Supercharge WebRTC: Accelerate TURN Services with eBPF/XDP

Tamás Lévai
Balázs Kreith
Gábor Rétvári
Diagram showing two VoIP systems interconnected with a green checkmark indicating successful connection.
TURN server

FW/NAT

VOIP

VOIP

Images: flaticon.com
What is TURN?

- **Traversal Using Relays around NAT**
  - extension to Session Traversal Utilities for NAT (STUN)
- Real-time communication systems rely on it
  - depletion of public IPv4 addresses and Carrier-grade NATs
  - part of the VoIP architecture, widely used in WebRTC
- Also used in VPNs, media gateways, cloud gaming, tunneling, …

- **Resource and network intensive**
  - requires large bandwidth and extremely low latency
  - small UDP packets
How TURN works?

- complex protocol: authentication, permission mgmt, etc.
- TURN sessions
  - initiated by end-users (client/peers)
  - server creates a transport relay connection to the peers
  - client/peers identified by 5-tuples (src/dst IP/port, protocol)
  - client and peer send data ...
    - with SendIndication
      - large overhead
      - adds a full TURN header to each packet
    - via channels
TURN Channels

- Lightweight method to send data
  - peer traffic is plain UDP
  - client traffic encapsulated with a **ChannelData** header

- bulk of TURN traffic is channel data
Challenge: Add/remove ChannelData header
Challenge: Add/remove ChannelData header
Challenge: Add/remove ChannelData header
Challenge: Update UDP checksum

- Add/remove ChannelData header invalidates the UDP checksum
  - UDP checksum: pseudo header + UDP data

UDP checksum update options:

- Recalc the full checksum
  - computation-heavy, limited in eBPF (loops -> limited pkt size)

- Use incremental checksum update
  - header updates: ✅
  - add/remove ChannelData: ❓
    - bpf_csum_diff()
Evaluation: Internal TURN server performance

No offload
Evaluation: Internal TURN server performance

No offload

XDP offload
Evaluation: End-to-end testbed

Tester

TURN clients

iperf client

iperf client

iperf server

TURN peer

turncat (turn proxy)

turncat (turn proxy)

SUT

TURN server
### Evaluation: End-to-end results

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Throughput [kpps]</th>
<th>Latency [ms]</th>
<th>Jitter [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 thread</td>
<td>68.5</td>
<td>0.1795</td>
<td>0.2261</td>
</tr>
<tr>
<td>4 threads</td>
<td>94.8</td>
<td>0.2492</td>
<td>0.3731</td>
</tr>
<tr>
<td>XDP</td>
<td>134.3</td>
<td>0.0852</td>
<td>0.0994</td>
</tr>
</tbody>
</table>

*baseline*
Summary

TURN

eBPF/XDP offload architecture

Code and artifacts available at

l7mp/turn/tree/server-ebpf-offload

UDP checksum

Monitoring

Challenges

Results

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Throughput [kpps]</th>
<th>Latency [ms] avg</th>
<th>Latency [ms] std dev</th>
<th>Jitter [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 thread</td>
<td>68.5</td>
<td>0.1795</td>
<td>0.2261</td>
<td>0.0265</td>
</tr>
<tr>
<td>4 threads</td>
<td>94.8</td>
<td>0.2492</td>
<td>0.3731</td>
<td>0.0351</td>
</tr>
<tr>
<td>XDP</td>
<td>134.3</td>
<td>0.0852</td>
<td>0.0994</td>
<td>0.0217</td>
</tr>
<tr>
<td>Baseline</td>
<td>134.3</td>
<td>0.0386</td>
<td>0.0132</td>
<td>0.0086</td>
</tr>
</tbody>
</table>