15-440 Distributed Systems
Homework 1

Due: September 15, 11:59 PM via electronic handin

Hand in to: `/afs/andrew.cmu.edu/course/15/440-f10/handin`

Name your file: `YOUR ANDREW USERID.pdf`

September 8, 2010

1. (a) Consider the password cracker and what you're learned about RPC. Is using RPCs for the worker clients a viable way to write the password cracker? If yes, please write the signatures for the stub functions on both the server and client side, along with a short explanation of what each function does. If not, explain in detail why this is not a good idea.

   (b) If you were to write a series of RPCs for a heterogenous system (ie, not the same exact system specs), what variables might you consider and how would you solve the problems that arise? (Hint: an example might be the size of an int, int *, etc). Discuss 3 variables of interest, and describe either a workaround for the problem, or explain why it's okay for this problem to remain unsolved.

2. (a) A client and a server are communicating using UDP. The server sends to the client 3 packets that each contain 2 characters. The contents of the packets are "AB" "CD" and "EF", which are sent by the server in that order. The client tries to receive messages from the server, and when a message is received, it prints the contents to STDOUT without any additional formatting, and blocks while waiting for the next message. If the server only sends these 3 packets, list all possible outputs by the client to STDOUT, assuming the contents of the packets are never corrupted and no packets are duplicated.

   (c) For each of the listed properties, which transport layer protocol (TCP or UDP) would you use if your application needed that property? Give one *succinct* reason why. Be specific.

   - High throughput
   - Sending very large messages
   - Short-duration interactions with little data
   - Sharing bandwidth fairly between competing flows
   - One-to-many communication (called "multicast")

   Note: We did not cover all of these properties in detail in class. But you should be able to reason about the answers from what you know about TCP and UDP at a fairly high level.

3. When sending a packet across a network, it may encounter delay due to one of several reasons. If the network looks like this:

```
Sender ---------------- Router --------------- Receiver
          Link A                   Link B
```

We need to know a few attributes of the network to understand the delay. First comes the length and the width of the links: Their *propagation delay* (the amount of time it takes the first bit of your transmission to get from one side to the other), and their *capacity* (the number of bits per second we can send on the link).

The propagation delay is usually some fraction of the speed of light. For a copper wire, it's about 0.6c. Let's assume some reasonable values for a cross-country link with a router in the middle, then:

Link	One-way propagation delay	Capacity
Link A	10ms	1Gbit/second
Link B	20ms	1Gbit/second

(a) If we run the `ping` program on the sender, it will send by default a small, 64 byte packet to the receiver. The receiver will echo the packet back to the sender, and the sender will print out the time it took.

Keeping in mind *both* components of the delay we talked about above, what is the minimum time that it will take the ping packet to return to the sender, ignoring processing time at the router and at the sender/receiver?

(b) From an Andrew machine, please ping the IP address 206.197.119.139. (This is dga's web server, so be nice to it.) Tell `ping` to send 10 packets:

```
ping -c 10 206.197.119.139
```

When it completes, ping will print out the min/avg/max and deviation of the round-trip times it observed. What values did you observe?

(c) Your friend looks at those values and tells you that it's impossible for Dave's web server to be located in China. You think about it for a while and agree. Tell us why by calculating the minimum possible round-trip time from Pittsburgh to a server in China. State your assumptions.

(d) Assume that you wanted to send a command to dga's web server. To do so, you establish a TCP connection to port 80 and send an HTTP request with a particular format. Assuming you don't have any pre-cached connections, what is the minimum latency between when you run your program locally to when the command is actually executed on dga's web server? Explain what needed to happen first.

4. In networks, we often characterize devices by what "layer" they operate at. You'll often hear people refer to, e.g., a "layer 2 switch" when talking about something like an Ethernet switch. These layers usually refer to a model called the OSI model, which looks like this:

Application	The data the application is sending
Presentation	Architecture-independent data representation (think what happens in RPC marshalling), encryption, etc.
Session	Not often used
Transport	End-to-end connections and reliability, flow control, etc. (TCP goes here)
Network	End-to-end addressing, forwarding, etc. (IP goes here)
Data Link	Physical addressing, packetization, etc. (Ethernet goes here)
Physical	The way the data is sent as an electrical, optical, wireless, etc. signal on a physical link/wire/etc.

Thus, an Ethernet switch operates at the "data link" layer, or layer 2. Ethernet switches examine Ethernet addresses to figure out the destination. An IP router operates at layer 3, the network layer, and examines the IP address (which is kept the same end-to-end from source to destination) to figure out where to send the packet next.

Recall from class that we usually implement layering by *packet encapsulation*: We put the UDP header and data "inside" an IP packet. We put the IP packet "inside" an ethernet packet to transmit it on the local Ethernet.

You send a packet from one of the Andrew Linux machines to Dave's web server (IP address 206.197.119.139). Let's figure out what this packet looks like.

(a) Use the `ifconfig` command (`/sbin/ifconfig`) to find out the IP address and Ethernet MAC address of the Andrew Linux machine you're using. The Linux ifconfig output calls the IP address the "inet" address and the MAC address the "HWaddr". (The "-a" flag to ifconfig will show all interfaces, but you'll have to figure out which one is the main interface. Which one has the most received - RX - and sent - TX - packets? Ignore `lo`, that's local.). What are the addresses?

(b) Remember that your computer isn't directly connected to Dave's Ethernet—the packet has to go from CMU to Dave's ISP. Therefore, a packet from you to Dave will have to go through multiple layer 3 "hops", each of which forwards the packet to the next hop. Use the `traceroute` utility (`/bin/traceroute`) to discover the IP-layer hops that your packet will take on the way to Dave's web server. Reproduce the output below.

(c) From the above discussion, you know that the destination IP address in the packet has to be that of the web server. So how does the packet get from your andrew machine to the next "hop"??

(d) Use the `route` command (`/sbin/route`) to show the contents of the local routing table on the andrew machine. This table tells the machine where to send packets in order to reach a particular destination. The "default" route is the one used, on these machines, to reach the rest of the Internet. HINT: Use `route -n` to show the IP address of the default gateway instead of a chopped-up hostname, but note that with -n, the default route is called "0.0.0.0". Include below the line of output matching the default route. Does the gateway address match what you saw from traceroute?

(e) Use the `arp` command (`/sbin/arp -na`) to show the contents of the local table that maps IP addresses to the Ethernet address that owns that address. What is the Ethernet address of the next hop for your packet?

(f) You send a single UDP packet from port 8888 on that host to port 9999 on Dave's web server (IP 206.197.119.139). Draw the packet headers for the Ethernet, IP, and UDP layers in the correct order, showing where the application data goes. You don't have to show all of the fields for the headers, but do show the source and destination addresses (for Ethernet and IP) and the ports (for UDP). You'll probably want to google to see the packet formats, and note that Ethernet is tricky.

5. Consider the following code snippet.

```
int check_next_ten_passwords(char *str) {
  char *tmp = malloc(strlen(str) + 1);
  strcpy(tmp, str);

  for (int i = 0; i < 10; i++) {
    if (!strcmp(crypt(tmp, ‘‘aa’’), PASSWORD_STRING)) {
       printf("I FOUND IT!  %s\n", tmp);
       return 1;
    }
    tmp[0]++;
  }
  return 0;
}
```

(a) This code has a bug. What is it?

(b) If you used this code by creating a thread that called the function to test a password and then destroyed the thread, the bug would affect you. What would happen if you instead `fork()`'d a new process to test the password, and then had the process exit? Explain briefly (1-2 sentences) the difference.