TCP: The Nuts and Bolts

Simulation and Analysis with NS3

Lab 5.2: CS3210 - Computer Networks Lab

Instructor: Krishna M. Sivalingam
Due Date: 11.55PM, 22rd March, 2015
Jan – May 2015, Indian Institute of Technology Madras

TCP and Congestion: A bird's eye view

TCP is a connection-oriented transport layer protocol that provides an end-to-end reliable, ordered and error-corrected delivery of data. In order to make these strong guarantees, TCP faces several challenges with respect to performance/throughput in different kinds of traffic scenarios. TCP has control only over the ends of the connection, but not over the nodes of the network that are in between. A TCP connection also does not have information about other connections that exist over the same network. Since this global information is unavailable to each connection, it is a challenge to ensure that the network resources are effectively utilized - for example: If there are no other connections shared with the intermediate nodes and the TCP sender sends at a very low rate, the network is underutilized; if there are too many other connections shared with the same intermediate nodes, and the sender sends at a high rate, excessive packet drops and retransmissions may kill throughput (See congestion collapse). To address these challenges, TCP implements several mechanisms to control congestion on the network e.g., slow start, congestion avoidance, fast recovery, fast retransmit etc. Essentially, a TCP connection keeps listening to the network for feedback about packet drops. Based on the current state of the connection and the nature of the drop, it re-configures the connection's behavior and some of its parameters on the fly (congestion window, sending rate etc.) with the hope of improving throughput. Most of these mechanisms are empirically evaluated on real networks. Also, different combinations of these mechanisms have resulted in different flavours of TCP - TCP Reno, TCP Vegas, TCP New Reno etc.

The Assignment

In this assignment, using NS3, your task is to gain a thorough understanding of TCP and the rationales behind the mechanisms that TCP implements. By seeing TCP in action, you will observe and analyze how variants of TCP perform and will also come up with a TCP variant of your own.

1 TCP Flavours

Using NS3 libraries, perform the following experiment. Link bandwidths/ delay values can be your choice.

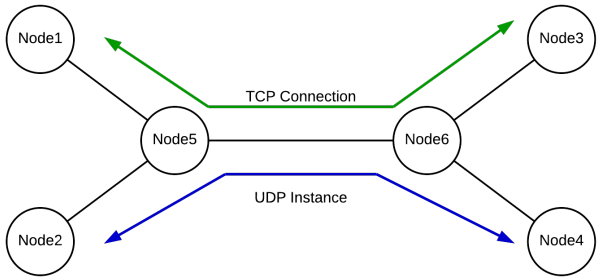

Figure 1: Setup 1

- Create a simple dumbbell topology, two clients `Node1` and `Node2` on the left side of the dumbbell and server nodes `Node3` and `Node4` on the right side of the dumbbell. Let `Node5` and `Node6` form the bridge of the dumbbell. Use point to point links.
- Install a TCP socket instance on `Node1` that will connect to `Node3`. Install a UDP socket instance on `Node2` that will connect to `Node4`.
- Start the TCP application at time 1s. Start the UDP application at time 20s at rate `Rate1` such that it clogs half the dumbbell bridge's link capacity.
- Increase the UDP application's rate at time 30s to rate `Rate2` such that it clogs the whole of the dumbbell bridge's capacity.
- Use the NS3 tracing mechanism to record changes in congestion window size of the TCP instance over time. Use gnuplot/matplotlib to visualise plots of `cwnd` vs time. Mark points of fast recovery and slow start in the graphs (Use an image editor of your choice to mark - e.g., gimp/inkscape).
- Perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with NS3. Do a comparative study of their behavior.

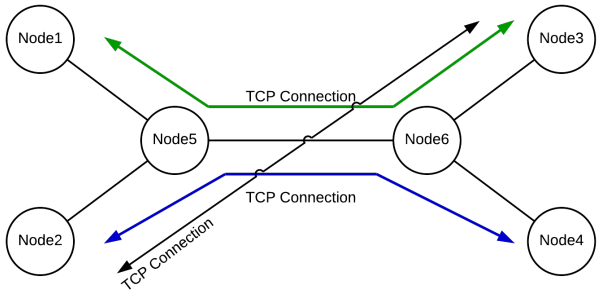

Figure 2: Setup 2

- Modify the above experiment to simulate Setup 2 with the following changes. (1) Install a **TCP Reno** socket instance on `Node1` that will connect to `Node3`. Install a **TCP NewReno** socket instance on `Node2` that will connect to `Node3`. Install a **TCP Tahoe** socket instance on `Node2` that will connect to `Node4`. (2) Start `Node1`–`Node3` flow at time 1s, Start `Node2`–`Node3` and `Node2`–`Node4` flows at time 15s. Record and plot congestion window changes over time for each of the three connections. Present your observations.

NOTE: To get started off quickly, you can refer to and re-use/modify sample code given in **moodle** and the TCP examples in `examples/tcp/`

2 Custom TCP Variant

In this section, you will create a TCP variant of your own. The source code for the pre-defined TCP variants are available in `src/internet/model/tcp-<variant>.h & .cc` If you carefully read the code, you can see how each of the variants is programmed to achieve the corresponding functionalities.

2.1 Setting up

To get started off with your own variant, you will do the following:

- Come up with a name for your TCP variant, say X.
- Copy `tcp-reno.h` and `tcp-reno.cc` into `tcp-X.h` and `tcp-X.cc` in `src/internet/model/`
- Replace all instances of `TcpReno` in the `.cc` file with `TcpX`. Change the header include statement to `#include "tcp-X.h"`. Also make similar changes in `tcp-X.h, also in the #IFNDEF`.
- In the `wscript` in `src/internet/`, add your `tcp-X.h` into `headers.source` and `tcp-X.cc` into `objects.source`
- Recompile using `./waf configure --enable-examples --enable-tests` and `./waf`
- Now you should be able to use the following command to use your tcp variant
  `Config::SetDefault ("ns3::TcpL4Protocol::SocketType", TypeIdValue (TcpX::GetTypeId ()));`

2.2 Writing the variant

Now modify the code in `tcp-X.h` and `tcp-X.cc` to your liking. You are free to implement your variant in any way you like. You can change existing parameters (Eg. Initial CW, CW after timeout, cwnd threshold for linear increase, Dup ACK threshold etc.) and/or modify the existing functions and/or add functions to make your variant behave in the way that you design (Note that this will require you to carefully go through the existing code first). This is an opportunity for you to try out any new ideas that you might have to make TCP perform better.

You must make at least three major changes (each can be either parameter change or functionality change).

2.3 Comparison

Compare the performance of your TCP variant with the other TCP variants in the previous experiment and report the results. (For the Setup 2 part - only replace the `Node1-Node3` TCP connection with your variant).

3 Deliverables

Create a folder `Lab5-2-CS1xBabc`, the main directory.

3.1 Code

Within the main directory, create a folder called `code`. For each of the three experiments, create a separate folder within `code` called `exp-n` where `n` is the experiment number. Each `exp-n` folder will have the corresponding code and the packet captures/traces.

3.2 Report

Within the main directory, create a folder called `report`, which contains your report in PDF, and also a COMMENTS file that describes your experience with the assignment.
NOTE: Your report must contain all relevant explanation, graph plots, and observations for **both** the experiments. For the custom tcp variant, provide a detailed explanation of your scheme: all the changes made, the rationale behind the modifications and functionalities that have been introduced. Also cite all references (see below).

Reference Policy

This is an INDIVIDUAL assignment. You are allowed to refer online sources to implement your code. You MUST cite every reference that you used in your report. If you have taken help from an online source and have not cited it, you will forfeit all the marks for the assignment.