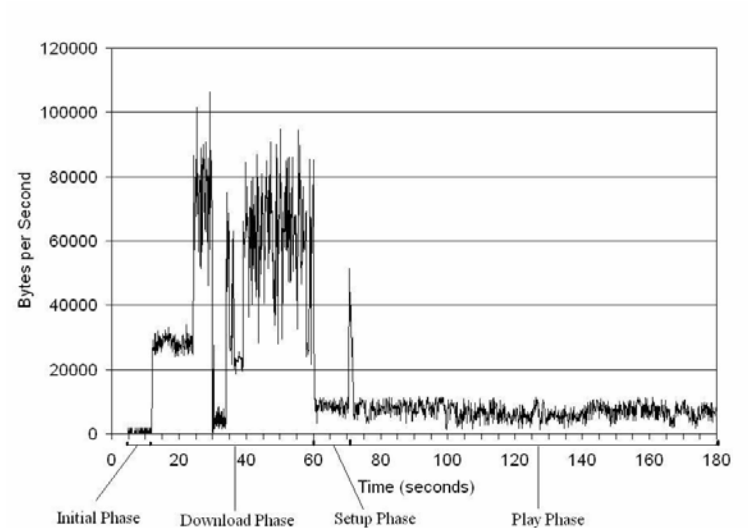


Figure 4.2.1 Overall Bandwidth: Super Mario 64 DS – 2 Player

Small bandwidth use is then seen before a large jump in activity when the clients connect to download the game from the host. This is the start of the download phase, at time 12 seconds. Super Mario 64 DS appears to exhibit two halves to its download phase. Initial data is sent to the client, probably enough to let it show the waiting screen with the players' characters. During this waiting screen, more data is downloaded to the clients, which is most likely the actual game data and maps. The overall average bandwidth for the download phase in this case is around 43834 bytes/sec. Once all the data has been sent, the bandwidth drops back down to around 9798 bytes/sec while the setup phase takes place. This phase shift can be observed at time 60 seconds. Once the play phase begins at time 72 seconds, the bandwidth then begins to varying from 1200 bytes/sec to 11660 bytes/sec. There was a sharp jump in bandwidth at the start of a game which actually helped to delineate between the setup phase and the play phase.

GoldenEye: Rogue Agent is the next game that we investigate (Figure 4.2.2). This figure shows the initial phase at the start followed by the download phase. After the download phase there is no activity from 30 seconds to 65 seconds and then the play phase starts. The missing data encompasses both the second half of the download phase and the entire setup phase. During this period of supposed inactivity, the client screens indicate that they are downloading game data from the host.

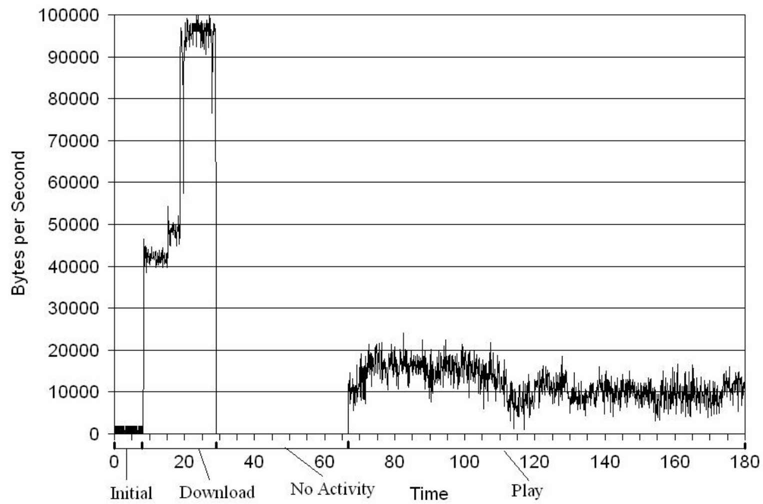


Figure 4.2.2 Overall Bandwidth: GoldenEye: Rogue Agent – 3 Player

If we look at another GoldenEye: Rogue Agent capture (Figure 4.2.3), there does appear to be activity during this time period. In Figure 4.2.2, we believe this data is present but is being sent on a different channel than the one we were monitoring. We noticed occasionally that DS games would switch channels between phases and it seems that in this case they switched channels in the middle of the download phase.

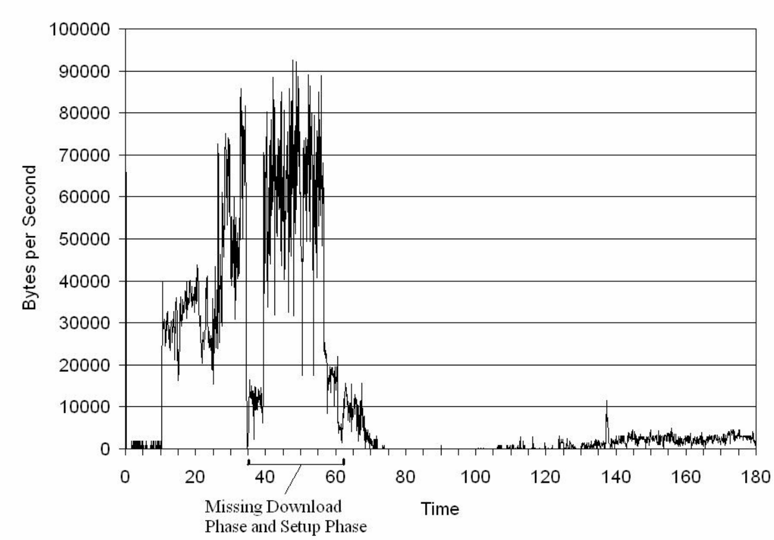


Figure 4.2.3 Overall Bandwidth: GoldenEye: Rogue Agent – 3 Player

Advance War: Dual Strike proved to be the most difficult of all the games to capture the game phases. The game would typically change channels after downloading or starting a game. We did manage to take one capture where the game stayed transmitting on the same channel (Figure 4.2.4). In this capture, the initial broadcast phase followed by the download phase is clearly visible. The setup phase correlates to the period of no network activity from 37 seconds to 74 seconds is when the players were choosing their characters and units for play. At 74 seconds, the units each player had chosen were transmitted to the other players and the game was initialized. Actual play started when the bandwidth jumped up to around 9000 bytes/sec.

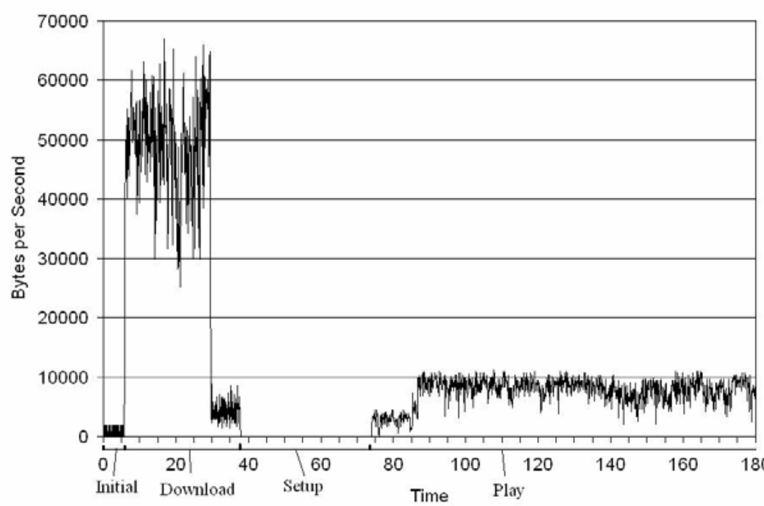


Figure 4.2.4 Overall Bandwidth: Advance War: Dual Strike – 2 Player

Pictochat was the only differing factor in terms of overall phases as there was no need for the clients to download it before connecting to a chat session. The only real phases for Pictochat were the broadcast phase and the play phase, as shown in Figure 4.2.5. The play phase for Pictochat clearly showed the points where chat messages were being transmitted between the DSes. These points are discernable due to the bandwidth spikes shown. Each spike in overall bandwidth usage correlates to one of the players transmitting a new message to the chat room.

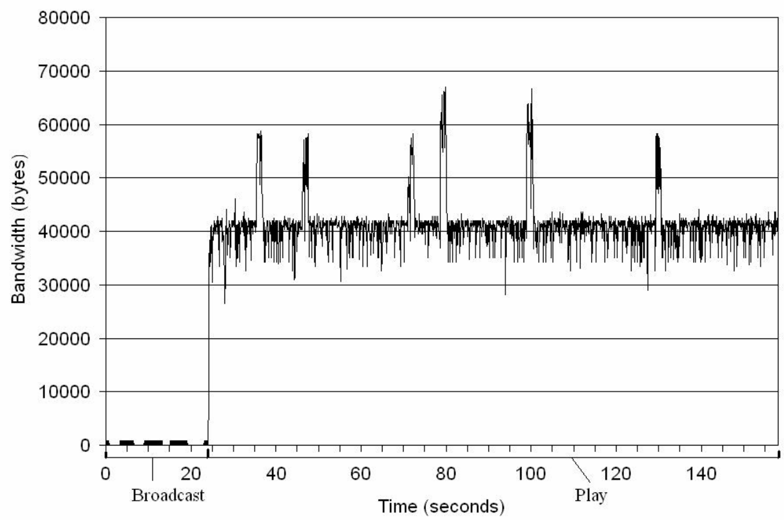


Figure 4.2.5 Overall Bandwidth: Pictochat – 2 Player

The games exhibited some characteristically similar behaviors. The download phase for Super Mario 64 DS, GoldenEye: Rogue Agent, and Advance War: Dual Strike each showed a significant use of bandwidth compared to the rest of the capture. The setup phases for each game differed, with Advance War using almost no bandwidth while Mario used up a continual amount before the game was started. GoldenEye: Rogue Agent's setup phase is non-existent and the play phase started almost immediately after download has been completed. Play phases for each game were fairly similar, with fairly constant bit rates for the duration of the phase. Table 4.2.1 shows the average bandwidth for the phases of each game.

Game	Initial Phase	Download Phase	Setup Phase	Play Phase
Advance War: Dual Strike	791 bytes/sec	37483 bytes/sec	788 bytes/sec	8118 bytes/sec
GoldenEye: Rogue Agent	848 bytes/sec	56106 bytes/sec	None	11870 bytes/sec
Pictochat	227 bytes/sec	None	None	41236 bytes/sec
Super Mario 64 DS	366 bytes/sec	43834 bytes/sec	9798 bytes/sec	6909 bytes/sec

Table 4.2.1 Average Bandwidth for Each Phase – All Games

4.3 Time Slices

Now that we have discussed the various phases involved in a typical start-to-finish graph of a Nintendo DS game, we are going to concentrate on only the play phase. We are most interested in the play phase due to that fact that most of any game session consists of the play phase. The play phase also will be easiest to compare across games, since the download phase will be heavily dependant on how much game information needs to be downloaded to the other systems. The setup phase also is not interesting due to the fact that there is typically low bandwidth usage as players configure the game. Therefore, this entire section will only consider 60 second time slices of the play phase for our captures. Since we see similar behavior in the phases between most DS games, can we expect to see similar behavior within these phases?

4.3.1 Overall Bandwidth

First we will examine the overall bandwidth usage of Super Mario 64 DS with 3 players (Figure 4.3.1). This graph shows a 60 second subsection of the play phase. The x-axis is the time (in seconds) relative to the beginning of this slice. In the case of this specific time slice, the graph represents data recorded between the third and fourth minute of the capture. The y-axis is the number of bytes being sent by all the DSes.

This graph has a few interesting features. The most notable aspect of the bandwidth usage in this graph is the y-range of the data. The minimum bytes/second on this graph is 1240 bytes/second while the maximum is 13740 bytes/second. The range from the minimum y-value to the maximum y-value was 12500 bytes/second, with an average of 8034 bytes/second. The standard deviation was 2448 bytes/second. Network behavior like this can be attributed to several different issues. The first issue we considered was that the overall bandwidth usage is relative to what is specifically occurring in game. For example, if all the players were in the same area, all interacting, we would expect to see higher overall bandwidth usage as opposed to a scenario where all the players were out of view of each other. The second issue that the variance might be denoting is processor usage. In cases where there is a heavy burden on the device's processor, we may see a lower overall bandwidth.

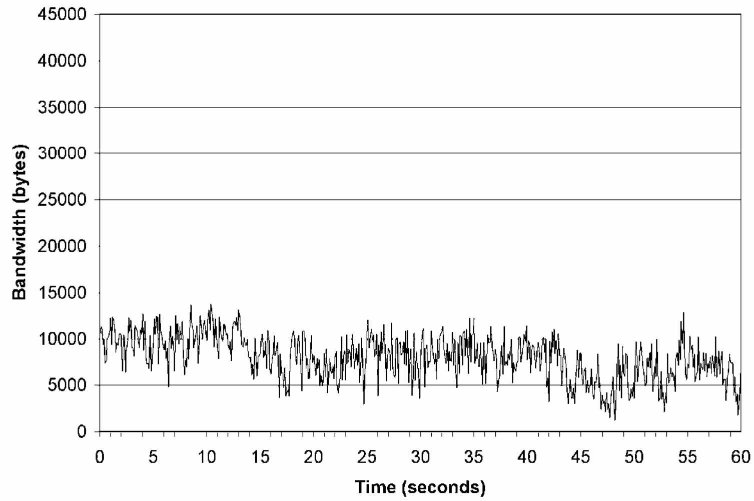


Figure 4.3.1 Overall Bandwidth: Super Mario 64 DS – 3 Players

In the interest of testing the bandwidth variation, we preformed two captures. The first capture consisted of three players playing GoldenEye: Rogue Agent, in which none of the players did anything at all (Figure 4.3.2). After the game started, the players idled at whatever position they spawned at. We referred to this capture as “Boring”. This graph depicts the same variance behavior as Figure 4.3.1.

The second capture was the same in the fact that we had three players playing GoldenEye: Rogue Agent. The difference in the second was that we would idle for the first minute, cause as much activity as we could for a minute, then go back to idling for the rest of the game. We referred to this capture as “Wigging out.”

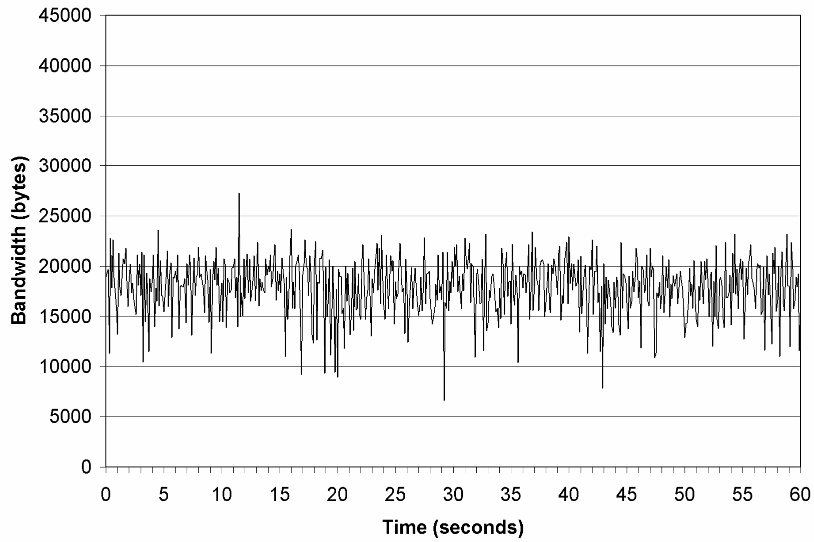


Figure 4.3.2 GoldenEye: Rogue Agent - 3 Player - Boring

Figure 4.3.3 is the section of the play phase in which we cause as much activity as possible. In order to generate activity, we tried to move, shoot, pick up items, and encounter each other as frequently as possible.

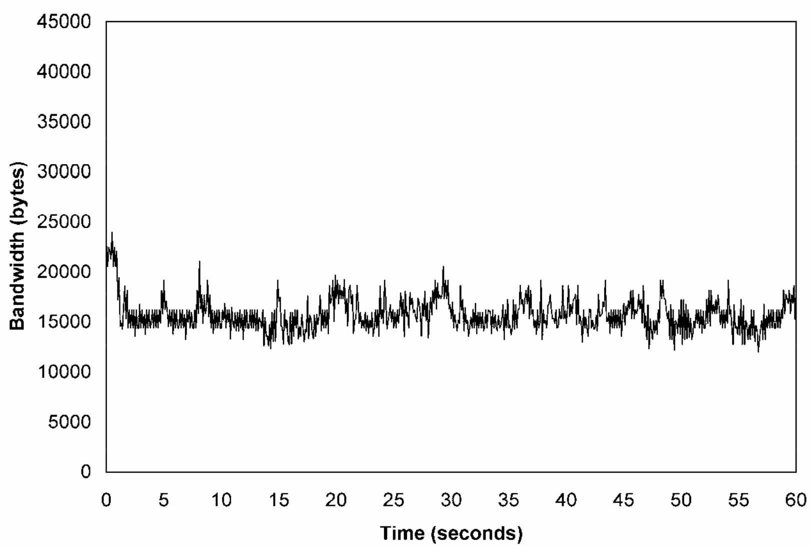


Figure 4.3.3 GoldenEye: Rogue Agent - 3 Player - Wigging Out

We do not see an intuitive change between “Boring” and “Wigging out”. “Wigging out” has both lower average bandwidth usage and also lower variance. “Boring” has an average bandwidth usage of 17916 bytes/second and a standard deviation of 2826. “Wigging out” has an average bandwidth usage of 15716 and a standard deviation of 1748. The hypothesis that bandwidth usage is higher when there is more interaction between the players is clearly not true. The hypothesis that bandwidth usage is lower when there is more strain in the processor seems more likely.

4.3.1.1 Bandwidth Variation across Games

The next step is to look at the overall bandwidth for all our games under similar conditions. Since each of our games involve different types of game-play, it will be interesting to compare their overall bandwidth usage. When originally considering this comparison, we made a few logical hypotheses. The first hypothesis was that we would see widely different echelons of bandwidth usage between the different games. The theory behind this hypothesis was that different genres would have different demands on the network. For example, a first person shooter like GoldenEye: Rogue Agent would need to constantly transmit positions and alignments of players, while a

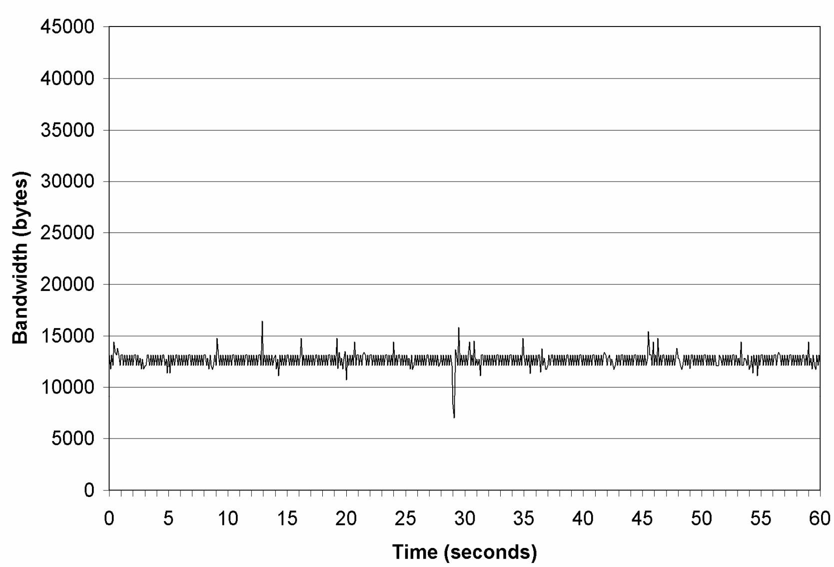


Figure 4.3.4 Advance Wars Dual Strike – 3 Player

chat program like Pictochat would only need to transmit new messages.

Super Mario 64 DS and GoldenEye: Rogue Agent have already been analysed in the previous section. We see some similarities between the two as far as behavior, but GoldenEye: Rogue Agent sends much more data than Super Mario 64 DS. Figure 4.3.4 shows us Advance Wars: Dual Strike. Advance Wars: Dual Strike consists of constant bandwidth usage with very little variance.

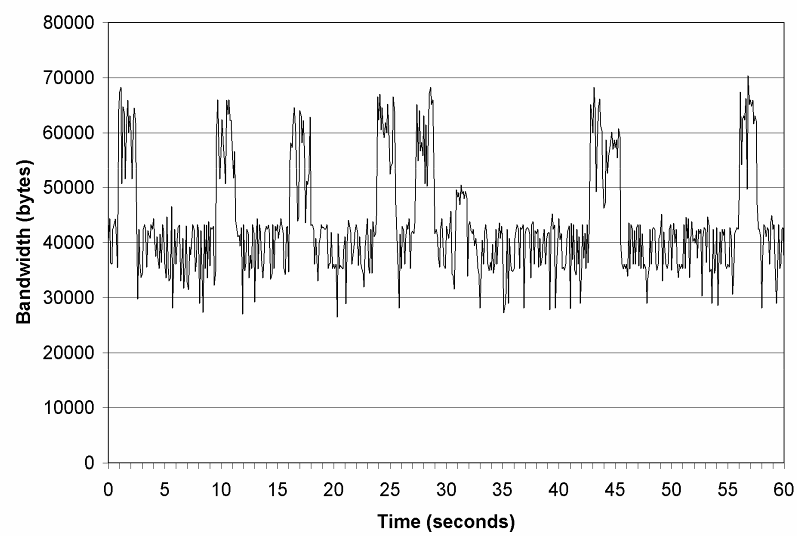


Figure 4.3.5 Pictochat - 3 Player

Such behavior can be a result of either the game-type, or the fact that the game itself would not be processor intensive. Lastly, Figure 4.3.5 illustrates a three player Pictochat session. It is important to note that figure 4.3.5 is on a different y-axis scale than the other three graphs. The reason for this is that Pictochat sends significantly more data than the other three games. Had the graphs been standardized to Pictochat's scale, the non-Pictochat games would have been effectively dwarfed. Pictochat has the most variation of all the games due to the data spikes created when one of the DSes transmits a new drawing.

We notice very little in common between these four graphs. Although each graph has the same player count and the same quality of connection, there is no constant behavior. Each game has a different average bandwidth, as well as a different standard deviation. Each game also has unique spiking and dropping of bandwidth usage. But this variability can simply be a result of the different game types producing different demands on communication.

How about our hypothesis about echelons of bandwidth usage? These graphs do, in fact, depict each game has its own echelon of bandwidth usage, which is high-lighted further in Table 4.3.1. This table reflects the 60 second time slices discussed previously. The trick here is that there echelons are not intuitive to the game type. In games that are dependant on the exact position of the players, we expected to see a high, constant stream of information between the DSes. Looking at the graphs, we do not see that behavior at all. Super Mario 64 DS displays the lowest average bandwidth, while Pictochat displays the highest.

Game	**Average Bandwidth (bytes/person)**	**Standard Deviation (bytes/person)**
Advance Wars: Dual Strike	12658	711
Super Mario 64 DS	8034	2448
Pictochat	43493	9634
GoldenEye: Rogue Agent	17916	2826

Table 4.3.1 Statistical Data of 3 Player Captures

Another point of comparison is the bandwidth variance. Due to its bandwidth spikes, Pictochat ended up with the highest standard deviation. When we examine Super Mario 64 DS and GoldenEye: Rogue Agent, we see that these two games share similar standard deviations even though their average bandwidth is far apart.

4.3.1.2 Scaling Based on Player Count

The next hypothesis we made about overall bandwidth was that we would most likely see a fairly linear scaling as we introduced more DSes. The original thought was that if two devices communicating were talking at 20,000 bytes per second, then the introduction of a third would bring the overall bytes per second up to 30,000.

Table 4.3.2 helps us determine what the scaling behavior is. Once again, we find that the logical hypothesis is only half right. We see linear scaling as more players are added to each game, but the scaling proves to be more efficient than originally anticipated. Advance Wars: Dual Strike serves as a good example. The two player capture averaged at 10480 bytes/second. When we add on a third player, we see an increase of 2178 bytes/second. Adding on a fourth player increases the average overall bandwidth by 2160 bytes/second. Each time we add a new player, we seem to get a very similar increase in the average overall bandwidth. These linear trends can be more easily seen in Figure 4.3.6. If we were looking at unicast communication, such behavior would simply not be possible. Now the question we are forced to ask, and answer, is "What is the behavior of the communication between each specific device?"

Game	Player Count	Average Bandwidth (bytes/second)	Standard Deviation (bytes/person)
Advance Wars: Dual Strike	2	10480	535
	3	12658	711
	4	14817	657
Super Mario 64 DS	2	6599	2107
	3	8034	2448
	4	10318	2272
Pictochat	2	41735	5198
	3	43493	9634
	4	48529	9570
GoldenEye: Rogue Agent	2	10764	2157
	3 Boring	17916	2826
	3 Wigging Out	15716	1748
	4	23312	2945

Table 4.3.2 Statistical Data between Player Counts

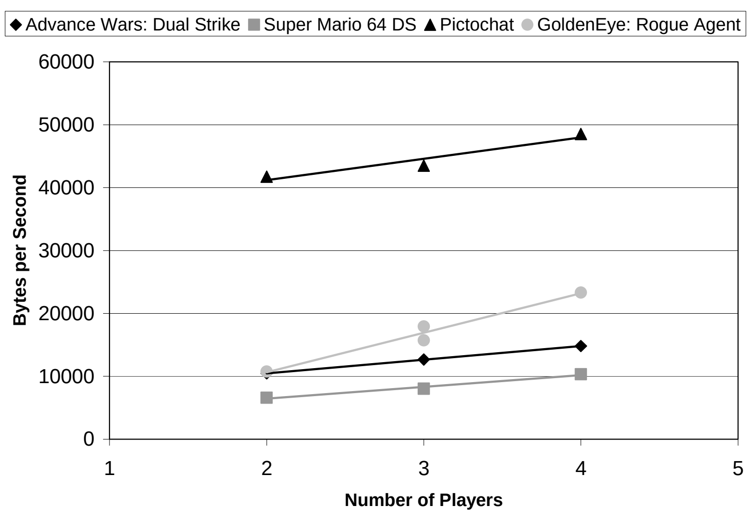


Figure 4.3.6 Linear Trends across Player Counts

4.3.2 Bandwidth Breakdown

Although observing the overall bandwidth gave us a lot of insight into the behavior of the Nintendo DS wireless networks, there is still a deeper level to consider. The next step is to examine what each device is doing.

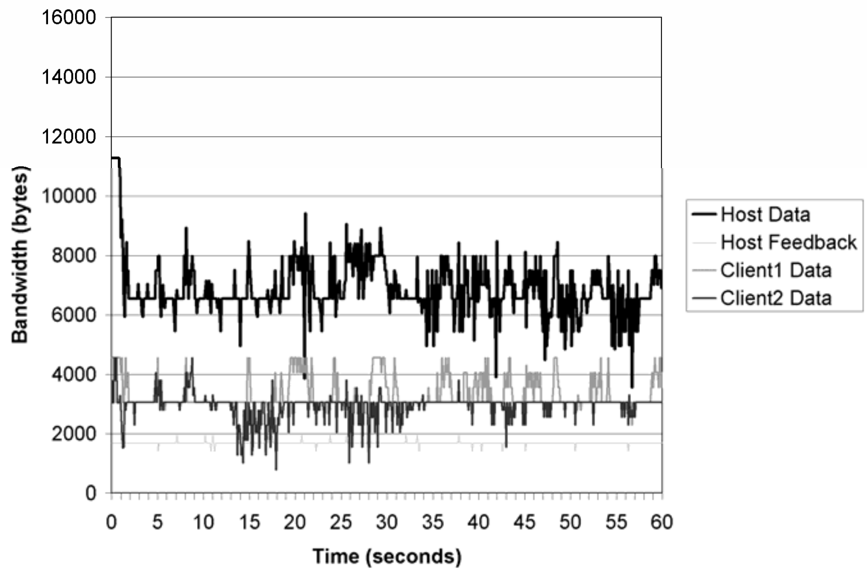


Figure 4.3.7 GoldenEye: Rogue Agent - 3 Player

In Figure 4.3.7, we took the same 60 second time slice of our GoldenEye: Rogue Agent 3 player capture that we used in Figure 4.3.3. On this graph, there are four different types of communication happening. As mentioned in section 4.1, we can see the host data flow, the host feedback flow, and the data flows from each of the two clients. The host data flow constantly sends much more data than any other flow, while the host feedback flow sends much less than the other flow. The two clients seem to both level off at the same intensity, but client 1 seems to have many small bandwidth spikes while client 2 has bandwidth dips.

To better understand this data, Table 4.3.3 displays some statistical data of interest. The host data flow averages higher than both the client data flow's averages combined. If the network architecture was simply ad-hoc, we would not see one DS sending at a higher rate than the other two. We will go further into network architecture in section 5.2. Next, we note that the clients' average bandwidth is separated by 410 bytes/second. Although the clients shared a common steady level of communication around 3100 bytes/second, client 1 seemed slightly more active due to its frequent spikes. The cause of this noticeable difference may be related to the conditions in the game. While playing, there was a much higher rate of interaction between the host and client 2 and client 1 did not encounter the other players as frequently. We can extrapolate that client 2's overall bandwidth is lower due to the larger amount of processing being done due to action occurring in close proximity to the player. Over the entire play phase, the average bandwidth of client 1 was 4248 bytes/second, while the average bandwidth of client 2 was 4109 bytes/second. This data suggests that the two clients operated at similar bandwidths with the exception of the action intensive time slice in question.

Flow	Average Bandwidth (bytes/second)	Standard Deviation (bytes/second)
Host Data	6940	1100
Host Feedback	1679	57
Client 1 Data	3361	530
Client 2 Data	2950	455

Table 4.3.3 Statistical information of Specific Data Flows

4.3.2.1 Flow Variation based on Player Count

The most intriguing aspect of the specific data flows was the behavior of the clients. We saw that the two clients exhibited different qualities within the time slice, but had a very similar average overall. We also observed some interesting scaling behavior when we were looking at the overall bandwidth between player counts. We now have ample reason to compare and contrast the traits of each data flow in 2 player and 4 player scenarios.

Figure 4.3.8 shows the 2 player capture of GoldenEye: Rogue Agent while figure 4.3.9 shows the 4 player capture. When visually comparing these two graphs, the most apparent difference between the two is the activity of the host data flow. In the 2 player capture, the host data flow is only slightly above the client data.

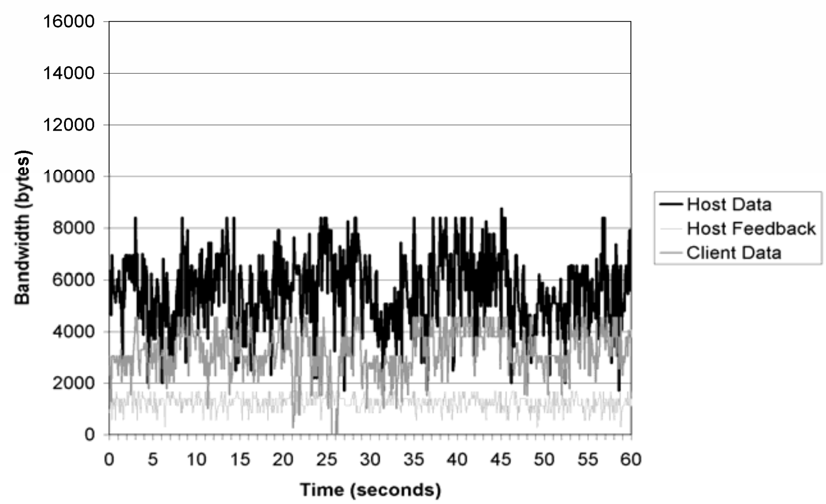
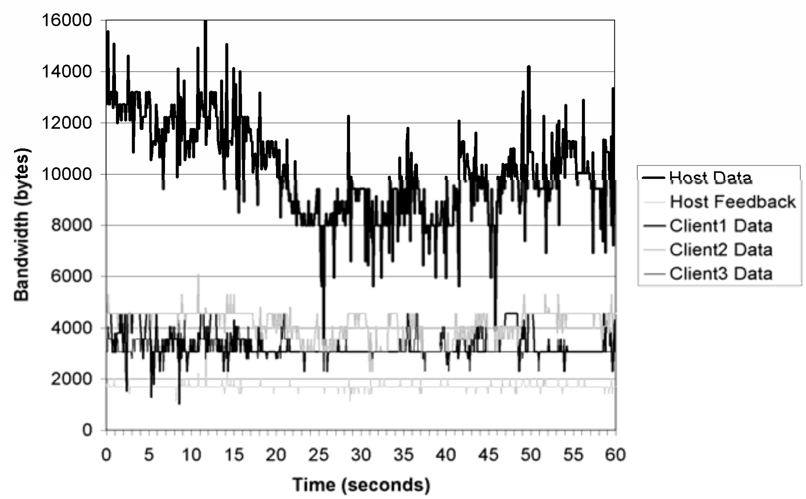


Figure 4.3.8 GoldenEye: Rogue Agent - 2 Player

Meanwhile, in the 4 player capture, the host data is significantly higher than all the client data. There is also more variance all around in the 2 player capture. The typically constant host feedback flow is widely varied in the 2 player capture, relative to



the other captures. The client data flow is also more varied and does not contain any of the constant segments noted in the other two captures. But what about the host data flow? There appears to be less constant noise, but there is a sizable dip about a third of the way into the time slice. Which graph depicts the most variance in the host data flow?

Player Count	Flow	Average Bandwidth (bytes/second)	Standard Deviation (bytes/second)
2	Host Data	5585	1470
	Host Feedback	1245	307
	Client Data	3214	910
3	Host Data	6941	1100
	Host Feedback	1679	57
	Client 1 Data	3361	530
	Client 2 Data	2950	455
4	Host Data	10146	1916
	Host Feedback	1689	140
	Client 1 Data	3331	553
	Client 2 Data	4119	599
	Client 3 Data	3164	300

Table 4.3.4 Statistical Data Regarding Data Flows across Player Counts of GoldenEye: Rogue Agent

Table 4.3.4 includes the average bandwidth and standard deviations of each flow from all three captures. First, we must answer the question we had about the host data variance. Between the three graphs, we see that the host data varies most in the 4 player capture, followed by the 2 player capture and then the 3 player capture. This phenomenon is unexpected because we clearly see that the 2 player capture has much higher variance in the host feedback flow and the client data flow.

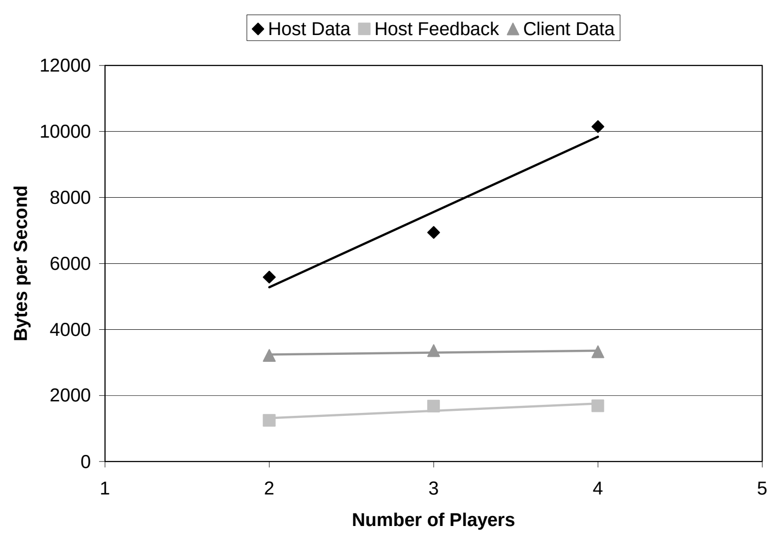


Figure 4.3.10 Linear Trends of Data Flows

It was apparent from the graphs that the host sent more bytes/second depending on the number of players involved, but what about the clients? Even though the clients in the three captures had a range of 1168 bytes/second, we can see that the clients typically send around 3300 bytes/second. This behavior is better visualized in Figure 4.3.10. The standard deviation also remains rather consistent between the three captures. We even see similar behavior in the host feedback flow. It seems that, at least for GoldenEye: Rogue Agent, the only flow that is affected by the player count is the host data flow.

4.3.2.2 Flow Variation across Games

When we were looking at the overall bandwidth graphs, we noted that there was very little constant behavior between the games. Each game used sizably different amounts of bandwidth, and also had very different variance. Will similar differences assert themselves in the specific flows?

Figure 4.3.11 shows a 2 player capture of Advance Wars: Dual Strike. Compare this with figure 4.3.12, a 4 player

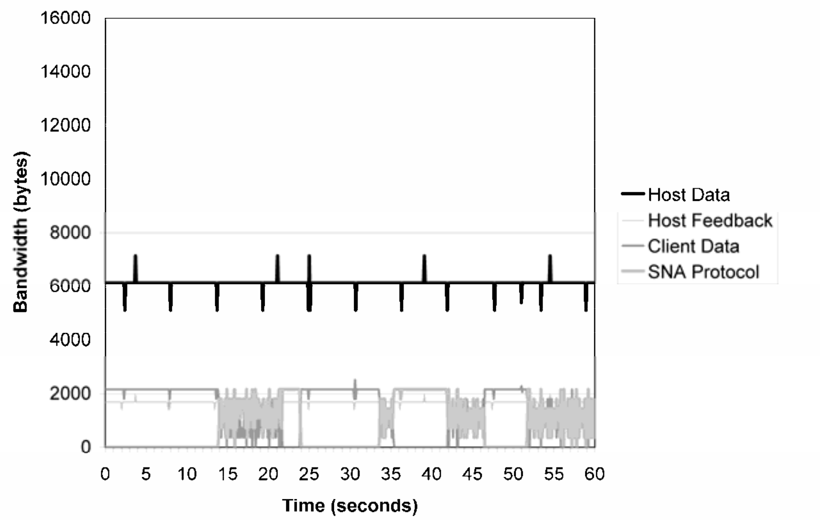


Figure 4.3.11 Advance Wars: Dual Strike - 2 Player

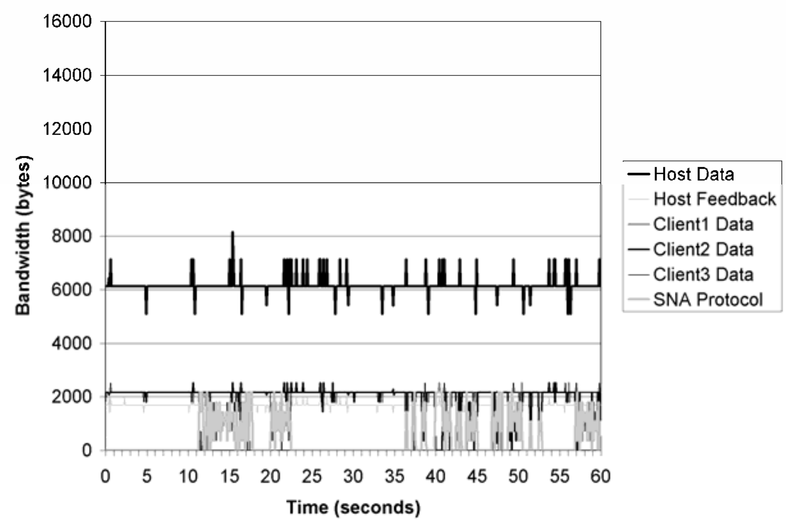


Figure 4.3.12 Advance Wars: Dual Strike - 4 Player

capture of Advance Wars: Dual Strike. In these graphs, there is an extra flow that shows up here that did not show up in the play phases of other games. A certain amount of data was reported to be using the SNA protocol. Apart from Advance War: Dual Strike, the usage of this protocol was limited to inside the download phase. For an unknown reason, Advance Wars: Dual Strike utilizes the SNA type packets as play phase transmissions. When visually comparing the SNA traffic to other traffic, we see that the SNA traffic behavior seems to inversely correlate with the behavior of the client data flows. In cases where the client data flow is not constant and shows variation, we see usage of the SNA flow spike. Such a dependant relationship may be evidence that the SNA flow might be some form of retransmissions. More research would need to be done into the SNA data in order to concretely understand what is happening.

Since nothing can reliably be concluded about the SNA flow, we will move on to examining the other flows. Just like in the GoldenEye: Rogue Agent captures, we see that the client data flows maintain a constant bandwidth usage regardless of the player count. This trait is displayed even stronger in the Advance Wars: Dual Strike captures. The host feedback flow also exhibits this behavior. There is a big difference between the two games though. In Advance Wars: Dual Strike captures, when we compare the host data flows, we see that they are utilizing the exact same amount of bandwidth. That behavior is a clear departure from the increasing bandwidth usage in the GoldenEye: Rogue Agent captures. Although we seem to have a standard for what to expect from the clients, we have uncovered two distinct patterns in the host data flow. Which of the two patterns will the other games demonstrate?

Our observations of Super Mario 64 DS and Pictochat show that the more typical behavior in DS games is for the host data flow to mirror the pattern seen in Advance Wars: Dual Strike. In both games, the host data flow used almost the exact same bandwidth regardless of the number of players involved. Of course, in the case of Pictochat, the overall averages are dependant on the number of messages sent during a capture. Another important note is that neither Super Mario 64 DS nor Pictochat had the perfect consistency we saw in Advance Wars: Dual Strike. This result is not too surprising due to the fact that Advance Wars: Dual Strike was eerily constant across the board.

4.4 Frame Analysis

In order to better understand the network behavior of the Nintendo DS, we were interested in characteristics of individual frames. To facilitate this analysis, we constructed graphs of the cumulative distribution function. We had no expectations for the makeup of individual frames other than the fact that a previous study (Claypool 2005) had noted a tendency toward small packet sizes. These analyses are based on the entirety of each gaming session as opposed to the subsections we examined previously.

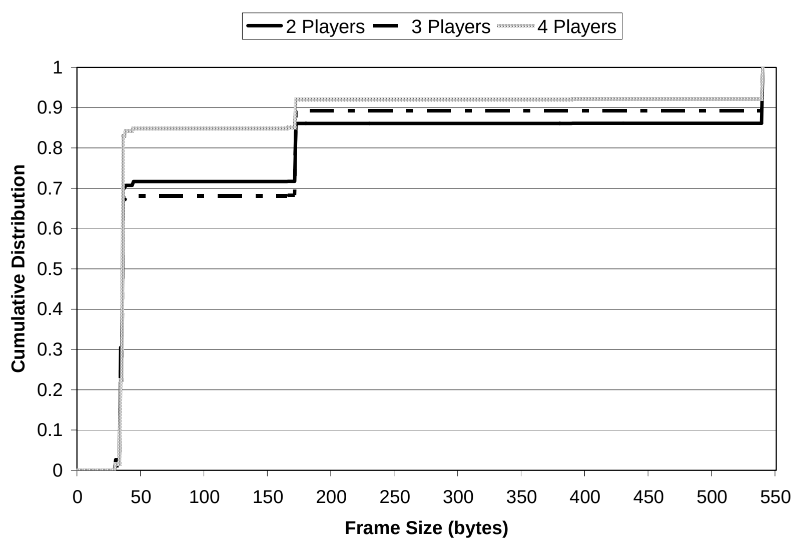


Figure 4.4.1 Cumulative Distribution Function across Player Counts: Super Mario 64 DS

Each gaming session consists of different durations.

The previous section demonstrated that the host bandwidth usage did not change significantly as the number of players increased. Figure 4.4.1 shows a comparison of the cumulative distribution for the host data flow across varying numbers of players in Super Mario 64 DS and shows that this trend is true of the packet sizes as well. In fact, we see that the four player session under examination actually sent a higher proportion of small packets than either of the sessions involving fewer players. The other data flows displayed such similarity to an even greater degree, so this section will concentrate on the trends displayed by comparing the behavior of different games.

Figure 4.4.2 shows a graph of the cumulative distribution function for total bandwidth of all four games we tested. A quick look at this graph demonstrates that the DS relies on small packets. While some larger ones appear, around 90% of all packets fall under 150 bytes for all four pieces of software. Further, around 60% are approximately 25-30 bytes, again regardless of which game is under inspection. While none of the games exhibit exactly identical behavior, the general trends for all four are remarkably similar.

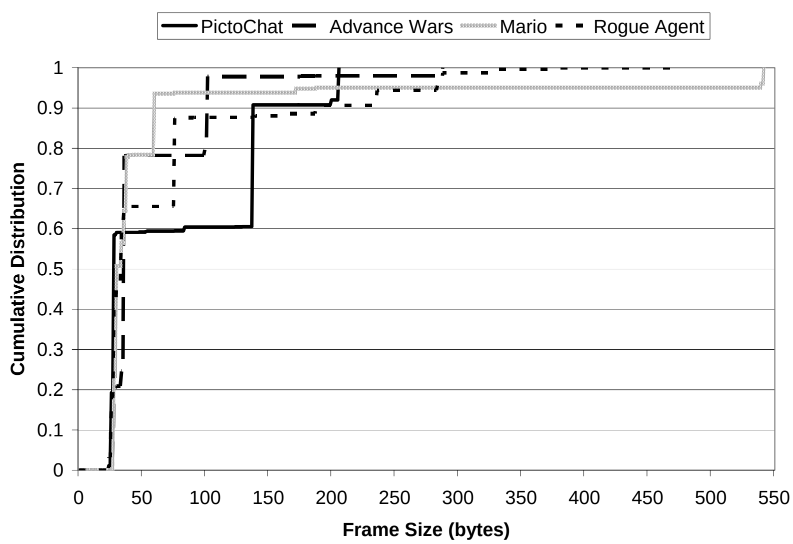


Figure 4.4.2 Cumulative Distribution Function for Total Bandwidth: All Games

In breaking the density information down further, it quickly becomes apparent that certain data flows provide little insight into our area of interest. Regardless of which game is looked at, 100% of all frames sent on the host feedback flow are 28 bytes. Similarly, the broadcast and SNA flows demonstrate very little variation. While these flows factor into the overall network behavior, the graphs that follow display only the host and client data flows.

In Super Mario 64 DS, shown in Figure 4.4.3, it is readily apparent that the vast majority of all packets are quite small, in the vicinity of 70 bytes or less. The host does, however, send a small number of packets that are notably larger, ranging up to over 500 bytes each. These numbers conform quite well to our phase analysis. During the download phase the host must quickly distribute large amounts of data as fast as possible. Once gameplay begins, packets are generally smaller, but there are far more of them, since actual gameplay occupies the majority of the time spent on the wireless network. Super Mario 64 DS exhibits the largest packets of any of the products we examined.

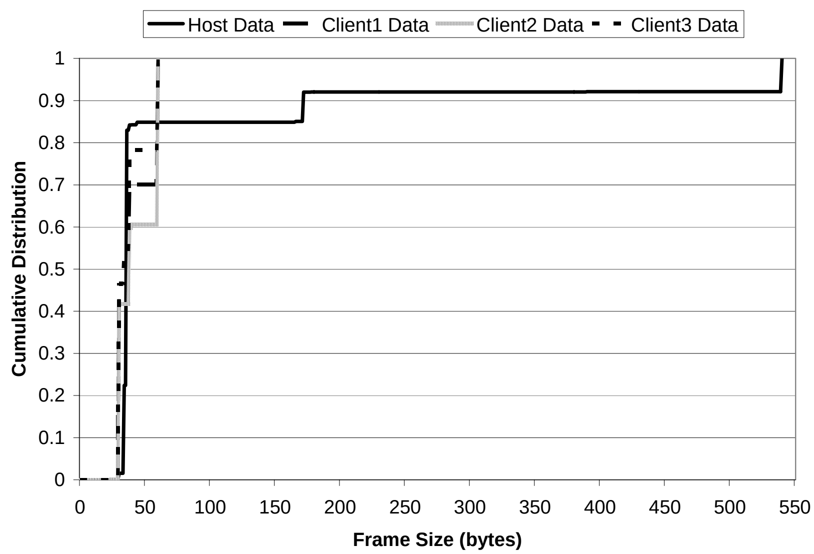


Figure 4.4.3 Cumulative Distribution Function for Data Flows: Super Mario 64 DS

The CDF of GoldenEye: Rogue Agent, seen in Figure 4.4.4, again demonstrates the regularity of the client transmissions. The client flows differ only slightly from those in Super Mario 64 DS. However, the GoldenEye: Rogue Agent host is less consistent. Although half of all the host's packets are under 50 bytes, 15% are 250 bytes or so, and another 20% are around 300 bytes. GoldenEye: Rogue Agent's host demonstrated a higher concentration of large packets than in any other game.

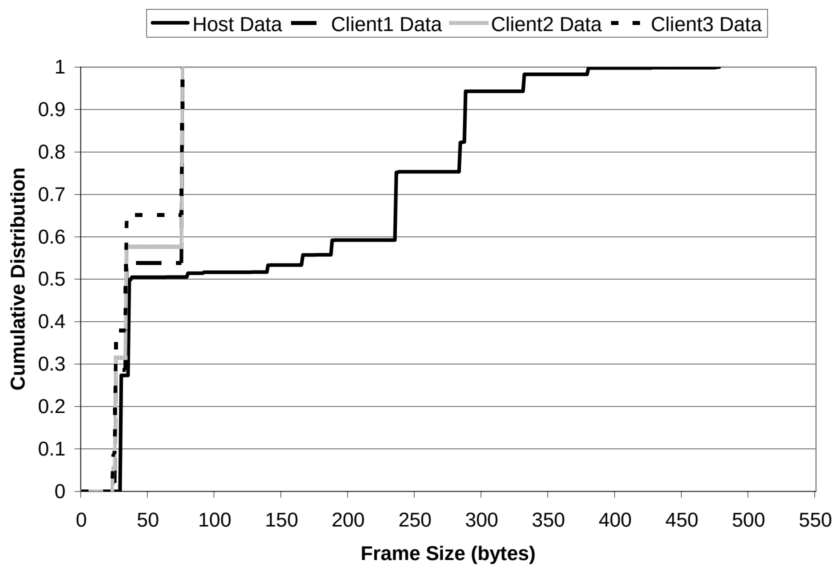


Figure 4.4.4 Cumulative Distribution Function for Data Flows: GoldenEye: Rogue Agent

We believe that the consistent size of the client transmissions is due to the nature of the information the client needs to transmit. The client likely always provides information about the activity of its player only. Therefore the content of its frames would vary only in the specifics such as player position and alignment, but not in the type of information it is sending. The host, conversely, must coordinate all players in addition to environmental

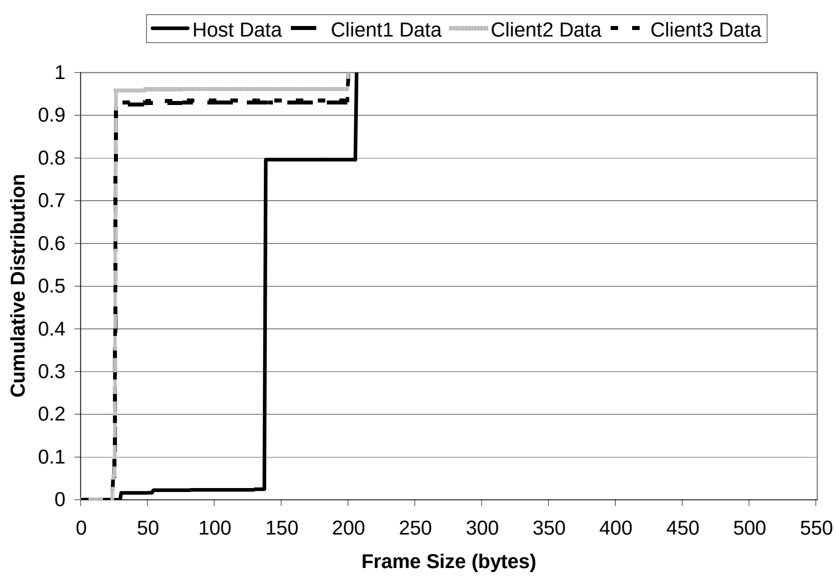


Figure 4.4.5 Cumulative Distribution Function for Data Flows: Pictochat

variables such as the spawning of weaponry.

Pictochat conforms to the general patterns that we have seen in other games. Figure 4.4.5 shows the CDF of the frame behavior of Pictochat. We still see the clients appearing nearly the same, and sending few large packets. The higher bandwidth consumption of Pictochat mentioned earlier could be attributable to the host sending far larger packets on average, as about 80% of the host data packets are around 140 bytes. The remaining packets for both the client and the host might correspond to the message spikes we saw in looking at bandwidth usage.

Advance Wars: Dual Strike depicted the least variance of any of the games. As the CDF in Figure 4.4.6 shows, the packet size within each client flow are essentially identical and show virtually no variation. The host also shows very little variation, with almost 90% of the packets sent on its data flow being approximately 100 bytes. As compared to the clients, the host sends significantl

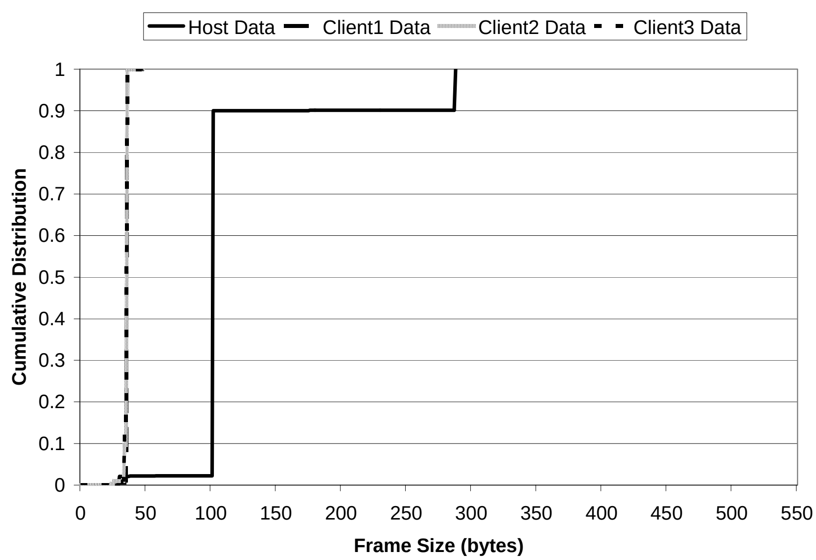


Figure 4.4.6 Cumulative Distribution Function for Data Flows: Advance War: Dual Strike

Although no other game we analyzed demonstrated the lack of variation in frame size seen in Advance Wars: Dual Strike, it typifies the general patterns we observe. The first of these observations is that the clients all send packets that seem to be virtually identical in size. The second is that the host sends much more data in each packet than any of the clients, but most of the packets, regardless of source, are fairly small.

These graphs make apparent that each individual game exercises a fair amount of control over the specific makeup of the communications sent along each flow. This is the case even during the download phase, which might have been supposed to be controlled by the DS architecture.

4.5 Quality of Connection

We initially wanted to devote some attention to what happens to the network traffic under connections of varying quality. Our focus shifted as we refined our areas of interest, and we no longer had interest in extensively testing out different quality connections for each game. We did make one capture of varying quality of a Pictochat session to see if there was much visible difference. Different quality connection levels were simulated by leaving one DS stationary next to the sniffer and having someone walk out of the Fossil Lab and into the ADP lab. After analyzing the capture data, there appeared to be no discernible points to indicate the DSes were compensating for a poor signal. There was no observable increase in bandwidth usage and Ethereal did not report a single retransmitted packet. While we could not see any noticeable difference between the quality test capture and our other captures, there may still have been retransmissions in the form of Nintendo’s own implementation. Since the frames sent by the DS often do not conform to established IEEE standards, we suspect that they utilize an alternate method of data assurance. Such alternate methods might exploit the data flows in order to retransmit lost packets. If this is the case, this means that retransmissions are not controlled by the data link layer, but are passed up to a different layer.

5 Conclusions and Further Work

With console sales down and handheld game sales up in 2005 (Bylund 2005), it is quite possible that the future of gaming is in the palm of our hands. Even if handhelds are not exactly slated to replace console gaming, more and more people are jumping into portable gaming. Because popularity and usage of handheld systems are increasing, it is important to understand the behavior, patterns, trends, and overall effects of the wireless traffic such devices generate.

5.1 *Two Players vs. Three+ Players*

When we were looking at graphs and tables comparing one scenario to another, we see an enormous amount of interesting numbers and statistics. What we have been able to derive from all these digits and calculations are some conclusions about the nature of wireless multiplayer with the Nintendo DS. As we compared our four games, we saw some similar behaviors along with some unique traits. In the case of Pictochat, we could observe exactly when a new image was sent and who sent it just by looking at the specific flows of bandwidth. If we saw an unlabeled graph of incredibly constant bandwidth usage, we could most likely assume it was an Advance Wars: Dual Strike capture. With all these differences, what conclusions can we offer?

Even though each game has its own, unique network model, there are many similarities in the face of all the differences. The first similarity is that none of the DS games exceed the 2 Mbps data rate echelon. The 802.11 IEEE wireless LAN standards allow for a maximum data rate of 54 Mbps. Although the “Nintendo Low Latency Protocol” seems only mildly based on 802.11b, it is certainly an achievement to keep bandwidth usage relatively low. Designing a network to maintain a relatively low bandwidth usage is an achievement for several different reasons. The first is that it is cost effective. Instead of paying high manufacturing prices for hardware that can broadcast as 54 Mbps, Nintendo can use a much cheaper model that has a much lower maximum data rate. Less expensive hardware makes the system both easier to manufacture and cheaper for the consumer. Another reason why maintaining a relatively low bandwidth usage can be found when considering turbulence. Turbulence is a large concern in networking, and the easiest way to avoid large amounts of turbulence is to not introduce large amounts of traffic. Keeping the data rates low increases network reliability.

Even more importantly, adding additional players to the fray of almost any game does not impact the bandwidth usage in an extreme manner. Scaling from a 2 player scenario to a 3 player scenario does not increase bandwidth usage by a third. As a matter of fact, there is a linear increase in bandwidth usage shown in most games. With the exception of GoldenEye: Rogue Agent, we see overall bandwidth usage increase by the same N number of bytes, regardless of whether the game is scaling from 2-to-3 players or 3-to-4 players. Even if we were playing an 8 player melee in Advance Wars: Dual Strike or drawing caricatures of 15 other people in Pictochat, we anticipate that we would see the same scaling behavior that we saw between the 2 player, 3 player, and 4 player test cases.

5.2 Network Architecture

Based on the bandwidth behavior of the Nintendo DS during wireless gameplay, there are a number of conclusions that can be drawn about architecture. First, until either selecting the download game option from the main menu or the multiplayer hosting functionality from within a game, the DS engages in no wireless activity whatsoever.

Once a game is made available, the hosting DS sends standard 802.11 broadcasts proclaiming the availability of the game on a certain wireless channel. Other DSes, when looking for a game, appear to monitor multiple channels in order to locate games being hosted.

Once a client selects a multiplayer game, the first part of the download phase begins. Here data is sent by the host along the host data flow to the client, which the client acknowledges on its corresponding data flow. Once the host player decides that enough other players have arrived, he advances the game state and the second part of the download phase kicks in. More data is transferred, in a similar fashion to the first portion of the download. At this point, the network characteristics may vary depending on the specific game. Once this part of the download phase begins, broadcast packets continue to be sent, but are smaller and less frequent.

When gameplay begins, the host sends most of the game information on its data flow, which functions as a limited, DS-specific broadcast. By using the virtual MAC addresses, these communications are processed by DSes but dropped by other devices. These DS-specific broadcasts are accessible by all involved DSes simultaneously, as opposed to conventional IP traffic. DSes involved in the game send updates to the host and the other clients. The uniformity of the packet sizes for the client data suggests that these packets follow a standard format for each game. The host sends acknowledgements along its feedback flow. There continues to be very minimal broadcast traffic during this phase.

When a game ends, the DSes maintain minimal wireless connectivity, in case the players want to start a new game. If such a selection is made, the play phase begins anew.

If not, the wireless connection is terminated and generally the players must shut down their DSes and restart them in order to be able to choose a new function.

In essence, the wireless communication between Nintendo DSes can be described as a client/server architecture. The notable deviation from the typical implementation of such a system is the usage of multicasting. In most cases where client/server architecture is utilized, we see unicast behavior, where the server sends unique data to each specific client. In multicasting, the server can send out one piece of data to all the clients. The host consistently has the highest bandwidth consumption of any device involved in the game due to the fact that the host must coordinate the entire game. The clients send less information, and each client's output is very similar to that of the other clients. Furthermore, the host packets are, on average, larger than those sent by the clients.

5.3 Impact of the Nintendo Low Latency Protocol

We have discussed, in detail, the behavior of the networks utilized by the Nintendo DS. Scaling based on player count, the similarities between games, the network architecture, and many other topics were examined. The data we gathered allowed us to answer many questions about traits and characteristics of the Nintendo Low Latency Protocol implemented by Nintendo. However, it is also important to take the technical answers and expand them. What does all this mean to the average gamer and to the average 802.11 user?

We looked into how scalable the DS games were as we included additional players. The results showed that the network expanded in a very efficient manner due to the multicasting exploited by the DS networking. Also, even though our analysis was rather shallow, we did not see a huge impact on bandwidth consumption as quality varied from great to poor. Consider the scenario of a gaming café where DS owners could meet and play with each other; it is clear that these attributes would come into play. Our studies only looked at a maximum of 4 players in a game, but in a gaming café, games could easily reach 8 or 16 players when supported. Because of the efficient scaling and low-impact of quality, the airspace would remain less cluttered than we would otherwise observe with typical 802.11 traffic.

These qualities will also play a huge roll in the future as more powerful handheld systems are released. As handheld systems become more powerful, companies will undoubtedly increase the maximum number of players supported in a single game. Imagine connecting with 31 other people to play Counter Strike: Portable. As demands on the wireless networks increase, it is important to have an efficient backbone like what we see in the Nintendo Low Latency Protocol.

Another important feature unveiled in our research was the host/client network architecture employed by the Nintendo DS games. In this architecture, one system is responsible for coordinating the players and the environment while the other systems are only responsible for communicating activities of their respective player. When we consider the relatively low-powered Nintendo DS processor, it would have been interesting to see a distributed processing approach to multiplayer gaming, especially since we saw that processor usage was restrictive to the communication in several games.

Host/client architecture can be rather limiting in the handheld world. Low-powered processors limit the amount of work a system can do, so developers have to be careful not to include too much activity in the game. The effects on the handheld gamers are fewer players, simpler graphics, and less involved environments. The host/client architecture also carries another weakness with it. As any gamer will attest, no one likes a laggy server. If the host of our Super Mario 64 DS session begins experiencing poor connectivity, all players will suffer. Although we did no specific testing in regard to this behavior, it is a typical occurrence in host/client architecture.

The research we have done and the conclusions we have drawn from our data show that the Nintendo Low Latency Protocol is an efficient execution of wireless networking. As computer scientists often prove, there is always room for improvement. But as the first generation of wireless gaming, the Nintendo DS offers strong foundations for future development in this area. Even if the Nintendo Low Latency Protocol does not survive to the next generation of handheld systems, we will surely see many of the traits refined and echoed.

5.4 Future Work

5.4.1 Connection Quality

One topic our group had originally talked about doing was analyzing the network traffic under varying connection qualities. While we managed to take one capture with a varying connection quality, there were no significant differences between this capture and a typical capture. We looked for retransmitted packets in the capture of varying quality but could not find any. However, the absence of retransmitted packets doesn't mean there really weren't any. There may be more to the Nintendo Low Latency Protocol than we have found out so far, and this may be a topic worthy of further research.

5.4.2 Even More Players

Our captures involved anywhere from 2 to 4 players yet some games are capable of supporting up to 8 players. What happens to the network behavior with the maximum number of players straining the wireless bandwidth? How well are the games capable of scaling when the maximum number of players is added?

5.4.3 Impact of Architecture

The research we did asserted that the DSes utilize host/client architecture instead of peer-to-peer or some other variant. A possible path of research would be to look into which systems are important to network quality. Does the host require a good connection to all clients in order to offer a lag-free game? If the host loses its connection with the clients, will one of the clients adopt the role of host? Does a client relay information between the host and another client of the other two systems are not in each others range?

5.4.4 What Truly is SNA/LLC?

Our analysis of the captured data led us to believe that the SNA and LLC packets that Ethereal outputted were not really what they were. Almost all of the SNA packets observed were labeled as malformed and SNA is fairly obsolete at this point. Since Nintendo claims the DSes are communicating via their proprietary Nintendo Low Latency Protocol, these SNA and LLC packets may just be packets mislabeled by Ethereal. If this is the case, what are the SNA and LLC packets in DS communications?

5.4.5 Analysis of TCP/IP Stack Games

Nintendo has recently released several games for the DS capable of multiplayer competition over the Internet. Most DS games communicate via an ad-hoc network, when the devices communicate directly with one another. This ad-hoc behavior means that the DS does not need to implement a TCP/IP stack. Games such as Mario Kart DS and Animal Crossing: Wild World implement a TCP/IP stack within the application in order to send and receive packets over the Internet. Players not within direct wireless contact could play a game as long as both have a connection to the Internet. Is the network behavior much different for games with TCP/IP stacks than for ones that must communicate directly?

5.4.6 Two DS Networks Occupying the Same Airspace

All our captures were taken with a single DS network game occurring at a time. Since we did observe some of the games switching wireless channels, this could mean that a DS network makes accommodations for the wireless traffic currently in the vicinity. Since the DS infrastructure is based on the DS-specific broadcast, what would happen if multiple DS network games were being played within close proximity to one another? Would they interfere with one another or would they be able to avoid this problem with channel switching?

5.4.7 DS Network Impact on Other 802.11 Networks

It is well known that congestion is a large issue in networking. In 802.11 wireless networks, congestion can play a drastic role in the viability of any networks utilizing the airspace. Even though Nintendo DSes operate using either 1Mbps or 2 Mbps, they can still contribute sizably to congestion. Future research would be possible in this area and researchers would want to explore how big of an effect DS traffic has on other 802.11 devices within the same airspace.

6 Bibliography

A. Bylund, Ars Technica, Console Sales Down, Handheld Games Sales Up in 2005 – 1/15/2006, http://arstechnica.com/news.ars/post/20060115-5983.html, Ars Technica, LLC, Copyright 1998

A. Jardosh, K. Ramachandran, K. Almeroth, E. Belding-Royer, Understanding Link-Layer Behavior in Highly Congested IEEE 802.11b Wireless Networks - Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW), University of California, Santa Barbara, November 2005

H. Yomogita, Nintendo DS: The Secret Within - February 2005 Issue, http://neasia.nikkeibp.com/neasia/000260, Nikkei Business Publications Asia Ltd Copyright 1996

J. Gretarsson, F. Li, M. Li, A. Samant, H. Wu, M. Claypool and R. Kinicki, Performance Analysis of the Intertwined Effects between Network Layers for 802.11g Transmissions – Wireless Multimedia Networking and Performance Modeling, Montreal, Canada, October 2005

J. Smed, T. Kaukoranta, and H. Hakonen, Aspects of Networking in Multiplayer Computer Games - Volume 20, pages 87-97, The Electronic Library, Copyright 2002

M. Claypool, On the 802.11 Turbulence of Nintendo DS and Sony PSP Hand-held Network Games –Proceedings of the 4th ACM Network and System Support for Games (NetGames), Hawthorne, NY, USA, October 2005

M. Heusse, F. Rousseau, G. Berger-Sabbatel and A. Duda, Performance Anomaly of 802.11b –Proceedings of IEEE INFOCOM, Grenoble, France, 2003

M. Wright, In the Game?, 6/9/2005, http://www.edn.com/article/CA605509.html, EDN Copyright 1997

M. Yarvis, K. Papagiannaki, and W. S. Conner., Characterization of 802.11 Wireless Networks in the Home., In Proceedings of 1st workshop on Wireless Network Measurements (WiNMee), Riva del Garda, Italy, Apr. 2005

Nintendo of America Inc., Customer Service email correspondent, 2005

Reuters, U.S. cities set up wireless networks, 5/4/2005, http://www.cnn.com/2005/TECH/internet/05/04/life.wireless.reut/, Reuters Copyright 2005

S. Zander and G. Armitage, A Traffic Model for the Xbox Game Halo 2 –Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Stevenson, Washington, USA , June 2005

T. Bramwell, Nintendo Plans Wireless GBA Adapter, 09/26/2003, http://gamesindustry.net/content_page.php?section_name=pub&aid=2309, Eurogamer Network Ltd. Copyright 2002

W. Feng, F. Chang, and J. Walpole Provisioning On-line Games: A Traffic Analysis of a Busy Counter-Strike Server –, Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW), Marseille, France, November 2005

Wikipedia, IEEE 802.11, http://en.wikipedia.org/wiki/IEEE_802, accessed February 26, 2006

Wikipedia, Systems Network Achitecture, http://en.wikipedia.org/wiki/Systems_Network_Architecture, accessed February 26, 2006