USC EECS450/30782 PROGRAMMING ASSIGNMENT 2 Fall 2015/Touch

<u>Due Nov. 5, 2015 by 2pm – submit source code as a ZIP file via Blackboard</u>

You are to write your own connectionless (UDP) client. Print out everything.

You can start with any open-source UDP client as a template, and may want to include some parts of the template from Program 1. UDP packets will have the following format:

FM:##_TO:##_ NP:#_HC:#_NO:###_VL:xxx_DATA:xxxxx…

FM – the source address (a number from 1 to 40, 00 as "no name yet" and 99 as "all")

TO – the destination address (same range)

NP – number of the protocol

1 = registration request	2= registration response
3= data	4= data confirmation
5= registry dump request	6= registry dump response
8= serial broadcast	9= serial broadcast confirmation

HC – a count of possible hops remaining (0-9), used only for relaying in Step 5.

NO – a message ID number; you will generate these in order (start with whatever number you want), and match the number in the request to its corresponding response

VL – visited list, a sequence of 2-digit numbers separated by commas that indicates which nodes have already seen this message, e.g., VL:03,04,08,05 – note that VL is empty (VL:_) unless the message has been relayed in Step 5.

DATA – the contents of the message

The entire UDP payload can be up to 500 bytes.

Underscore (_) indicates a single space.

The # fields indicate decimal numbers; they are exactly as many digits as there are #s above.

The xxxx… DATA field indicates text data; it can contain any printable ASCII character and \n.

The program should take the following arguments (i.e., see similar values from Program 1):

-s	server DNS name or IPv4 address
-p	server port number

1. Send a UDP packet to "register" your program (1 point)

At the start of your program, you need to register to get a name. Send a UDP packet to DNS.POSTEL.ORG on port 60450, from your server IP address and port number, and wait up to 2 seconds for a response.

The message should read:

FM:00_TO:01_ NP:1_HC:1_NO:nnn_VL:_DATA:register_me

The response will read:

FM:01_TO:00_ NP:2_HC:1_NO:nnn_VL:_DATA:you_have_been_registered_as_##

Repeat up to a total of 5 times until you succeed, or stop and report an error.

2. Pull down the registry (1 point)

When the user requests a registry (by typing "r"), send a message to DNS.POSTEL.ORG to get the registry map of currently active nodes. Do this up to 5 times until a response is received. Make sure the FM node number is the one you received when you registered (step 1, above):

FM:##_TO:01_ NP:5_HC:1_NO:###_VL:_DATA:GET MAP

The response will be:

FM:01_TO:##_ NP:6_HC:1_NO:###_VL:_DATA:nn=ip@port,nn=ip@port,…

Where "ip" is an IPv4 address, in dotted decimal, and "port" is a port number, in decimal.

There will always be at least one node in the map, but there may be more. The entire list will fit in a single message (the list is YOUR entire list; it may not reflect ALL available nodes).

Print out the received info as a table.

3. Implement a server program to send/receive short messages (1 point).

Loop waiting for either the user to ask to send a message (or download the registry again) or a packet to arrive.

If a packet arrives with NP:3 that matches your registered address, print it to the screen.

If the user indicates they want to send something (e.g., by typing "s"), ask them for the destination (a 2-digit integer from 01 to 40, presumably from the registry you showed them in step 2), then ask them to type a message (ending with ".").

Send the message to the appropriate IP address and port (from the table in Step 2) from your node identifier to theirs:

FM:##_TO:##_ NP:3_HC:1_NO:###_VL:_DATA:xxxx…

Send this message up to 5 times until you get a matching response with the FM and TO identifiers swapped with the same NO field and NP:4 and DATA:OK

If you receive such a message from anyone, send a matching response with the FM and TO identifiers swapped with the same NO field and NP:4 and DATA:OK

4. Serial broadcast (1 point)

When the user requests to send a broadcast (by typing "b"), let them type a message (ending in ".") and send one copy to every node in your registry EXCEPT yourself and the nameserver (01):

FM:##_TO:##_ NP:8_HC:1_NO:###_ VL:_DATA:xxxx…

The FM: address will be your node number, HC: (hopcount) should be set to 1. Send the message up to 5 times until you get a confirmation response with NP:9 from each party or give up (and report that as an error). If you receive such a message correctly, respond by swapping TO and FM identifiers, NP:9, HC:1, and DATA:OK

5. Relaying (1 point)

Recall that step 3 will handle messages with NP:3 that are TO: your node number. Now handle other NP:3 messages that are NOT TO: you as follows:

- Send a confirmation back to the source (swap TO:/FM: identifiers, NP:4, HC:1, DATA:OK), just as you would for messages that are addressed to you

If HC=0, then print out the message "dropped – hopcount exceeded" (and any message info. for the user).

If HC>0 and your node number IS in the VL list, then print out the message "dropped – node revisited" (and any message info. for the user).

If HC>0 and your node number IS NOT in the VL list, then send a relay copy to 5 nodes selected at random in the registry EXCEPT yourself and the nameserver (with HC decremented by 1, retaining the original TO:/FM: fields, and append ,## with your node number to VL list)