ECE 697J Fall 2003:
Lab 2 – IPv4 Forwarding and Classification on IXP 1200

Due: 11/25/03 (Part II)

The goal of this lab is to explore the basic functionalities of the IXP1200 software development kit and microengines. There are two parts to the project. Part I uses an existing sample application (IP forwarding) to collect a number of workload statistics from the IXP simulator. Part II requires the programming of a small application on the microengines to implement a simple packet classification mechanism.

Both parts require access to a machine that has the Intel SDK installed. You can remotely log on to the course machine (address and account information given in class) or you can use a machine in Marcus 15B. If you want, you can also request an installation CD for your own machine.

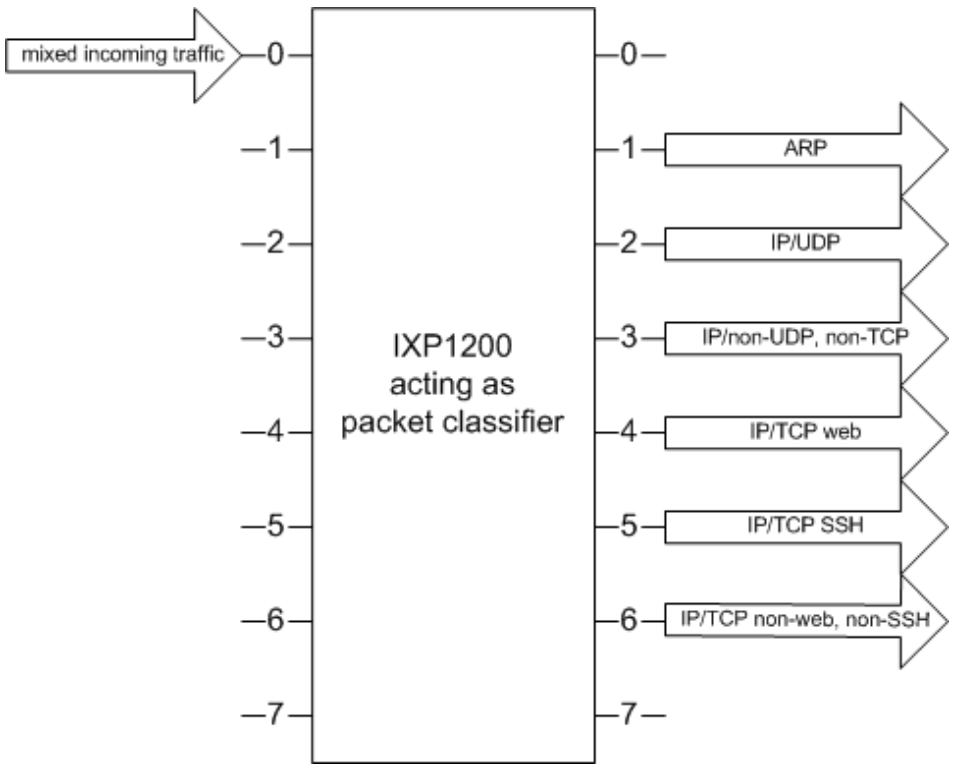
Part II: Packet Classification

For this part, you will implement a simple packet classification mechanism on the IXP1200. This requires that you add a small part of microengine assembly to a given IP forwarding implementation.

The function of the classifier is to separate different types of packets to different outputs on the IXP. There are six types of traffic that are considered in this lab:

- ARP traffic
- UDP over IP traffic
- Web traffic over TCP over IP
- SSH traffic over TCP over IP
- Non-web and non-SSH traffic over TCP over IP
- Non-TCP and non-UDP IP traffic (e.g., IP-over-IP tunnel)

The IXP1200 should classify each packet according to these rules and send it to an output port as shown in the figure below. Note that the classification step replaces the IP destination address lookup.



For this part, you should do the following:

- Extend the given forwarding code to implement the classification as described above.
- Determine the traffic mix. What fraction of the traffic belongs to each of the six classes?
- Use the execution coverage window in the simulator to verify that the instruction coverage of your classifier matches the traffic mix results.
- Assume that the classification step was really critical for performance. In your implementation, you have a choice of making classification decisions in different orders (e.g., check for UDP packets before checking the type of TCP packet etc.). In what order should packets be classified given the traffic mix in this example? In general, if the traffic mix is known, in what order should classification be done?

It is recommended that you go through the following steps:

- Download the classifier skeleton code from the course web page: http://www.ecs.umass.edu/ece/wolf/courses/ECE697J/labs/ece697j-lab2.zip and unzip it such that the L3fwd16.classifier directory is in the same directory as the L3fwd16 directory that you used for Part I.
- Open the project L3fwd16.dwp in the L3fwd16.classifier directory. Rebuild the code, start debugging, and run the simulator. Look at the IX Bus window and you will see that all packets arrive on port 0 and are sent back out on ports 0 and 1. When you have implemented the classifier there will be packets on outputs 1 through 6.
- Open the file rx_ether100m.uc in your project. Look for the places where comments were added that include the word "697J". That's where you want to pay close attention. You need to add code in the section that does the classification. A basic structure that identifies TCP/IP traffic and SSH/TCP/IP traffic is already given. Extend this to consider all the necessary cases. Also, ensure that all the traffic goes to the right ports (this is done with the instruction `move(output_intf, 0x0000000)`, where the hex value is the port number).
- Determining the port number: This is a bit unusual. The hex value that you put into the instruction is the port number (0-7) left-shifted by three bits. Thus, port 0x1 becomes 0x8 (as in the example for TCP traffic), and port 0x3 becomes 0x18.
- When you have implemented the classifier, you should see traffic on ports 1-6. Note that traffic is randomized, so you need to let the simulation run for a while (at least 40 or so packets) to make sure all packet types have appeared. To determine the traffic mix, set the simulator to stop after 1000 packets have been transmitted. Then check the number of packets on each port and determine the fraction.
- Look at the execution coverage window. Note that only microengine 0 is used. Your classification code is data-dependent and thus not all instructions are executed for all packets (e.g., the distinction between SSH and web is only done for TCP packets, not for UDP or others). So, you should easily find the classification code by looking at the instruction frequency. Look at which instructions of your classification code are executed how many times. Does this match the traffic mix?

Your report should include:

- The code that you wrote in the classification part of the rx_ether100m.uc file PLUS a brief description of how it works. Please don't submit the entire rx_ether100m.uc file.
- The traffic mix percentages for 3 different runs of the simulator.
- A discussion of the execution coverage that you observed and how it matches the traffic mix.
- An answer to the question on how to order the classification steps such that performance is optimized (see above).

If you have any questions or problems, contact me or ask during the help session at the end of the lecture on Thursday, 11/20/03.