DRAFT

DevTools Network Throttling Calibration

Status: Complete
Authors: chenwilliam
Last Updated: 2017-05-08
PUBLIC

Objective

Improve the quality of DevTools network throttling presets to more closely approximate the experience of loading popular websites on mobile network connections.

Goals

- Benchmark how DevTools network throttling **compared to leading network throttling tools (e.g. dummynet, GIN)** for fast 3G connections (e.g. U.S.) and slow 3G connections (e.g. emerging markets)
- Analyze **how much adjustment** DevTools network throttling factors (throughput and RTT) would need in order to match a similar loading experience as these other tools
- Provide a set of **short-term recommendations** on how to adjust DevTools network throttling presets to help our users understand how slowly their websites load on mobile connections

Non-Goals

- **Address specific shortcomings** of DevTools network throttling such as TCP slow start or H2 push
- Assess how to re-architect DevTools network throttling to match kernel-level traffic shaping tools
- Compare DevTools network throttling directly with real-world mobile connections

Background

Concerns have been raised that DevTools network throttling is inaccurate relative to other

network throttling tools.[1] [2] Previously, Paul Irish conducted a detailed feature gap analysis between DevTools and other network throttling tools. Anecdotally, the DevTools team heard that our network throttling conditions were not realistic and were too optimistic compared to other network throttling tools and real-world 3G connections. Prior to this study, there has been sparse benchmarking between DevTools network throttling and other types of throttling.

Technical Challenges

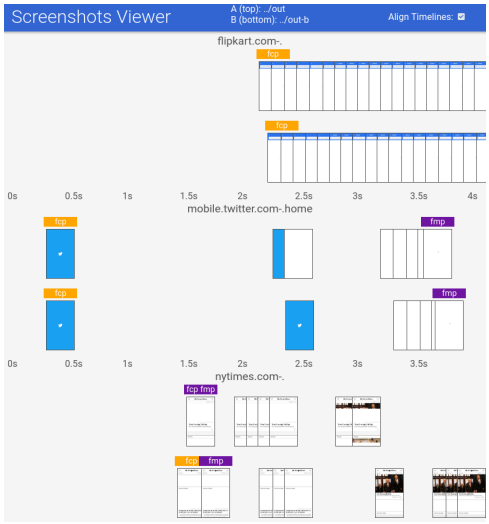
The shortcomings of DevTools network throttling had been known for over a year and without a major rewrite of our Network Throttling mechanism which relies on on shaping HTTP transactions rather than packet-level like many other network throttling tools.

Overview

Methodology

To measure the user experience of using various network throttling tools, I used an internal tool I developed for Lighthouse called A/B Screenshot Viewer, which allows users to compare the loading experience of a set of sites with two different settings (hence the A/B name).

Example screenshot of the A/B Screenshot Viewer (more details):



Across a set of 17 popular websites (see Detailed Analysis section for specifics), I used Lighthouse command line tool to automate loading each site in a new Chrome instance and capture screenshots and measure key performance metrics. As a baseline, I used an alternative network throttling tool and then used comparable settings for DevTools network throttling. I then

[1] https://bugs.chromium.org/p/chromium/issues/detail?id=590880
[2] https://calendar.perfplanet.com/2016/testing-with-realistic-networking-conditions/

iteratively adjusted our network throttling settings (RTT and throughput) to find the right balance that simulates a similar loading experience as the alternative network throttling tool.

Benchmarking Fast 3G Condition

I did a baseline run using Network Link Conditioner using settings to approximate a fast 3G connection.[3] Because Network Link Conditioner uses Dummynet under the hood, which is the same traffic shaping tool that WebPageTest has relied on, it is a well-tested network throttling tool.

The analysis shows that on average DevTools network throttling conditions are currently too optimistic and don't emulate the loading experience that other network throttling tools provide.

Before: 3G Fast (U.S.) Comparison **without adjustments** (status quo)
After: 3G Fast (U.S.) Comparison **with adjustments** (proposed)

Benchmarking Slow / Developing World 3G Condition

I did a baseline run using gin3g, Google Impaired Network's wifi for emulating developing world 3G mobile connections. To account for the significant variability that gin3g has, I did 10 runs for each of the sites and found the median run for each site based on loading performance metrics.

Before: 3G Slow (Emerging Market) Comparison **without adjustments** (status quo)
After: 3G Slow (Emerging Market) Comparison **with adjustments** (proposed)

Proposed Change

Within DevTools, we propose simplifying the set of network throttling presets for mobile connections:

1. **3G - Fast**: provide comparable experience to WebPageTest Mobile 3G - Fast (1.6Mbps download / 750Kbps upload / 150ms RTT)
2. **3G - Slow**: provide comparable experience to gin3g (500Kbps download / 500Kbps upload / 400ms RTT)
3. **Wifi:** 30-60ms RTT and no throughput shaping

We are also proposing changing DevTools UI so that we no longer display specific throughput or RTT numbers. As the below adjustments show, DevTools network throttling tool requires such a large adjustment factor, that it seems less helpful to to surface that level of precision in the UI when internally our tool is not that accurate. Finally, since we're no longer exposing the specifics on how DevTools throttling works, we'll no longer provide the option of custom presets.

---

[3] Settings used: 1638kbps download (approx 1.6mbps) / 750kbps upload / 75ms downlink delay / 75ms uplink delay

3G Fast Internal Adjustment

Proposed adjustment factors:

- RTT will be multiplied by: 3.75
- Throughput will be divided by: 1.1

This means to simulate 1.6Mbps download / 750Kbps upload / 150ms RTT, DevTools internally is setting its throttling to 1.45Mbps download / 682Kbps upload / 562.5ms RTT.

3G Slow Internal Adjustment

Proposed adjustment factors:

- RTT will be multiplied by: 5
- Throughput will be divided by: 1.25

This means to simulate 500Kbps download / 500Kbps upload / 400ms RTT, DevTools internally is setting its throttling to 400Kbps download / 400Kbps upload / 2000ms RTT.[4]

Caveats

There are several important caveats with the recommendation:

- The adjustment factors provide a more realistic user experience, and most notably, provide a much slower loading experience on real-world websites compared to DevTools current network throttling presets, however they do not directly address deficiencies in our network throttling mechanism (e.g. H2 Push)
- This aligns DevTools network throttling closer to mainstream network throttling tools that run on the kernel-level, however there was no analysis done to see if this closes the gap between DevTools network throttling and real-world mobile network conditions.
- Previously, a handful of users had requested for custom network throttling presets (crbug.com/441858) - one potential solution for satisfying these users' requirements would be to add an advanced setting option to enable additional throttling levels (similar to what we do for device mode settings).

Detailed Analysis

Sites analyzed (17 total):

'https://en.wikipedia.org/wiki/Google',
'https://mobile.twitter.com/ChromeDevTools',

[4] I've updated these numbers after re-running the analysis - I made a mistake in setting the throughput settings before which is why the new numbers are lower.

'https://www.instagram.com/stephencurry30',
'https://amazon.com',
'https://nytimes.com',
'https://www.google.com/search?q=flowers',
'https://flipkart.com',
'http://www.espn.com/',
'https://www.washingtonpost.com/pwa/',
'http://www.npr.org/',
'http://www.booking.com/',
'https://youtube.com',
'https://reddit.com',
'https://ebay.com',
'https://stackoverflow.com',
'https://apple.com',
'https://www.nasa.gov/', (could not analyze w/ gin-3G because it loads too slowly)

Code used to run the analysis:
https://github.com/GoogleChrome/lighthouse/pull/2183

Network Throttling Landscape Review

Network Throttling Tools

DevTools Network Throttling[5]

Bandwidth throttling is done by rate limiting the number of packets received or sent in a time window and then queueing excess packets.

RTT throttling is done by calculating the delta between actual request RTT and target RTT and then adding the delta to the request RTT. This is notably different than other network throttling techniques which add the RTT time to the original network request latency.

For example, if we set RTT delay to 150ms in DevTools, and a request has an actual RTT of 100ms (e.g. server-side processing), then the throttled network request RTT is 150ms. In essence, network request latency becomes:

```
user RTT = Math.max(target RTT, actual RTT)
```

[5] DevTools currently has three adjustable dimensions in the GUI: 1) download throughput and 2) upload throughput and 3) RTT (although labeled as latency in the UI).

Network Link Conditioner

Mac OS provides a convenient GUI for controlling RTT delay and throughput by using pf (packet filter) as the firewall and dummynet for traffic shaping, which is described below.[6]

Dummynet

Dummynet is a popular network emulation tool that has been ported to multiple OSes:[7]
It simulates/enforces queue and bandwidth limitations, delays, packet losses, and multipath effects…. and works by intercepting selected traffic on its way through the network stack… and passing packets to objects called **pipes** which implement a set of queues, a scheduler, and a link, all with configurable features (bandwidth, delay, loss rate, queue size, scheduling policy...). Traffic selection is done using the **ipfw** firewall, which is the main user interface for dummynet. ipfw lets you select precisely the traffic and direction you want to work on, making configuration and use incredibly simple.

RTT throttling is done by adding the target RTT to the actual RTT:

```
user RTT = target RTT + actual RTT
```

ipfw and dummynet are frequently used together and in earlier versions of Mac OS, Network Link Conditioner used ipfw and dummynet, but then switched to using pf and dummynet.[8]

"WebPageTest has historically relied very heavily on dummynet for all of its traffic-shaping and configures one pipe for inbound traffic and one pipe for outbound traffic." [9]

NetEm

Provides similar capabilities to Dummynet but it has first-class support on Linux.[10]

It has a complex and powerful interface. I was only able to properly configure the RTT throttling but wasn't able to properly configure the bandwidth throttling.[11] [12]

Augmented Traffic Control

Flexible tool created by Facebook for managing NetEm settings with a user-friendly UI. Designed to run on a gateway.

---

[6] https://spin.atomicobject.com/2016/01/05/simulating-poor-network-connectivity-mac-osx/
[7] From Professor Luigi Rizzo's webpage: http://info.iet.unipi.it/~luigi/dummynet/#2a39
[8] https://apple.stackexchange.com/questions/24066/how-to-simulate-slow-internet-connections-on-the-mac/46378#46378
[9] https://calendar.perfplanet.com/2016/testing-with-realistic-networking-conditions/
[10] Internal discussion on NetEm and Dummynet
[11] http://mark.koli.ch/slowdown-throttle-bandwidth-linux-network-interface
[12] https://www.iplocation.net/traffic-control

Trickle

Designed as a simple to use userland tool for network throttling. However, I was unable to have it properly throttle Chrome - likely due to their limitation:
“Trickle can only limit traffic of programs that do not fork, so shaping a FTP server's traffic will not work that way.”[13]

Common Throttling Settings

Today, within Google there are several disparate recommendations on the different qualities of 3G connections.

DevTools Status Quo

- **Good 3G:** 1.5mbps download, 750kbps upload, 40ms RTT.
- **Regular 3G:** 750kbps download, 250kbps upload, 100ms RTT.

Lighthouse Status Quo

- **Average mobile:** 1.6mbps, 750kbps, 150ms RTT.

Google Impaired Network

- **3G (Emerging Market):** 400-600kbps throughput, 400ms RTT w/ 100ms variability, 2% packet loss, and burst packet loss

Note: Based on go/gin-mobile

WebPagetest

- **Fast 3G:** 1.6Mbps download, 768kbps upload, with 150ms of latency.
- **3G:** 1.6Mbps download, 768kbps upload, with 300ms of latency.
- **Emerging Market 3G:** 400kbps download, 400kbps upload, with 400ms of latency.

[13] https://wiki.archlinux.org/index.php/Trickle