



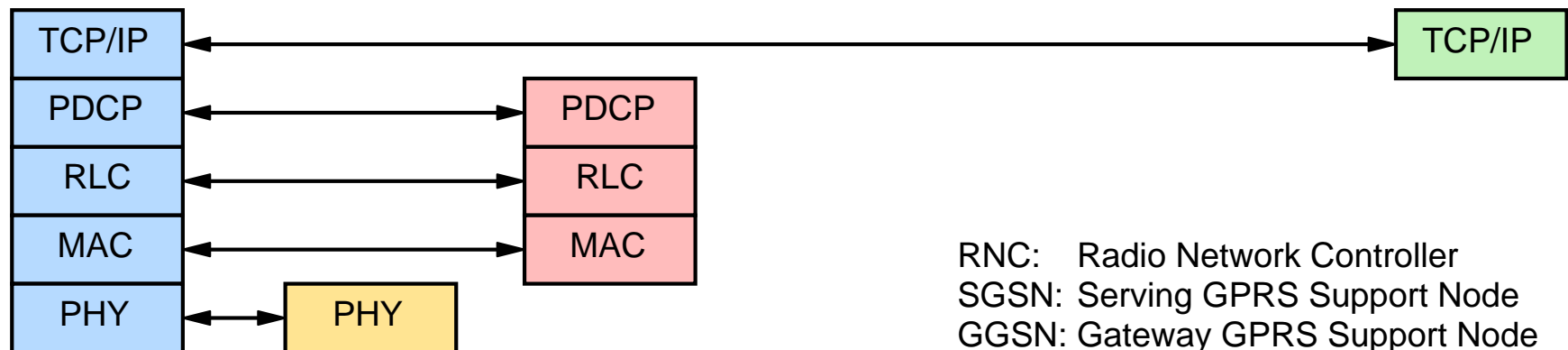
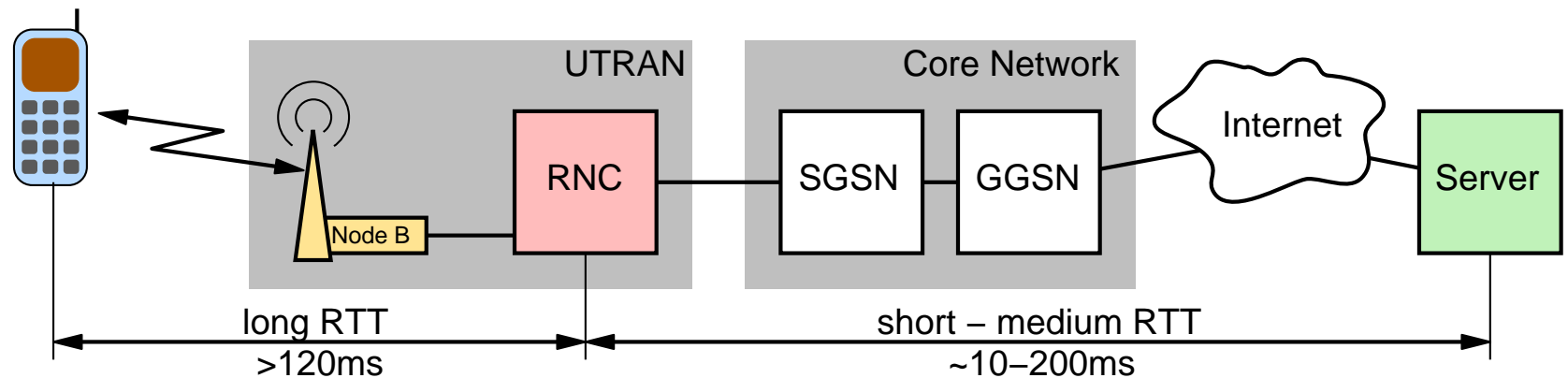
Performance Evaluation of different proxy concepts in UMTS networks

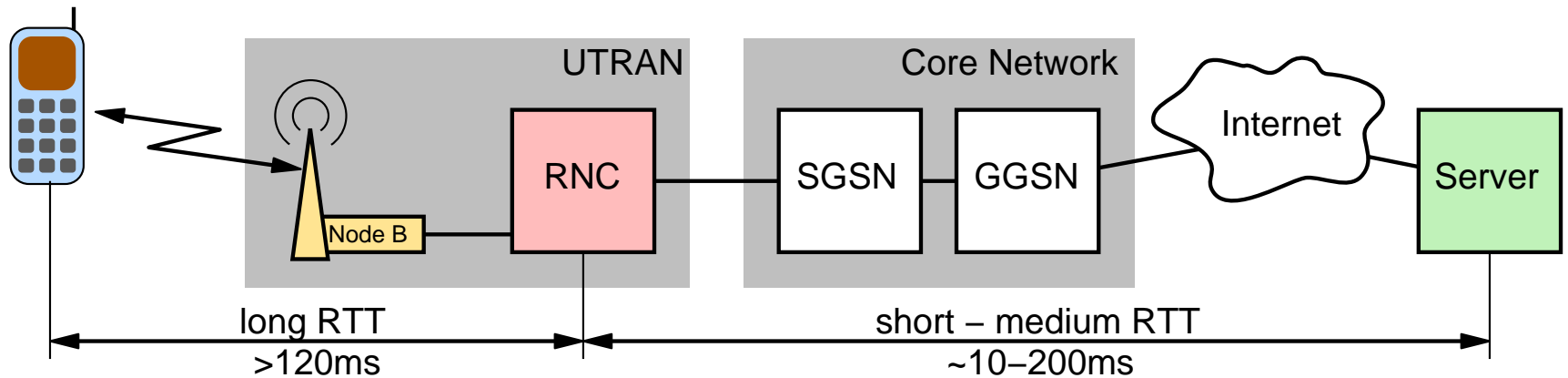
Marc Necker (IKR), Andreas Weber (Alcatel SEL)
necker@ikr.uni-stuttgart.de

Gozo, October 8, 2004

- Outline:**
- Introduction to UTRAN and Proxies
 - Szenario and Simulation Model
 - Simulation and Emulation Environment
 - Results

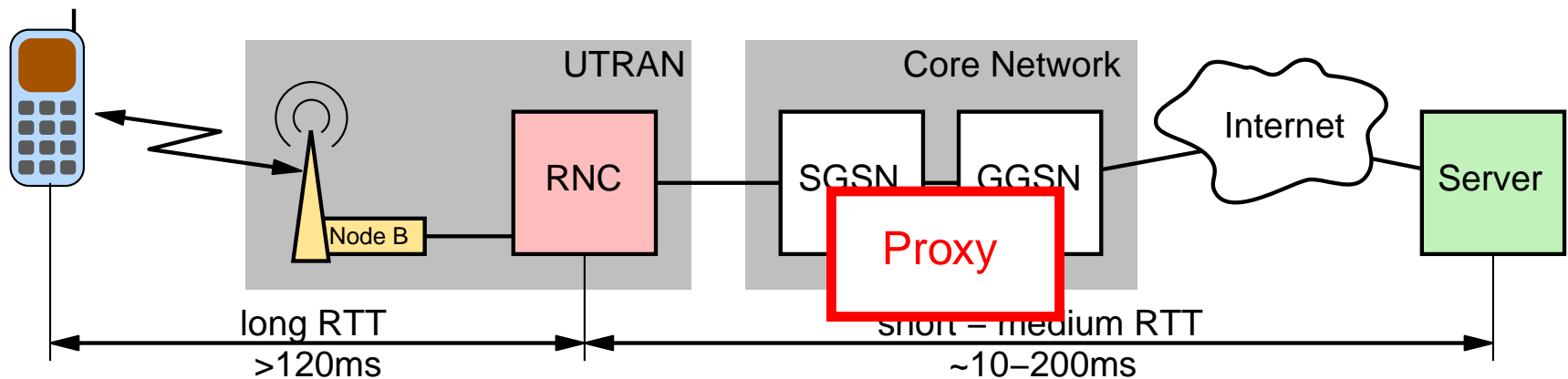
UMTS Terrestrial Radio Access Network (UTRAN)





Main Problem: large RTT in UTRAN

- RTT may be on the order of 0.5-1s for IP-packets
- ↳ long slow-start phase with TCP
- ↳ slow reactions to errors (packet losses)

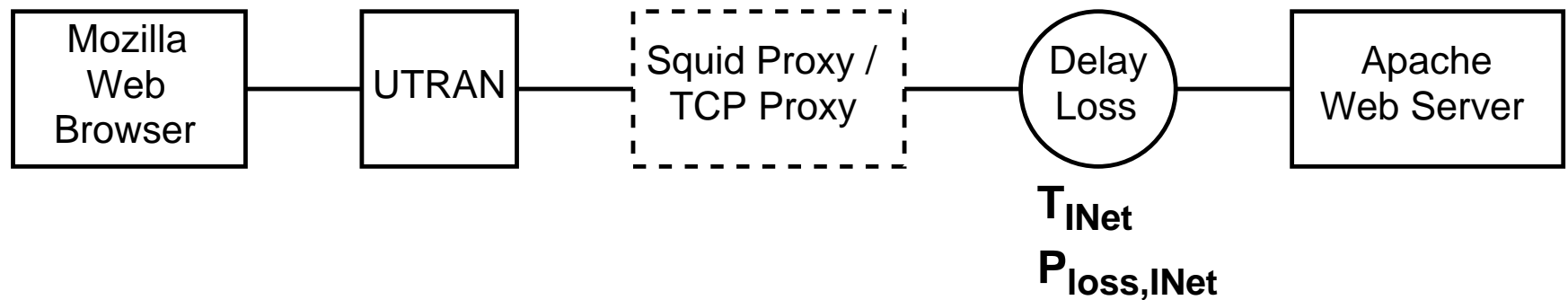
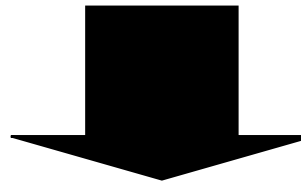
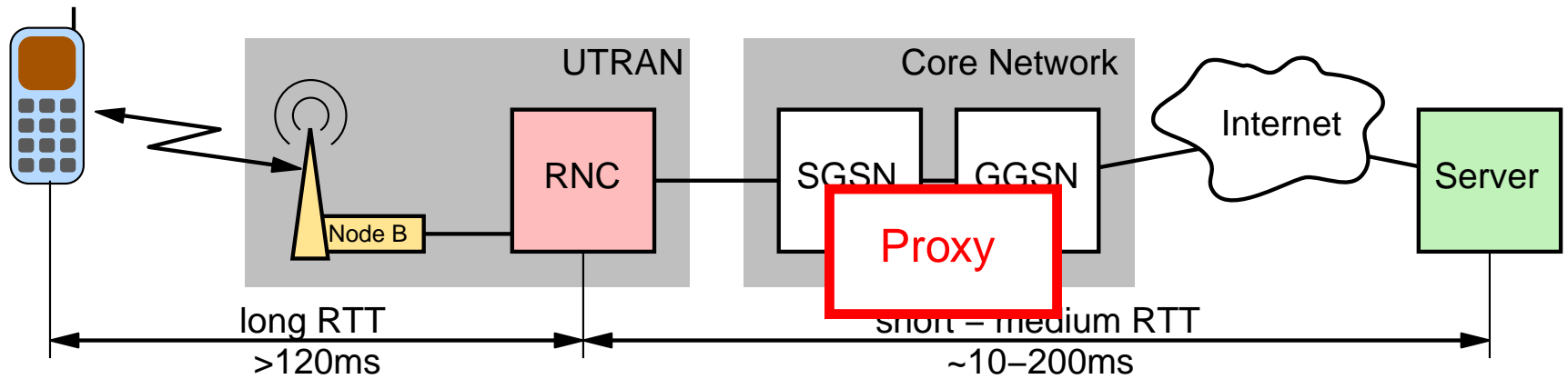


Possible solution

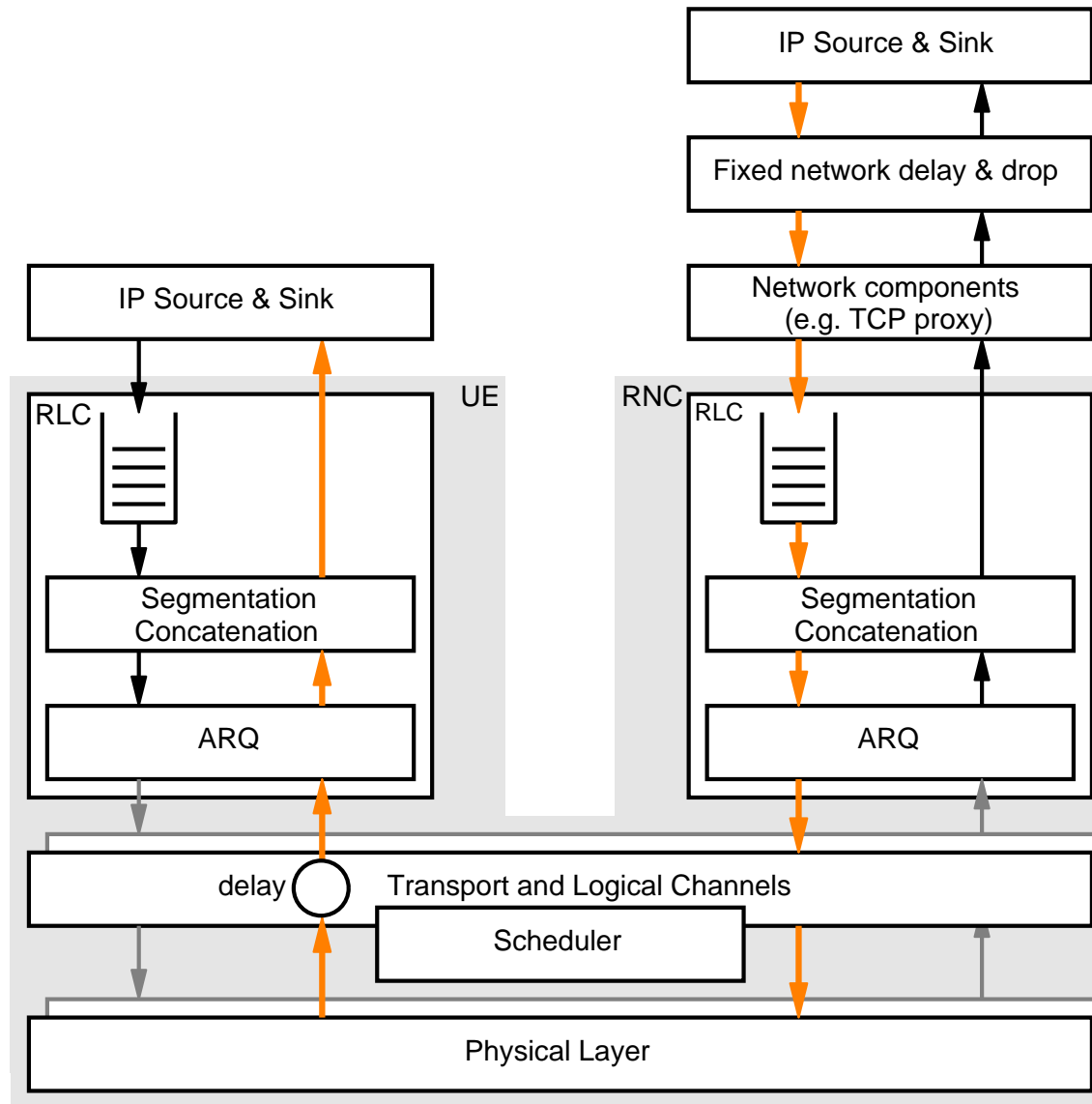
Introduce split-connection proxy in core network

- **Decouple fixed network part from wireless network part**
 - ↳ Fast reaction to errors within the fixed part
- **Use optimized TCP stack towards the wireless network**
 - ↳ Reduce time in slow-start
- **Proxy may be on the TCP or HTTP layer**

Szenario



UTRAN Simulation and Emulation Model



- **single cell**
- **single user**
- **ideal power control**
- **Downlink**
 - DCH 256 kBit/s
 - $P_{\text{loss,DL}}=20\%$
- **Uplink**
 - DCH 256 kBit/s
 - $P_{\text{loss,UL}}=10\%$

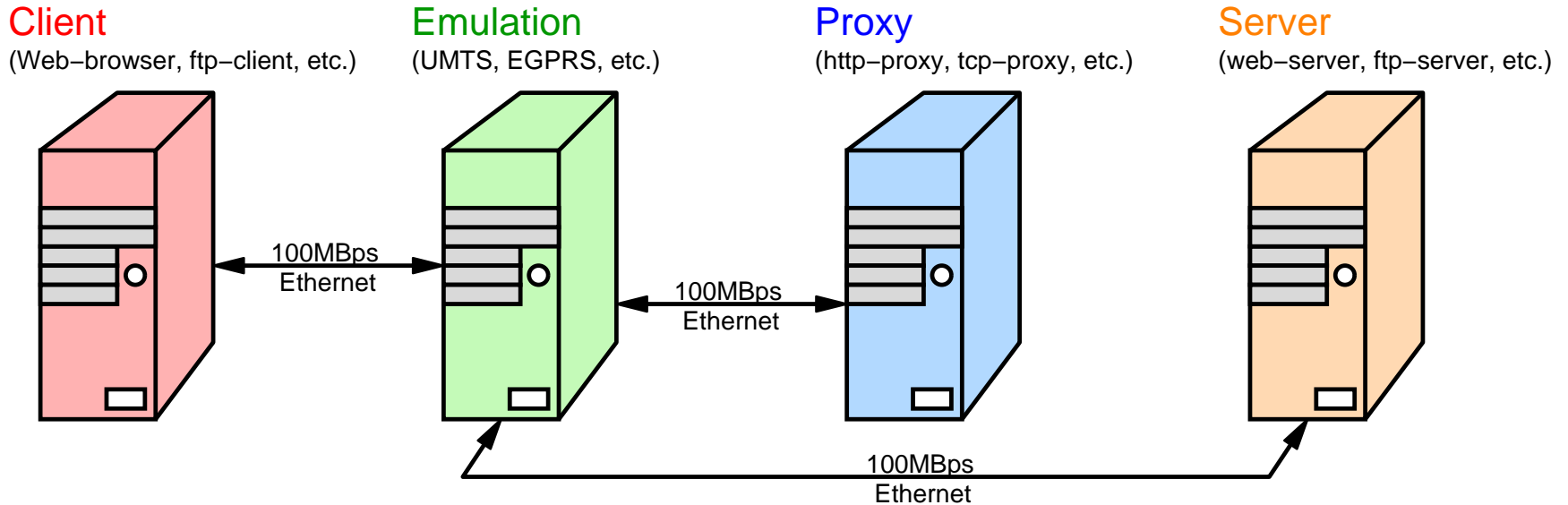
Simulation Environment

- event-driven simulation
- based on *IKR SimLib*

Emulation Environment

- inclusion of real world components (Linux TCP stack, Mozilla, etc.)
- based on *IKR SimLib* and *IKR EmuLib*
- uses the same simulation model (no modifications to simulation tool necessary)
- ↳ easy performance evaluation with web traffic sources (HTTP traffic)

Emulation Setup



Applications

- **Mozilla 1.6**
- **Apache 1.3.29, Mirror of `www.tagesschau.de`**
- **Squid 2.5 or hand-written TCP proxy**

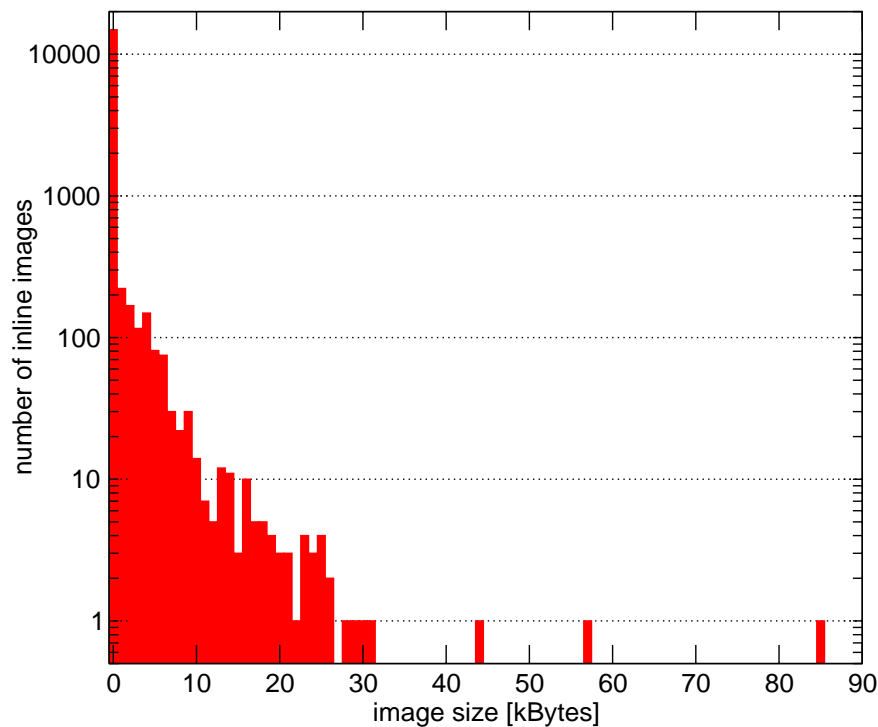
- Mozilla:**
- **HTTP-Version: 1.1**
 - keep-alive / no keep-alive
 - pipelining / no pipelining
 - **Cache: 1 MByte**
 - Cache small images (transparent pixels, arrows, etc.)
 - do not cache large images
 - replace large images and HTML pages during one cycle

- Apache:**
- **3 important parameters to control persistent connections:**
 - KeepAlive: on
 - MaxKeepAliveRequests: 100
 - KeepAliveTimeout: 15 [seconds]

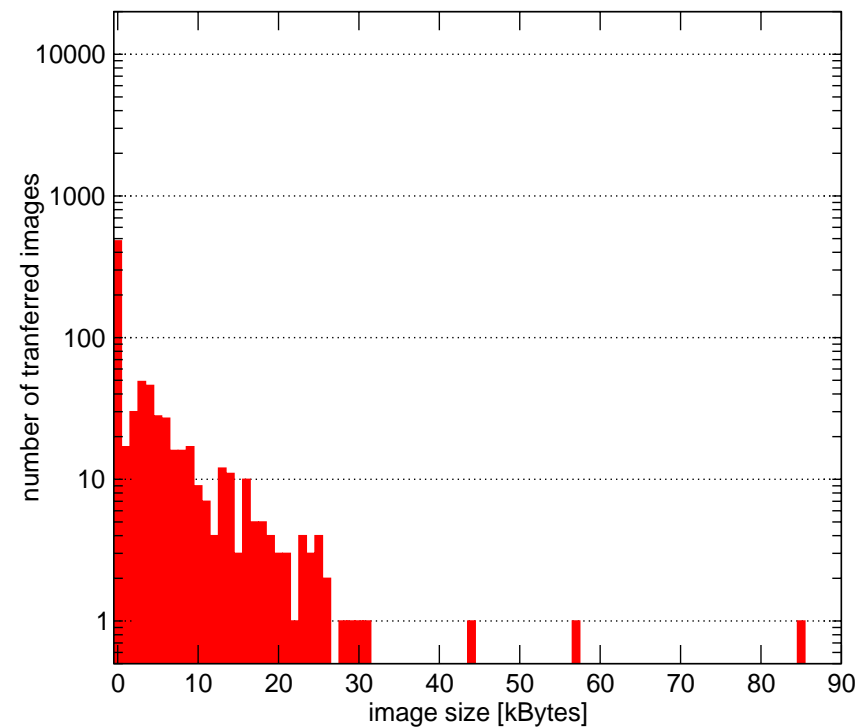
- Squid:**
- **1MByte of memory and disc cache, respectively**
 - **replace large images and HTML pages during one cycle**
 - **forced to use cached copies, if available**

- **Subset of www.tagesschau.de and related sites locally hosted**
- **Surf list of 135 pages**

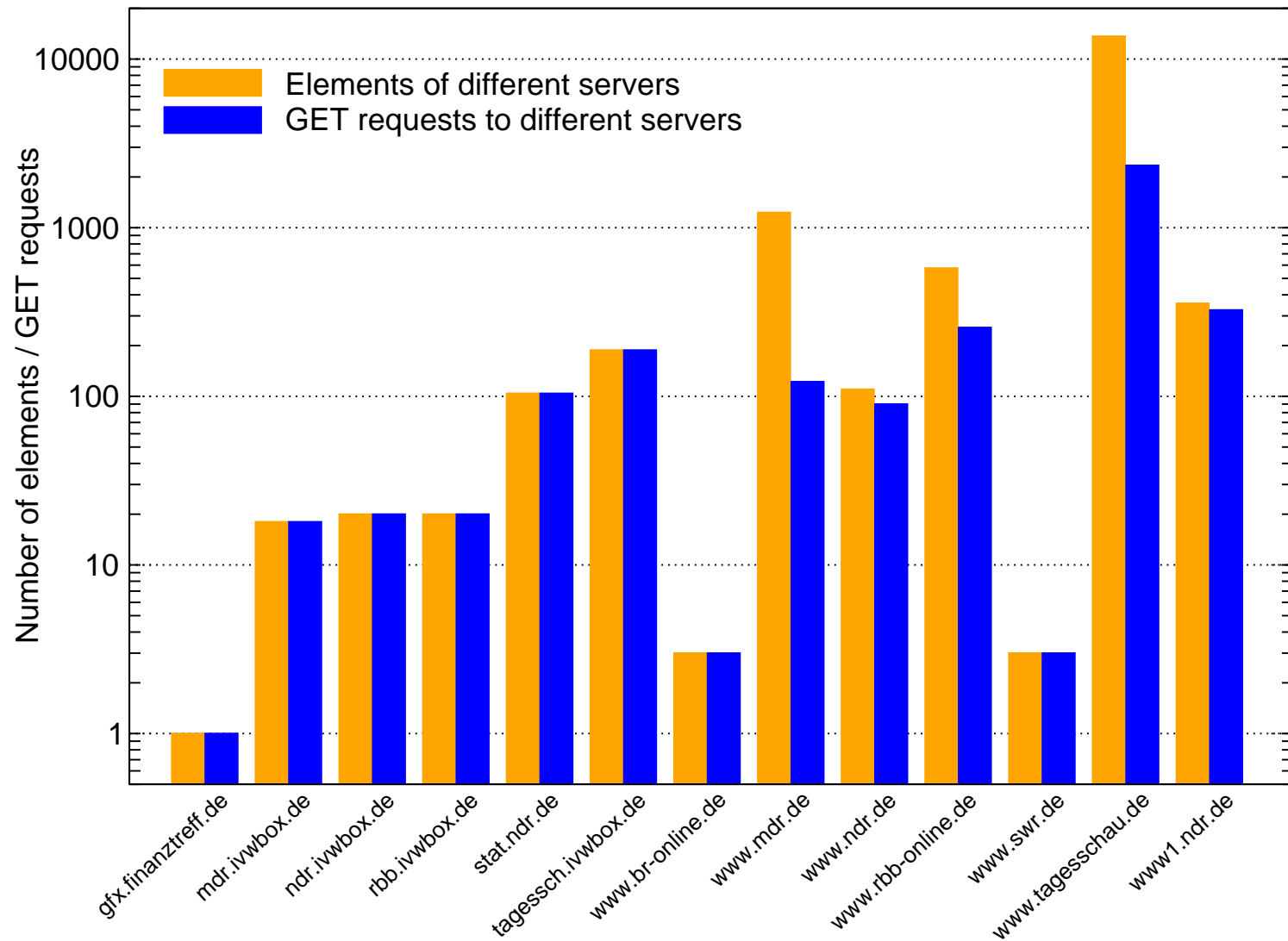
Histogram of inline images



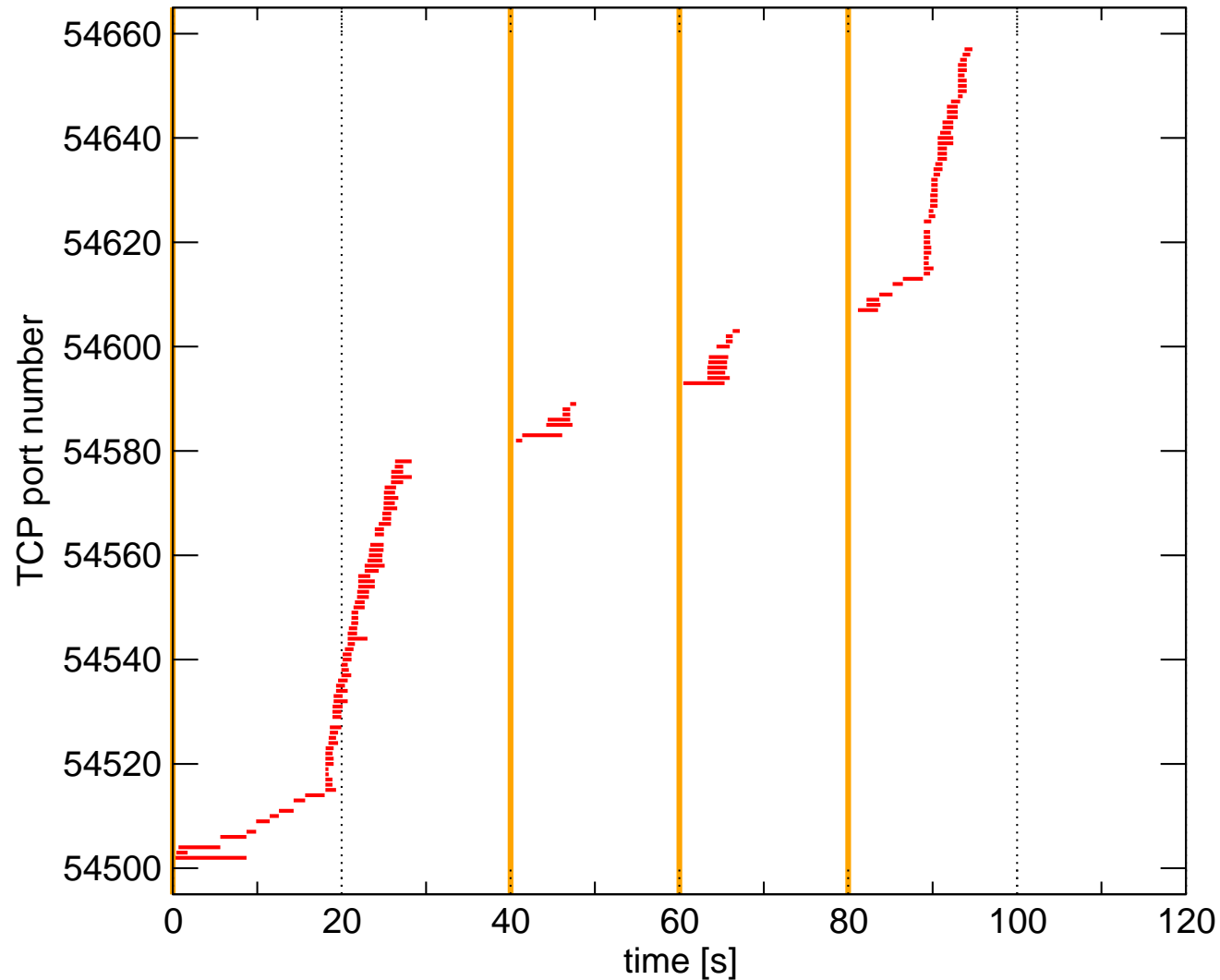
Histogram of transmitted inline data



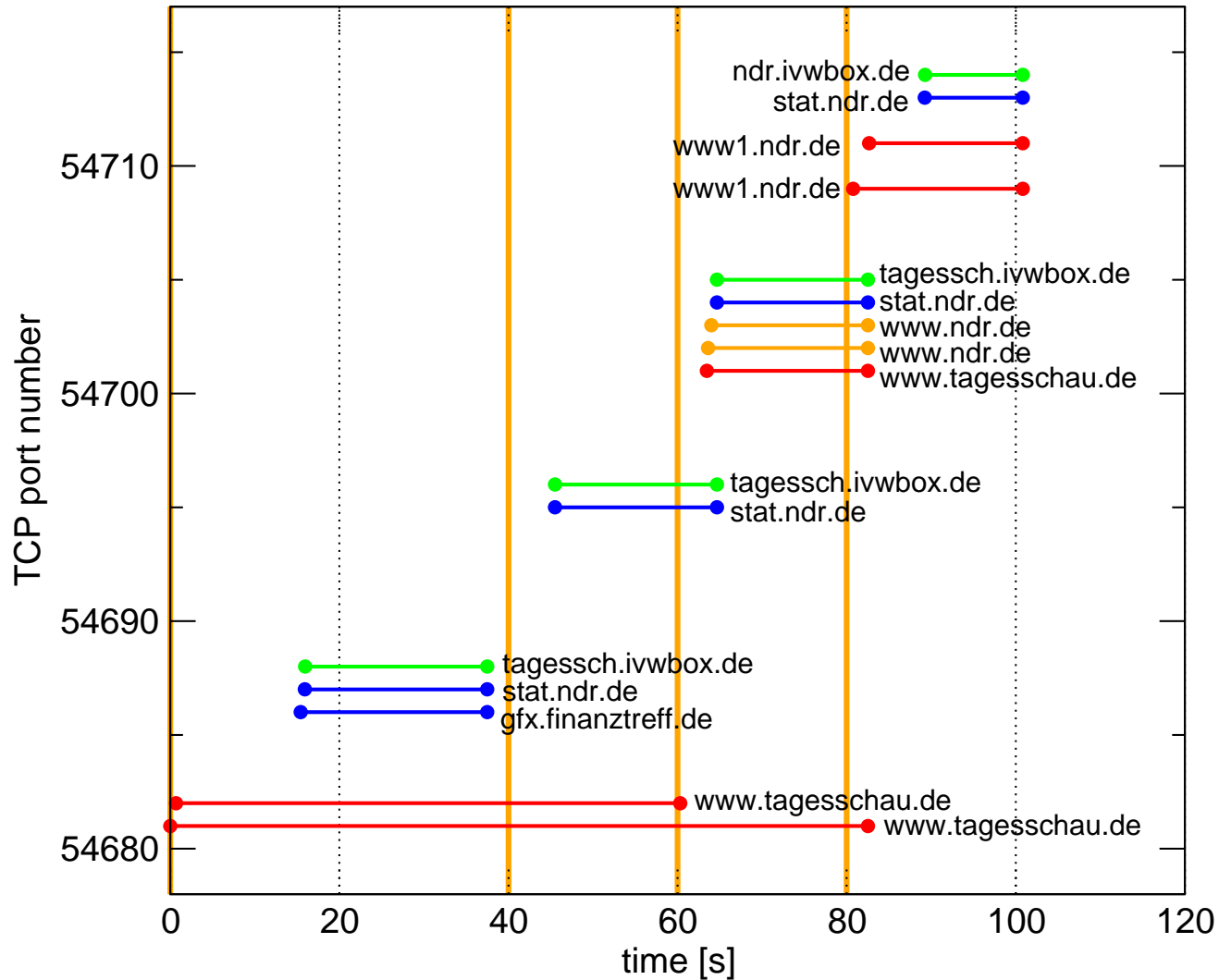
Histogram of accessed servers during one cycle



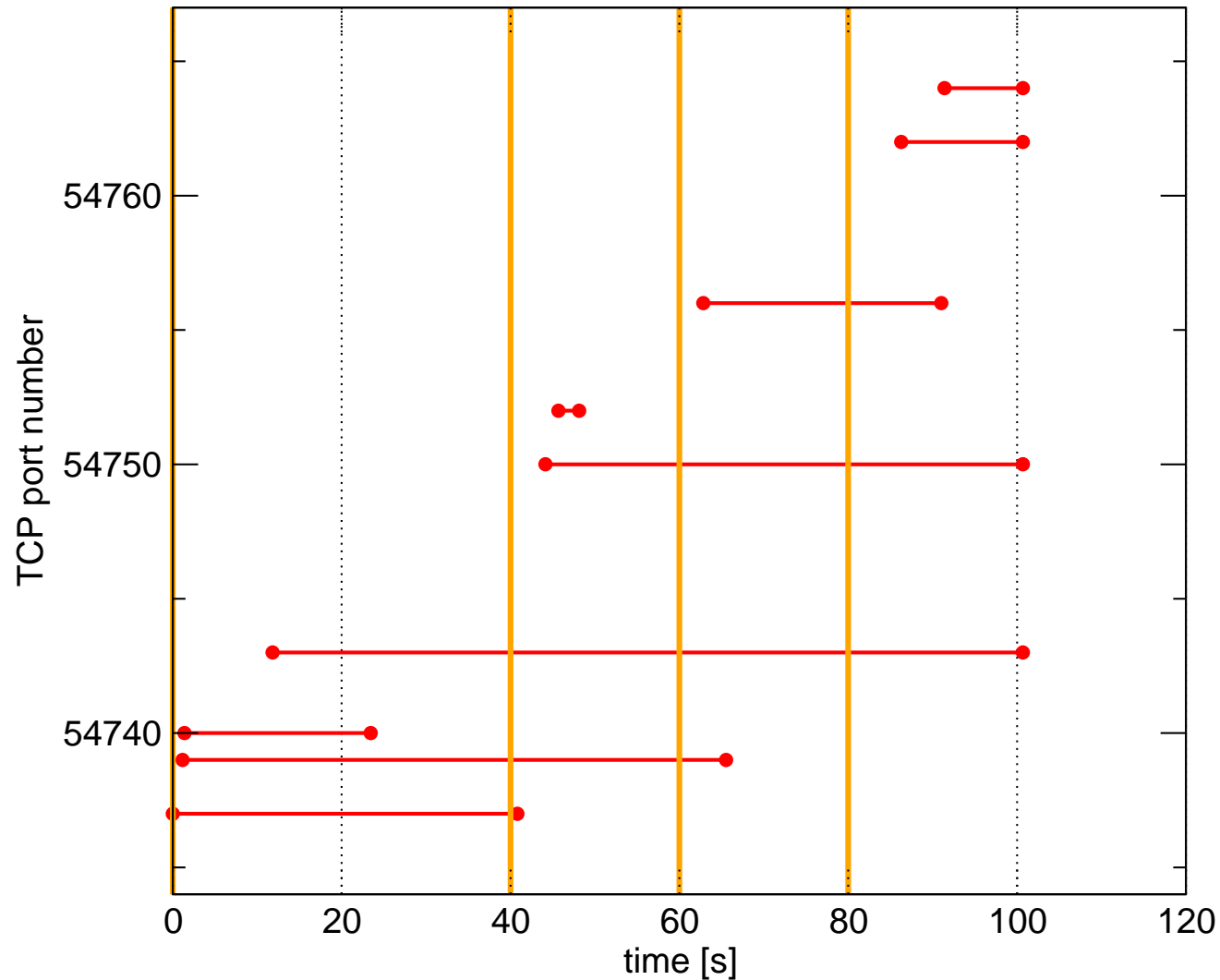
HTTP 1.1, no keep-alive, no proxy



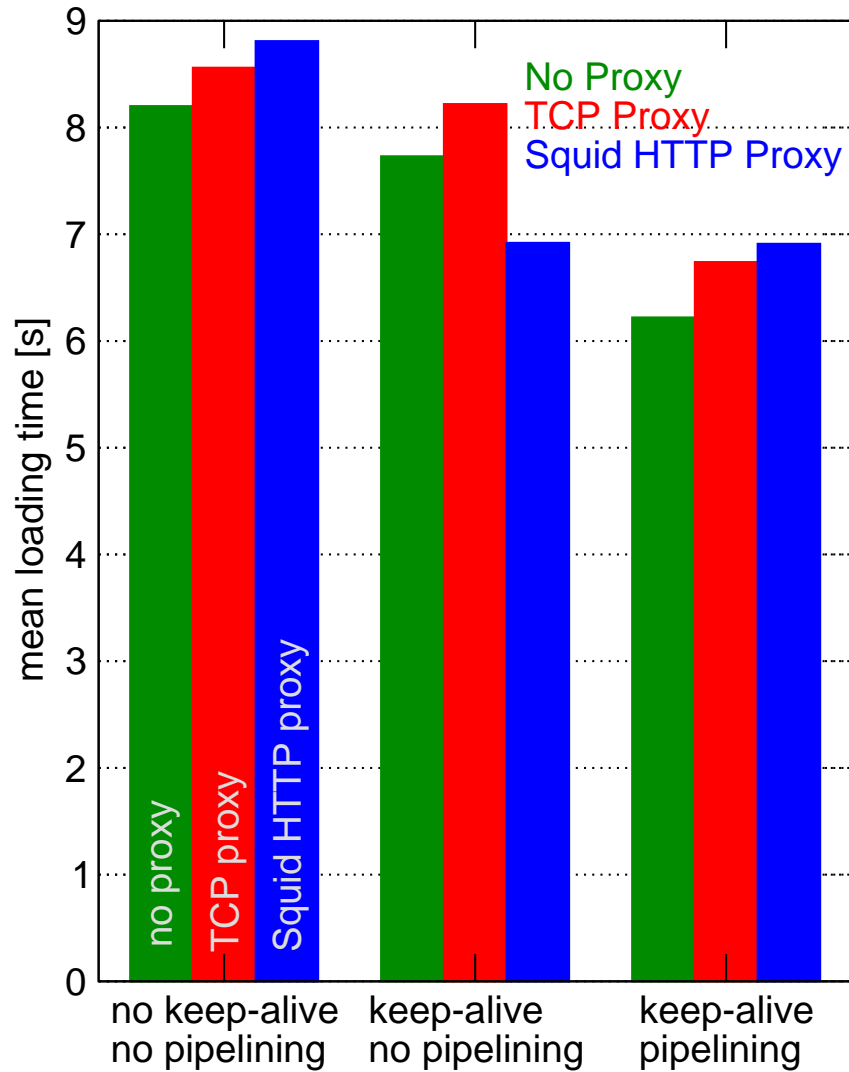
HTTP 1.1, keep-alive, no proxy



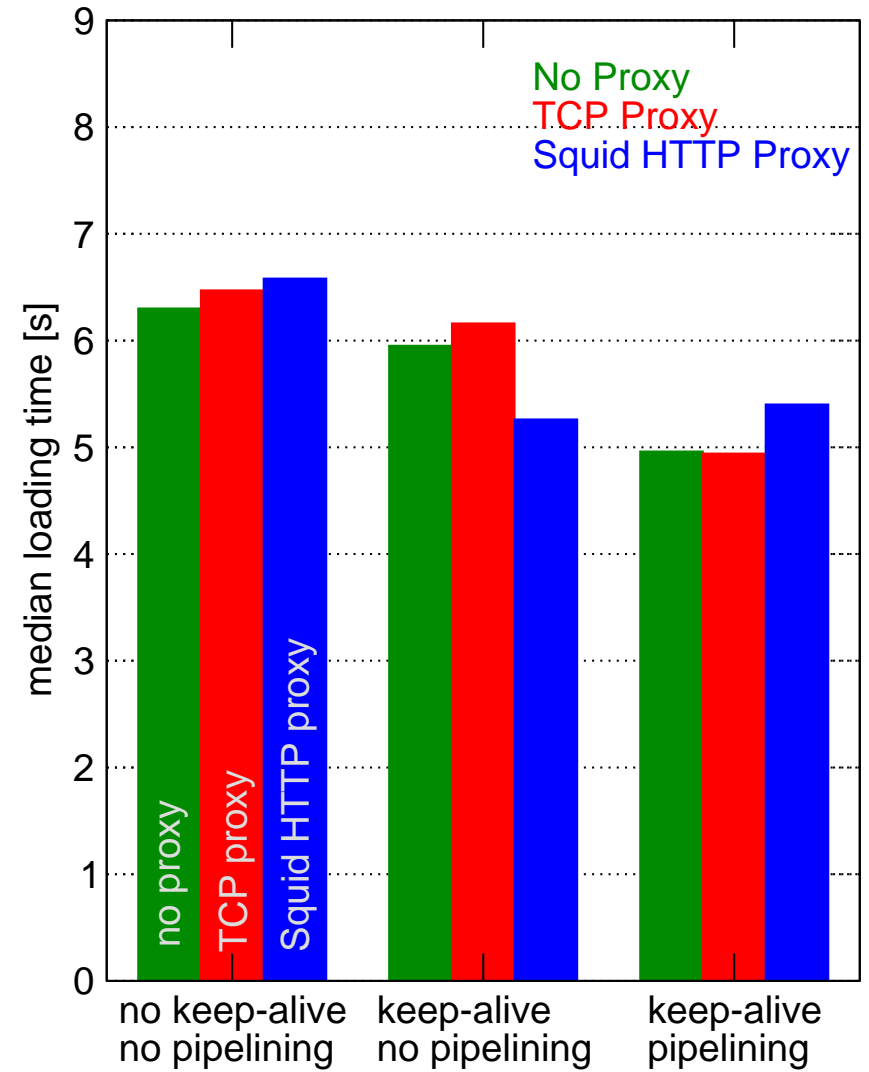
HTTP 1.1, keep-alive, Squid HTTP proxy



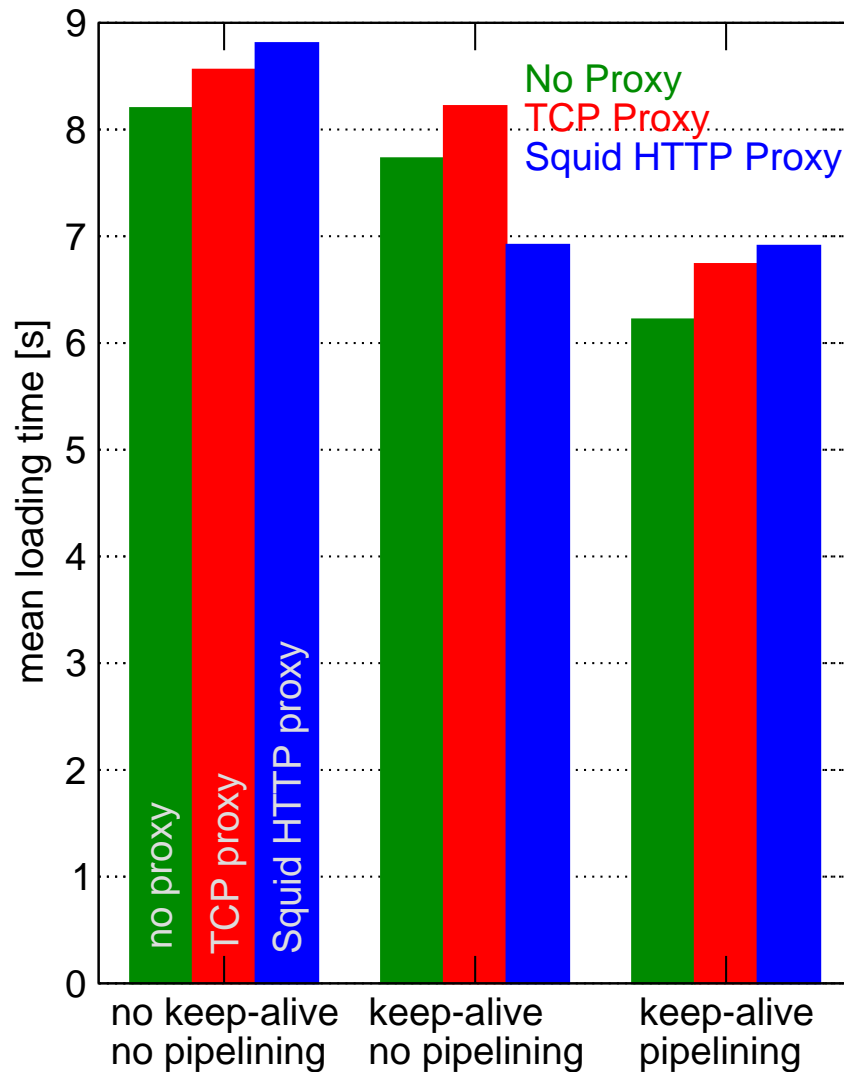
Mean



Median



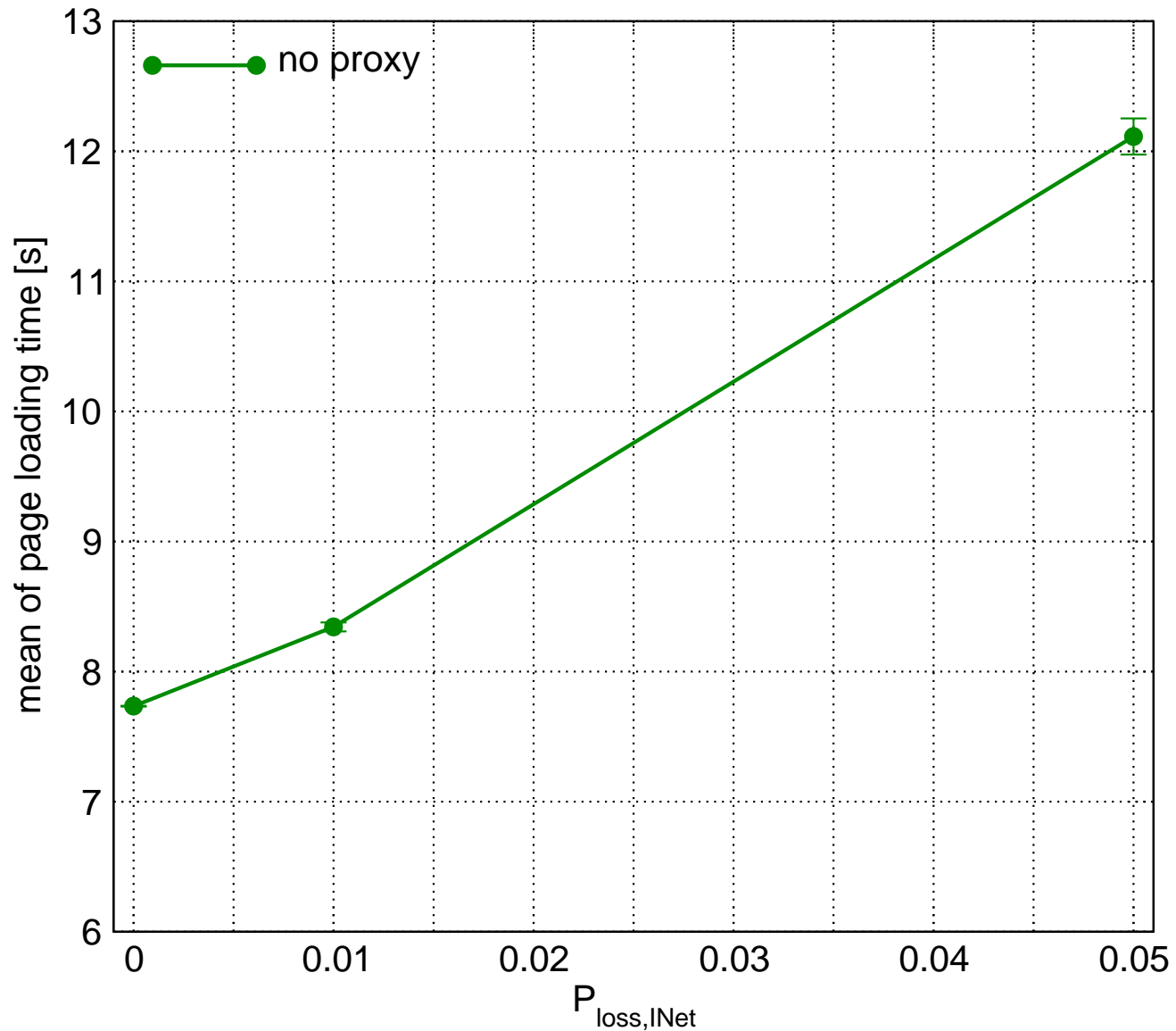
Mean



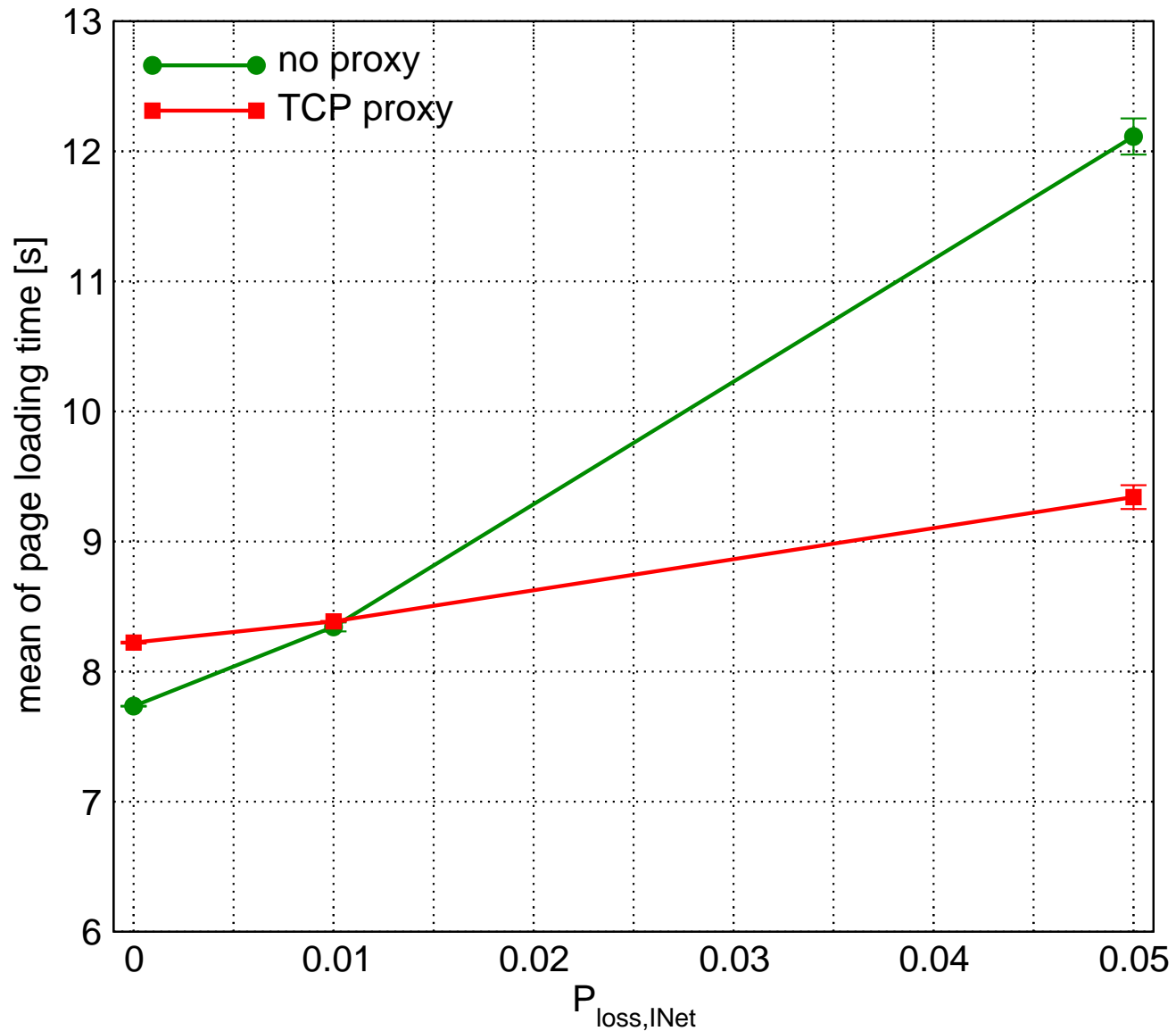
Observations for

$P_{\text{loss,INet}}=0$ and $T_{\text{INet}}=20\text{ms}$

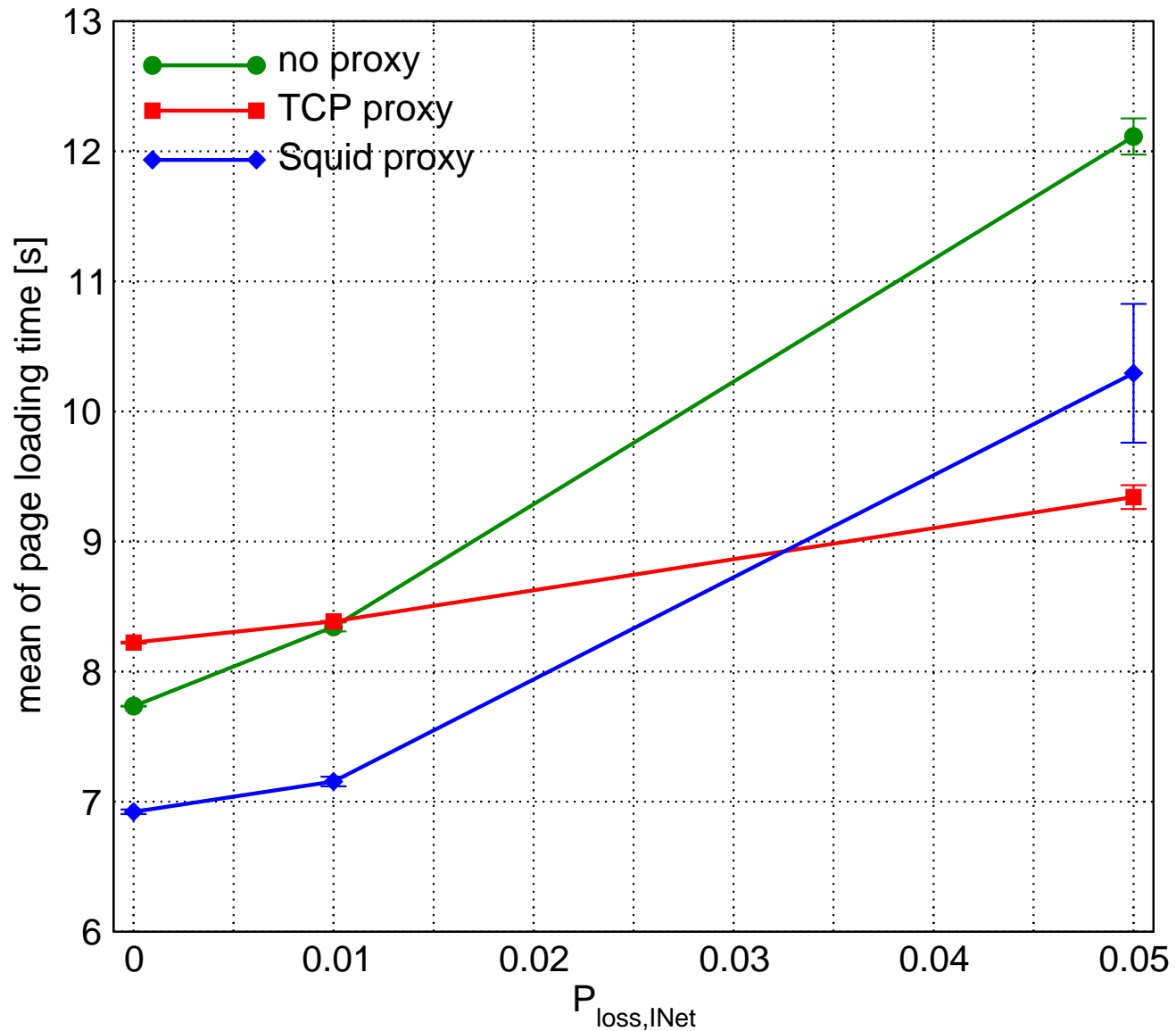
- keep-alive and pipelining always delivers best performance
- A TCP proxy worsens the performance
- An HTTP proxy can effectively utilize persistent connections
- The Squid HTTP proxy does not support pipelining very well



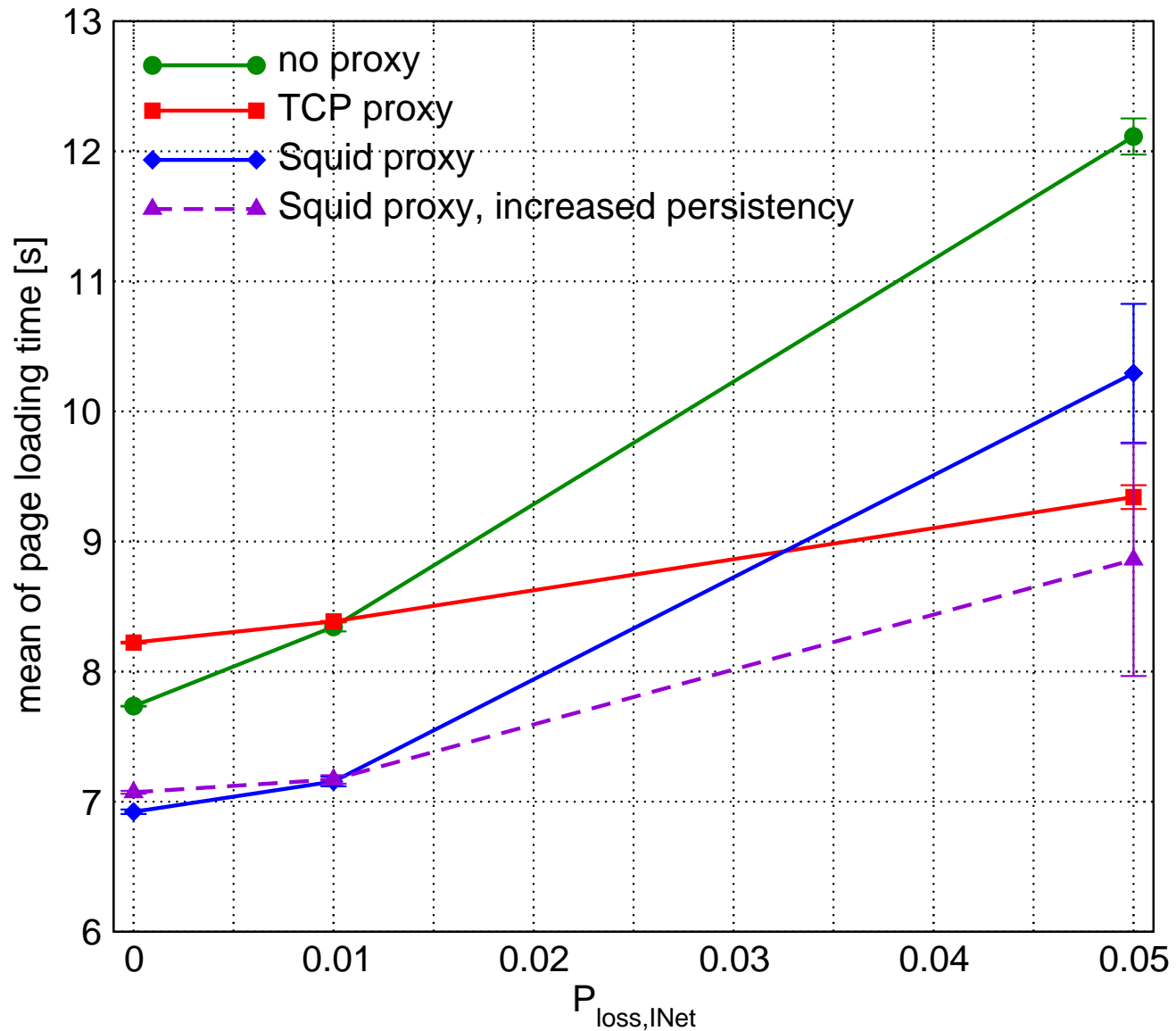
- **keep-alive**
- **no pipelining**



- keep-alive
- no pipelining



- keep-alive
- no pipelining



- keep-alive
- no pipelining

Conclusion

- **TCP proxy**
 - is advantageous especially if the fixed network part is non-optimal
 - may be disadvantageous if fixed network part behaves well
- **HTTP proxy**
 - is usually advantageous with persistent connections
 - may be disadvantageous if protocol mechanisms are not supported (e.g. pipelining)
- **Performance highly depends on configuration of components**
 - HTTP keep-alive is essential for good performance
 - HTTP pipelining greatly improves performance for pages with many inline objects

Future Work

- **Optimized TCP stack within the TCP proxy**
- **Extend studies to HSDPA, including cross-traffic**