1

Network Processor Experiments

1.1 Introduction

Chapter 21 in the text *Hands-On Networking With Internet Technologies* (Comer, Prentice Hall, 2001, ISBN 0-13-048003-7) describes a network systems engineering lab that uses testbed facilities for network processors available from Intel Corporation. This chapter discusses experiments that can be conducted in such a lab. All the experiments assume that the lab uses Intel's IXP 1200 testbed and the associated Software Development Kit that includes a cross-compiler and download facilities. Although similar experiments may be possible with facilities from other vendors, the details will differ.

Name: _____

Experiment 1.1

Configure your Account

Purpose

To set up your account so you can program a network processor.

Background Reading And Preparation

Read documentation for account configuration specific to your lab.

Overview

Download network processor support software and up your account configuration files to enable you to compile a simple network processor. application.

Procedure And Details (checkmark as each is completed)

_____ Obtain and read the instructions for configuring your personal account to compile network processor applications.

_____ Obtain a copy of the software source files necessary to build network processor applications for your network processor. Extract these files into an appropriate location in your account.

_____ Configure your account environment and start-up scripts according to the lab-specific instructions for your facility.

_____ Compile a simple network processor application to test that your development environment is configured correctly.

Optional Extensions (checkmark options as they are completed)

_____ Try compiling several network processor applications that require different development tools.

_____ Run a network processor application on a network processor.

Name: _____

Experiment 1.2

Compile and Download Code to a Network Processor

To learn how to compile network processor applications, download them to a network processor, and test them using lab facilities.

Background Reading And Preparation

Read the directions on compiling a network processor application and accessing a network processor in in your lab. Also consult your lab instructor as to how to how to send packets to your network processor.

Overview

Compile the code for a network processor application that counts packets, load it onto a network processor and test it by sending packets to it.

Procedure And Details (checkmark as each is completed)

_____ Download the code for a simple network processor application that counts packets on a network.

_____ Compile the packet counting application.

_____ Upload the packet count application to the network processor along with any necessary configuration files.

_____ Configure your network so you can send packets to the network processor.

_____ Run the packet count application and transmit packets to the network processor. Make sure the network processor counts the correct number of packets.

Optional Extensions (checkmark options as they are completed)

_____ Try compiling and running other network processor applications such as programs to bridge Ethernet segments or route IP packets.

_____ Stress test the packet count application by flooding the network with as many packets as possible. How well can the application keep up with the network traffic.

Name: _____

Experiment 1.3

Build a Packet Analyzer

Purpose

To learn how to build a packet analyzer on a network processor.

Background Reading And Preparation

Review Ethernet, IP, and TCP headers (chapters 9, 20, and 24 in Computer Networks and Internets and IETF RFCs 791 and 793). Also read the the documentation for the simplified network processor API.

Overview

Build an application that captures packets from the network and analyzes them.

Procedure And Details (checkmark as each is completed)

_____ Write an application that reads a set of packets and analyzes them. When a user enters the command "send_command summary"† while logged into the network processor, the application should produce the following summary:

Layer 2 Summary (Ethernet)

- Total number of frames processed
- Average frame size (excluding header)
- Number and percentage of broadcast frames
- Number and percentage of multicast frames
- Number and percentage of unicast frames
- Number and percentage of each of the top five frame types

Layer 3 Summary (IP)

- Total number of datagrams processed
- Average datagram size (excluding header)
- Number and percentage of datagram fragments
- Number and percentage of datagrams sent to network broadcast address
- Number and percentage of datagrams sent to limited broadcast
- Number and percentage of datagrams carrying TCP
- Number and percentage of datagrams carrying UDP
- Number and percentage of datagrams carrying ICMP

Layer 4 Summary (TCP)

Total number of TCP segments processed
Average segment size (excluding header)
Number and percentage of acknowledgements (no data)
Number and percentage of data segments
Number and percentage of SYN/FIN segments
Number and percentage of each of top five destination ports
Number and percentage of each of top five source ports

_____ Use two computers and a network processor to test the application. Connect the two computers and the network processor to a private LAN. Generate traffic between the computers using applications such as telnet, ping and traceroute. Use tcpdump or an equivalent program to capture the traffic and check that your application produces the correct output.

_____ Modify the application so that it can display the headers of the packets it analyzes. When a user enters the command "send_command verbose" the application should print out the field contents of each header it parses (i.e. Ethernet, IP and TCP) in each packet it subsequently receives. Disable this feature when the user enters the command "send_command quiet".

Optional Extensions (checkmark options as they are completed)

_____ Modify the application to take arguments which specify which packets to examine. When a user enters a command of the form "send_command filter <pattern>" the application only analyzes (and prints if in verbose mode) packets matching the pattern. For example, the pattern "-ip" may cause the application to only analyze and print IP packets. It should also be possible to limit processing to frames sent by a specified computer. When issued the command "send_command nofilter" all packets should be analyzed.

_____ Extend your program to allow boolean combinations of pattern options. For example it should be possible to limit processing to "frames carrying IP that also contain TCP, or ARP frames."

Notes

†See the simplified API manual for instructions on how to program your ACE to handle input from *send_command* program.

Name: _____

Experiment 1.4

Build an Ethernet Bridge

Purpose

To learn how to forward Ethernet frames according to the Ethernet bridging algorithm.

Background Reading And Preparation

Read Section 5.8 of Network System Design with Network Processors to learn about the bridge algorithm.

Overview

Build an application on a network processor which bridges two Ethernet segments using the simplified API.

Procedure And Details (checkmark as each is completed)

_____ Build an application which forwards packets between two ports of an IXP1200 (i.e. when it receives a frame on one port it sends it out the other).

_____ Connect an IXP1200 to two Ethernet segments, and a host to each Ethernet segment. Test that your application works by running ping, telnet or traceroute between the two hosts.

_____ Connect an IXP1200 to two Ethernet segments, and a host to each Ethernet segment. Test that your application works by running ping, telnet or traceroute between the two hosts.

_____ Connect an IXP1200 to two Ethernet segments, and a host to each Ethernet segment. Test that your application works by running ping, telnet or traceroute between the two hosts.

_____ Add a module to your application which extracts the Ethernet source address from each frame and builds a table mapping Ethernet addresses to the Ethernet segments on which the address resides. Have the application print out this table when a user enters the command "send_command showtable".

_____ Modify your application so it only forwards Ethernet frames according to the Ethernet briding algorithm. Be sure to treat broadcast and multicast addresses differently than regular Ethernet addresses.

_____ Connect an IXP1200 to two Ethernet segments. Connection two hosts to one segment, and a third to the other. Send traffic between the three hosts. Use tcpdump or a program like it on each Ethernet segment to verify that your bridge is forwarding traffic correctly.

Optional Extensions (checkmark options as they are completed)

_____ Extend your bridge so that it can forward traffic according to the bridge algorithm between more than two Ethernet ports. (e.g. the extend the bridge so it bridges 4 Ethernet segments)

_____ Extend your bridge so that it can forward traffic according to the bridge algorithm between more than two Ethernet ports. (e.g. the extend the bridge so it bridges 4 Ethernet segments)

_____ Implement the Distributed Spanning Tree protocol for avoiding bridging loops in your application.

_____ Stress test your bridge. See if your bridge can forward packets with minimal (or no) loss when a host on one segment floods the network with frames for the other segment. (Use a packet analyzer on each segment to count the frames sent on each.) See if you can reduce the amount of loss.

Notes

Name: _____

Experiment 1.5

Build an IP Fragmenter

Purpose

To learn how to fragment IP datagrams on a network processor.

Background Reading And Preparation

Read Section 5.9 in Network System Design with Network Processors for information on IP fragmentation and reassembly. Consult RFC 791 to learn more details of IP header fields used in fragmentation, and RFC 815 for information about reassembly.

Overview

Construct and test a network processor application that fragments IP datagrams.

Procedure And Details (checkmark as each is completed)

_____ Create an application using the simplified API which forwards frames between two Ethernet segments. Have the application forward all frames except IP datagrams that are larger than some fixed MTU (e.g. 128). (Do not count the Ethernet header when computing the size of the IP datagram.) The application should drop frames containing IP packets that exceed the chosen MTU. Allow the MTU to be set using the "send_command" application.

_____ Modify your application so that instead of dropping IP frames that exceed the (artificial) MTU, it fragments them according to IP fragmentation rules. Your application should be able to fragment frames that contain both whole and previously fragmented IP datagrams.

_____ Test your application by connecting two hosts to independant Ethernet segments and connecting those Ethernet segments to the network processor. Send large IP or UDP packets from one host to the other using either Ping, Netcat, or an equivalent program and check that they are recieved properly on the destination host.

_____ Try sending packets of various sizes including: packets less than the MTU of both Ethernet and the fragmentor, packets smaller than the IP-over-Ethernet MTU (1460 bytes) but larger than the fragmentor MTU, and packets larger than both MTUs. Change the MTU on your application and test the application again.

Optional Extensions (checkmark options as they are completed)

_____ Extend your fragmentor so that it sends the appropriate ICMP message back to the source host when it would fragment a datagram with the Don't Fragment bit set. (It should also drop the original datagram.)

_____ Have your fragmentor implement PATH MTU discovery. When it fragments a datagram have it also set the Don't Fragment bit in the datagram. If it receives an ICMP Destination Unreachable datagram with a Code of 4 (framentation needed and DF bit set) the application should make an entry in a table for the destination address of the datagram and fragment all subsequent datagrams to that destination at a smaller MTU. If it receives subsequent ICMP messages for a given destination it should continue to decrement the MTU for that destination.

_____ Create a network processor application that performs IP reassembly.

Notes

Name: _____

Experiment 1.6

Build a Traffic Classifier

Purpose

To learn about optimized traffic classification and classification languages.

Background Reading And Preparation

Read Chapter 16 of Network System Design Using Network Processors to learn about classification languages. Determine which classification language to use for this lab.

Overview

Create network processor code which classifies packets according to the data they carry up through application layer protocols.

Procedure And Details (checkmark as each is completed)

_____ Write network processor code to classify incoming packets into each of the following categories labelling each packet with the specified numeric tag. Have the network processor drop packets that don't fall into given classes. If a packet falls into more than one class, place it in the class with the lowest number.

Description	Tag
Any IP datagram to 10.1.2.3	1
TCP control (SYN/FIN/RST) to/from port 21	2
TCP control (SYN/FIN/RST) to/from port 22	3
TCP control (SYN/FIN/RST) to/from port 25	4
TCP control (SYN/FIN/RST) to/from port 80	5
TCP bulk data	6
RIP requests	7
UDP to port 139	8
UDP	9
ICMP	10
IP with options	11
ARP	12

_____ Augment your classifier program so that it reports packets and their classifications as it recieves them.

_____ Create test traffic which contains packets in each of the above traffic classes. Include packets that overlap some traffic that belongs to two or more classes.

_____ Load your classifier onto a network processor for testing. Send the packets you generated to the network processor and check that your program classifies them correctly.

Optional Extensions (checkmark options as they are completed)

_____ Stress test your classifier by transmitting large volumes of packets at line rate if possible to a network processor running your classifier.

_____ Extend your classifier into a firewall by adding explicit rules as to whether to drop or forward packets in each class.

_____ Modify your classifier so that it places packets that are destined to a list of IP addresses in class 1. Write code to modify the list of IP addresses in class 1 at runtime.

Notes

Name: _____

Experiment 1.7

Create a Bare-bones MicroACE

Purpose

Learn how to create a basic MicroACE.

Background Reading And Preparation

Read chapters 22 through 24 of Network System Design to learn about MicroACEs and the basics of how they are programmed. Read the Intel IXP1200 Programmer's Reference Manual to learn the microcode of the IXP1200.

Overview

Construct a MicroACE that counts frames.

Procedure And Details (checkmark as each is completed)

_____ Obtain a basic MicroACE from your lab instructor which initializes the ACE but drops all frame that come to it. The StrongARM component should also include a mechanism for receiving commands from applications. Learn how the code operates. Try compiling and running the MicroACE.

_____ Modify the microengine component so that instead of dropping all frames it receives from an Ethernet port, it raises them as exceptions to the StrongARM component.

_____ Alter the StrongARM component so that it keeps count of the number of frames raised as exceptions, and prints that number when issued a command from a command-line application. Have the StrongARM component drop all frames after counting them.

_____ Test your ACE by connecting the IXP1200 to an Ethernet LAN and generating traffic on that LAN. Check that the MicroACE counts packets correctly.

Optional Extensions (checkmark options as they are completed)

_____ Create your own cross-call mechanism to allow arbitrary applications to retrieve the frame counts from your MicroACE.

_____ Extend the MicroACE so that each frame is counted by the microengines instead of the StrongARM. The frame counts should be kept in a table in memory shared with the StrongARM core so the StrongARM can print the frame counts upon application request.

Extend the microengine-based counting mechanism even further so that it also counts different types of frames such as IP, ARP, TCP, TCP destined for a particular port, IP fragments, etc...

Notes

Name: _____

Experiment 1.8

Create a Classifier Microblock

Purpose

Learn to analyze packets in microcode.

Background Reading And Preparation

Read Chapters 24 in Network System Design with Network Processors for an overview of microcode. See Chapter 26 for an example ACE. Also read the Intel IXP1200 Programmer's Reference manual for microcode specifics.

Overview

Create a microblock that classifies packets according to their network headers.

Procedure And Details (checkmark as each is completed)

_____ Create a microblock to examine a frame received using the Intel Dispatch Loop macros and classify the frame into one of five classes based on its headers. The microblock should consist of two macros : Classify_Init[] and Classify[]. The Classify[] macro should set its one parameter to the class of of each frame it inspects. The Classify_Init[] macro should initialize any global data structures needed for Classify[]. The classes are:

Class	Eth Src Port	IP?	TCP to Port 80?	SYN|FIN|RST?
1	1 or 2	X	X	X
2	0	NO	X	X
3	0	YES	NO	X
4	0	YES	YES	NO
5	0	YES	YES	YES

_____ Obtain a sample MicroACE from your lab instructor whose dispatch loop calls the Classify microblock you wrote, and whose StrongARM component prints the first 60 bytes of each frame along with the exception code for every frame it receives. Assemble the MicroACE with your microblock.

_____ Test your code by attaching ports 0, 1 and 2 of your IXP1200 to three separate Ethernet LANs and generating traffic on each. Check the specifications returned by Classify[] (and passed to the StrongARM) against the values in the frame headers.

Optional Extensions (checkmark options as they are completed)

_____ Extend the microblock to match against arbitrary patterns in the header of each frame. Have the microblock read the patterns from memory.

_____ Extend the microblock further to read the patterns from scratch memory before testing each frame header. Modify the StrongARM component to accept patterns via crosscalls to install into scratch memory at runtime.

Notes

Name: _____

Experiment 1.9

Microengine Frame Forwarding Code

Purpose

To learn how to forward packets in an IXP1200.

Background Reading And Preparation

Read about transmitting frames in microcode on the IXP1200 in Sections 25.14 and 25.15 of Network System Design with Network Processors. See Chapter 26 for an example ACE. Also read the Intel IXP1200 Programmer's Reference Guide for microcode details.

Overview

Create a MicroACE which forwards packets according to a classification tag.

Procedure And Details (checkmark as each is completed)

_____ Learn how to modify the destination port of a frame from microcode.

_____ Create microcode for a dispatch loop to run a microblock and take different packet handling actions based on the value placed in dl_next_block register by the microblock.

dl_next_block	Action
IX_BUFFER_NULL	Drop
1	Set Dst Port 0. Send to next microblock.
2	Raise frame as exception #1.
3	Set Dst Port 1. Send to next microblock.
4	Set Dst Port 2. Send to next microblock.
5	Raise frame as exception #2.

_____ Create a StrongARM exception handler which performs one of two actions on the exception code.

Exception Code	Action
1	Copy the frame. Set the Dst Port of the first frame to 1, and the second frame to 2. Send both frames to the default target.
2	Set the Dst Port to 2. Send the frame to the default target.

_____ Obtain a MicroACE which sets the dl_next_block value according to a Classify[] macro. Insert your microcode into the dispatch loop after the Classify[] macro, and your exception handler in place of the default exception handler of the MicroACE's StrongARM component.

_____ Compile and test the MicroACE. Connect an IXP1200 to three separate LANs and generate traffic on each. Use the application command to change the classifcation tag for each incoming frame. Check that frames get forwarded appropriately for each tag.

Optional Extensions (checkmark options as they are completed)

_____ Create your own classifier microblock (see Lab 6) and use it in place of the MicroACE shell.

_____ Extend the exception handling code for type 2 exceptions so that it maintains state about open web connections that pass across the network processor. Have the ACE report the number of open web connections via a crosscall.

Notes