CS 4390.001

Team Project Report

1. a description of the program

For this project, we have to design a simple IP router. First we separate the ARP and IP packets, sending the ARP to a different handler already mostly provided for us. The router must then verify the IP packet, i.e. check minimum length and checksum. Afterwards the router decrements the Time To Live (TTL) and computes the new checksum. The router has a static forwarding table, whose entries are used to forward the Ethernet frames it receives to the appropriate outgoing interface. The router must implement Address Resolution Protocol (ARP) for the next-hop’s MAC address of the Ethernet frames. ARP replies are cached and timeout after 15 seconds. If there is no matching address in cache, ARP requests are sent to the IP address, one per second, for five seconds. If no reply, host is determined to be unreachable. If invalid length, checksum, or ether type we drop the packet.

The router should respond to its interface being pinged with an ICMP Echo Reply (type 0). Additionally, it must generate ICMP packets to notify a sending host of errors, including Destination Net Unreachable (type 3, code 0) when there is no entry in table that matches destination, Destination Host Unreachable (type 3, code 1) when five ARP requests go unanswered, Port Unreachable (type 3, code 3) when the packet contains a TCP or UDP payload and destination matches router address, and Time Exceeded (type 11, code 0) when packet times out because the TTL reached 0.

1. the challenges that you and your team had and how you overcame them

One of the challenges of this project was getting everyone actually working together. It was hard to find times for everyone to meet up in person. To communicate with each other, we started with texting each other’s phone numbers, but eventually switched to using GroupMe. At one point Discord was brought up as a means of sharing files, so I downloaded and set up an account. However, in the end, it was decided for everyone to share our source code files with GitHub, since it performs automatic merging and allows for easier version control, and I set up a repository. The biggest challenge was figuring out how the provided code files related to each other because there was no quick reference document, we had to scroll through every file to figure things out. There was no explicit description of the hierarchy of the files, and finding struct definitions proved extremely tedious. The comments were occasionally ambiguous and not detailed enough. Another challenge was getting mininet to work, to update, to not crash.

c) what you have learned by doing the project

d) a discussion about algorithms and techniques used in the program

Some of the techniques that were used in the provided code were ones none of us had ever attempted to use before. Overlaying struct pointers to a data block created by malloc in order to apply order to it was especially useful for packet processing. Being able to parse the packets with a series of structs, each having its pointer shifted down in order, made finding specific cases for branching statements fast and easy. The passing of pointers along the methods meant that the packet stayed as one continuous block through each layer, only being slightly modified as it went. Each layer only looked at the point in the block that directly concerned itself.

The checksum algorithm was provided in the given code, but we had to create our own methods of processing queued packets waiting on ARP replies. They were queued by the ARP request process as a linked list, but we had to create the loop to properly encapsulate them and forward them to the proper gateway. In the event of reply timeouts we had a similar loop that had to construct new ICMP packets, using a malloc call and assigning a new ICMP header, and return them to the senders. The most frustrating algorithm was the longest prefix match, partially because it needs to be handled as a bitwise approach, and partially because of the many, many difficulties we had finding the relevant definitions of various structures. We compared table entries and the destination IP address using bitwise XOR, treating them as integers, and then left shifting until the first differing character was found, saving the longest match (ties favored prior choice).

e) contributions of each team member