# CS 4596/CS 6596 Fall 2016

### Program #3

**Due Wednesday November 30.**

Mobile IP is typically implemented within the kernel of the average operating system (to make routing very fast.) While extremely inefficient, similar functionality could be provided at the application layer. Since it's much easier to code at the application layer, we will develop an application layer solution.

We have five entities to create, the Mobile Node, Home Agent, 2 Foreign Agents, and a Data Source. We will assume that the Foreign Agents and Mobile Node processes run on the same physical machine.

Since all the nodes are likely on the same network, we need to be a little creative about how we set this up. In Mobile IP, the Mobile Node's address must remain the same regardless of which link it is attached to. We will assume that the fixed home address consists of the port (and not the IP address.) So let's say that the Mobile Node's home address is 5900.

The **Home Agent** should be advertising reachability for the home link. Let's assume that everyone then knows the Home Agent's IP address is the address to send the packets to (say 134.154.11.10.) They will use the port number associated with the Mobile Node though (5900.) So the Home Agent can intercept packets destined for the Mobile Node by opening a UDP socket at (134.154.11.10/5900.) The Home Agent should then send these packets to the care-of address currently registered for the Mobile Node.

The **Data Source** should open a UDP socket and send packets **once per second** addressed to the Mobile Node's port and the Home Agent's IP address. Each packet should contain a sequence number so that the Mobile Node can keep track of which packets it has received.

The **Mobile Node** and **Foreign Agents** will run on the same machine but you must implement them as three processes. The Mobile Node always has a UDP socket open which listens for incoming packets. Using the port we suggested above, and assuming that the Mobile Node and Foreign Agents are on machine 134.154.15.75, the Mobile Node would open a socket bound to 134.154.15.75/5900. When the Mobile Node receives a packet, it looks to see if it received the packet from its current Foreign Agent. It can tell this by inspecting the source port. **Every 5 seconds**, the Mobile Node moves to a new link. Behind the scenes, we assume that the Mobile Node heard an Agent Advertisement and so it knows the **Care-of address** for the new Foreign Agent. The Mobile sends a registration request including the care-of address to the Home Agent on the Home Agent's well known address and port. No reply is necessary.

When the **Home Agent** receives a registration request from the Mobile Node, it uses the included care-of address to change the destination address of its UDP socket to the care-of address. Any new packets intercepted for the Mobile Node will be sent to this new care-of address.

We actually need 2 or more **Foreign Agents**. The Foreign Agent is a process which listens on a UDP socket bound to the care-of address. Let's assume both Foreign Agents execute on the same machine and they bind to 134.154.15.75/6001 and 134.154.15.75/6002. We assume that the Foreign Agents know the Mobile Node's IP address and port (we're skipping this part of the registration process.) When a Foreign Agent receives a packet (from the Home Agent), it forwards it on to the Mobile Node. Each Foreign Agent then needs one UDP socket with source address of 134.154.15.75/6001 or 6002 and destination address of 134.154.15.75/5900 (the Mobile Node.)

**Note:** Don't hard-code port numbers or IP addresses into your programs. For one, it's very bad style, and for two, if you try to use the same IP address/port pair over again too quickly, the OS will not let you. There is a timeout mechanism which is designed to ensure that all packets destined for the old socket are dead before allowing another socket with a similar address to be created.

You can use sleep(), gettimeofday() or alarm() to tell when 1 second has gone by on the data source. I would suggest using gettimeofday() to implement the 5 second timers on the mobile.

---

You will need to write 4 programs (all very small).

The **Data Source** opens a UDP port, sets the destination to the Home Agent IP/Mobile Node port, and sends a packet once per second, with increasing sequence numbers. The source IP/port do not matter.

The **Home Agent** opens a UDP port with source bound to Home Agent IP/Mobile Node port and sets destination to Foreign Agent IP/Foreign Agent port. If it receives a packet on this port, it is either a registration request from the Mobile Node or a data packet from the Data Source. If it is a registration packet, the Home Agent changes the destination to Foreign Agent IP/ **new** Foreign Agent port. If it is a data packet, it sends the packet to the current care-of address. (Note, that in real life, the HA would listen for registration packets on its own port, not the mobile node's. It's easier to listen on one port than two, but you may do it either way.)

The 2 **Foreign Agents** open a UDP port, bind the source to Foreign Agent IP/Foreign Agent port, and set the destination to Foreign Agent IP/Mobile Node port. (Remember that Foreign Agent IP = Mobile Node IP.) Each time they rceive a packet, they forward it on to the mobile.

The **Mobile Node** opens a UDP port with source bound to Foreign Agent IP/Mobile Node port and destination set to Home Agent IP/Mobile Node port. Every 5 seconds, it sends a registration request to the Home Agent with the alternate Foreign Agent's care-of address in the packet. (We assume that the Mobile Node knows the care-of addresses for the 2 Foreign Agents.) When it receives a packet, it must make sure that it has come from the right Foreign Agent (by comparing the source port to the currently registered Foreign Agent). When the Mobile Node has run for 100 seconds it should exit.

---

**Output**

Each program must print out an indication of events which occur.

The **Data Source** must print a line showing the sequence number of the packet sent, the time sent, and the destination IP/port.
Example:
Sequence Number = 1 Time = 15:22:34 Dest = 134.154.11.15/45000
Sequence Number = 2 Time = 15:22:35 Dest = 134.154.11.15/45000

The **Home Agent** receives two types of messages, data packets and registration packets. For data packets, it should print the sequence number, the time, and which Foreign Agent it is forwarding the packet to. For registration packets, it should print the time, and a message indicating what the new care-of address is.
Example:
Sequence Number = 2 Time = 15:22:34 Forwarded to 134.154.11.10/6000
Sequence Number = 3 Time = 15:22:35 Forwarded to 134.154.11.10/6000

Registration packet received. Time = 15:22:36 Changing care-of address to 134.154.11.10/6001
Sequence Number = 4 Time = 15:22:37 Forwarded to 134.154.11.10/6001

The **Foreign Agents** should print an indication of the time, the sequence number, and the destination each time they receive a packet and forward it.
Example:
Sequence Number = 2 Time = 15:22:35 Forwarded to 134.154.11.10/5900
Sequence Number = 3 Time = 15:22.35 Forwarded to 134.154.11.10/5900

The **Mobile Node** will receive data packets and send registrations. For data packets, it must print out the sequence number, the time of arrival, the Foreign Agent address, and an indication of whether the packet came from the right Foreign Agent or not (based on the source port). If not, it must be rejected (because in real life, the Mobile could no longer communicate with the old Foreign Agent). Registration packets should be indicated with the time and the new care-of address.
Example:
Sequence Number = 3 Time = 15:22:35 FA = 134.154.11.10/6000 Accepted
Registration sent. Time = 15:22:36 New care-of address = 134.154.11.10/6001
Sequence Number = 4 Time = 15:22:36 FA = 134.154.11.10/6000 Rejected
Sequence Number = 5 Time = 15:22:37 FA = 134.154.11.10/6001 Accepted

---

I have posted (separately) two example programs which send and receive UDP packets respectively.
usend4.c - C UDP sender
urecv4.c - C UDP receiver
See the man pages for socket, bind, connect, sendto, recvfrom, close for more details. You need to specify the section "man -s 3N" for some of the commands.
I also posted a PowerPoint lecture which describe how to use the socket functions.

**Notes:**

**You must produce the output I've asked for.** It shows me what I need to know as well as showing you whether your programs work or not.

**The data source and home agent must run on different machines than the foreign agents and mobile node. The 2 foreign agents and mobile can all run on the sam**

Any information that we are assuming the entities know automatically (e.g., the data source must know the Home Agent's address and port) should be passed in on the command line. You will need to think about the order you will invoke the programs to be sure that you can provide the information they need (in case the port numbers you want to use are not available.)

Do your development **incrementally**. So:
1. Get usend and urecv compiled and working. Look at their headers for the format of the command line.
2. Modify the programs so that the sender sends once per second and produces the Data Source output. Modify the receiver so that it produces the Mobile Node output.
3. Modify another program so that it forwards packets (alternate receiving and sending on the same socket). This will be the basis of both your Home Agent and your Foreign Agents.
4. Get the chain Data Source -> Home Agent -> Mobile Node working.
5. Get the chain Data Source -> Home Agent -> Foreign Agent -> Mobile Node working.
6. Add in the registration functionality so that the Mobile Node sends registration packets and the Home Agent accepts them.
7. Modify the Home Agent so that it actually modifies its destination address based on the registration packet contents.
8. You're basically done! Collect your output.
Make sure you have some working version so that you can hand in something on the due date.

If you are unfamiliar with C function calls, here is an excellent link to on-line **man pages**, specifically for Linux. http://www.die.net/doc/linux/man/

See the man pages for ctime(3) to print out timestamps, and inetntoa(3) for printing IP addresses.

The various C and C++ compilers on the Linux machines are called gcc, cc, g++ (and possibly others.) I use gcc myself. See the man pages for details.

A **makefile** is always a good idea. Here is an example. Also see the man page on makefiles.

```
# CS 4596 Project 3
all: usend urecv
usend: usend.c
	gcc -Wall usend.c -o usend
urecv: urecv.c
	gcc -Wall urecv.c -o urecv
```

**Your programs need to run on the College of Science Linux machines.** The machines you have available are:
trig.sci.csueastbay.edu
geometry.sci.csueastbay.edu
calculus.sci.csueastbay.edu
arithmetic.sci.csueastbay.edu
algebra.sci.csueastbay.edu

You can work in any language, Java, C++, C, Pascal, Prolog, LISP..., however I only have example programs for C.

**To hand in:**

1. A printout of your commented code for the data source, home agent, foreign agent, and mobile node.
2. Scripts or screenshots of the output of the data source, the home agent, the two copies of the foreign agent, and the mobile node (5 scripts total). **Be sure to include the program invocation line in the script**.
You can generate a script by executing the "script" command before invoking your program. When your program is done, type exit to end the script. The file will be called "typescript". You can also name the scriptfile by using one of the options to the command.
So you might sit down at algebra (a linux machine) and start up the GUI. You could then open a terminal and run the data source on this machine. You could open another terminal and telnet or ssh to geometry and run the home agent. Open three more terminals and ssh to the same machine and run the two foreign agents and the mobile node.
Run the programs for around 60 seconds so that we see 10-12 changes of control and then kill them.
Exit from the script and you're done!
3. Specify which machines and ports each of your processes ran on.

---