Multipath Transport for Virtual Private Networks

Daniel Lukaszewski\textsuperscript{1}  Geoffrey Xie\textsuperscript{2}

\textsuperscript{1}U.S. Department of Defense
\textsuperscript{2}Department of Computer Science
Naval Postgraduate School

USENIX Security 2017
Computer Security Experimentation and Test, 2017
Motivating Scenario

- VPNs provide increases security of Internet traffic
- Additional overhead results in reduced end-to-end speeds

![Diagram]

<table>
<thead>
<tr>
<th>Web/VPN Client</th>
<th>VPN Server</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1*</td>
<td>IP_c</td>
<td>IP_s</td>
</tr>
<tr>
<td>P1</td>
<td>IP_{tun_c}</td>
<td>IP_{web}</td>
</tr>
<tr>
<td>P2</td>
<td>IP_{web}</td>
<td>IP_{tun_c}</td>
</tr>
<tr>
<td>P2*</td>
<td>IP_s</td>
<td>IP_c</td>
</tr>
</tbody>
</table>

**ENCAPSULATION**

- P1: IP_{tun_c} IP_{web} TCP DATA
- P2: IP_{web} IP_{tun_c} TCP DATA
- P1*: IP_s IP_c Transport P1
- P2*: IP_s IP_c Transport P2
VPNs provide increases security of Internet traffic
Additional overhead results in reduced end-to-end speeds
TCP tunnels may suffer from TCP-in-TCP effects
(Why TCP in TCP is a Bad Idea, Olaf Titz, 2001)
- Two layers of congestion control
- Packet loss on lower layer can increase RTO
- Upper layer has shorter RTO causing unnecessary re-transmissions

**Motivating Scenario**

**P1**

- $IP_c$
- $IP_s$
- TCP
- $IP_{tun_c}$
- $IP_{web}$
- TCP
- DATA

$RTO_L$: 2 sec
$RTO_U$: 1 sec
UDP vs. TCP Tunnels

- UDP tunnels remove extra layer of congestion control
- Usually default mode for tunnel setup

![Graph showing improvement above TCP in different packet loss rates and link propagation RTT values.](image)
UDP vs. TCP Tunnels

- UDP tunnels remove extra layer of congestion control
- Usually default mode for tunnel setup

Multipath Transport Protocols Further Improve Performance
Related Multipath Work

MPTCP:

- **Improving Multipath TCP** [C. Paasch, 2014]
  - PhD Dissertation
  - In depth design and analysis of MPTCP
- **Binder** [Boccassi et. al, 2013]
  - MPTCP with gateway-to-gateway VPN model
  - Aggregate paths in community network

MPUDP:

- **MP-MOSH** [Boutier and Chroboczeck, 2015]
  - Multiple UDP paths established for Mobile Shell (MOSH) application
  - Not a general solution since application specific
How MPTCP Establishes Connection

Client

\[ \text{SYN}_{\text{MP\_CAPABLE}} \]

\[ \text{ACK} \]

\[ \text{SYN}_{\text{MP\_JOIN}} \]

\[ \text{ACK}_{\text{MP\_JOIN}} \]

Server

\[ K_c \]

\[ K_s \]

\[ \text{Token}_s \]

\[ \text{HMAC}_s \]

\[ \text{SYN/ACK}_{\text{MP\_CAPABLE}} \]

\[ \text{SYN/ACK}_{\text{MP\_JOIN}} \]

\[ \text{ACK} \]

\[ \text{HMAC}_c \]
MPTCP Path Managers

Ndiffports:

MPTCP Client  \[\rightarrow\]  Port 3  \[\rightarrow\]  Port 1  \[\rightarrow\]  Port 2  \[\rightarrow\]  MPTCP Server

Fullmesh:

MPTCP Client  \[\leftrightarrow\]  Wi-Fi  \[\leftrightarrow\]  LTE  \[\leftrightarrow\]  MPTCP Server
Design Testbed

- Physical Linux machines operating Ubuntu 14.04
- MPTCP v0.90 on VPN Client and VPN Server
- OpenVPN 2.3.12

![Diagram of network setup]

- Primary Link (10 Mbps)
- Secondary Link (10 Mbps)
- Web Server (1 Gbps)
Traffic Loss Models

- Uniform: basic loss model in Linux traffic controller (tc)
- Gilbert-Elliot (GE):

\[
\begin{align*}
\text{Gap} & : 1 - k \\
\text{Burst} & : 1 - h
\end{align*}
\]

<table>
<thead>
<tr>
<th>$1 - k$</th>
<th>Observed Packet Loss Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>1%</td>
</tr>
<tr>
<td>0.5%</td>
<td>2%</td>
</tr>
<tr>
<td>1%</td>
<td>3%</td>
</tr>
<tr>
<td>3%</td>
<td>7%</td>
</tr>
</tbody>
</table>
## Design Metrics

### General:
- **Goodput**: primary measure of performance
- **Link Speeds**:
  - VPN Client $\Rightarrow$ VPN Server: 10 Mbps
  - VPN Server $\Rightarrow$ Web Server: 1 Gbps
- **RTTs**: 1, 10, 20, 40, 60, 80, 100ms
- 16 MB text file downloads performed 20 times to get 90% confidence interval
- No VPN encryption or compression to facilitate data analysis

### Asymmetric RTT Ratios:

<table>
<thead>
<tr>
<th>RTT Ratio</th>
<th>Primary Path</th>
<th>Secondary Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>20ms</td>
<td>20ms</td>
</tr>
<tr>
<td>1:2</td>
<td>20ms</td>
<td>40ms</td>
</tr>
<tr>
<td>1:3</td>
<td>20ms</td>
<td>60ms</td>
</tr>
<tr>
<td>1:4</td>
<td>20ms</td>
<td>80ms</td>
</tr>
<tr>
<td>1:5</td>
<td>20ms</td>
<td>100ms</td>
</tr>
</tbody>
</table>
MPTCP: Symmetric Links

- Fullmesh and Ndiffports show improvements over single path TCP
- Performance under 1% packet loss:

(a) Uniform loss probability
(b) GE bursty loss model
MPTCP: Assymetric Links

- Multiple links unlikely to have same delay rates
- MPTCP handles larger ratios of path asymmetry without significant performance loss
MPUDP Design

- Leverage existing MPTCP Linux kernel
- Used Linux Kernel Modules (LKM) to facilitate testing
  - Allowed quickly changing MPUDP parameters without need to re-compile kernel
  - Changed code in `udp.c` kernel file to allow LKM to change operation of UDP protocol
- Probabilistic shift of traffic across primary and secondary links (50/50)
  - Keep simplistic design of UDP
- Overwrite incoming IP to match primary VPN link IP if over secondary link
MPUDP Evaluation: Initial Results

- Test under 0 simulated packet losses
- MPUDP maintains high goodput values as MPTCP begins to suffer
Detailed Look at MPTCP vs. MPUDP Download

- Evaluated the first 0.3 seconds of download at 40ms RTT
- MPTCP slower to grab available bandwidth
MPUDP Evaluation: Symmetric Links

- MPUDP still shows improvements

(e) 1% Packet Loss (Uniform)

(f) 3% Packet Loss (GE)
MPUDP Evaluation: Asymmetric Links

- 50/50 Probabilistic shift of traffic not ideal for asymmetric links
- When ratio low, MPUDP performs slightly better than MPTCP
Effects of Changing MPUDP Split Settings

(g) 75/25 Split

(h) 90/10 Split
Summary

Key Take-Aways:

- MPTCP can provide increased performance when used for VPN tunnels
- MPUDP tunnels are capable of out-performing MPTCP tunnels

Future Work

- Provide mechanism for MPUDP to track link characteristics
- Expand analysis to end-to-end UDP traffic
- Evaluate the possible benefits of adding a FEC mechanism to MPUDP