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



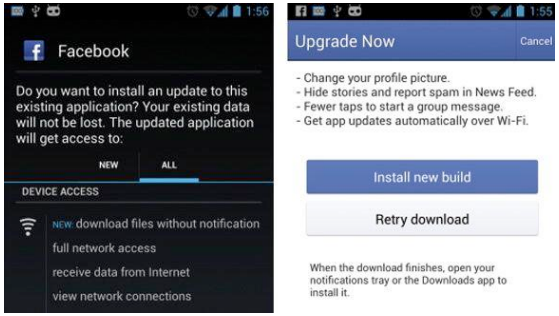
Mobile applications

HTTP Performance Matters

- **HTTP Client Apps** updated frequently



Browsers built every 6 weeks



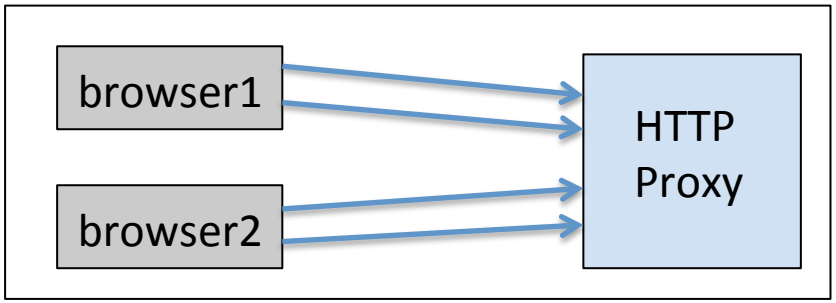
Mobile apps updated monthly

- **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

Current Approaches	Shortcomings
Test with Live Users	• Not scalable • Difficult in early development • Hard to understand *why*
Record-and-Replay Tools • Google's web-page-replay • Telerik's Fiddler	• Not accurate • No isolation



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

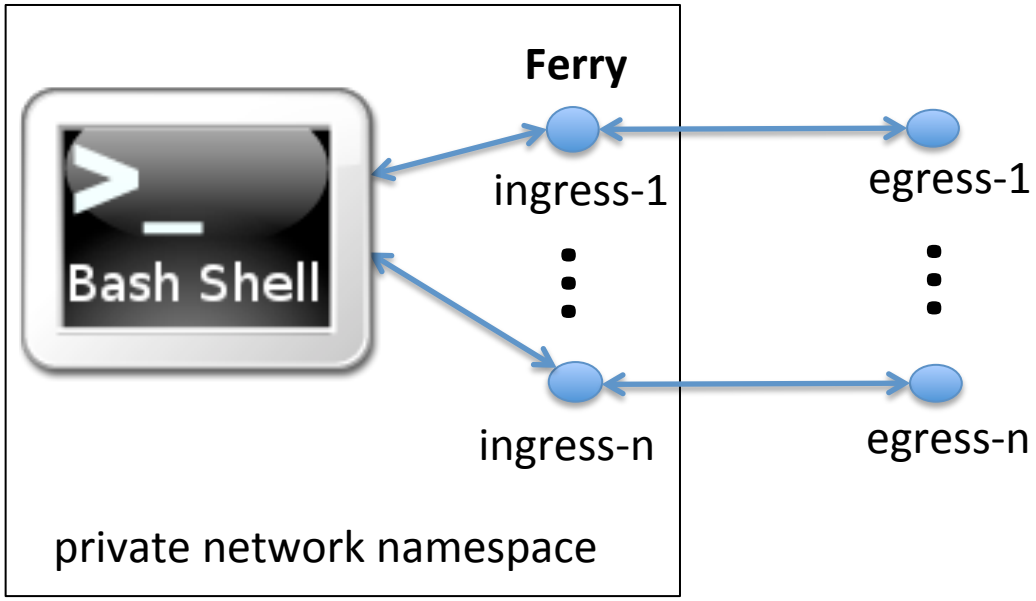
- Mahimahi
  - Design
  - Goals
- Case Studies
  - Evaluating Web multiplexing protocols
    - SPDY 1.15-4.93x worse than Optimal
  - Improving Web performance with Cumulus
    - Cumulus outperforms SPDY by 1.07-3.60x

Design

5 main tools

Record and Replay
- **RecordShell**
- **ReplayShell**

Network Emulation
- **DelayShell**
- **LinkShell**
- **LossShell**

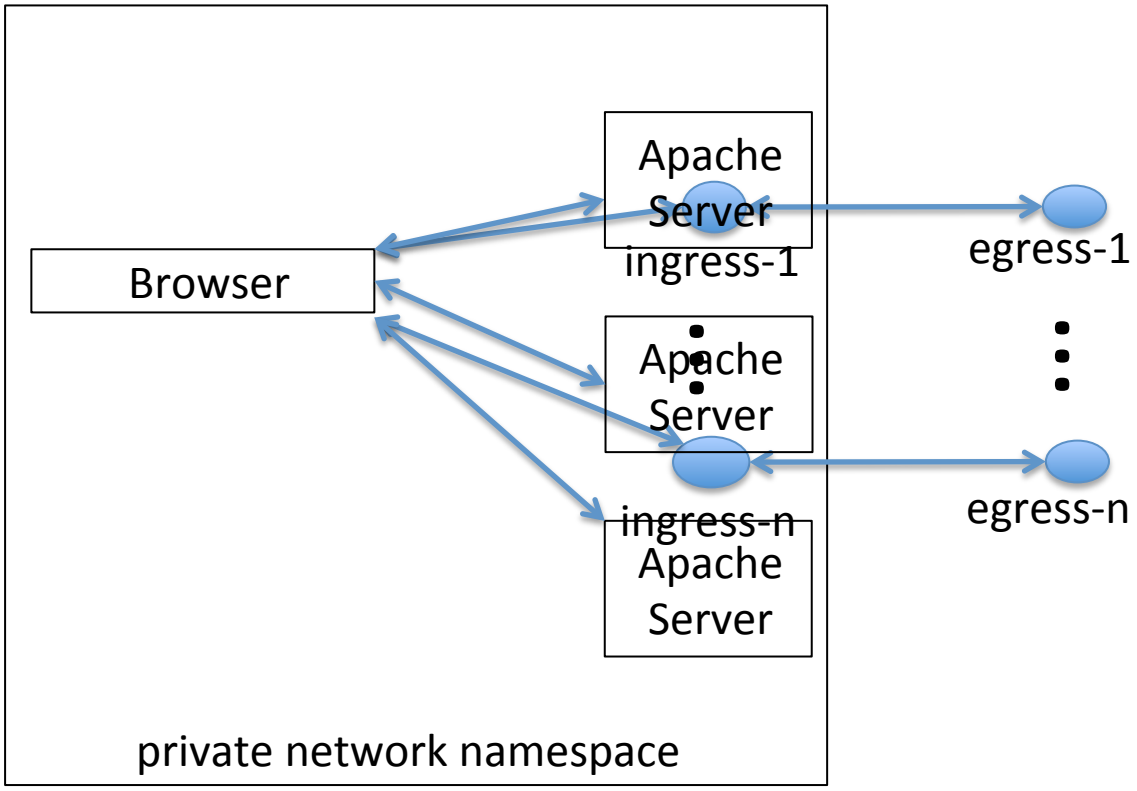


Ferry can implement arbitrary policies to handle packets:
- Record traffic
- Serve recorded content
- Delay packets

ReplayShell

Replay recorded traffic locally

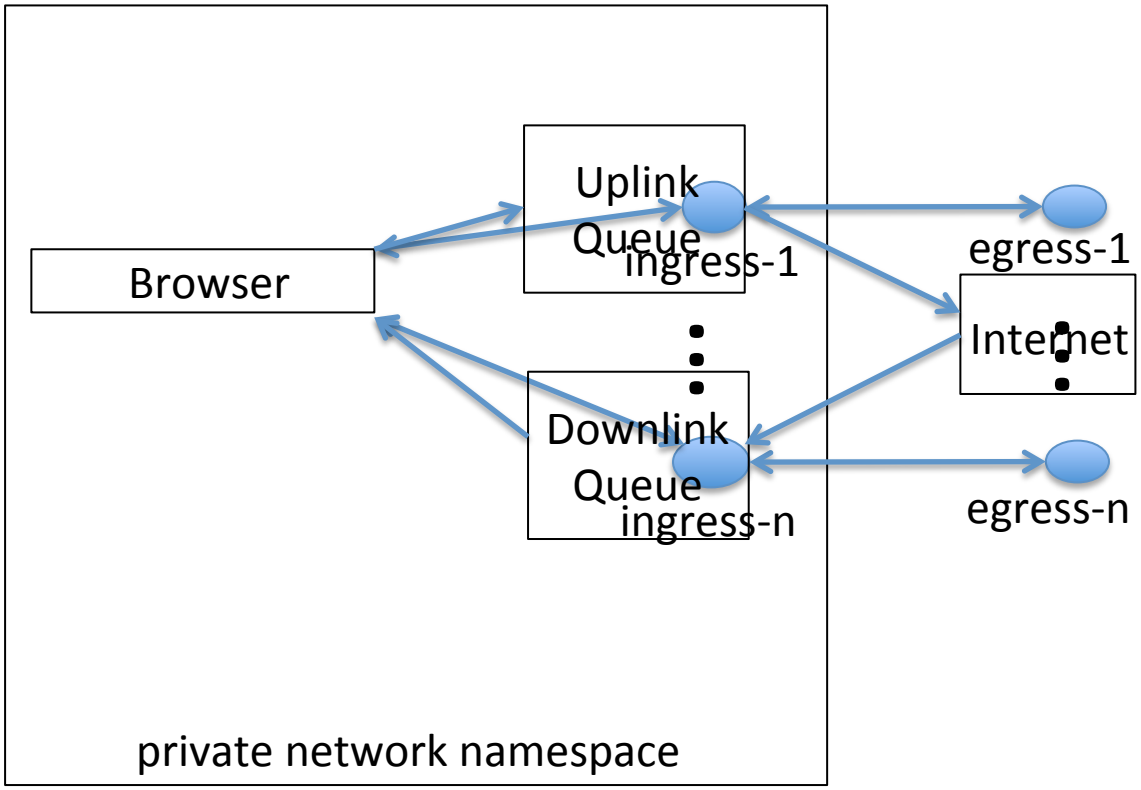
- Mirror server side of app locally
  - Preserve recorded server distribution
  - Transparent



DelayShell/LinkShell

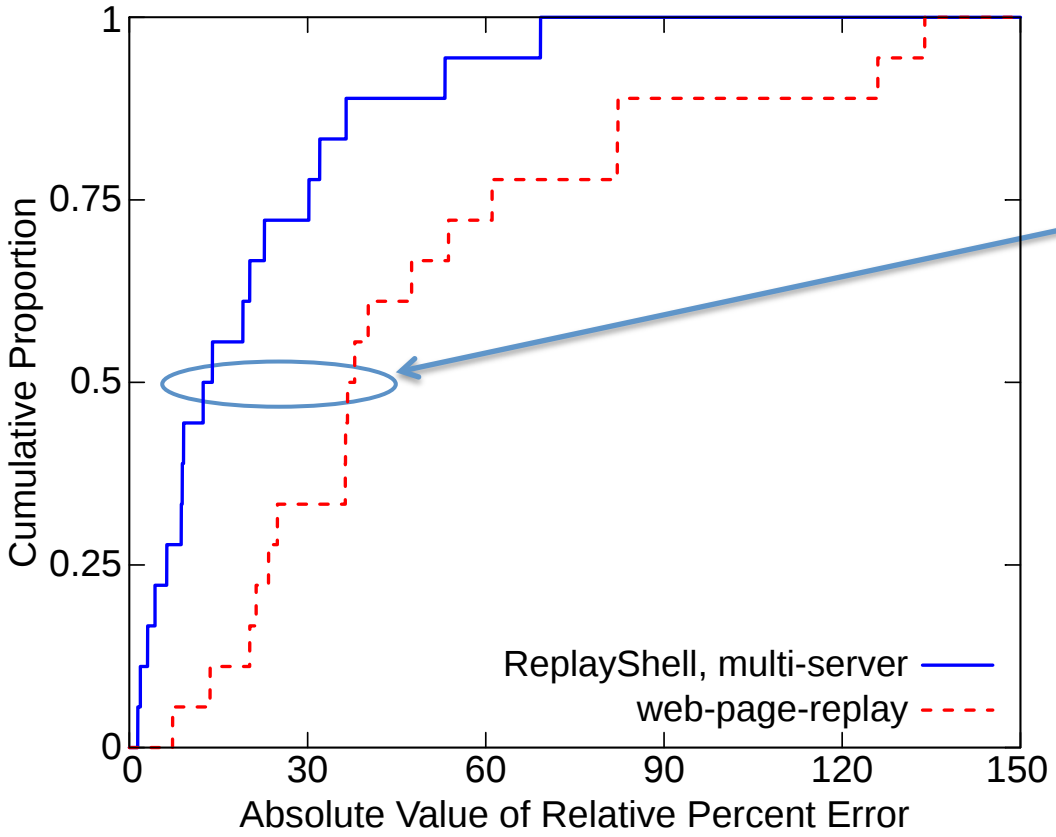
Emulate fixed propagation delay or fixed/variable-rate links

- 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



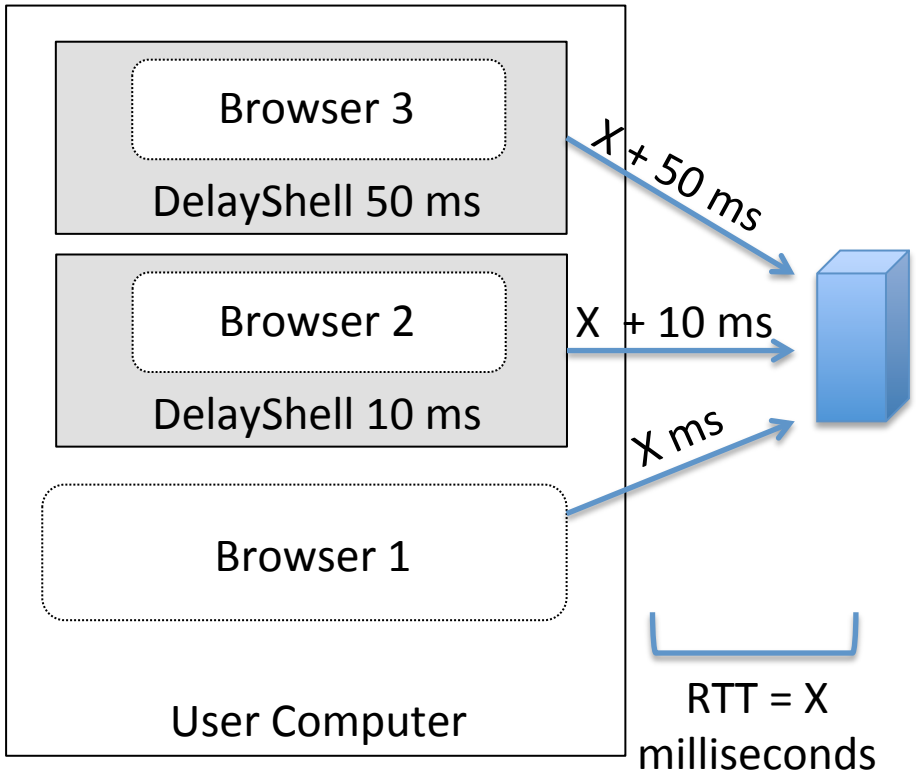
ReplayShell, multi-server: 12.4%

web-page-replay: 36.7%

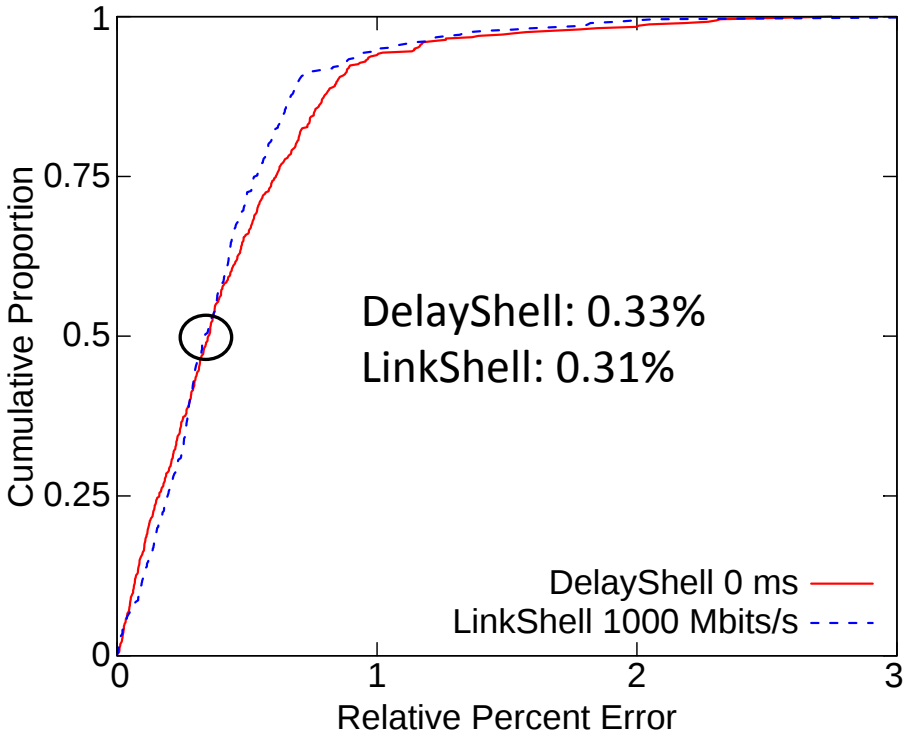
Isolation

Each shell creates a new network namespace

Multiple instances in parallel



Low overhead



Composability

Each tool is structured as a UNIX shell

- Unmodified apps can run within each shell
- Shells can be arbitrarily nested within one another

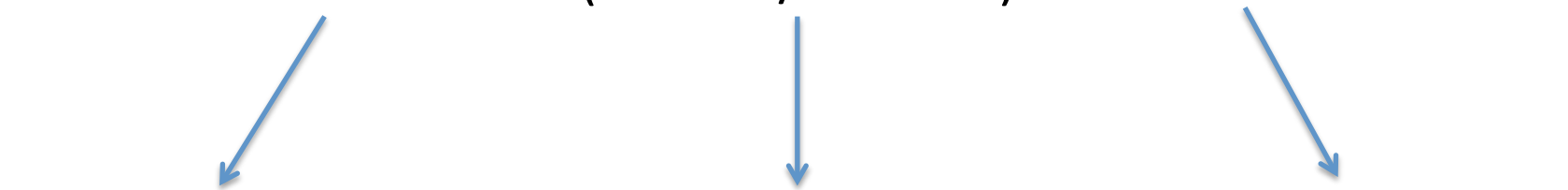
Case Studies

- 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

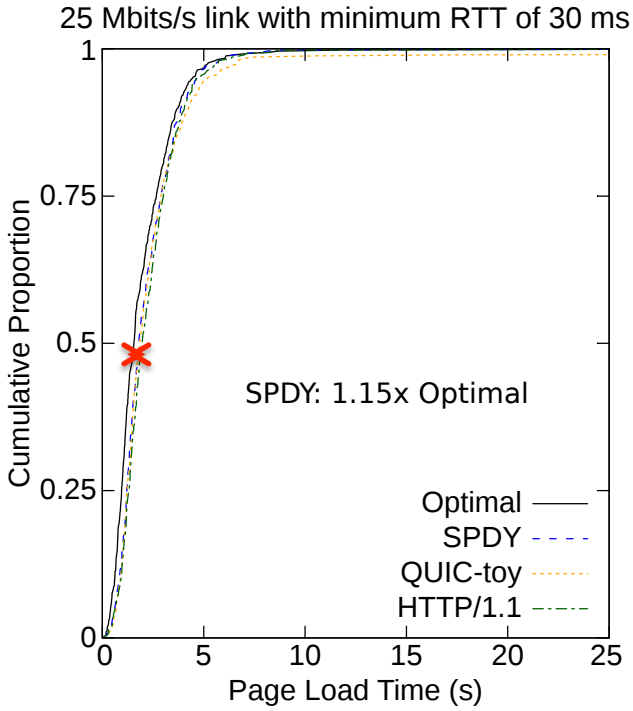
Multiplexing Protocol Setup

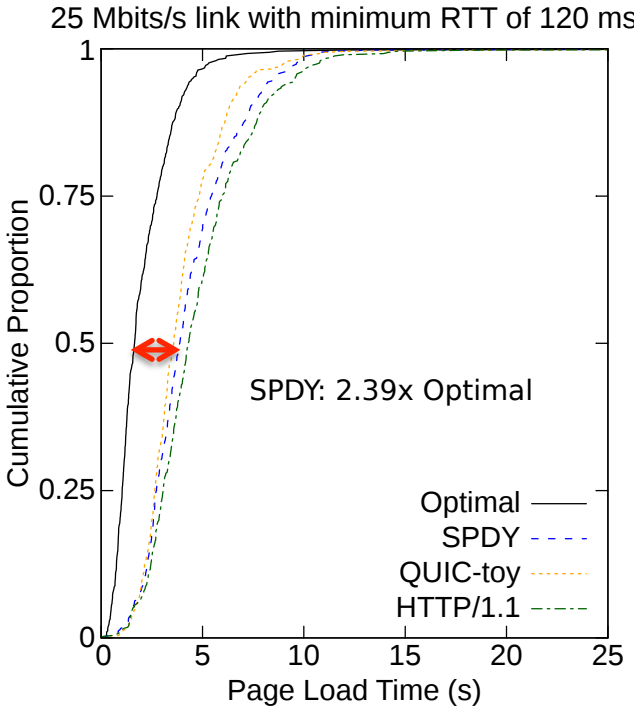
- HTTP → default Apache servers
- SPDY → Apache servers + mod_spdy module
- QUIC → replace Apache with QUIC test servers
- Optimal

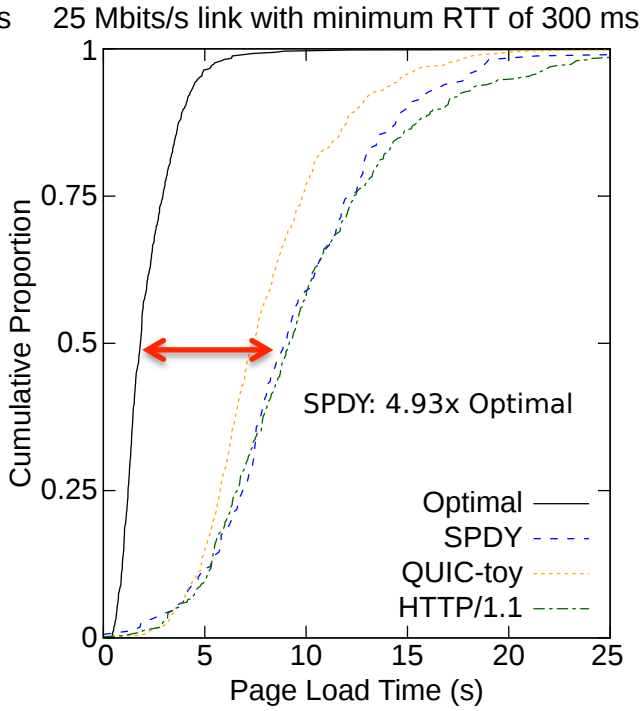
minimumRTT + (siteSize/linkRate) + browserTime



Evaluating Multiplexing Protocols



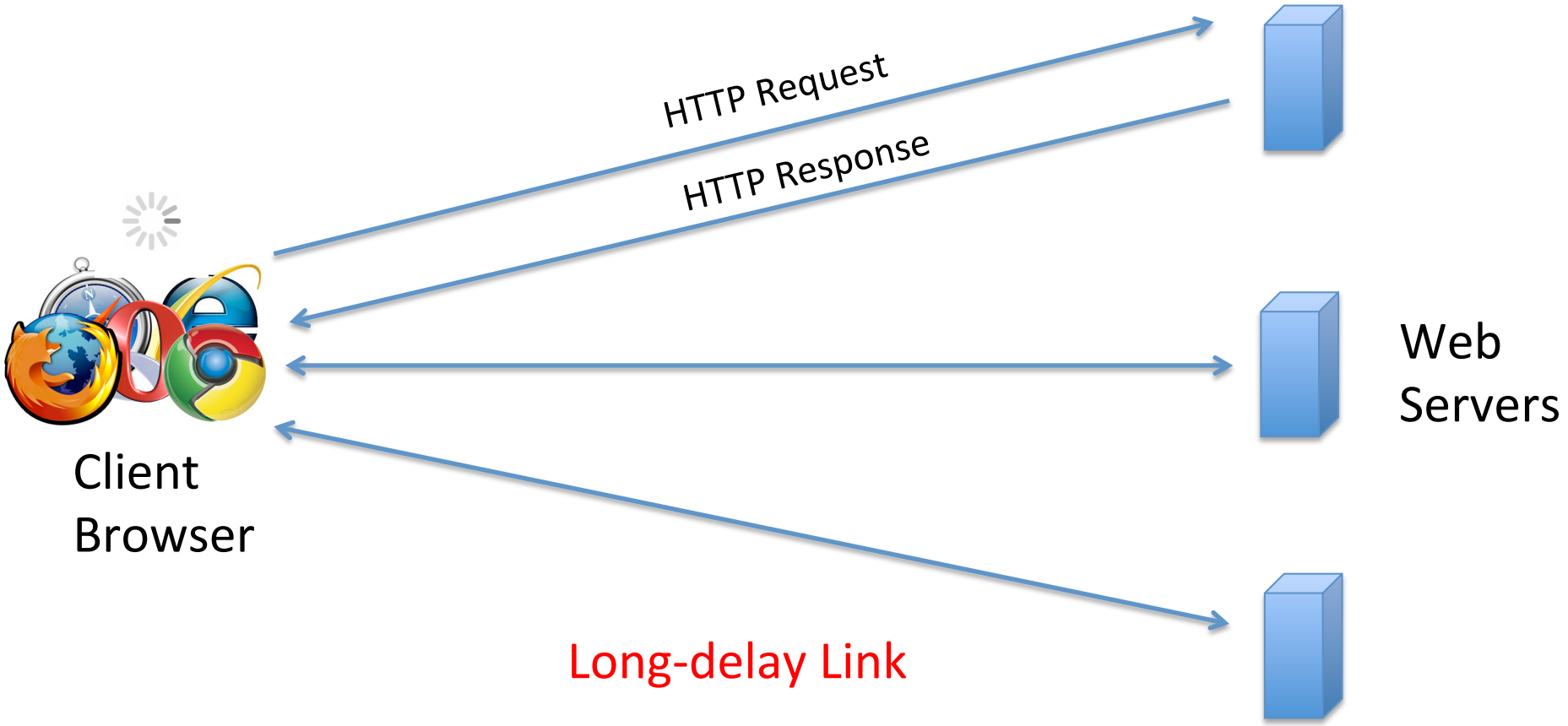




Multiplexing protocols are suboptimal

– Suboptimality increases as RTT increases

Understanding Suboptimality

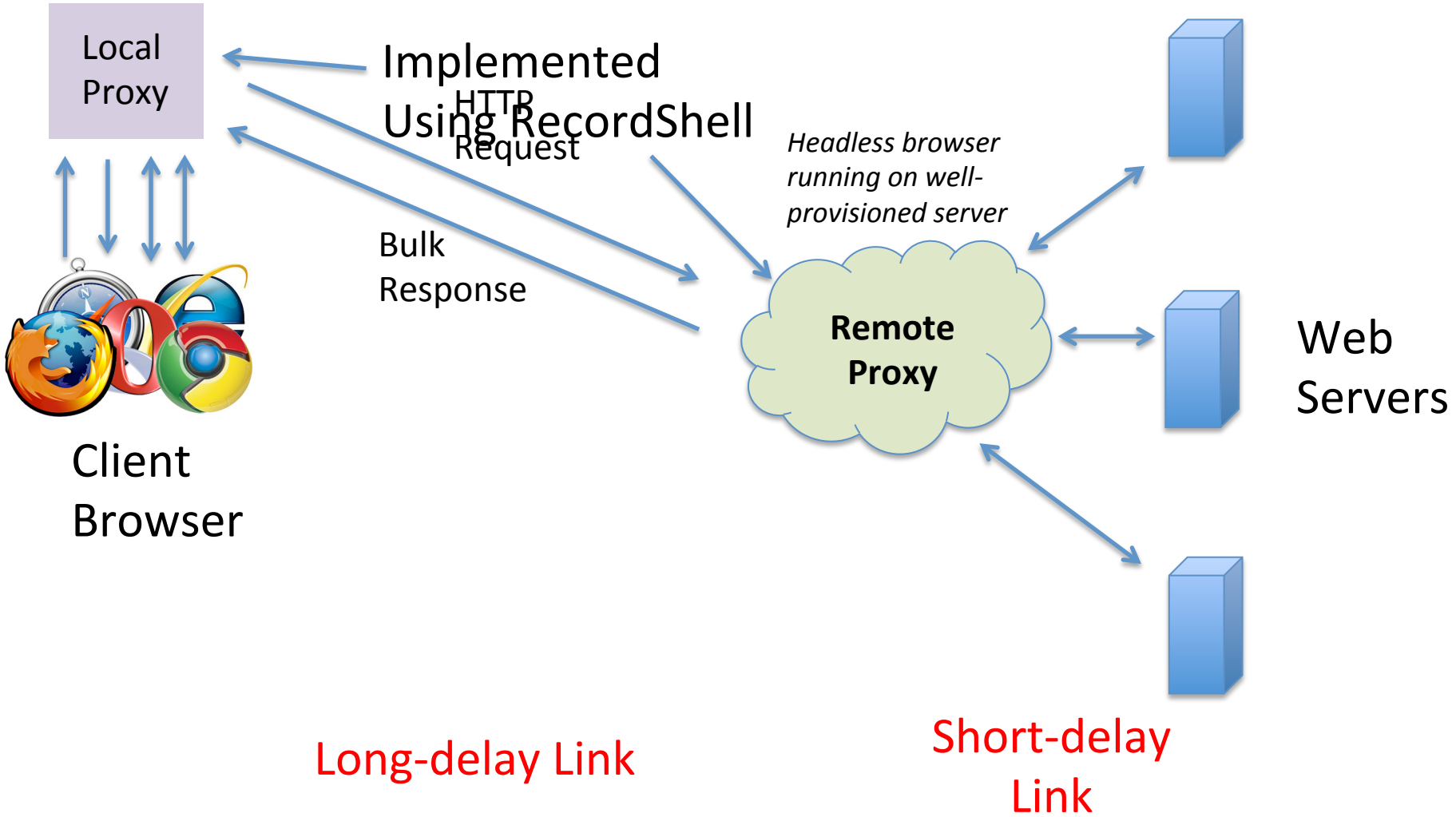


HTTP requests become serialized → link not fully utilized

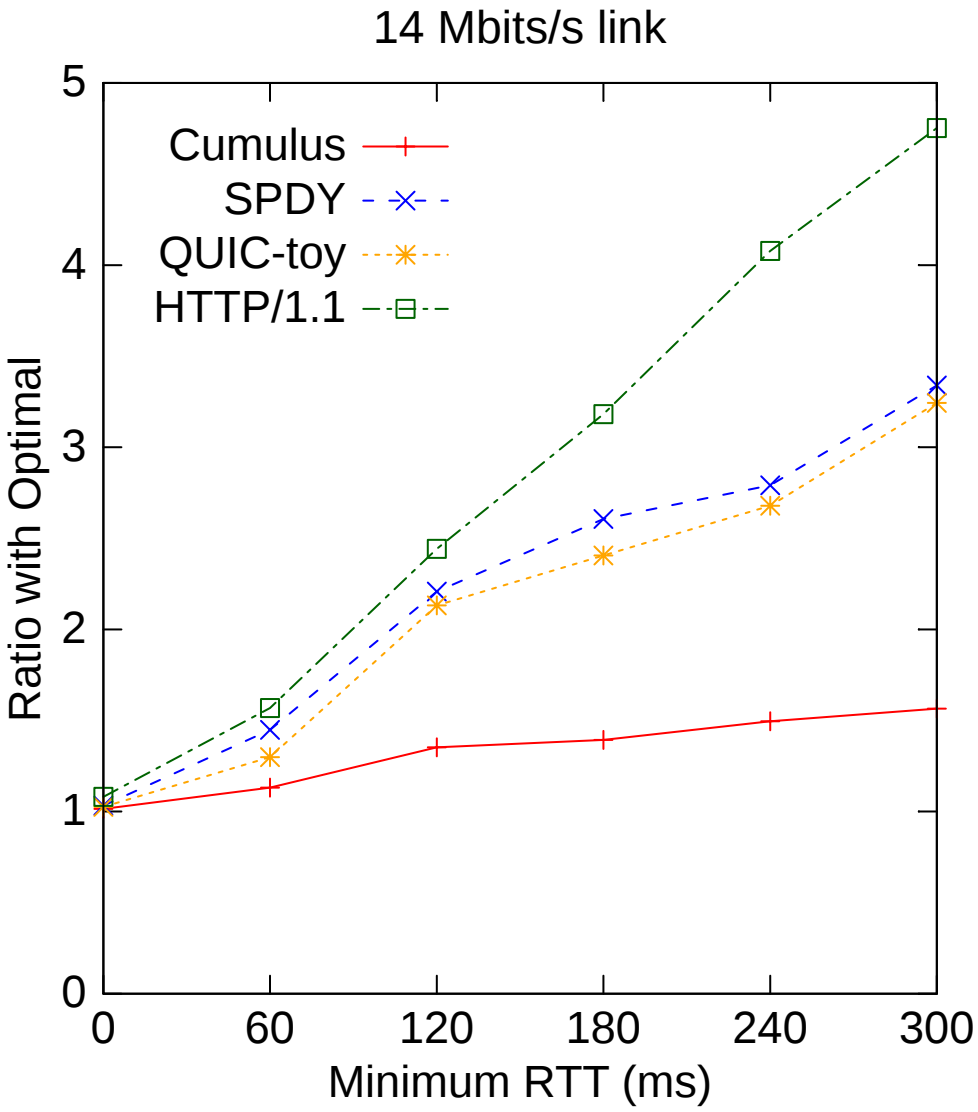
Cumulus

Transparent HTTP caching proxy running on user's machine

Reduces the effective RTT



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**