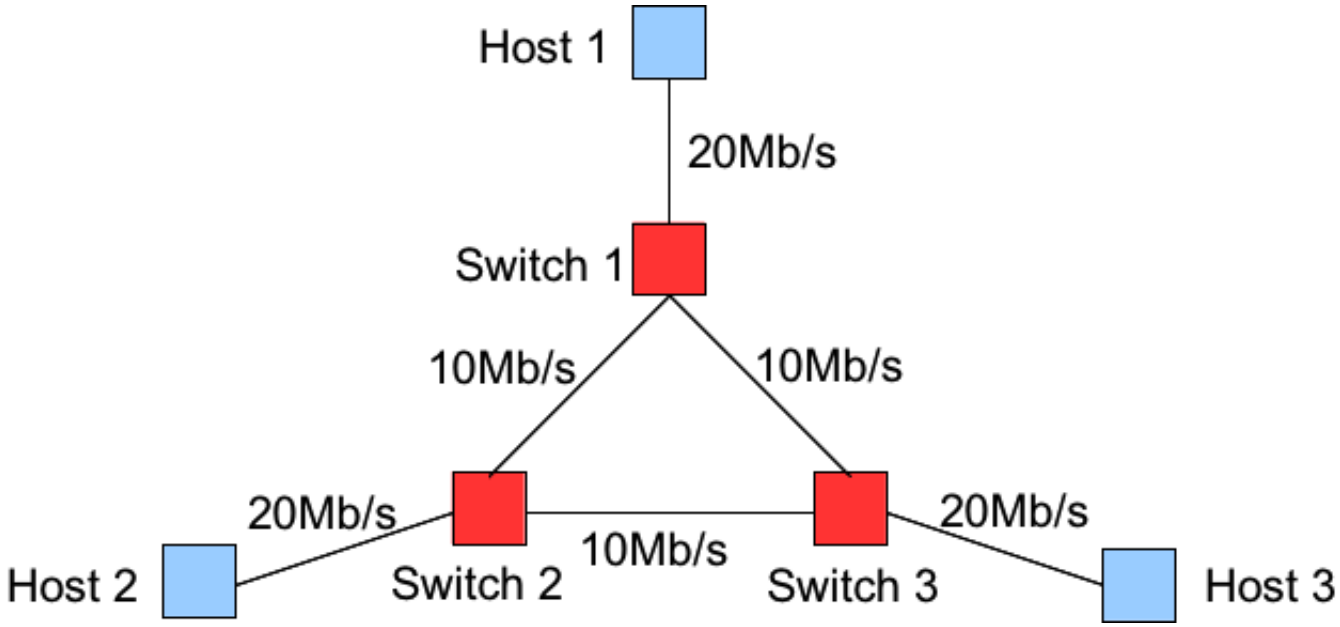
Metode avansate în sisteme distribuite (Seria: AAC)

Challenge 2: Smart Traffic Management



You are given the topology in the figure above, containing three Openflow switches (red boxes) and three hosts (blue boxes). The links between the switches are 10Mb/s, and each host connects to one switch using a 20Mb/s link.

For every connection there is a path that passes through two switches (the short path), and a path that passes through three switches (the long path). You need to use Openflow to route some flows on the long paths.

The controller will have access to flow information from the Openflow switches and will be able to also control the switches. The controller has no access to any host state.

Your task

Your task is two-fold: you will have to implement a controller that takes advantage of the multiple existing paths and to run some experiments comparing the two.

The controller(s) must implement the following strategies (selected manually at startup):

- For non-MPTCP traffic: Round-robin (or a variation thereof).
- For MPTCP traffic: Pushing one subflow down a path and the second subflow down the other path.

The two experimental setups that you are expected to work with are: using standard TCP and using MPTCP with two subflows (each with its appropriate controller strategy).

The goal of the experiments is to come up with a reproducible scenario which, by some measure, runs significantly better under one of the experimental setups than under the other. Any of the following can be considered underperformance:

- Underutilization of a link
- Needless oversubscription of a link
- Unfairness toward traffic travelling down a short path
- Disruption of a long-lived flow
- etc.

For the purpose of generating the traffic, you are free to use whatever software you want, as long as it's freely available and/or open source.

Hints

Regarding the controller:

- You only need to concern yourself with TCP traffic.
- ARP can be easily dealt with: you can add static entries on each of the hosts (use "arp -s")
- Try to minimize the number of flow mods you send. A switch can easily tell if the traffic it receives from another switch is destined for the nearest host or for the other switch. Only traffic coming straight from the adjacent host needs per-connection flow entries.
- The ingress port numbers, MACs, IPs etc. are predictable; use this to your advantage. There's no need to try to write topology discovery code.

Regarding MPTCP:

- Use the ndiffports path manager (sysctl -w net.mptcp.mptcp_path_manager=ndiffports) and set the numer of subflows to 2 (sysctl -w net.mptcp.mptcp_ndiffports=2).
- If you need to revert back to vanilla TCP, set the path manager to default.
- You can determine which the secondary subflow is by looking at the initial SYN's TCP options. The option you are looking for is MP_JOIN. (See RFC 6824.)

Regarding the experiments:

- You needn't be thorough with your experiments. One clearly documented instance of underperformance is enough.
- The scenario has to be somewhat automated. It is ok if some human interaction is needed, but anything complex must be scripted.
- You can use an Apache web server and wget to simulate HTTP traffic (both short and long flows).
- You can use iperf to generate long-lived connections.
- We have provided a traffic generator that creates multiple short flows (see resources).

- You must provide graphs that showcase the difference in performance and a rationale of why the behaviour that you are seeing is undesirable.

Submission

You should submit an archive containing the following:

- All programs/scripts
- A document explaining your approach, with an emphasis on the experiments
- A graph (or multiple graphs) that shows how one experimental setup offers better performance than the other

Resources

1. Virtual machine images https://github.com/mininet/mininet/wiki/Mininet-VM-Images
2. MPTCP installation http://multipath-tcp.org/pmwiki.php/Users/AptRepository
3. Topology file http://nets.cs.pub.ro/~vlad/atds/challenge2/challenge2-topo.py (use the following extra Mininet flags: --link=tc --mac)
4. Traffic generator http://nets.cs.pub.ro/~vlad/atds/challenge2/tg.tgz (To execute it you must first start nc_multi -l port_num on each server (this waits for connections). Next, to generate traffic run tg using as parameters the address of the remote server and the port, the local ip address to use, the frequency of connections (arrivals/sec) and the total number of connections generated.)
5. RFC 6824 http://tools.ietf.org/html/rfc6824

Got an idea?

Drop us a line if you would like to do something different.

Submission status

Submission status	No attempt
Grading status	Not graded
Termen predare	duminică, 23 noiembrie 2014, 23:55
Time remaining	9 zile 10 ore

Add submission

Make changes to your submission

MENIU PRINCIPAL

Pagina mea

- Pagina principală
  - Pagini site
  - Profilul meu
  - Current course
    - M-2-MAS-AAC
      - Participanţi
      - Badges
      - General
      - Forum Stiri
      - Forum Discutii Generale
      - Challenge 1: Covert Channels
      - **Challenge 2: Smart Traffic Management**
      - 6 octombrie - 12 octombrie
      - 13 octombrie - 19 octombrie
      - 20 octombrie - 26 octombrie
      - 27 octombrie - 2 noiembrie
      - 3 noiembrie - 9 noiembrie
      - 10 noiembrie - 16 noiembrie
      - 17 noiembrie - 23 noiembrie
      - 24 noiembrie - 30 noiembrie
      - 1 decembrie - 7 decembrie
      - 8 decembrie - 14 decembrie
      - 15 decembrie - 21 decembrie
      - 22 decembrie - 28 decembrie
      - 29 decembrie - 4 ianuarie
      - 5 ianuarie - 11 ianuarie
      - 12 ianuarie - 18 ianuarie
      - 19 ianuarie - 25 ianuarie
  - Cursurile mele

SETĂRI

- Administrare curs
- Setări profilul meu

OPENSTACK BETA

Create user

Check status

Goto Dashboard
Goto Documentation

Sunteţi autentificat ca Constantin-Claudiu GHIOC (Ieşire)
M-2-MAS-AAC