ModNet: A modular approach to network stack extension

Sharvanath Pathak

Vivek S. Pai

Princeton University

Changing Trends in Networking

- Use of TCP for multimedia streaming
  - Netflix + YouTube > 50% internet traffic at peak [Global Internet phenomena report]
- Tremendous growth of mobile devices
  - Wireless/mobile > wired traffic by 2018 [Cisco Visual Networking Index]

TCP Stack Limitations

- Little **explicit** feedback to applications
- Applications rely on **implicit** feedback
  - BW estimate via connection/request latency
- Hinders rapid content adaptation

TCP Stack Limitations

- Large buffers for **performance** (128kB-1MB)
  - Hides BWD-product, scheduler delays
- Slow to react to network hiccups
  - Large buffers **drain slowly**
  - Especially when BW drops

TCP Stack Limitations

- TCP stack modification
  - Want: better management of socket contents, timing and parameters
- Changing OS/stack/protocol difficult
  - Adoption: chicken & egg problem
  - Deployment: middlebox problems

ModNet Overview

- Loosen the boundary between the OS and network applications
- Stick with TCP for practicality
- Provide opportunity for low-latency reaction

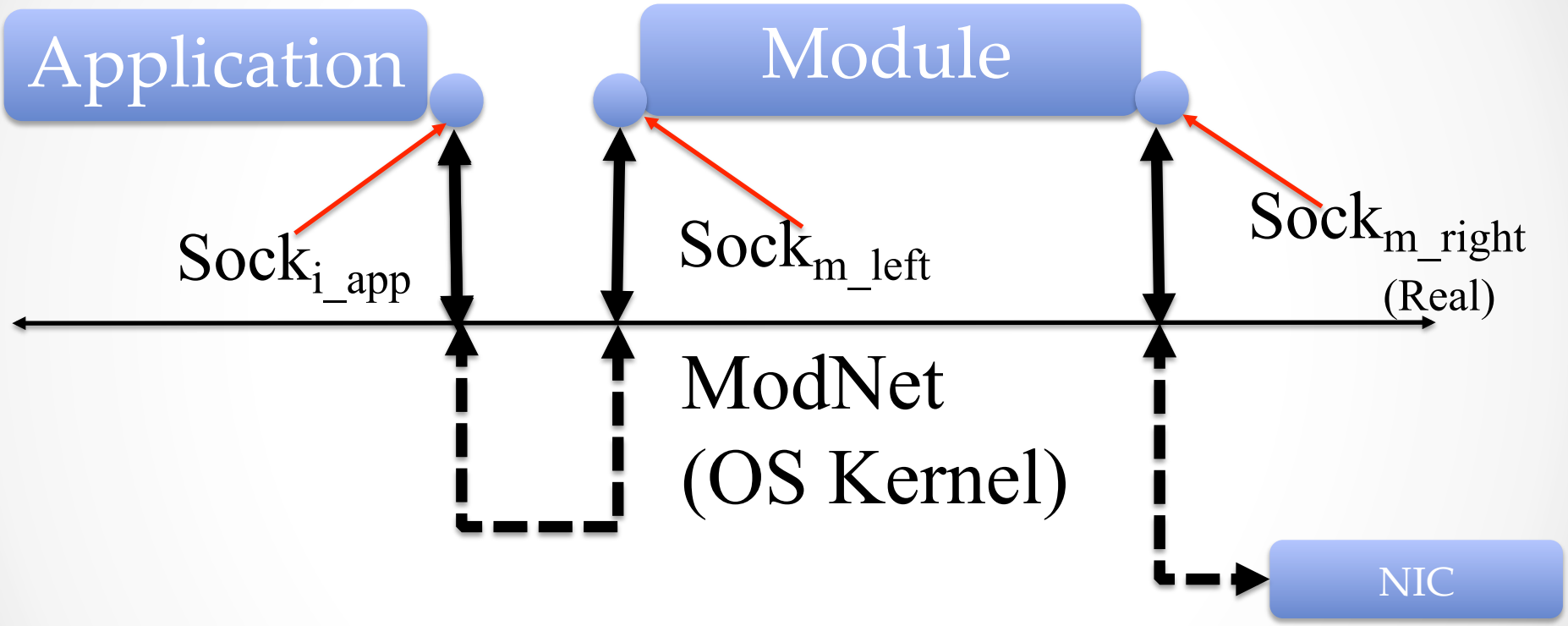
ModNet Techniques

- Delegation
  - Network modules for customizing TCP stack
- Inspection
  - Interfaces for exposing socket state
- Revocation
  - Modifying unsent socket buffer content

Delegation

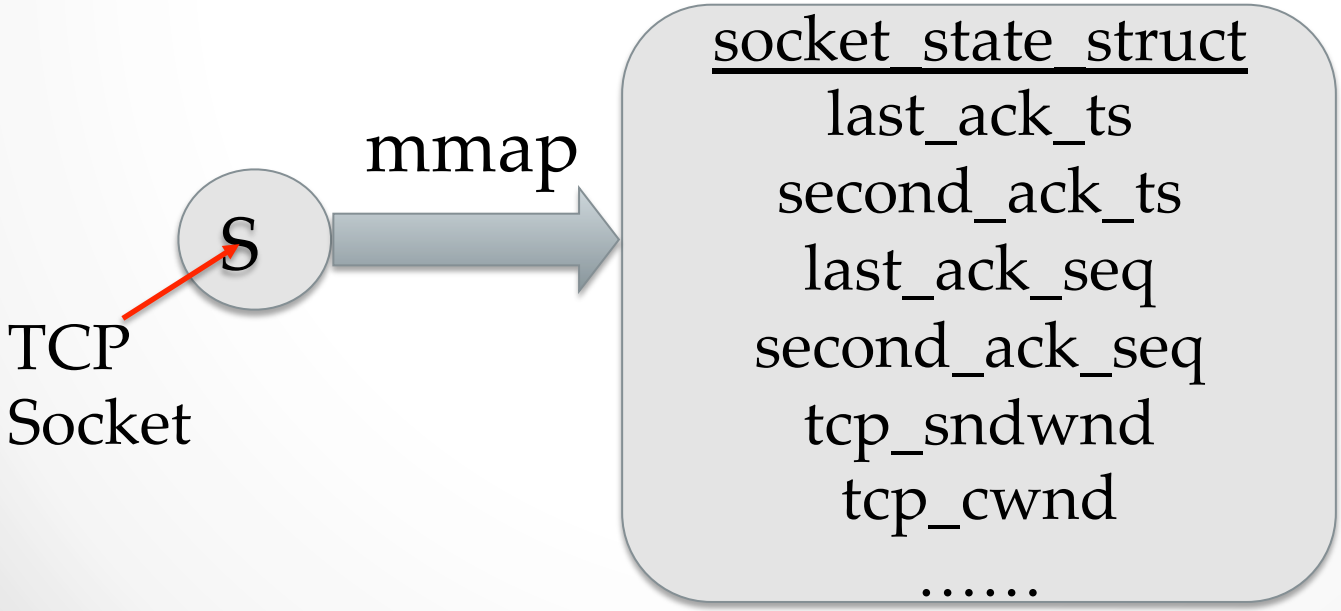
- Allow sockets to be intercepted by one or more user-level modules
  - OS-like functionalities, e.g. socket buffer swapping
  - Reusable across applications, e.g. adaptive HTTP compression
  - Composable modules

Socket Stealing



Inspection

- Memory-mapped socket state
  - **mmap (socket_fd, …)**
  - Low cost: atomic counts & shared memory
  - Safe access: shadowed values



Revocation

- Allow applications to inspect or yank existing socket buffer data.

  modnet_yank (int socket_fd, char * buffer, int length, …)

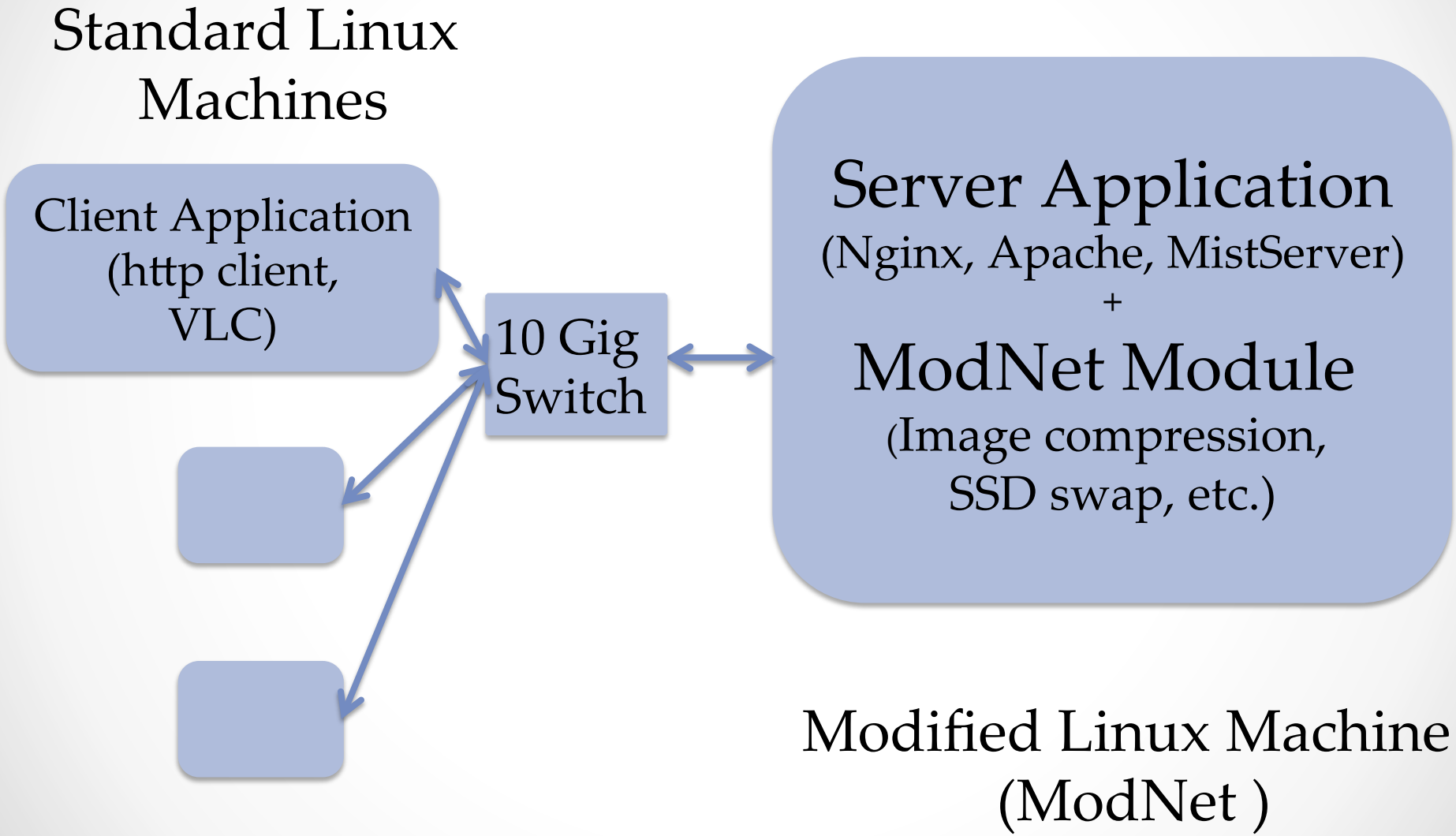
- Affects only unsent (but OS-buffered) content

- Normal mode: large buffers for **performance**
  Problems: yank & replace for **fast reaction**

Experiments

- Quantify the module overheads
- Evaluate the end-to-end performance of some interesting modules
- Evaluate the utility of revocation for more reactive video streaming

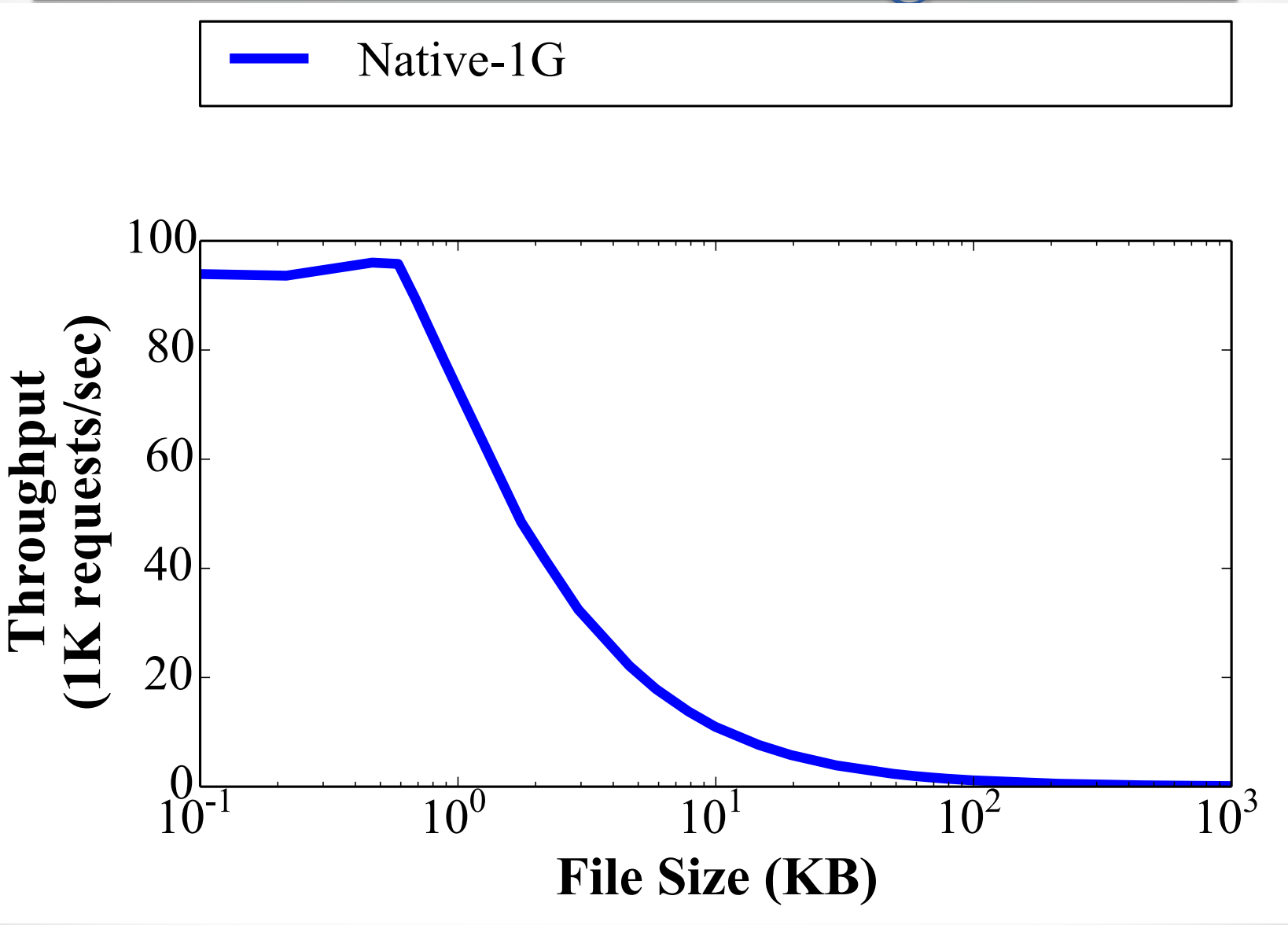
Experimental Setup



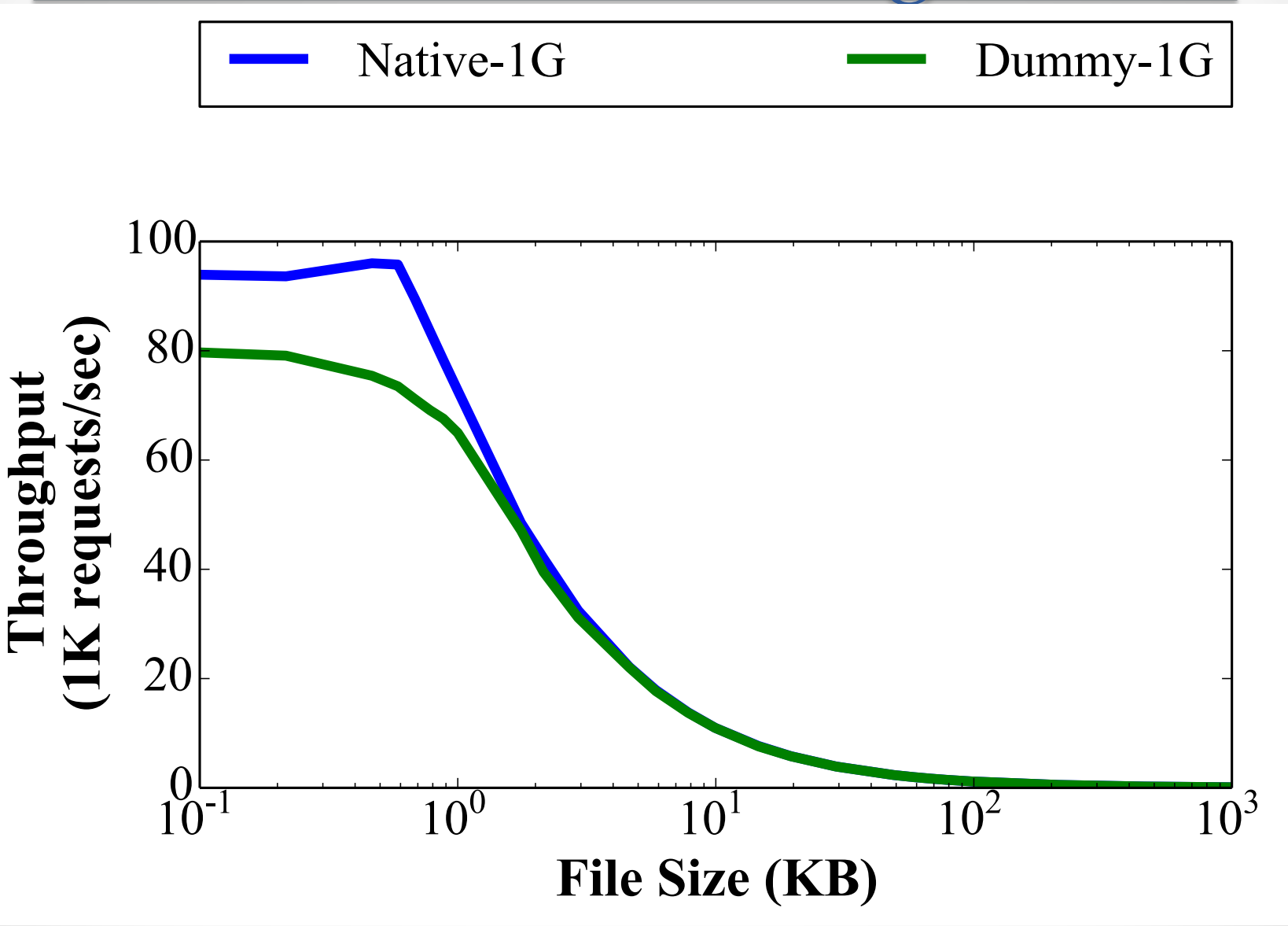
Overheads of Delegation

- Web server benchmark
- Nginx with and without dummy module
- 400 concurrent connections from 2 client machines

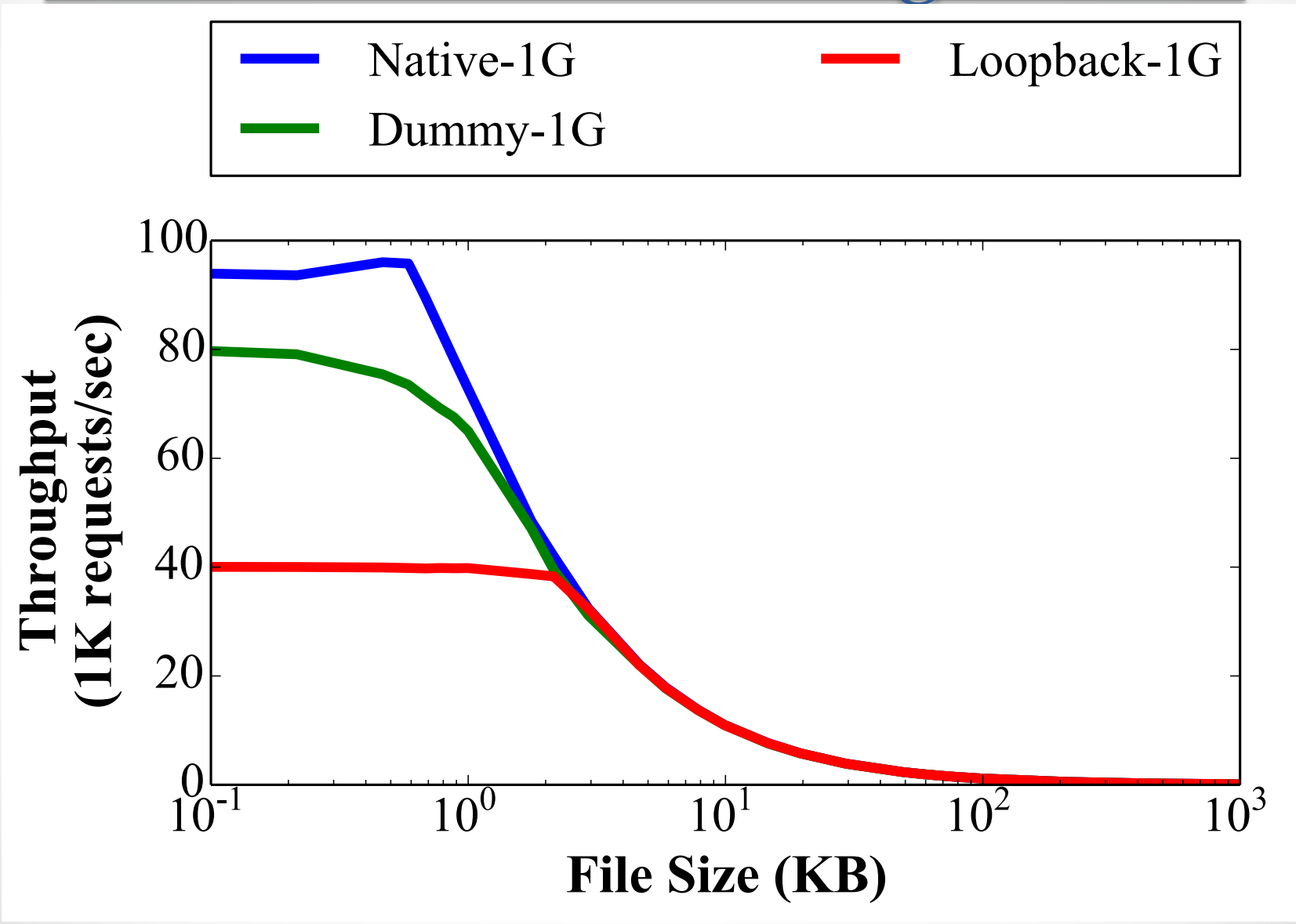
Overheads of Delegation



Overheads of Delegation



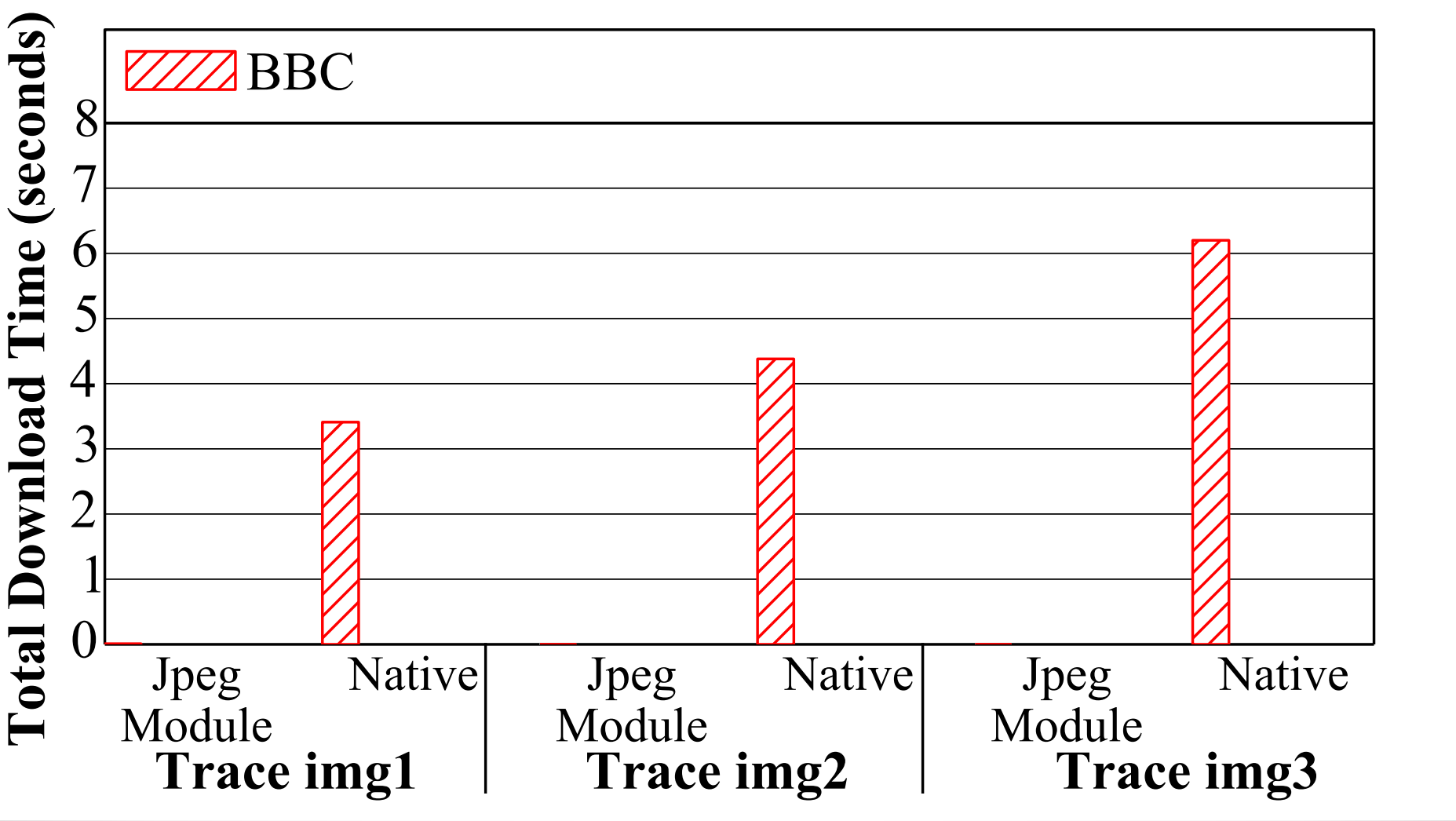
Overheads of Delegation



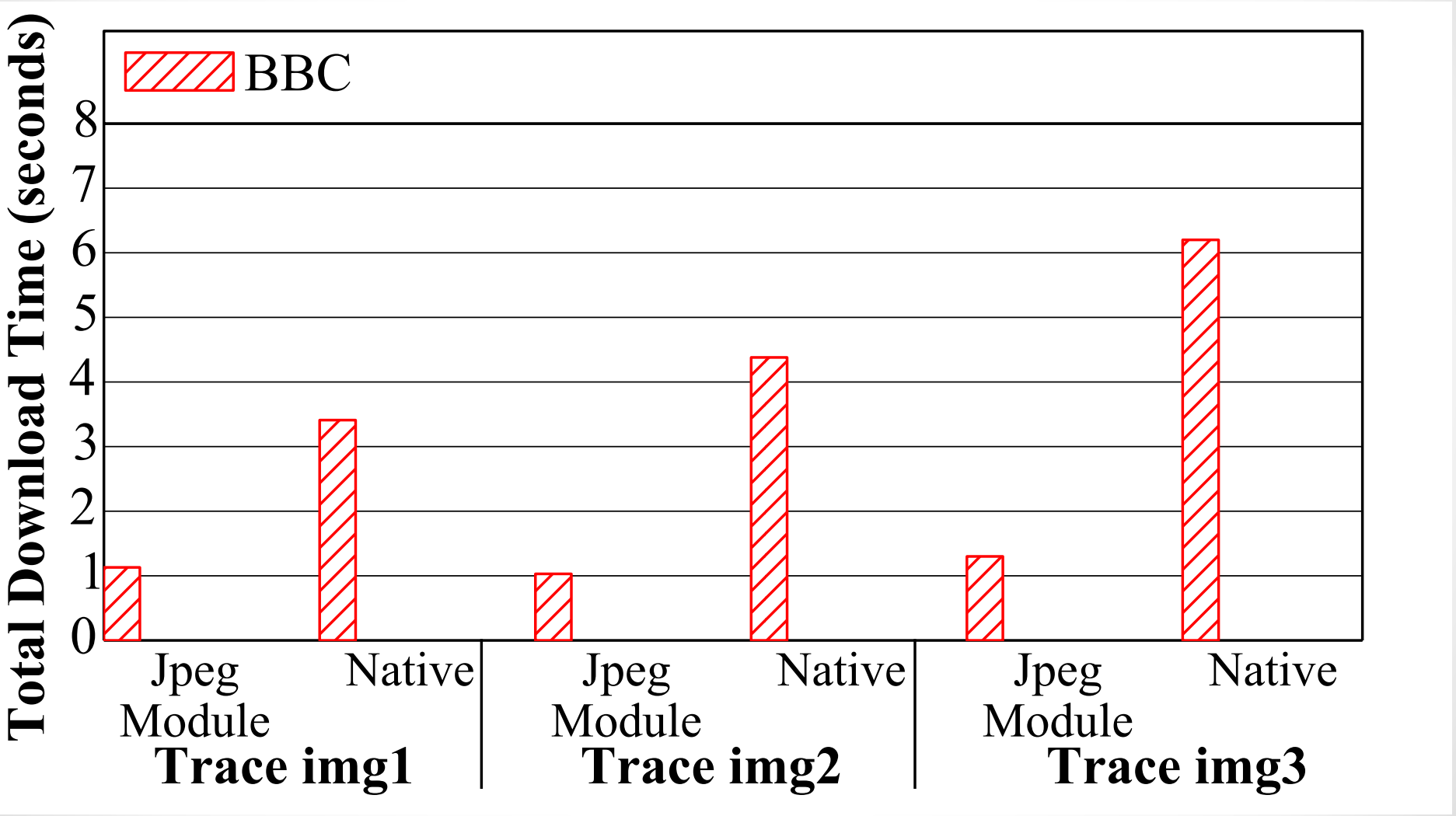
Adaptive Image Compression

- Application-oblivious HTTP Jpeg compression (delegation)
- Passive bandwidth estimation (inspection)
- New dynamic Jpeg compression scheme
  - Adjust DCT coefficients mid-transmission
  - No client-side changes
- Similar experimental setup as last one

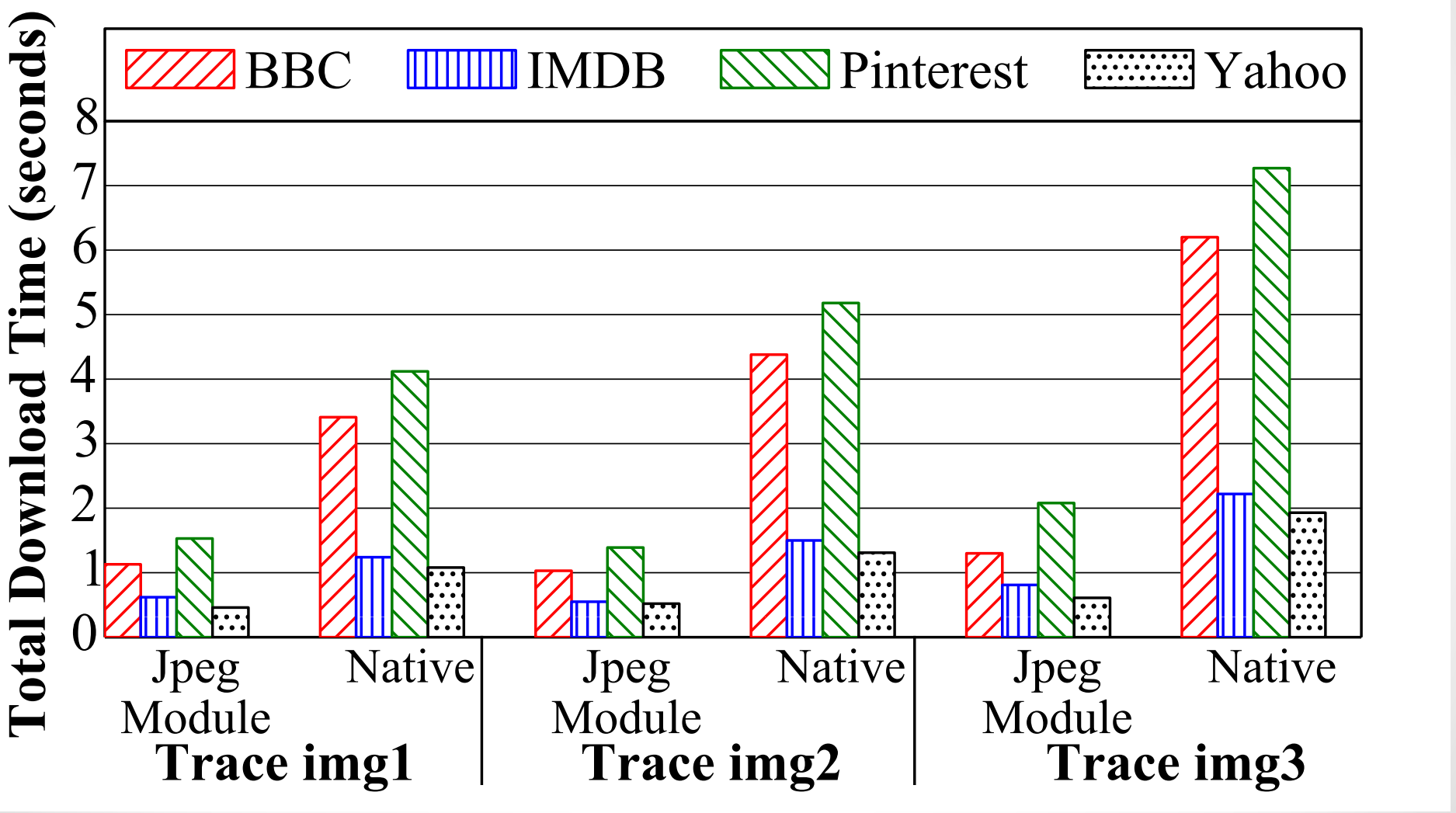
Jpeg module (3G network traces)



Jpeg module (3G network traces)



Jpeg module (3G network traces)



Swappable Socket Buffers

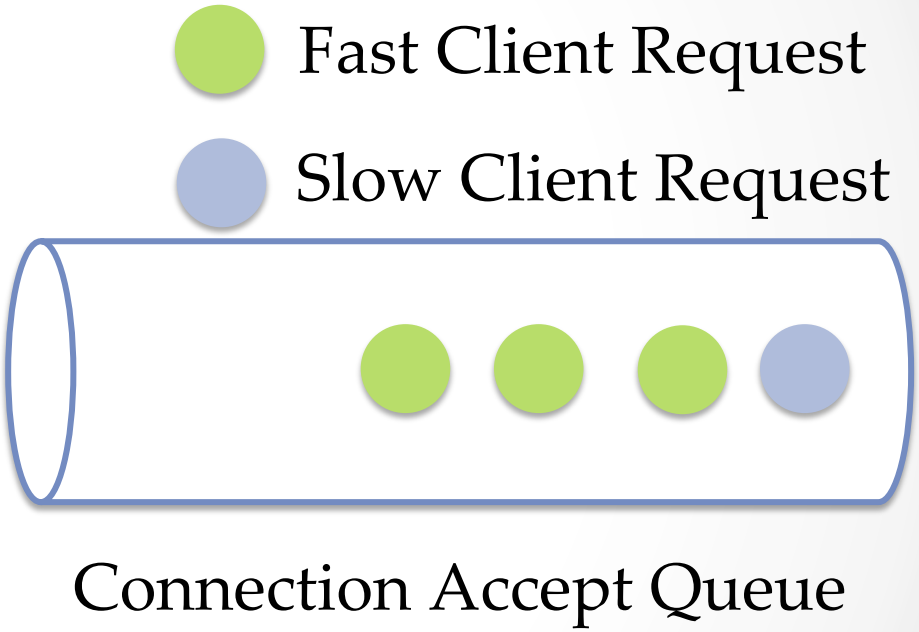
- Dynamic content generation

Dynamic Content Server

THREAD 1 BLOCKED on SEND!

THREAD 2 BLOCKED on SEND!

THREAD 3 BLOCKED on SEND!



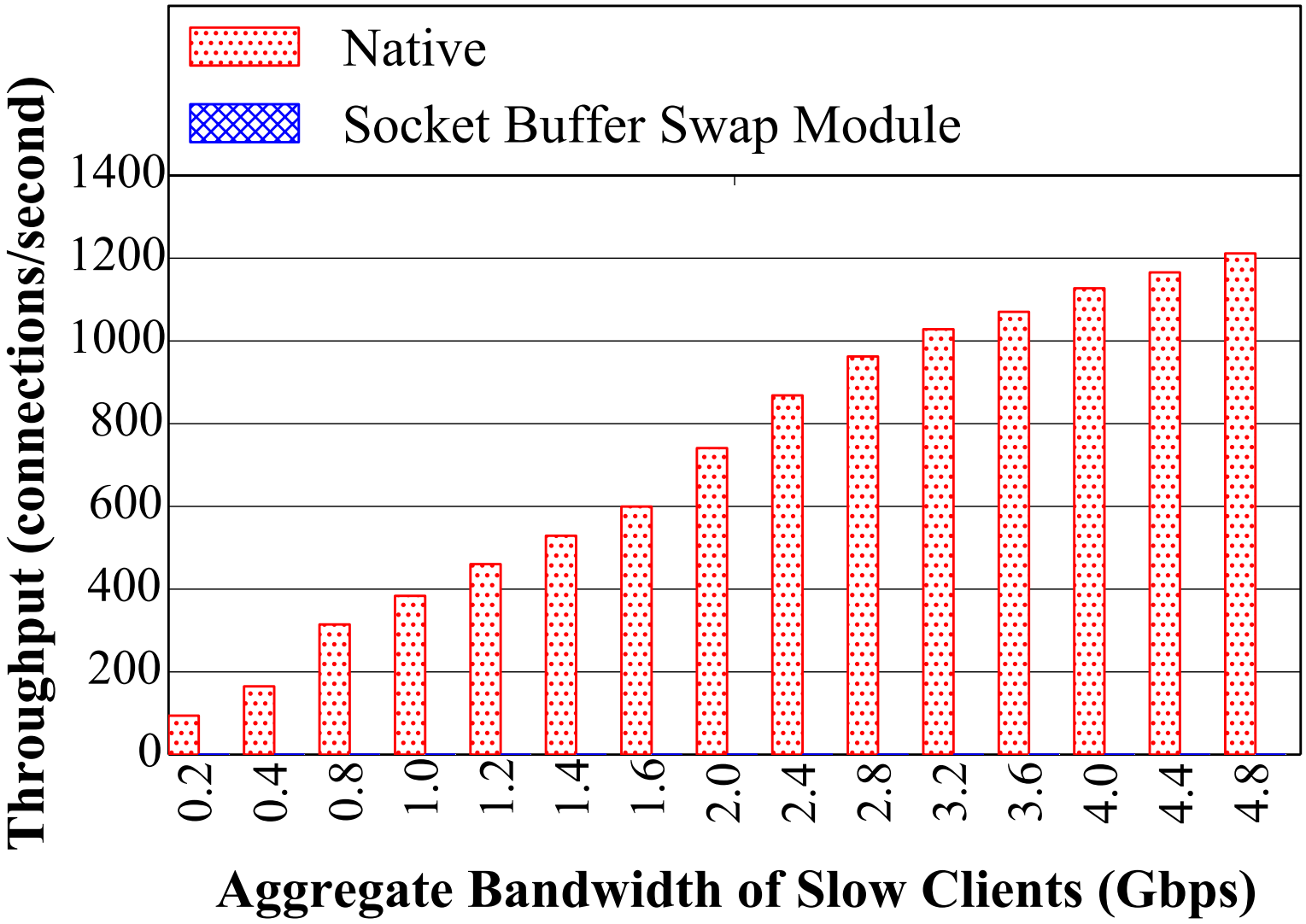
Swappable Socket Buffers

- Dynamic content generation
  - Flash crowd ties up all threads/processes
  - Server crawls, despite available bandwidth
- Socket buffer swap module
  - Swapping slow buffers to SSDs
- Dynamic file download with mixed workload

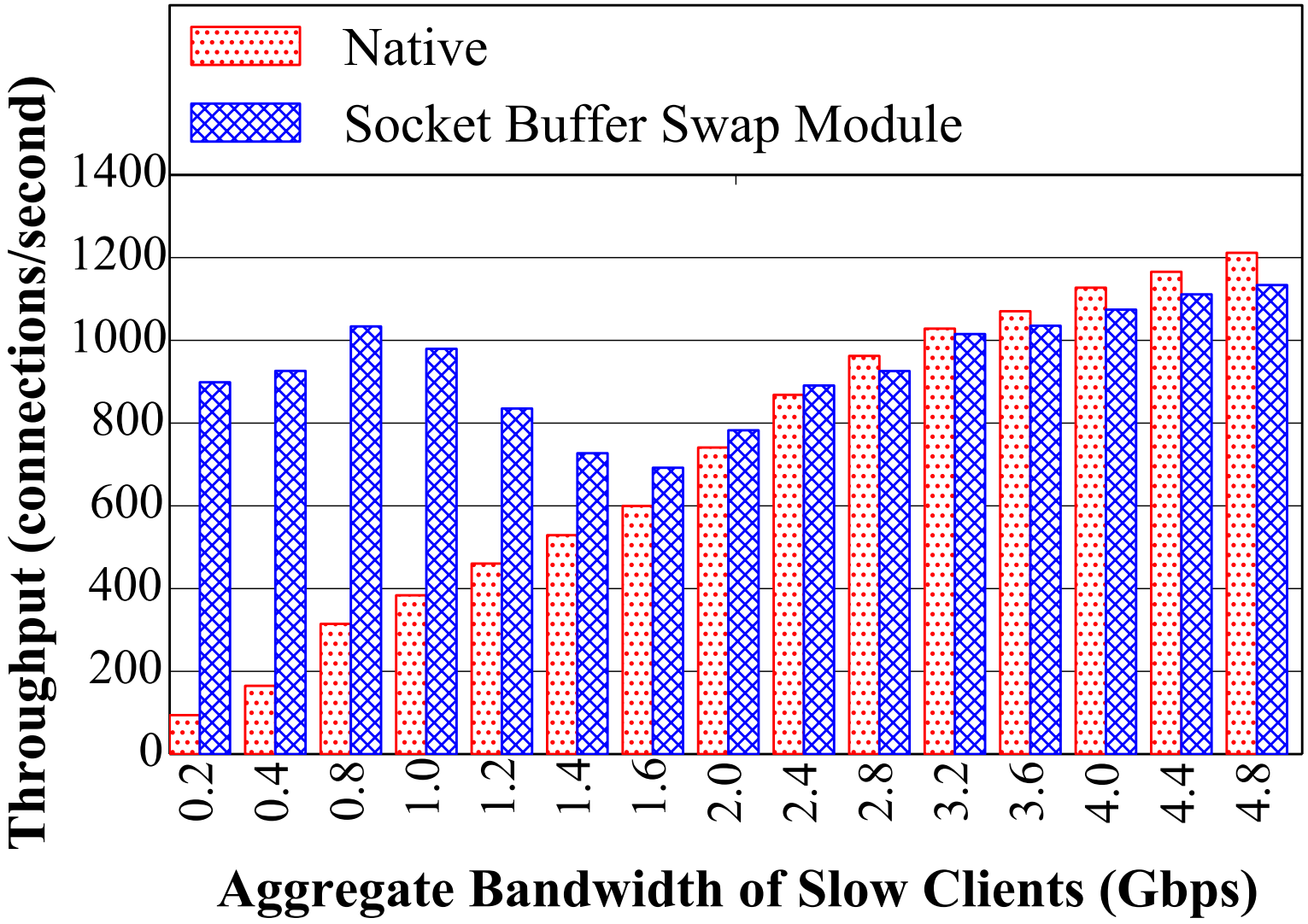
Swappable Socket Buffers

- OS-like functionalities in user space
  - **Delegation** for appropriate privileges, resource limits and scheduling
- **Inspection** - decide when to offload
- **Revocation** – yank slow content

Socket buffer swap module



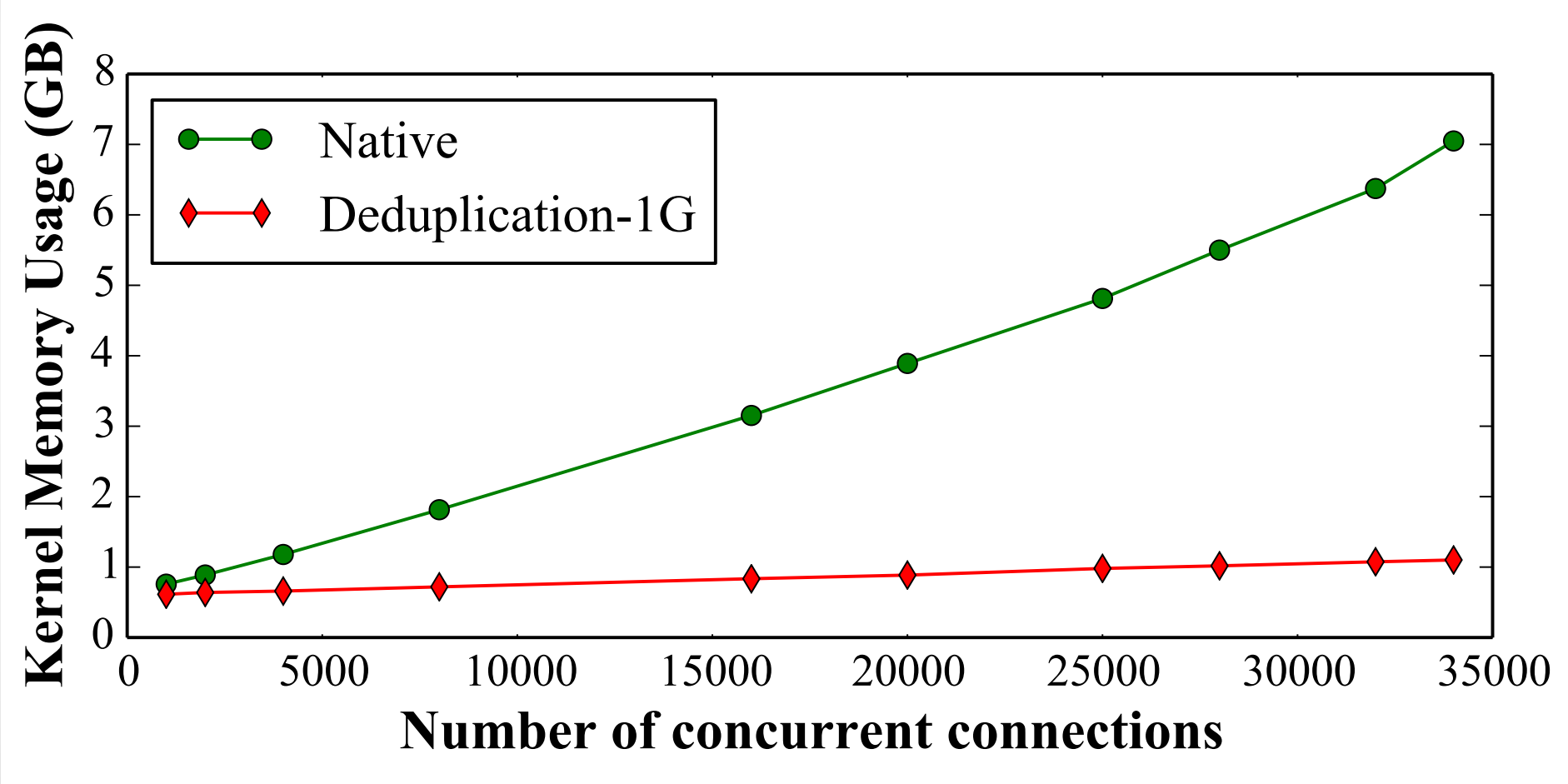
Socket buffer swap module



Deduplicating Socket Buffers

- Templated dynamic web pages
  - Lots of near-duplicate content
- Reduce memory pressure via dedup
- Utilize the spare CPU cycles as opposed to spare SSD bandwidth in SSD swap module

Deduplication Module



Modified HTTP live streaming

HLS (HTTP live streaming) protocol

- When BW drops, buffered data causes slowdown

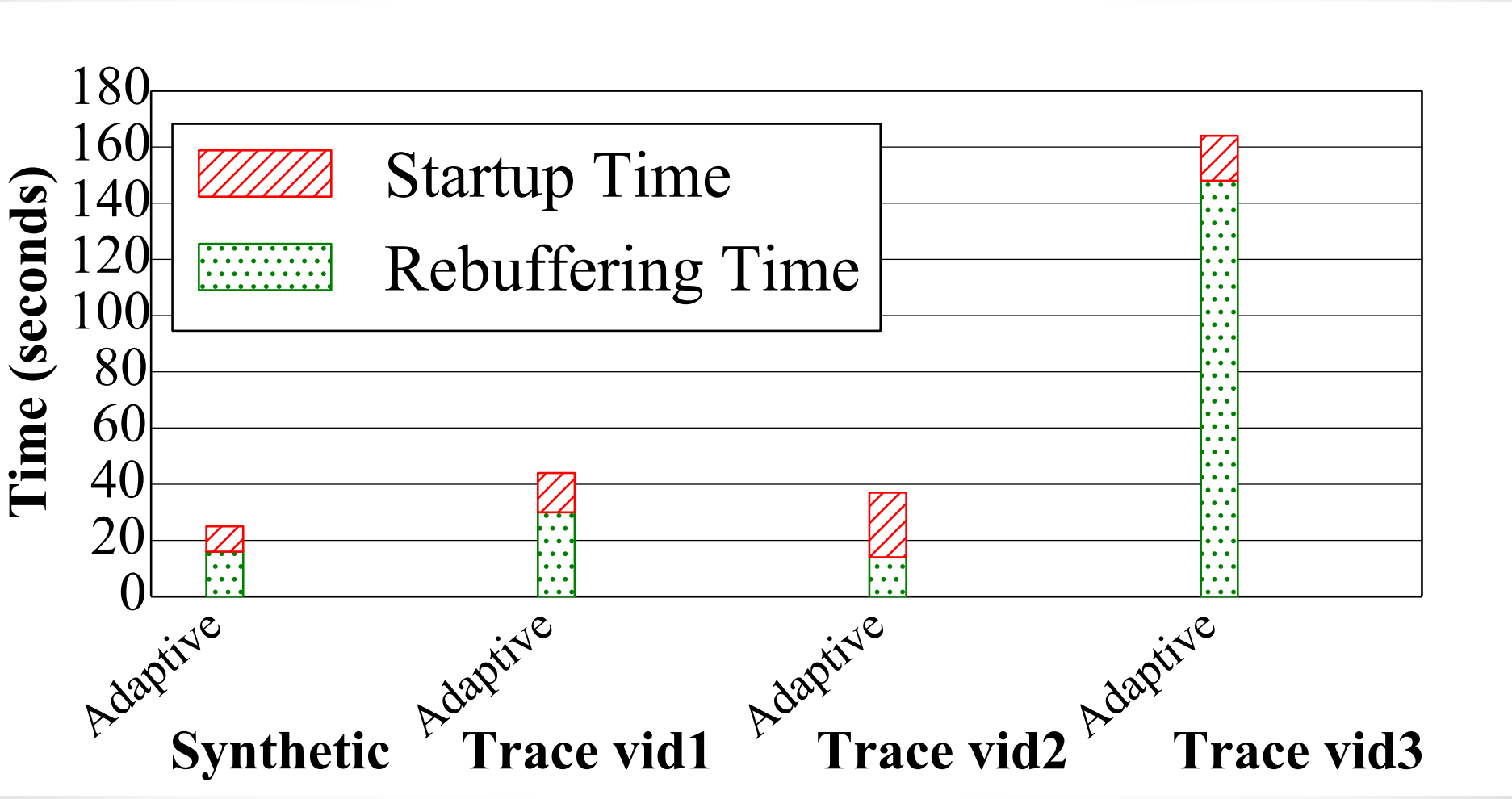
With ModNet, server observes slowdown & reacts as well

- Truncation mechanism
- Faster reaction using yank

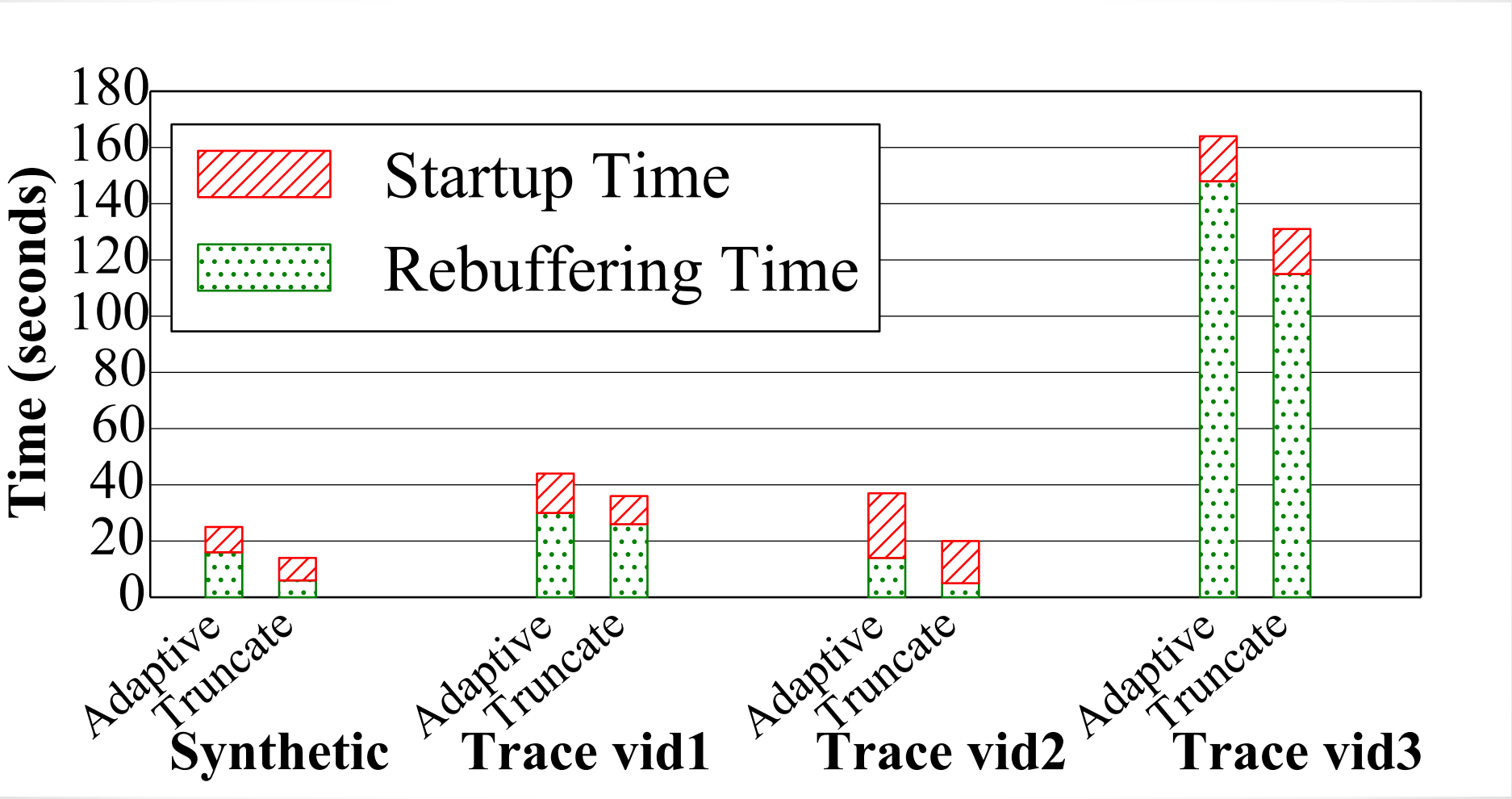
Modified HTTP live streaming

- **Inspection** for estimating network conditions
- **Revocation** for yanking buffer content

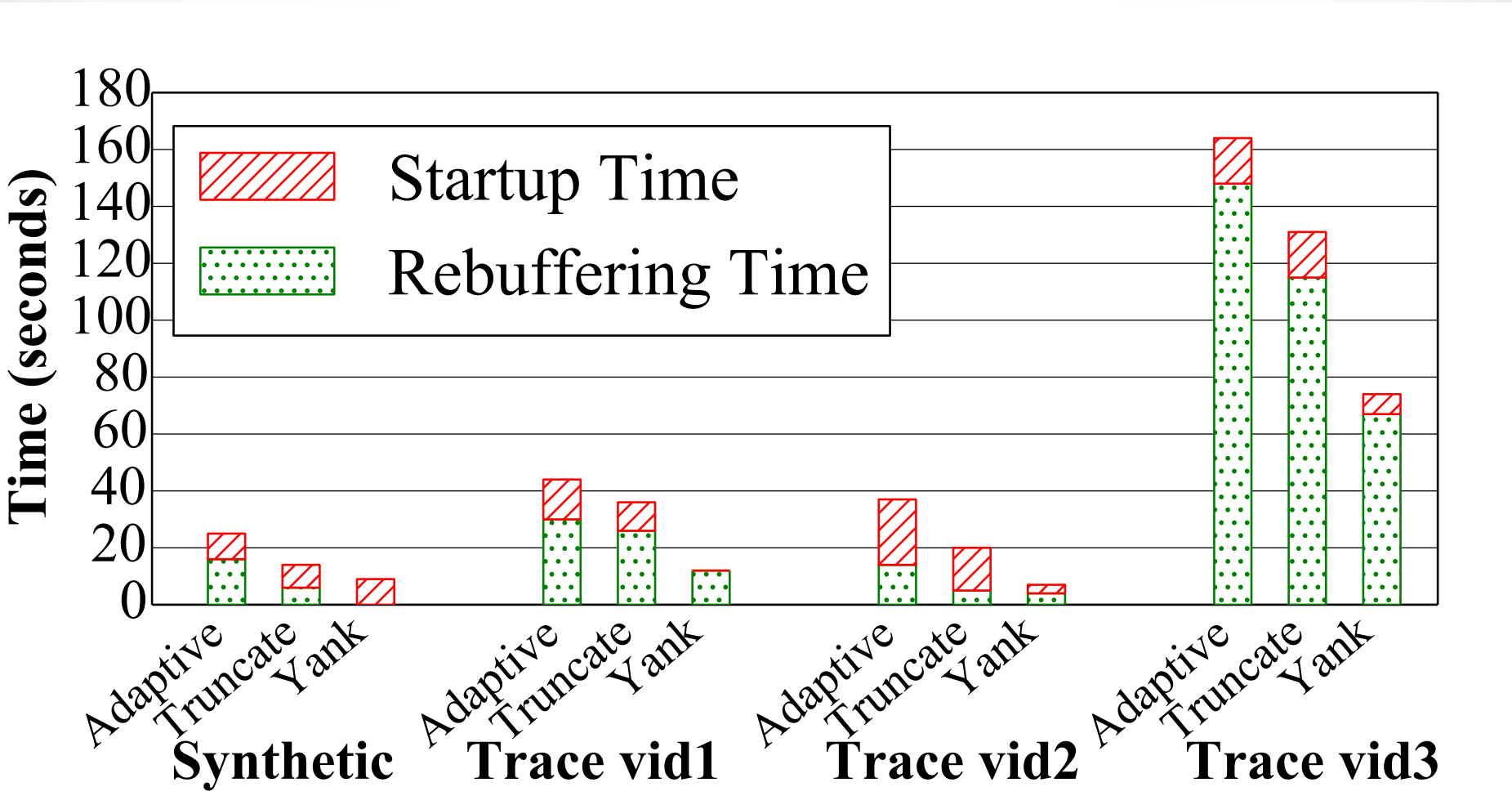
Video streaming with revocation



Video streaming with revocation



Video streaming with revocation



Conclusions

- ModNet enhances OS network API
- Delegation
  - Eases implementation and deployment of network stack extensions
- Inspection and Revocation
  - Better control and insight for applications and modules
- New interesting modules
  - Prove the utility of ModNet

Thanks

sharvanath.info/modnet
sharvanath@gmail.com