Lab8: Flow Fairness and Network Routing

COM-208: Computer Networks

Objectives

- Study how competing flows share network resources.
- Experiment with flow characteristics to discover how each of these affects fairness.
- Examine how routing protocols react to changes to network connectivity.

Prerequisites:

- Lab 7 (Introduction to ns2 and Transport-layer protocols).
- Textbook Chapters 3 and 4.

Flow Fairness

Shared bottleneck link

In this part of the Lab, we study how competing TCP flows with similar characteristics behave when they share a single bottleneck link.

Setup

We will use the following files: tp_fairness.tcl, fairness_pps.plot, and fairness_pkt.plot. You can find them on Moodle in the scripts directory.

tp_fairness.tcl generates 5 source-destination pairs which all share a common network link. Each source uses a single TCP flow which transfers FTP traffic to the respective destination. The flows are created one after the other at 5-second intervals (i.e., flow `i` ↪ `+1` starts 5 seconds after flow `i` for `i` in `[1,4]`). You can invoke tp_fairness.tcl as follows:

```
$ ns tp_fairness.tcl
```

The figure below shows the resulting topology; there are 5 sources (2,4,6,8,10), 5 destinations (3,5,7,9,11), and each source is sending a large file to a single destination.

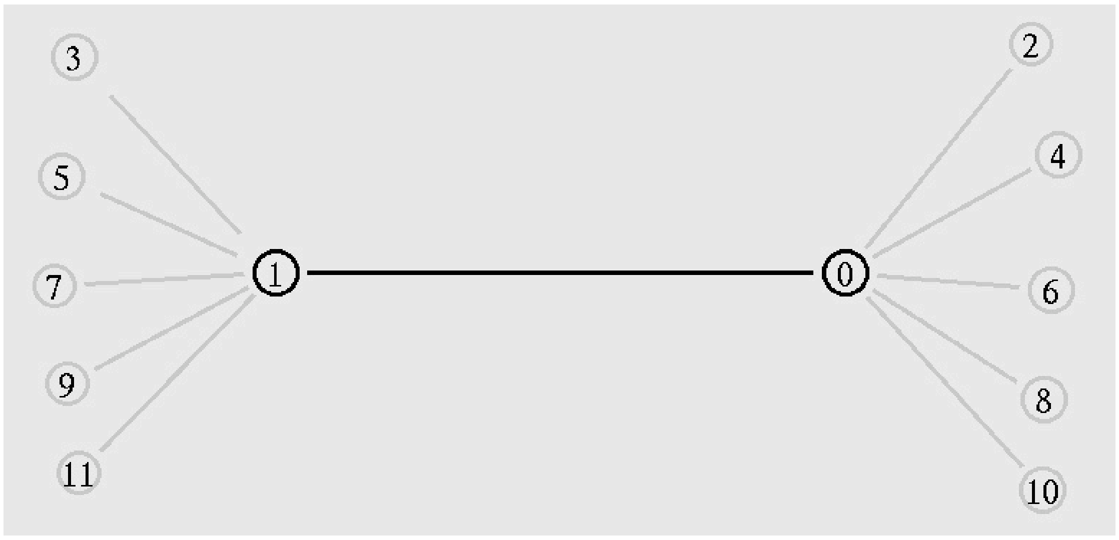

Figure 1: Topology

Output

The tp_fairness.tcl script produces one output file per flow; fairnessMon*[i]*.tr, for each `i` ↪ `in [1,5]`. Each of these files contains three columns:

```
time | number of packets delivered so far | throughput (packets per second)
```

You can plot the following statistics for all flows using the plotting scripts provided:

- Throughput as a function of time:

```
$ gnuplot fairness_pps.plot
```

- Sum of packets delivered as a function of time:

```
$ gnuplot fairness_pkt.plot
```

If you want to display the NAM window (graphical interface), modify tp_fairness.tcl and uncomment the fifth line of the `finish` procedure:

```
proc finish {} {
 global ns file1 file2
 $ns flush-trace
 close $file1
 close $file2
 # exec nam out.nam &
 exit 0
}
```

Questions

Run the tp_fairness.tcl script and observe the output of the fairness_pps.plot plotting script (explained above). Answer the following questions:

- Does each flow get an equal share of the capacity of the common link (i.e., is TCP fair)?
  - Explain which observations lead you to this conclusion.
- What happens to the throughput of the pre-existing TCP flows when a new flow is created?
  - Explain the mechanisms of TCP which contribute to this behavior.
  - Argue about whether you consider this behavior to be fair or unfair.

Now, use the fairness_pkt.plot plotting script and answer the same set of questions. Does this plot confirm your intuitions?

RTT and fairness.

In this part of the Lab, we will examine how two competing flows behave when they experience different round-trip times to their destination. (i.e., the RTT of one flow is longer than the RTT of the other flow).

Setup

We will use the following files: tp_fairness2.tcl, fairness2_pps.plot, and fairness2_wnd.plot. You can find the files on Moodle in scripts file.

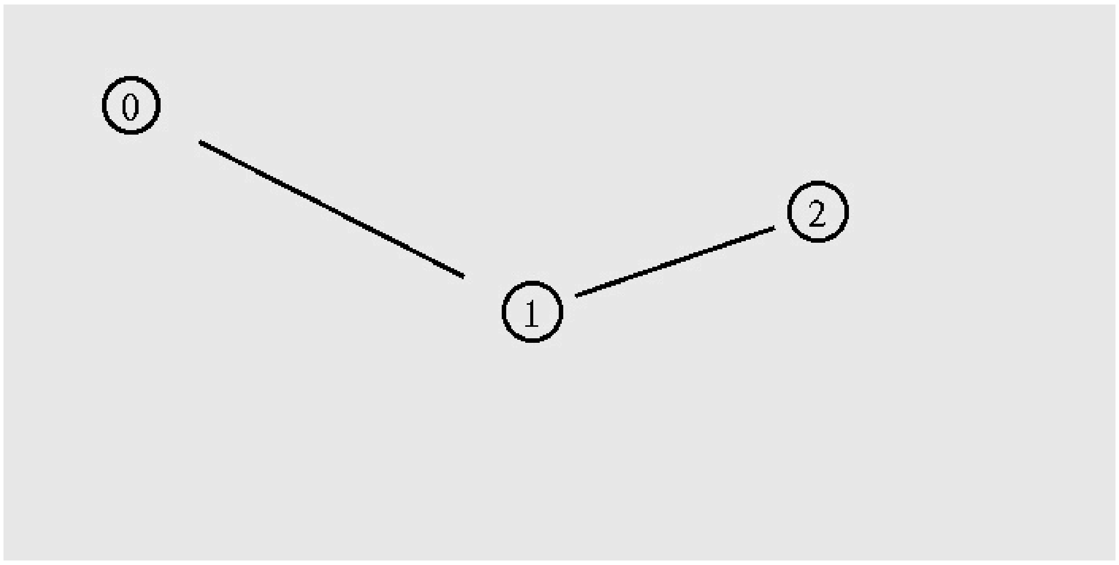

Figure 2: Topology

The tp_fairness2.tcl script takes 4 arguments and generates the simple 3-node network topology shown in the figure above. The 4 arguments of the ns script are:

1. The capacity of link 0-1 (i.e., the link between node 0 and node 1).
2. The capacity of link 1-2.
3. The delay of link 0-1.
4. The delay of link 1-2.

Output

tp_fairness2.tcl also generates some TCP flows. You can plot the following statistics for all flows using the plotting scripts provided:

- Throughput as a function of time:

  ```
  $ gnuplot fairness2_pps.plot
  ```

- Size of TCP congestion window as a function of time:

  ```
  $ gnuplot fairness2_wnd.plot
  ```

Questions

Execute the tp_fairness2.tcl script as follows:

```
$ ns tp_fairness2.tcl 1Mb 1Mb 10ms 50ms
```

Observe the NAM window output and try to answer the following questions:

- Which nodes communicate with which other nodes? Hint: Different colors of NAM flows show different connections.
- Is there any flow that appears to starve another flow?

Observe the outputs of fairness2_pps.plot and fairness2_wnd.plot plotting scripts.

- How does the network delay of link 1-2 affect fairness?
- Why does the size of the TCP congestion window differ for the competing flows? Try to come up with a simple theory about why this happens. Hint: the link queue size is 10 packets and the maximum congestion window is 30 packets (see Line 9 and 10 of tp_fairness2.tcl script).
- What will happen if the link queue size becomes considerably larger than the maximum congestion window? Do you believe that this will affect fairness?

Now, it is time for you to verify your hypotheses. Make the following changes to the tp_fairness2.tcl file: increase the link queue size from 10 packets to 60 packets, and increase the simulation duration from 10 secs to 100 secs (the latter will give more time for the flows to stabilize).

- Does this experiment confirm your intuition (i.e., did this modification affect fairness at all)?

TCP competing with UDP

In this part of the Lab, we are going to see how a TCP flow reacts when it has to share a bottleneck link that is also used by a UDP flow.

Setup

We will use the following files: tp_TCPUDP.tcl and TCPUDP_pps.plot. You can find them on Moodle in scripts file.

The tp_TCPUDP.tcl script takes a link capacity value as an argument. It creates a link with the given capacity and creates two flows which traverse that link, one UDP flow and one TCP flow. A traffic generator creates new data for each of these flows at a rate of 4Mbps.

Output

When you are done running the simulation, you can plot the throughput of the two flows as follows:

```
$ gnuplot TCPUDP_pps.plot
```

Questions

- How do you expect the TCP flow and the UDP flow to behave if the capacity of the link is 5Mbps?

Now, you can use simulation to test your hypothesis. Execute the tp_TCPUDP.tcl script as follows:

```
$ ns tp_TCPUDP.tcl 5Mb
```

Use TCPUDP_pps.plot to plot the throughput of the two flows.

- Why does one flow achieve higher throughput than the other? Try to explain what mechanisms force the two flows to stabilize to the observed throughput
- List the advantages and the disadvantages of using UDP instead of TCP for a file transfer, when our connection has to compete with other flows for the same link. What would happen if everybody started using UDP instead of TCP for that same reason?

Network Routing

In this part of the Lab, we are going to see how routing protocols react when network conditions change (e.g., a network link fails).

Setup

We will use the following file: tp_routing.tcl. You can find the file on Moodle in scripts.zip file.

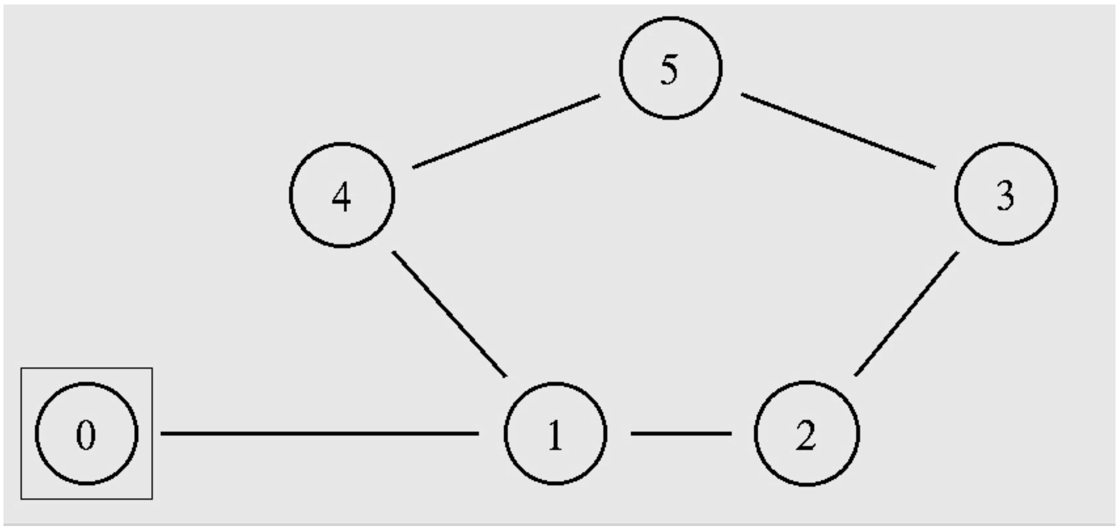

Figure 3: Topology

The tp_routing.tcl script takes no arguments and generates the network topology shown in the figure above. You can run the simulation with the following command:

```
$ ns tp_routing.tcl
```

Questions

Run the tp_routing.tcl script and observe the NAM window output. You may have to refer to the source code of tp_routing.tcl to answer the following questions:

- Which nodes communicate with which other nodes?
- Which is the transport protocol used?
- Which route do the packets follow? Does it change over time?

Modify tp_routing.tcl and add the following lines just before command `ns at 0.5 "` `↪ $cbr0 start"`:

```
$ns rtmodel-at 1.0 down $n1 $n4
$ns rtmodel-at 1.2 up $n1 $n4
```

Rerun the simulation and observe the NAM window output.

- What happens at time 1.0 and at time 1.2?
- Does the route between the communicating nodes change as a result of that?

The nodes in the topology use a static routing protocol (i.e., preferred routes do not change over time). We are going to change that, so that they use a Distance-Vector routing protocol. Modify tp_routing.tcl and add the following line before the definition of the finish procedure:

```
$ns rtproto DV
```

Rerun the simulation.

- How does the network react to the changes that take place at time 1.0 and time 1.2?

Remove these two lines you previously added ($ns rtmodel-at 1.0 down $n1 $n4 and $ns rtmodel-at 1.2 up $n1 $n4) and add the following the following line, instead:

```
$ns cost $n1 $n4 3
```

- How does this change affect routing? Explain why.

Now, replace the line you just added with:

```
$ns cost $n1 $n4 2
```

and uncomment the following line, which is located right after the finish procedure definition:

```
 Node set multiPath_ 1
```

- Describe what happens and deduce the effect of the line you just uncommented.