Exploring Alternative Routes Using Multipath TCP

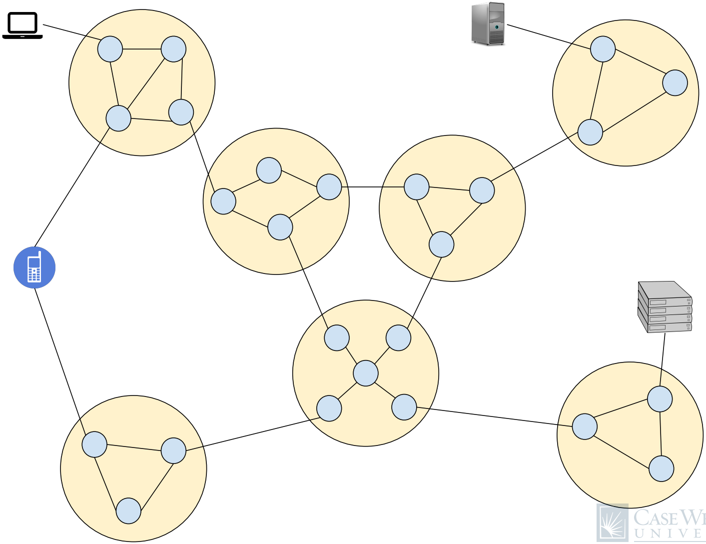
Stephen Brennan

Case Western Reserve University

June 5, 2017

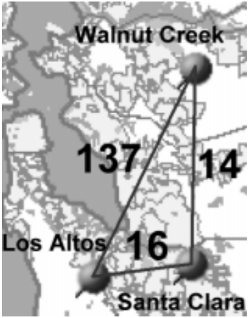
Overview

Internet Architecture



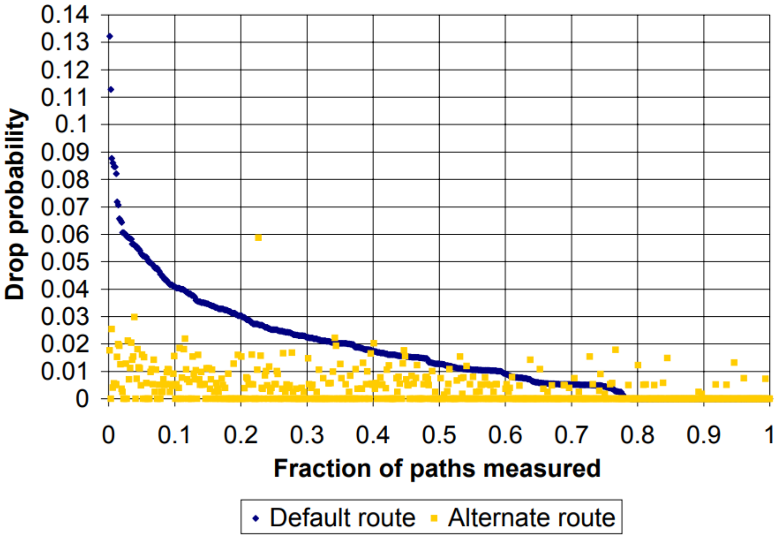
Internet Routing Inefficiencies

- The default route is not always the best, in terms of latency or reliability
- Peering agreements and policy based routing can result in suboptimal routing decisions [1]
- A route that passes through a "detour" may be better



Example of an inefficient default route [1]

[1] Savage et al. "Detour: Informed Internet routing and transport". 1999

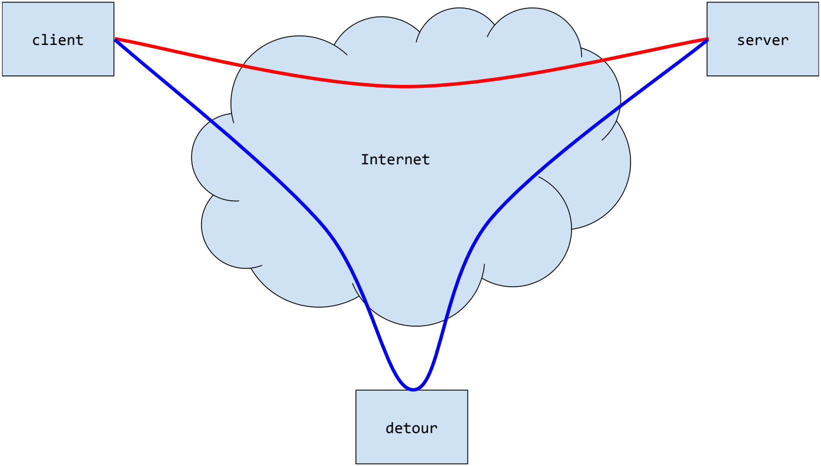

Drop probability
0.14
0.13
0.12
0.11
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Fraction of paths measured
Default route
Alternate route

Access Link Underutilization

- Residential bandwidth constantly improves
- However, residential bandwidth is not fully utilized [2]
  - Short-lived TCP sessions?
  - Anemic send buffers?
  - Network core can't support bandwidth?
- Using alternative routes can improve performance
- Aggregating multiple routes can perform even better

[2] Sargent and Allman. "Performance within a fiber-to-the-home network". 2014

Concept



Contributions

Problem: Unmodified applications cannot use detour routing to circumvent Internet routing inefficiencies.

Solution: An OS-level detour routing system that leverages Multipath TCP (MPTCP).

Contributions:

- A method for performing detour routing with unmodified applications
- A prototype implementation in the Linux kernel
- An evaluation of this mechanism on emulated networks and the Internet

Multipath TCP

- Multi-homed devices are becoming more common
  - Smartphones
  - Datacenters
  - Laptops
- TCP still views a connection as a five-tuple: (TCP, Source IP, Source port, Destination IP, Destination Port)
- Multi-homed devices are forced to choose a network interface
- Multipath TCP is an extension to TCP, allowing hosts to use multiple addresses in the same connection

Design Goals

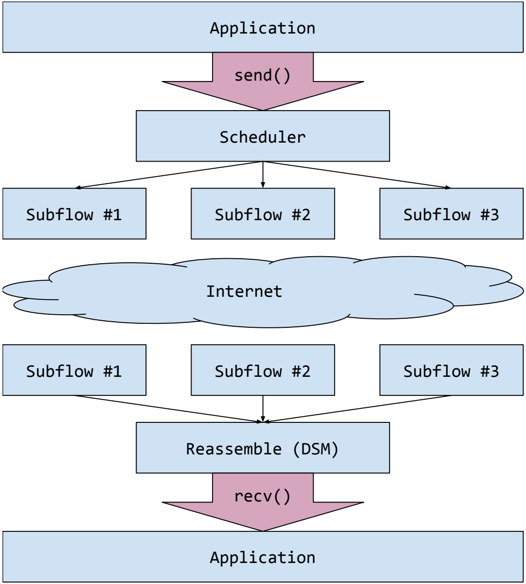
- Remain compatible with TCP applications and the Internet
  - *Present the same socket API to applications*
  - Remain similar to TCP on the wire, to remain compatible with Internet middleboxes
- Improve performance and reliability over current TCP, by aggregating paths created by multiple interfaces.
- Do no harm to single-path TCP, by taking no more bandwidth over shared bottlenecks than standard TCP would

Architecture

```
                    +-------------------------------+
                    |           Application         |
+---------------+   +-------------------------------+
|  Application  |   |             MPTCP             |
+---------------+   + - - - - - - - + - - - - - - - +
|      TCP      |   | Subflow (TCP) | Subflow (TCP) |
+---------------+   +-------------------------------+
|      IP       |   |       IP      |      IP       |
+---------------+   +-------------------------------+
```

Path Management

- Subflows are established with a three way handshake
- First subflow uses `MP_CAPABLE` option
- Subsequent subflows use `MP_JOIN` option
- Additional addresses may be advertised using `ADD_ADDR` at any time
- Either side may create new subflows at any time


Application
send()
Scheduler
Subflow #1
Subflow #2
Subflow #3
Internet
Subflow #1
Subflow #2
Subflow #3
Reassemble (DSM)
recv()
Application

Resilient Overlay Networks [6]

- Rather than use only one detour, create an overlay network
- Overlay nodes use the Internet as their "link layer"
- Routing performed at each node using measured link characteristics
- Several studies based on RON:
  - Redundant multipath routing [3]
  - "Biologically inspired" multipath routing [4]
  - mTCP [5]

[3] Andersen, Snoeren, and Balakrishnan. "Best-path vs. multi-path overlay routing". 2003

[4] Leibnitz, Wakamiya, and Murata. "Biologically inspired self-adaptive multi-path routing in overlay networks". 2006

[5] Zhang et al. "A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths." 2004

[6] Andersen et al. *Resilient overlay networks*. 2001

Application Layer

- Gnutella [7]
  - Requests forwarded via overlay network
  - Content exchanged via single path
- BitTorrent [8]
  - Pieces of content exchanged between many pairs of peers
  - Multiple paths simulate detour routing
- HTTP Range Requests [9]
  - Range requests allow requesting byte ranges of a file
  - Request from different network interfaces or to different endpoints to create alternative paths

---

[7] Adar and Huberman. "Free riding on Gnutella". 2000

[8] Cohen. "Incentives build robustness in BitTorrent". 2003

[9] Kaspar et al. "Enhancing video-on-demand playout over multiple heterogeneous access networks". 2010

Introduction

Background

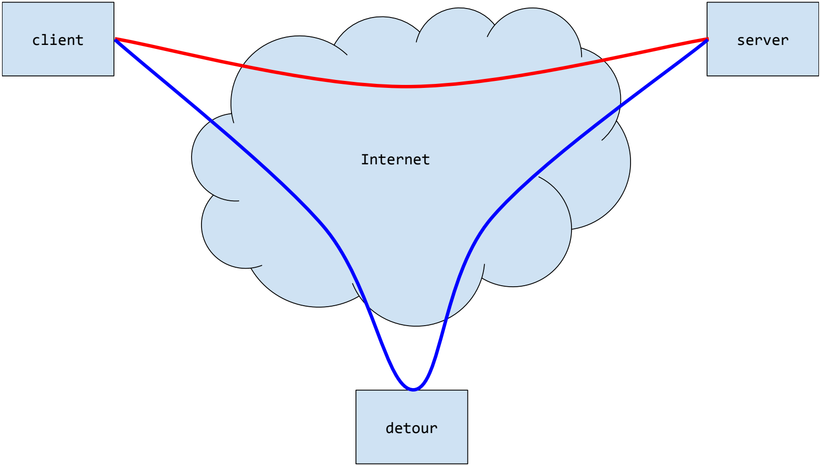
Related Work

Implementation

Evaluation

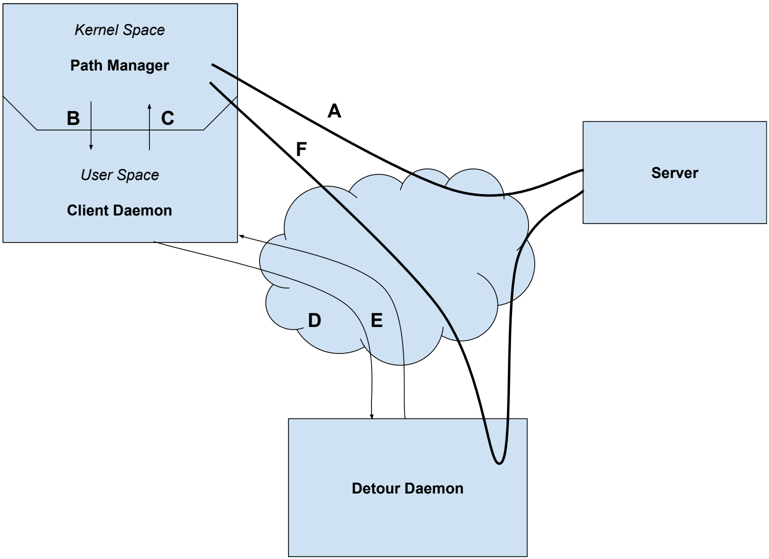
Conclusion

Concept Overview



Ingredients

- Multipath TCP Linux Implementation v0.91
- Custom path manager
- OpenVPN
- Netfilter / IPTables frameworks


Kernel Space
Path Manager
B
C
User Space
Client Daemon
A
F
Server
D
E
Detour Daemon

Strategies for Detours

- OpenVPN Approach
  - Establish an OpenVPN connection with detour
  - Send packets as normal through the virtual interface
  - Packets encapsulated via OpenVPN protocol
- NAT Approach
  - Address packets directly to detour
  - Detour alters source and destination address, forwards packet
  - Address information must be arranged ahead of time

OpenVPN Approach

- OpenVPN typically provides encryption and authentication
- Configure to only provide authentication on startup, no encryption or message signatures
- Use UDP as transport, to avoid "TCP Meltdown"
- VPN appears as network device to the kernel
- No per-MPTCP-connection signalling, but has per-packet overhead

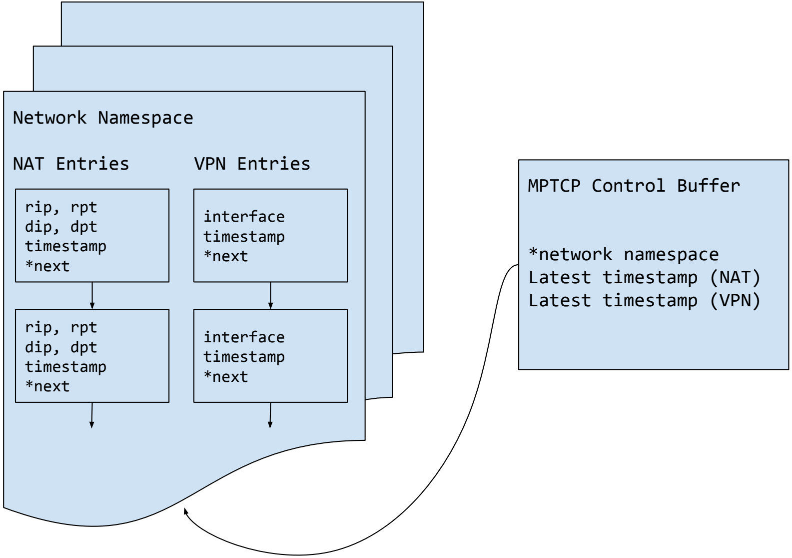
NAT Approach

<table>
<tr><td>0</td><td>7 8</td><td>15 16</td><td>23 24</td><td>31</td></tr>
<tr><td colspan="2">ver</td><td>op</td><td colspan="2">reserved</td></tr>
<tr><td colspan="5">rip</td></tr>
<tr><td colspan="3">rpt</td><td colspan="2">dpt</td></tr>
</table>

Custom protocol for arranging NAT detours

Path Manager

- Once a MPTCP connection is established, path manager is informed
- Path manager runs in a background thread
- Requests detours from client daemon
- Adds up to N additional subflows, where N is configurable. By default $N = 2$
- Whenever a new detour becomes available, runs again


Network Namespace
NAT Entries
VPN Entries
rip, rpt
dip, dpt
timestamp
*next
interface
timestamp
*next
rip, rpt
dip, dpt
timestamp
*next
interface
timestamp
*next
MPTCP Control Buffer
*network namespace
Latest timestamp (NAT)
Latest timestamp (VPN)

Client Daemon

- Userspace daemon required for tasks which are not well-suited for the kernel:
  - Starting processes
  - Using UDP sockets
- Daemon reads configuration file containing NAT and VPN detours.
- VPN instances are started up first and reported to kernel
- Wait for detour requests from kernel, send UDP requests, report replies to kernel
- All communication over Generic Netlink

Putting it Together (NAT)

1. Application creates MPTCP connection to MPTCP supporting server
2. Once 3WHS completes, path manager requests a detour from client daemon
3. Client daemon receives request and sends UDP request to every detour listed in configuration file
4. Detour daemon sets up detour, sends reply
5. Client daemon forwards reply to kernel
6. The path manager restarts the MPTCP connection's thread, which creates a new subflow via this detour

Putting it Together (VPN)

(0) At startup, client daemon connects to VPN and reports VPN to kernel

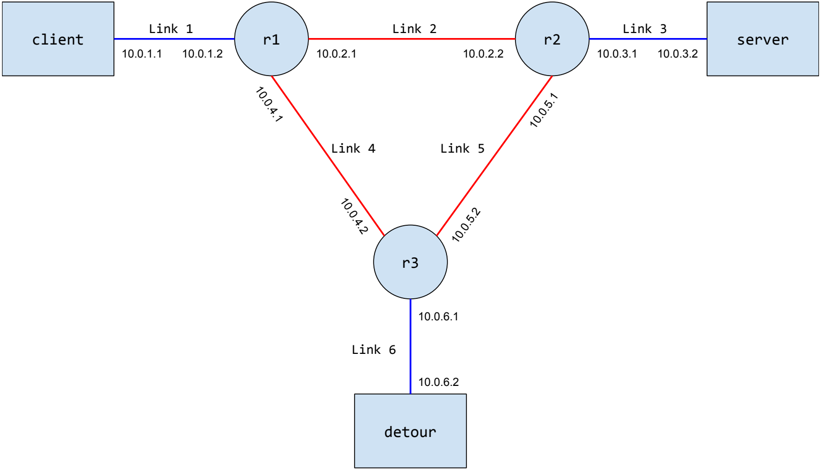
1. Application creates MPTCP connection to MPTCP supporting server
2. Once 3WHS completes, path manager requests a detour from client daemon.
3. Meanwhile, it uses the VPN already available and establishes a subflow.

Types of Experiments

- Previous work has established that there do exist common scenarios where detour routing can improve path characteristics
- We simply attempt show mechanism works as expected
- Answer the following
  - Can we achieve throughput of best available path?
  - When bandwidth aggregation is possible, can we aggregate path bandwidth?
  - What overheads exist in this mechanism?
  - Can this mechanism be used across the Internet at higher throughput?

Mininet Experiments

- Mininet allows you to create arbitrary network topologies
- Uses host networking stack rather than alternative or simulation
- Uses namespacing (foundation of containerization) rather than virtualization


client
Link 1
10.0.1.1
10.0.1.2
r1
Link 2
10.0.2.1
10.0.2.2
r2
Link 3
10.0.3.1
10.0.3.2
server
10.0.4.1
Link 4
10.0.4.2
Link 5
10.0.5.1
10.0.5.2
r3
10.0.6.1
Link 6
10.0.6.2
detour

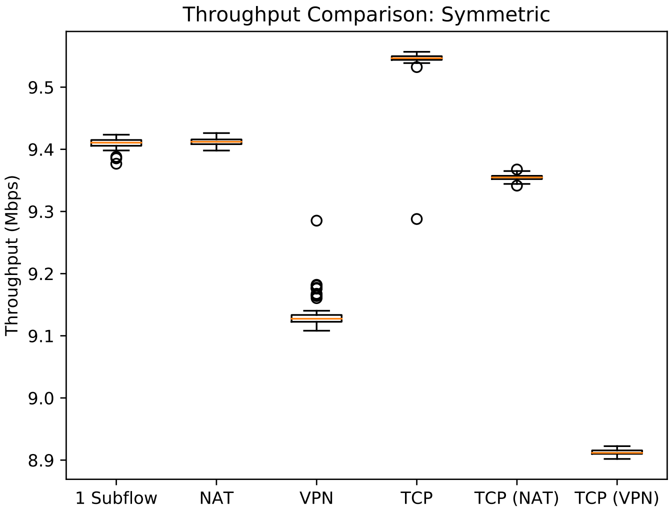
Scenarios

- Two types of network:
  - **Symmetric:** every link has 10Mbps bandwidth
  - **Core-limited:** core links have 10Mbps, access links have 20Mbps
- Three variations:
  - **Normal:** no loss
  - **Lossy:** 1% packet loss on Link 2
  - **Delayed:** 100ms delay on Link 2
- Workload: send as much data as possible from client to server

Mechanisms

- **1-Subflow:** MPTCP with no available detours
- **NAT:** Using NAT detour
- **VPN:** Using VPN detour
- **TCP:** TCP over default route
- **TCP(NAT):** TCP via the NAT tunnel
- **TCP(VPN):** TCP via the VPN tunnel

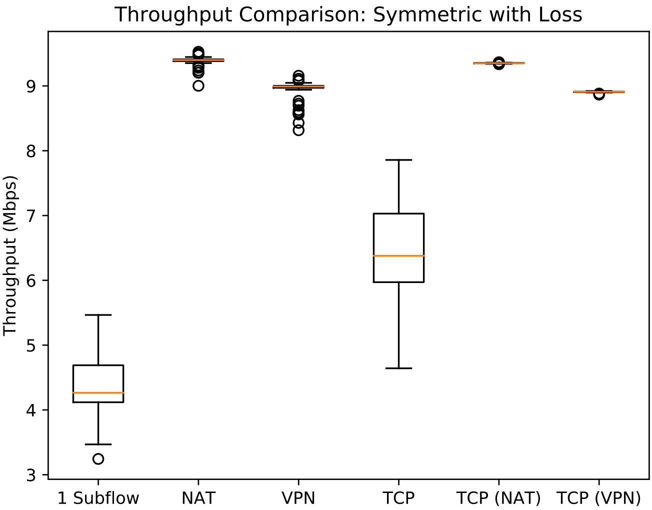
Results



Throughput Comparison: Symmetric
Throughput (Mbps)
9.5
9.4
9.3
9.2
9.1
9.0
8.9
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

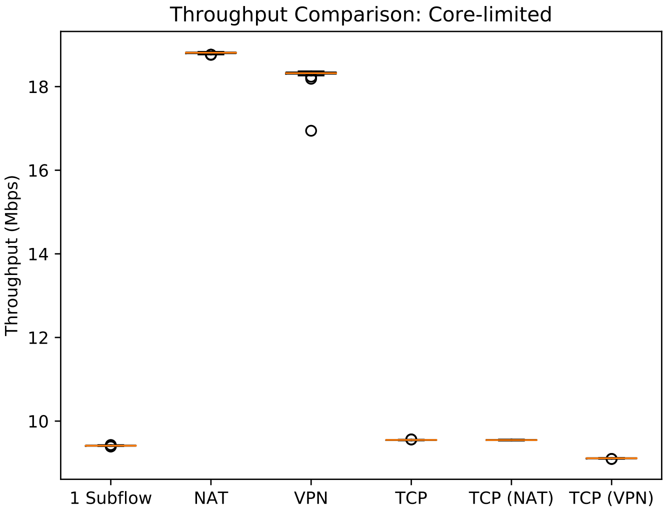
Results

- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible


Throughput Comparison: Symmetric with Loss
Throughput (Mbps)
3
4
5
6
7
8
9
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

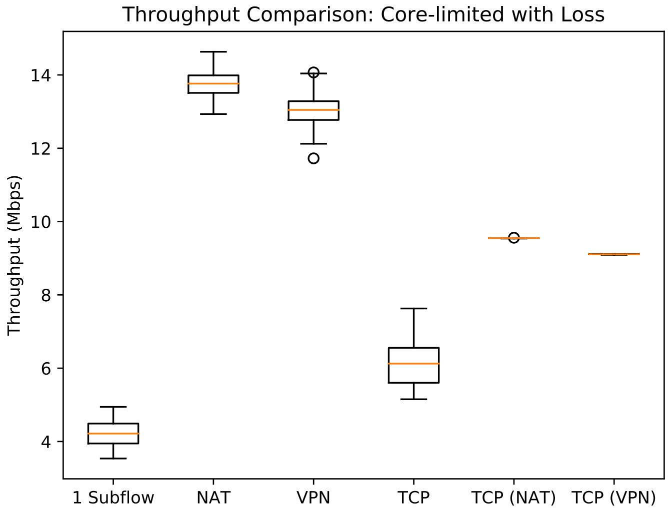
Results

- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible
- Mechanism performs similarly to TCP over best path when the default path has loss


Throughput Comparison: Core-limited
Throughput (Mbps)
18
16
14
12
10
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

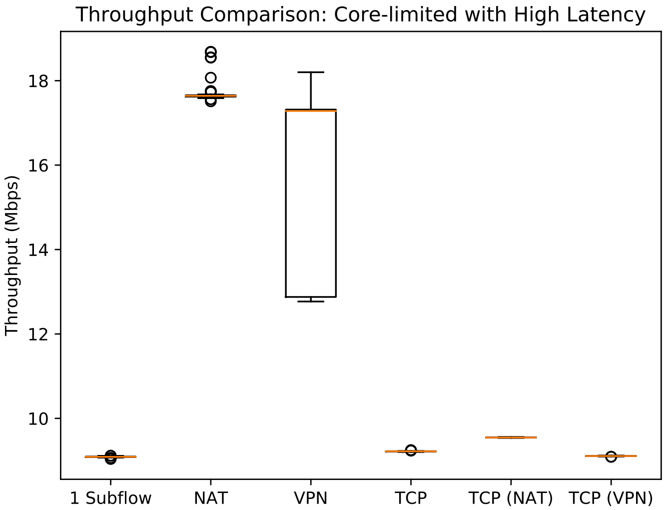
Results

- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible
- Mechanism performs similarly to TCP over best path when the default path has loss
- Mechanism can effectively aggregate bandwidth when the network has the potential


Throughput Comparison: Core-limited with Loss
Throughput (Mbps)
4
6
8
10
12
14
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

Results

- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible
- Mechanism performs similarly to TCP over best path when the default path has loss
- Mechanism can effectively aggregate bandwidth when the network has the potential, even in the presence of loss


Throughput Comparison: Core-limited with High Latency
Throughput (Mbps)
10
12
14
16
18
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

Results

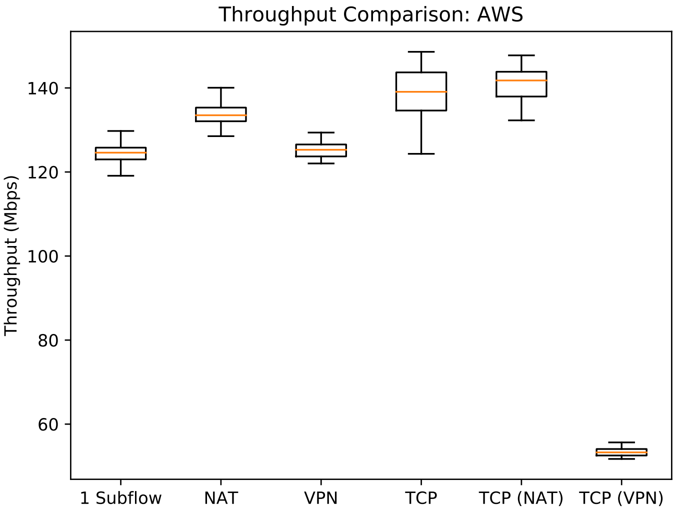
- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible
- Mechanism performs similarly to TCP over best path when the default path has loss
- Mechanism can effectively aggregate bandwidth when the network has the potential, even in the presence of loss or high latency

Results

- MPTCP has 140kbps, or about 1.5% overhead
- VPN approach has overhead of about 6.6%
- Mechanism performs well even when no aggregation benefit possible
- Mechanism performs similarly to TCP over best path when the default path has loss
- Mechanism can effectively aggregate bandwidth when the network has the potential, even in the presence of loss or high latency
- NAT consistently outperforms VPN both in MPTCP and TCP, but by a small amount.

AWS Experiments

- Deployed client, server, and detour implementations to different AWS regions
- Ran similar throughput measurements for MPTCP
- Performed at much higer level, but didn't show similar improvements
- OpenVPN cannot sustain above 60Mbps in our setup


Throughput Comparison: AWS
Throughput (Mbps)
60
80
100
120
140
1 Subflow
NAT
VPN
TCP
TCP (NAT)
TCP (VPN)

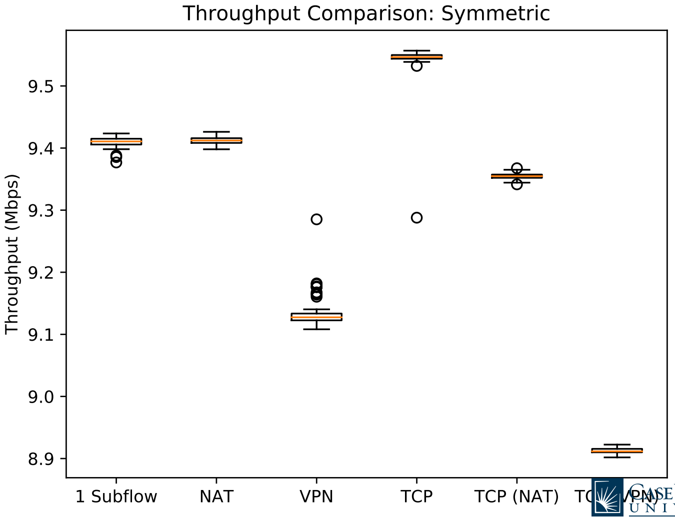
Summary

- Created a system for adding detour routes to MPTCP connections between single-homed devices.
- Like MPTCP, this system works with unmodified applications
- System is capable of achieving similar performance to the best available path when no aggregation is possible
- System is capable of aggregating throughput when possible

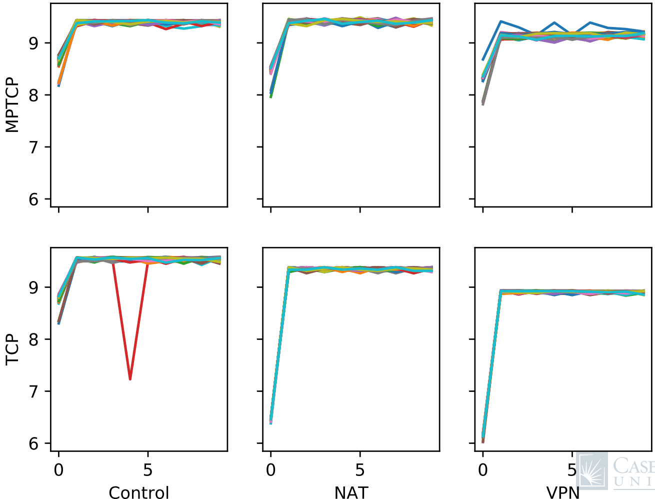
Future Work

- Deployment scenarios
- Dynamic subflow addition and removal
- Data scheduling
- 0-RTT NAT establishment

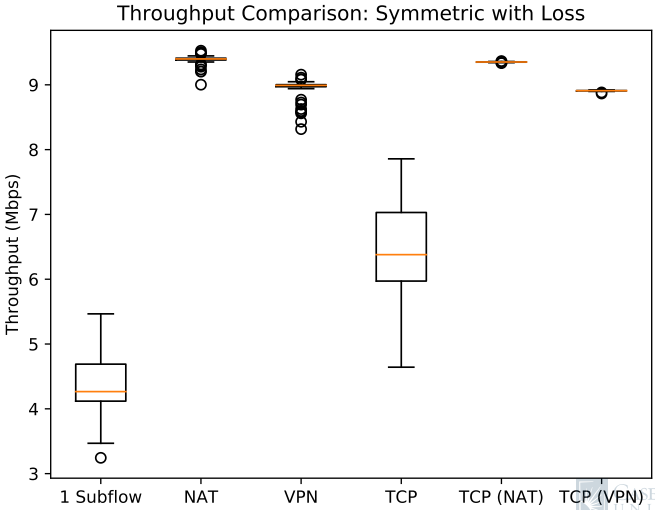
Throughput Comparison, Symmetric



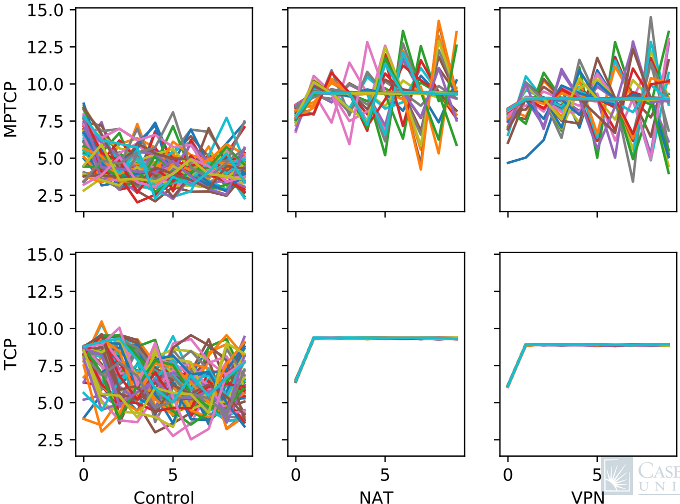
Timelapse, Symmetric



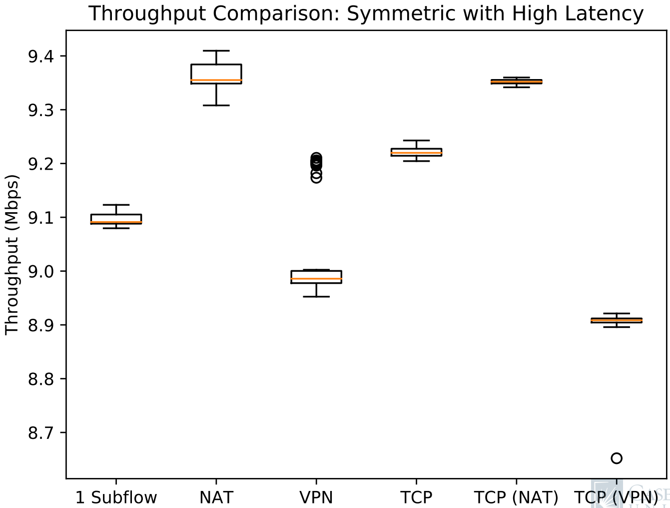
Throughput Comparison, Symmetric with Loss



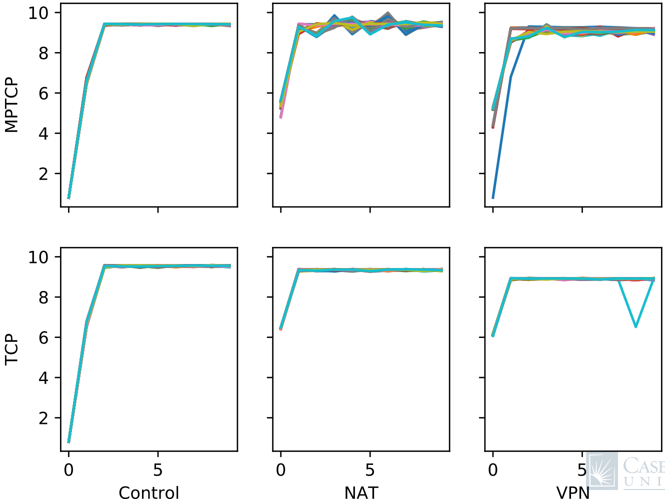
Timelapse, Symmetric with Loss



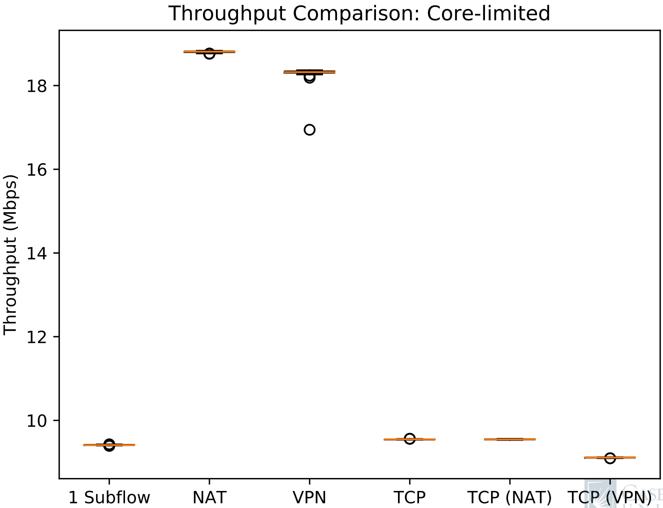
Throughput Comparison, Symmetric with Delay



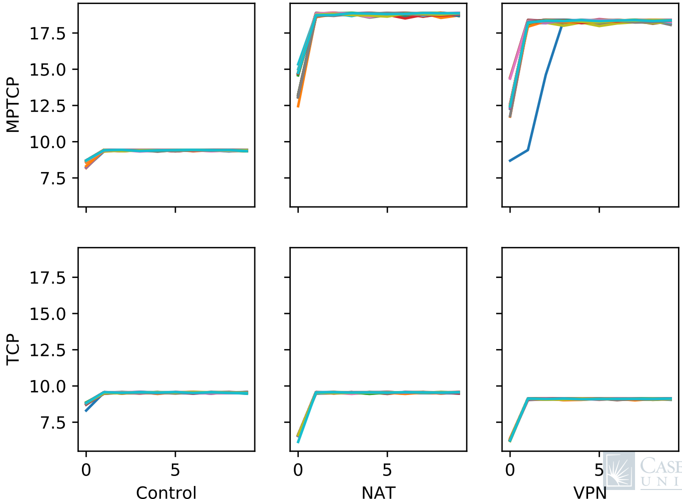
Timelapse, Symmetric with Delay



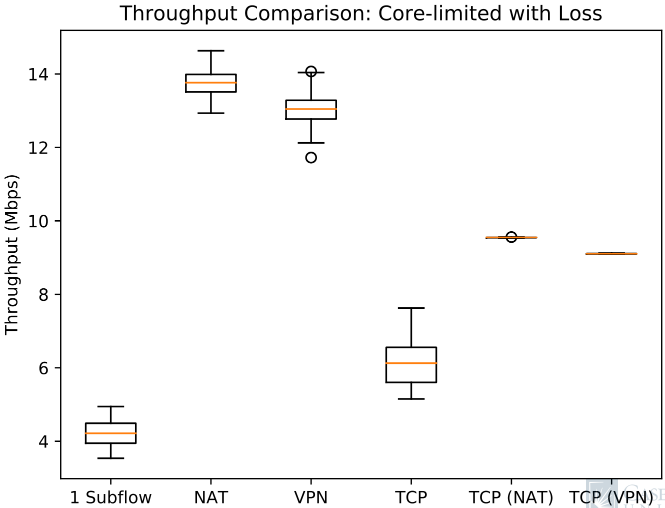
Throughput Comparison, Core-limited



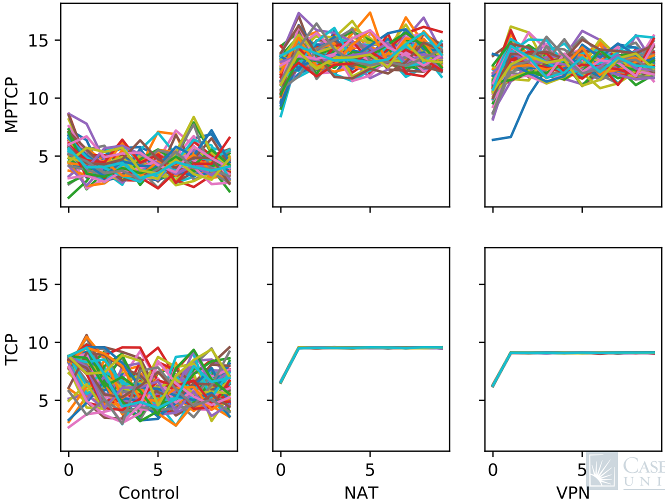
Timelapse, Core-limited



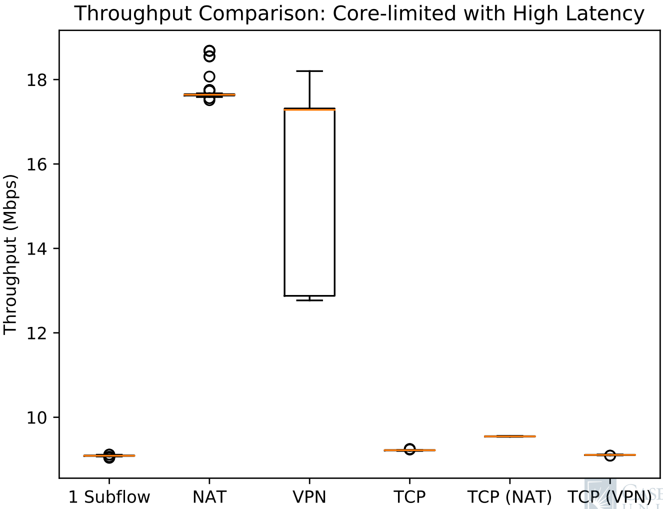
Throughput Comparison, Core-limited with Loss



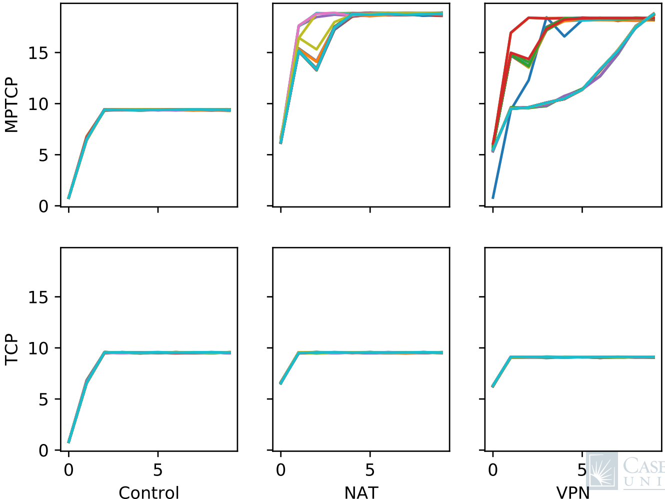
Timelapse, Core-limited with Loss



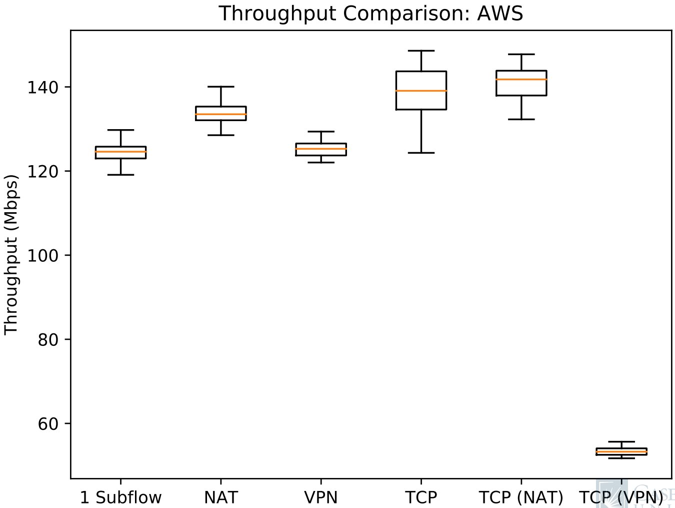
Throughput Comparison, Core-limited with Delay



Timelapse, Core-limited with Delay



Throughput Comparison, AWS



Timelapse, AWS