How speedy is SPDY?

Xiao (Sophia) Wang, Aruna Balasubramanian, Arvind Krishnamurthy, David Wetherall

University of Washington
HTTP/1.1: The standard to load Web pages

HTTP/1.1 becomes slow for rich, modern pages

Google developed **SPDY** to make the Web faster

- Starting to be deployed
- Basis for HTTP/2.0 now being standardized
HTTP/1.1: The standard
to (the)
load (Web)
pages
for rich, modern pages

Google developed **SPDY** to make the Web faster
- Starting to be deployed
- Basis for HTTP/2.0 now being standardized

**How much better is SPDY than HTTP?**
HTTP/1.1 problems
HTTP/1.1 problems

- Opens too many TCP connections
- Opens too many TCP connections
- Initiates object transfers strictly by the client

HTTP/1.1 problems
HTTP/1.1 problems

- Opens too many TCP connections
- Initiates object transfers strictly by the client
- Compresses only HTTP payloads, not headers
HTTP/1.1 problems

- Opens too many TCP connections
- Initiates object transfers strictly by the client
- Compresses only HTTP payloads, not headers

SPDY is proposed to address these issues
SPDY

- Opens too many TCP connections
- Multiplexes sliced frames into a single TCP connection
SPDY

- Opens too many TCP connections
- Multiplexes sliced frames into a single TCP connection
- Prioritizes Web objects
SPDY

- Initiates object transfers strictly by the client
- Allows servers to initiate Web object transfers
SPDY

- Compresses only HTTP payloads, not headers
- Compresses both HTTP payloads and headers
How well does SPDY perform?

Google

SPDY helps 27% to 60%

Microsoft

SPDY sometimes helps and sometimes hurts. Overall, SPDY helps < 10%.
How well does SPDY perform?

SPDY sometimes helps and sometimes hurts.

Measurement results conflict

Google
SPDY helps 27% to 60%

Microsoft
Goals

- A systematic study of SPDY that
  - Extensively sweeps the parameter space
  - Links SPDY performance to underlying factors
  - Identifies the dominant factors

SPDY v.s. HTTP/1.1
Many factors **external** to SPDY affect SPDY

**Approach**

- Isolate factors, sweep the parameter space
  - Network parameters
  - TCP settings
  - Web page effects
  - RTT
  - Bandwidth
  - Loss rate
  - TCP initial congestion window
  - Synthetic objects
  - Real objects
  - Real pages
Challenge

Page load time has high variance

Variance: 0.5 second
Difference: 0.02 second

Approach

Control source of variability by
- Experimenting in a controlled network
- Using our emulator instead of browsers
Dependencies between network and browser computation affect page loads
**Dependencies** between network and browser computation affect page loads

**Challenge**

Preserve dependencies.

**Approach**

Browser computation
Outline

• Understanding SPDY’s performance with
  – Synthetic objects
  – Real objects
  – Real pages
Outline

• Understanding SPDY’s performance with
  – Synthetic objects
  – Real objects
  – Real pages
Extensively sweep parameter space

<table>
<thead>
<tr>
<th>Factors</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT</td>
<td>20ms, 100ms, 200ms</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>1Mbps, 10Mbps</td>
</tr>
<tr>
<td>Loss rate</td>
<td>0, .5%, 1%, 2%</td>
</tr>
<tr>
<td>TCP IW</td>
<td>3, 10, 21, 32</td>
</tr>
<tr>
<td>Web obj. size</td>
<td>100B, 1K, 10K, 100K, 1M</td>
</tr>
<tr>
<td># of objects</td>
<td>2, 8, 16, 32, 64, 128, 512</td>
</tr>
</tbody>
</table>

Make HTTP requests
Link SPDY performance to factors

→ Decision tree analysis

Six factors, Thousands of data points
SPDY helps on small objects

Explanation
Unlike in HTTP, a TCP segment can carry multiple Web objects in SPDY.
SPDY helps on large objects, low loss

Explanation

In HTTP, Multiple connections compete with each other

→ More retransmissions
In a single connection, SPDY reduces cwnd more aggressively under loss.
SPDY hurts on large objects, high loss

Most performance impact of SPDY comes from a single TCP connection.

In a single connection, SPDY reduces cwnd more aggressively under loss.

Most performance impact of SPDY comes from a single TCP connection.
Identify dominant factors

# obj  BW

RTT  obj size

loss  IW

Importance?
Identify dominant factors

RTT: 200ms
BW: 10Mbps
Loss: 0
IW: 3
obj size: 10KB
# obj: 8

# obj shows a trend
TCP IW doesn't show a trend
Does SPDY help stragglers?

In our experiments, we find that SPDY helps little for stragglers.
Does SPDY help stragglers?

HTTP/1.1

SPDY

- In our experiments, we find that SPDY helps little for stragglers.

This hypothesis is weak since it only argues with cwnd
Outline

• Understanding SPDY’s performance with
  – Synthetic objects
  – Real objects
  – Real pages
## Synthetic objects \(\rightarrow\) Real objects

<table>
<thead>
<tr>
<th>Factors</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network parameters</strong></td>
<td></td>
</tr>
<tr>
<td>RTT</td>
<td>20ms, 100ms, 200ms</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>1Mbps, 10Mbps</td>
</tr>
<tr>
<td>Loss rate</td>
<td>0, .5%, 1%, 2%</td>
</tr>
<tr>
<td><strong>TCP settings</strong></td>
<td></td>
</tr>
<tr>
<td>TCP IW</td>
<td>3, 10, 21, 32</td>
</tr>
<tr>
<td><strong>Web objects</strong></td>
<td></td>
</tr>
<tr>
<td>Web obj. size</td>
<td>Top 200 Alexa pages</td>
</tr>
<tr>
<td># of objects</td>
<td></td>
</tr>
</tbody>
</table>

Make HTTP requests
SPDY helps 60% in the median case because it largely reduces retransmissions.
Outline

• Understanding SPDY’s performance with
  – Synthetic objects
  – Real objects
  – Real pages

Browser effects
Assumption that objects are fetched at the same time does not hold.
Upload captures browser effects

- **Recorder:** capture the dependency graph
- **Replayer:** make network requests while simulating the computation portions
Epload captures browser effects

- Recorder: capture the dependency graph
- Replayer: make network requests while simulating the computation portions

Epload makes experiments reproducible
# Real objects $\Rightarrow$ Real pages

<table>
<thead>
<tr>
<th>Factors</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT</td>
<td>20ms, 100ms, 200ms</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>1Mbps, 10Mbps</td>
</tr>
<tr>
<td>Loss rate</td>
<td>0, .5%, 1%, 2%</td>
</tr>
<tr>
<td>TCP IW</td>
<td>3, 10, 21, 32</td>
</tr>
<tr>
<td>Web obj. size</td>
<td>Top 200 Alexa pages</td>
</tr>
<tr>
<td># of objects</td>
<td></td>
</tr>
</tbody>
</table>

Emulate page loads with **Epload**
SPDY helps marginally because

- Computation and dependencies increase PLT of both SPDY and HTTP
- Throttled object fetches result in fewer retransmissions in HTTP
Dependencies and computation in real page loads reduce the impact of SPDY.
Other experiments in the paper

• Using Server Push
  – Experimented with new policy
  – Saves **10% - 30%** latency like mod_spdy but pushes **80%** less data

• With SSL/TLS
  – Tested SPDY and HTTP over SSL/TLS
  – Larger latencies but same conclusions
Conclusions

• We experimented with SPDY page loads over a large parameter space
• Most performance impact of SPDY over HTTP comes from its single TCP connection
• Browser computation and dependencies in real pages reduce the impact of SPDY
• To improve further, we need to restructure the page load process
Data
We release the data obtained by sweeping the parameter space and welcome further analysis on this data. Here is our setting.

Download all data (211KB) (downloaded 3 times)

We tabulate our data below and allow sort by column. We provide plots that show trends in one parameter by fixing the other parameters. Guide on how to plot trends. To download the network trace of a data point, just click on the link to the PLT (page load time) of that data point.

<table>
<thead>
<tr>
<th>RTT/2</th>
<th>Bandwidth</th>
<th>Loss rate</th>
<th>IW</th>
<th>File size</th>
<th># objects</th>
<th>PLT http (s)</th>
<th>PLT spdy (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10ms</td>
<td>10Mbits</td>
<td>0</td>
<td>3</td>
<td>100B</td>
<td>2</td>
<td>0.04 0.02 0.06 0.05 0.02 0.03</td>
<td>0.02 0.02 0.03 0.02 0.02 0.02</td>
</tr>
<tr>
<td>10ms</td>
<td>10Mbits</td>
<td>0.005</td>
<td>3</td>
<td>1K</td>
<td>2</td>
<td>0.03 0.02 0.03 0.03 0.03 0.03</td>
<td>0.02 0.02 0.02 0.03 0.02 0.02</td>
</tr>
</tbody>
</table>