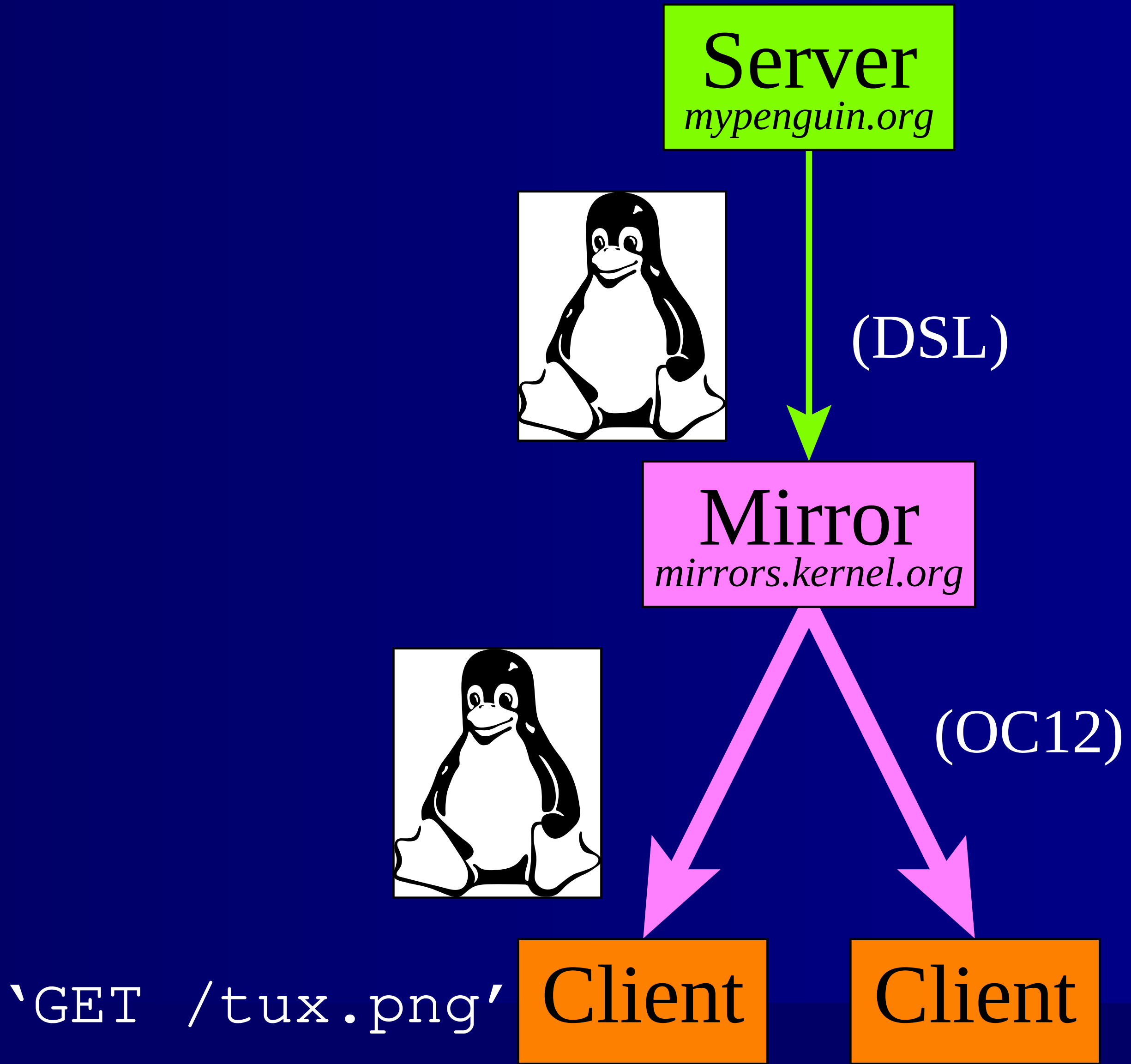
SSL Splitting

Christopher Lesniewski-Laas and M. Frans Kaashoek

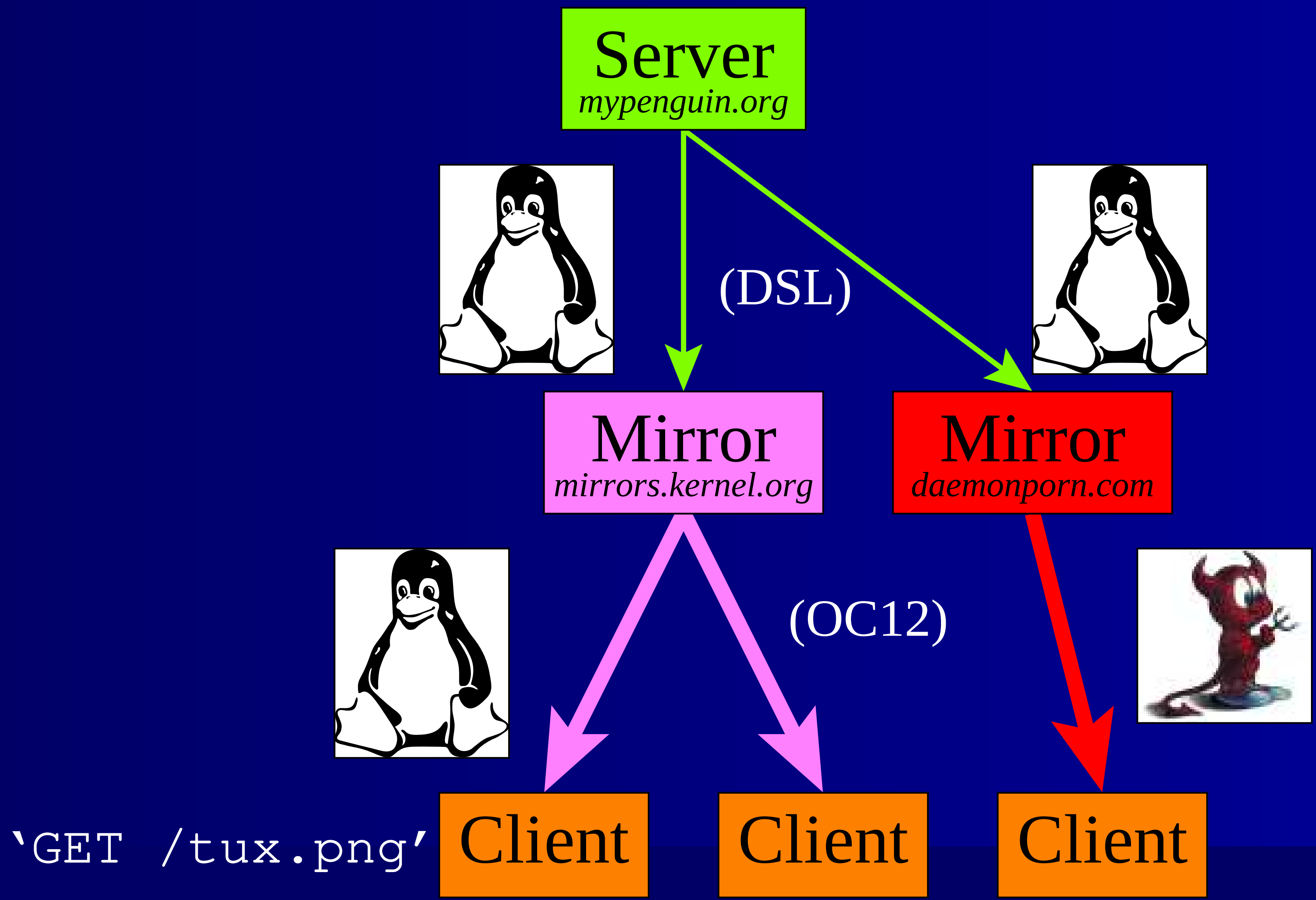
{ctl,kaashoek}@mit.edu

MIT LCS

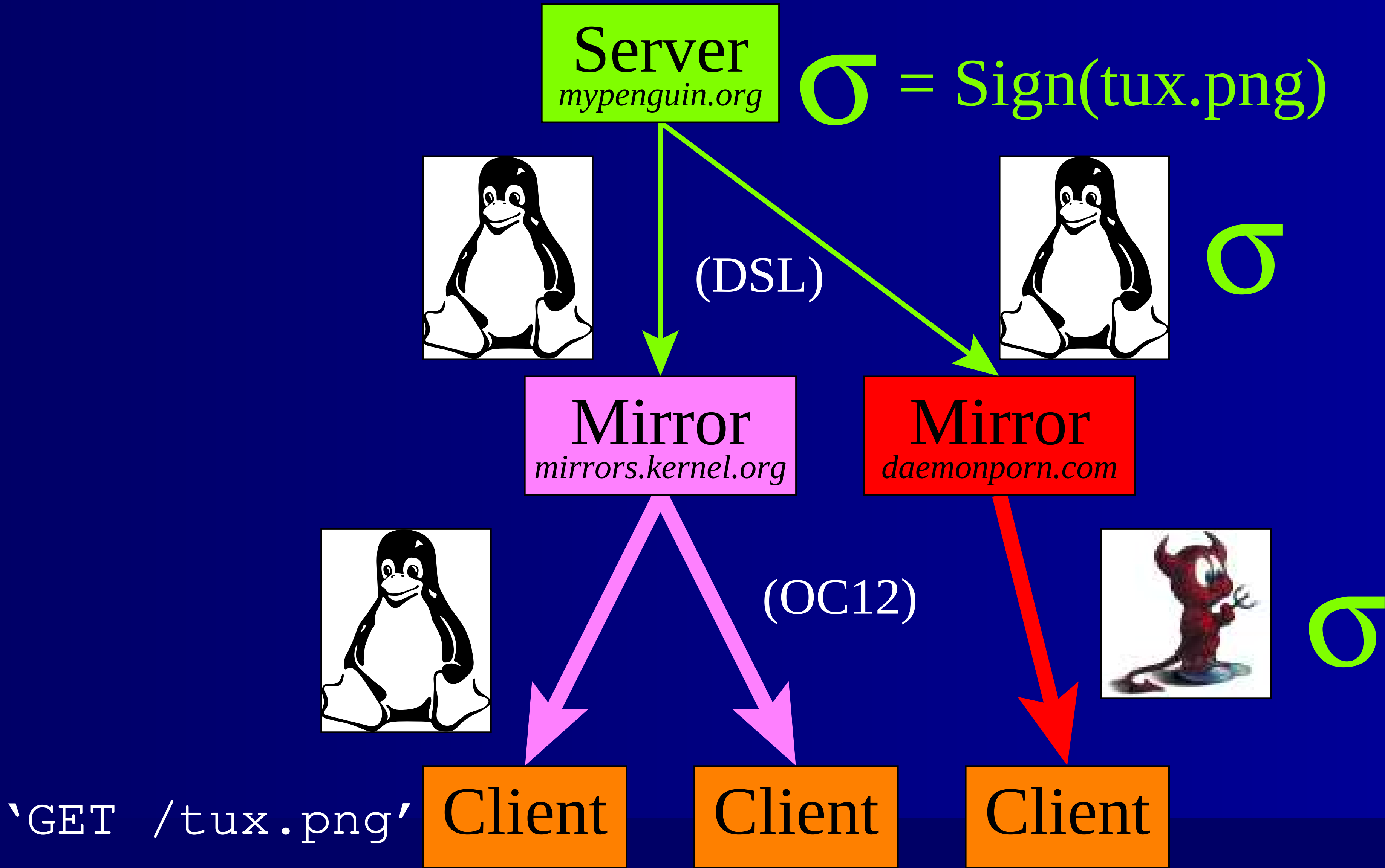
Bandwidth Offloading



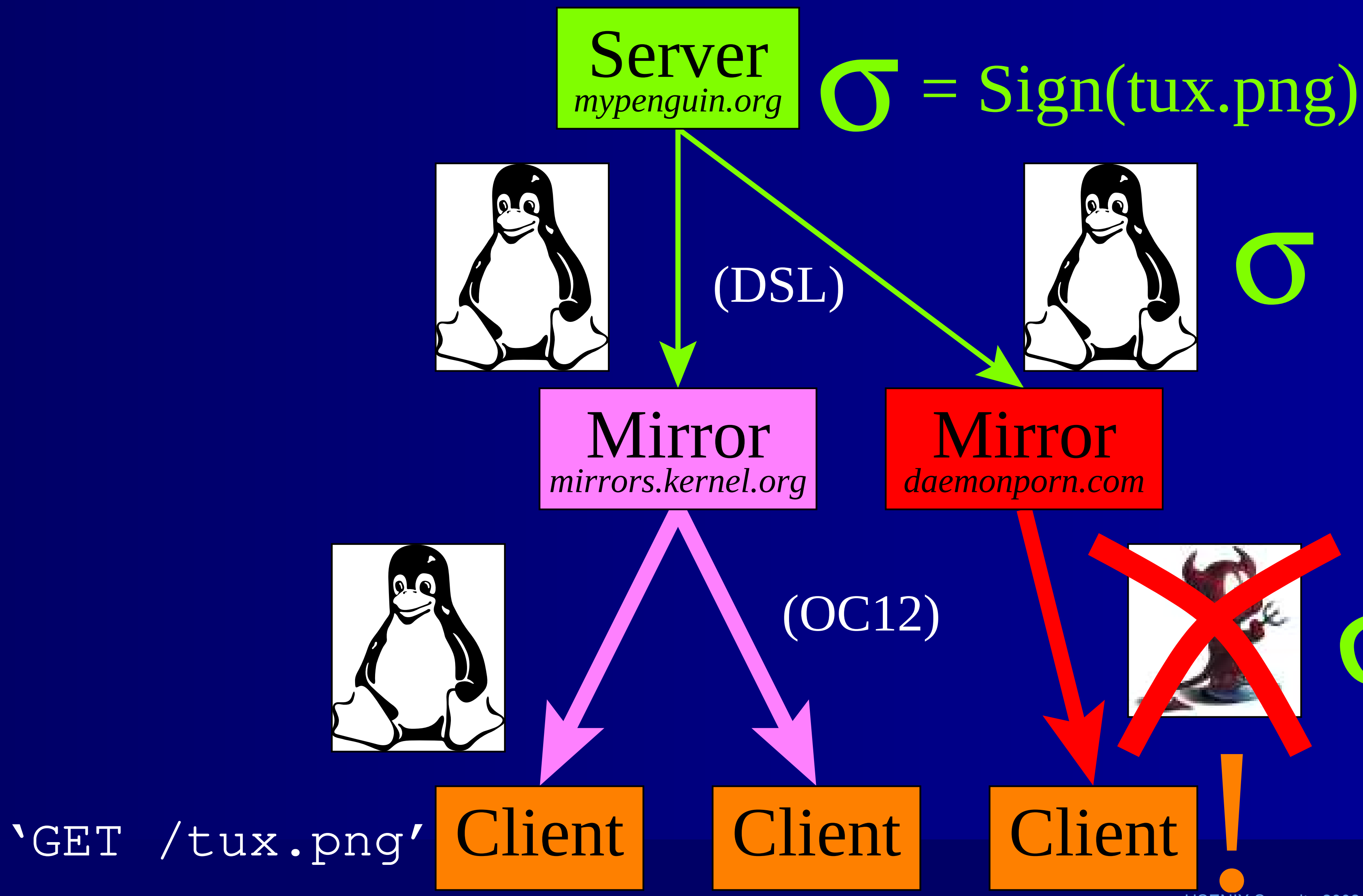
Bandwidth Offloading



Secure Bandwidth Offloading



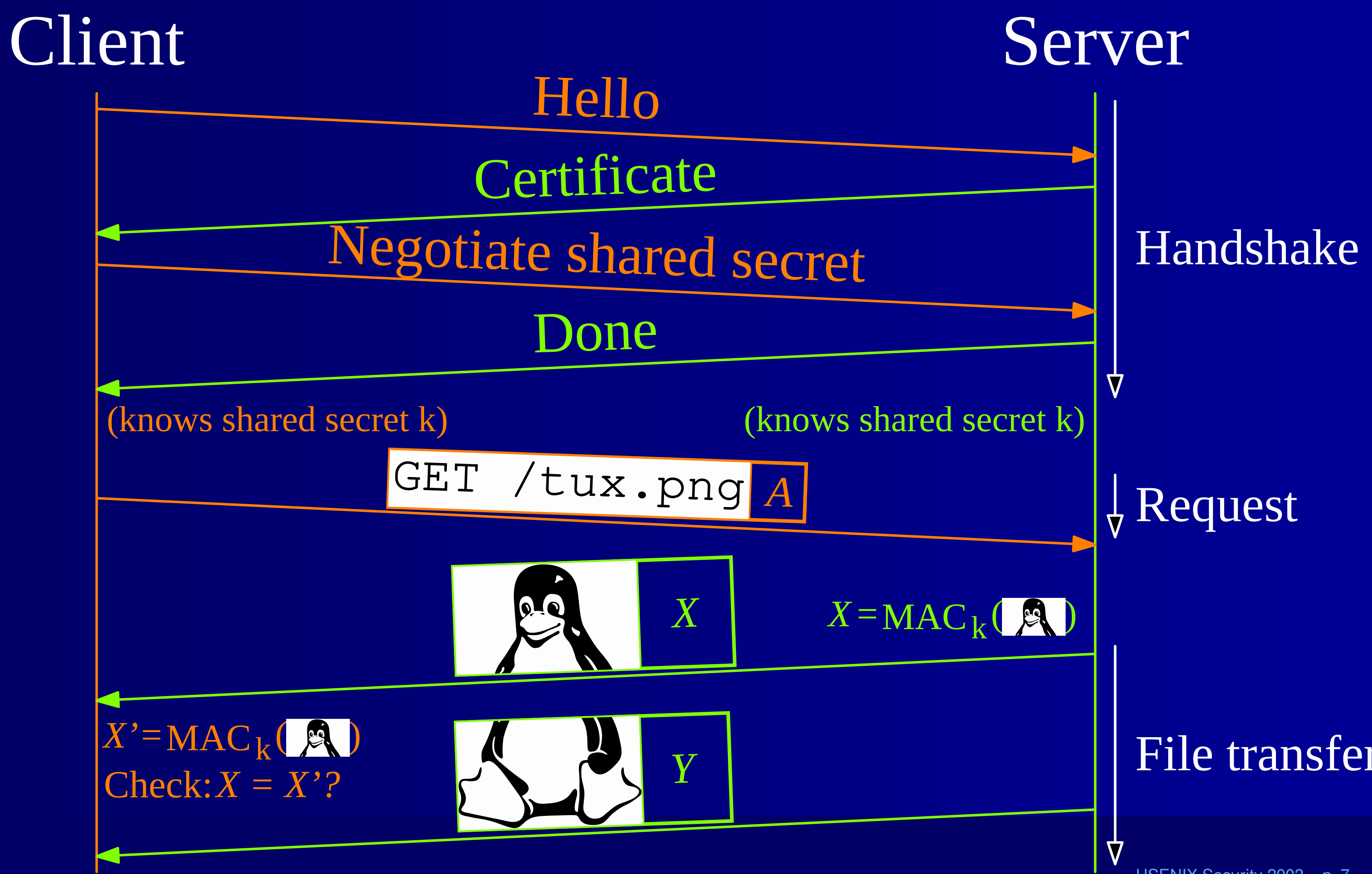
Secure Bandwidth Offloading



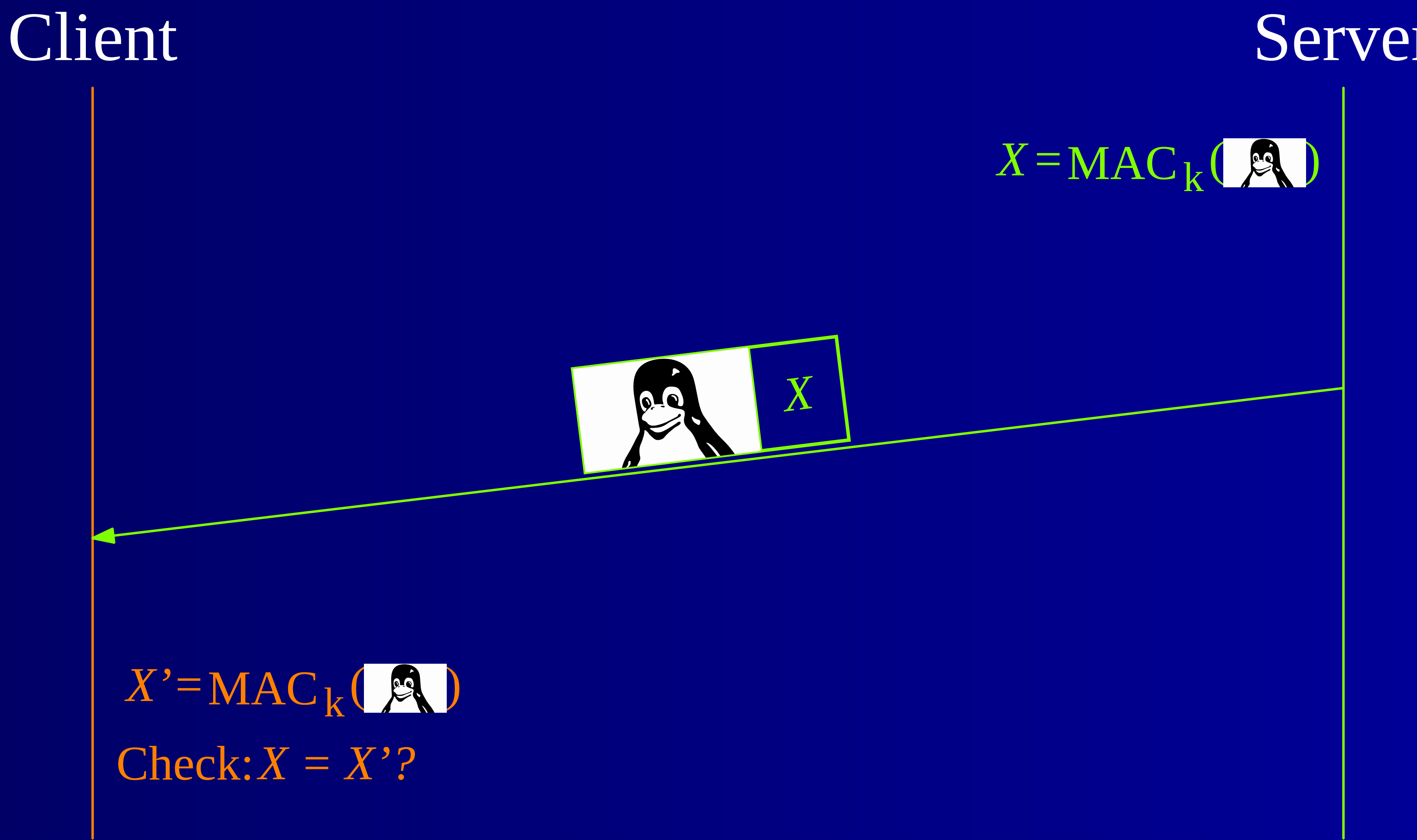
Existing Solutions Aren't Practical

- Force users to install specialized browser
  - Ex: S-HTTP, SFSRO, BitTorrent, RPM+PGP
- Operates at the channel level, not file level
  - Ex: SSL

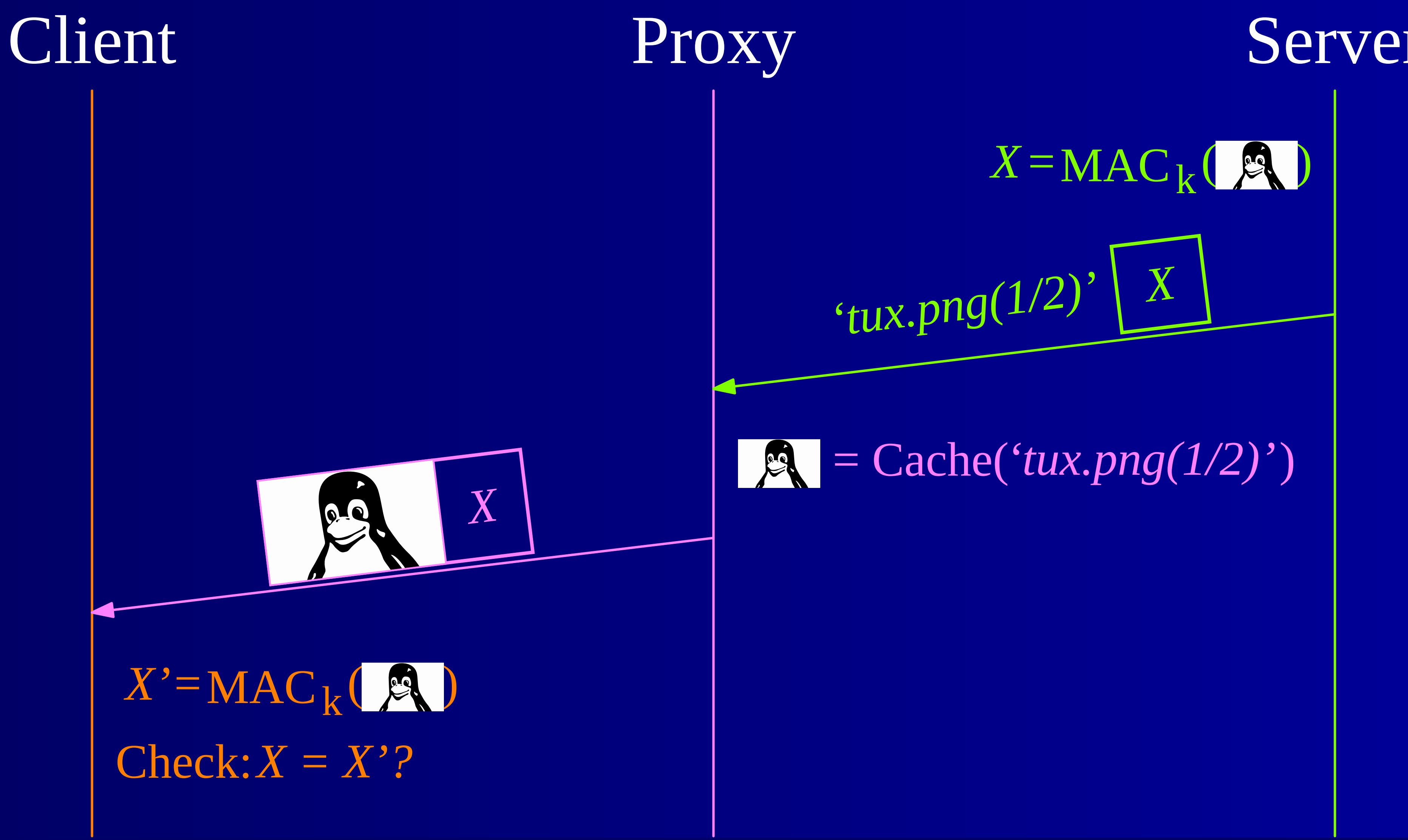
SSL’s Authentication Layer



When All You Have Is A Hammer...

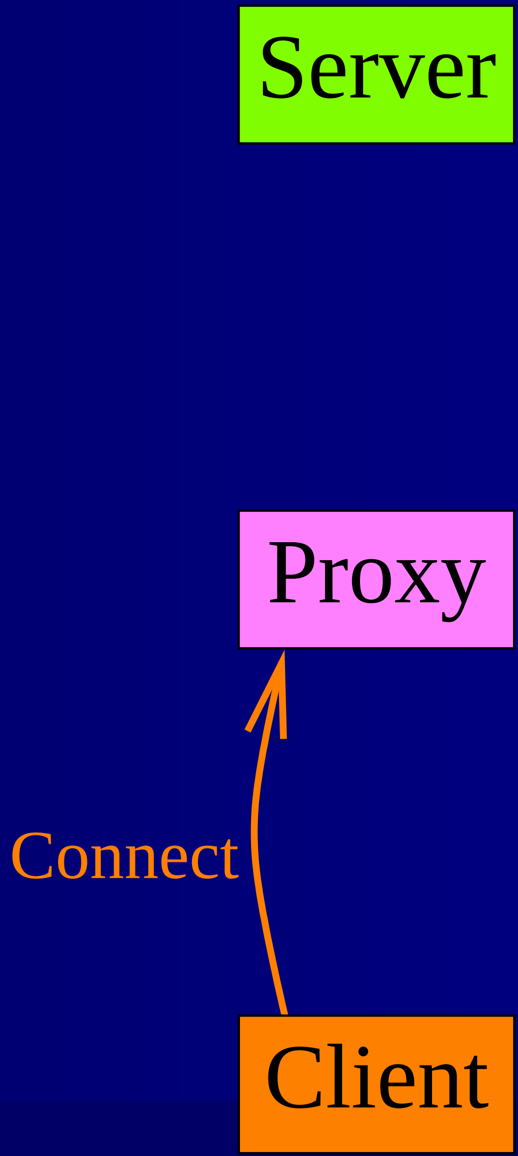


SSL Splitting



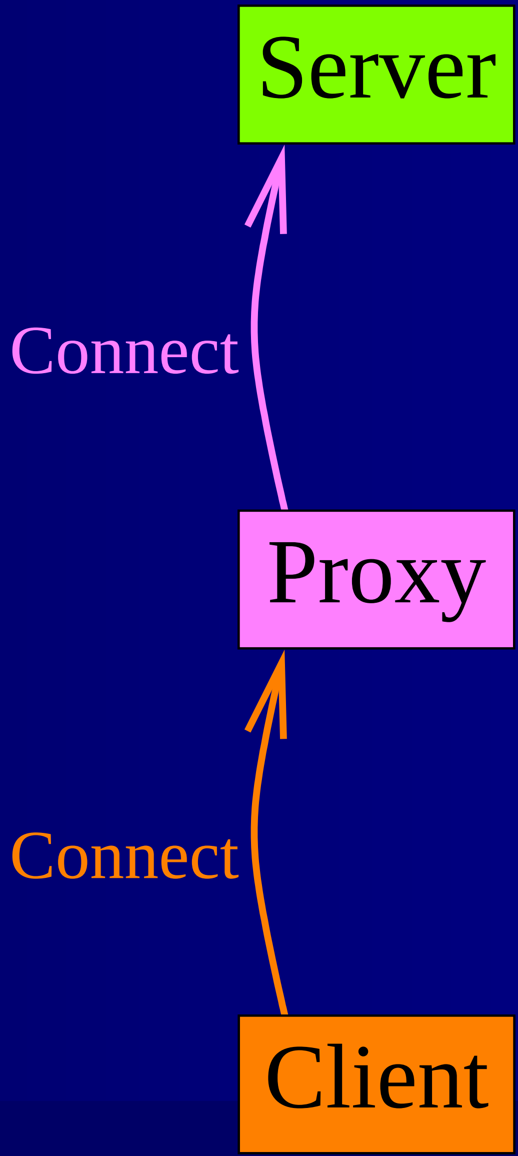
SSL Splitting

1. Connect



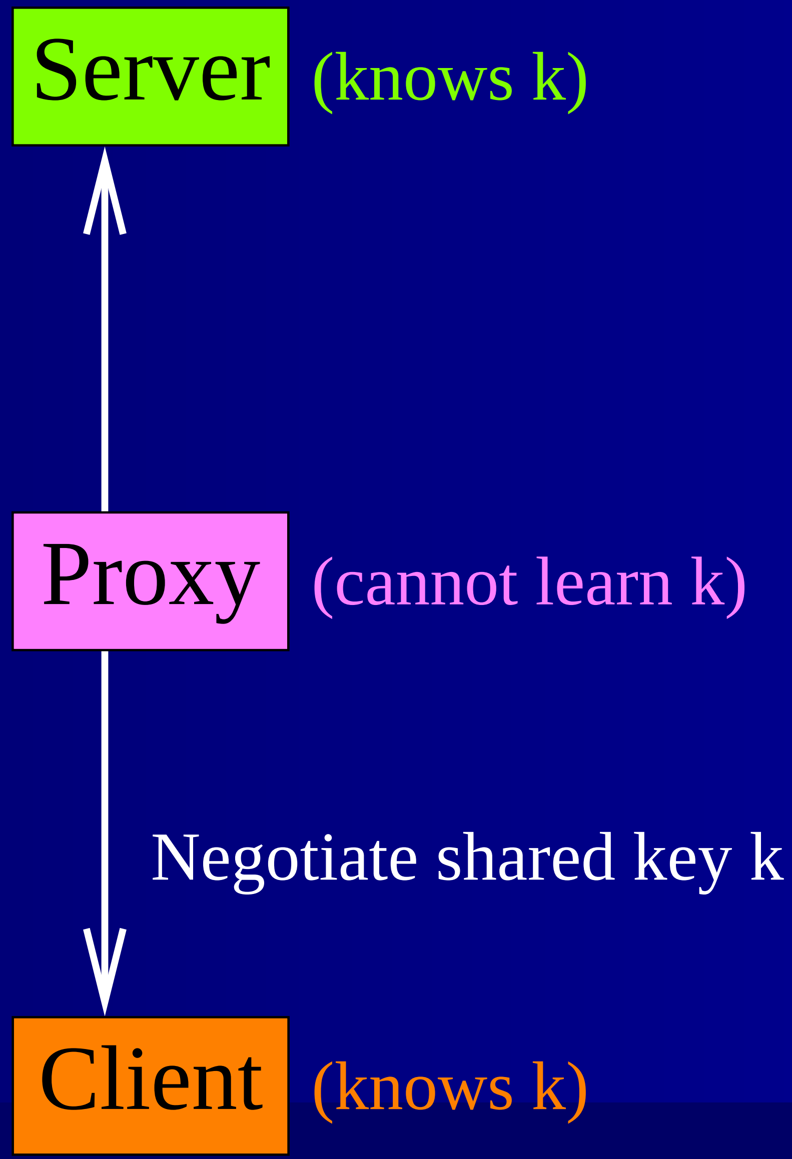
SSL Splitting

1. Connect



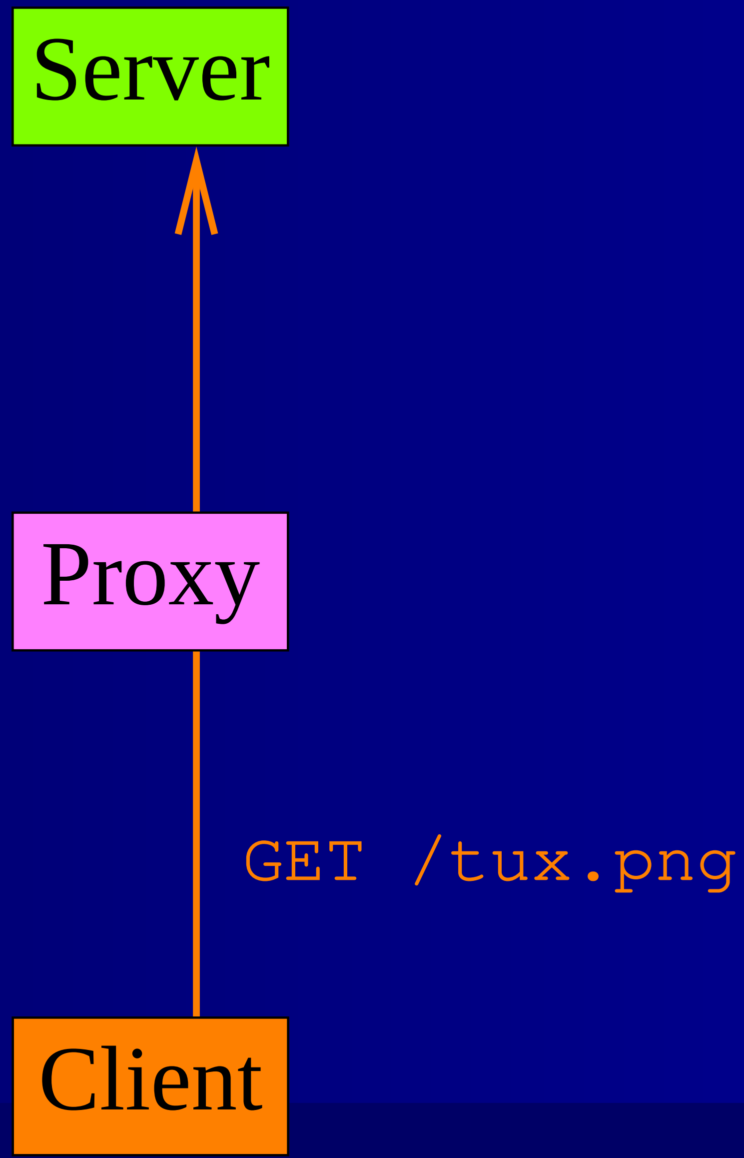
SSL Splitting

1. Connect
2. Handshake



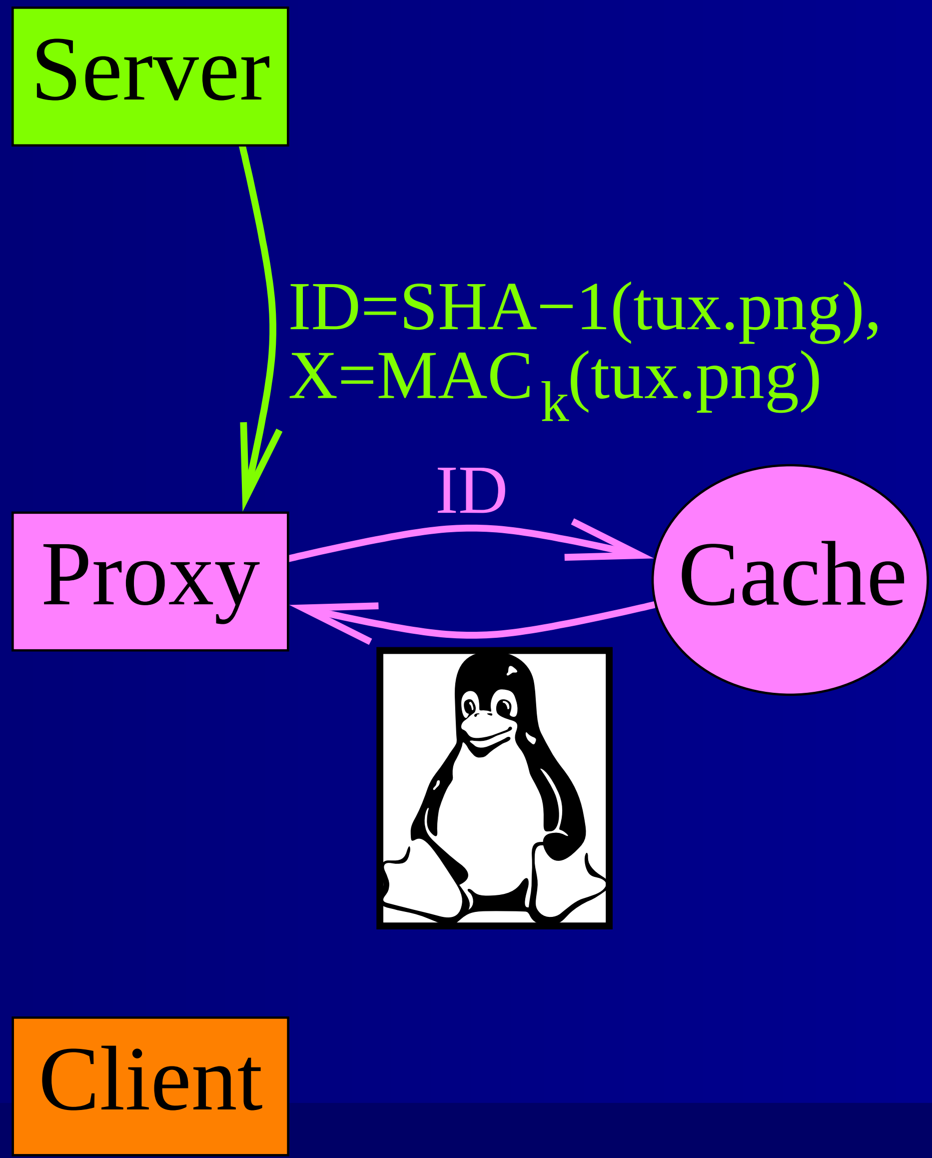
SSL Splitting

1. Connect
2. Handshake
3. Request



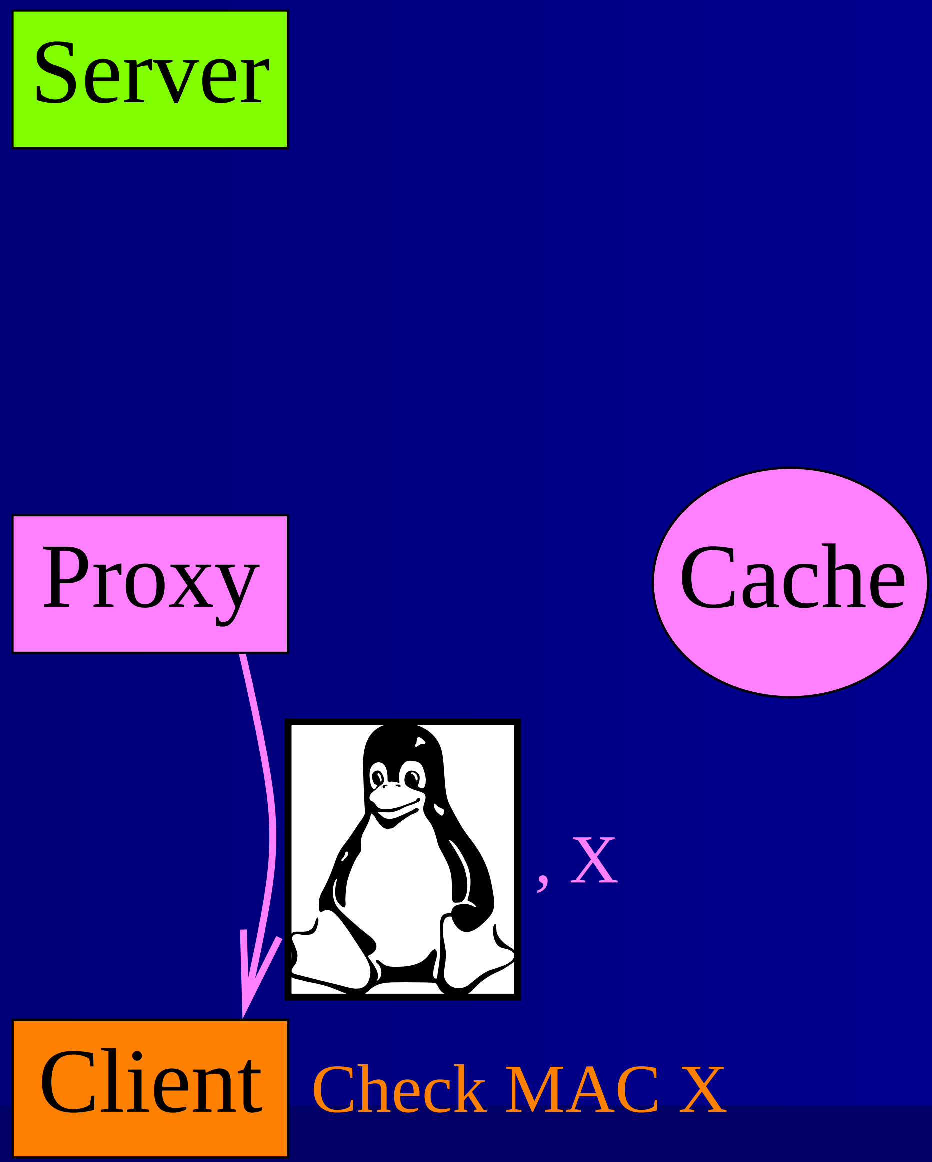
SSL Splitting: Cache Hit

1. Connect
2. Handshake
3. Request
4. Stub record

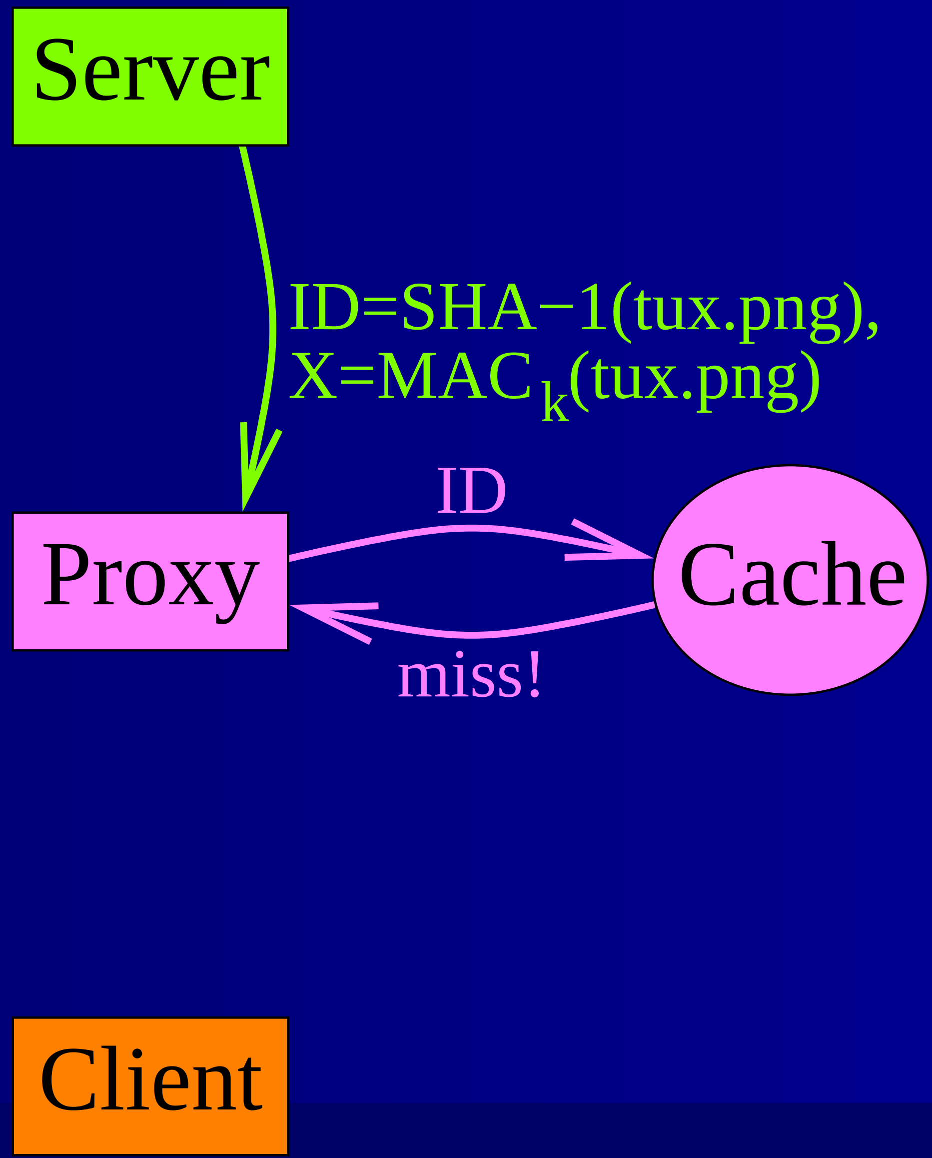


SSL Splitting: Cache Hit

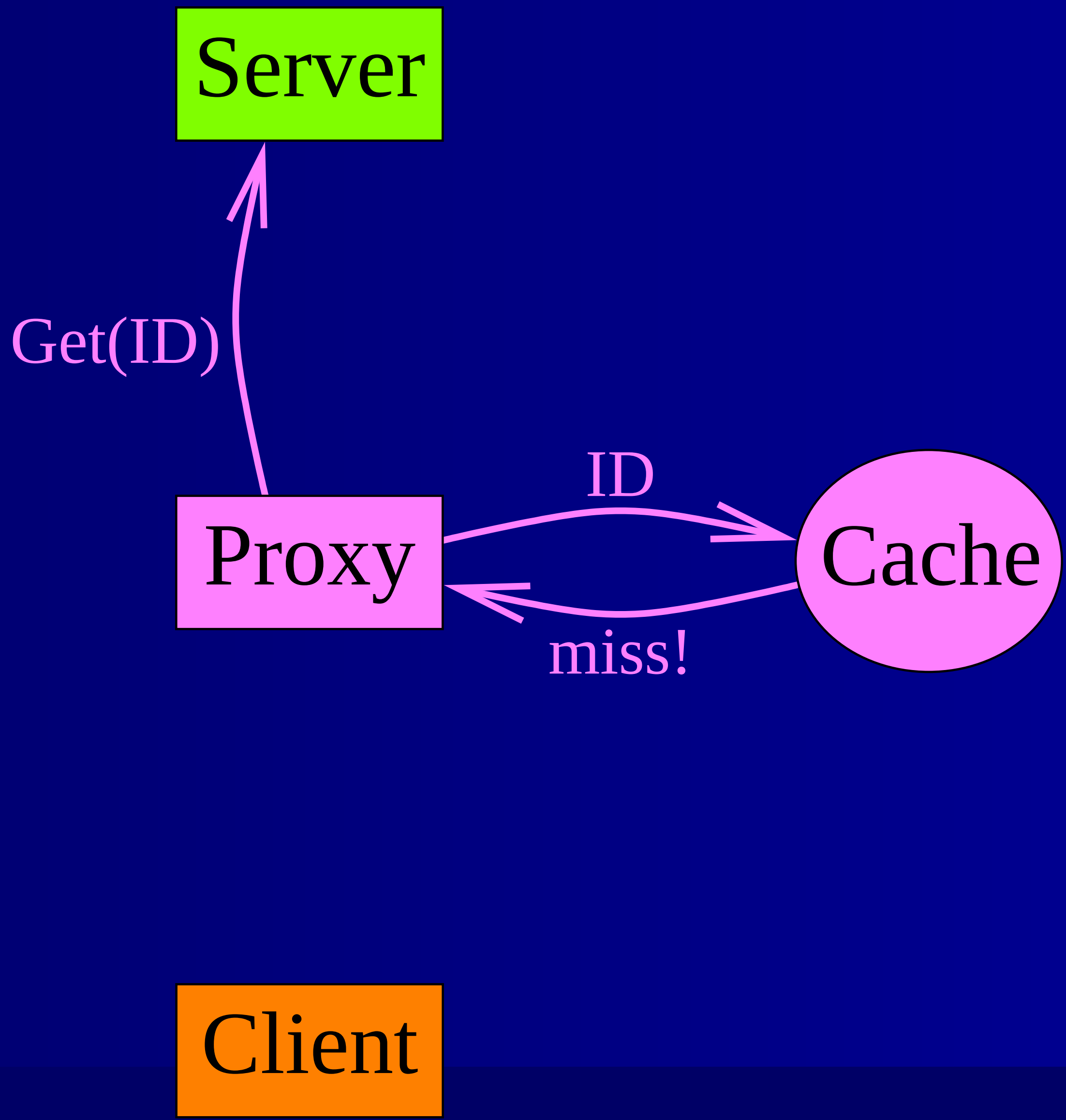
1. Connect
2. Handshake
3. Request
4. Stub record
5. Spliced record



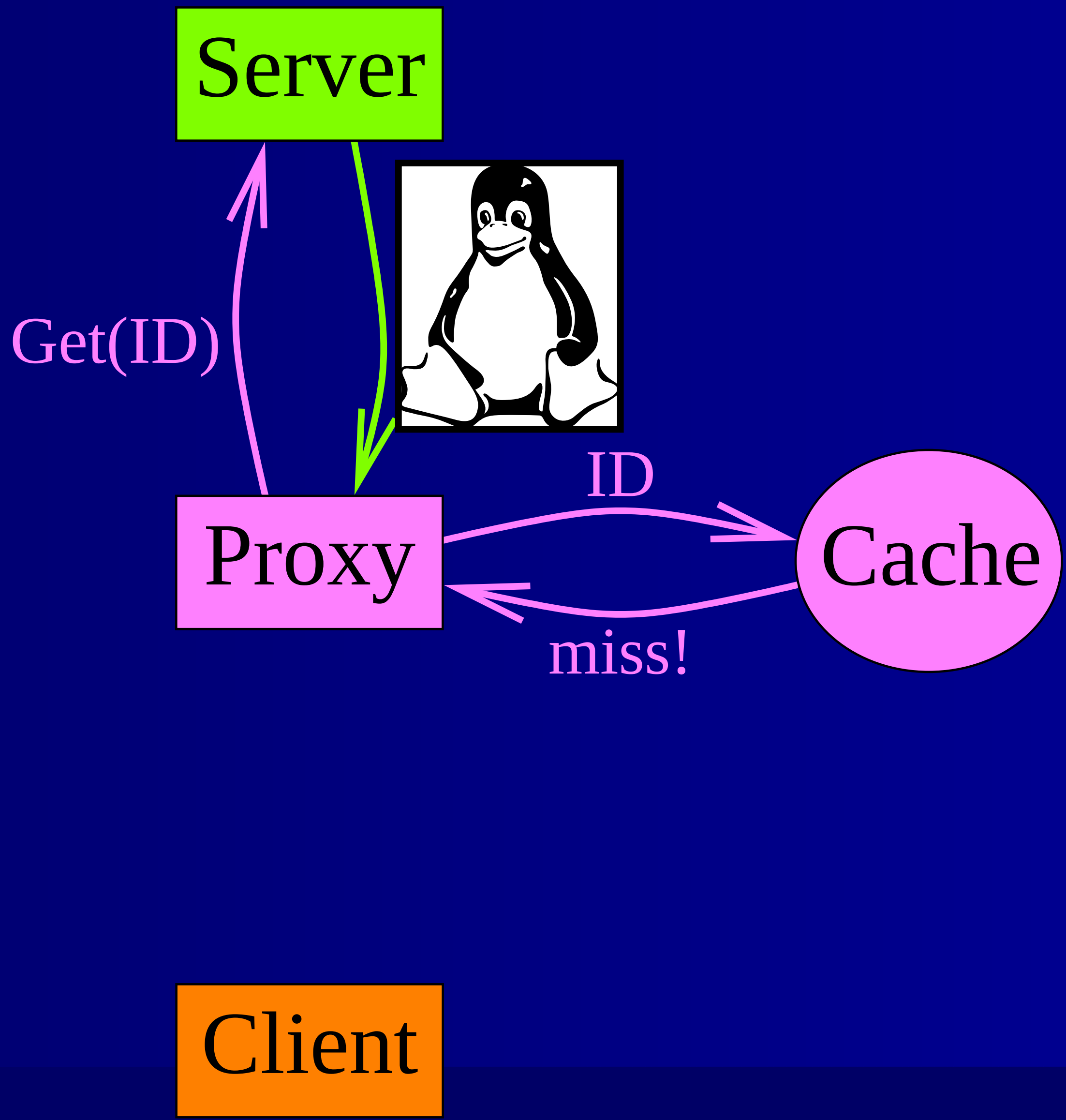
SSL Splitting: Cache Miss



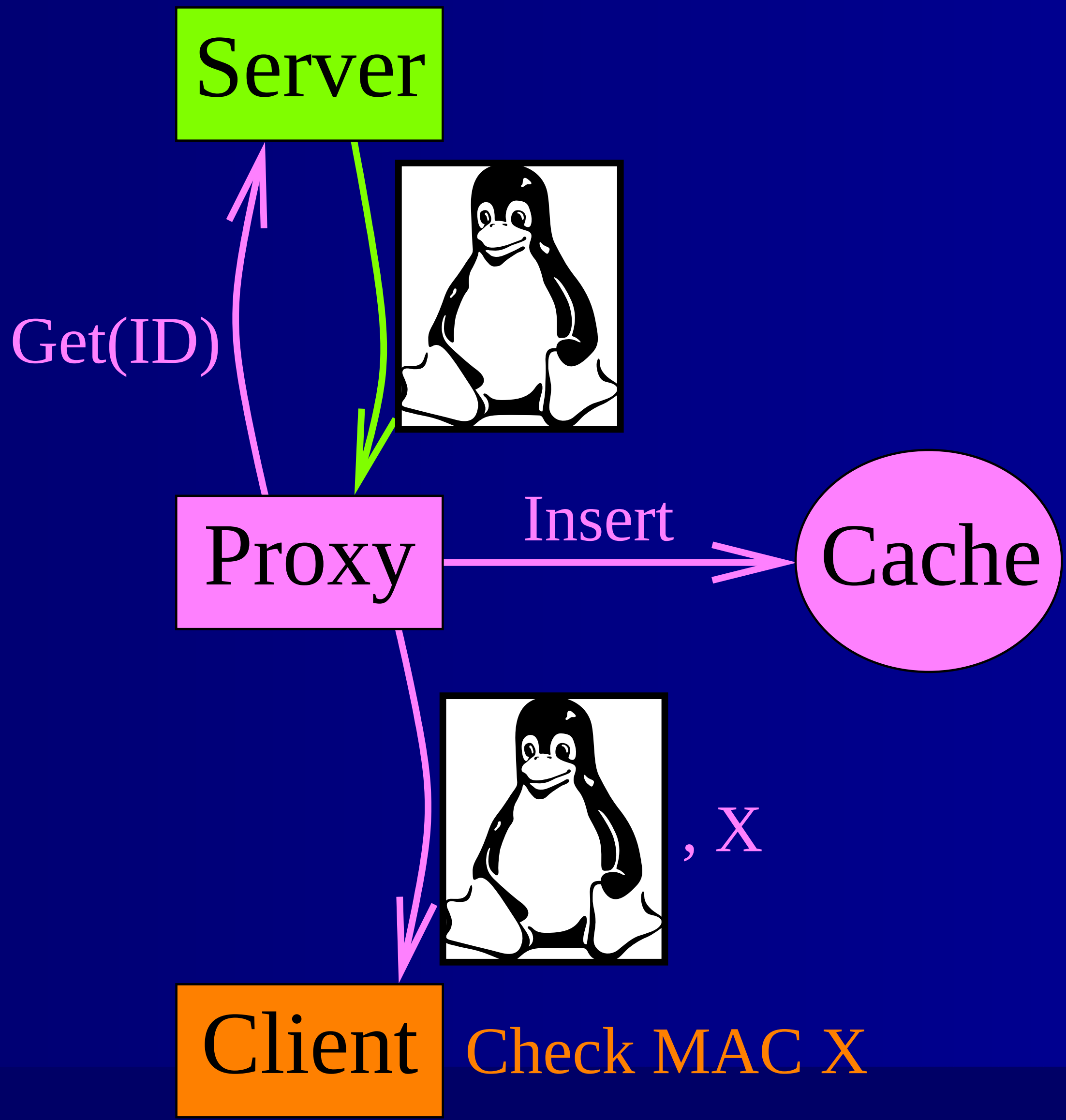
SSL Splitting: Cache Miss



SSL Splitting: Cache Miss



SSL Splitting: Cache Miss



Caveats

- No end-to-end confidentiality
- Only distributes bandwidth load, not CPU

Implementation

- Server
  - Unmodified Apache
  - Modified OpenSSL library
- Proxy: Perl and C
  - Splicing is not a cryptographic operation
- Client: Netscape, IE, w3m...

Performance Questions

- How much data do we send over the server-proxy link?
- How does overhead vary with file size?
- How much overhead with realistic file size distributions?

Experiments

- Client replayed prerecorded request patterns
- Measured bytes over server interfaces
- Key performance metric is "rate" r:

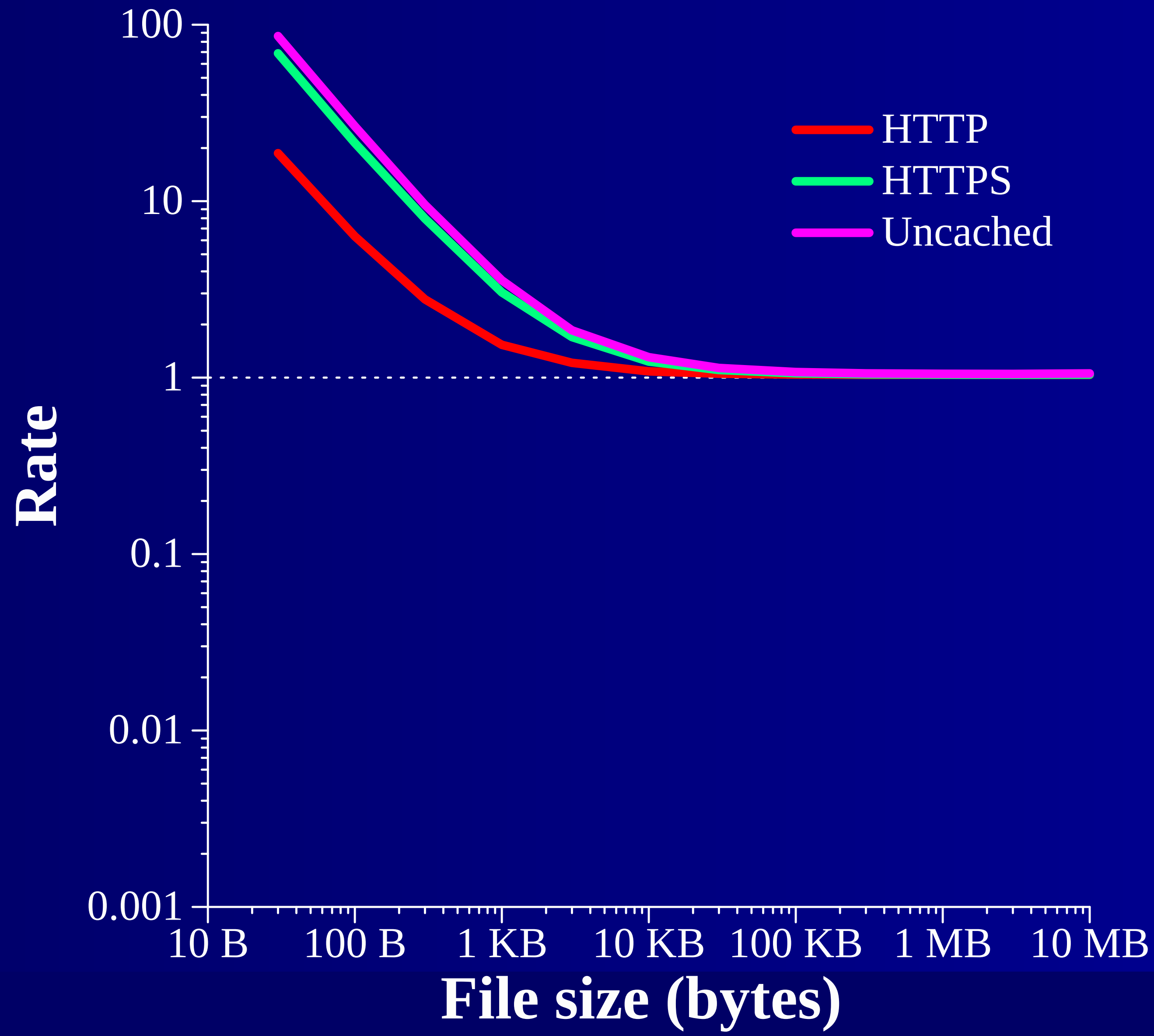
$$r = \frac{\textit{wire bytes sent by server}}{\textit{total size of files received by clients}}$$

  - Smaller is better
  - If no caching, $r = 1 + \%$ *overhead*

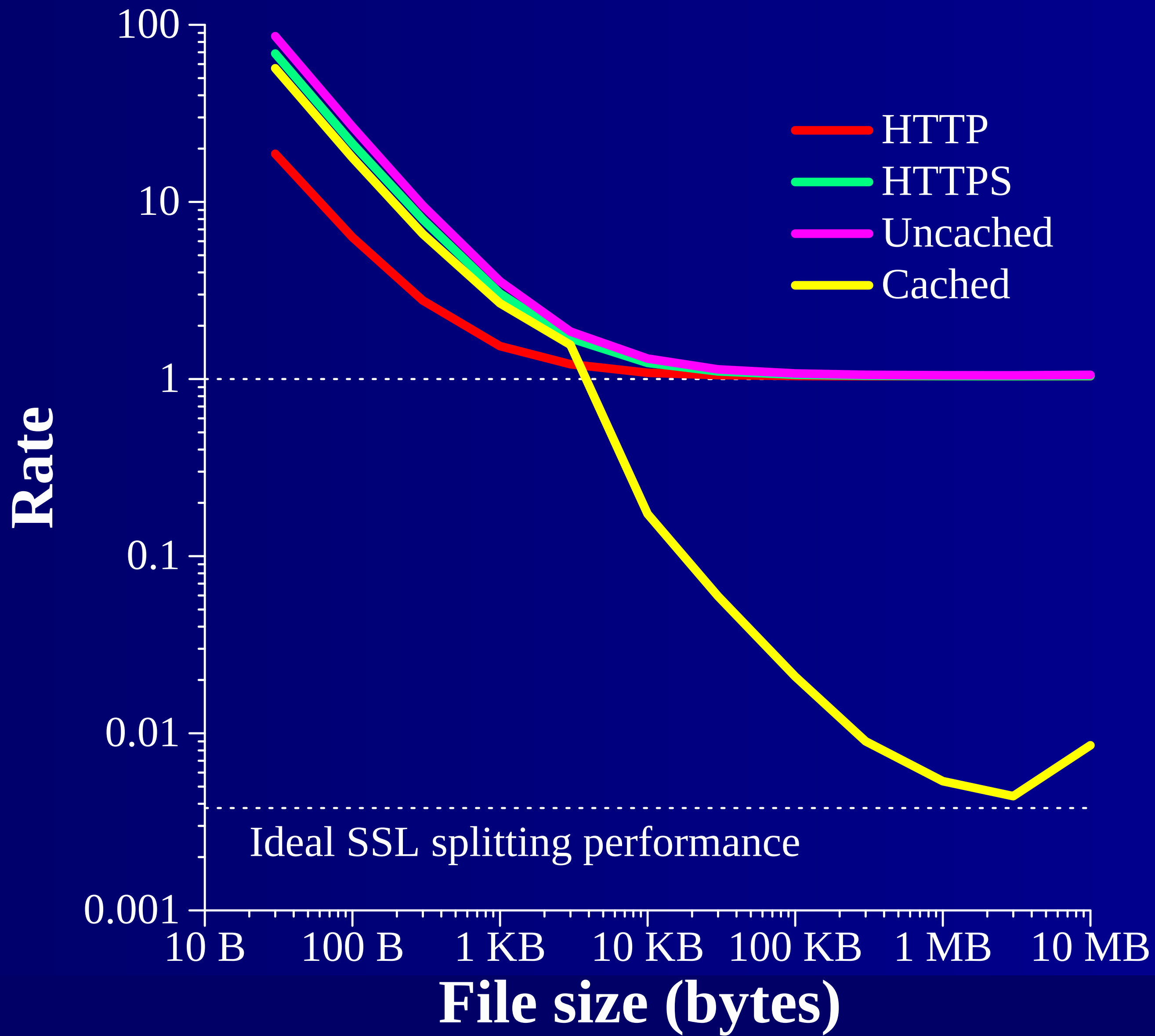
Experimental Setup

- Server: 160 kbps upstream, 500 MHz AMD
  - CPU could push $\approx$ 4 Mbps using HTTPS
- Client: 100 Mbps LAN, 1.2 GHz Athlon
- Proxy: 100 Mbps LAN, 700 MHz P3

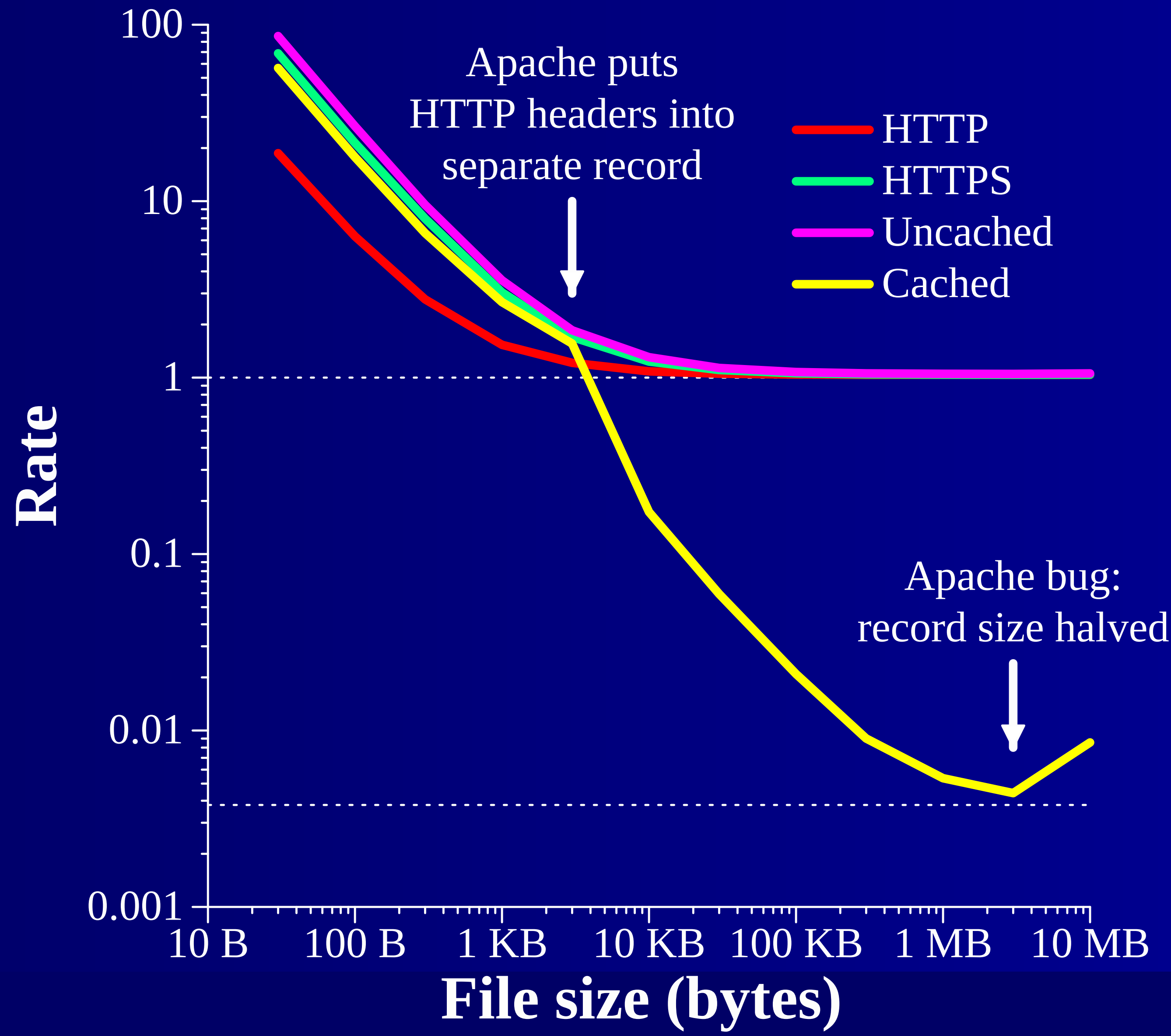
Single File Microbenchmark



Large Files Compress Well



Some Apache Quirks



Understanding Single File Results

- Model: $r = f(\textit{file size})$
- Constant 1.5 KB overhead per file
- Uncached: 5% overhead per byte
- Cached: 62 bytes sent per 16 KB record
  - 8 KB records for files > 4 MB

Real Workloads

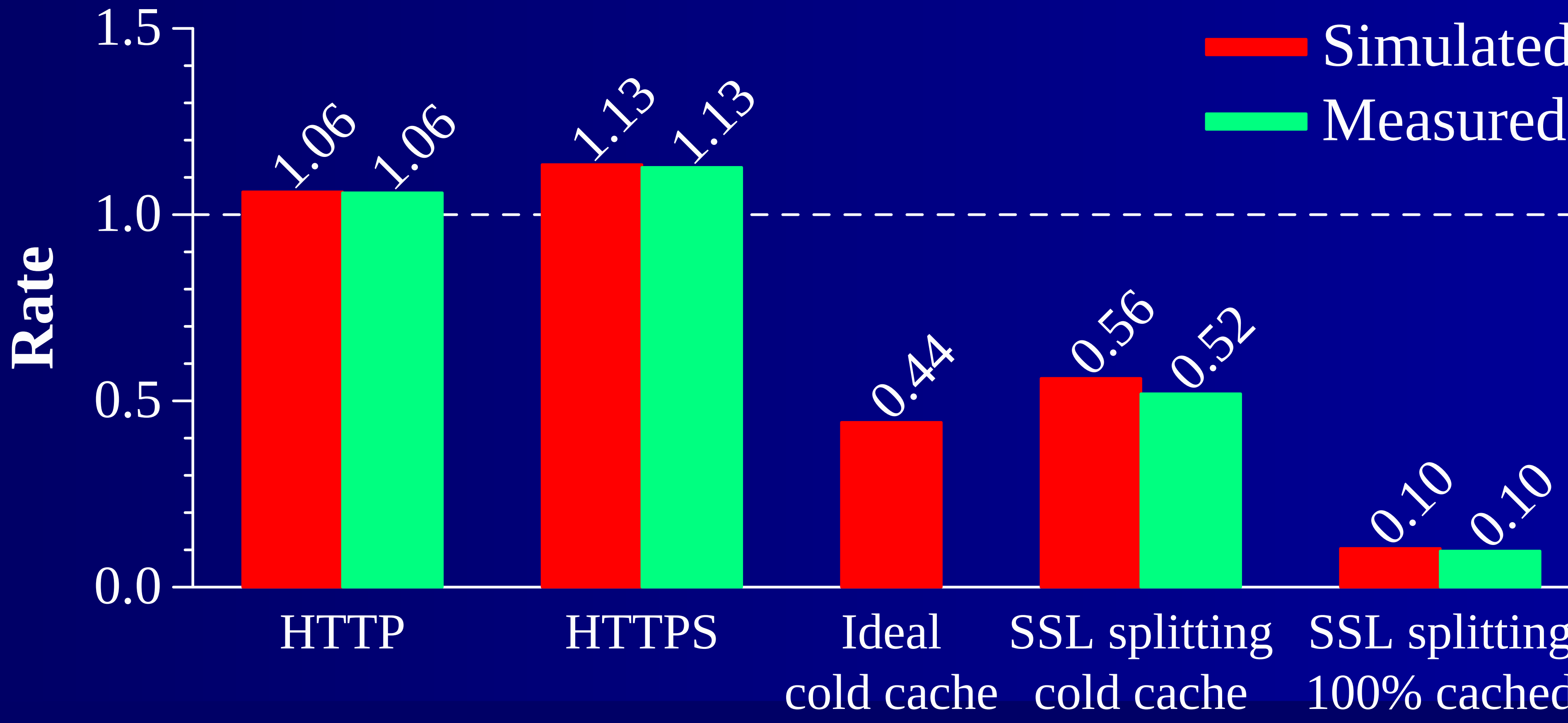
- Do real access patterns benefit from SSL splitting?
- 7-month web traces taken from `www.lcs.mit.edu` and `amsterdam.lcs.mit.edu`

How The Simulator Works

- Input: list of file requests and sizes
- Use microbenchmark results to predict number of bytes sent by server
- Infinite cache

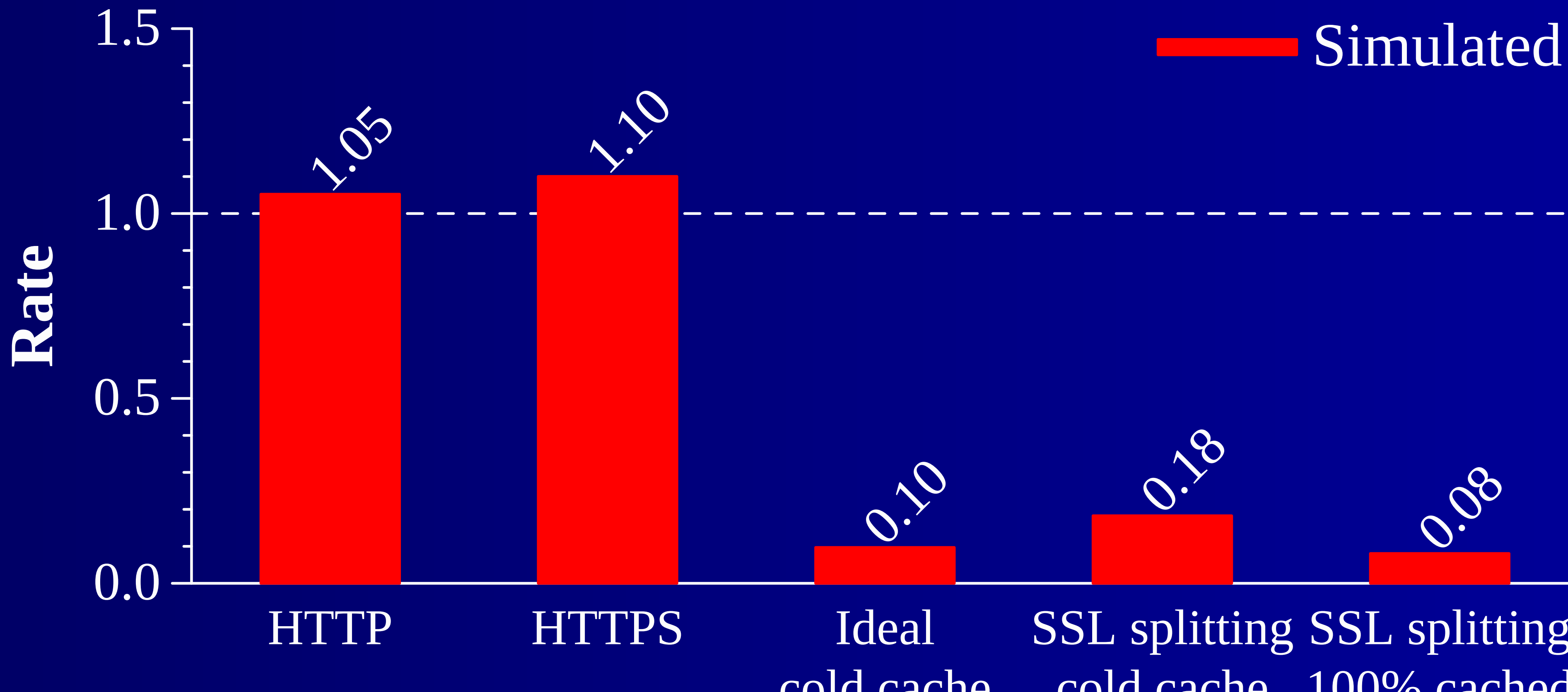
Simulation Accuracy

- 2 hours, 10 MB transferred, 4.43 MB of files



Long-Term Savings ≈ 83%

- 7 months, 109 GB transferred, 10.6 GB of files



Summary

- SSL Splitting does not:
  - Provide confidentiality
  - Reduce server CPU load
- SSL Splitting does:
  - Reduce server bandwidth use by 25–90%
  - Guarantee end-to-end data integrity
  - Work with normal Web browsers!
- You might use it if: you're a Web site admin and you're not sure you trust your mirrors.

Availability

`http://pdos.lcs.mit.edu/barnraising/`