

CHARACTERIZING THE EFFECTS OF DEVICE COMPONENTS
ON NETWORK TRAFFIC

A Thesis
Presented to
The Academic Faculty

by

Supreeth Sathyanarayana

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2013

Copyright © 2013 by Supreeth Sathyanarayana

CHARACTERIZING THE EFFECTS OF DEVICE COMPONENTS
ON NETWORK TRAFFIC

Approved by:

Dr. Raheem A. Beyah
Advisor and Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Douglas M. Blough
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yusun Chang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 1 April 2013

To my parents, my reason.

ACKNOWLEDGEMENTS

I would like to express my immense gratitude towards Dr. Raheem A. Beyah, my advisor and mentor. He has had the biggest influence on me here at Georgia Tech, and I am forever indebted to him for his steady guidance and unwavering patience. I will always cherish the opportunity to work with him and carry forward the scientific hunger he instills in all around him.

My sincere thanks to Dr. Selcuk Uluagac, who has been an inspirational leader, a perfectionist we all strive to be, and a dear friend. The care and concern he has shown me is heart-warming and something I will never forget. I must also thank him for introducing me to the magical world of Linux! I would also like to thank Dr. Douglas M. Blough and Dr. Yusun Chang for kindly agreeing to serve on my thesis reading committee, and for their helpful feedback.

My heartfelt thanks goes out to my Indian coterie of Sakthi, Ram, and Venkat, whose constant companionship and camaraderie made Georgia Tech a home away from home. Special thanks to Sakthi, working with whom has been a pleasure. I would also like to thank all my other colleagues at the Communications Assurance and Performance group for their support.

Last, but certainly not the least, I would like to acknowledge the pivotal role my family, relatives, and friends played in this two-year voyage with their support, strength, wishes, and encouragement.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
SUMMARY	xi
I INTRODUCTION	1
1.1 Research Objectives	2
1.2 Summary of Contributions	2
1.2.1 Traffic Measurement and Characterization	2
1.2.2 Counterfeit Detection: An Application	2
1.3 Outline	3
II TRAFFIC MEASUREMENT AND CHARACTERIZATION	4
2.1 Motivation and Related Work	4
2.2 Testbed	5
2.2.1 FPGAs	5
2.2.2 Monitor System	7
2.3 Technique	8
2.4 Experiments	9
2.4.1 CPU Clock	9
2.4.2 CPU Data Cache	13
2.4.3 CPU Instruction Cache	14
2.4.4 RAM	16
2.5 Discussion	17
2.6 Conclusion	18
III COUNTERFEIT DETECTION: AN APPLICATION	19
3.1 Motivation and Related Work	19
3.2 Testbed and Technique	20
3.2.1 Neural Networks	21

3.3	Tests conducted	22
3.3.1	CPU	22
3.3.2	RAM	32
3.4	Discussion, Limitations, and Conclusion	33
IV	CONCLUSION	35
4.1	Future Work	36
APPENDIX A	— CHARACTERIZATION	37
APPENDIX B	— COUNTERFEIT DETECTION	49
REFERENCES	54

LIST OF TABLES

1	Recall values for i3 and i7 for various traffic types	25
2	Recall values for 6GB and 2GB RAM for various traffic types	33

LIST OF FIGURES

1	Testbed	6
2	Pender GR-XC3S-1500 Development Board	7
3	Xilinx XtremeDSP Starter Platform - Spartan-3A DSP 1800A Edition . . .	8
4	NetFPGA 1G	8
5	PDFs of UDP IATs for clock variations on XC3S1500 with 1000 bins of $10\mu s$ width	10
6	PDFs of TCP IATs for clock variations on XC3S1500 with 1000 bins of $10\mu s$ width	11
7	PDFs of ICMP(56 bytes) reply IATs for clock variations on XtremeDSP with 1000 bins of $10\mu s$ width	12
8	PDFs of ICMP(1400 bytes) reply IATs for clock variations on XC3S1500 with 1000 bins of $10\mu s$ width	12
9	PDFs of ICMP (56 bytes) reply IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu s$ width	13
10	PDFs of ICMP (1400 bytes) reply IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu s$ width	14
11	PDFs of ICMP (56 bytes) reply IATs for instruction cache variations on XtremeDSP with 1000 bins of $10\mu s$ width	15
12	PDFs of UDP IATs for instruction cache variations on XC3S1500 with 1000 bins of $10\mu s$ width	15
13	PDFs of ICMP (56 byte) reply IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu s$ width	16
14	PDFs of UDP IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu s$ width	17
15	PDFs of TCP IATs for CPU variations on Optiplex 7010 with 1000000 bins of 1ns width	22
16	Processor changed on the PC motherboard	23
17	PDFs of ICMP (56 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width	24
18	PDFs of TCP IATs for CPU variations on Optiplex 7010 with 1000000 bins of 1ns width	24
19	PDFs of ICMP request (56 bytes) IATs for different i3s and i7s on Optiplex 7010 with 10000 bins of $1\mu s$ width	26

20	PDFs of ICMP request (56 bytes) IATs for different i3s and i7s on Optiplex 7010 with 10000 bins of $1\mu\text{s}$ width captured using a NIC	28
21	Testbed for WAN setting	29
22	PDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on WAN with 100000 bins of 50ns width	30
23	PDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on LAN with 100000 bins of 50ns width	31
24	CDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on WAN and LAN	31
25	Traceroute output for path between DUT and VAIO when on different networks	32
26	PDFs of UDP IATs for RAM variations on Optiplex 7010 with 1000 bins of $1\mu\text{s}$ width	32
27	PDFs of UDP IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	37
28	PDFs of TCP IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	38
29	PDFs of ICMP(56 bytes) IATs for clock variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	38
30	PDFs of ICMP(1400) IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	39
31	PDFs of UDP IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	39
32	PDFs of UDP IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	40
33	PDFs of TCP IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	40
34	PDFs of TCP IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	41
35	PDFs of ICMP reply IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	41
36	PDFs of ICMP (1400 bytes) reply IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	42
37	PDFs of UDP IATs for instruction cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	42
38	PDFs of TCP IATs for instruction cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	43

39	PDFs of TCP IATs for instruction cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	43
40	PDFs of ICMP (56 bytes) reply IATs for instruction cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	44
41	PDFs of UDP IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	44
42	PDFs of TCP IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	45
43	PDFs of TCP IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	45
44	PDFs of ICMP (56 byte) reply IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	46
45	PDFs of ICMP (1400 byte) reply IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width	47
46	PDFs of ICMP (1400 byte) reply IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width	48
47	PDFs of UDP IATs for CPU variations on Optiplex 7010 with 1000 bins of $1\mu\text{s}$ width	49
48	PDFs of ICMP (1400 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width	50
49	PDFs of ICMP (56 bytes) request IATs for CPU variations on Optiplex 7010 with 1000 bins of $2\mu\text{s}$ width	50
50	PDFs of ICMP (1400 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width	51
51	PDFs of TCP IATs for RAM variations on Optiplex 7010 with 1000000 bins of 1ns width	51
52	PDFs of ICMP (56 bytes) request IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width	52
53	PDFs of ICMP (1400 bytes) request IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width	52
54	PDFs of ICMP (56 bytes) reply IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width	53
55	PDFs of ICMP (1400 bytes) reply IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width	53

SUMMARY

When a network packet is formed by a computer's protocol stack, there are many components (e.g., Memory, CPU, etc.) of the computer that are involved in the process. The objective of this research is to identify, characterize and analyze the effects of the various components of a device (e.g., Memory, CPU, etc.) on the device's network traffic by measuring the changes in its network traffic with changes in its components. We also show how this characterization can be used to effectively perform counterfeit detection of devices which have counterfeit components (e.g., Memory, CPU, etc.).

To obtain this characterization, we measure and apply statistical analyses like probability distribution functions (PDFs) on the interarrival times (IATs) of the device's network packets (e.g., ICMP, UDP, TCP, etc.). The device is then modified by changing just one component (e.g., Memory, CPU, etc.) at a time while holding the rest constant and acquiring the IATs again. This, over many such iterations provides an understanding of the effect of each component on the overall device IAT statistics. Such statistics are captured for devices (e.g., field-programmable gate arrays (FPGAs) and personal computers (PCs)) of different types. Some of these statistics remain stable across different IAT captures for the same device and differ for different devices (completely different devices or even the same device with its components changed). Hence, these statistical variations can be used to detect changes in a device's composition, which lends itself well to counterfeit detection.

Counterfeit devices are abundant in today's world and cause billions of dollars of loss in revenue. Device components are substituted with inferior quality components or are replaced by lower capacity components. Armed with our understanding of the effects of various device components on the device's network traffic, we show how such substitutions or alterations of legitimate device components can be detected and hence perform effective counterfeit detection by statistically analyzing the deviation of the device's IATs from that of the original legitimate device. We perform such counterfeit detection experiments on

various types of device configurations (e.g., PC with changed CPU, RAM, etc.) to prove the technique's efficacy. Since this technique is a fully network-based solution, it is also a non-destructive technique which can quickly, inexpensively and easily verify the device's legitimacy. This research also discusses the limitations of network-based counterfeit detection.

CHAPTER I

INTRODUCTION

With the explosion of the Internet, there have been many works and measurement studies that look at network traffic dynamics [1, 2]. Specifically, fingerprinting works like [3, 4, 5] have techniques that rely on traffic variations and packet IATs. Despite so much focus on network traffic dynamics, none of the prior works explain or delve into the hardware components' effects on the traffic. An understanding of such effects can perhaps be used to find weaknesses or prove the robustness of techniques that depend on traffic patterns. Hence, there is a definite need to understand what contributes to the network traffic patterns.

The network packet creation process is complicated and many-part. Each packet is formed of multiple layers wrapped one within another [6]. Many components of the computer (e.g., CPU, cache, RAM, etc.) are involved in the packet forming, and it is intuitive to think that each of them will have an influence on how the packet is formed. The author of [6] has in fact shown how a 'slow' receiver affects the traffic dynamics differently than a 'fast' receiver. This hints at a certain dependence of internal hardware architecture on the packet delays. A study and characterization of the effects of internal device components on the IATs of packets can provide an understanding of network traffic dynamics.

Counterfeiting is the forging or imitating of a legitimate device by an illegitimate device, usually of lower capacity or quality so as to afford the seller a higher profit margin through a lower manufacturing cost. Counterfeiting of devices is running rampant in the industry today [7] and is causing immense loss in revenues to companies [8] to the tune of billions of dollars. It is also causing security risks and vulnerabilities in critical military systems [7]. Governments and companies are in turn spending money, time and resources to detect, replace and combat these counterfeits [9, 8]. Thus, counterfeit devices are widespread and counterfeit detection is of very high importance today.

There are as many counterfeit detection techniques as there are counterfeiting techniques

themselves [8, 10, 11]. However, most techniques require specialized and expensive hardware, or have complicated procedures, or are destructive to the device under test (DUT), or are too slow to be practical in real-world scenarios where a large number of devices are being checked. Hence, there is a need for a broad-scoped, simple, non-destructive, and cheap counterfeit detection technique which can quickly and effectively differentiate counterfeit devices from legitimate ones.

1.1 Research Objectives

The objective of this thesis is to: (1) characterize the effects of a device's different components on its network traffic; and (2) show how the characterization can be applied to counterfeit detection of illegitimate devices and also discuss some limitations of network-based counterfeit detection.

1.2 Summary of Contributions

This thesis is presented in two parts as discussed below:

1.2.1 Traffic Measurement and Characterization

We build and emulate different configurations of a Linux-based networked system on different FPGA boards. The configurations are varied based on CPU clock speed, CPU cache and RAM capacity. For each configuration, we use a NetFPGA [12] to accurately capture ICMP, TCP, and UDP traffic and measure the packet IATs. These IAT measures are then subjected to various statistical analyses and are compared to the IAT statistics of the other configurations. With such an analysis, we bring out the inherent differences in the effects the different device components have on the network traffic.

1.2.2 Counterfeit Detection: An Application

Armed with the knowledge obtained from the previous part, we are able to show that the device characterization's statistical techniques can be used to effectively identify counterfeit devices. We do this by experimenting with a real computer system. ICMP, TCP, and UDP traffic are captured from the computer and the statistical analysis is performed on the packet IATs as before. The computer's CPU and RAM are then replaced one at a

time by cheaper alternatives and the experiments are repeated. We then show how the IAT statistics are in fact affected and hence detectable: a means of counterfeit detection. We also discuss the limitations of such network-based counterfeit detection.

1.3 Outline

The rest of the thesis is organized as follows: Chapter 2 discusses the traffic measurement and general characterization of system configurations emulated on FPGAs; Chapter 3 discusses the experiments with a real computer and the application of the characterization to counterfeit detection as well as its limitations; and Chapter 4 forms the conclusion of this thesis, after drawing inferences from results obtained, and discussing future work.

CHAPTER II

TRAFFIC MEASUREMENT AND CHARACTERIZATION

2.1 Motivation and Related Work

Since the dawn of the Internet, there has been an increasing focus on networks in the academic community as well as in industry. Today, the Internet is undeniably all around us. The increasing popularity of networked, distributed, and mobile devices only reaffirms the ubiquitous nature of networks. Nonstop mobile connectivity is the rage. There is a mass exodus of users from powerful, stand-alone machines towards the unmatched power of networked distributed systems and cloud computing infrastructures. The introduction of distributed systems has also brought network management complexities [13]. Being at the center of all this attention has led to the generation of a plethora of measurement works and studies on network traffic [1, 14, 15]. Furthermore, fingerprinting works like [3, 4, 5] analyze the network traffic and packet IATs to effectively fingerprint devices, while works like [16, 17] analyze the network traffic IATs to determine system resource utilization. Given that there is so much focus on network traffic dynamics, it stands to reason that there is a necessity for a sound characterization of all the facets that affect network traffic. Such a characterization will help better understand, test for weaknesses of, formulate attacks for, or prove the impregnability of the techniques described above. Surprisingly however, a characterization based on the hardware of the system itself is lacking. In [5], the authors try to explain the IAT variations as being resultant of softwares running on the system, transmission rates, and wireless access methods; however, they fail to recognize the effects of hardware components of the device on the IATs. The authors of [16, 17] even go to the extent of using IATs to analyze hardware usage, but do not actually characterize the effects of different hardware components themselves on the network traffic.

Perhaps the earliest inference of a dependence of network traffic on hardware components comes from [6]. Here, the author discusses the difference in packet delays between a normal

receiver and a slow receiver, thereby hinting that the packet IATs are not just dependent on transmission rates, but also on the underlying system hardware. This is expected since a system is a complex conglomeration of various components (e.g., CPU, cache, RAM, etc.), each of which has a unique role to play in the construction of a multi-layered network packet, which is a complex process in itself. Hence, the different components uniquely affect the rate at which the network packet is generated, thereby adding their own idiosyncratic delays into the packet. This is the basic essence of this thesis, and in the following sections, we show how we can characterize the device components' effects on network traffic based on network packet IATs.

2.2 Testbed

We set out with the strategy to measure network traffic from different computer systems that differed from each other in one component only. Hence, it made sense to emulate these devices on FPGAs, which provide high reconfigurability while also providing a faithful representation of the idiosyncrasies of real device components. These emulated devices were then loaded with a variant of Linux and subjected to a battery of network traffic tests both active (ICMP) and passive (TCP, UDP). A monitor node captured the traffic from the DUT via a nTAP as shown in Figure 1. The sender/receiver (Sony VAIO laptop) was used to generate traffic to the DUT for active (ICMP echo requests to the device) tests and to receive the DUT's traffic for passive (TCP and UDP traffic from the device) tests. The following subsections discuss the various parts of the testbed in more detail.

2.2.1 FPGAs

2.2.1.1 Pender GR-XC3S-1500 Development Board

The GR-XC3S-1500 Development Board [18] shown in Figure 2 features a Xilinx Spartan3 XC3S1500-FG456 FPGA, 64MB SDRAM of on-board memory, Ethernet 10/100 Mbit/s MAC and PHY (LXT971A), a JTAG programming and configuration port, and 25MHz and 50MHz on-board oscillators.

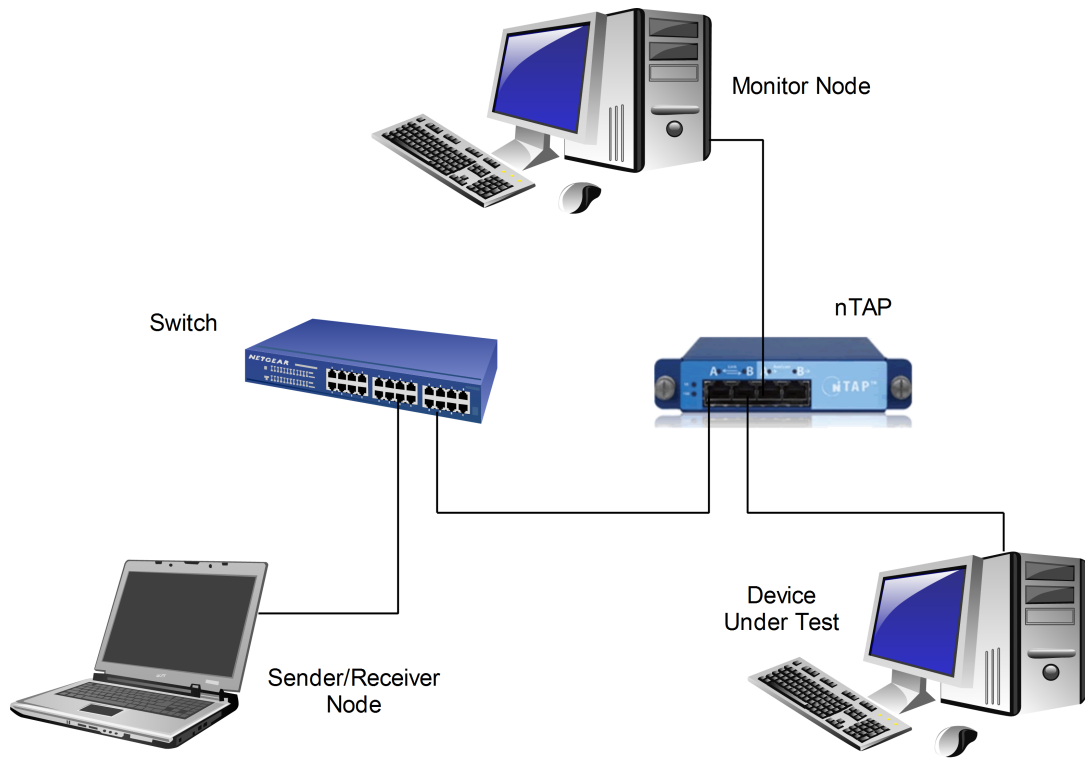


Figure 1: Testbed

2.2.1.2 Xilinx XtremeDSP Starter Platform - Spartan-3A DSP 1800A Edition

The XtremeDSP Starter Platform [19] shown in Figure 3 features a Xilinx 3SD1800A-FG676 FPGA, 128MB DDR2 SDRAM of on-board memory, Ethernet 10/100/1000 Mbit PHY, a JTAG programming and configuration port, and a 125MHz LVTTTL SMT on-board oscillator.

2.2.1.3 Leon3 Processor

The Leon3 [20] is a synthesisable VHDL model of a SPARC-V8 based 32-bit processor. It has a fully pipelined IEEE-754 floating-point unit (FPU); hardware multiply, divide and MAC units; and highly configurable, separate data and instruction cache (Harvard Architecture). Its VHDL model source code is distributed as part of the GRLIB IP library and is freely available under the GNU GPL license. The GRLIB IP is configured and downloaded onto the FPGAs using Xilinx ISE for each device configuration using a JTAG cable.

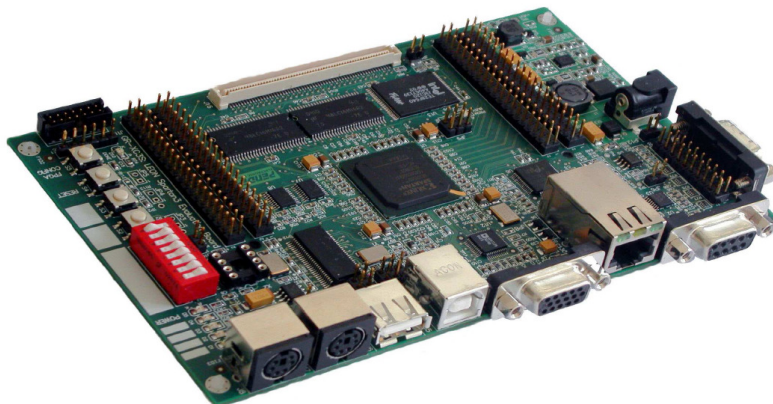


Figure 2: Pender GR-XC3S-1500 Development Board

2.2.1.4 *SnapGear*

A special Leon version of the SnapGear Embedded Linux distribution provided by Aeroflex Gaisler is installed on the Leon3-based system synthesized on the FPGAs. This provides a simple yet potent version of Linux for use on the emulated systems. It includes all the basic functionalities of a Linux system, 10/100/1000 Ethernet networking support, BusyBox [21] utilities, etc. It is configured and downloaded onto the Leon3-based system in the FPGAs using a JTAG cable and the GRMON debugger [22].

2.2.2 Monitor System

The monitor node runs all the necessary scripts to control the Sony VAIO and the DUT in an automated fashion. It controls the sending of ICMP requests from the VAIO to the DUT, and the receiving of TCP and UDP packets from the DUT at the VAIO. All packets leaving the DUT are captured via the nTAP by a NetFPGA 1G [12] installed in the monitor system. The NetFPGA, shown in Figure 4, a reconfigurable hardware platform for high-speed networking, loaded with the packet generator program, provides accurate hardware timestamping of up to 8ns with its 125MHz core clock. Regular network interface controllers (NICs) interrupt the system for each packet that comes in and these interrupts are serviced either instantly or in groups by the operating system (OS), which then timestamps the packets using the system clock. This leads to less precise timestamps. Once the packets are captured and timestamped, the monitor filters the packets using tshark, extracts the IATs

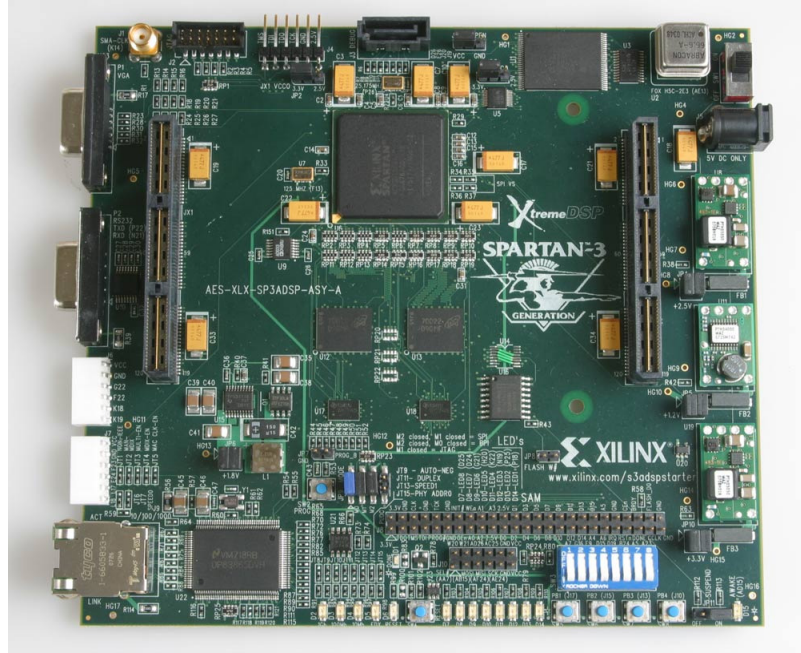


Figure 3: Xilinx XtremeDSP Starter Platform - Spartan-3A DSP 1800A Edition

using Matshark [23] in MATLAB, and processes them statistically.

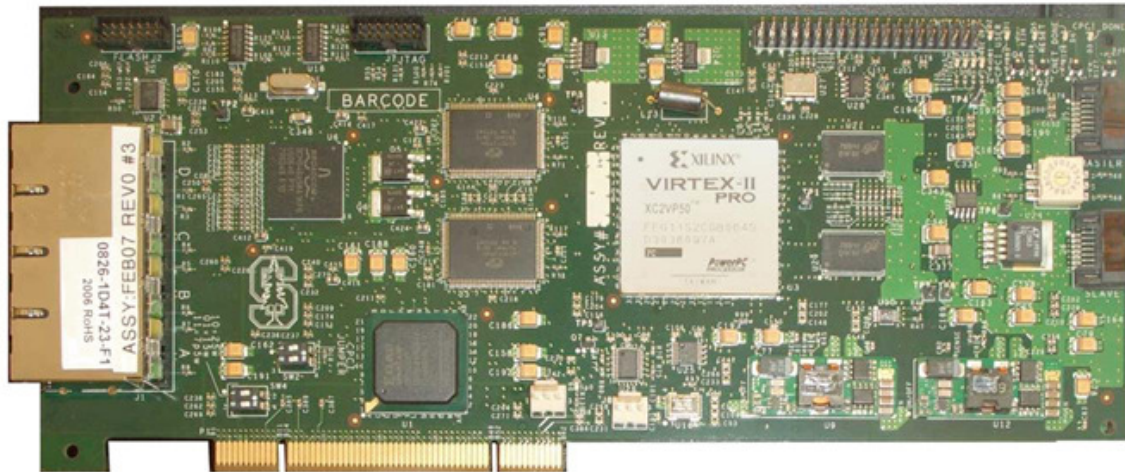


Figure 4: NetFPGA 1G

2.3 Technique

We captured four 15 minute captures each, for ICMP replies (Ping), TCP (Iperf), and UDP (Iperf) packets received from the DUT, for each FPGA configuration. The ICMP (56 and 1400 byte data sizes) requests were made every 0.001s, and the UDP (56 data bytes) traffic

from the DUT was sent at 1Mbps. TCP had no rate control. The FPGA configurations differed by just one component at a time. The components varied were, CPU clock speed, CPU data cache replacement policy, CPU instruction cache replacement policy, and RAM size. The entire experiment was done for both the FPGA boards.

The captured packet IATs were extracted and statistically processed, and we discuss them in the next section. While we discuss the most relevant results in the following section, all combinations not discussed below are listed in APPENDIX A.

2.4 Experiments

Here, we discuss the results and statistics based on the IATs obtained for the different configuration scenarios. We discuss the results obtained for different traffic types for each configuration on both the FPGA boards. The default design we emulated was a system having, a Leon3 processor with MMU, 40MHz clock, 64MB RAM for XC3S1500 and 128MB RAM for XtremeDSP, 8KB data cache, 4KB instruction cache, and running SnapGear Linux.

2.4.1 CPU Clock

After capturing each type of traffic, we changed the configuration by varying the clock speeds and repeated the traffic captures for each clock speed, for each board.

UDP packets of 56 data bytes each were generated on the DUT using Iperf at 1Mbps. Figure 5 shows the variations in the PDFs of the IATs for clock speeds of 20MHz, 25MHz, 33MHz, and 40MHz on the XC3S1500 board. The PDF histograms were plotted with 1000 bins of $10\mu\text{s}$ width. Each clock speed has four 15 minute captures, whose PDFs are shown with a common color in the figure. As we can see, the four captures for each clock speed overlap with each other and are consistent. This shows the repeatability of the experiment. There is a lateral shift between the peaks of the different clock speeds showing that increasing clock speed decreases the mean IATs. There is also a noticeable decrease in the heights of the PDF peaks with decreasing clock speeds, which leads to more spread of the IAT values along the x-axis for the lower clock speeds.

TCP data was generated using Iperf on the DUT. Figure 6 shows the PDFs of the

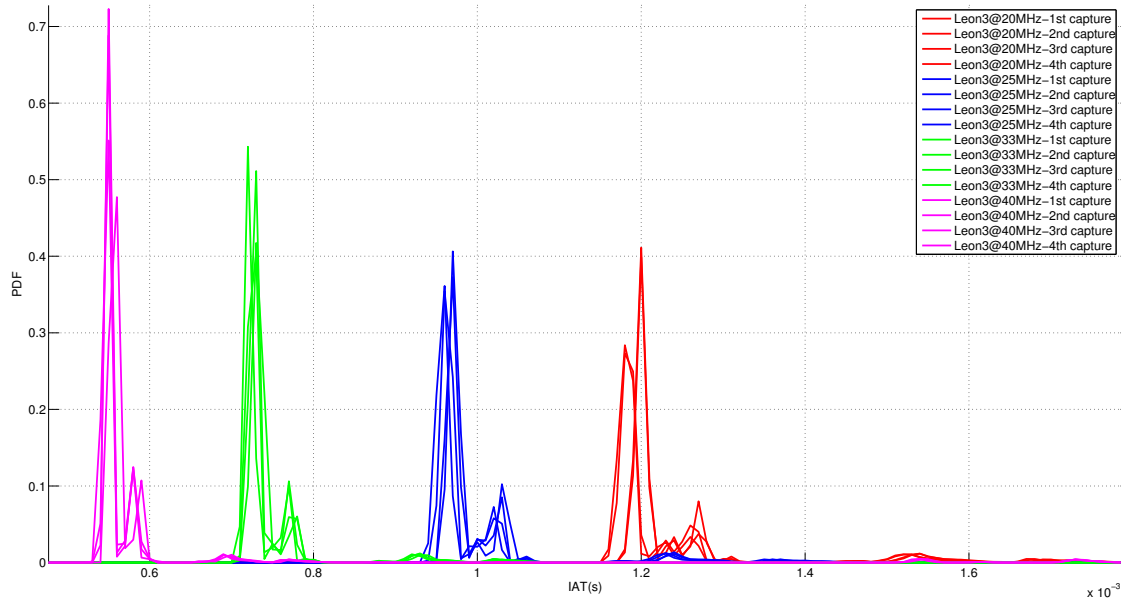


Figure 5: PDFs of UDP IATs for clock variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

IATs for clock speeds of 20MHz, 25MHz, 33MHz, and 40MHz on the XC3S1500 board. The histograms have 1000 bins of $10\mu\text{s}$ width. Here again, we see a shift to the right with decreasing clock speeds. This indicates that the configurations at the lower clock speeds were unable to match the peak TCP rate and hence had higher mean IATs. The decreasing heights of the peaks with decrease in clock speeds also indicates that the configurations at the lower clock speeds struggled to maintain a constant high data rate, thereby producing a more spread out PDF, with lower peaks.

The DUT was sent ICMP requests of 56 bytes packet size from the VAIO every 0.001s and the replies captured by the monitor. Figure 7 shows the PDFs of the IATs for clock speeds of 20MHz, 30MHz, 35MHz, and 40MHz on the XtremeDSP board. The histograms have 1000 bins of $10\mu\text{s}$ width. All the clock speeds have their peaks at the required 1ms IAT (rate at which ICMP requests were generated); however, the higher clock speeds, being able to match the required 1ms IAT with more ease, have higher peaks and lesser spread of PDFs, while the lower clock speed PDFs are more spread out.

The DUT was also sent ICMP requests of 1400 bytes packet size from the VAIO every 0.001s. Figure 8 shows the PDFs of the IATs for clock speeds of 20MHz, 30MHz, 35MHz,

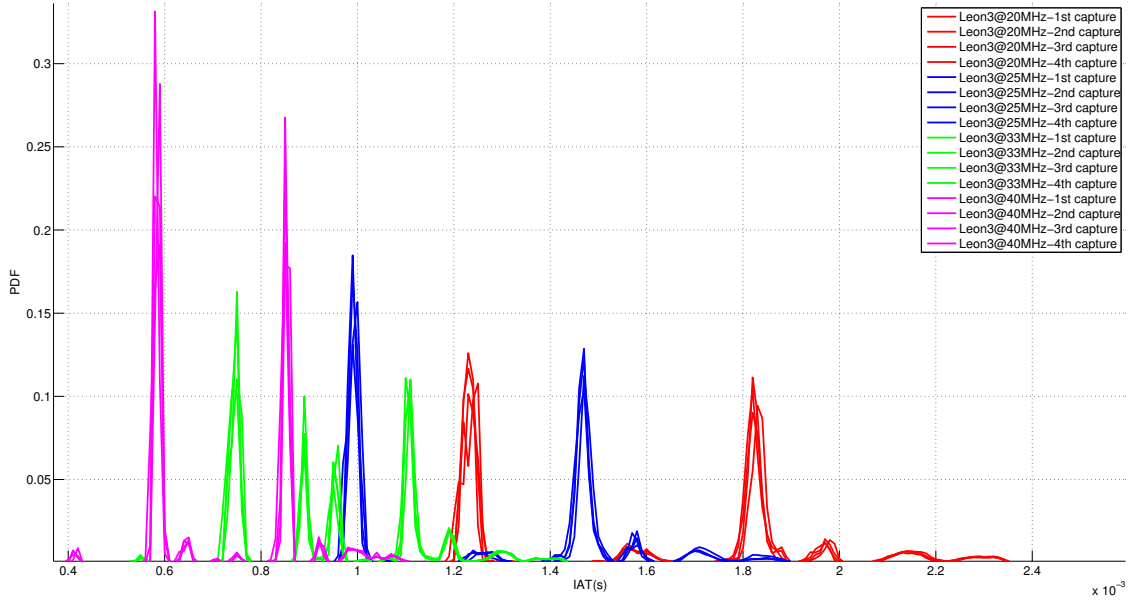


Figure 6: PDFs of TCP IATs for clock variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

and 40MHz on the XC3S1500 board. The histograms have 1000 bins of $10\mu\text{s}$ width. Here too, the higher clock speeds are able to approach the required 1ms mean IAT (echo request send rate was 1 per 1ms) closer with their peaks, while the lower clock speeds are farther off. This can also be attributed to the fact that when a device is interrupted by a ICMP request, the processor has to switch state and service the request. Hence, the slower clock speed configurations, which take longer to switch state and service the ICMP requests, end up with higher mean IATs (the right shift of the PDF peaks).

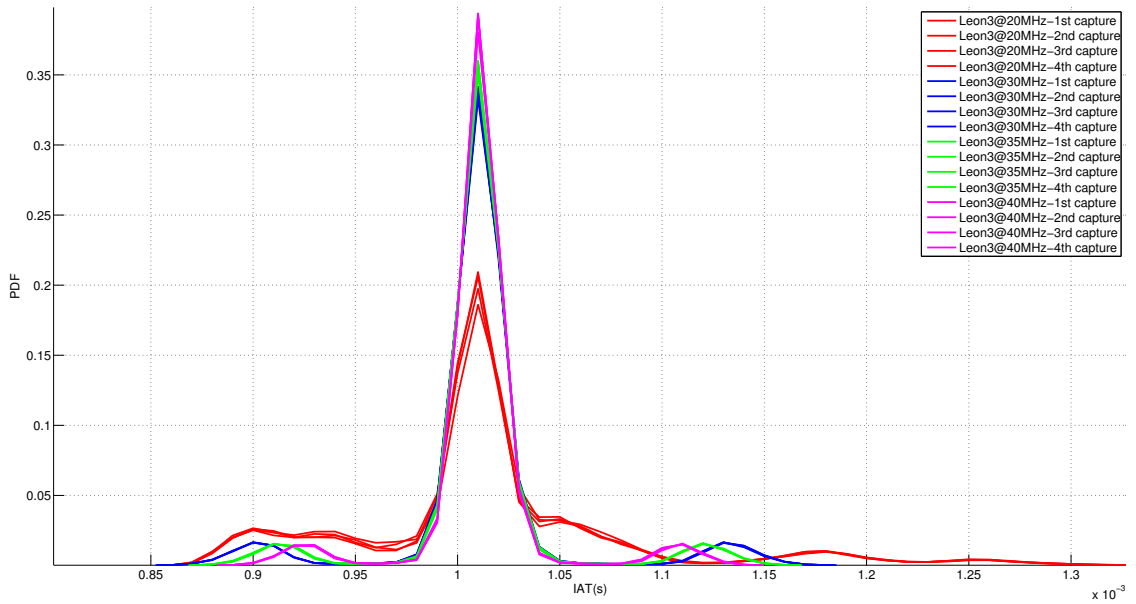


Figure 7: PDFs of ICMP(56 bytes) reply IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

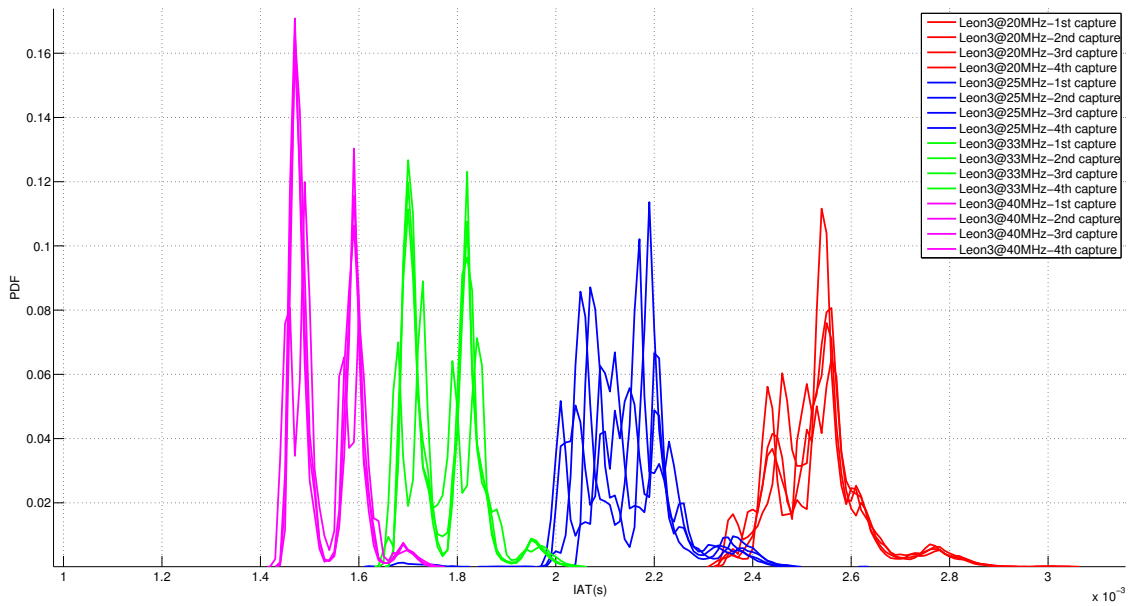


Figure 8: PDFs of ICMP(1400 bytes) reply IATs for clock variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

2.4.2 CPU Data Cache

To see the effects of data cache on the IATs, we emulated 40MHz clock, 8KB data cache configurations and varied the data cache replacement policy between random replacement and least recently used (LRU) for both boards and captured different types of traffic. The ICMP traffic was not significantly affected by data cache variations, as shown by Figure 9, which shows the XC3S1500 board's reply IATs to ICMP (56 data bytes) sent once every 0.001s from the VAIO to the DUT. This is because of the low memory usage upon receiving a 56 byte ICMP packet. The histograms have 1000 bins of $10\mu\text{s}$ width.

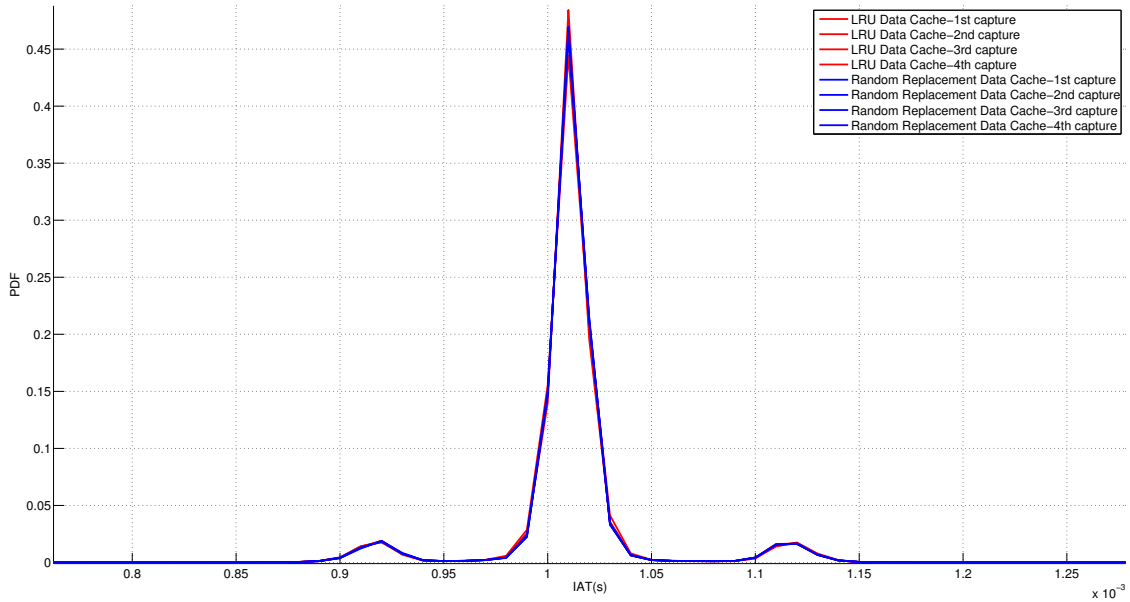


Figure 9: PDFs of ICMP (56 bytes) reply IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

Other memory dependent traffic types were more susceptible to data cache variations. Figure 10 shows the PDFs for IATs of ICMP (1400 bytes) replies from the XtremeDSP board. The ICMP requests were sent every 0.001s and the the histograms have 1000 bins of $10\mu\text{s}$ width. Here, the random replacement policy seems to be better and has lower mean IATs. However, its PDFs are more spread out while that of LRU are more peaked, though of higher mean IAT.

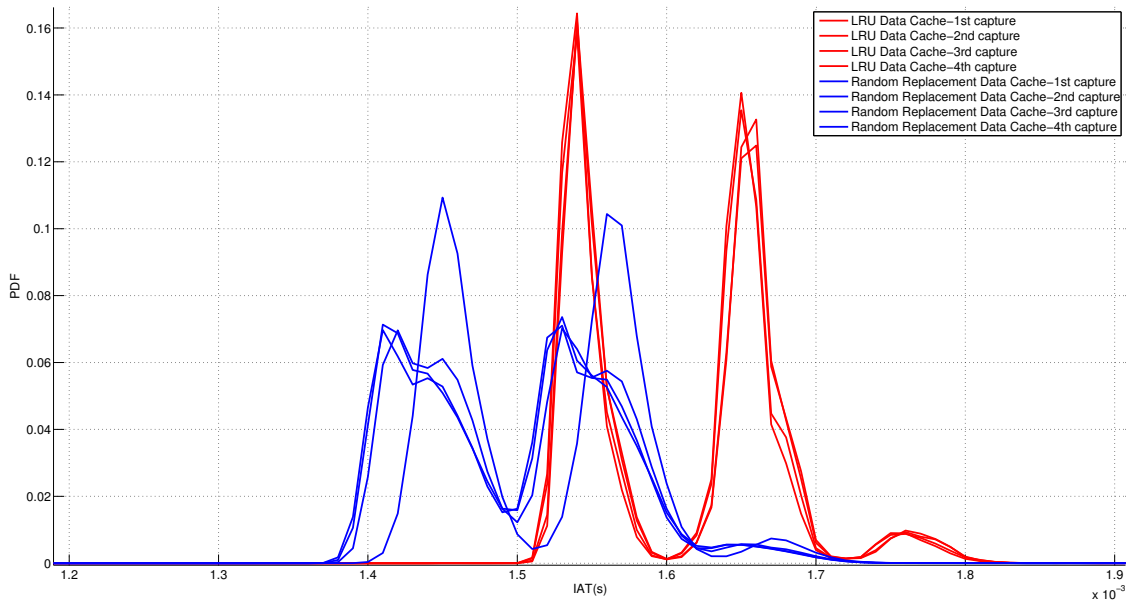


Figure 10: PDFs of ICMP (1400 bytes) reply IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

2.4.3 CPU Instruction Cache

To bring out the effects of instruction cache on the network traffic, we emulated a 40MHz, 8KB instruction cache configuration on both boards, varied their instruction cache replacement policies from LRU to random replacement, and captured the different traffic types. ICMP traffic was generally unaffected except that the PDF peaks were higher for LRU and more spread out for random replacement. This is because of the low effect of an incoming ICMP packet on the instruction cache. Figure 11 shows the PDFs for ICMP (56 bytes) reply IATs from the XtremeDSP board. The ICMP requests were sent every 0.001s and the histograms have 1000 bins of $10\mu\text{s}$ width.

Traffic which relied on the instruction cache such as TCP and UDP showed a marked shift in the random replacement PDFs to higher mean IATs. This is evident from Figure 12, which shows the PDFs for IATs of UDP (56 bytes data at 1Mbps) packets from the XC3S1500 board. This is due to the instructions that are executed by the processor to run Iperf to generate UDP and TCP packets, thereby depending upon the instruction cache

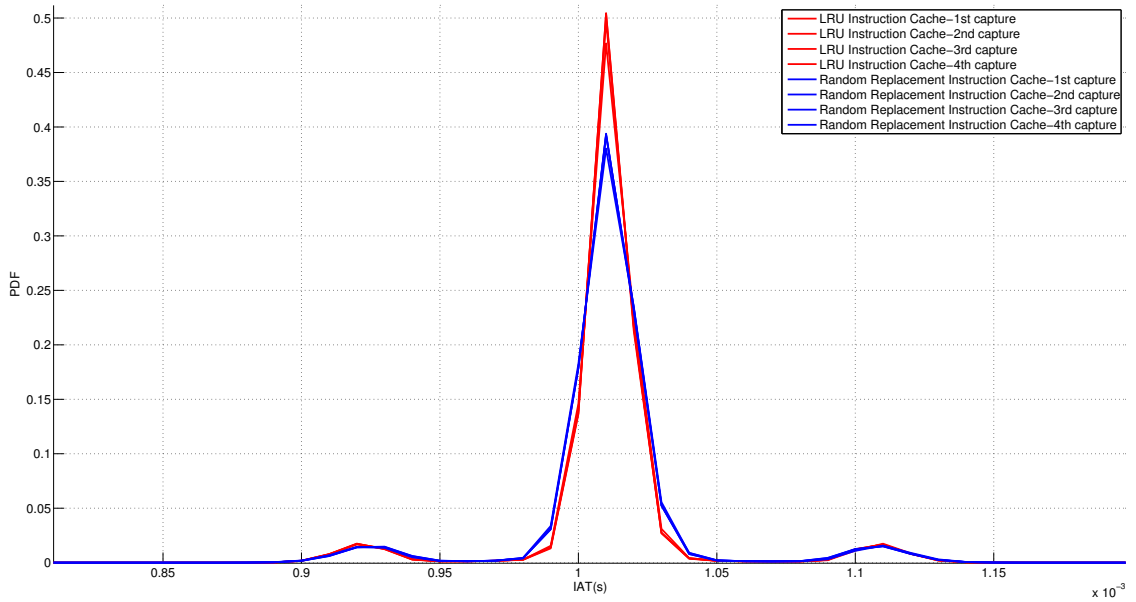


Figure 11: PDFs of ICMP (56 bytes) reply IATs for instruction cache variations on XtremeDSP with 1000 bins of 10 μ s width

replacement policy.

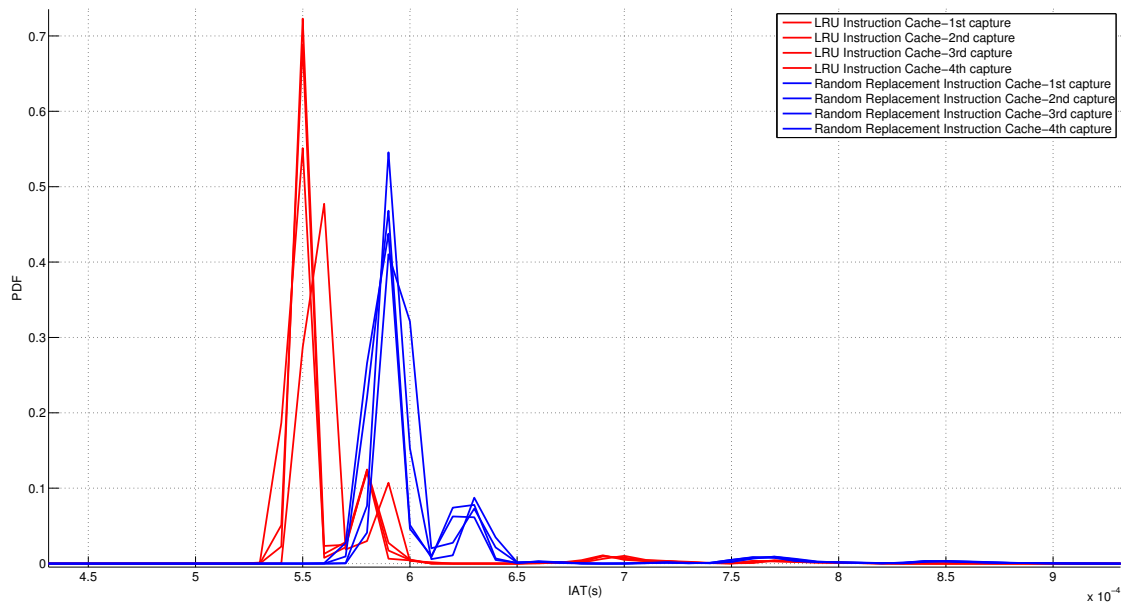


Figure 12: PDFs of UDP IATs for instruction cache variations on XC3S1500 with 1000 bins of 10 μ s width

2.4.4 RAM

Now we look at the effects of RAM on the network traffic. We evaluated this by emulating 40MHz configurations on both boards (XtremeDSP had 128MB RAM and XC3S1500 had 64MB) and varying their RAM sizes to 16 MB, which was the lowest RAM the SnapGear OS would successfully boot with. With this, we bring out the effects of a memory constraint on the system performance for each traffic type. As one would expect, RAM variations did not affect ICMP traffic a lot, as can be seen from Figure 13. It shows the PDFs for ICMP (56 bytes) reply packets' IATs from the XtremeDSP board. The ICMP requests were sent every 0.001s and the histograms have 1000 bins of $10\mu s$ width.

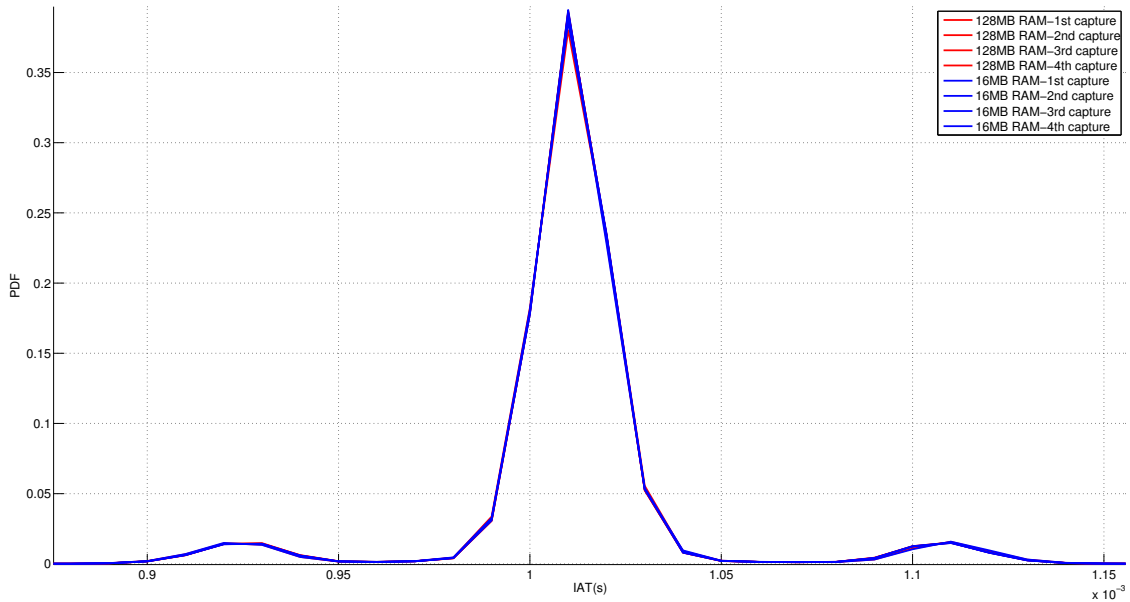


Figure 13: PDFs of ICMP (56 byte) reply IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu s$ width

However, the effects of RAM size constraints are evident for UDP, TCP, and ICMP (1400 bytes), which all depend on the RAM. Hence, any restriction on available RAM capacity leads to increased mean IATs. Figure 14 shows the PDFs for UDP (56 bytes data at 1Mbps) packet IATs from the XC3S1500 board. The histograms have 1000 bins of $10\mu s$ width. Here we see the 16MB configuration having higher mean IATs (as the PDF peaks are shifted to the right of that of 64MB configuration) than that of the 64MB configuration.

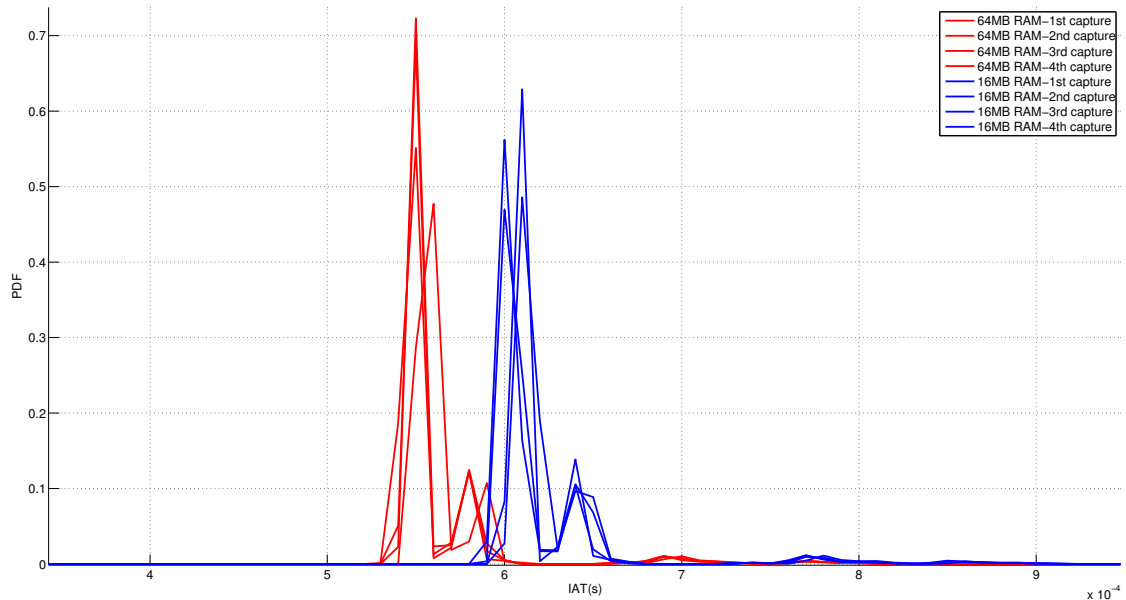


Figure 14: PDFs of UDP IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

2.5 Discussion

From the above results we find that all traffic types are affected by changes to the CPU clock. This is expected since the CPU clock is a crucial part of the processor and involved in deciding how fast a packet or any instruction for that matter, is processed. Thus, we see a lateral right shift in PDFs to higher mean IATs with increasing clock speeds.

CPU data cache affected TCP, UDP, and ICMP (1400 bytes) traffic, which make data accesses. Hence, any change in data cache replacement policy was met with change in the performance of the device that was noticeable in its network traffic. ICMP (1400 bytes) had the most deviation in PDFs due to its high data payload. ICMP (56 bytes) packets were the least affected since they make very little use of the data cache; hence, any changes to the data cache replacement policy had very little effect on the IATs.

CPU instruction cache affected TCP and UDP more than it did ICMP. This again is due to the fact that TCP and UDP packets were generated on the DUT using Iperf, running which would have meant stepping through many instructions, which is affected by the instruction cache replacement policy. We found a lateral right shift of the PDF peaks

of random replacement configurations to higher mean IATs, when compared to the LRU configurations. ICMP IATs just had more spread out PDFs of shorter heights for random replacement, while the mean IATs (PDF peak positions) remained the same as LRU.

RAM sizes affected only traffic types whose instructions were memory intensive i.e., UDP, TCP, ICMP (1400 bytes). These traffic types make memory accesses and immediately feel the effects of RAM size constraints. ICMP (56 bytes) on the other hand does not entail a high memory burden for its echo replies. Hence, the RAM size effect is minimal.

Thus, we see consistent effects that different components of the device have on various types of traffic depending on the type of traffic.

2.6 Conclusion

From the previous section, we can conclude that the effects of CPU and RAM on the device's network traffic IATs are very consistent and noticeable, since they are crucial parts of a computer system and will cause instructions (hence, packets sent or received) to be executed with different but consistent delays. Variations made to the CPU (clock speed or data cache replacement policy or instruction cache replacement policy) or the RAM (size) are instantly differentiable from the original variant by analyzing nothing but simple UDP, TCP, and ICMP traffic IATs. This lends itself very well to counterfeit detection, since these crucial components (CPU and RAM) are usually the most expensive in a system and hence most susceptible to counterfeiting. In the next chapter, we will discuss this promising application.

CHAPTER III

COUNTERFEIT DETECTION: AN APPLICATION

3.1 Motivation and Related Work

Counterfeit devices are today a common bane to companies and governments alike [24, 25]. Counterfeit devices amount to billions of dollars of loss in revenue to US semiconductor companies [8]. Counterfeiting also impacts the customer, since counterfeit devices usually have either old, repackaged versions of the components or components made from lower quality materials. Hence any customer is either receiving a degraded, outdated version of the device or a low quality, low reliability version of the device. This impacts no customer more than the military, whose critical systems have often been found to be compromised with counterfeit devices [7, 9, 26]. The resulting impact on national security can be devastating, not to mention the risk to the lives of the operators of lethal equipment with counterfeit, unreliable components.

Companies and governments are in turn, spending huge amounts of precious time and money to detect and counter these counterfeit devices [26, 25]. However, most of the techniques available today for counterfeit detection are either expensive, complex, require specialized hardware, are destructive, narrow-scoped, or slow. There is thus, a definite need for a more efficient and effective counterfeit detection mechanism, without all the drawbacks of the existing methods.

There are as many counterfeit detection techniques today as there are counterfeiting methods themselves. They include, intellectual property (IP) watermarking [27], hardware metering and auditing (both intrinsic [28, 29, 30] and extrinsic [31]), and physical unclonable functions (PUFs) [32]. All these (except intrinsic hardware metering) involve adding something extra to the legitimate design to make it uniquely identifiable. The work in [10] deals with a fingerprinting technique based on unique RF emissions from the device. The technique in [33] uses X-Rays to analyze the authenticity of the device. All these methods

either need intervention during the manufacturing process, or need specialized/expensive equipment to carry out their counterfeit detection techniques.

We found from the component characterization in the previous chapter, that variations in device components resulted in unique and consistent variations in the network traffic of the device. Armed with this knowledge, we implemented a first-of-its-kind network-based counterfeit detection solution, which is discussed in the following sections.

3.2 Testbed and Technique

We used the same testbed and setup as was used for the characterization in the previous chapter (Figure 1). The DUT was now a real PC and not a FPGA. The PC used was a Dell Optiplex 7010 with an Intel Core i3-3220 3.3GHz processor, 6GB RAM, running Ubuntu 11.10 version of Linux. The PC’s ICMP reply traffic (56 bytes and 1400 bytes, every 0.001s), ICMP request traffic (56 bytes and 1400 bytes, every 0.001s), UDP traffic (56 data bytes at 1Mbps), and TCP traffic (rate not controlled) were captured and the IATs analyzed statistically as before. All captures were four 15 minute captures for each traffic type except for TCP, whose high data rate allowed us to use four 5 minute captures to obtain approximately the same number of packets for analysis as the other traffic types.

For device component variations, we specifically targeted the CPU and RAM of the PC, since these are the most counterfeited devices [9, 34]. The CPU and the RAM are the most targeted components by counterfeiters because they are the most expensive parts of the computer. They are the most expensive parts of the computer because they are more central to the computer’s operation. We saw from the previous chapter that the more crucial parts of a device like CPU and RAM gave better statistical separation for network traffic IATs due to their central role in the packet generation process. Hence, it fits in nicely that the crucial parts of a computer are the most expensive, hence the most targeted for counterfeiting, while also giving the most stark differences in our technique of statistical analysis of network traffic IATs.

Besides showing the effects of device component variation on traffic IATs using PDFs, we also used the captured IATs to train neural networks for pattern recognition based

classification as described next, and obtained recall results for the different configurations.

3.2.1 Neural Networks

Neural networks, or more specifically, artificial neural networks (ANNs), are mathematical models derived from and inspired by biological networks of neurons. They are used to model relationships between inputs and outputs. The relationships are modeled as a series of interconnections between neurons, which accept an input, operate on it based on some function, and then produce an output. Thus, the entire neural network can be thought of as a function taking in an input and producing an output based on the input. This function is in turn a compound of all the functions represented by the individual neurons in the ANN. Just like the biological nervous system that they mimic, these ANNs can be trained and used for pattern recognition.

For our IAT-based counterfeit detection technique, we use a neural network based classifier. The pattern recognition technique we use (`patternnet` function in MATLAB) involves the use of the 'feedforward' class of ANNs to train on and classify the histogram-based signatures. Feedforward ANNs are essentially acyclic variants of ANNs, as is shown in Figure 15, with each neuron operating on the output of the neuron before it, thereby forming a complex one-way network of neuron functions. The feedforward ANN is set to use 'scaled conjugate gradient backpropagation' as the training function. The feedforward network used has two layers, hidden and output. This kind of ANN can learn any input-output relationship provided there are enough hidden neurons. We have empirically found 50 hidden neurons to be optimal for our application. We form histograms from blocks of 2500 packet IATs and use half of these histogram distributions to train the network and half to test. This is done with all the device configurations' packet IATs. The trained neural network then contains the signature database against which the test histograms are compared. The neural network returns a value ranging from 0 (dissimilar) to 1 (identical) for each trained signature when fed a test histogram. The highest value from amongst these, which signifies the signature most similar to the test histogram, is chosen as the device to classify the test histogram. This training process is repeated for each traffic type, thereby creating a trained

neural network for each traffic type. Then, the corresponding neural network is used for testing histograms of each traffic type.

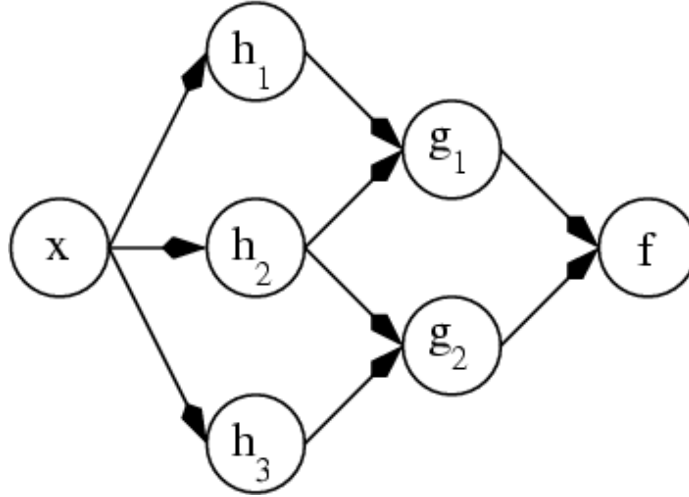


Figure 15: PDFs of TCP IATs for CPU variations on Optiplex 7010 with 1000000 bins of 1ns width

3.3 Tests conducted

We now discuss the results obtained for different traffic types for CPU and RAM variations on the PC. Only relevant results are discussed here. Other figures are listed in APPENDIX B.

3.3.1 CPU

We captured ICMP (56 bytes and 1400 bytes) traffic to and from the PC, as well as UDP and TCP from the PC. Then, we exchanged the i3 processor for an Intel Core i7-3770 3.4GHz processor and repeated the experiments. Figure 16 shows the processor being replaced from the motherboard of the PC after removing the processor fan. For all captured traffic types, the PDFs of IATs were plotted. We found that ICMP requests sent from the PC gave the best separation of PDFs. Figure 17 shows the PDFs of IATs of ICMP requests (56 bytes) sent out from the PC every 0.001s and plotted with 10000 histogram bins of 200ns bin width. The i3 PDFs peak at the required mean IAT of 0.001s (ICMP send rate), while the i7 PDFs have a smaller peak at 0.001s and two lateral peaks. This wide separation between the two can be easily spotted by a classification algorithm. Other traffic types also show good separation. TCP results on the other hand, are not so clearly separated, being almost

the same except at the tips of the histogram peaks. Figure 18 shows the PDFs for TCP IATs plotted with 100000 histogram bins of 1ns bin width.

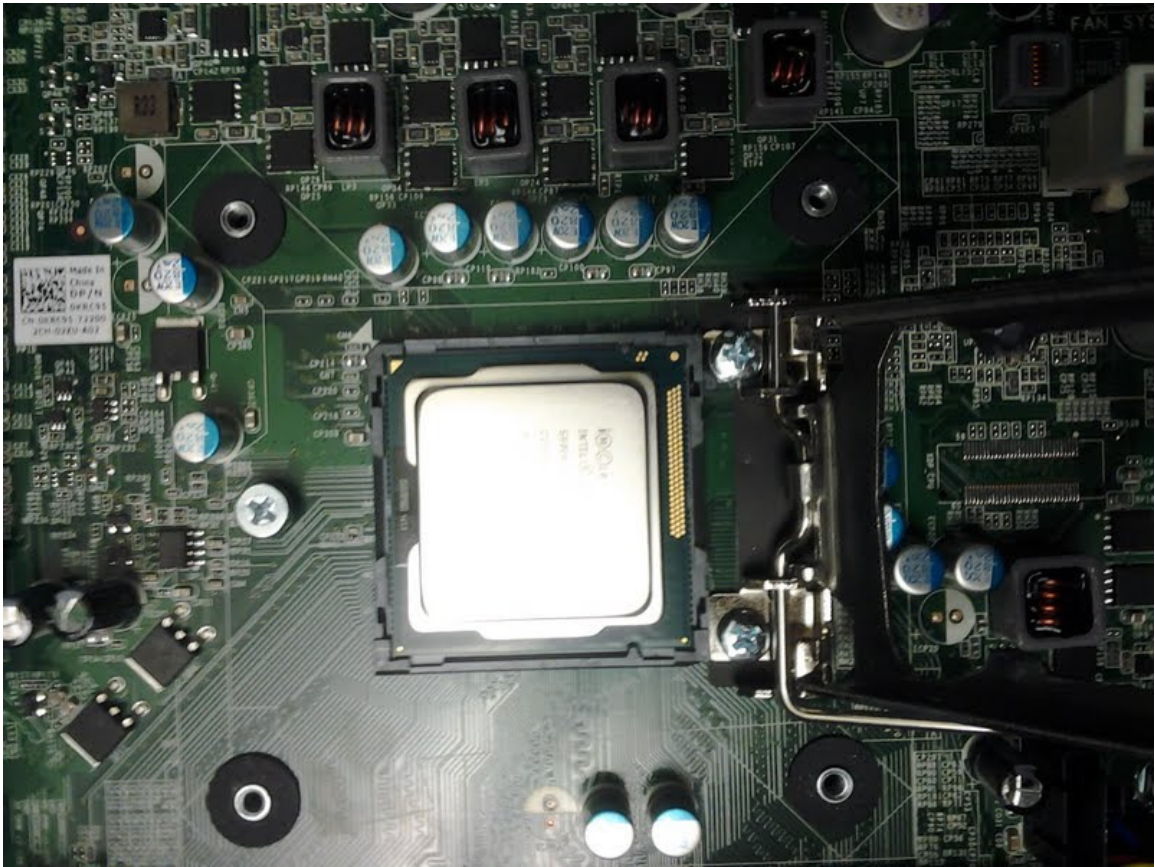


Figure 16: Processor changed on the PC motherboard

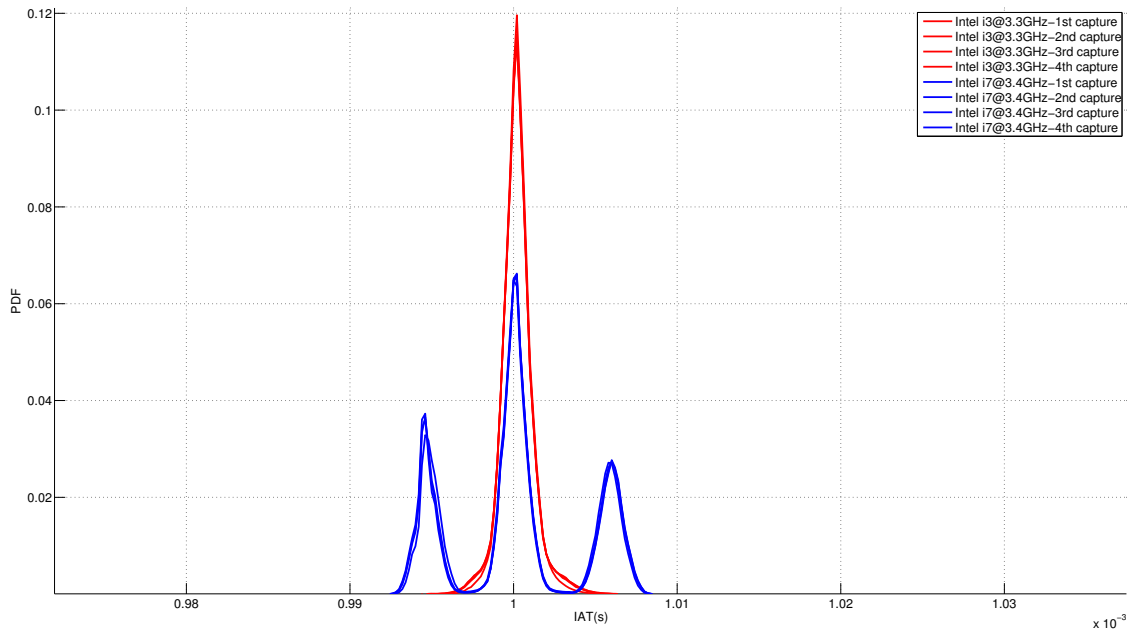


Figure 17: PDFs of ICMP (56 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width

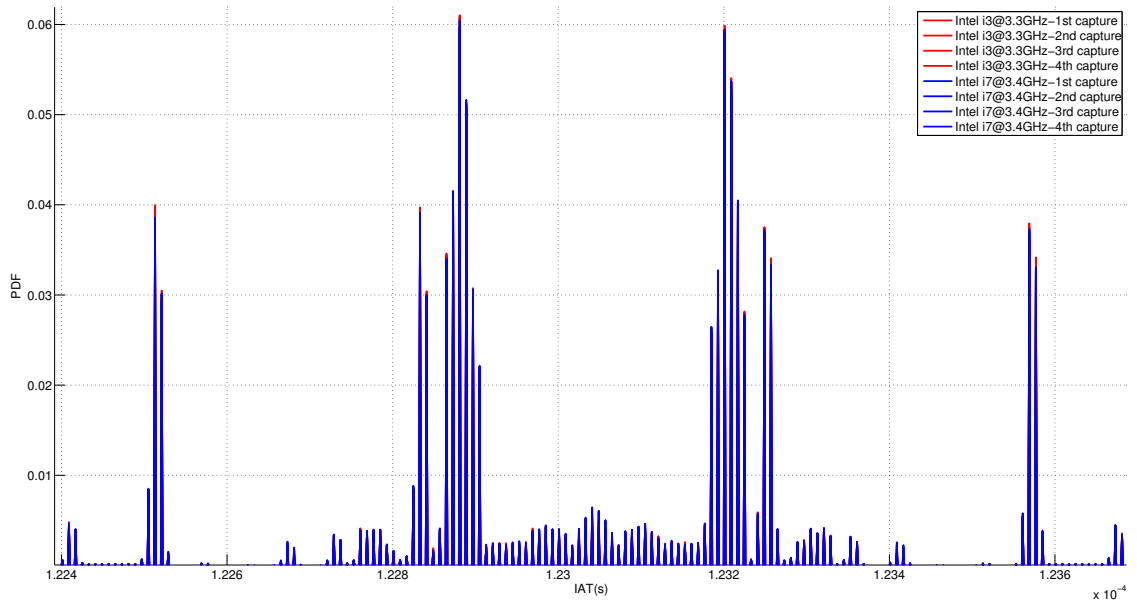


Figure 18: PDFs of TCP IATs for CPU variations on Optiplex 7010 with 1000000 bins of 1ns width

For each traffic type, the captured traffic IATs were fed to a neural network based

classifier as histograms formed from blocks of 2500 packets. Half of the histograms for each traffic type was used to train the neural network and the other half were used for testing. The other traffic types were similarly used to train the network and run classification tests. Table 1 shows the recall values for the different traffic types for i3 and i7; where, recall is the ratio of the number of true positives (Tp) to the sum of the number of true positives and false negatives (Fn).

$$Recall = \frac{Tp}{Tp + Fn} \quad (1)$$

Table 1: Recall values for i3 and i7 for various traffic types

	TCP	UDP	ICMP (56B) replies	ICMP (1400B) replies	ICMP (56B) requests	ICMP (1400B) requests
i3	0.503333	0.877778	0.938719	0.792479	1	0.924896
i7	0.515556	0.983333	0.828929	0.927677	1	0.869444

Here we see that the recall values for ICMP (56 bytes) requests are a perfect 1. As expected, TCP has low recall values of 0.5, which is not very useful for counterfeit detection.

3.3.1.1 IAT Stability Across Multiple CPUs of the Same Type

To show the IAT stability and repeatability of the captures, we also ran the above tests for two more Intel i3s and Intel i7s each. Again, we captured ICMP (56 bytes and 1400 bytes) traffic to and from the PC, as well as UDP and TCP from the PC, for each CPU. We thus had captures for three i3s and three i7s, for all six traffic types. Figure 19 shows the PDFs of IATs of ICMP requests (56 bytes) sent every 0.001s and plotted with 10000 bins of $1\mu s$ width for the three i3s and i7s. As we can see, the PDFs for a processor are very close across four iterations of its captures, and are quite separated from the other processors. This provides a good window to classify the processors based on the PDFs.

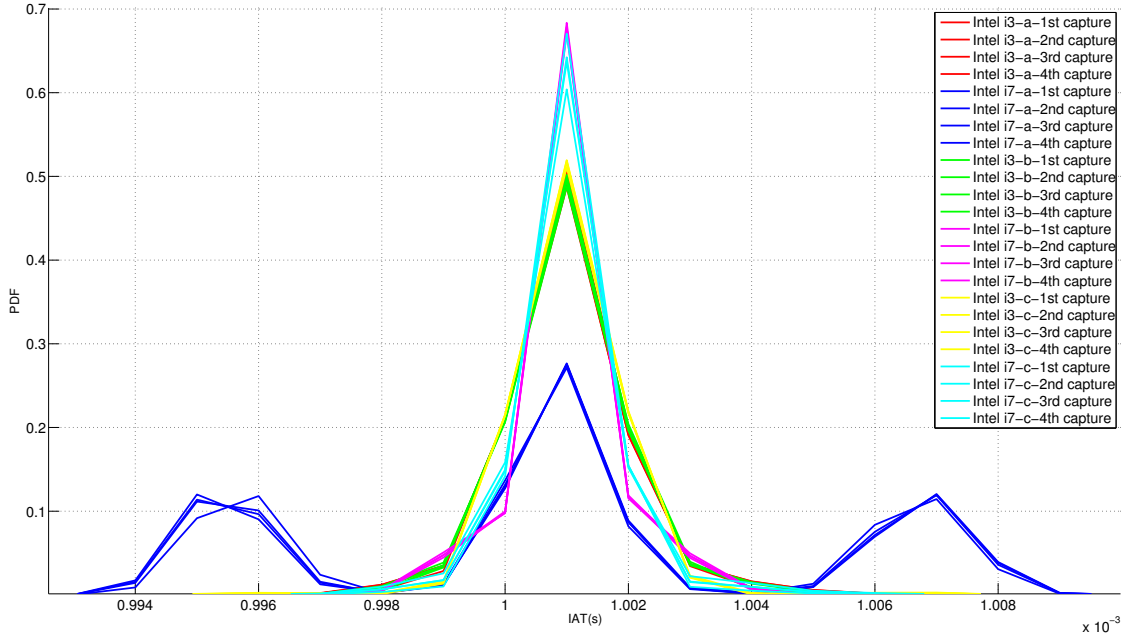


Figure 19: PDFs of ICMP request (56 bytes) IATs for different i3s and i7s on Optiplex 7010 with 10000 bins of $1\mu\text{s}$ width

For each traffic type, the captures of two of the three i3s were used to train a neural network, while the third was used to test it. The recall results were calculated and the entire process was repeated for the three i7s. Empirically it was found that ICMP requests (56 bytes), which showed the best recall values in the individual comparison between i3 and i7, gave the best recall value of 0.84 for classification of the category of the device (i3 or i7).

3.3.1.2 NIC vs NetFPGA

Here we show the recall values of NIC based captures and why we chose to use NetFPGA instead. We repeated the same experiments as just described with the three i3s and i7s. We captured ICMP (56 bytes and 1400 bytes) traffic to and from the PC, as well as UDP and TCP from the PC, for each CPU. This time however, we used a Broadcom NetXtreme BCM5722 NIC to capture the network traffic on the monitor instead of a NetFPGA. The PDFs were again plotted and compared. Figure 20 shows the PDFs of IATs of ICMP requests (56 bytes) sent every 0.001s and plotted with 10000 bins of $1\mu\text{s}$ width for the three

i3s and i7s. Comparing Figure 20 to the similarly plotted (same bin width, traffic, and processor set) PDFs of Figure 19, which used a NetFPGA for capturing the same ICMP request (56 bytes) traffic, we can easily see the higher amount of overlapping in the NIC based plots. This will lead to greater difficulties in classifying the CPUs from each other. Training neural networks and calculating recall values as we did for the previous sub section confirmed this. The category (i3 or i7) recall for ICMP requests (56 bytes) dropped from 0.84 using NetFPGA to 0.50 using NIC. The overlapped PDFs can be attributed to the fact that incoming packets are not immediately timestamped by the NIC. They are instead queued to be picked up by the kernel, which then uses the system clock to timestamp the packets as and when the kernel processes the interrupt request, based on its current scheduling queue. The way these timestamps are generated is very OS dependent and can sometimes even lead to multiple queued packets to be timestamped with the same time value. The NetFPGA on the other hand, uses hardware timestamping and immediately timestamps the arriving packet upon receipt using its own on-board 125MHz clock. This also affords the NetFPGA a precision of up to 8ns while kernels (when using NICs) usually timestamp packets in the order of microseconds.

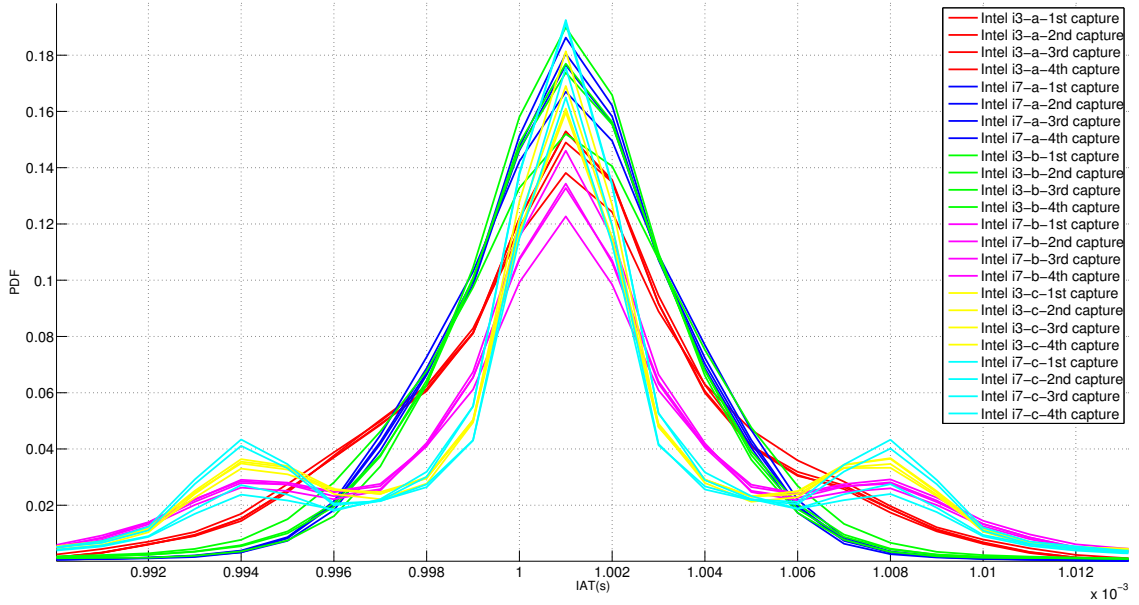


Figure 20: PDFs of ICMP request (56 bytes) IATs for different i3s and i7s on Optiplex 7010 with 10000 bins of $1\mu\text{s}$ width captured using a NIC

3.3.1.3 WAN vs LAN

All the experiments in this thesis were done in a LAN setting as Figure 1 already showed. However, to study the scope of this technique on a WAN setting, we also conducted a single experiment where we captured UDP traffic from the DUT located across a WAN, as portrayed by Figure 21, for a single i3 and a single i7. Here again, we used the same Sony VAIO as the sender/receiver and the Optiplex 7010 as the DUT. The DUT was however situated on a different WAN, and the DUT's UDP packets (four 15 min captures at 1Mbps) to the VAIO were captured. UDP traffic was chosen since it was the slowest from our regular set of six traffic types. Due to the delays inherent in a WAN link, we found that TCP and ICMP traffic could not be sent or received at the same rate that we did on the LAN. We were able to however, receive UDP at the regular 1Mbps rate. This enables us to make a fair comparison with LAN results.

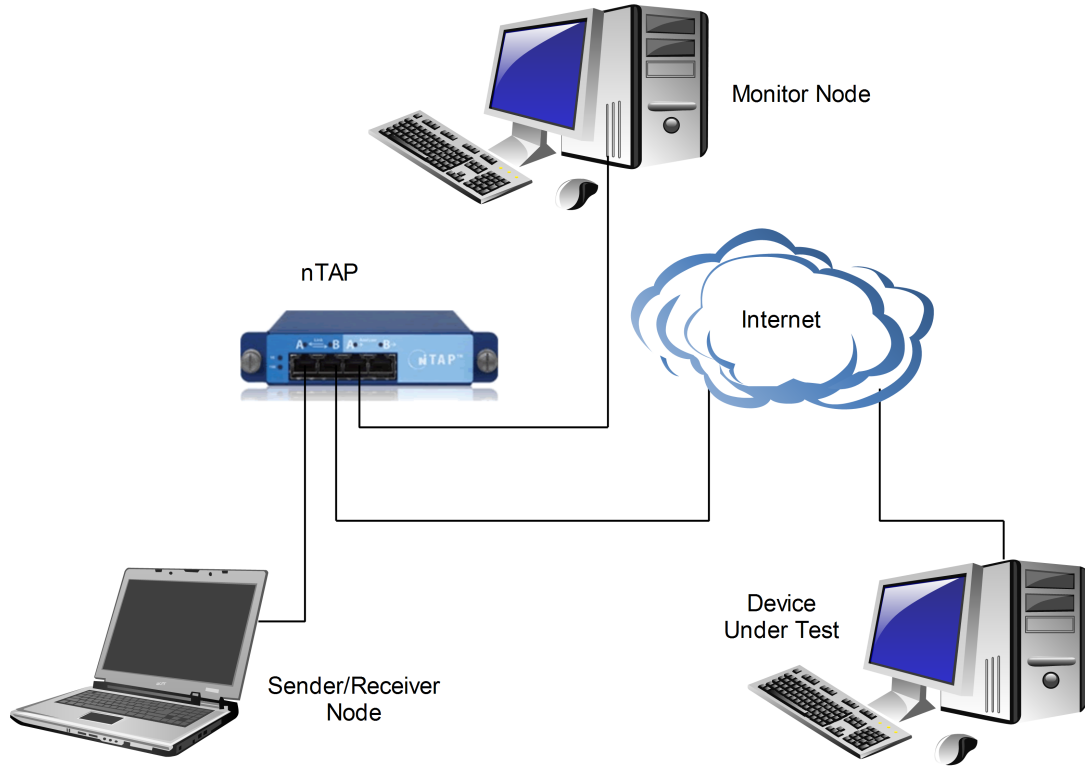


Figure 21: Testbed for WAN setting

Figure 22 and Figure 23 show the PDFs of UDP (56 bytes data size at 1Mbps) traffic for CPU variations obtained for the WAN experiment as well as the previous LAN based experiment respectively, plotted with the same bin size (50ns) and number of bins (100000). Strangely, WAN PDFs have lower average IATs (PDF peaks at lower IAT values) than LAN PDFs. This, however, is misleading, as the corresponding CDFs plot of the same data combined into Figure 24 shows. This clearly indicates that there are very high IATs for over six percent of the WAN captures. Correspondingly, the variances for LAN and WAN traffic are $4.216e-11$ and $2.463e-6$ respectively. This can be attributed to the higher delays inherent in the WAN paths. Figure 25 shows traceroute output between the DUT and the VAIO, indicating the many routers encountered between the DUT and the VAIO while traversing the WAN. At each hop, each router queues packets and forwards them at best-effort. Hence, any fingerprinting done will be heavily influenced by the ever dynamic inter-WAN path, other traffic in the routers, as well as the rate at the routers queue and

forward packets. Hence, the IATs we measure are more a representation of the path delays than the minuscule delays introduced by the processor at the packet origin. Thus, this is a limitation of our technique. Our technique is more suitable for counterfeit detection of devices on a LAN, where the paths are constant and there are no uncertainties added by routers.

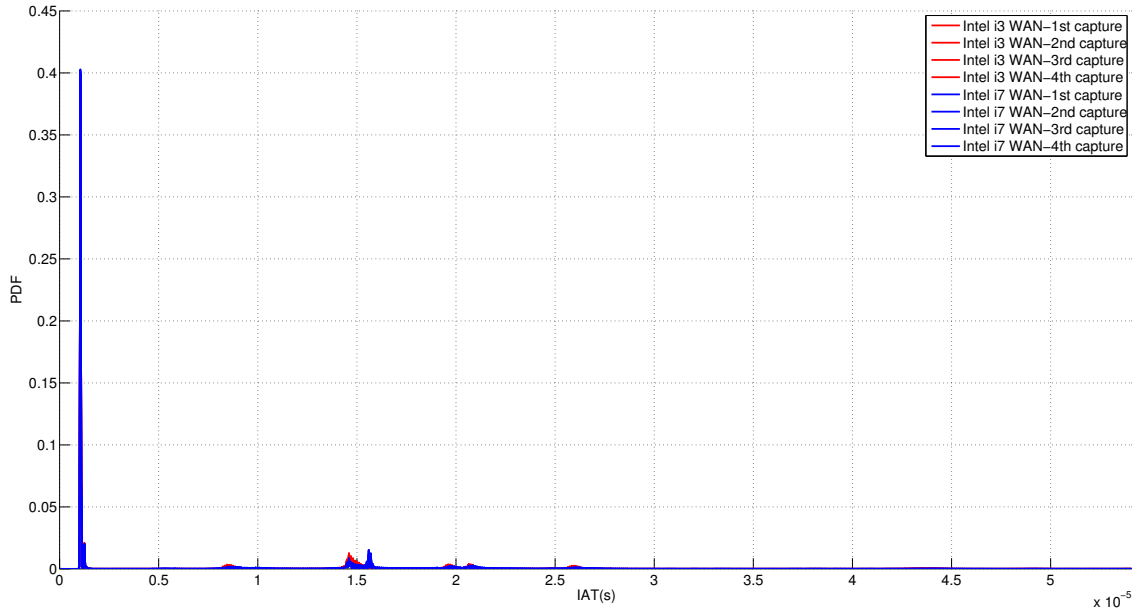


Figure 22: PDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on WAN with 100000 bins of 50ns width

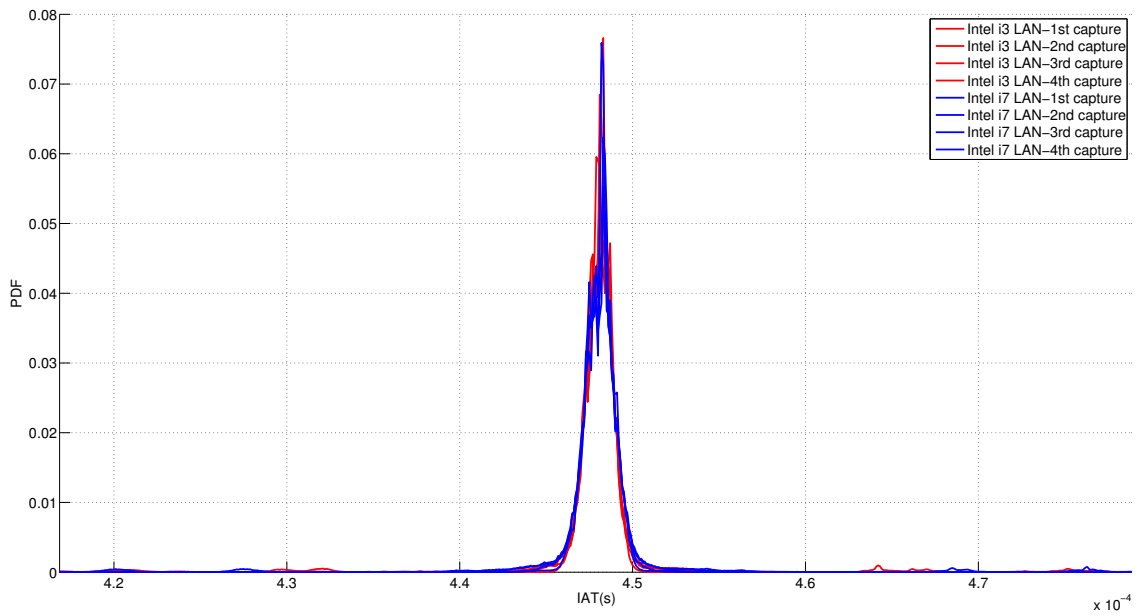


Figure 23: PDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on LAN with 100000 bins of 50ns width

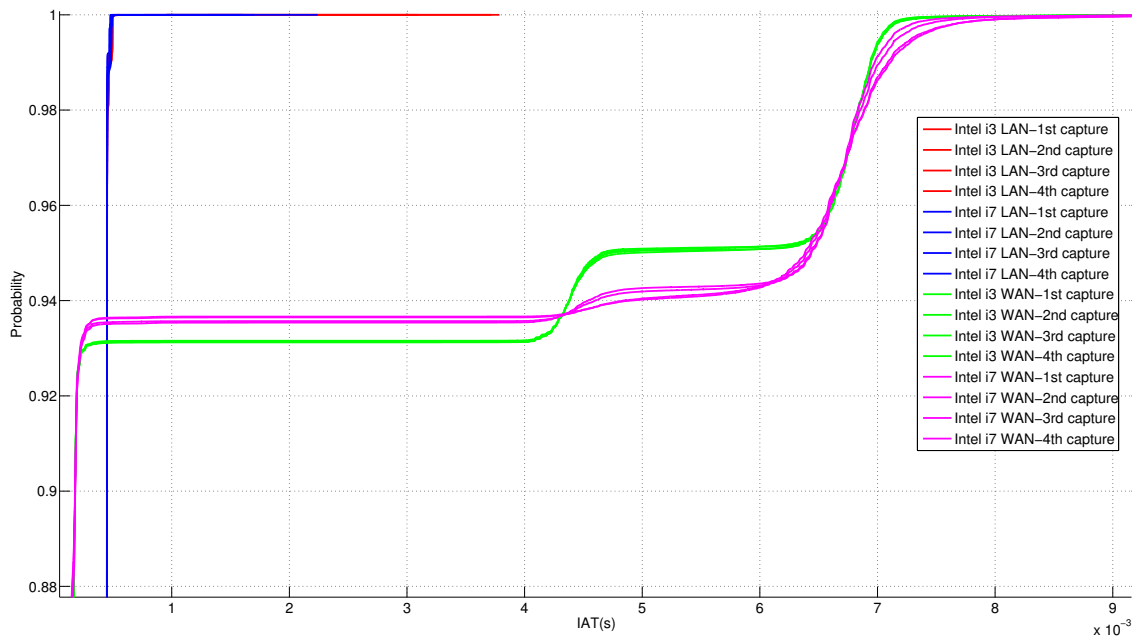


Figure 24: CDFs of UDP (56 byte data size at 1Mbps) IATs for CPU variations on Optiplex 7010 on WAN and LAN

```

supreeth@supreeth-PORTEGE-R835:~$ traceroute 143.215.207.12
traceroute to 143.215.207.12 (143.215.207.12), 30 hops max, 60 byte packets
 1  50.155.20.1 (50.155.20.1)  28.673 ms  29.551 ms  29.610 ms
 2  xe-10-0-0-32767-sur01.a6atlanta.ga.atlanta.comcast.net (68.86.109.17)  13.546 ms  14.462 ms  14.516 ms
 3  xe-10-1-0-0-ar01.b0atlanta.ga.atlanta.comcast.net (68.86.107.6)  14.695 ms  14.676 ms  14.728 ms
 4  te-9-2-ur01.b0atlanta.ga.atlanta.comcast.net (68.85.109.146)  14.720 ms  te-9-1-ur01.b0atlanta.ga.atlanta.comcast.net (68.85.109.102)  15.367 ms
 5  15.384 ms
 * * *
 6  sox-to-rich-gw2.sox.net (143.215.194.6)  34.569 ms  29.969 ms  34.256 ms
 7  143.215.254.97 (143.215.254.97)  35.169 ms * *
 8  * 143.215.253.137 (143.215.253.137)  34.888 ms  34.268 ms
 9  * * *
10  lawn-143-215-207-12.lawn.gatech.edu (143.215.207.12)  36.628 ms * *
supreeth@supreeth-PORTEGE-R835:~$

```

Figure 25: Traceroute output for path between DUT and VAIO when on different networks

3.3.2 RAM

We captured ICMP (56 bytes and 1400 bytes) traffic to and from the PC, as well as UDP and TCP from the PC. We then reduced the RAM size from 6GB to 2GB and repeated the captures. The PDFs of IATs for all traffic types were plotted. Once again we saw good PDF separation for ICMP and UDP, but not TCP. Figure 26 shows the PDFs of IATs of UDP (56 byte data size at 1Mbps) packets plotted with 1000 bins of $1\mu\text{s}$ width. Decreasing the amount of RAM from 6GB to 2GB decreases the PDF peak heights and caused them to be spread out more. This shows an inability to maintain the required 1 Mbps rate for most of the packets, as compared to the PDFs of the 6GB RAM configuration.

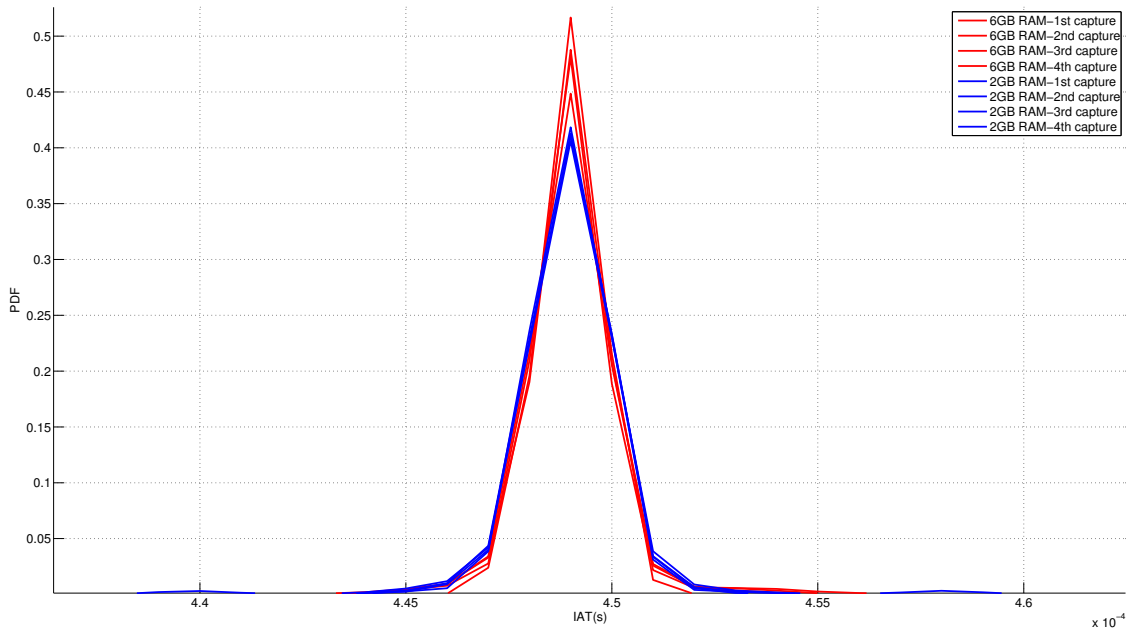


Figure 26: PDFs of UDP IATs for RAM variations on Optiplex 7010 with 1000 bins of $1\mu\text{s}$ width

Once again, for each traffic type, the IATs are fed in as histograms of 2500 packet blocks to a neural network based classifier. Half the IAT values are used for training the neural network and the other half for testing the classifier results. The recall values are tabulated in Table 2, which shows the recall values for each traffic type for both 6GB RAM and 2GB RAM IAT histograms.

Table 2: Recall values for 6GB and 2GB RAM for various traffic types

	TCP	UDP	ICMP (56B) replies	ICMP (1400B) replies	ICMP (56B) requests	ICMP (1400B) requests
6GB	0.648889	0.918889	0.948468	0.694986	0.997218	0.988873
2GB	0.464444	0.941111	0.853964	0.798331	0.993056	0.96662

Here again, we see the highest recall of 0.99 for ICMP requests and the lowest recall of 0.46 recall for TCP traffic.

3.4 Discussion, Limitations, and Conclusion

It is evident that statistical analysis of IATs can be used to detect the modification of an internal component of a device like CPU and RAM. This can be done with a quick capture of just 2500 packets to form IAT histograms, which are used as signatures, to compare to a signature database of a legitimate device. In an implementation scenario, this legitimate signature could be provided by the manufacturer by a secure means to the customer. The recall results were shown for all traffic types and the results for ICMP and UDP traffic are the most promising for this technique of counterfeit detection. High data rate TCP traffic gave very low recall rates and is not suitable for counterfeit detection by this technique. The stability of the IAT signatures across multiple i3s and i7s was also shown. The accuracy afforded by the use of a NetFPGA, and the lower recall values of NIC based captures were shown. An experiment involving the DUT on another WAN was also shown and the results discussed.

Thus, this technique of counterfeit detection based on network traffic IATs is very fast, since it only needs 2500 packets to provide results. It is cheap, since it does not need any expensive and specialized hardware to capture traffic and perform detection. All we need is a simple computer, nTAP, and NetFPGA. It is simple since it just involves capturing

traffic and creating histograms of IATs, hence there are no implementation difficulties or steep learning curves. This method is non-invasive and is non-destructive to the device, as it can passively (monitor traffic being sent out by the DUT) or actively (send the device ICMP requests so as to monitor its responses) monitor a device's network traffic without changing the hardware or software of the device in any way. Thus it can be used on devices without damaging them and also without rendering their software corrupt. It is also, we believe, the first network-based counterfeit detection technique. Being that it is simple and network-based, it can be used on a wide range of devices and device types. Thus, it is also a broad-scoped technique, as all that is required is that the DUT has a network protocol stack and is able to send and receive traffic. This can be leveraged to perhaps deploy the technique remotely on a large distributed network. This technique is also much faster and simpler than running benchmark software like [35] to determine changes to the device, as there is no need to individually access each device and install custom benchmarking software. The nodes can simply be booted, connected to a switch or access point, pinged (in the case of ICMP) for a about a minute (2500 packets), and immediately classified, with no need for specialized software on the DUTs. Thus, the technique is available to a wider range of devices, is easily scaleable, and is more feasible on large networks than applying benchmarking techniques to counterfeit detection.

However, the network-based nature of the technique brings with it the obvious limitation that the DUT is required to be a networked device. This is a limitation that applies to all network-based approaches. Also, components that are non-crucial or those that do not directly affect the packet generation process will perhaps not be so easily detected by this technique. These component variations are perhaps more easily identifiable using a benchmarking technique. Another limitation of the technique became evident from the WAN experiment conducted, where the WAN paths introduced large delays and masked the subtle delays caused by the processors. Hence, this technique is only suitable for stable LANs, and not WANs, where it ends up fingerprinting the link itself.

CHAPTER IV

CONCLUSION

As networks continue to evolve, there have been many studies and techniques that look at traffic dynamics. There is thus a need to characterize the effects the devices and their components themselves have on the network traffic they create. This thesis provided a characterization of the effects of the device components like CPU and RAM on the different network traffic types.

This work introduced the concept of the unique and consistent idiosyncratic effects that device components have on network traffic. We characterized the individual effects that device components like CPU clock, CPU data and instruction cache, and RAM have on different types of network traffic. This characterization was based on the IATs of the packets sent out of the FPGA DUT. The IATs were then statistically analyzed using PDFs. Each device component was varied one at a time, and the PDFs were compared to bring out the effects of that particular device component on network traffic.

We also looked at using this technique of IAT based analysis to determine the legitimacy of a device and its components. Counterfeit detection is of special interest because of all the counterfeit devices that have plagued industry and the military alike, costing them time, money, and putting lives at risk. However, none of the current techniques are fast, efficient, effective, simple, cheap, non-invasive, non-destructive, and broad-scoped, like our technique is. It provides a first of its kind network-based approach to counterfeit detection. We conducted similar experiments as we did before, but this time on a real PC and showed promising recall results of up to 100% using a classifier, thereby proving the efficacy of our technique. We also discussed the strengths and limitations of our technique, and that of network-based counterfeit detection in general.

4.1 Future Work

There is immense potential for this work, especially in the application of counterfeit detection. Possible future work includes experimenting on more computers for different types of component combinations. We plan to test many more (beyond the three each we have already shown) Intel i3 and Intel i7 processors to verify the stability of the signatures, and also plan to test other processors like Intel i5. We will also run experiments varying multiple components at a time, and observing the effects they have on the network traffic. Also, we plan to use different classification algorithms (e.g., cosine similarity, cross-correlation, etc.) to compare with the neural network based one we have used. Another future work is to look at the efficacy of the technique when targeting a loaded system that is running other tasks. While we can expect a new system to be idle when we apply our counterfeit detection technique, it is useful if one could check the legitimacy of systems that are already deployed in networks and are executing jobs that cannot be halted for the counterfeit tests.

APPENDIX A

CHARACTERIZATION

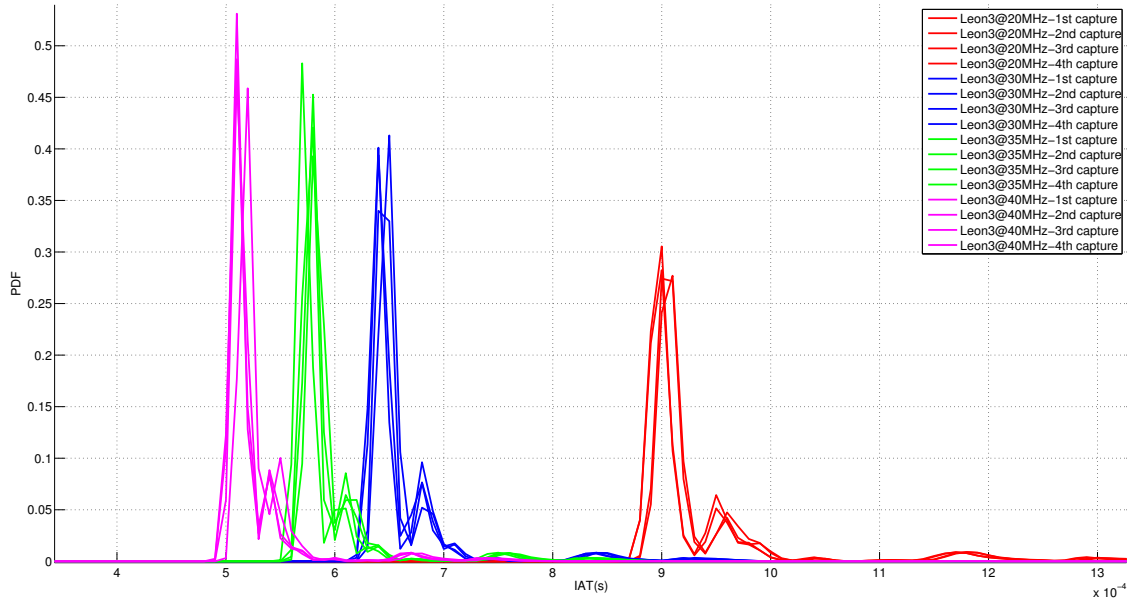


Figure 27: PDFs of UDP IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

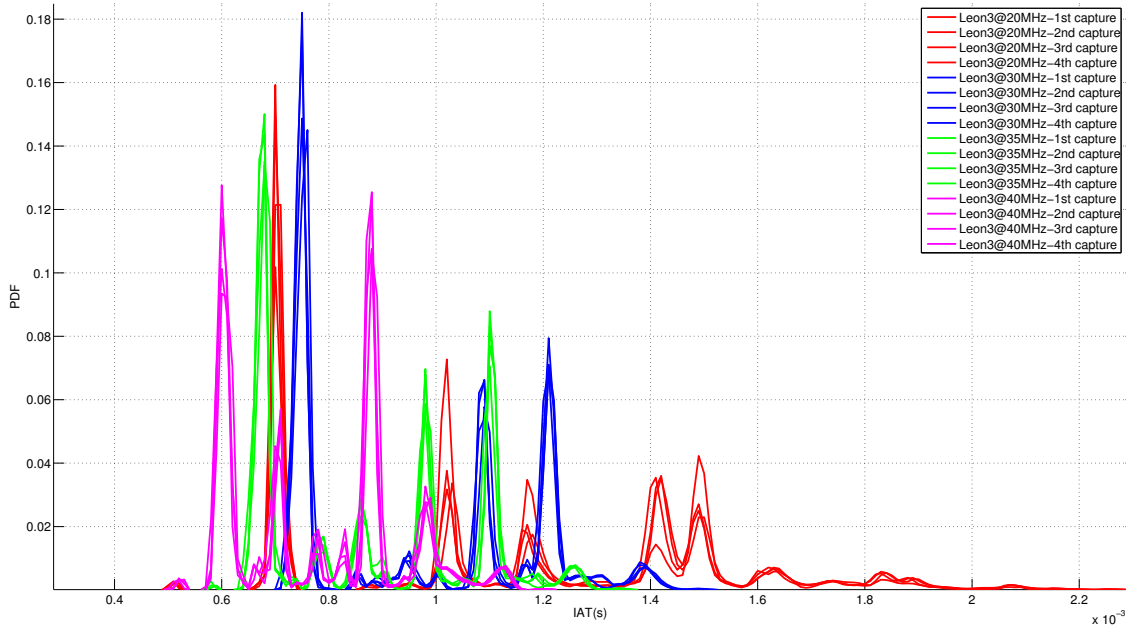


Figure 28: PDFs of TCP IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

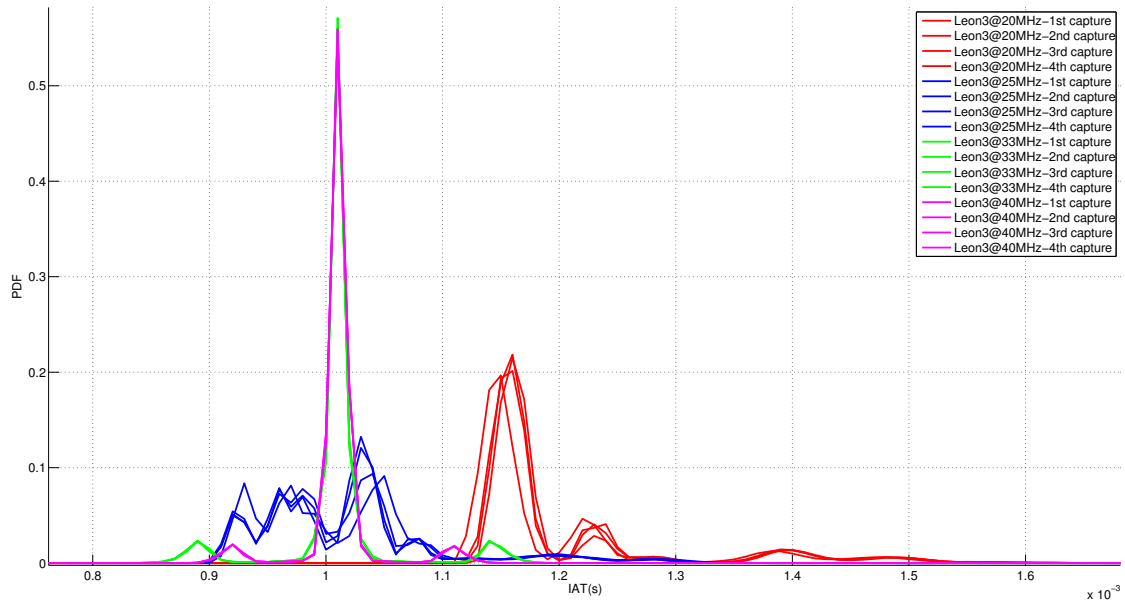


Figure 29: PDFs of ICMP(56 bytes) IATs for clock variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

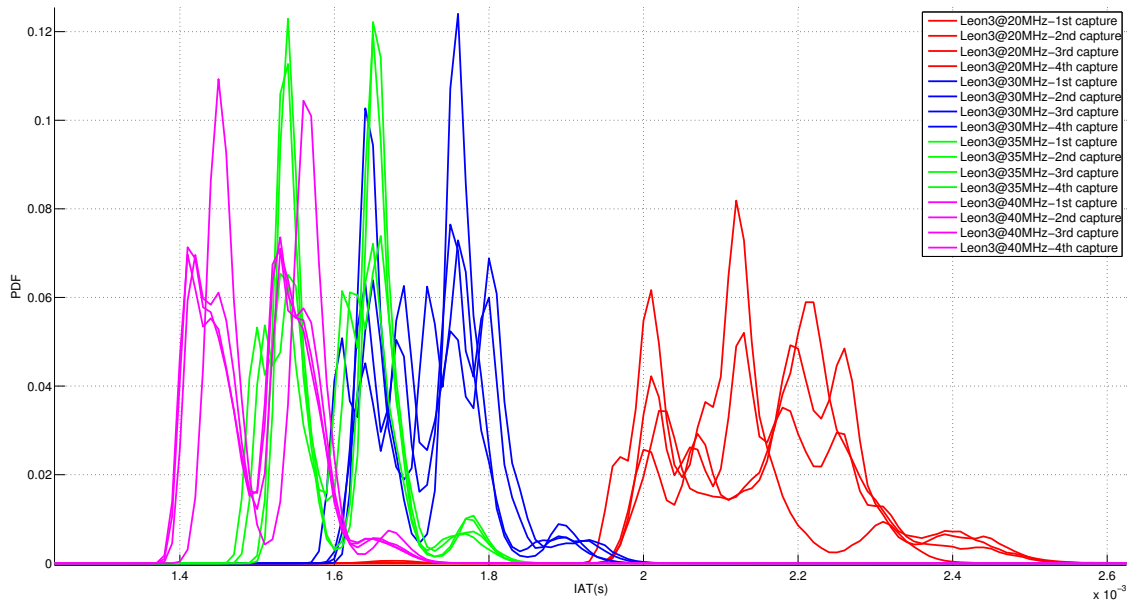


Figure 30: PDFs of ICMP(1400) IATs for clock variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

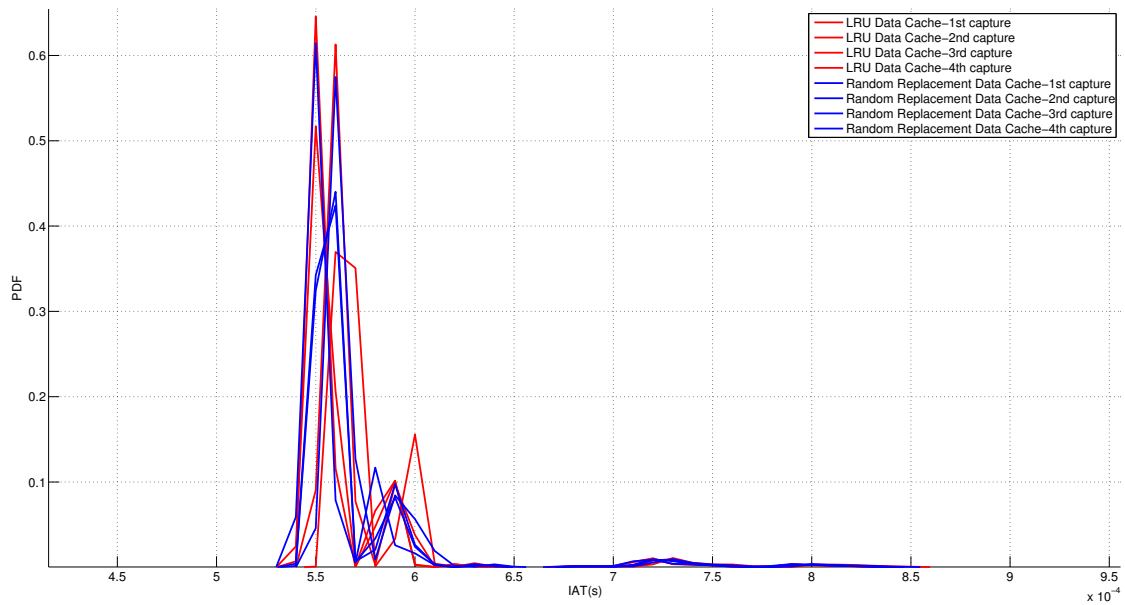


Figure 31: PDFs of UDP IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

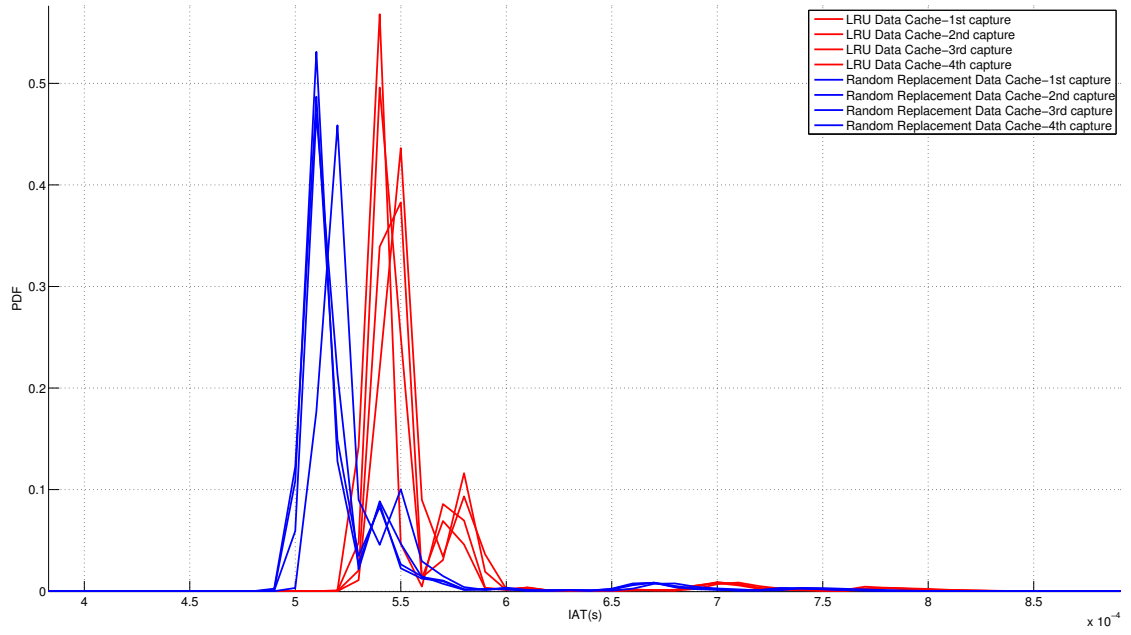


Figure 32: PDFs of UDP IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

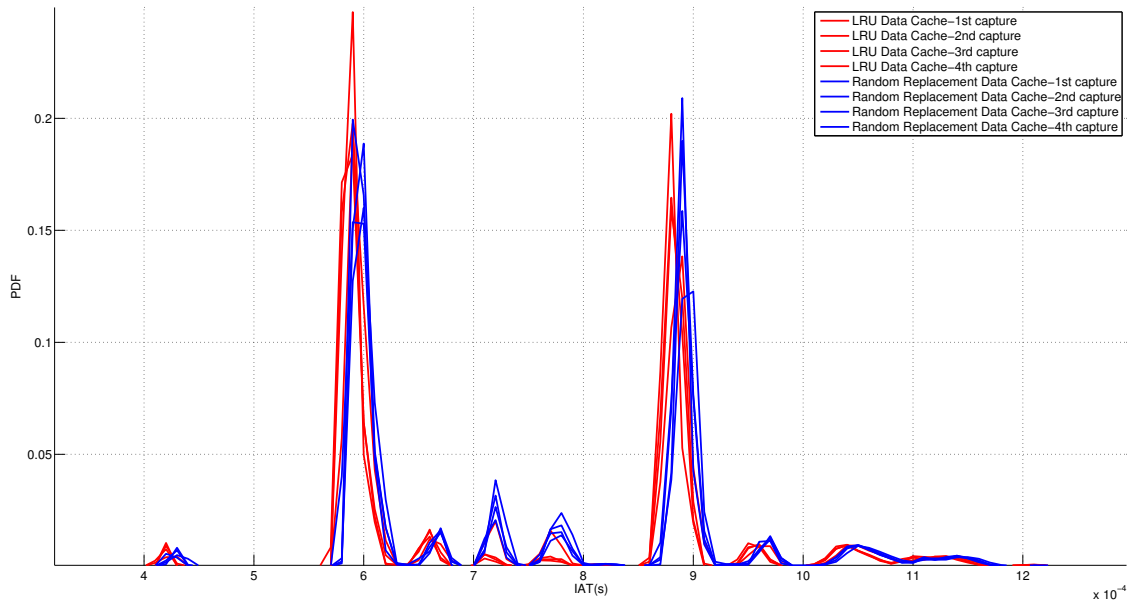


Figure 33: PDFs of TCP IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

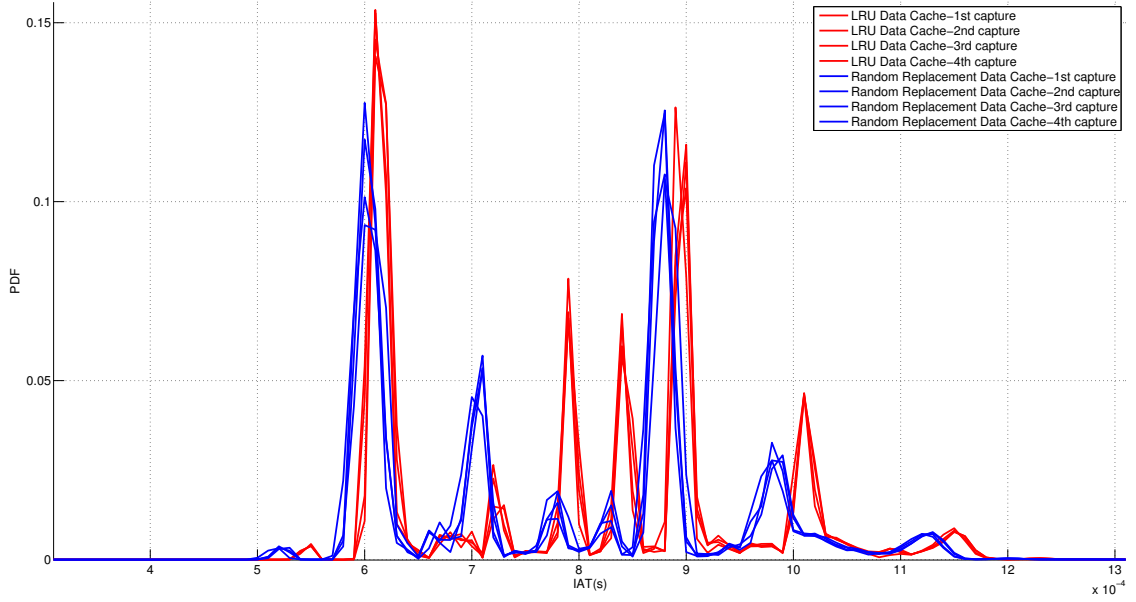


Figure 34: PDFs of TCP IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

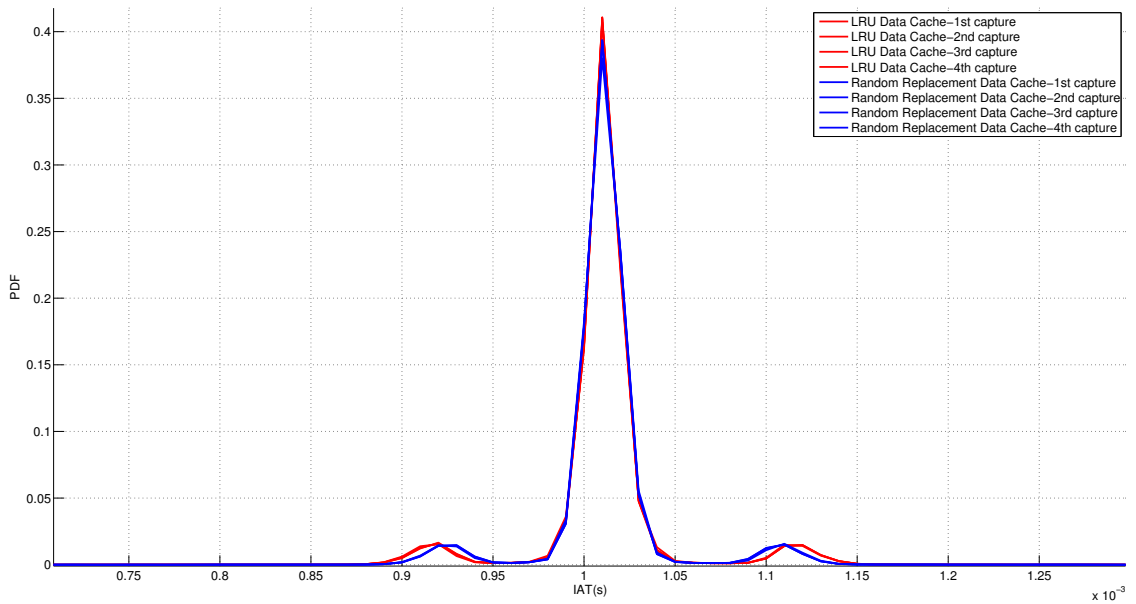


Figure 35: PDFs of ICMP reply IATs for data cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

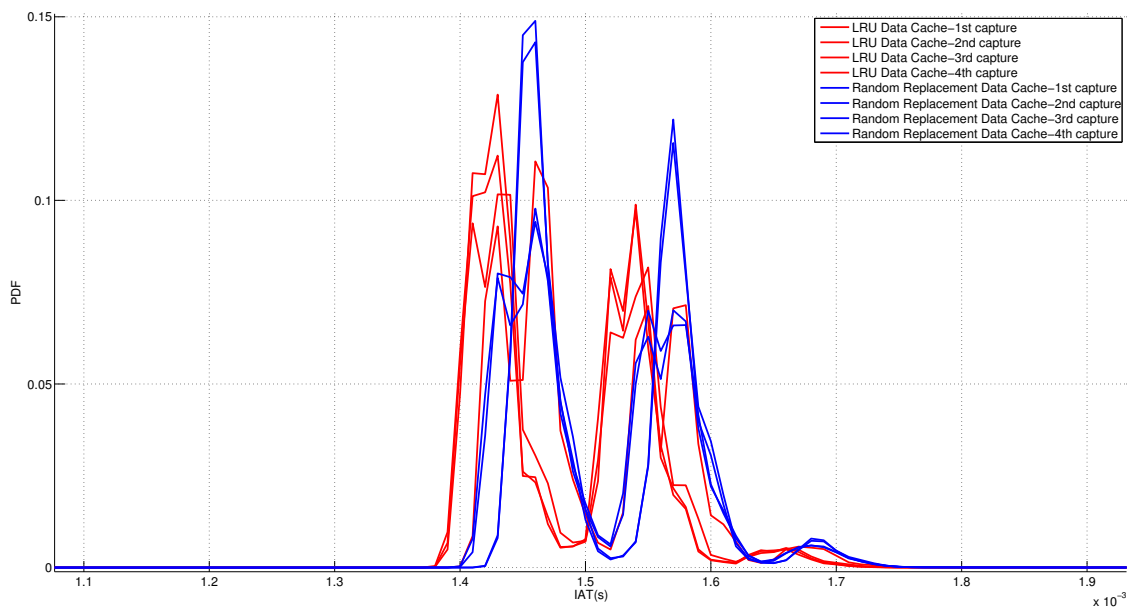


Figure 36: PDFs of ICMP (1400 bytes) reply IATs for data cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

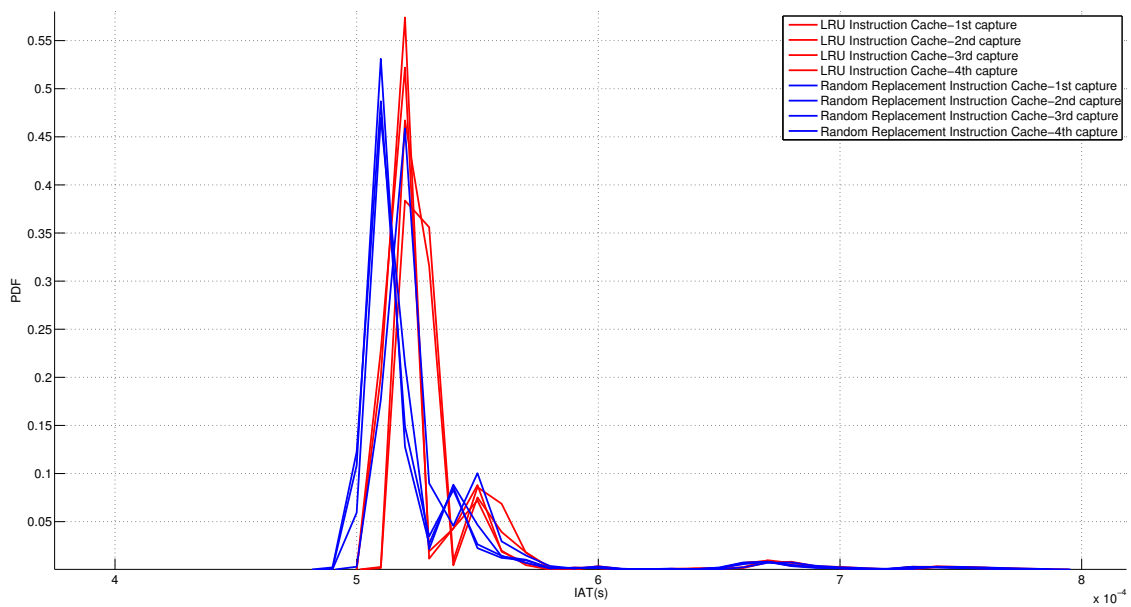


Figure 37: PDFs of UDP IATs for instruction cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

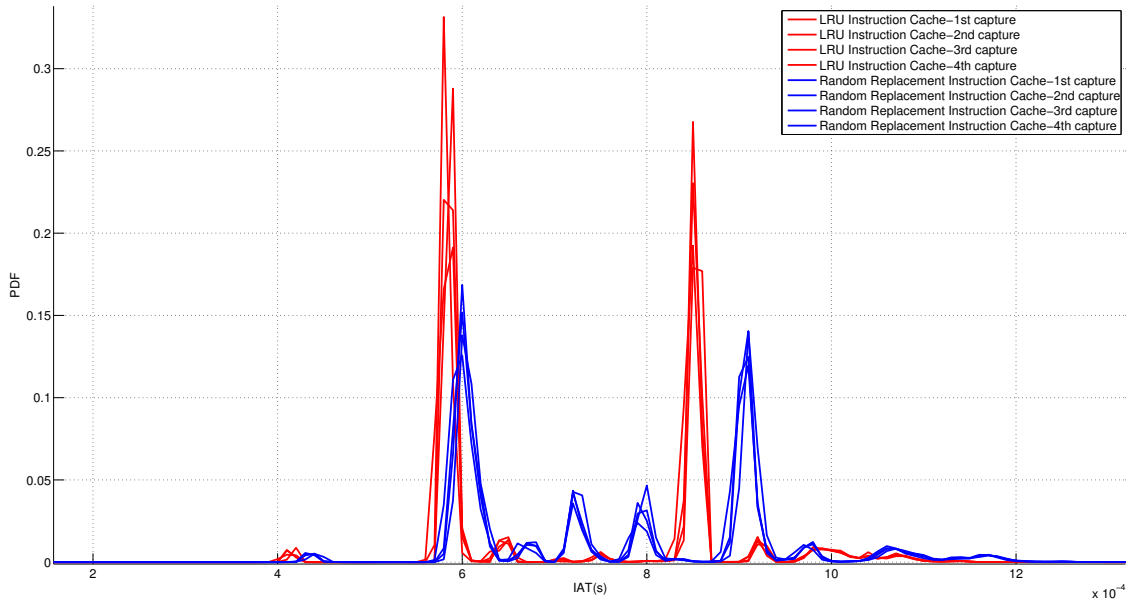


Figure 38: PDFs of TCP IATs for instruction cache variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

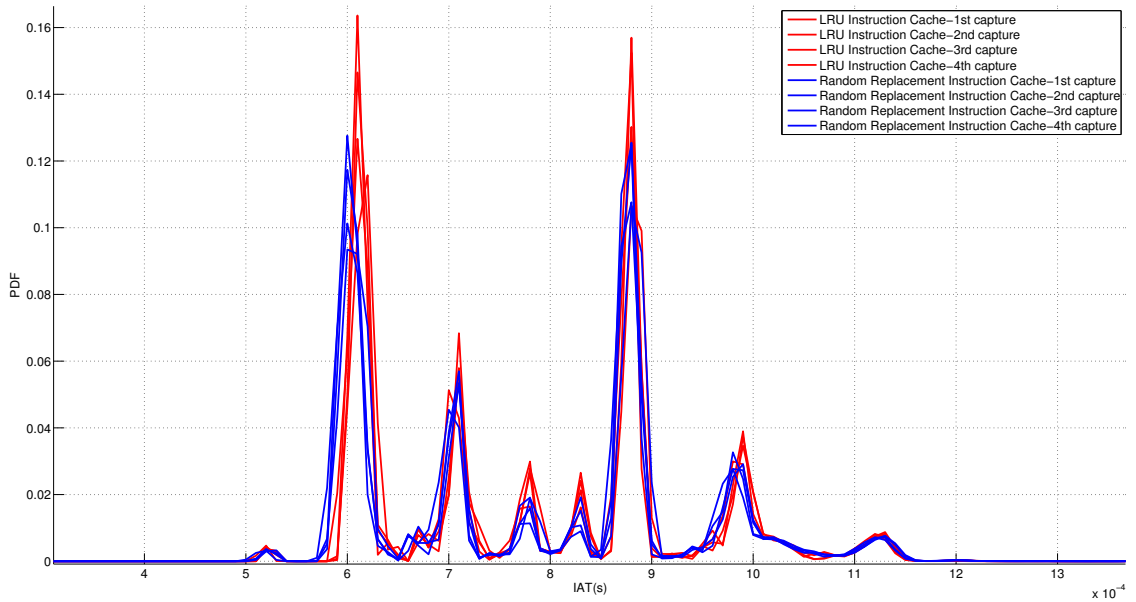


Figure 39: PDFs of TCP IATs for instruction cache variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

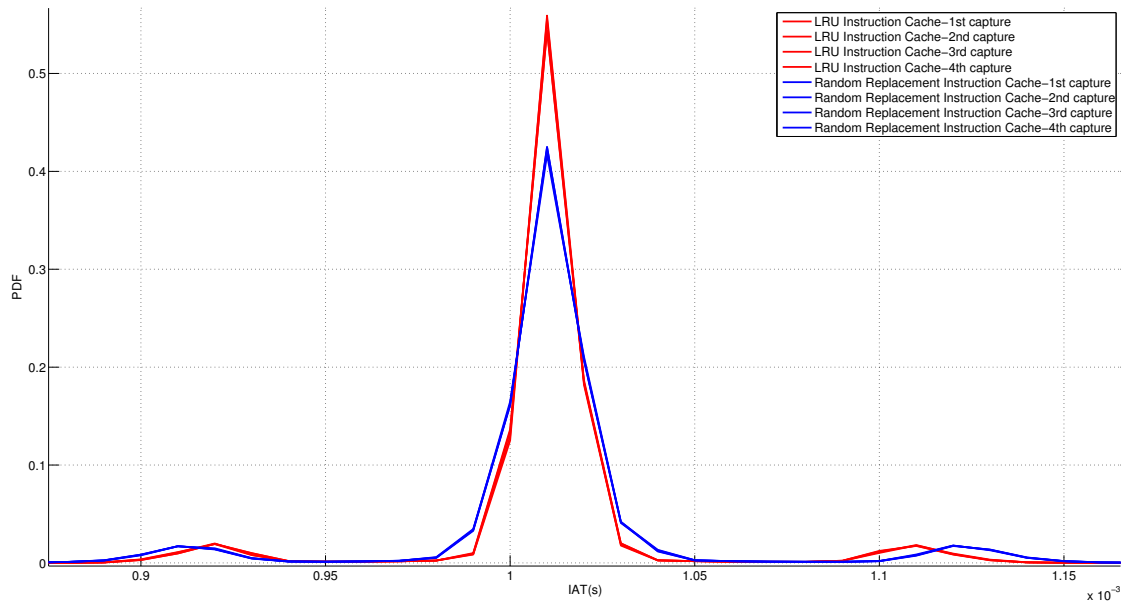


Figure 40: PDFs of ICMP (56 bytes) reply IATs for instruction cache variations on XC3S1500 with 1000 bins of 10 μ s width

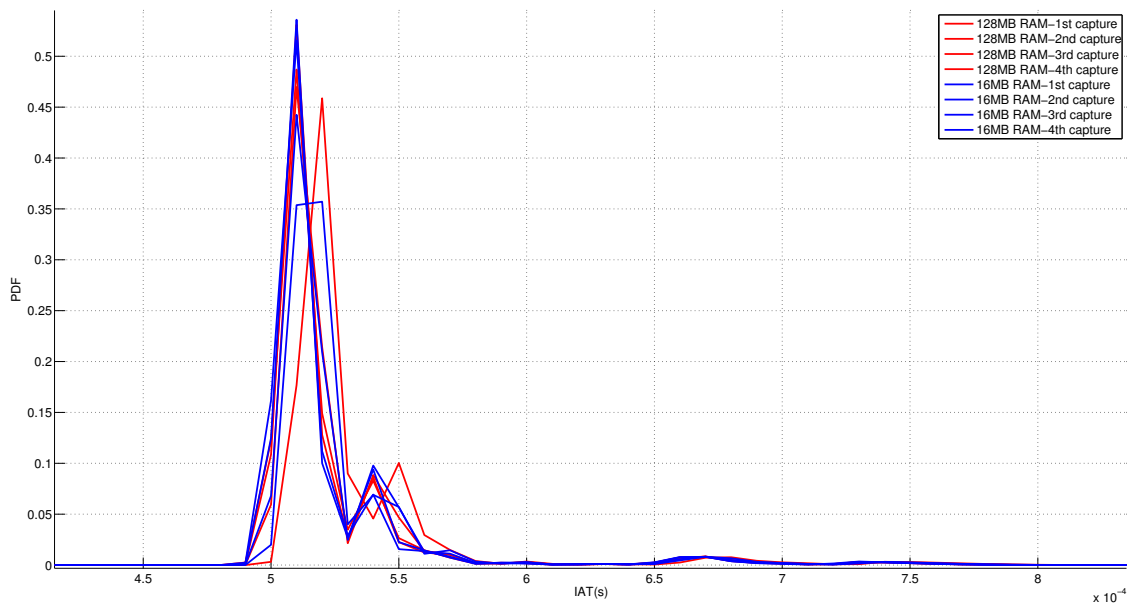


Figure 41: PDFs of UDP IATs for RAM variations on XtremeDSP with 1000 bins of 10 μ s width

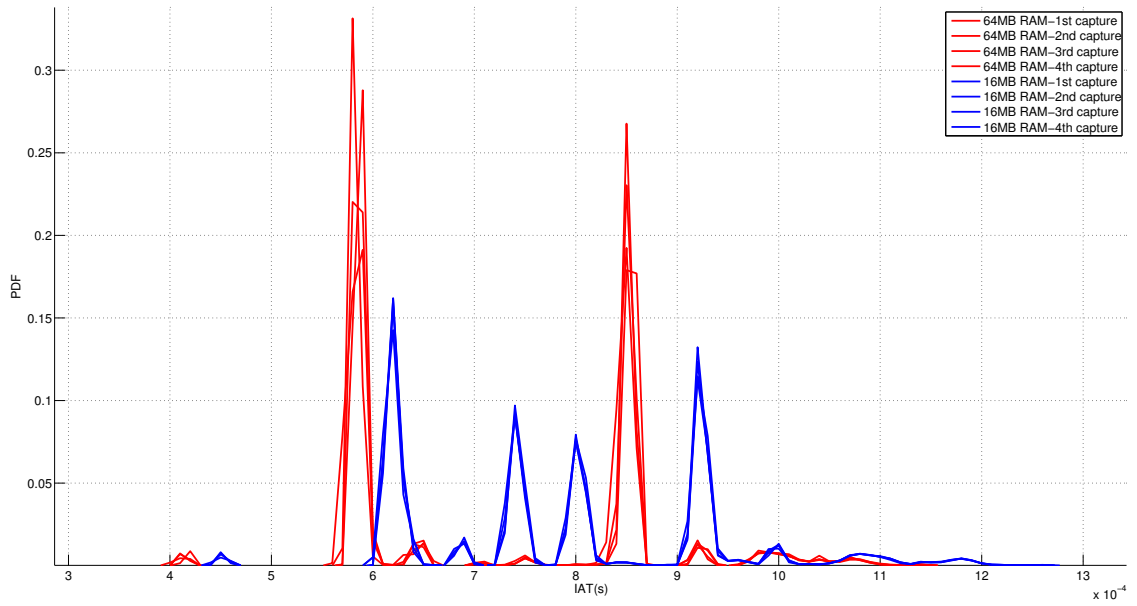


Figure 42: PDFs of TCP IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

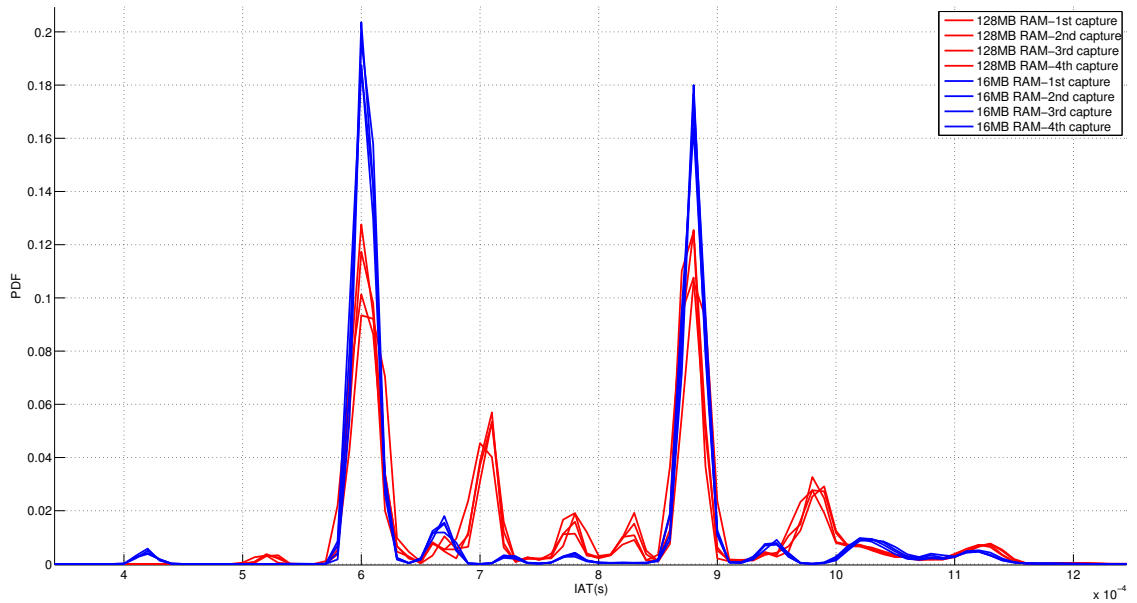


Figure 43: PDFs of TCP IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu\text{s}$ width

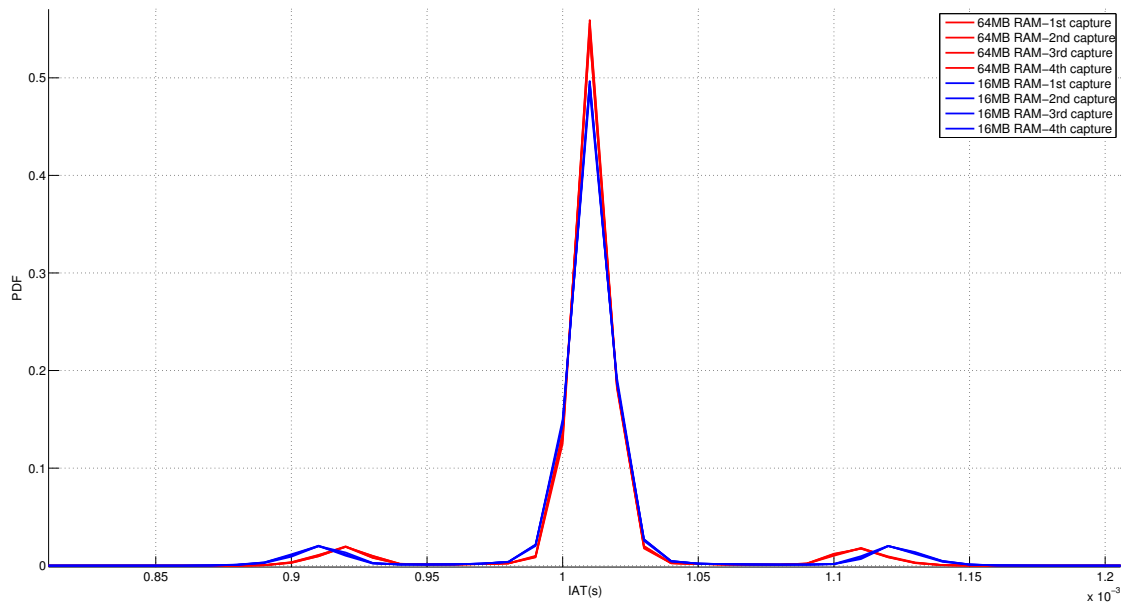


Figure 44: PDFs of ICMP (56 byte) reply IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

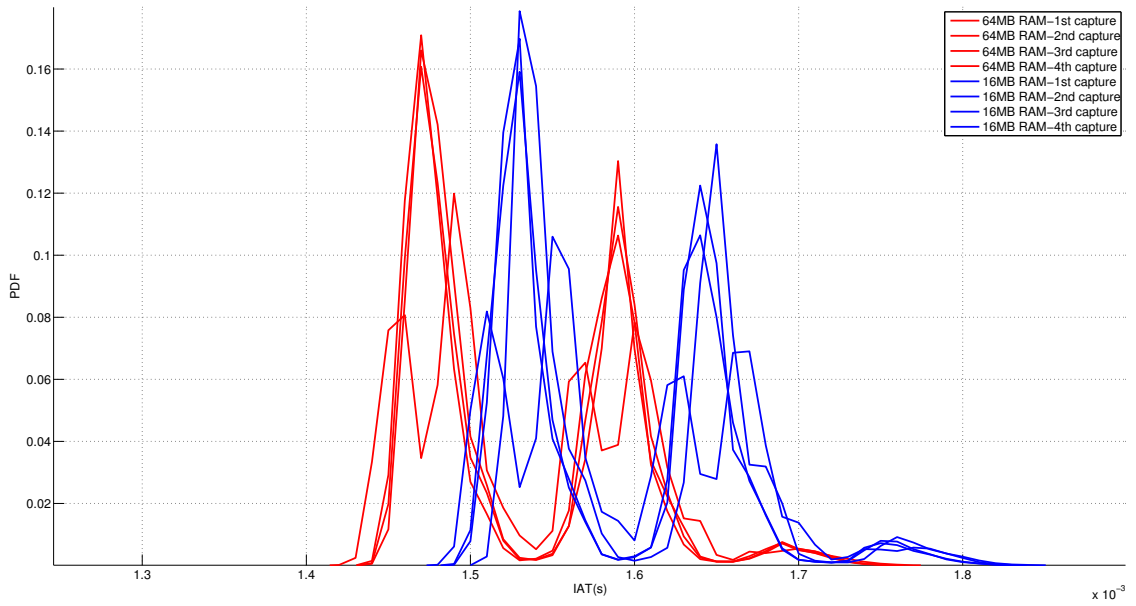


Figure 45: PDFs of ICMP (1400 byte) reply IATs for RAM variations on XC3S1500 with 1000 bins of $10\mu\text{s}$ width

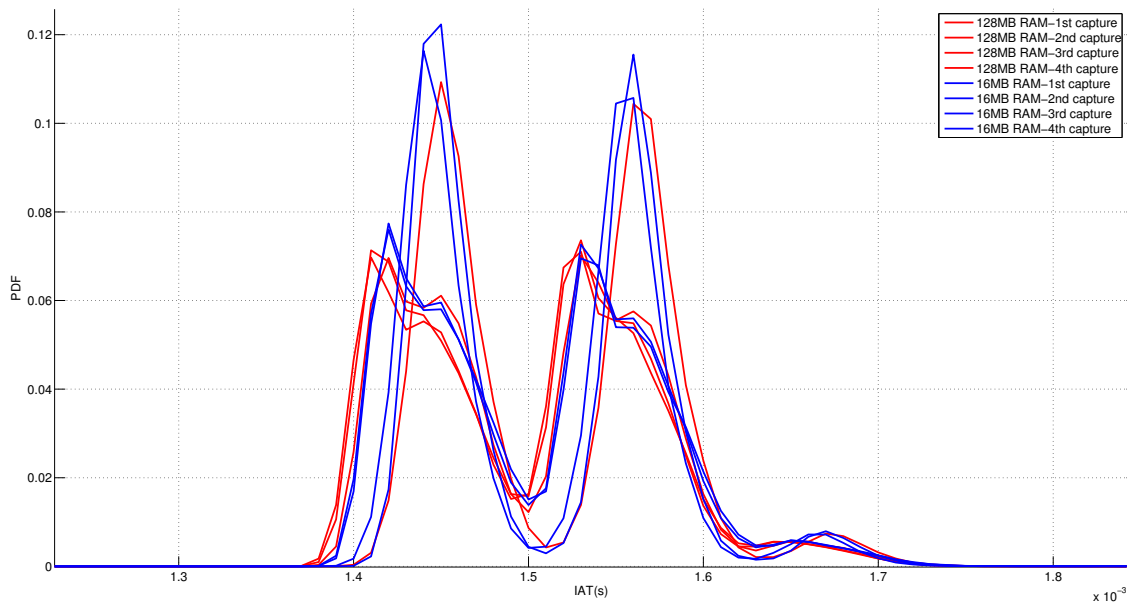


Figure 46: PDFs of ICMP (1400 byte) reply IATs for RAM variations on XtremeDSP with 1000 bins of $10\mu s$ width

APPENDIX B

COUNTERFEIT DETECTION

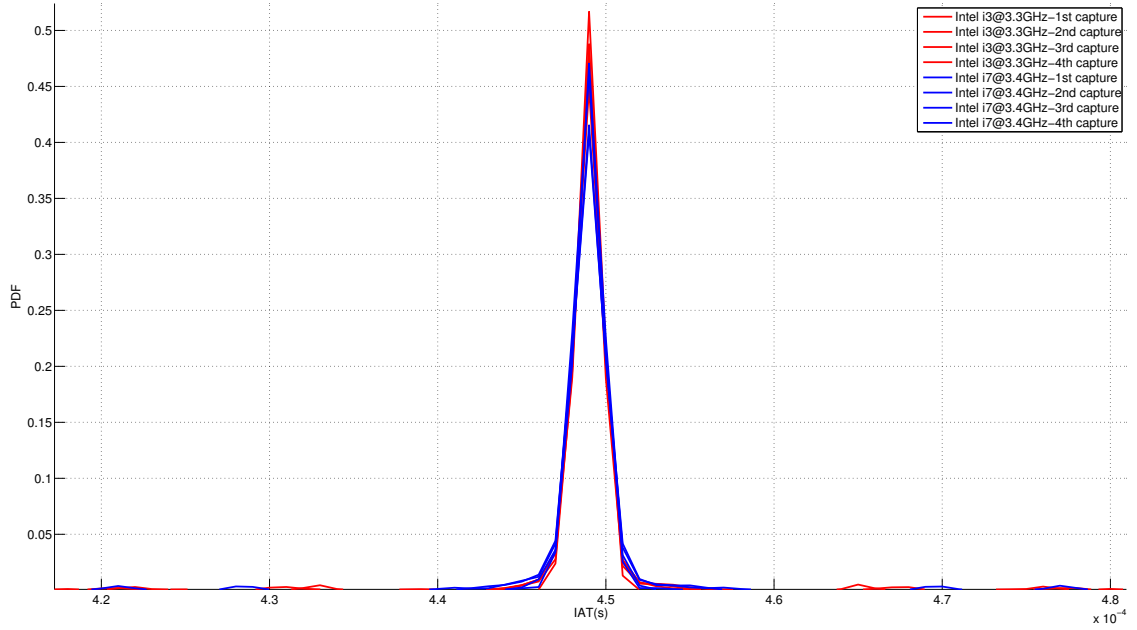


Figure 47: PDFs of UDP IATs for CPU variations on Optiplex 7010 with 1000 bins of $1\mu s$ width

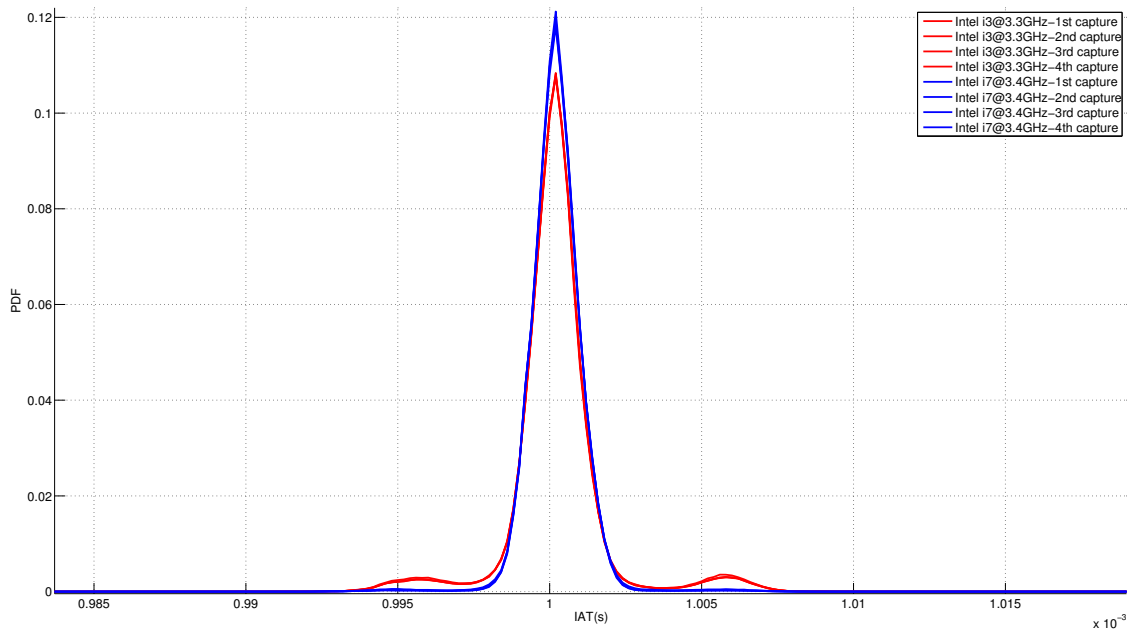


Figure 48: PDFs of ICMP (1400 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width

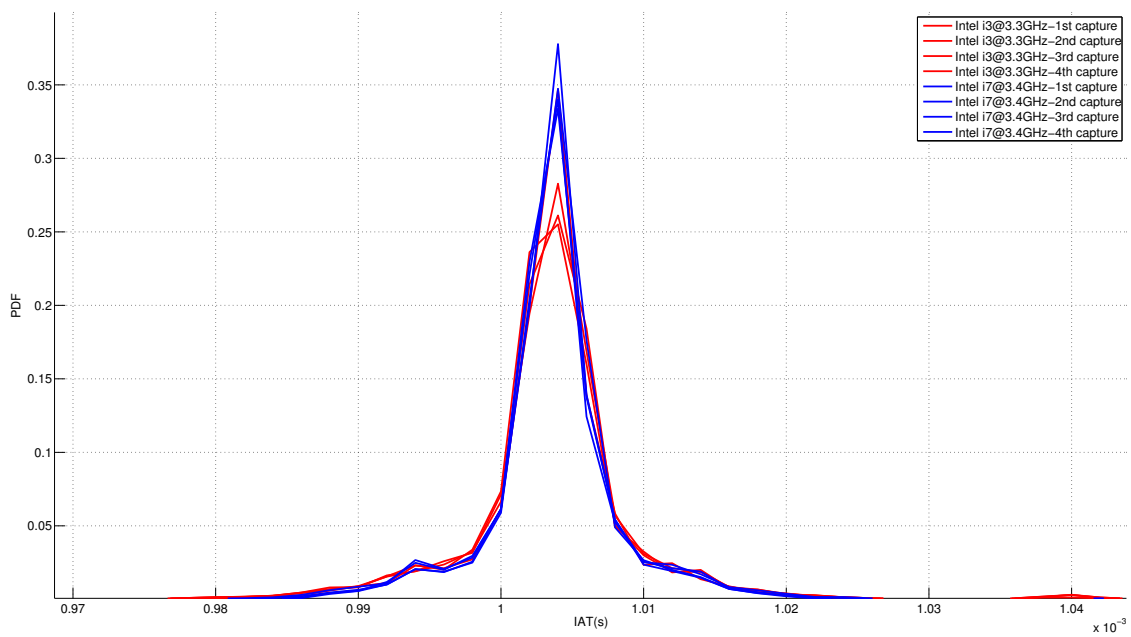


Figure 49: PDFs of ICMP (56 bytes) request IATs for CPU variations on Optiplex 7010 with 1000 bins of 2μs width

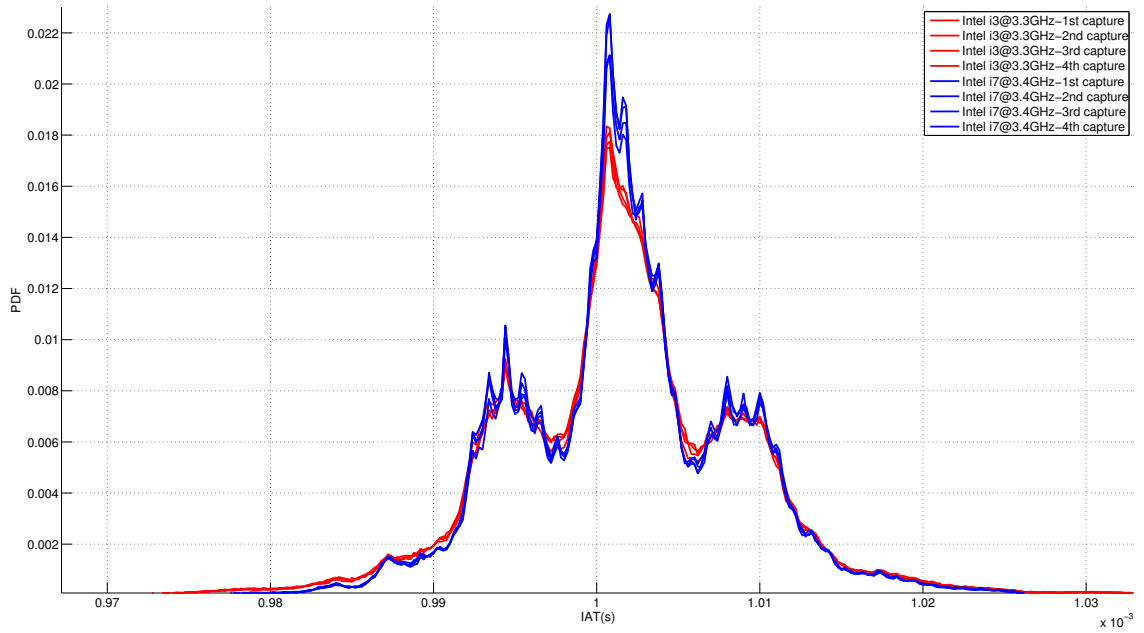


Figure 50: PDFs of ICMP (1400 bytes) request IATs for CPU variations on Optiplex 7010 with 10000 bins of 200ns width

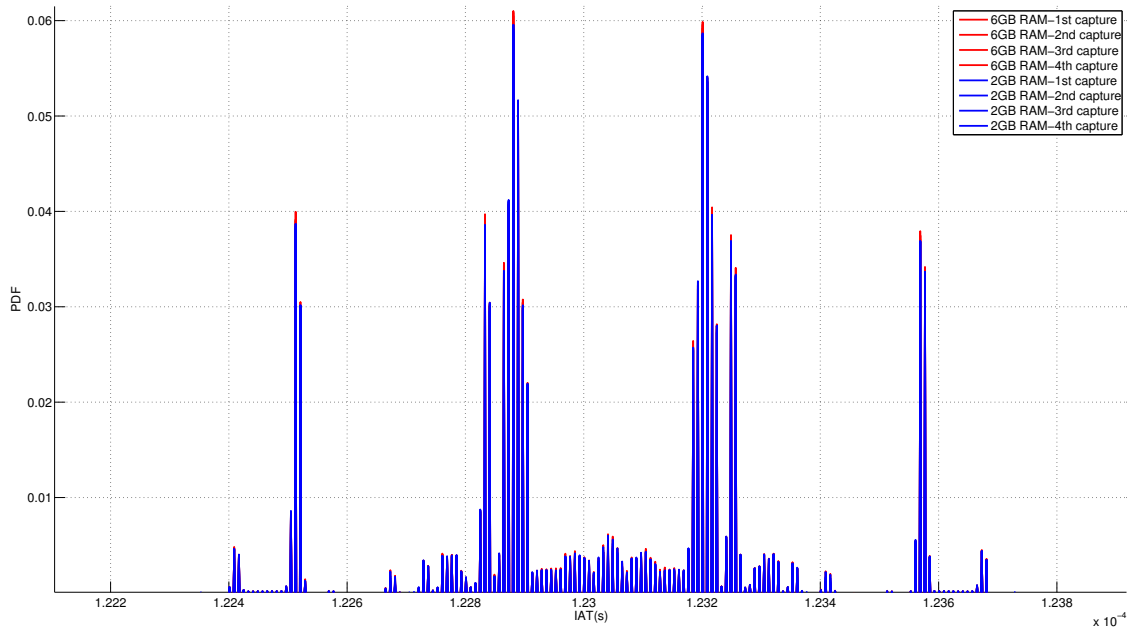


Figure 51: PDFs of TCP IATs for RAM variations on Optiplex 7010 with 1000000 bins of 1ns width

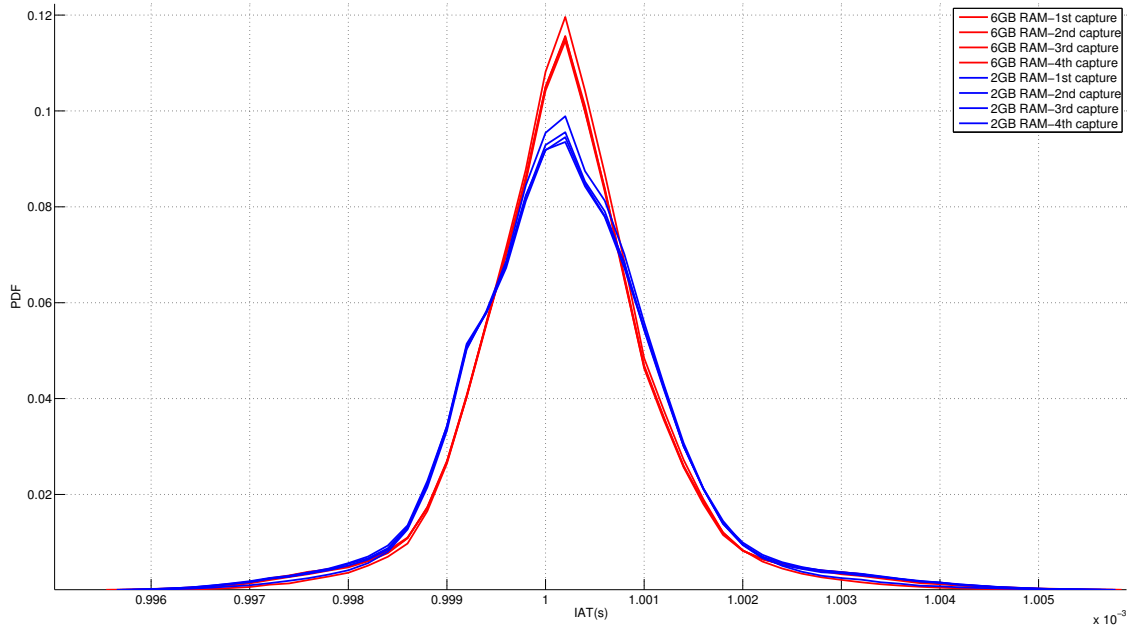


Figure 52: PDFs of ICMP (56 bytes) request IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width

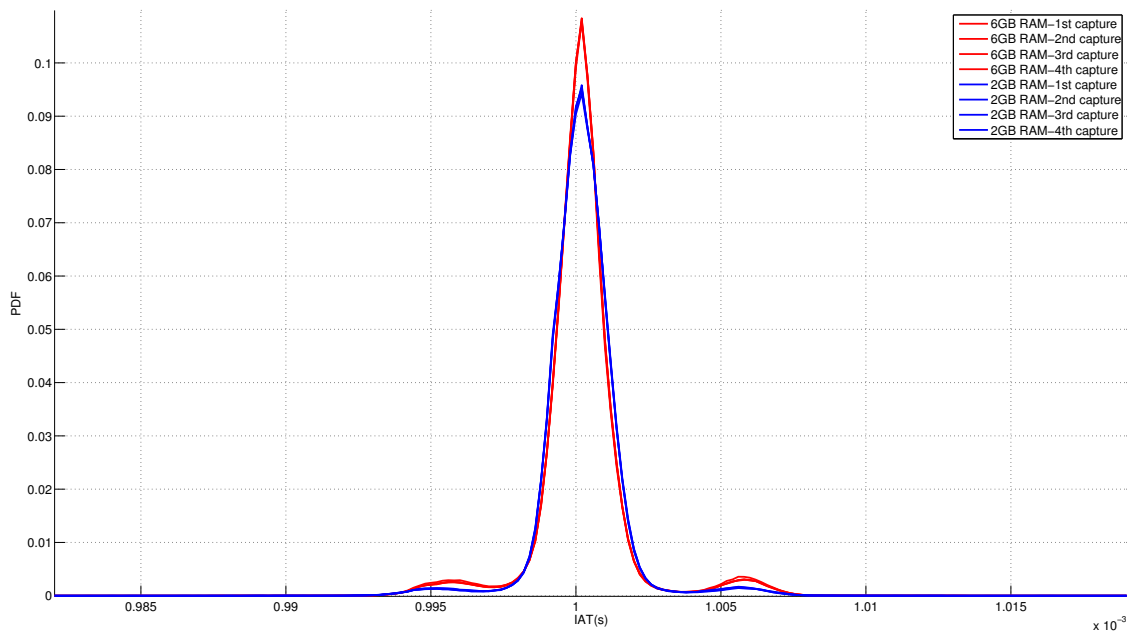


Figure 53: PDFs of ICMP (1400 bytes) request IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width

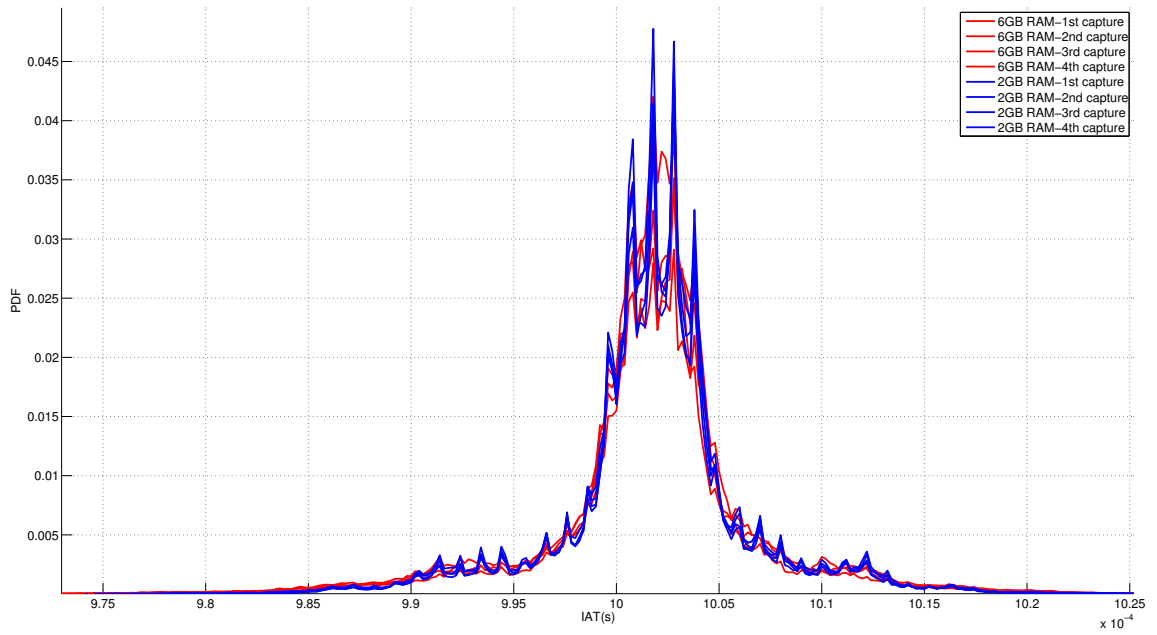


Figure 54: PDFs of ICMP (56 bytes) reply IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width

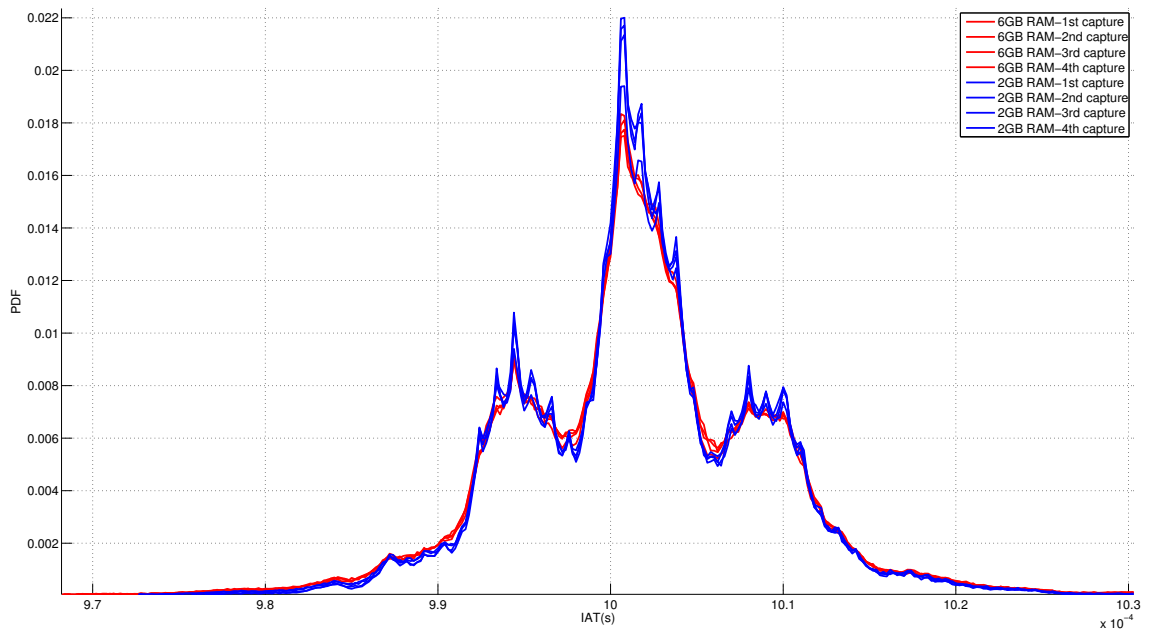


Figure 55: PDFs of ICMP (1400 bytes) reply IATs for RAM variations on Optiplex 7010 with 10000 bins of 200ns width

REFERENCES

- [1] V. Paxson, "End-to-end internet packet dynamics," in *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, 1999, pp. 277–292.
- [2] Z. Hu, K. Thulasiraman, and P. K. Verma, "Complex networks: Traffic dynamics, network performance, and network structure," *American Journal of Operations Research*, vol. 3, pp. 187–195, 2013.
- [3] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *IEEE/IFIP International Conference on Dependable Systems and Networks, 2010*, pp. 383–392.
- [4] B. Sieka, "Active fingerprinting of 802.11 devices by timing analysis," in *3rd IEEE Consumer Communications and Networking Conference, 2006*, vol. 1, pp. 15–19.
- [5] C. Neumann, O. Heen, and S. Onno, "An empirical study of passive 802.11 device fingerprinting," in *32nd International Conference on Distributed Computing Systems Workshops, 2012*, pp. 593–602.
- [6] W. R. Stevens, *TCPIP Illustrated, Volume 1- The Protocols*. Addison-Wesley Professional, 1993.
- [7] C.-C. T. C. E. Brian Grow and B. Burnsed, "Dangerous fakes," *Businessweek*, Tech. Rep., October 2008.
- [8] F. Koushanfar, S. Fazzari, C. McCants, W. Bryson, P. Song, M. Sale, and M. Potkonjak, "Can eda combat the rise of electronic counterfeiting?" in *49th ACM/EDAC/IEEE Design Automation Conference, 2012*, pp. 133–138.
- [9] C. on Armed Services, "The committee's investigation into counterfeit electronic parts in the department of defense supply chain," United States Senate, One Hundred and Twelfth Congress, First Session, Tech. Rep., November 2011. [Online]. Available: <http://www.gpo.gov/fdsys/pkg/CHRG-112shrg72702/html/CHRG-112shrg72702.htm>
- [10] W. Cobb, E. Garcia, M. Temple, R. Baldwin, and Y. Kim, "Physical layer identification of embedded devices using rf-dna fingerprinting," in *Military Communications Conference, 2010*, pp. 2168–2173.
- [11] F. McFadden and R. Arnold, "Supply chain risk mitigation for it electronics," in *Technologies for Homeland Security (HST), 2010 IEEE International Conference on*, Nov., pp. 49–55.
- [12] "Netfpga 1g," http://netfpga.org/1G_specs.html.
- [13] S. Marston, Z. Li, S. Bandyopadhyay, and A. Ghalsasi, "Cloud computing - the business perspective," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, Jan., pp. 1–11.

- [14] G. Lazarou, J. Baca, V. Frost, and J. Evans, “Describing network traffic using the index of variability,” *Networking, IEEE/ACM Transactions on*, vol. 17, no. 5, pp. 1672–1683, Oct.
- [15] S. Floyd and V. Paxson, “Difficulties in simulating the internet,” *Networking, IEEE/ACM Transactions on*, vol. 9, no. 4, pp. 392–403, Aug.
- [16] L. Watkins, W. Robinson, and R. Beyah, “A passive solution to the cpu resource discovery problem in cluster grid networks,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 12, pp. 2000–2007, Dec 2011.
- [17] L. Watkins, W. Robinson, and Beyah, “A passive solution to the memory resource discovery problem in computational clusters,” vol. 7, no. 4, Dec 2010, pp. 218–230.
- [18] *GR-XC3S-1500 Development Board*, http://www.pender.ch/docs/GR-XC3S-1500_user_manual_rev2-0.pdf, Pender.
- [19] *XtremeDSP Starter Platform*, http://www.xilinx.com/support/documentation/boards_and_kits/ug454_sp3a_dsp_start_ug.pdf.
- [20] *Leon3 Processor*, <http://gaisler.com/index.php/products/processors/leon3>, Aeroflex Gaisler.
- [21] *BusyBox*, <http://www.busybox.net/>.
- [22] *GRMON Debugger*, <http://www.gaisler.com/index.php/products/debug-tools/grmon>.
- [23] A. Babikyan, “Sharktools 0.1.5: Matshark and pyshark,” <http://www.mit.edu/~armenb/sharktools/>, 2010.
- [24] A. Kimery, “Pitfalls of counterfeit-part epidemic exposed,” *HSToday.US*, 2012.
- [25] “Defense industrial base assessment: Counterfeit electronics,” http://www.bis.doc.gov/defenseindustrialbaseprograms/osies/defmarketresearchrpts/final_counterfeit_electronics_report.pdf, 2010.
- [26] H. Livingston, “Preventing and detecting counterfeit electronic components in the supply chain,” http://www.nhjes.org/2011_Joint_Conference/presentations/2BNHJES_Livingston_Henry_Oct6%5B2%5D.pdf, 2010.
- [27] A. Kahng, D. Kirovski, S. Mantik, M. Potkonjak, and J. Wong, “Copy detection for intellectual property protection of vlsi designs,” in *IEEE/ACM International Conference on Computer-Aided Design, 1999*, pp. 600–604.
- [28] F. Koushanfar and M. Potkonjak, “Cad-based security, cryptography, and digital rights management,” in *44th ACM/IEEE Design Automation Conference, 2007*, pp. 268–269.
- [29] S. Wei, S. Meguerdichian, and M. Potkonjak, “Gate-level characterization: Foundations and hardware security applications,” in *47th ACM/IEEE Design Automation Conference, 2010*, pp. 222–227.
- [30] S. Wei, A. Nahapetian, and M. Potkonjak, “Robust passive hardware metering,” in *IEEE/ACM International Conference on Computer-Aided Design, 2011*, pp. 802–809.

- [31] A. Caldwell, H.-J. Choi, A. Kahng, S. Mantik, M. Potkonjak, G. Qu, and J. Wong, “Effective iterative techniques for fingerprinting design ip [vlsi cad],” in *Proceedings 36th Design Automation Conference, 1999*, pp. 843–848.
- [32] M. Majzoobi, F. Koushanfar, and M. Potkonjak, “Techniques for design and implementation of secure reconfigurable pufs,” *ACM Transactions on Reconfigurable Technology and Systems, article 5*, vol. 2, no. 1, 2009.
- [33] C. Torrioni, “X-ray imaging to detect counterfeits,” *SMT Magazine, Aug 2012*, pp. 44–47.
- [34] *Fake Intel CPUs at Newegg*, http://hardocp.com/article/2010/03/05/newegg_selling_fake_intel_cpus.
- [35] T. Wolf and M. Franklin, “Commbench-a telecommunications benchmark for network processors,” in *Performance Analysis of Systems and Software, 2000. ISPASS. 2000 IEEE International Symposium on*, 2000, pp. 154–162.