Mahimahi:
Accurate Record-and-Replay for HTTP

Ravi Netravali*, Anirudh Sivaraman*, Somak Das*, Ameesh Goyal*, Keith Winstein†, James Mickens‡, and Hari Balakrishnan*

*MIT CSAIL, †Stanford University, ‡Harvard University
HTTP is Everywhere

Initially developed for web browsing (browsers)

Instant messaging

Video streaming

Mobile applications
HTTP Performance Matters

- **HTTP Client Apps** updated frequently

  Browsers built every 6 weeks
  Mobile apps updated monthly

- **New Multiplexing Protocols**: HTTP/2.0, SPDY, QUIC

- **Modifications to HTTP**: Pipelining, Persistent Connections

- **New Congestion Control**: TCP Fast Open, IW10, PRECONNECT
Problem

Difficult to evaluate changes in controlled settings

<table>
<thead>
<tr>
<th>Current Approaches</th>
<th>Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test with Live Users</td>
<td>• Not scalable</td>
</tr>
<tr>
<td>Record-and-Replay Tools</td>
<td>• Difficult in early development</td>
</tr>
<tr>
<td>• Google’s web-page-replay</td>
<td>• Hard to understand why</td>
</tr>
<tr>
<td>• Telerik’s Fiddler</td>
<td>• Not accurate</td>
</tr>
<tr>
<td></td>
<td>• No isolation</td>
</tr>
</tbody>
</table>

![Diagram](http://example.com/diagram.png)
Mahimahi: Accurate Record-and-Replay for HTTP

3 Goals

- **Accuracy** → Multi-server emulation
- **Isolation** → Use private network namespaces
- **Composability** → Structure components as UNIX shells
Rest of This Talk

• Mahimahi
  – Design
  – Goals

• Case Studies
  – Evaluating Web multiplexing protocols
    • SPDY 1.15-4.93x worse than Optimal

  – Improving Web performance with Cumulus
    • Cumulus outperforms SPDY by 1.07-3.60x
Design

5 main tools

Record and Replay

- RecordShell
- ReplayShell

Network Emulation

- DelayShell
- LinkShell
- LossShell

Ferry can implement arbitrary policies to handle packets:
- Record traffic
- Serve recorded content
- Delay packets

private network namespace
ReplayShell

Replay recorded traffic locally

- Mirror server side of app locally
  - Preserve recorded server distribution
  - Transparent
DelayShell/LinkShell

Emulate fixed propagation delay or fixed/variable-rate links

- **DelayShell (min RTT)**
  - Release packets after specified delay
- **LinkShell (link rates)**
  - Release packets based on traces
  - Live graphing of network usage/queuing delay
Accuracy

ReplayShell correctly emulates multi-server apps

More accurate measurements

ReplayShell, multi-server: 12.4%
web-page-replay: 36.7%
Isolation

Each shell creates a new network namespace

Multiple instances in parallel

Browser 1
DelayShell 10 ms

Browser 2
DelayShell 50 ms

Browser 3
DelayShell 10 ms

User Computer

RTT = X milliseconds

Low overhead

DelayShell: 0.33%
LinkShell: 0.31%

Cumulative Proportion

Relative Percent Error

DelayShell 0 ms
LinkShell 1000 Mbits/s
Composability

Each tool is structured as a UNIX shell

– Unmodified apps can run within each shell

– Shells can be arbitrarily nested within one another
Case Studies

• Evaluating web multiplexing protocols
  – SPDY
    • Multiplexed streams
    • Header compression
    • Server push
  – QUIC
    • Multiplexing without head of line blocking
    • 0-1 round trips to establish secure connection (uses UDP)

• Reducing page load times with Cumulus
Multiplexing Protocol Setup

- **HTTP →** default Apache servers
- **SPDY →** Apache servers + mod_spdy module
- **QUIC →** replace Apache with QUIC test servers
- **Optimal**

\[
\text{minimumRTT} + \frac{\text{siteSize}}{\text{linkRate}} + \text{browserTime}
\]

- Time between first client request and first byte of response
- Time to transfer all bytes for web page over link with rate ‘linkRate’
- Time for browser to process all responses and render page
Evaluating Multiplexing Protocols

Multiplexing protocols are suboptimal
– Suboptimality increases as RTT increases
Understanding Suboptimality

HTTP requests become serialized → link not fully utilized
Cumulus

Reduces the effective RTT

Implemented Using RecordShell

Transparent HTTP caching proxy running on user’s machine

Local Proxy

HTTP Request

Bulk Response

Remote Proxy

Headless browser running on well-provisioned server

Web Servers

Client Browser

Long-delay Link

Short-delay Link
Evaluating Cumulus

Performance stays close to Optimal as RTTs increase

- Cumulus incurs only one RTT on the long-delay link (reduces effective RTT)
Other Uses

• Measuring speed index

• Evaluating mobile apps over wireless networks
  – Shuo Deng

• Emulating mobile multi-homing (MPTCP)
  – Shuo Deng

• Replaying streamed video
  – Devasia Manuel, Victor Vasiliev, Saunders Hayes
Conclusion

- Mahimahi records HTTP traffic and replays it over emulated networks
  - **Accuracy** - multi-server emulation
  - **Isolation** - private network namespaces
  - **Composability** - UNIX shells

- Case studies using Mahimahi
  - Web multiplexing protocols are suboptimal
  - Cumulus reduces effective RTT to lower page load times

http://mahimahi.mit.edu    mahimahi@mit.edu