Reducing Latency in Tor Circuits with Unordered Delivery

Michael F. Nowlan, David Wolinsky, and Bryan Ford
Yale University
Dedis Lab
Tor – The Onion Router

Bob

Server

Anonymizing Relays

Server

Server

Server

Facebook
Tor – The Onion Router

Anonymizing Relays
Performance Problems

- Traffic between two relays streams across 1 TCP Conn
Performance Problems

- Traffic between two relays streams across 1 TCP Conn
- Cross circuit Head-of-line blocking
Performance Problems

- Traffic between two relays streams across 1 TCP Conn
- Cross circuit Head-of-line blocking
- TCP Buffers can create delays

Outgoing queue head

---

Carol  Bob  Alice  Carol  Alice
Outline

• Introduction to Tor
• Tor / TCP Bottlenecks
• Design Space
• Tool #1: uTCP
• Tool #2: uTLS
• Eliminating Head-of-Line Blocking in Tor
• Conclusions
Our Goals

- Improve Tor QoS for low latency, low bandwidth
  - Eliminate head-of-line blocking
  - Improve latency for low latency, low bandwidth Apps
- Use existing transport (TCP/TLS)
- Minimal modifications to Tor
- Interoperability with existing Tor deployments
Design Space

• Different transports
  • DTLS/TCP over UDP, PCTCP, UDP
  • Tradeoff between user-space networking stack or add reliability layer
  • IPSec not universally configurable

• One connection per-circuit
  • OS limitations
  • Potential performance hit
  • No shared congestion state

• One connection per-priority
  • Torquestra
  • May leak information about the flows
  • Doubles active sockets, no shared congestion state
Outline

• Introduction to Tor
• Tor / TCP Bottlenecks
• Design Space
• Tool #1: uTCP
• Tool #2: uTLS
• Eliminating Head-of-Line Blocking in Tor
• Conclusions
Unordered TCP (uTCP)

- Small kernel modification and module
  - Less than 600 lines of code
  - Supported on Linux Kernel 2.6 and 3
  - Recent IETF activity by Apple
- Out-of-order reads
- Priority sending
- Disable congestion control
uTCP Out-of-Order Reads
uTCP Priority Sending

- Give low latency, low bandwidth circuits preference
- Initiate by a setsockopt
- User prepends priority to each packet
- Inserted into send buffer ahead of lower priority data
- Optional squash to erase data from the buffer

Outgoing queue head

```
0  Carol 0  Bob 0  Alice 1  Carol 1  Alice 2  Alice 3  Carol
```

1 Bob
uTCP Disable Congestion Control

- Initiated with a setsockopt
- Null TCP congestion control kernel module
- Always sets congestion window to maximum size
uTCP

- Out-of-order reads
- Priority sending
- Disable congestion control

Remaining issues:
  - Tor uses TLS!
  - Need a framing mechanism for out-of-order msgs!
    - TLS provides framing...
Outline

• Introduction to Tor
• Tor / TCP Bottlenecks
• Design Space
• Tool #1: uTCP
• Tool #2: uTLS
• Eliminating Head-of-Line Blocking in Tor
• Conclusions
Processing TLS

- Header
  - Application data – 0x17
  - TLSv1.2 – 0x0303
  - Length – Up to 16 KB
- Data length check
- MAC check
- Decryption
- Deliver
uTLS

- Identifying records
  - Search for header:
    - Type: 0x17
    - Version: 0x0303
    - Size: $\leq 2^{16}$
  - Verify sufficient bytes for size
- Perform a MAC check – Must guess record number
  - Try a sequence
  - Rollback state when processing out of order
  - Fail otherwise
- Requires explicit IV an then decrypt and deliver
Outline

• Introduction to Tor
• Tor / TCP Bottlenecks
• Design Space
• Tool #1: uTCP
• Tool #2: uTLS
• Eliminating Head-of-Line Blocking in Tor
• Conclusions
From Tor to Unordered Tor

- Apply uTCP and uTLS to Tor to make it support unordered delivery – uTor
- Tor has multiple circuits sharing a single TCP stream
  - Each circuit passes datagrams or cells
  - Each circuit has independent state
uTor – Cell Processing

- One cell per record
- Each cell specifies its circuit
- Can we process cells out of order?
  - Assume no, for now
- Process cells in order
  - Add sequence number to cells
  - Header size goes from 5 to 7 bytes
  - Payload remains the same

<table>
<thead>
<tr>
<th></th>
<th>Tor</th>
<th>uTor</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC</td>
<td>81418</td>
<td>81513</td>
<td>+95 (0.0001%)</td>
</tr>
</tbody>
</table>
Deployment Challenges

• uTCP / uTLS adoption
  • Not found in main stream kernels
  • Currently only truly beneficial for ORs
  • Virtualization?

• uTor adoption
  • uTor must negotiate to use 2 byte cell ID
  • Compatible with existing Tor deployments
Preliminary Evaluation of uTor

- Tor configuration
  - 3 Relays
  - 3 Directory Authorities
  - A single proxy
  - OP takes the same path through the 3 relays
  - Version 0.2.5.0-alpha-dev (git-7c670895b02ba731)
- Normally distributed, 50 ms latency
- 0 and 5% loss
- Experiment
  - Requests to 40 popular sites on the Internet
  - Data sizes 10 KB to 400 KB
Preliminary Evaluation

Download Completion with Tor Variants

Fraction Completed Downloads

Time (seconds)

0 2 4 6 8 10 12 14 16 18

0 0.2 0.4 0.6 0.8 1

uTor
Tor
uTor-Loss
Tor-Loss
Outline

• Introduction to Tor
• Tor / TCP Bottlenecks
• Design Space
• Tool #1: uTCP
• Tool #2: uTLS
• Eliminating Head-of-Line Blocking in Tor
• Conclusions
Future Work

• uTor shows promise
  • Removes head-of-line blocking
  • No userspace networking libraries (DTLS/TCP Tor)
  • No need to change Tor to use UDP
• We can do more
  • Investigate improved congestion control algorithms
  • Priority sending – Similar to Torchestra
  • Impact of head-of-line blocking on congestion control
• Out of order circuits interesting but dangerous
  • Processing a destroy before receiving all packets
  • Assumed in order nature
Thank you for your attention

http://dedis.cs.yale.edu/2009/tng/
The Challenges with TCP

• Overlays multiplex a single TCP link for multiple streams
• A missed packet in one stream causes delays in all streams
  • Logically the streams are not blocked
  • TCP blocks until all packets are received in order
• Unable to perform priority sending
  • Cannot insert into TCP buffer
• Layered congestion control
  • TOR runs TCP between end points as well as hop by hop
  • Receiving a dropped packet in the hop by hop could result in a burst between the end points causing undesirable ramping up