MENU

CS 4700 / CS 5700 - Network Fundamentals

Project 3: Performance Analysis of TCP Variants

This first milestone for this project is due at 11:59pm on October 10, 2016.
The final version of the project is due at 11:59pm on October 24, 2016.

Assignment Overview

The original design of the Transmission Control Protocol (TCP) worked reliably, but was unable to provide acceptable performance in large and congested networks. Several TCP variants have been proposed since then (TCP Tahoe, Reno, NewReno, Vegas, BIC, CUBIC, SACK, and others) with mechanisms to control and avoid congestion. The objective of this project is for you to analyze the performance of these different TCP variants. You will use the NS-2 network simulator to perform experiments that will help you understand the behavior of the TCP variants under various load conditions and queuing algorithms.

In order to perform a complete analysis of the TCP variants, you should first read the paper "Simulation-based Comparisons of Tahoe, Reno and SACK TCP" (papers/tcp-sim.pdf) to gain a better understanding of the type of analysis you are required to do. The paper also includes details on the relevant TCP variants used in this assignment.

The final product of this assignment is a report that describes the results of your simulated experiments. This report will follow the formatting of a submission to an academic workshop. Your report must be <=6 pages long, and include an introduction, a methodology section, sections for experimental results, and a conclusion. Basically, your report should look like the papers we have been reading in class, just shorter.

Getting Started With NS-2

You will find the complete NS-2 package installed on the CCIS Linux machines in the following directory: /course/cs4700f12/ns-allinone-2.35/. In order to run NS-2, use the following command:

% /course/cs4700f12/ns-allinone-2.35/bin/ns <your_program.tcl>

For example, download: ex-tcl.tcl (http://nile.wpi.edu/NS/Example/ex-tcl.tcl) and enter:

% /course/cs4700f12/ns-allinone-2.35/bin/ns ex-tcl.tcl

NS-2 will output:

```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

You can save yourself the time and trouble of typing in the full path to NS-2 by modifying the PATH variable of your shell.

As you can see in the above example, each NS-2 simulation is controlled by a script file written in TCL. There are several excellent online resources that will tell you how to write NS-2 scripts:

- NS by example (http://nile.wpi.edu/NS/): An excellent tutorial with many examples
- Marc Greis's tutorial (http://www.isi.edu/nsnam/ns/tutorial/index.html) (Alternate link (http://cs.tju.edu.cn/faculty/byr/NS2/ns-turorial/nsintro.html)): The most official tutorial from the NS-2 website
- The NS-2 Manual (http://www.isi.edu/nsnam/ns/ns-documentation.html) (Alternate link (http://zfadlullah.org/files/course-material/information-engineering/Additional/ns_kiso.pdf))

Generating Graphs from NS-2 Data

In order to analyze the results of your NS-2 simulations, you should enable NS-2's trace file by putting the following lines of code in your script:

```
#Open the trace file (before you start the experiment!)
set tf [open my_experimental_output.tr w]
$ns trace-all $tf
...
# Close the trace file (after you finish the experiment!)
close $tf
```

These commands will tell NS-2 to create an output file that includes traces of all the packets sent during your experiment. You will need to write other scripts (in whatever language you choose) that parse this data and analyze the results. Your scripts should report things like average bandwidth, packet drops, throughput over time, end-to-end latency (average and over time), sequence numbers of packets sent, and the congestion window of all TCP flows. This information will enable you to analyze the results of your experiments and answer the questions below. You need to understand the format of NS-2s trace file in order to successfully write these scripts.

My suggestion would be to write simple Python scripts to parse/analyze the trace (e.g. compute the average throughput of each flow), output the results to a file, and then use gnuplot or Excel to plot graphs of the data.

Note that many of the NS-2 tutorials refer to tools like nam and xgraph that come bundled with NS-2. I have never used these tools, and I do not recommend that you use them. However, these tools are\ installed in the NS-2 directory and you may use them if you wish. Do not ask for technical support about these tools, because we cannot help you!

Experiment 1: TCP Performance Under Congestion

In the first set of experiments, you will analyze the performance of TCP variants (Tahoe, Reno, NewReno, and Vegas) under the influence of various load conditions. To study the behavior of the four TCP variants, you need to first set up a network topology. Use the following topology for your experiments, and set the bandwidth of each link to 10Mbps:

```
N1                  N4
  \                /
   \              /
    N2------------N3
   /              \
  /                \
N5                  N6
```

Using this topology, you will perform tests that analyze the performance of different TCP variants in the presence of a Constant Bit Rate (CBR) flow. You can think of the CBR flow as an unresponsive UDP flow: it uses a specific amount of bandwidth, it does not care about dropped packets, and it does not perform congestion control. Basically, it just sends packets blindly at a constant rate.

Add a CBR source at N2 and a sink at N3, then add a single TCP stream from N1 to a sink at N4. Analyze the throughput, packet drop rate, and latency of the TCP stream as a function of the bandwidth used by the CBR flow. That is, the bandwidth of the CBR flow is the parameter you need to vary for this experiment. For example, start the CBR flow at a rate of 1 Mbps and record the performance of the TCP flow. Then change the CBR flow's rate to 2 Mbps and perform the test again. Keep changing the CBR's rate until it reaches the bottleneck capacity. The TCP stream's performance will change depending on the amount of contention with the CBR flow. Conduct this experiment using the following TCP variants: Tahoe, Reno, NewReno, and Vegas.

Based on the results from your tests you should be able to answer the following questions: which TCP variant(s) are able to get higher average throughput? Which has the lowest average latency? Which has the fewest drops? Is there an overall "best" TCP variant in this experiment, or does the "best" variant vary depending on other circumstances?

Experiment 2: Fairness Between TCP Variants

Next, you will conduct experiments to analyze the fairness between different TCP variants. Out on the Internet, there are many different operating systems, each of which may use a different variant of TCP. Ideally, all of these variants should be fair to one another, i.e. no particular variant gets more bandwidth than the others. However, is this ideal assumption actually true?

For these experiments, you will start three flows: one CBR, and two TCP. Add a CBR source at N2 and a sink at N3, then add two TCP streams from N1 to N4 and N5 to N6, respectively. As in Experiment 1, plot the average throughput, packet loss rate, and latency of each TCP flow as a function of the bandwidth used by the CBR flow. Repeat the experiments using the following pairs of TCP variants (i.e. with one flow from N1 to N4, and the other flow from N5 to N6):

- Reno/Reno
- NewReno/Reno
- Vegas/Vegas
- NewReno/Vegas

Based on your results you should be able to answer the following questions: are the different combinations of variants fair to each other? Are there combinations that are unfair, and if so, why is the combination unfair? To explain unfairness, you will need to think critically about how the protocols are implemented and why the different choices in different TCP variants can impact fairness.

Experiment 3: Influence of Queuing

Queuing disciplines like DropTail and Random Early Drop (RED) are algorithms that control how packets in a queue are treated. In these experiments, instead of varying the rate of the CBR flow, you will study the influence of the queuing discipline used by nodes on the overall throughput of flows.

Use the same topology from experiment 1 and have one TCP flow (N1-N4) and one CBR/UDP (N5-N6) flow. First, start the TCP flow. Once the TCP flow is steady, start the CBR source and analyze how the TCP and CBR flows change under the following queuing algorithms: DropTail and RED. Perform the experiments with TCP Reno and SACK. In these tests, you will need to plot the performance of the TCP and CBR flow over time (i.e. you are no longer varying the bandwidth of the CBR flow).

Based on the results of these experiments, you should be able to answer the following questions:

- Does each queuing discipline provide fair bandwidth to each flow?
- How does the end-to-end latency for the flows differ between DropTail and RED?
- How does the TCP flow react to the creation of the CBR flow?
- Is RED a good idea while dealing with SACK?

Grading

Your report will be graded as if it were a paper submission to a workshop. Basically, if your paper is clear and concise, your methodology is sound and carefully explained, and your results are clearly presented and thoughtfully analyzed, then your report is "accepted" and you get a perfect score. The reviewers (graders) will score papers using the following guidelines:

- **Intro:** 5% of your grade. Your intro should explain what it is that you are studying and motivate why this is an important problem to study. Good introductions also include a "preview" that highlights the major experiments you will be presenting, and some of your key results.
- **Methodology:** 10% of your grade. The methodology section should clearly explain how your experiments were conducted. What tools were used? Why did you choose to use these tools, and how do you know they are sound? What were the key parameters/variables/settings used during the experiments? Why did you choose these values?
- **Results for Experiment 1:** 30% of your grade. Present and thoroughly analyze the results from the first set of experiments. Explain why the results are the way they are by discussing how the different TCP variants react to the presence of congestion. Does each variant behave the same way, and if not, why not?
- **Results for Experiment 2:** 30% of your grade. Thoroughly and carefully present the results from the two-TCP stream experiments. When there are two TCP streams, do they share bandwidth fairly, and if not, why not? If there are any unexpected or "shocking" findings, highlight them in your text and discuss what leads to these results.
- **Results for Experiment 3:** 20% of your grade. Use the same guidelines as for the first two experimental sections. Discuss the impact of different queuing techniques on each TCP variant. Why does each TCP variant react the way it does to different types of queues? Do specific combinations of TCP variants and queues work well together? Are there other combinations that should not be used together?
- **Conclusion:** 5% of your grade. Restate the significance of your study and highlight key results. Discuss the real-world significance of your results, and what the implications may be on deployed systems. Highlight any open questions that may motivate future studies.

Your report must be <=6 pages in length. Points will be subtracted for reports that are too long.

Just like any other writing assignment, you may (and are encouraged) to cite related work. You do not need to cite sources when discussing the implementation details of TCP or queue variants (it's safe to assume that the inner workings of protocols are common knowledge). However, it may be useful to cite other studies that confirm your results or use similar methodology, in order to bolster your claims.

Tips for Paper Writing

Writing scientific papers is a challenging endeavor. Here are a few tips for polishing your paper.

- Published papers are typically in font size 10.
- If you look at the papers you've been reading for homework, you'll notice that they use small graphs, usually with several graphs in a row. This is an excellent space saving technique.
- The 6 page limit means that you will not be able to include graphs/plots/tables of all your experimental results in your report. Do not attempt to cram all your results into a small number of graphs: papers get rejected all the time for having difficult to read graphs! Instead, you must pick and choose which result graphs to include in your report.
- **Always put labels on the x- and y-axis of your graphs.** If we can't interpret your graphs, then you will not receive credit for them.

Submitting Your Milestone

Before turning in your project, you and your partner(s) must register your group. To register yourself in a group, execute the following script:

```
$ /course/cs5700f16/bin/register project3 [team name]
```

This will either report back success or will give you an error message. If you have trouble registering, please contact the course staff. **You and your partner(s) must all run this script with the same [team name]**. This is how we know you are part of the same group.

There is a milestone for this project. At two weeks in, you need to turn in a plain-text README file that contains your methodology for running experiments. **You must be explicit about how you will run experiments that generalize to TCP behavior outside of the simulated environment.** This is due on **October 10th, 2016 at 11:59:59pm**.

Before submitting your README file, you must explain your methodology to one of the TAs to make sure that you address the generalization problem. In particular, it is **not** sufficient to run the same experiment with the exact same settings over and over again. Instead, you need to consider how two TCP flows might interact in general -- for example, what happens if one flow starts one, five, or even ten seconds before the next? Once you address the problem of exploring the possible behaviors, you must describe a strategy for interpreting a result. For example, if TCP A is faster than TCP B *on average*, how do you know if that is a *statistically significant* result?

Your README and any other supporting code should all be placed in a directory. You submit your project by running the turn-in script as follows:

```
$ /course/cs5700f16/bin/turnin project3-milestone [project directory]
```

[project directory] is the name of the directory with your submission. The script will print out every file that you are submitting, so make sure that it prints out all of the files you wish to submit!

Submitting Your Final Project

To turn-in your project, you should **submit your report as a pdf file**. In addition to your report, you must turn-in your NS-2 TCL scripts, as well as any custom scripts you write to parse NS-2 trace files. Your report, source code, etc. should all be placed in a directory. You submit your project by running the turn-in script as follows:

```
$ /course/cs5700f16/bin/turnin project3 [project directory]
```

[project directory] is the name of the directory with your submission. The script will print out every file that you are submitting, so make sure that it prints out all of the files you wish to submit! The turn-in script will not accept submissions that are missing a README file. **Only one group member needs to submit your project.** Your group may submit as many times as you wish; only the last submission will be graded, and the time of the last submission will determine whether your assignment is late.

Grading

This project is worth 12 points. You will receive credit based on the above guidelines for your report.

Extra Credit

There is extra credit available for this assignment. However, I will warn you in advance that this extra credit will be quite challenging (and possibly very annoying). Basically, the number of TCP variants support by NS-2 is very limited. If you implement another TCP variant (i.e. BIC, CUBIC, Hybla, Compound) in NS-2 and add it to the tests in Experiments 1, 2, and 3, you will earn 1 point of extra credit. Each additional variant that you incorporate into your report will earn you an additional bonus point. You may increase the page limit of your report by 0.5 pages for each additional variant you include.

Implementing new protocols in NS-2 is tricky. You will need to create a local installation of NS-2 and modify it's source code (it is written in C/C++). You will need to carefully vet your implementation to make sure that it is performing as expected for that particular TCP variant. There are tutorials and various hacky patches online that can show you how to modify NS-2. However, as someone who has modified NS-2 in the past, I can tell you that it is non-trivial. If you want to do the extra credit, you are on your own.

The list of possible TCP variants is not strictly limited to the ones I have listed here. If you find a different variant you would like to try and tackle, send me an email and I will approve it for you.
