Schrader, K.R., Mullins, B.E., Peterson, G.L., and Mills, R.F., "An FPGA-Based System for Detecting and Tracking Contraband Digital Information Transmitted Via Peer-to-Peer Protocols," International Journal of Security and Networks (IJSN), (Accepted: Feb 09, Estimated Publication Date: Fall 2010)

An FPGA-Based System for Tracking Digital Information Transmitted Via Peer-to-Peer Protocols

Karl R. Schrader, Barry E. Mullins, Gilbert L. Peterson, and Robert F. Mills
Air Force Institute of Technology

***Abstract*—At issue for any organization is the illicit dissemination of sensitive information using file sharing applications within a network, and tracking terrorist cells or criminal organizations that are covertly communicating using Voice over IP (VoIP) applications. This paper presents a field programmable gate array (FPGA)-based embedded software tool designed to process file transfers using the BitTorrent peer-to-peer protocol and VoIP phone calls made using the Session Initiation Protocol (SIP). The tool searches a network in real time for selected peer-to-peer control messages using payload analysis and compares the unique identifier of the file being shared or phone number being used against a list of known contraband files or phone numbers. If the identifier is found on the list, the control packet is added to a log file for later forensic analysis.**

Results show that the FPGA tool processes peer-to-peer packets of interest 92% faster than a software-only configuration and is 99.0% accurate at capturing and processing BitTorrent Handshake messages under a network traffic load of at least 89.6 Mbps. When SIP is added to the system, the probability of intercept for BitTorrent Handshake messages remains at 99.0% and the probability of intercept for SIP control packets is 97.6% under a network traffic load of at least 89.6 Mbps, demonstrating that the tool can be expanded to process additional protocols with minimal impact on overall performance.

***Index Terms*—P2P, network forensics, BitTorrent, VoIP, packet analysis.**

I. INTRODUCTION

Peer-to-peer (P2P) networking has changed the way users search for, send, and receive digital information over the Internet. Instead of relying on interactions with centralized servers to upload and download digital content, users now share music, movies, documents, and conversations directly with other users. While P2P networking provides new and powerful applications for the legitimate distribution of digital information, it is also being used for many illicit purposes as well.

One high-profile illicit use of P2P networking technology is the intentional or inadvertent distribution of sensitive government information to unauthorized personnel. In the summer of 2008, sensitive engineering and communication documents about the Marine One presidential helicopter were sent to an Internet Protocol (IP) address in Iran. Upon investigation by the United States Navy, it was determined that a defense contractor inadvertently released the documents after one of the company's computers was loaded with a file sharing program, exposing the documents to users worldwide [1].

Another illicit use of P2P networking technology is for the dissemination of child pornography. The Federal Bureau of Investigation's (FBI) Regional Computer Forensics Laboratory states in its 2007 annual report that "cybercrime, which includes crimes against children and child pornography, is the offense for which law enforcement requested assistance most often" [2]. In addition, a 2005 Government Account Office report stated that "[P2P] technology is increasingly popular for disseminating child pornography" [3].

Another area in which P2P networking is being used for illegal activities is covert communication among terrorist cells through Voice over IP protocols. According to one British Government official, VoIP calls are "seriously undermining" MI6's ability to intercept and track Taliban communications [4]. In addition, during the December 2008 terrorist attacks in Mumbai, India, the attackers' Pakistan-based handlers sent instructions, intelligence, and encouragement using VoIP-based Internet phones [5]. During the 3-day series of attacks, the terrorists were able to communicate without their calls being traced or intercepted by authorities.

To help combat these illicit uses for P2P networking, this research develops a system to identify and track any type of digital information transmitted on a network using P2P protocols. Several methods have already been developed to accomplish this task and are in use today, but they depend on the use of honeypots to lure targets into downloading contraband material [6], physical access to the suspected file sharer's computer [7], active searching of the Internet for contraband files to download [8], or active interception and modification of contraband file sharing requests [9]. All of these methods are active attempts to discover illicit file sharing, with the drawback that they can all be detected and possibly circumvented by file sharers that are aware of their presence. In contrast, the system developed for this research consists of a suite of tools that passively detects and tracks illicit file sharing on a target network without affecting the flow of traffic on the network, making it impossible for users of the network to determine the presence of the system.

The developed digital forensic tool, TRacking and Analysis for Peer-to-Peer (TRAPP) system, allows an investigator or system administrator to monitor network traffic in real-time for any digital information that meets the user's definition of contraband being shared using P2P protocols. The TRAPP system (Figure 1) is designed to be set up on the gateway between an owned network and the Internet. As packets pass through the gateway, copies are sent to the system for analysis. For each packet received, TRAPP inspects the packet to determine if it is a control packet for a P2P protocol of interest. If the packet is not a P2P control packet, it is discarded. If the packet is a control packet, the system extracts from the packet's payload the unique identifier for the data being shared, and attempts to match the identifier against a list of files of interest in the system's memory. If a match is not made, the packet is discarded. If a match is made, the control packet is recorded in a log file for future analysis.

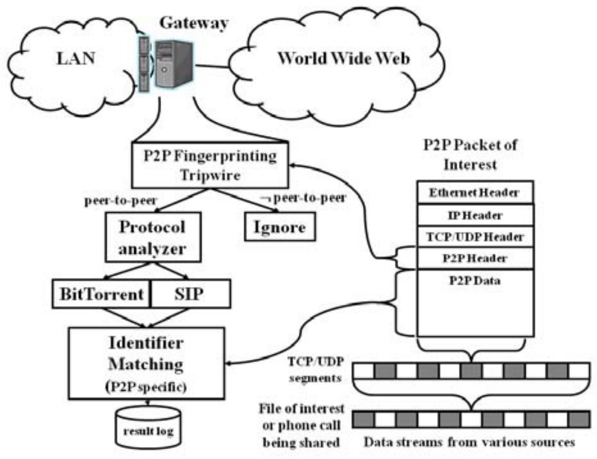

Figure 1. The Proposed TRAPP System

There are three primary goals for this research. The first goal is to construct a hardware-based system using a field programmable gate array that analyzes all network traffic sent to it, detects packets belonging to a specific P2P protocol, compares the digital information being shared against a list of interest, and in the case of a match, records selected control packets from the P2P session in a log file. The second goal is to optimize the system to increase the probability of detecting and recording all control packets, even when network traffic is at nearly the full capacity of the system's Ethernet controller. The last goal is to demonstrate the system's expandability by modifying it to accept an additional P2P protocol with no impact on overall performance.

The following section presents related work in the area of detecting illicit file sharing and background on how the BitTorrent and SIP protocols work. Section 3 describes the construction of the research prototype and outlines the methodology used to design, set up, and conduct the experiments to test the effectiveness of the TRAPP system. Section 4 provides a discussion and analysis of the experimental results. Following this, the conclusions drawn from the experimental results, the significance of the completed TRAPP system, and areas for future research are given.

II. BACKGROUND AND RELATED WORK

A. Current Methods of Identifying Downloaders of Illegal Files

Given the rapid rise of P2P file sharing, law enforcement agencies and copyright holders are struggling to keep up with illegal file sharers. Currently, there are several methods available to these entities to identify and track illegal file downloaders, several of which are discussed and analyzed below.

1) Honeypots: One common method of identifying and tracking uploaders and downloaders of contraband files is through the use of honeypots. In the context of this discussion, a honeypot is a trap set by a government entity or private corporation with the purpose of detecting and tracking illegal activities. In its most basic execution, a computer is set up on the Internet with a collection of illegal files. When a computer attempts to download the illegal files, the downloader's IP address, port number, date, time, and the packets being downloaded are recorded by the honeypot owner. Badonnel et al. designed and tested a management platform for tracking illegal file sharers in P2P networks [6].

While effective against illegal downloaders who access the honeypots, there are several shortcomings to using this method for identifying and tracking illegal downloaders. First, the illegal downloader has to access the honeypot. To circumvent them, blacklists of IP addresses known to host honeypots have been created. Today, programs such as Peer Guardian are specifically designed to act as a downloading firewall that blocks these blacklisted IP addresses, preventing the user's P2P software from downloading from them [10]. Second, illegal downloaders have to locate and actively download from the honeypot in order for the authorities to identify them. For certain classes of highly illegal files, such as child pornography, hard to find and password protected websites are used to keep the general public (and law enforcement) from accessing and downloading them [11].

2) The BitTorrent Monitoring System: Another method for detecting and tracking illegal file downloaders is the BitTorrent Monitoring System (BTM), designed and presented by Chow et al. [8]. BTM is a system that automatically searches the Internet for BitTorrent-based downloadable files, analyzes the files to determine if they are illegal, attempts to download the suspected illegal files, and finally records tracking information on who provided the files for download.

BTM has the potential to become a powerful law enforcement tool in combating illegal file sharing. However, there are problems with the system. First, due to the sheer number of torrent files that are available on most torrent websites, the BTM system currently suffers from a very slow processing time. As the number of sublevels covered by the search algorithm increases, the number of total torrent files to be analyzed increases exponentially, leading to a drastically reduced total processing time. Because the BTM system cannot run in real time, it currently cannot keep up with the constantly changing peer lists being produced by the tracker sites being monitored.

3) Hardware Recovery of Illegal Files: Another method of identifying potential illegal file downloaders is to search the suspect's computer for illegally downloaded files using digital forensic techniques. In their research, Adelstein and Joyce introduce a digital forensic tool called File Marshal which allows law enforcement to automatically detect and analyze P2P software usage on a hard drive [7].

As with the use of honeypots, there are several drawbacks to the hardware recovery method. First, the investigator must physically possess the hard drive. In most cases, this requires some kind of legal action to force a suspect to turn over his computer to the investigator, which can be an extremely invasive procedure.

Second, in order to recover a suspect's hard drive for analysis, investigators must first determine that the hard drive is worth analyzing. In other words, investigators must already suspect the computer as containing illegal files before acting to confiscate the drive for analysis. This type of investigation is extremely time and labor intensive, limiting the ability of law enforcement to tackle widespread illegal file sharing using this method.

4) The CopyRouter Peer-to-Peer Tracking System: In October 2008, MSNBC reported that an Australian company, Brilliant Digital Entertainment Ltd., was marketing a new Internet monitoring tool known as CopyRouter [9]. The

CopyRouter system inspects every packet entering or leaving a target network, looks for P2P search results that reference files that are on a known contraband list (such as child pornography, the stated primary application of the system), and replaces the illicit file reference with one that leads the user to a law enforcement server instead [12].

While the CopyRouter system seems like an effective contraband tracking system, there are several lingering questions surrounding its implementation. First, CopyRouter is a proprietary system, and to date, Brilliant Digital Entertainment has yet to release any specifications or experimental data on the system's speed, effectiveness, or ability to process all packets at full network speed.

Second, while seemingly effective for P2P systems where only one uploader is involved, such as Gnutella, the system's ability to monitor distributed P2P systems such as BitTorrent is questionable. As discussed later in this section, as a BitTorrent client downloads pieces of the file from peers, each piece is hashed and compared against the .torrent file. If the hashes do not match, which is always the case when the user downloads from the law enforcement content server, the piece is simply discarded.

Finally, CopyRouter is an active detection and tracking system, meaning that each packet entering or leaving the network is read and possibly modified before being allowed to continue through the gateway. One consequence of this is that the system's presence can theoretically be detected by users with enough knowledge of how the system works. These users can then modify their behavior and simply not use the monitored network to share illicit information.

B. The BitTorrent Protocol

The P2P protocol of interest is the BitTorrent protocol [13]. BitTorrent differs from other distributed P2P protocols in that it allows downloaders to download pieces of files from tens or hundreds of other users simultaneously. To further speedup downloads, every user that downloads pieces of files also uploads those pieces he already possesses. By aggregating the slower upload speeds of hundreds of peers, the protocol can achieve very high download rates [14].

The key BitTorrent component used in this research is the info hash of the file dictionary. To create the info hash, the SHA-1 algorithm [15] is applied to the information dictionary contained in the .torrent file. The resulting message digest is labeled as the "file info hash", which uniquely identifies the file being offered for download regardless of the file description contained in the .torrent file. The client provides the file info hash as the file identifier in the request for a peer list and when establishing connections using the handshake message. By comparing this hash against a list of hashes compiled from the .torrent files of data of interest, we will be able to determine if the client is attempting to download or upload a file of interest.

In this research, the TRAPP system identifies and analyzes BitTorrent handshake packets, which are used by BitTorrent's Peer Wire Protocol to establish data transfer session between peers. The system extracts the 20-byte file info hash, identifies the data being transferred from the handshake packet and attempts to find a match to the list of interest.

C. The Session Initiation Protocol

In 1999, Henning Schulzrinne submitted the plan for a protocol to establish and control multiparty multimedia sessions, and was approved by the IETF as RFC 2543, the Session Initiation Protocol [16]. According to the updated version of the protocol, IETF RFC 3261, "SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls" [17]. The goal of SIP is not to exchange data between participants; rather, its purpose is to allow the participants to find one another, and to manage the data connection once established. This allows SIP to be used for a large number of data transfer applications, such as interactive gaming, media on demand, and voice or video conferencing [16].

Because SIP is an open source protocol, it is rapidly becoming the de facto standard for multimedia session control. SIP is currently used by the popular VoIP provider Vonage [18], by Microsoft for its MSN Messenger system [16], and by Yahoo! for its Yahoo! Messenger system [19]. SIP has also been selected by the 3G community to be its session control protocol for the 3G cellular network [16], and Google is planning to incorporate SIP into the protocol used by its popular Google Talk service [20].

In this research, the TRAPP system identifies and analyzes SIP INVITE and BYE packets, used to set up and take down communications sessions, in order to extract the caller and receiver SIP Uniform Resource Identifiers (URIs). These two types of packets are also used to determine the beginning and end of the communication session.

III. THE TRAPP SYSTEM APPARATUS AND TESTING METHODOLOGY

The objective of this research is to develop a system to identify and track specific digital information being transmitted on a network using P2P protocols. The proposed system will detect P2P transmissions on a target network, classify them by specific P2P protocol, compare the digital file being transmitted against a list of interest, and identify the sender and recipient by IP address.

The goals of this research are to:

- Construct an FPGA-based system that analyzes traffic on a network, detects a selected P2P protocol, compares the digital information being shared against a list of interest, and in the case of a match, records selected control packets ("packets of interest") from the P2P session in a log file.
- Optimize the system such that it is able to detect and record all packets of interest on the network, even under a heavy (approximately 90 percent utilization) non-P2P traffic load.
- Modify the system to detect and record control packets of interest belonging to a second P2P protocol with no negative impact on overall performance.

A. Approach

The forensic tool is designed using the Virtex II Pro FPGA development board [21] and for the BitTorrent P2P protocol. Implementing the system on an FPGA allows the software application to directly access the Ethernet controller buffers, bypassing the rest of the network stack and increasing the system's simplicity and speed. Once the system is optimized and tested using this protocol, the system is expanded to also process the Session Initiation Protocol, and tested again.

Figure 2 shows the overall functionality of the design. When the system processes a packet, the following occurs:

1) The tool fingerprints the frame received from the network

by extracting the first 32 bits of the frame's payload.

2) The 32-bit fingerprint is then compared to the first 32 bits of a BitTorrent Handshake message, which is `0x13426974` [22], a SIP INVITE message, which is the ASCII string "`INVI`", or a SIP BYE message, which is the ASCII string "`BYE `" [17].
3) If the fingerprint of the frame's payload is not a match to any of these strings, the frame is discarded.
4) If the fingerprint matches that of a BitTorrent Handshake message, the first 32 bits of the Handshake's file info hash is extracted from the frame, and compared against a list of hashes belonging to files of interest using a binary search.
5) If the fingerprint matches that of a SIP message, the first 12 characters of the TO and FROM SIP URIs are extracted from the frame, and each is compared against a separate list of SIP URIs of interest using a binary search.
6) If the file info hash/SIP URI is not on the list, the frame is dropped.
7) If the file info hash/SIP URI is on the list, the frame is saved in a Wireshark-readable log file and placed on a compact flash card. The frames recorded in the log file are later analyzed to extract IP address information, which can then be used to perform tracking and forensic analysis.

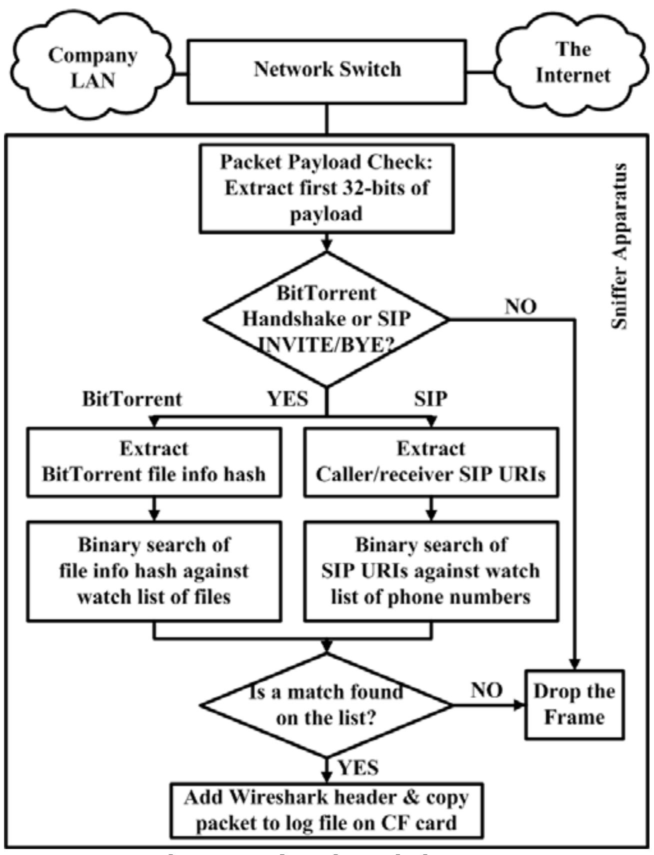
Company LAN
Network Switch
The Internet
Sniffer Apparatus
Packet Payload Check: Extract first 32-bits of payload
BitTorrent Handshake or SIP INVITE/BYE?
NO
BitTorrent
YES
SIP
Extract BitTorrent file info hash
Extract Caller/receiver SIP URIs
Binary search of file info hash against watch list of files
Binary search of SIP URIs against watch list of phone numbers
Is a match found on the list?
NO
Drop the Frame
YES
Add Wireshark header & copy packet to log file on CF card

Figure 2. Packet Data Flow through the TRAPP System

This research is divided into two experiments: finding a software configuration for the system that processes BitTorrent packets of interest as quickly as possible and expanding the system to incorporate the SIP protocol without sacrificing overall performance. Each of the two experiments is comprised of three tests: calculating packet processing time, calculating probability of packet intercept under a non-P2P workload, and calculating probability of packet intercept under an all-P2P workload. Overviews of the two experiments are outlined below.

1) Experiment 1: Finding an Optimal Software Configuration: The first experiment seeks to determine the optimal hardware/software configuration of the system that processes BitTorrent packets of interest as quickly and as accurately as possible. This experiment is split into three parts. First, each hardware/software configuration is tested against several types of packet sizes and formats, and the amount of processor time needed to process each packet is examined. Second, a series of BitTorrent packets of interest are sent to each configuration in a high non-P2P network utilization environment, and the overall probability of intercept of a packet of interest is calculated for each configuration. Finally, a series of BitTorrent packets is sent to the system at near-full network utilization in order to determine the probability of intercepting consecutive packets of interest.

2) Experiment 2: Expanding the Forensic Tool to Incorporate VoIP Functionality: The second experiment seeks to determine if adding functionality to process SIP packets, in addition to BitTorrent packets, degrades overall system performance. For this experiment, the optimal configuration found in the first experiment is modified to also include detection and processing of SIP packets of interest. As with the first experiment, this experiment consists of three parts. First, the modified configuration is tested against several types of BitTorrent and SIP packets, and the amount of time needed to process each packet is examined. Second, a series of BitTorrent and SIP packets of interest are sent to the modified configuration in a high network utilization environment, and the probability of intercept of each type of packet of interest is calculated. The results of this experiment are then compared against the results of the first experiment to determine if the system's overall performance in processing BitTorrent packets of interest is negatively impacted. Finally, a series of P2P packets is sent to the system at near-full network utilization in order to determine a measure of the probability of intercepting consecutive packets of interest for each P2P protocol.

B. System Boundaries

The System Under Test (SUT) for this research is the TRAPP Forensic Tool System. A block diagram of the SUT is shown in Figure 3. It consists of the following FPGA components: the TRAPP software, the Power PC processor, the system clock, the Ethernet controller, the compact flash card controller, and the RS232 controller. The Component Under Test (CUT) is the TRAPP software. Specifically, various modifications to the software execution flow will be compared to the baseline software architecture.

The performance metrics of the system consist of the time required to process a packet and the probability of successful intercept of a packet. The system parameters consist of the size of the list of interest used by the system, network speed, the software configuration of the TRAPP system, and the number and types of P2P protocols supported by the system. Workload parameters include the type of BitTorrent Packet used, the type of SIP packet used, and the total network utilization as a percentage of network capacity.

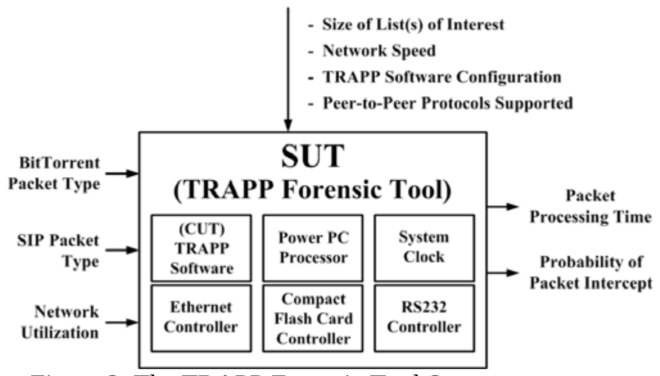

Figure 3. The TRAPP Forensic Tool System

C. Performance Metrics

In order for the system to be effective, it must have a high probability of successfully intercepting, processing, and recording those packets on the network that belong to a P2P protocol supported by the system, and whose identifiers are on the lists of interest. By extension, in order for the system to successfully intercept these packets of interest, it must have the capability to analyze all traffic on a network, which necessitates the requirement of processing each packet as quickly as possible for a given set of parameters. Thus, the following performance metrics are defined:

- *Packet Processing Time*: The number of CPU cycles, as measured by the Power PC processor's System Timer, that are required to accomplish the following: determine if a packet has been received by Ethernet controller, inspect the packet for P2P protocols, match the packet's identifier against the appropriate list, record the packet if necessary, and make the Ethernet controller available to receive another packet entering the system.
- *Probability of Packet Intercept*: The probability of a packet, whose format matches a P2P protocol supported by the system, and whose identifier matches an entry on a list of interest, being successfully recorded in the system intercept log.

D. System Parameters

- Size of List of Interest: This is the size of the list of interest, expressed by the number of entries in the list. For the BitTorrent protocol, an entry is a 160-bit file info hash. For the SIP protocol, an entry is the first 12 digits of a SIP URI. Because the system uses a binary search algorithm to perform the hash/SIP URI matching process, each doubling of the list size will add a maximum of one comparison to the total algorithm execution time. For this research, a sample list size of 1000 entries is used for both the file info hash list and the SIP URI list.
- Network Speed: This is the maximum speed of network data entering the system through the Ethernet connection. The Ethernet controller on the Xilinx II Pro board used in this research is capable of connecting to either a 10 Mbps or a 100 Mbps network. For this research, the 100 Mbps connection option is used.
- TRAPP Software Configuration: The software code used to execute TRAPP functions using the Power PC processor on the FPGA. As procedures and features contained in the software are added, removed, or modified, the overall functions and performance of the system are affected.
- Peer-to-Peer Protocols Supported: This is the set of P2P protocols that the system is capable of detecting and analyzing. For the first experiment, the BitTorrent protocol is the only member of this set. In the second experiment, the Session Initiation Protocol is added to the set.

E. Workload Parameters

- BitTorrent Packet Type: In this study, three different types of BitTorrent packets are used: a BitTorrent Handshake packet whose file info hash matches an entry on the list of interest, a BitTorrent Handshake packet whose file info hash does not match an entry on the list of interest, and a packet that is not a properly formatted BitTorrent Handshake packet.
- SIP Packet Type: Three different types of SIP packets are used in this study: a SIP INVITE packet whose TO and FROM SIP URIs match an entry on the list of interest, a SIP INVITE packet whose TO and FROM SIP URIs do not match an entry on the list of interest, and a SIP BYE packet whose TO and FROM SIP URIs match an entry on the list of interest.
- Network Utilization: This is the total amount of traffic entering the system. For the first test, the network utilization is limited to single packets injected into the system to measure the time required to fully process the packet. For the second test, a 1.1 gigabyte video file is transferred from one node on the network to another node on the network using the Windows NETBIOS file transfer protocol to generate a non-P2P traffic load. This transfer injects a load of between 89.6 Mbps and 89.7 Mbps into the system, which equates to approximately a 90% utilization of the 100 Mbps Ethernet connection. For the third test, a continuous stream of identical P2P packets of interest is injected into the system, with the network utilization varying with the type of P2P packet.

F. Configuration

The software configuration is the factor designated as the Component Under Test. It controls all aspects of the system, including what information is provided to the user, how network data is captured and analyzed, and how packets of interest are stored. The six levels chosen for this factor are detailed below.

1) Control Configuration: The system is implemented as an embedded software application using the Power PC core on the Virtex II Pro FPGA development board. Xilinx-provided drivers and built-in functions are used where possible, with custom software built to accomplish the following functions: read the data file containing the file info hashes of the list of interest, perform packet payload inspections, copy BitTorrent Handshake frames to on-chip RAM, perform the hash matching, and write the frame data to the log file on the compact flash card.

Listed below are the salient features of the Control configuration:

- All modules are executed in software. The only hardware modification made is to enable the Ethernet controller to operate in promiscuous mode.
- To simplify the software code as much as possible, the Ethernet controller is limited to one receive buffer, caching is not used, and no user alerts are generated for the user.
- Packets of interest are copied three times. The first copy is from the Ethernet controller buffer to block RAM upon

detection of the 32-bit BitTorrent signature in the packet's payload. If the file info hash is found on the list, the frame is copied from RAM to a character array, and then from the array to the log file on the compact flash card.

- Frames are copied to the compact flash card as they are processed. The system waits until the current frame has been completely processed and sent to the compact flash card before beginning to process another frame.

2) User Alerts Configuration: This modification adds to the system user notifications, via the serial port, of any P2P control packets that are found by the system. The messages consist of the type of P2P packet found, whether the file info hash matches an entry on the list of interest, and the file info hash's position on the list of interest. Because the serial port runs at a much lower speed than the CPU and the processing bus, it is hypothesized that sending any data over the RS232 connection causes a dramatic slowdown in overall processing time.

3) Packet Write Configuration: In this modification, all captured packets of interest are stored within a RAM block instead of writing them individually to the compact flash card. When the system is shut down, all data are then transferred from the block RAM to the compact flash card. By storing the data within RAM, the only write functions to the compact flash card are performed before packet sniffing begins and after packet sniffing terminates. It is hypothesized that writing to the compact flash card is a high-latency process, and that its removal will result in a significant processing time savings.

4) Dual Buffer Configuration: This modification adds a second receive buffer to the Ethernet controller [23]. This allows one frame to be read and processed while another frame is received. The goals for this optimization are to give the comparison and copying routines additional time to execute, and limit the number of frames dropped due to a full receive buffer.

5) Cache Configuration: This modification enables the instruction and data caches for the Power PC processor. By allowing the FPGA to cache processor instructions, heap data, and stack data instead of performing multiple reads and writes to block RAM, a significant amount of processing time should be saved.

6) Combined Configuration: This is the combined case of the CUT incorporating the Packet Write, Dual Buffer, and Cache optimizations into a single system. The goal for the integration is to take advantage of each optimization individually and to possibly gain synergistic time savings from the combination of all four optimizations.

G. Experimental Environment

To conduct the two experiments, the experimental setup shown in Figure 4 is created. The experimental environment consists of the following components:

- One Cisco Catalyst 2900XL 100 Mbps switch configured with 22 standard ports and 2 spanning ports.
- Two Dell Inspiron Windows XP Service Pack 2 laptops loaded with µTorrent 1.7.7 [24], a popular BitTorrent client, and X-Lite 3.0 [25], a popular VoIP phone client, and connected to the switch.
- One Dell Inspiron laptop that is dual-equipped with the BackTrack 2.0 Linux environment [26] and Windows XP Service Pack 2, and is connected to the switch. The BackTrack environment contains the hping 3.0.0 [27] utility, which is used to inject the crafted BitTorrent and SIP packets. The Windows environment contains the VMWare 2.0.5 [28] utility to run a TrixBox 2.2 [29] SIP proxy and registrar server for use with the X-Lite clients.
- One Virtex II Pro FPGA system (the SUT), which is connected to a spanning port on the switch.
- One Dell Inspiron Windows XP Service Pack 2 laptop loaded with Wireshark 1.0.1 [30], which is connected to a second spanning port to act as a packet sniffer for an experimental control.
- One Dell Windows XP Service Pack 2 laptop, which is connected to the SUT and is used to configure and load the Virtex II Pro via a USB port. The laptop is also equipped with TTermPro [31], a HyperTerminal application used to receive alerts from the SUT via a RS232 serial port.

The actual experimental setup used is shown in Figure 5.

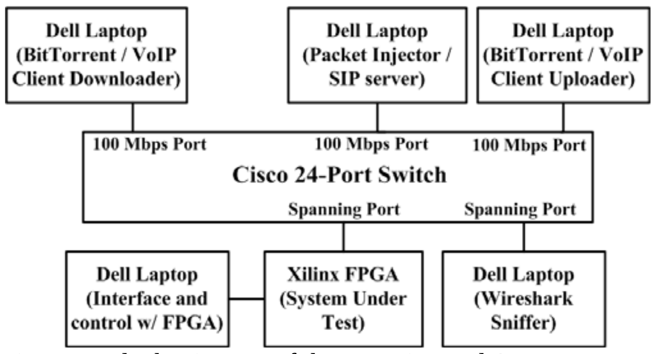
Dell Laptop (BitTorrent / VoIP Client Downloader)
Dell Laptop (Packet Injector / SIP server)
Dell Laptop (BitTorrent / VoIP Client Uploader)
100 Mbps Port
100 Mbps Port
100 Mbps Port
Cisco 24-Port Switch
Spanning Port
Spanning Port
Dell Laptop (Interface and control w/ FPGA)
Xilinx FPGA (System Under Test)
Dell Laptop (Wireshark Sniffer)

Figure 4. Block Diagram of the Experimental Setup

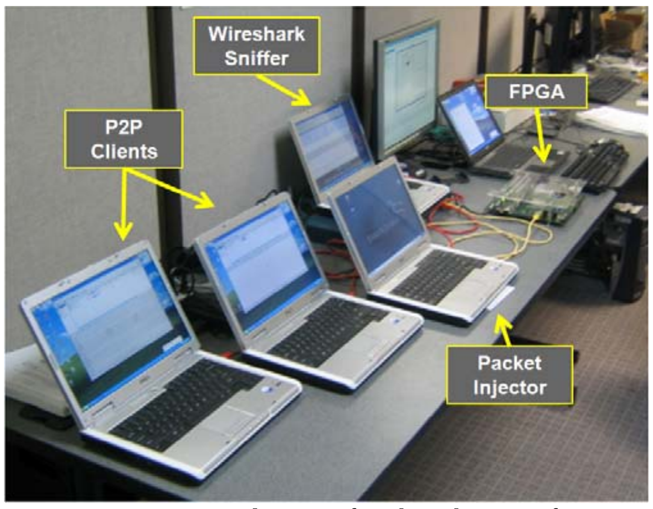

Figure 5. Experimental Setup for the Three Performance Tests

H. Evaluation Techniques

Outlined below are the tests used for collecting data on the *packet processing time* and *probability of packet intercept* metrics for each experiment.

1) Calculating Packet Processing Time: The first test consists of a series of packets sent from the Linux laptop to one of the Windows laptops via the Cisco switch. For each run, a series of 50 crafted packets are sent, and the CPU cycles needed by the system to process each packet is recorded.

A total of 50 identical packets are sent in one second intervals through the network. Based on the testing of several different sample sizes, 50 packets is the minimum sample size that results in sufficiently small confidence intervals to perform a meaningful comparison between system configurations. To ensure the independence of each trial, one second is chosen as the interarrival interval. To determine the number of cycles required to process each packet received by the system, a Power PC System Timer time stamp is taken prior to the beginning of the processing, and another time stamp is taken immediately after the processing routine ends. To compute the number of cycles required to process the packet, the two values are simply subtracted from each other. Since the Power PC processor in the SUT is configured to run at 300 MHz, to convert the processing time from clock cycles to standard time units, the formula (*time = cycles/300*) is used, where *cycles* is the number of cycles as determined by the System Timer and *time* is the time to process the packet in microseconds.

2) Calculating Probability of Intercept Under a Non-P2P Load: The second test consists of a series of packets sent from the Linux laptop to one of the Windows laptops via the Cisco switch. For this test, however, an additional non-P2P traffic load is generated on the network. For this test, the number of crafted packets successfully intercepted and processed by the system is recorded.

For each test, a series of three hundred crafted packets are injected into the network 500 milliseconds apart. By injecting the packets 500 milliseconds apart, the results of each trial (either the packet was captured or not captured) are assured to be independent of each other. Based on the testing of several different sample sizes, 300 packets is a good sample size to produce a binomial distribution that results in sufficiently small confidence intervals to perform a meaningful comparison between system configurations. Again, to ensure the independence of each trial, i.e., to ensure that the system is not processing one crafted packet when another one arrives at the system, the packets are sent 500 milliseconds apart.

3) Calculating Probability of Intercept Under an All-P2P Load: The third test also consists of a series of packets sent from the Linux laptop to one of the Windows laptops via the Cisco switch. As in the first test, a series of crafted packets are sent across the network. However, for this test, the packets are sent as quickly as possible from the Linux laptop using the hping `--flood` switch.

In order to determine how many packets were sent by the Linux laptop, a particular feature of the hping program is exploited. When the hping program sends a series of packets, each packet contains a different source port, and the source port number is incremented by one each time a packet is transmitted. Thus, for a given series of packets, the total number of packets sent in the series can be calculated by subtracting the first packet's source port number from the last packet's source port number.

In this test several thousand packets are sent over the network as quickly as possible using hping and the flood is then terminated manually. To determine the *probability of packet intercept*, the following procedure is used:

1) Inspect the capture log file and record the number of frames successfully captured by the system.
2) Record the source port number of the first packet in the log file, and the source port number of the last packet in the log file.
3) Compute the *probability of packet intercept* using

$$P(packet\ intercept) = \frac{number\ of\ packets\ in\ log}{Port(last\ packet) - Port(first\ packet)} \quad (1)$$

When performing these tests, another important parameter is the network utilization or network load. To determine the minimum overall network load, the Wireshark utility on the laptop that is connected to the second spanning port is used to analyze all traffic sent during the test. At the conclusion of the test, a capture summary is extracted, and the Average MBit/sec value is recorded. Because Wireshark itself may not capture all packets on the network, this value is assumed to be the *minimum* network traffic load.

IV. RESULTS AND ANALYSIS

A. Results and Analysis of Experiment 1 -BitTorrent

1) Test 1: Calculating Packet Processing Time: The first test performed on the system is used to determine how many CPU cycles are required to process each type of packet. Outlined below are the results of the Calculating Packet Processing Time test.

a) Non-P2P Packet Workload: Table I shows the results of a one-variable t-test performed on each of the six configurations using the Non-P2P packet type. The table gives the mean number of CPU cycles required to process the non-P2P packet, the standard deviation, and a 95% confidence interval for the mean. As shown in the table, the number of cycles required ranges from 276 cycles to 1,344 cycles, which equates to a range of 0.92 to 4.48 microseconds per packet.

Table I. Packet Processing Times for Non-BitTorrent Packets

Configuration	Mean	Standard Deviation	Confidence Interval (95%)
Control	1206.00	0.00	(1206.00, 1206.00)
User Alerts	1152.00	0.00	(1152.00, 1152.00)
Dual Buffer	1344.00	109.10	(1313.00, 1375.00)
Packet Write	1146.00	0.00	(1146.00, 1146.00)
Cache	276.00	0.00	(276.00, 276.00)
Combined	303.50	25.76	(296.18, 310.82)

b) BitTorrent Packet Not On the List Workload: Table II shows the results of a one-variable t-test performed on each of the six configurations using the BitTorrent Packet Not On the List packet type. As shown in the table, the number of cycles required ranges from 1,145 cycles to 7,770 cycles, which equates to a range of 3.82 to 25.90 microseconds per packet.

Table II. Packet Processing Times for BitTorrent Packets Not on the List

Configuration	Mean	Standard Deviation	Confidence Interval (95%)
Control	7296	0.00	(7296, 7296)
User Alerts	1044756	730	(1044549, 1044963)
Dual Buffer	7770	0.00	(7770, 7770)
Packet Write	7593	0.00	(7593, 7593)
Cache	1145	0.00	(1145, 1145)
Combined	1205	0.00	(1205, 1205)

c) BitTorrent Packet On the List Workload: Table III shows the results of a one-variable t-test performed on each of the six configurations using the BitTorrent Packet On the List packet type. As shown in the table, the number of cycles required ranges from 3,783 cycles to 118,986 cycles, which equates to a range of 12.61 to 396.62 microseconds per packet.

Table III. Packet Processing Times for BitTorrent Packets on the List

Configuration	Mean	Standard Deviation	Confidence Interval (95%)
Control	116207	22418	(109836, 122578)
User Alerts	1702125	22880	(1695623, 1708628)
Dual Buffer	118986	22391	(112623, 125350)
Packet Write	23292	318	(23202, 23382)
Cache	14679	2064	(14093, 15266)
Combined	3783	75	(3762, 3805)

Analyzing this data, the following observations are made about the results of the first test:

- For the two BitTorrent packet workloads, messages to the user are sent in the User Alert configuration, resulting in a several order of magnitude increase in processing time. This increase is due to the fact that the user alerts are transmitted via serial port at 115,200 baud, which is significantly slower than the 300 MHz processor speed and 100 MHz bus speed used by the FPGA board. Based on this significant increase in *packet processing time*, and the corresponding decrease in overall system performance, all user alerts are eliminated from the final design.
- Adding a second receive buffer results in more CPU cycles required to process a packet, regardless of the type of packet. This is due to the additional processing cycles required to check both receive buffers in order to determine which one contains the next packet to be processed. However, as shown in Section IV.A.2, though this modification increases processing time, the second Ethernet receive buffer also increases the system's overall *probability of packet intercept.*
- As expected, the modification to the packet writing routine only decreases the *packet processing time* when packets are actually written to the log file. For the cases where packets are not written, no significant processing time is gained or lost with this optimization.
- Enabling the caches results in a significant decrease in CPU cycles required to process a packet, regardless of packet type.

2) Test 2: Calculating Probability of Intercept Under a Non-P2P Load: Table IV shows the results of the packet intercept test under a heavy non-P2P network load. For each configuration tested, the number of packets captured out of the 300 sent is shown. The table also shows the *probability of packet intercept* and the corresponding 95% confidence interval for each configuration. In all tests, the total load on the network is measured by the Wireshark packet sniffer to be between 89.6 Mbps and 89.7 Mbps, which equates to an 89.6% load on the 100 Mbps network. However, this measurement is not absolute, as the Wireshark program itself can drop packets under a heavy load. Since it is unknown how many packets were dropped by Wireshark, 89.6% is considered to be the minimum load on the test network.

Table IV. Probability of Packet Intercept Under a Non-P2P Workload

Configuration	Packets Captured (Events)	Packets Sent (Trials)	Probability of Packet Intercept	Confidence Interval (95%)
Control	159	300	0.5300	(0.4718, 0.5876)
User Alerts	166	300	0.5533	(0.4951, 0.6105)
Dual Buffer	292	300	0.9733	(0.9481, 0.9884)
Packet Write	174	300	0.5800	(0.5219, 0.6365)
Cache	289	300	0.9633	(0.9353, 0.9816)
Combined	300	300	1.0000	(0.9901, 1.0000)
Wireshark	298	300	0.9933	(0.9761, 0.9992)

Table IV shows that while the User Alerts and Packet Write configurations capture more packets of interest than the Control (166 and 174 versus 159), the overlapping confidence intervals suggest that the differences are not statistically significant. The table also shows that the Cache and Dual Buffer configurations perform significantly better than the Control. Moreover, the Combined configuration performs better than the other five configurations (300 out of 300 packets captured), returning a test result of 100% *probability of packet intercept* for packets of interest, which is comparable to the 99% capture rate of the Wireshark packet sniffer.

To further determine the statistical significance of these results, hypothesis tests are performed between the various optimizations versus the Control configuration. The p-value for the one-sided test involving the User Alerts and the Control is too high (0.283) to state with confidence that the increase in *probability of packet intercept* is statistically significant. In the one-sided test involving the Packet Write optimization and the Control, again the p-value is too high (0.109) to accept the alternative hypothesis, but it can be inferred that the optimization did provide some improvement to the *probability of packet intercept*. For the Cache, Dual Buffer, and Combined configurations, the p-value for the one-sided test is 0.000, indicating a strong statistical certainty that these configurations are better than the Control configuration.

To determine the overall performance of the Combined configuration, another set of hypothesis tests are performed between the Combined configuration versus the individual optimizations and Wireshark. The p-value for the one-sided tests involving the User Alerts, Packet Write, Cache, and Dual Buffer optimizations ranges between 0.000 and 0.002, indicating a strong statistical certainty that the Combined configuration is better than each individual optimization by itself. When the Combined configuration is compared to the performance of Wireshark, the p-value for the one-sided test is 0.078, which is too high to accept the alternative hypothesis that the Combined configuration performs better than Wireshark, but does indicate that the *probabilities of packet intercept* of the two are comparable.

Analyzing this data, the following observations are made about the results of the second test:

- Adding User Alerts to the system has no statistical impact on the *probability of packet intercept*, positive or negative. However, given the vast increase in *packet processing time* associated with messages sent to the user (5,673.8 microseconds with the User Alerts versus 387.4 without User Alerts), their removal is still justified in the final system design. This point is discussed further in Section IV.C.
- The alternate Packet Writing scheme does not, by itself, significantly improve overall system performance.

However, this optimization, when combined with other improvements, does provide some benefit to the Combined configuration's performance.

- The optimizations that enable the caches and the second receive buffer each have a significant positive impact on overall system performance. For each optimization, system performance increased over 80% from the Control configuration.
- The combination of all three optimizations returned the best performance of any configuration. Even with the additional processing required to analyze all packets on the network for the BitTorrent protocol signature, the system returned similar performance to the dedicated software-based packet sniffer, Wireshark.

3) Test 3: Calculating Probability of Intercept Under an All-P2P Load: Table V shows the results of the packet intercept test under an all-P2P network load. For each configuration tested, the number of BitTorrent Handshake packets that are sent over the network in order for the system to capture 400 of them is shown in the table. The table also shows the *probability of packet intercept* and the corresponding 95% confidence interval for each configuration. In all tests, the total load on the network is measured by the Wireshark packet sniffer to be between 23.35 and 24.10 Mbps, which equates to approximately a 23.3% load on the 100 Mbps network. This is the maximum network throughput possible using the hping utility and the BitTorrent Handshake workload.

Table V shows that the only configuration that performs worse than the Control is the User Alerts configuration (3.4% capture rate for Control versus 1.5% capture rate for User Alerts). The Cache and Dual Buffer configurations perform slightly better than the Control, but still are only able to capture less than 12% of packets sent. The Packet Write configuration performs moderately better than the Control (40.4% versus 3.4% capture rate), but it is still unable to capture more than 1 in 2 packets. Finally, the Combined configuration performs significantly better than the other five configurations (400 out of 400 packets captured), returning a test result of 100% *probability of packet intercept* and comparing very favorably with Wireshark's result of 99.0%.

Table V. Probability of Packet Intercept Under an All-P2P Workload

Configuration	Packets Sent (Trials)	Probability of Packet Intercept	Confidence Interval (95%)
Control	11757	0.0340	(0.0308, 0.0375)
User Alerts	26810	0.0149	(0.0135, 0.0164)
Dual Buffer	9188	0.0435	(0.0395, 0.0479)
Packet Write	990	0.4040	(0.3733, 0.4354)
Cache	3599	0.1111	(0.1011, 0.1219)
Combined	400	1.0000	(0.9925, 1.0000)
Wireshark	404	0.9901	(0.9748, 0.9973)

To further validate the statistical significance of these results, hypothesis tests are performed between the various optimizations versus the Control configuration. The p-value for the one-sided test involving the User Alerts and the Control is 1.000, which corresponds to the fact that the User Alerts configuration actually performed worse than the Control. In the one-sided tests involving the other four optimizations, the p-value for the one-sided test is 0.000, indicating a strong statistical certainty that these configurations have a better *probability of packet intercept* than the Control configuration.

To determine the overall performance of the Combined configuration, another set of hypothesis tests are performed between the Combined configuration versus the individual optimizations and Wireshark. The p-value for the one-sided tests involving the User Alerts, Dual Buffer, Packet Write, and Cache optimizations are all 0.000, indicating a strong statistical certainty that the Combined configuration is better than each individual optimization by itself. When the Combined configuration is compared to the performance of Wireshark, the p-value for the one-sided test is 0.022, which is low enough to accept, with statistical confidence, the hypothesis that *probability of packet intercept* for the Combined configuration is higher than the *probability of packet intercept* using Wireshark.

Analyzing this data, the following observations are made about the results of the third test:

- Adding User Alerts to the system results in a 56% decrease in performance, as measured by *probability of packet intercept*. In this case, the vast increase in *packet processing time* associated with messages sent to the user, discussed above, is almost certainly the root cause of the decrease in performance.
- The Dual Buffer and Cache optimizations, by themselves, modestly improve system performance, but are still unable to capture more than 50% of packets of interest. However, combining them with the Packet Write optimization provides a tremendous benefit to the Combined configuration's performance.
- The alternate Packet Writing scheme by itself provides a moderate improvement to overall system performance. The full benefit of this optimization is seen when combined with caching and an improved Ethernet receive buffer.
- The combination of all three optimizations (Cache, Dual Buffer, and Packet Write) has a synergistic effect on the overall performance of the system when processing back-to-back BitTorrent packets. By themselves, each optimization returned moderate performance gains over the Control configuration. When combined, however, they created a system that is able to achieve a 100% *probability of packet intercept*, which is comparable to the dedicated software packet sniffer, Wireshark.

Overall, the Combined configuration is clearly the best of the possible configurations for the CUT. The Combined configuration consistently returned very low *packet processing times*, indicating that it is able to process a variety of packets faster than any individual optimization. The Combined configuration also returned the highest values in both *probability of packet intercept* tests, indicating that it has a higher probability of intercepting packets of interest in both non-P2P and all-P2P workload environments than any of the other optimizations. Finally, using a 95% confidence interval, the Combined configuration returns a minimum capture rate of 99.0% across all workloads, which is comparable to the performance of Wireshark, which returned a minimum capture rate of 97.5%.

B. Results and Analysis of Experiment 2 -BitTorrent and SIP

1) Test 1: Calculating Packet Processing Time: Table VI shows the results of a one-variable t-test performed on the Optimized (BT + SIP) configuration using six different packet types. The table gives the mean number of CPU cycles required to process the workload packet, the standard deviation, and a 95% confidence interval for the mean. As shown in the table, the number of cycles required ranges from 419 cycles to 34,779 cycles, which equates to a range of 1.40 to 115.93 microseconds per packet, depending on the type of packet.

Table VI. Packet Processing Times for SIP and BitTorrent Packets Using the Expanded System

Configuration	Mean	Standard Deviation	Confidence Interval (95%)
Not P2P	418.6	36.0	(408.4, 428.4)
BT Not on List	1323.5	31.8	(1314.5, 1332.5)
BT Handshake	3883.0	85.6	(3858.7, 3907.4)
SIP Not on List	19450.0	97.8	(19422.2, 19477.8)
SIP BYE	29951.3	224.2	(29887.6, 30015.0)
SIP INVITE	34778.6	226.2	(34714.3, 34842.9)

Analyzing this data, the following observations are made about the results of the first test:

- Adding SIP processing capability to the SUT results in a higher *packet processing time* for both non-P2P and BitTorrent Handshake packets. This is due to the additional processing required by the system to determine if a packet belongs to either the BitTorrent or SIP protocols, as opposed to looking for only BitTorrent packets.
- The *packet processing time* required for any SIP packet is several times longer than the time required to process BitTorrent packets. The reasons for this increase in processing time are explained in Section IV.C.

2) Test 2: Calculating Probability of Intercept Under a Non-P2P Load: Table VII shows the results of the packet intercept test under a heavy non-P2P network load using the modified Optimized (BT + SIP) configuration. For each of the three P2P packet types tested, the number of packets captured by the system out of the 300 sent is shown. For comparison purposes, the number of packets captured by the Wireshark packet sniffer for each workload is also shown. In addition, the table shows the *probability of packet intercept* and the corresponding 95% confidence interval for each workload.

Table VII. Probability of Packet Intercept for BitTorrent and SIP Packets Under a Non-P2P Workload

Workload	Packets Captured (Events)	Probability of Packet Intercept	Confidence Interval (95%)
BT Handshake	300	1.0000	(0.9901, 1.0000)
Wireshark BT	298	0.9933	(0.9761, 0.9992)
SIP BYE	300	1.0000	(0.9901, 1.0000)
Wireshark BYE	300	1.0000	(0.9901, 1.0000)
SIP INVITE	298	0.9933	(0.9761, 0.9992)
Wireshark INVITE	300	1.0000	(0.9901, 1.0000)

The modified Optimized (BT + SIP) configuration performed perfectly (300 out of 300 packets captured) for two out of the three workloads, and returned a 99% *probability of packet intercept* for the other. This performance compares very favorably with the results returned by the Wireshark packet sniffer, which also returned a test result of near-100% *probability of packet intercept* for packets of interest.

Analyzing this data, the following observations are made about the results of the second test:

- The *probability of packet intercept* performance of the system is unchanged when processing BitTorrent Handshake packets. Regardless of whether the SUT is processing a single BitTorrent Handshake packet amid a high non-P2P network load or a steady stream of Handshake packets, the system is able to achieve a 100% *probability of packet intercept*. In addition, the SUT performs slightly better than the Wireshark packet sniffer, regardless of the overall workload.
- When processing SIP BYE packets, the SUT sees a 9% decrease in *probability of packet intercept* when processing the packets back-to-back over processing a single packet amid a high non-P2P network load. The extended *packet processing time* required for this type of packet causes the system to occasionally drop the next frame entering the SUT because it is still processing the current SIP frame in the Ethernet receive buffer. The Wireshark packet sniffer, however, returns perfect scores regardless of workload type.
- When processing SIP INVITE packets, the SUT sees a 33.8% decrease in *probability of packet intercept* when processing the packets back-to-back over processing a single packet amid a high non-P2P network load. The reason for this is the same as that for the SIP BYE packet. However, the SIP INVITE packet, due to its larger overall packet size, requires a longer *packet processing time* than the BYE packet, resulting in a lower *probability of packet intercept* than that of the SIP BYE packet. Wireshark, however, does not suffer from this problem, returning a *probability of packet intercept* of at least 99.7% for both workloads.

3) Test 3: Calculating Probability of Intercept Under an All-P2P Load: For each of the three all-P2P workloads (BitTorrent Handshake, SIP INVITE, and SIP BYE), the total load on the network is measured by the Wireshark packet sniffer, and the results shown in Table VIII. The low maximum network load for the BitTorrent Handshake packet workload, which is also seen in the first experiment, is likely due to the fact that the BitTorrent peer wire protocol runs on top of TCP. Both the exponential backoff mechanism and the reliable data transfer features of TCP add additional time between packets sent over the network, causing a decrease in the maximum throughput that the hping program can achieve. The SIP INVITE and BYE packets, on the other hand, use UDP, which allows hping to achieve a throughput of over 94 Mbps on the 100 Mbps network.

Table VIII. Observed Network Load for Various All-P2P Workloads

Configuration	Network Load (Mbps)
BitTorrent Handshake	23.35
SIP BYE	94.61
SIP INVITE	96.28

Table IX shows the results of the packet intercept test under an all-P2P network load. For each P2P packet type tested, the number of workload packets that were sent over the network in order for the system to capture 400 of them is shown. For comparison purposes, the number of packets captured by the Wireshark packet sniffer for each workload is also shown. In addition, the table shows the *probability of packet intercept* and the corresponding 95% confidence interval for each configuration.

Table IX. Probability of Packet Intercept for BitTorrent and SIP Packets Under an All-P2P Workload

Workload	Packets Sent (Trials)	Probability of Packet Intercept	Confidence Interval (95%)
BT Handshake	400	1.0000	(0.9901, 1.0000)
Wireshark BT	404	0.9901	(0.9748, 0.9973)
SIP BYE	440	0.9091	(0.8783, 0.9343)
Wireshark BYE	400	1.0000	(0.9901, 1.0000)
SIP INVITE	608	0.6579	(0.6187, 0.6956)
Wireshark INVITE	401	0.9975	(0.9862, 0.9999)

Both the SUT and Wireshark perform very well under the BitTorrent Handshake packet type, returning a *probability of packet intercept* of over 99%. For the SIP BYE packet type, the SUT returns a *probability of packet intercept* of just over 90%, while Wireshark returns a perfect score of 100%. Finally, for the SIP INVITE packet type, the SUT achieves a 65.8% *probability of packet intercept*, while Wireshark performs much better, returning a near-perfect score of 99.8%.

C. Overall Analysis

1) Analysis of Packet Processing Time: The first step in the research methodology is to find a system configuration that requires the minimum number of CPU cycles to process packets entering the system. Based on the results presented here, the most significant improvement to system speed occurs when the data and instruction caches are enabled for the Power PC processor. By allowing the FPGA to cache both processor instructions and heap and stack data, packet processing time is reduced by 77% to 87%, depending on the type of packet. In addition, by delaying the compact flash write operations until after the termination of system processing, the *packet processing time* is reduced by 80% for packets written to the log file. When all four optimizations are combined, the resulting Combined configuration achieves a 75% to 92% reduction in processing time of packets of interest over the Control configuration, depending on the type of packet. Therefore, the Combined configuration is confirmed to be the best system configuration for minimizing the overall *packet processing time* for all packets entering the system.

When the ability to process SIP packets is added to the system, the mean *packet processing time* required to process non-P2P packets increases by 115 cycles (0.38 microseconds) and the time required to process BitTorrent Handshake packets increases by 100 cycles (0.33 microseconds). This increase in *packet processing time* is due to the additional software code required to check each packet for the signature of a SIP control packet as well as the signature of a BitTorrent Handshake packet.

2) Analysis of Probability of Packet Intercept Under Load: In the first experiment, the overall goal is to find the configuration that returns the highest *probability of packet intercept* for both non-P2P and all-P2P workloads. In the non-P2P case, where a single BitTorrent packet is sent while the network is under a heavy NETBIOS file transfer load, the Dual Buffer optimization returns a capture rate of over 95%, while the single buffer configurations (with the exception of the Cache configuration, discussed below) all return capture rates of less than 60%. This significant packet loss rate for the single receive buffer configurations is likely due to the inability of a non-P2P frame to be processed and cleared from the buffer before the BitTorrent Handshake packet arrives at the system. At 100 Mbps, the mandatory inter-frame gap required by the Ethernet protocol results in a 0.96 microsecond delay between the end of one frame and the beginning of the next. Since the system processes instructions at 300 MHz, it is able to perform at most 300 instructions per microsecond. Therefore, because multiple instructions are required to transfer data from the Ethernet buffer, read the payload contents, and analyze the data, the system cannot keep up with the data flow, resulting in significant packet loss as the system approaches 100% utilization. However, note that the Cache optimization is the exception to this observation; even with a single buffer, enabling the caches results in a capture rate of 96%. This is likely due to the fact that the extremely small processing times provided by the cache enable a packet to be processed in the short interframe time gap.

Adding a second receive buffer to the Ethernet controller dramatically increases the probability of packet intercept under a non-P2P workload, achieving a 97% capture rate even with no other optimizations incorporated. The use of two receive buffers allows a non-P2P packet to be processed from one buffer while the BitTorrent Handshake packet is being received in the second buffer. Specifically, the additional buffer provides a minimum of 576 additional bit times ((7-byte preamble + 1-byte delimiter + 64-byte minimum frame size) x 8 bits per byte) for the processing of the non-P2P frame over the single buffer option [32]. Although this improvement comes at the cost of additional processing cycles, the expanded processing window provided by the second buffer more than offsets the cost in individual *packet processing times*. When combined with caching and the improved packet writing scheme, the infrequency of packets of interest, and the small likelihood of traffic saturation on the network link, the final Combined configuration allows the system to successfully capture and process all BitTorrent packets of interest sent into a network with a high non-P2P traffic load.

For the situation where BitTorrent Handshake packets are sent to the system back-to-back, as in the case of the all-P2P workload, each system optimization returns a much different *probability of packet intercept*. Using this workload, only the Packet Write optimization results in a greater than 40% *probability of packet intercept*; the other three optimizations all return capture rates of less than 15%. These low capture rates are due to the increased *packet processing time* required for BitTorrent packets over non-P2P packets. When all four optimizations are used together, the resulting Combined configuration is able to achieve a 100% *probability of packet intercept* for BitTorrent Handshake packets that are received back-to-back by the SUT, which is comparable to the results for the dedicated Wireshark packet sniffer.

When the ability to process SIP packets is added to the system, the overall system performance is unchanged when processing single packets of interest under a high non-P2P network load. Regardless of P2P packet type (BitTorrent Handshake, SIP INVITE, or SIP BYE), the system returns at least a 97.5% *probability of packet intercept*. This performance is comparable to that of the Wireshark packet sniffer, which also returned a minimum 97.5% *probability of packet intercept*.

However, when the system is tasked with processing back--to-back P2P packets arriving at near-line speed, the system's performance depends greatly on the type of P2P packets arriving at the SUT. For the BitTorrent Handshake packet workload, the system still returns a *probability of packet* *intercept* of 100%, which is unchanged from the non-SIP processing system. When processing back-to-back SIP BYE packets, the *probability of packet intercept* drops to 90.9%, and when processing back-to-back SIP INVITE packets, the *probability of packet intercept* drops further to 65.8%. These results are expected, since the *packet processing time* of a SIP BYE packet is much greater than that of a BitTorrent Handshake packet, and the *packet processing time* of a SIP INVITE packet is greater than that of a SIP BYE packet. Thus, for the case where the system receives back-to-back packets of interest, the probability of successfully intercepting both packets is at least 90% for BitTorrent Handshake and SIP BYE packets, and is less than two-in-three for SIP INVITE packets. In comparison, the Wireshark packet sniffer achieved a *probability of packet intercept* of greater than 99% for all three packet types, which is likely due to the fact that Wireshark does not perform any extraction or comparison of payload data in the frames it collects.

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions of Research

This paper presents an optimized FPGA-based system that analyzes traffic on a network in real time, detects selected P2P protocols, compares the digital information being shared against a list of interest, and in the case of a match, records selected control frames from the P2P session in a log file. With one exception, across all types of P2P packets tested, the system captures packets of interest with an intercept probability of at least 97.6% under an 89.6 Mbps network load. In the rare case where the system is processing packets at a rate of over 94 Mbps, and it receives two SIP control packets in a row, the intercept probability decreases to 66%.

Optimizations to the system return varying changes to the *probability of packet intercept* for packets of interest, ranging from an increase of 4.4% for the User Alerts to an increase of 89% for the Combined configuration over the original system design, further justifying the Combined configuration's use for the final optimized design. Using the Combined configuration, the optimized system is able to capture a packet of interest with a *probability of packet intercept* of at least 99.0%, using a 95% confidence interval and given an 89.6 Mbps network utilization.

B. Significance of Research

This research provides network administrators with a unique method of detecting and tracking both illicit file sharing and VoIP phone call patterns. This system differs from other methods of tracking illicit file sharing in that it is completely passive, meaning the system transmits absolutely no information into the network being monitored, making it completely invisible to users of the network. By designing the system to be completely self-contained on an FPGA, the TRAPP system can be easily and inexpensively implemented on any LAN. The simplicity of the system enables it to run at very high speeds, even when monitoring a heavily utilized network. Because the tool operates on a spanning port of the network gateway, any failure of the TRAPP system will have no negative impact on the network's performance. Finally, the system can be easily expanded to include additional P2P protocols with minimum impact on overall system performance.

C. Recommendations for Future Research

The next logical step for this research is to determine how the system performs on a more robust network. Specifically, the TRAPP system should next be tested on a gigabit Ethernet network using a more advanced FPGA board that contains both a faster processor and a gigabit Ethernet controller, such as the Virtex 5-series FPGA [33]. Another area for future research is addressing the encryption and obfuscation capabilities of P2P networks. The next version of TRAPP should be able to detect P2P control packets that have been obfuscated or encrypted, and the relevant data extracted. Future research should also investigate how much larger data sets affect the overall performance of the TRAPP system.

REFERENCES

[1] "Navy Releases New Information On Presidential Security Leak," March 2009, http://www.wpxi.com/news/18818589/detail.html.

[2] United States Department of Justice, "RCFL Program Annual Report for Fiscal Year 2007," 2008, http://www.rcfl.gov/downloads/documents/RCFL Nat Annual07.pdf.

[3] Government Accounting Office, "File Sharing Programs: The Use of Peer-to-Peer Networks to Access Pornography," May 2005, http://www.gao.gov/new.items/d05634.pdf.

[4] G. Owen, "Taliban Using Skype Phones to Dodge MI6," September 2008, http://www.dailymail.co.uk/news/worldnews/article-1055611/Taliban-using-Skype-phones-dodge-MI6.html.

[5] J. Kahn, "Mumbai Terrorists Relied on New Technology for Attacks," December 2008, http://www.nytimes.com/2008/12/09/world/asia/09mumbai.html? r=1.

[6] R. Badonnel, R. State, I. Chrisment, and O. Festor, "A Management Platform for Tracking Cyber Predators In Peer-to-Peer Networks," *Proceedings of the Second International Conference on Internet Monitoring and Protection*, p. 11, 2007.

[7] F. Adelstein and R. A. Joyce, "File Marshal: Automatic Extraction of Peer-to-Peer Data," *Digital Investigation*, vol. 4, no. Supplement 1, pp. 43–48, September 2007.

[8] K. P. Chow, K. Y. Cheng, L. Y. Man, P. K. Y. Lai, L. C. K. Hui, C. F. Chong, K. H. Pun, W. W. Tsang, H. W. Chan, and S. M. Yiu, "BTM An Automated Rule-Based BT Monitoring System for Piracy Detection," *Proceedings of the Second International Conference on Internet Monitoring and Protection*, p. 2, 2007.

[9] B. Dedman and B. Sullivan, "GFR/CopyRouter Process Flow," October 2008, http://msnbcmedia.msn.com/i/msnbc/Sections/NEWS/PDFs/ 081016 copyrouter.pdf.

[10] P. Gil, ""Peer Guardian" Firewall: Keep Your P2P Private," January 2008, http://netforbeginners.about.com/od/peersharing/a/peerguardian.htm.

[11] R. MacManus, "The Underground World of Private P2P Networks," August 2006, http://www.readwriteweb.com/archives/private p2p.php.

[12] B. Dedman and B. Sullivan, "ISPs are Pressed to Become Child Porn Cops," October 2008, http://www.msnbc.msn.com/id/27198621.

[13] B. Cohen, "The BitTorrent Protocol Specification," February 2008, http://www.bittorrent.org/beps/bep 0003.html.

[14] B. Cohen, "Incentives Build Robustness in BitTorrent," May 2003, http://www.bittorrent.org/bittorrentecon.pdf.

[15] "FIPS 180-1 -Secure Hash Standard," May 1993, http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[16] Ubiquity, "Understanding SIP," July 2008, http://www.sipcenter.com/sip.nsf/html/WEBB5YNVK8/$File/ Ubiquity SIP Overview.pdf.

[17] "RFC 3261 -SIP: Session Initiation Protocol," June 2002, http://www.faqs.org/rfcs/rfc3261.html.

[18] Cisco, "Cisco Introduces New SIP-enabled Voice over IP Solutions," March 2002, http://newsroom.cisco.com/dlls/prod 031102.html.

[19] CounterPath Corporation, "Xten Softphone SDK Delivers PC-to-PC VoIP in New Yahoo! Messenger," June 2005, http://www.counterpath.com/xten-softphone-sdk-delivers-pc-to-pc-voip-in-new-yahoo-messenger.html. [20] Google, "Google Talk for Developers," October 2008, http://code.google.com/apis/talk/open communications.html.

[21] Xilinx, "Xilinx University Program Virtex-II Pro Development System," June 2008, http://www.xilinx.com/products/devkits/XUPV2P.htm.

[22] "BitTorrent Protocol Specification v1.0," May 2008, http://wiki.theory.org/BitTorrentSpecification.

[23] Rice University WARP Project, “Wireless Open-Access Research Platform,” June 2006, http://warp.rice.edu/trac/browser/PlatformSupport/WARPMAC/warpmac.c.
[24] µTorrent, “µTorrent -The Lightweight and Efficient BitTorrent Client,” June 2008, http://www.utorrent.com/.
[25] CounterPath Corporation, “X-Lite VoIP Softphone,” September 2008, http://www.counterpath.com/x-lite.html&active=4.
[26] Remote-Exploit, “BackTrack,” June 2008, http://www.remote-exploit.org/backtrack download.html.
[27] hping, “Hping -Active Network Security Tool,” July 2008, http://www.hping.org/.
[28] VMWare, “VMware Player,” September 2008, http://www.vmware.com/products/player/.
[29] Trixbox, “Trixbox, an Asterisk-based PBX Phone System,” September 2008, http://www.trixbox.org/.
[30] Wireshark, “Wireshark Network Protocol Analyzer,” July 2008, http://www.wireshark.org/.
[31] Tera Term Pro, “Tera Term Pro Terminal Emulator,” July 2008, http://hp.vector.co.jp/authors/VA002416/teraterm.html.
[32] IEEE, “IEEE Standard 802.3,” December 2005, http://standards.ieee.org.
[33] Xilinx, “Virtex-5 Family Overview,” August 2008, http://www.xilinx.com/support/documentation/data sheets/ds100.pdf.