Assignment 4: Internetworking

Due October 8/9, 2015

Our topic this week will be the notion of internetworking in general and IP, the Internet Protocol, in particular. IP is the foundation of the Internet Protocol Suite, the collection of protocol standards that make the Internet possible. These include protocols you have probably heard of including HTTP, FTP, TELNET, and TCP, all of which depend on IP to provide the basic service of data delivery. In addition, there are a number of protocols in the Internet Protocol Suite that are critical to the functioning of IP itself. These include ARP, RARP, ICMP, OSPF and BGP. While our focus this week is on IP, to fully understand IP, we will also need to consider the role of these support protocols and understand how the Internet transport protocols, UDP and TCP, depend on IP.

The main reading for IP will be the appropriate sections of the text. Please read §3.2 and 4.1.3.

To give you a bit more background on the design of IP, please read:

- Clark, David, The Design Philosophy of the DARPA Internet Protocols, SIGCOMM '88, pp. 106 - 114.

and

- Balakrishnan, Hari, A Graduate Course in Computer Networks, Chapter 2 (you don't need to read the Appendix).

The paper by Clark, who played a key role in the transition of the Internet from a research experiment to its current role, provides interesting insights on the factors that influenced the design of the Internet protocols. The sections from Balakrishnan's textbook/course notes provides similar insights to some more modern features of the network.

In addition, to introduce you to how TCP and UDP depend on IP, please read the beginning of Chapter 5 of Peterson and Davie up to but not including the section on *The State-Transition Diagram* on page 404.

Exercises

1. When an IP packet is sent across an Ethernet, the entire IP packet is placed in the data field of an Ethernet packet. The Ethernet packet has 6 byte destination and source address fields. In addition, the IP packet stored within the data field has two 4 byte fields for the source and destination IP addresses.

   Suppose, you spent some time eavesdropping on the packets flowing across some Ethernet attached to the Internet. After a while, you notice that in most packets the IP destination address and the Ethernet destination address refer to the same host. For one host, however, you see a significant number of packets containing the host's Ethernet address as their Ethernet destinations but containing IP destination addresses that don't match the host's IP address. Is something wrong? Explain. (If you think the answer to this question is obvious, then you understand the "trick" that makes IP work. If not,...)

2. Suppose some neophyte network programmer constructs a network application using IP to transmit data. Knowing that the application will primarily be used to communicate between hosts connected by an Ethernet, and that the maximum packet size on an Ethernet is 1500 bytes, the

programmer decides to send 1500 bytes of data in each IP packet created. If the program is used to transfer 1,500,000 bytes of data between two computers on such an Ethernet, how many Ethernet packets will actually be sent? What number of data bytes/packet would minimize the number of Ethernet packets sent? How many packets would be sent in this case? (Assume no retransmissions, collisions, etc. Just count the number of packets IP will try to send. Also, assume the program directly uses IP rather than UDP, TCP, or any other transport protocol even though this would be rather unusual.)

3. Complete question 3.72.

4. Wednesday afternoon I went into TCL 312 and I logged into the iMac, made a few changes to the machine's standard configuration and then typed a command. The command and its output are shown below:

```
-> netstat -nr -f inet
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            137.165.8.1        UGSc            9        0     en0
default            137.165.120.1      UGScI           3        0     en1
127                127.0.0.1          UCS             0        0     lo0
127.0.0.1          127.0.0.1          UH              1    51773     lo0
137.165.8/22       link#4             UCS             4        0     en0
137.165.8.1/32     link#4             UCS             3        0     en0
137.165.8.1        0:0:c:7:ac:8       UHLWIir         7        0     en0   1198
137.165.8.3        0:25:90:9b:14:dc   UHLWI           0       12     en0   1157
137.165.8.7        0:3e:e1:c6:7:d3    UHLWIi        457    10390     en0   1192
137.165.8.50/32    link#4             UCS             0        0     en0
137.165.8.227      a8:20:66:43:16:3b  UHLWIi          1        3     en0   1155
137.165.11.2       0:26:98:2f:ce:c1   UHLWI           0        0     en0   1200
137.165.120/21     link#5             UCS             0        0     en1
137.165.120.1/32   link#5             UCS             1        0     en1
137.165.120.1      0:0:c:7:ac:78      UHLWIir         3        0     en1   1154
```

Explain as much of the output that results as you can. This may require some extra reading. At the least, you will probably want to type

```
man netstat
```

on a Mac or FreeBSD system and read the manual page for the netstat command. In addition, there are many other sources of Unix documentation online that you may want to consult.

It may help you to try the same command in a terminal window on one of our Macs.

5. Every protocol's packet format represents a compromise between flexibility and overhead. If two many bits are allocated for fields such as addresses and error checks in a protocol's packet header, then the efficiency with which data can be delivered will be reduced. If too few bits are made available, the flexibility of the protocol will be limited in some way. For example, as the discussion of IPv6 in chapter 4 suggests, the decision to allocate only 32 bits for addresses in IP headers is now limiting the ability of the Internet to grow.

The complete layout of the standard (i.e. non-optional) components of an IP packet header is shown in Figure 4.3 of Peterson and Davie. Obviously, there are many fields in addition to the address fields and each of these fields has been allocated a fixed number of bits. Each of these allocations may limit the flexibility of the protocol in some way.

I would like you to consider whether the sizes of each the following fields limit the flexibility of IP:

- HLen
- Length
- Ident
- Offset
- TTL
- Checksum

Be precise. Specify which features of the current protocol would be affected. Discuss whether the limitation would become a factor as a result of growth in the size of the network, increases in supported transmission rates, introduction of new underlying network hardware (such as wireless), or other factors. If possible, be quantitative.

6. Many networking problems would be trivial if you could assume you already had a way for independent computers to communicate, but become difficult when you realize you need to solve them before you can communicate effectively. For example, writing an algorithm to find shortest paths is not difficult. To run such an algorithm, however, you must collect information about the state of distant parts of the network. This requires communication, which is difficult to perform until you are able to run your routing algorithm.

   Internet routing protocols are based on routing algorithms we have already discussed. In particular, RIP is based on the Bellman-Ford distance vector algorithm and OSPF is based on Dijkstra's shortest path first algorithm.

   To exchange routing updates, the routers running these Internet protocols depend on TCP and UDP, which depends on IP to deliver packets, which depends on these algorithms to build its routing tables! I'd like you to explain how Internet routing protocols can get away with this apparently circular dependence of IP routing on IP routing.

   To keep things concise, limit your attention to RIP (which uses Bellman-Ford) and explain precisely which features of the IP routing mechanism and the Bellman-Ford Algorithm make it possible for RIP to depend on UDP to deliver routing update messages (i.e. distance vectors).

7. Before Purple Air, the wireless network at Williams had a particularly annoying security system. In order to use the wireless network, you had to enter an ID and password through a web browser every time your laptop restarted or was woken from sleep. If you tried to use any network application other than a web browser before you did this, the network would appear to be dead. If you opened a web browser and tried to visit any website, what would actually show up in your browser window was the campus network login page. Once you entered your login information correctly on this page, everything would start to work correctly.[1]

[1] You have probably encountered other wireless networks that work in a similar manner. One key feature is that you do not have to enter a wireless password to authenticate to the wireless network infrastructure itself. That is, you get some wireless service without entering any password. The other feature is that no matter what you try, the only way to actually get useful network access is to attempt to access any web page and respond by providing the data requested in some form of login web page that appears on your machine instead of the page you actually requested.

I would like you to use what you now know about IP, DHCP, ARP, OSPF, BGP, etc. to explain how such a security system could be implemented.

Although this question was inspired by the Williams network, you should base your answers strictly on the characteristics of the network described within the problem statement rather than on any knowledge of the techniques actually used on the Williams network now or in the past. In particular:

- Assume that the network in question is a single hardware network based on one of the common IEEE 802 standards. That is, it might be an Ethernet or an 802.11 wireless network. You should not assume anything peculiar to just one of these standards, but you can rely on their common features (i.e. you can assume that they all use similar hardware addresses, they all support broadcast messages, none of them guarantee reliable delivery, they all impose a maximum packet size, etc.).
- Assume that no components of the hardware network will be modified to provide security.
- Assume that no component of or software on the client machines using the network will be modified. In particular, users will not be required to modify the software on their systems or install any specialized software on their computers. At the same time, assume that users will not modify any of the standard network device drivers, TCP/IP code, or network applications on their machines to circumvent the security mechanisms. That is, assume all of the networking hardware and software used by the client machines is "standard."
- Assume the client machines are configured for automatic network configuration using DCHP.
- Assume that the goal of the security mechanism is "total isolation until authentication." That is, a client machine should not be able to exchange IP packets with any machines other than the machine installed to implement the security scheme until its user has sent a user id and password to that machine.
- Note that the word "machine" in the preceding sentence is singular. That is, you should also assume that only one of the servers on the network will run modified code to implement this scheme. We will refer to this machine as the "security appliance." The security appliance may run several services (Hint: one of them will be DHCP), but other servers (e.g., the campus mail server) should be able to run without any modification.
- A complete answer to this question might seem to involve more information about TCP or the domain name lookup service than we have covered at this point. In fact, as far as TCP is concerned all you need to know about TCP is that web browsers, mail programs, chat clients and just about all other network applications send packets through TCP but that they eventually end up in IP packets. The domain name service provides a way for a machine to translate a symbolic name like cs.williams.edu into an IP address like 137.165.8.2. You should just assume that this process happens by magic for this question.
- Finally, you may assume any modifications you like to the implementations of the members of the TCP/IP protocol suite on the security appliance.