Overlay Traffic Routing in EGOIST

November 12, 2007

The purpose of this project is to develop a routing tool in order to send traffic over EGOIST.

For this project you will need $n+1$ dedicated machines. n machines will form the overlay and 1 machine will be the coordinator. Save the list of the peers in `nodes.dat`, *e.g.*,
1.2.3.1
1.2.3.2
1.2.3.3
...

and the IP of the coordinator in `coordinator.dat`. A routing daemon is running in each peer waiting for routing requests.

Assuming that the coordinator is aware of the optimized topology of EGOIST, it can initiate a transfer from any peer of the overlay to any other peer. Let v_s be the source peer and v_t be the target peer. The coordinator can send a request to v_s (in the command line):

```
routetraffic -s 1.2.3.1 -t 1.2.3.4 -p path.dat -v 0.8
```

where 1.2.3.1 is the source peer, 1.2.3.4 is the target peer, `path.dat` is the path that has to be routed, following this format:
1.2.3.2
1.2.3.6
1.2.3.9
1.2.3.4

and 0.8 is the traffic volume that has to be sent in Mbytes. Upon the above request, 0.8 Mbytes have to be generated and routed: 1.2.3.1 $\rightarrow$ 1.2.3.2 $\rightarrow$ 1.2.3.6 $\rightarrow$ 1.2.3.9 $\rightarrow$ 1.2.3.4. The total routed traffic must not be stored-and-forwarded, but rather the individual packets. The easiest way to achieve that is by encapsulating the IPs in the header, removing one IP per overlay hop. You can use the standard packet size (1500bytes) and padding if needed.

Each peer (either a source or an intermediate peer) must be able to handle multiple requests, *e.g.*,

```
routetraffic -s 1.2.3.1 -t 1.2.3.4 -p path.dat -v 0.8
routetraffic -s 1.2.3.1 -t 1.2.3.7 -p path2.dat -v 1.2
routetraffic -s 1.2.3.6 -t 1.2.3.4 -p path3.dat -v 1.8
routetraffic -s 1.2.3.4 -t 1.2.3.1 -p path4.dat -v 2.5
```

Statistics about the request service time must be maintained in the source peer in the file `log.dat`, following this format:

```
1.2.3.1  1.2.3.4  path.dat  0.8  24.3  secs
1.2.3.1  1.2.3.7  path2.dat  1.2  4.6  secs
...
```

On a technical note, overlay creation where the network metric is available bandwidth is quite involved, even with the assistance of a coordinator. Also, to deploy a fully distributed version (without the need of coordinator) is quite demanding and is application specific. We have developed frameworks for these extensions. Should you finish the baseline requirement for the project and want to extend the project further for a higher grade, you are welcome to make use of these.

Requirements: UNIX and socket programming in C and/or C++. You might find it useful to develop parts of the code in python or perl.

For additional questions and clarifications, do not hesitate to contact
Mema Roussopoulou (mema@eecs.harvard.edu) and George Smaragdakis (gsmaragd@cs.bu.edu)