Routing Race

EE122 Fall 2012

Due date to be announced later

THIS IS OPTIONAL AND WORTH NO CREDIT

The Routing Race is an entirely optional portion of EE122 designed to allow you to test out ideas about routing. As an optional portion of the course, there are no points or credits associated with this. However the two best performers will receive prizes.

You will be using a slightly modified version of the project 2 simulator. The additions and modifications are noted in the Simulator section of this document. To participate in the routing race you will write new switch/routers which maximize the probability that packets are delivered to their intended destination in an extremely unstable network. To be precise, we will run your switches on multiple topologies, with links failing and being repaired at a rapid rate (several a second). At the same time we will attempt to send packets between end hosts connected to this network. Your aim is to maximize the number of packets delivered to their intended destination, while minimizing the amount of time taken on average. Precise rules are stated below.

Rules

1. The first rule of routing race is that you may not talk about the routing race with anyone outside of your own team.
2. You may work individually or in teams of up to three people.
3. Your aim, should you choose to participate, is to maximize the number of packets delivered to a destination. Your solution will be compared against other solutions and the highest ranked solution would have won the race (and therefore whatever prize we associate with this victory). Ranks will be determined by the following factors in order.
    i. Percentage of packets delivered.
    ii. Average time taken to deliver a packet packet.

iii. Route inflation, i.e average of hop count (i.e. number of hops for packet to be delivered)/hop count for shortest path.

iv. If the packet header is modified, the average size of the new header.

What this means is that if we have the five entries listed below (imagine for all paths tested the shortest hop count was 5 hops)

a. A delivers 50% of the packets, takes 1 second per packet on average, increases header length by 0 with a route inflation of 6/5

b. B delivers 70% of the packets, takes 5 seconds per packet on average, increases header length by 10 with a route inflation of 7/5

c. C delivers 70% of the packets, takes 0.1 seconds per packet on average, increases header length by 0 with a route inflation of 7/5

d. D delivers 70% of the packets, takes 0.1 seconds per packet on average, increases header length by 0 with a route inflation of 8/5

e. E delivers 70% of the packets, takes 0.1 seconds per packet on average our ranking scheme will rank these entries as

i. C

ii. D

iii. E

iv. B

v. A

4. Your routing algorithm can do anything it wants to guarantee the packets get to the end. Which is to say, that while you may not have packets traverse non-existent links, your routing algorithm can be as fancy as you wish.

5. You **may not** ask the TAs any questions about this, you **may not** post to Piazza about this assignment. This is a competition and you must carry this task out in total secrecy. The one exception to this rule is that you should e-mail apanda@cs.berkeley.edu if you think you have found a bug in the framework, of course after you are sure that this is in fact the case.

6. You may not change the framework itself. In particular we know that Python is self modifying and you could "improve" the simulation to your advantage. Don't do this, for one it is not very sportsman like, for another we will read the code for whoever is coming first or second, and will disqualify entries. So there's nothing to be gained from this.

7. An individual switch should use no more than 10 MB of storage regardless of traffic pattern or topology. You can assume that we will test on topologies no larger than a 100 switches, and with no more than a 100 end hosts.

Simulator

This project uses the same simulator that was used for project 2. Nothing has been changed, though a couple of utilities and flags have been added for your convenience. The framework can be acquired by cloning git://github.com/apanda/routing-race.git.

We recommend using the newly added routing_race.py for testing your switch implementations. routing_race.py is a Python file similar to simulator.py which was used for project 2, but requires a text file describing both the topology and a schedule for link failures, recoveries, and pings being sent. An example input file can be found in tests/small_schedule.txt. Let us start by looking at a very simple test file

```
create host h1
create host h2
create switch s1
link h1 s1
link h2 s1
ping h1 h2 0.5
ping h1 h2 0.6
unlink h1 s1 0.7
ping h1 h2 0.7
ping h1 h2 0.8
link h1 s1 1.2
```

The first three lines declare that this topology has two end hosts h1, and h2, and one switch s1. All statements other than create take up to three arguments: source, destination, and an optional time argument (in seconds). Any statements without the time argument are executed when the topology is being set up.

Therefore the test schedule above starts off by creating a topology where h1 is connected to s1, and h2 is also connected to s1. Initially h1 sends a ping packet to h2 0.5 seconds after the start of the simulation. 0.7 seconds after the simulation starts the h1 is disconnected from s1, and the link is finally restored 1.2 seconds after the simulation start. Your switches will be tested against similar, but more complex. To run routing_race.py with this schedule you would do something similar to what is shown below:

```
python routing_race.py tests/small_schedule.txt
INFO:simulator:h1 up!
INFO:simulator:h2 up!
INFO:simulator:s1 up!

INFO:user:linking h1->s1>>>
INFO:user:linking h2->s1
```

```
INFO:user:h1:tx: 1353437522 <Ping h1->h2 ttl:255: 0>
INFO:user:h1:tx: 1353437522 <Ping h1->h2 ttl:255: 1>
INFO:user:unlinking h1->s1
INFO:user:h1:tx: 1353437522 <Ping h1->h2 ttl:255: 2>
INFO:user:h1:tx: 1353437522 <Ping h1->h2 ttl:255: 3>
INFO:user:linking h1->s1
INFO:user:h2:rx: 1353437523 <Ping h1->h2 ttl:253: 0> s1,h2
INFO:user:h2:tx: 1353437523 <Pong <Ping h1->h2 ttl:253: 0>>
INFO:user:h2:rx: 1353437523 <Ping h1->h2 ttl:253: 1> s1,h2
INFO:user:h2:tx: 1353437523 <Pong <Ping h1->h2 ttl:253: 1>>
INFO:user:h1:rx: 1353437524 <Pong <Ping h1->h2 ttl:253: 0>> s1,h1
INFO:user:h1:rx: 1353437524 <Pong <Ping h1->h2 ttl:253: 0>> s1,h1
INFO:user:h1:rx: 1353437524 <Pong <Ping h1->h2 ttl:253: 1>> s1,h1
INFO:user:h1:rx: 1353437524 <Pong <Ping h1->h2 ttl:253: 1>> s1,h1
```

The tx and rx lines in the output above tell you about the exact time when a packet was sent out and received by the different hosts. The ping packets now carry a sequence number which allows you to distinguish between them, and therefore figure out what made it through, and how long it too.

To change the switch used by routing_race.py (currently it uses a Hub) you should change line 18 in routing_race.py.

Getting Started

1. Clone the repository at https://github.com/apanda/routing-race. This can be done by running git clone git://github.com/apanda/routing-race.git
2. Read the rules, make sure you want to participate.
3. E-mail apanda@cs.berkeley.edu with [EE122-Routing Race] in the subject line, and the word "shibboleth" in the body.
4. Work on the routing race.

Handing In, Participating, Etc

If you are actually participating in the routing race, please send an e-mail with [EE122-Routing Race] in the subject line, and "shibboleth" in the body to apanda@cs.berkeley.edu by November 27th so we can actually figure out how many people are participating. We will announce a handin procedure in the next week.

Cheating and Other Things

Do not modify the simulator code, we will look through the code of the winner, so do not modify the simulator to get more packets through.

Do not share code with anyone other than your teammates. In particular do not have any public repositories. While this race is not for credit, the same rules of academic honesty apply here as have applied for any of your projects, so do not cheat.