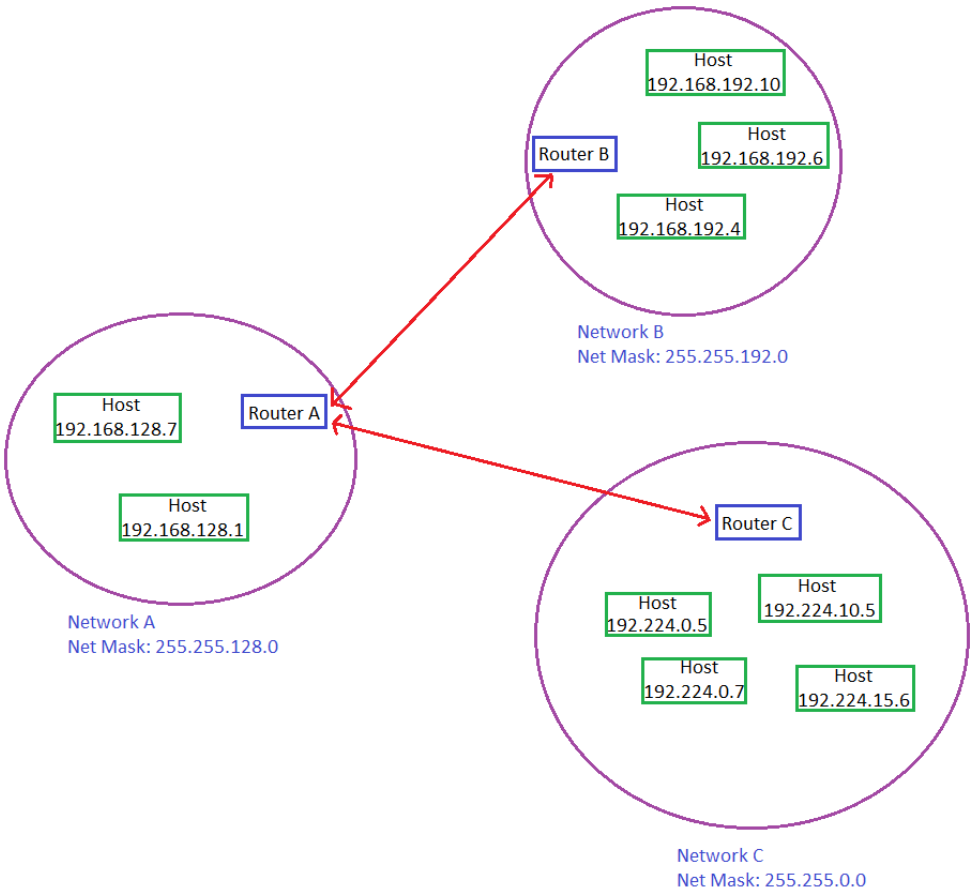
CMPUT 379 Assignment 2a, Winter 2015

Due Friday, Mar 21 (submit by 11:55 pm)

Worth 20 marks (however, 5% of total course weight)
(IPC, sockets)



In this assignment, you will simulate the task of a simple packet router. Three networks A, B and C are defined in the figure mentioned above. Any two hosts within the same network can directly communicate. However, a host needs to contact its default router when it wants to communicate to an external host. Routers A, B and C are the designated default routers for networks A, B and C respectively. There are direct links between Router A and Router B, and between Router A and Router C. But there is no direct link between Router B and Router C. In this assignment, you would simulate the routing task of Router A. You need to run a program that would receive incoming packets and make necessary routing decisions based on packet-destinations. You would run another program on the same machine that would generate and send packets to the router program. The source of each generated packet may be any host from any of the networks A, B and C, but packet destination would be a host from a network different from the source network (we assume that two hosts within a network can communicate directly). For some of the

generated packets, the destination would be host from a non-existing network (say, network D). This would verify whether Router A can handle incoming packets that have invalid destinations. When a new packet is generated, the source host would be randomly chosen from networks A, B or C (again chosen randomly). When you need an invalid destination for a generated packet, you can choose 168.130.192.01. Packet ID's of generated packets should be sequential numbers, and TTL values should be randomly chosen between 1 and 4 inclusive. TTL (Time-To-Live) is the maximum number of hops the packet may be allowed to move before being dropped by any router/host.

The **router program** will take three command-line arguments:

<port number to listen to> <routing table file path> <statistics file path>

The router program will:

- read its routing table from the specified file
- The file will consist of multiple lines with the format:
  <network-address> <net-prefix-length> <nexthop>
  The network-address would be in IPv4 dotted decimal. The nexthop value would 0, RouterB or RouterC. The three values will be separated by spaces. For example:

  192.224.0.0 16 RouterC

  A 0-value of nexthop means a direct delivery of a packet to a host within network A.
- listen to the specified **UDP port**
- accept simplified IP packets of the format:
  <packet ID>, <source IP>, <destination IP>, <TTL>, <payload>
  Example: 215, 192.168.192.4, 192.224.0.7, 64, "Hello"

For each received packet, your program will:

- decrement the TTL field of the packet
  - if it reaches zero, the packet will be dropped and the "expired packets" counter will be updated
- figure out which of the entries in the routing table (if any) match the packet
  - it will do this by checking whether the destination address in the packet matches the <network-address> entry for a particular route. To do son, you need to extract the first <net-prefix-length> number of bits from the destination address.
  - It will scan the routing table sequentially and will use the first route that matches (you can assume the table is sorted according to the longest-prefix first)
  - if there is no matching entry, the packet will be dropped and the "unroutable packets" counter will be updated
- if the value of the nexthop field for the entry is 0:
  - print the following:
    Delivering direct: packet ID=<packet ID>, dest=<dest. IP address>
  - update the "delivered direct" counter
- if the value of the nexthop field for the entry is RouterB or RouterC
  - update the counter corresponding to the packets forwarded to that router

Note: whenever a routing decision about a packet is done, the packet would not be processed any further.

Statistics file:

After every 20 packets are processed, you will update the statistics file (the path to which was given as a command-line argument). The format of the file will be:

expired packets: <number of packets expired>

unroutable packets: <number of packets containing invalid destination>

delivered direct: <number of packets delivered to network A>

router B: <number of packets forwarded to RouterB>

router C: <number of packets forwarded to RouterC>

The **packet generator program** will take two command-line arguments:

<port number to connect to router> <packets file path>

The program will:

- create packets and set <packet ID>, <source IP>, <destination IP>, <TTL>, <payload> as mentioned in the first paragraph
- sends packets to router program

Packets file:

After every 20 packets are generated, you will update the packets file (the path to which was given as a command-line argument). The format of the file will be:

NetA to NetB: <number of packets generated with source host in Network A and destination host in Network B>

NetA to NetC: <number of packets generated with source host in Network A and destination host in Network C>

NetB to NetA: <number of packets generated with source host in Network B and destination host in Network A>

NetB to NetC: <number of packets generated with source host in Network B and destination host in Network C>

NetC to NetA: <number of packets generated with source host in Network C and destination host in Network A>

NetC to NetB: <number of packets generated with source host in Network C and destination host in Network B>

Invalid Destination: <number of packets generated with invalid destination >

When the user interrupts with Ctrl+C, you will also update the statistics and packets files before the program exits.

Error handling:

You will need to handle any I/O-related exceptions. If the routing table file does not exist or cannot be read, your program should print an error message and exit. If the statistics or packets files does not exist and the program is unable to create it, it should print an error message and exit. If, later on, an error occurs while trying to update the file, the program should print an error message and it should continue running (i.e. it should not exit in this case).

If your program is unable to listen to or connect to the specified UDP port it should print an error message and exit.

If an error occurs while receiving an incoming packet, the program should print an error message and it should continue running (i.e. it should not exit in this case).

Marking

This assignment will be marked out of 20 marks.

- `router.c` would implement **router program** and will be worth 10 marks
- `pktgen.c` would implement **packet generator program** and will be worth 10 marks

Deliverables

You must deliver the code for `router.c` and `pktgen.c`, as well as a `Makefile` to build them, where the command "`make all`" will build both **router** and **packet generator**.

Mandatory Coding Style

Consistent and readable C coding style is *very* important to spotting bugs and having your code readable by others. You **MUST** comply with following coding style:

- Proper indentation should be included
- TAB should be used for indentation
- NO line may be longer than 80 chars
- All functions should be prototyped
- All variables are declared at the start of a block
- C style comments must be used (not C++ style)

Important: please make sure there are no stray processes left on your machine before leaving.

Note:

Please see http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing to know about IPv4 addresses with *slash notation*.

Please see http://en.wikipedia.org/wiki/Longest_prefix_match to know about the *longest prefix match* forwarding technique.