

cmsc 417 programming assignment #5

May 2, 2006

Goal: Routing

1 Deadline

DUE: Friday, May 12. The late deadline will be Monday, May 15; you have only one late deadline to play with.

2 Context

Build a new type of message that will contain a list of route advertisements of the form:

```
struct route_advertisement {
    uint32_t destination_address;
    uint8_t cost;
    unsigned char reserved[3];
}
```

The absence of an entry does not signify its unreachability. Infinity is 16. Let an advertisement with cost 16 signify a withdrawal. Advertisements with cost zero are invalid.

The reserved field ensures that the visible length of the route advertisement is 8 bytes, because an array of route advertisements would have each one starting on a new four-byte boundary.

You must send triggered updates. You must not send more than one triggered update per incoming message (if many routes are changed, these routes should be bundled into the same message). The triggered update should be delayed by a second.

You must send periodic updates at least once per minute to refresh the routes of your neighbors. Use a 50 second timer plus ten seconds uniformly at random. ($50 + \text{random}() \% 10$) You may replace the 25 second hello messages with periodic updates.

3 Protocol two

Send and receive route advertisements with ttl 1 using protocol 2. Protocol 1 will remain the string payload. Every 8 bytes of a protocol 2 packet is a route advertisement. Upon receipt, accept new routes using the distance vector route acceptance rules.

Use split horizon in assembling protocol two packets. Split horizon across the multiple-access network formed by the multicast involves not advertising routes learned from the multicast interface back to the multicast interface.

4 Your routing table

Your routing table should consist of elements of the form:

```
struct route_table_entry {
    uint32_t destination_address;
    uint8_t cost;
    uint32_t next_hop_neighbor;
};
```

with pointers if needed for a linked list or open hash table.

5 Printing the routing table

In response to “print routing table” (feel free to also do so on “prt”), print each routing table entry, comma-separated. Since a few of you had non-standard views of what comma separated values should look like, here’s a snippet.

```
printf("%u, %u, %u\n", r.destination_address, r.cost, r.next_hop_neighbor);
```

Do not print entries that have cost infinity, if, for some reason, you decided to continue storing them.

6 By popular demand

When you see a packet of protocol 1 and a destination of your address, print the contents. Hooray!

7 Forwarding

When you see a packet of any protocol destined for someone in your routing table, decrement the ttl and forward unless the ttl is zero or is more than infinity (16) (just in case someone else forgot to discard. Forwarding should be the same as sending; restated, sending should use forwarding.

That is, attempt to forward if the destination address of the message is different from your address and is not zero. Processing a message to be forwarded should not affect the neighbor table. That is, the source address should remain unchanged and might represent a distant node, so shouldn’t affect the neighbor table. Only update the neighbor table with an address when the destination address of a packet is zero.

Forwarding should use the sendmsg rate limiting.

8 Withdrawals

Generate a withdrawal when a TCP connection goes down or a neighbor is expired from the neighbor table. If you’re not connected by TCP to anyone, propagating a withdrawal is unnecessary (all others on the multicast will notice the neighbor’s absence, and split horizon means that no one is using you to get to that neighbor).

9 When there’s no route

Print ERR:NOROUTE as on sendmsg when the neighbor is absent. We’ll not build an error message.

10 Hints

- To construct a triggered update, build a routing update as you process one you've received. If anything in your table changes, add the new or changed route to the update. Note that this triggered update would have to go everywhere but the sender. (It could go to the sender, but split horizon would prevent all routes in this update from being sent there.)
- Be sure to set the source address correctly and the ttl of a routing message to 1. The destination may be the address of the neighbor or 0.
- You may send routing messages to the multicast address, or send individual updates to everyone in the neighbor table.
- Do not fear global variables.
- When select returns -1 with errno set to EBADF, it means you didn't remove a closed file descriptor from the set. You can figure it out by checking each socket using `fcntl(socket_number, F_GETFL)` for every socket... if it too returns -1, that's the socket that's been toasted.
- `signal(SIGPIPE, SIG_IGN)` to not crash when another endpoint goes down.

11 Objectives

At the end of this assignment, you should understand:

- routing.
- forwarding.
- why refactoring is good.

12 Questions

As you complete the assignment, think about the following questions.

- Why the randomness in periodic update scheduling?
- Why can hello messages be supplanted by periodic updates?
- Why is it okay to send a route update as a multicast message or as a unicast to each neighbor?