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
  o Netflix + YouTube > 50% internet traffic at peak [Global Internet phenomena report]

• Tremendous growth of mobile devices
  o Wireless/mobile > wired traffic by 2018 [Cisco Visual Networking Index]
• Little *explicit* feedback to applications

• Applications rely on *implicit* feedback
  o 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
  o Want: better management of socket contents, timing and parameters

• Changing OS/stack/protocol difficult
  o Adoption: chicken & egg problem
  o 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

Application

Sock\textsubscript{i\_app}

Module

Sock\textsubscript{m\_left}

ModNet

(OS Kernel)

Sock\textsubscript{m\_right}

(Real)

NIC
Inspection

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

```
socket_state_struct
  last_ack_ts
  second_ack_ts
  last_ack_seq
  second_ack_seq
  tcp_sndwnd
  tcp_cwnd
  ...
```

TCP Socket

mmap

S
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

Standard Linux Machines

Client Application (http client, VLC)

10 Gig Switch

Server Application (Nginx, Apache, MistServer) + ModNet Module (Image compression, SSD swap, etc.)

Modified Linux Machine (ModNet)
Overheads of Delegation

• Web server benchmark
• Nginx with and without dummy module
• 400 concurrent connections from 2 client machines
Overheads of Delegation

**Native-1G**

Graph showing throughput (1K requests/sec) vs. file size (KB).
Overheads of Delegation

- **Native-1G**
- **Dummy-1G**

**Graph:**
- **Y-axis:** Throughput (1K requests/sec)
- **X-axis:** File Size (KB)
Overheads of Delegation

- Native-1G
- Loopback-1G
- Dummy-1G

Throughput (1K requests/sec) vs. File Size (KB)
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)

<table>
<thead>
<tr>
<th>Total Download Time (seconds)</th>
<th>Jpeg Module Trace img1</th>
<th>Native</th>
<th>Jpeg Module Trace img2</th>
<th>Native</th>
<th>Jpeg Module Trace img3</th>
<th>Native</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Jpeg Module Trace img1: native time is 2 seconds
- Jpeg Module Trace img2: native time is 6 seconds
- Jpeg Module Trace img3: native time is 8 seconds

Note: The BBC row indicates that the Jpeg module trace is significantly slower than the native option for all three 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!

Fast Client Request

Slow Client Request

Connection Accept Queue
Swappable Socket Buffers

• Dynamic content generation
  o Flash crowd ties up all threads/processes
  o Server crawls, despite available bandwidth

• Socket buffer swap module
  o 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

Throughput (connections/second)

0 200 400 600 800 1000 1200 1400

Native
Socket Buffer Swap Module

Aggregate Bandwidth of Slow Clients (Gbps)

0.2 0.4 0.8 1.0 1.2 1.4 1.6 2.0 2.4 2.8 3.2 3.6 4.0 4.4 4.8
Socket buffer swap module

![Graph showing throughput (connections/second) vs. aggregate bandwidth of slow clients (Gbps)](image-url)

- **Native**
- **Socket Buffer Swap Module**
Deduplicating Socket Buffers

• Templated dynamic web pages
  o 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

![Graph showing Kernel Memory Usage (GB) vs. Number of concurrent connections]

- **Native**
- **Deduplication-1G**

The graph illustrates the memory usage for different numbers of concurrent connections. The green line represents the Native mode, while the red line represents the Deduplication-1G mode. As the number of concurrent connections increases, the memory usage for both modes shows a linear increase.
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

![Diagram showing startup time and rebuffing time for different video files]
Video streaming with revocation

- Startup Time
- Rebuffering Time

<table>
<thead>
<tr>
<th></th>
<th>Adaptive Truncate</th>
<th>Synthetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace vid1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace vid2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trace vid3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Video streaming with revocation

![Bar chart showing startup time and rebuffering time for different video streams and methods: Adaptive, Truncate, Yank. The chart compares synthetic data and trace data for vid1, vid2, and vid3.]
Conclusions

• ModNet enhances OS network API

• Delegation
  o Eases implementation and deployment of network stack extensions

• Inspection and Revocation
  o Better control and insight for applications and modules

• New interesting modules
  o Prove the utility of ModNet
Thanks
sharvanath.info/modnet
sharvanath@gmail.com