

Project 4 - TCP Fast Open

Goal

The goal of this project is to learn about a specific change to TCP that reduces transfer latency. In the past lesson you learned about persistent TCP connections which enable a web browser to reuse a connection for multiple data requests. However, many HTTP requests occur over **new** TCP connections. For example, an image embedded in a web page may be located on a separate content server, or an ad is being loaded from a third party server, requiring a new TCP connection to be opened to obtain that image to properly display the webpage.

This insight led researchers at Google to investigate reducing the setup costs for a TCP connection. They developed TCP Fast Open (TFO) which permits sending data during the TCP handshake, ultimately reducing the load latency of a web resource by one round trip time (RTT). In this and the next two projects, you will dive into recent research in Computer Networks by replicating and extending the experimental results from a research paper in Mininet. In this project specifically, you will observe how TCP Fast Open improves HTTP transaction times for real websites.

The test setup provided uses a modified Chrome binary to download web pages from Mininet hosts acting as web servers. These hosts are hosting mirrored data pulled from real websites. The tests run for different latencies with TCP Fast Open disabled, and then enabled. TFO is now a part of the Linux kernel, so it is built into our course VM.

NOTE: By default, the VM boots to a MPTCP enabled kernel, which will be used in Project 5. You will need to boot a different kernel for this project only to prevent these issues.

To boot the original kernel:

1. Shut down your VM
2. Launch the VM, and click inside the VM window during the boot process.
3. Hold Shift during the boot screen.
4. The GRUB Menu should open. Select "Advanced Options for Ubuntu"

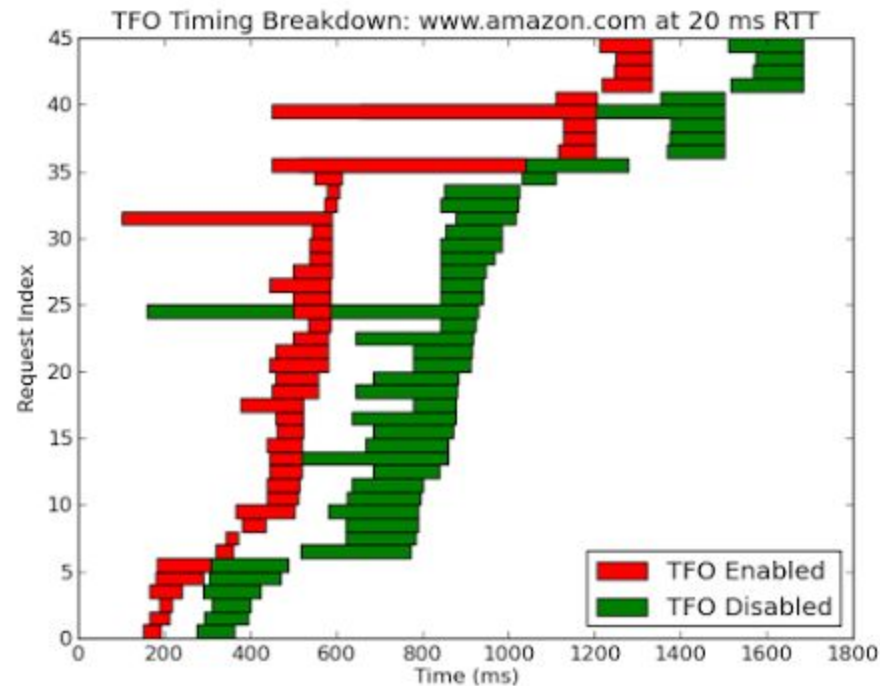
5. Select Ubuntu, with Linux 3.16.0-57 generic and hit enter.

Directions

1. Download and unzip the project code.
2. Reference the assigned [reading](#). We will be replicating the experimental results, as well as trying our own test cases. Additionally, the paper is a good reference for interpreting your results answering the questions asked at the end of the experiment.
3. Familiarize yourself with the code that will run the experiment.
 - a. `run.sh` - Shell script that runs the experiment under several different delay parameters with TFO disabled and enabled.
 - b. `tfo.py` - Python script used to execute the simulation. Step through this file at a high level to understand how the experiment is run, and compare this to the methodology used in the original paper. You SHOULD NOT modify this file.
 - c. `tfo-test/` - directory containing various scripts and executable code run on the hosts in the simulation.
 - d. `fetch.py` - script to automate downloading your own test data from the Internet.
 - e. `output-figures/` - directory for storing output performance graphs generated at the end of the experiment.
4. Run the experiment to replicate the paper by executing the following commands:
 - a. `vnc4server` (The command may ask you to set a password, you can just use `mininet` for consistency)
 - b. `sudo ./run.sh`
5. The experiment takes some time to complete. When it's finished you should see results similar to the image below. Be sure to copy and paste this text into a text file named `observations.txt` for your final submission.

Page	RTT(ms)	PLT: no TFO (s)	PLT: TFO (s)	Improv.
httpen.wikipedia.org/wiki/Transmission_Control_Protocol				
200		7419.859	5817.565	21.594669117
20		2570.471	2145.34	16.5390311737
100		4183.631	3400.245	18.7250261794
httpwww.amazon.com				
200		6111.644	4184.72	31.5287343307
20		1684.071	1332.384	20.8831456631
100		4148.951	2191.288	47.1845292943

6. You can also view graphs of the results in the `output-figures` folder. On the graph, each request index represents one of the multiple requests to load the entire web page as the HTML references multiple objects on the page. For each request index, the bars represent the start and end time to download the page with TFO enabled and TFO disabled. In some cases, the red bar for TFO Enabled may overlap the timing for TFO Disabled.



7. Next, you will run the experiment using your own set of websites. To do this, you'll need to create a `.pages` file with a list website URLs each on a single line similar to the `Paper.pages` file included with the project. For simplicity, name the file `myURLS.pages`. Be sure your file contains at least 3, but no more than 5 distinct URLs, from different domains (i.e. do not specify 5 different pages on gatech.edu).

NOTE: The script has issues with Javascript heavy web pages, as well as `https` sites. Homepages for academic institutions like GT, blog posts, and newspaper websites work well as test sites. Also, if you shutdown the virtual machine in between runs of the experiment, be sure to restart VNC Server. **You are welcome to share sites that work with the fetch script on Piazza with your classmates!**

8. Next, run the `fetch.py` script to download the web pages: `./fetch.py --name myURLS`
9. You will also need to modify the `run.sh` script on lines 21 and 29 to reference `myURLS.pages`.
10. Re-run the experiment and generate your experimental data: `sudo ./run.sh`
11. Finally, you will create a detailed yet concise (no longer than 2 pages with default font size, spacing and margins) report analyzing your results.

Your analysis should include

- a. A brief introduction of raw experimental data generated (copied from the console)
 - b. Your analysis should answer the following questions for **each** URL you specified:
 - i. What effect does TFO have on the timing?
 - ii. How does the RTT value affect these results?
 - iii. Does the particular content available at this URL lend itself to performance enhancements provided by TFO?
 - iv. Were these results surprising in any way?
 - v. Include any relevant graphs from the `output-figures` folder (images do not count toward your two page limit).
 - c. Include a brief summary of your findings and state what conclusions you can draw based on the results of your experiment.
 - d. Based on the reading and your experiment, where do you see TCP Fast Open having the best potential for improvement? What about the worst?
12. Save your report as a PDF file named `Analysis.pdf`. An easy way to accomplish this is to compose your analysis in Google Docs, and download it as a PDF for turn-in.

What to turn in

You will turn in the following files on T-Square:

1. `observations.txt` - The log data you obtained during step 5.
2. `myURLS.pages` - The sites that you ran your own version of the experiment on in steps 7-10.

3. `Analysis.pdf` - A detailed yet concise analysis of your experimental results that you wrote during step 11.

What you can (and cannot) share

Do **not** share the following files with your fellow students, on Piazza, or publicly:

1. `observations.txt`
2. `Analysis.pdf`

You **may**, and are highly encouraged to, share your interesting and or unique `.pages` files as well as the graphs of their performance. Are there sites out in the wild that TFO does not provide a significant increase in performance? Are there sites for which TFO had unexpectedly amazing results? Feel free to discuss with your fellow classmates on Piazza!

Grading

5 pts	Correct Submission	for turning in the correct files, with the correct names, and significant effort has been made towards completing the project per the instructions.
-------	--------------------	---

10 pts	Replicated Research	for correct experimental data produced in steps 5-7, located in <code>observations.txt</code> .
35 pts	Experimental Analysis	for your detailed, yet concise analysis of your results obtained in steps 8-11. Be sure to answer the questions specified and include the appropriate data/graphs in order to receive full credit.

Notes

[1] Based on a [reproducing network research post](#).