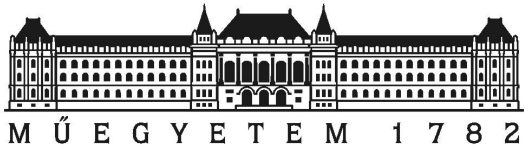
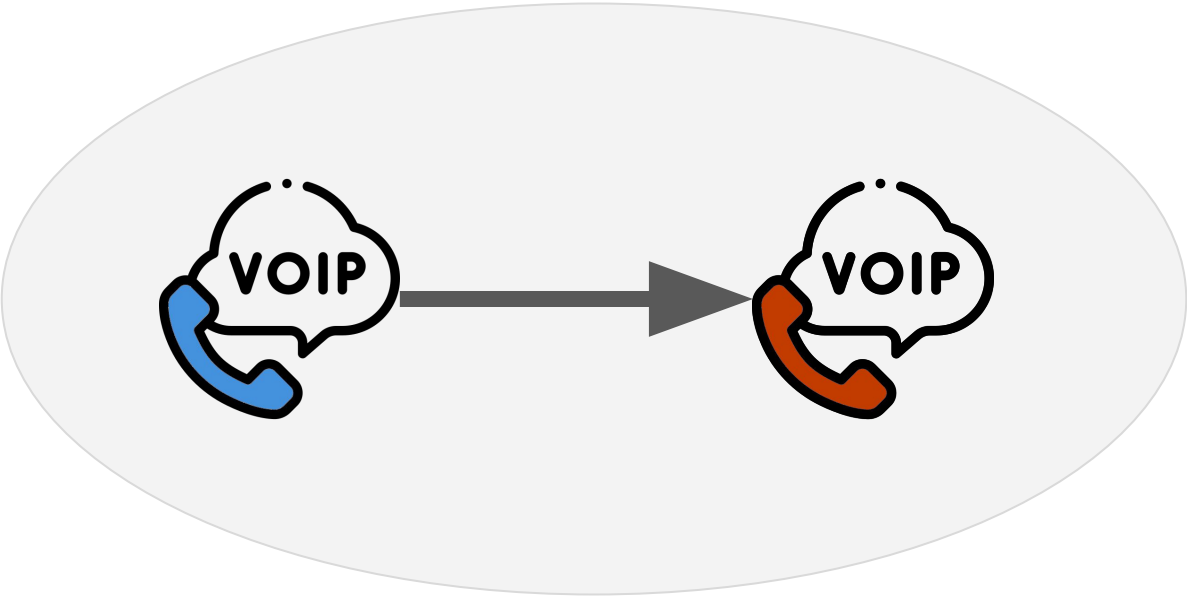
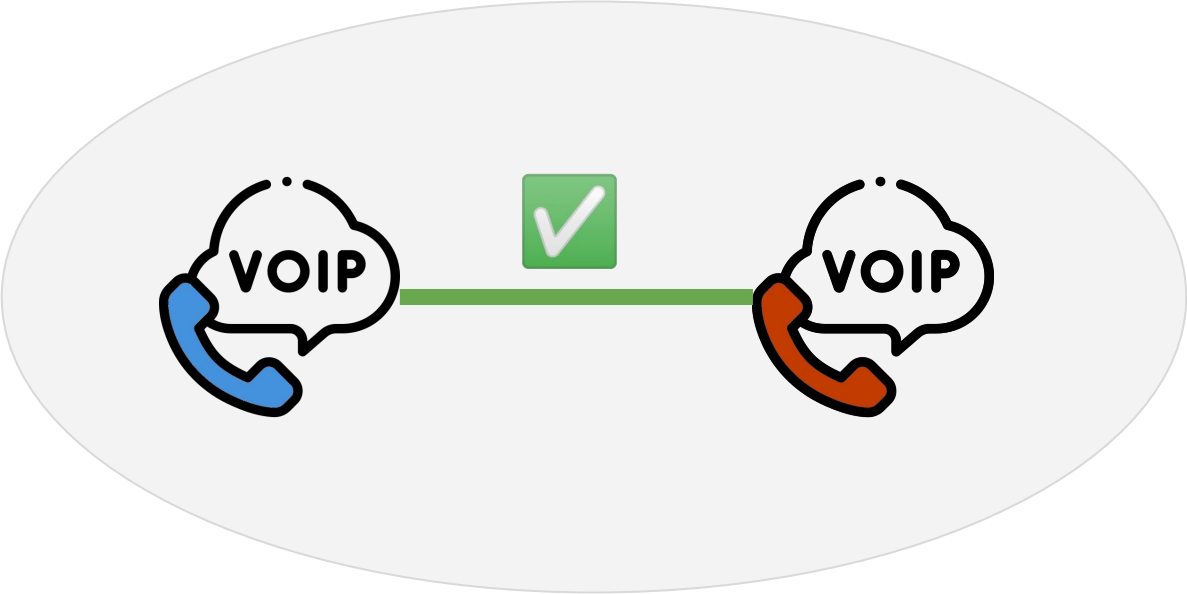
Supercharge WebRTC:
Accelerate TURN Services with eBPF/XDP

Tamás Lévai
Balázs Kreith
Gábor Rétvári

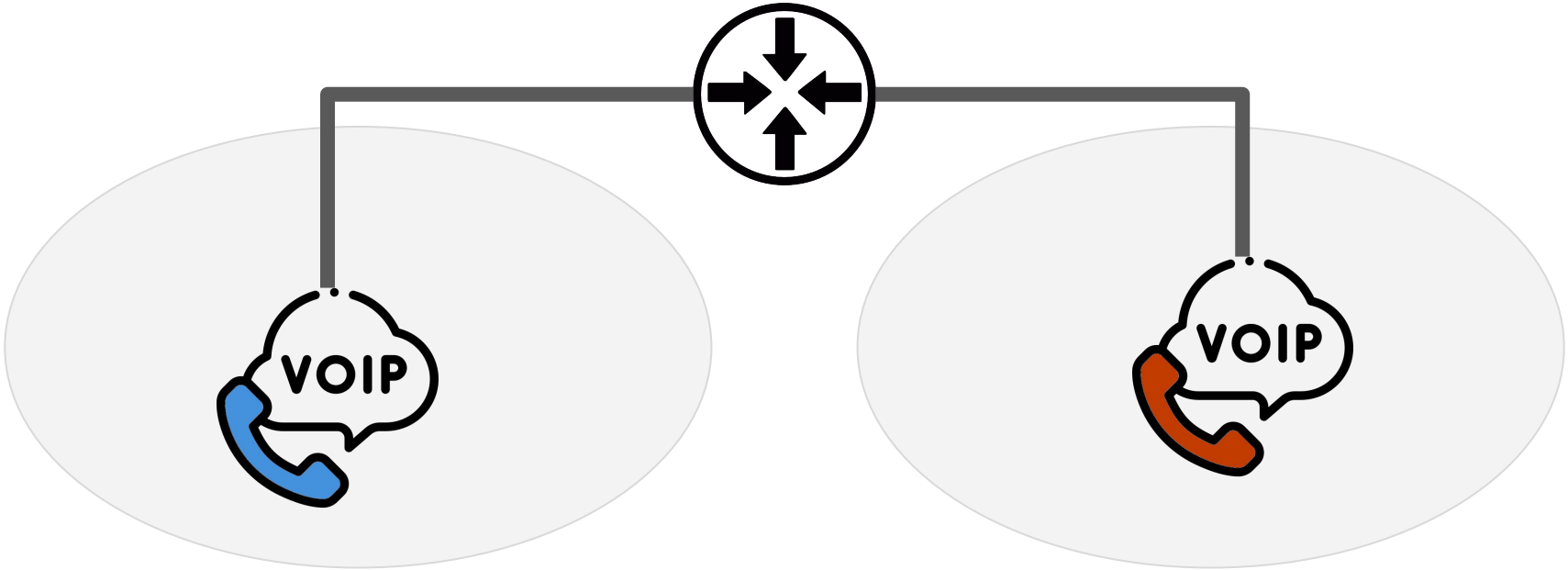




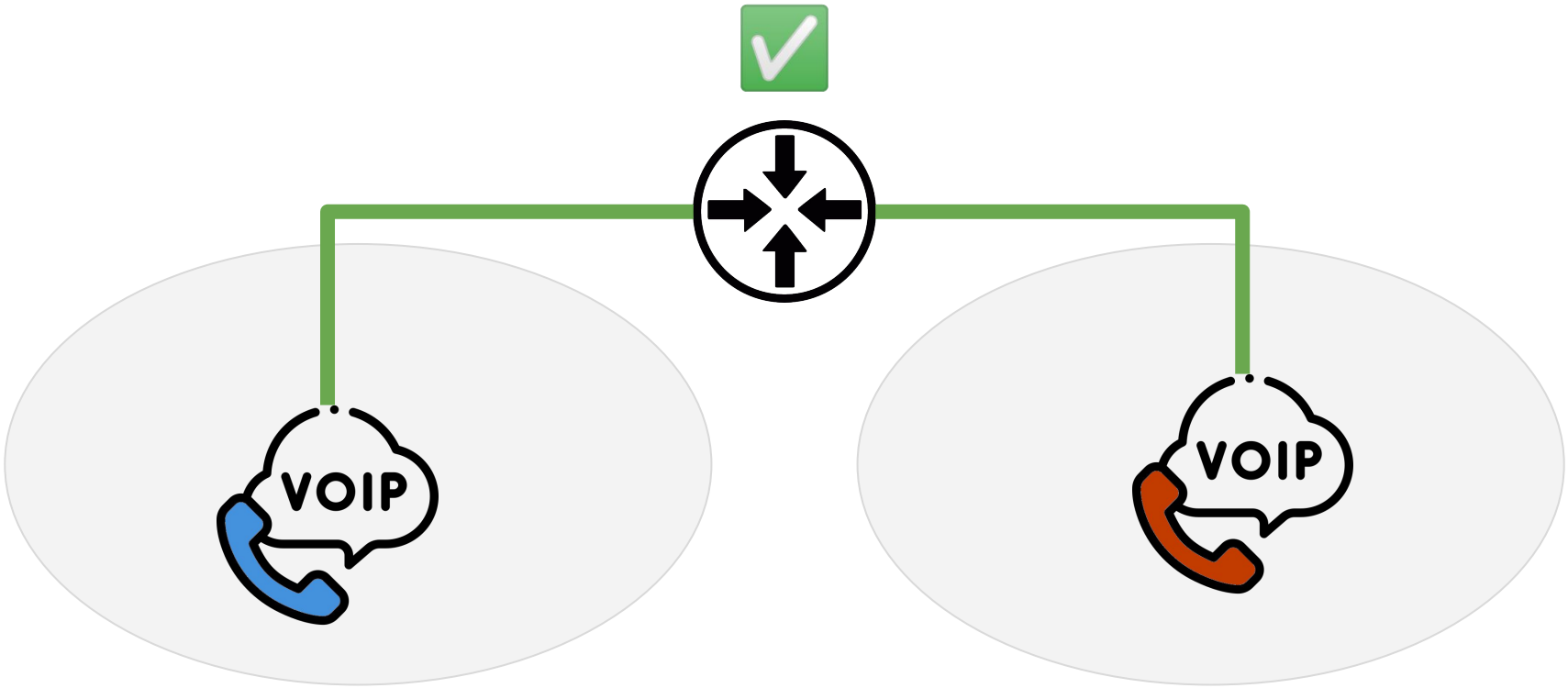
Images: flaticon.com



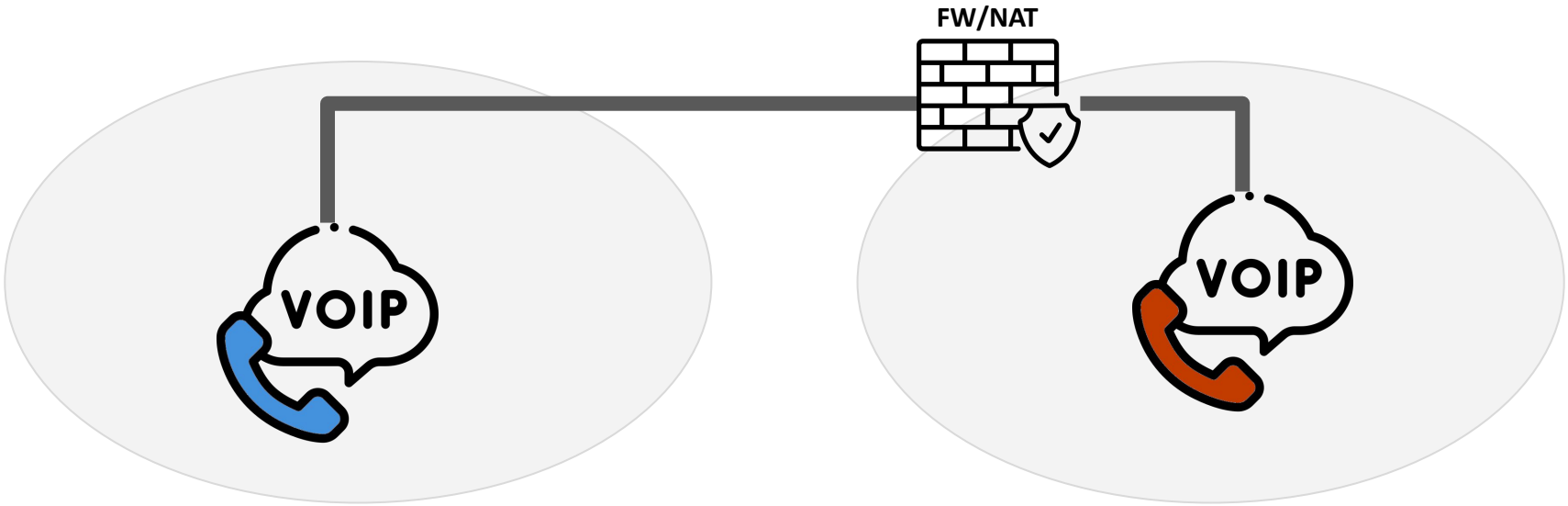
Images: flaticon.com



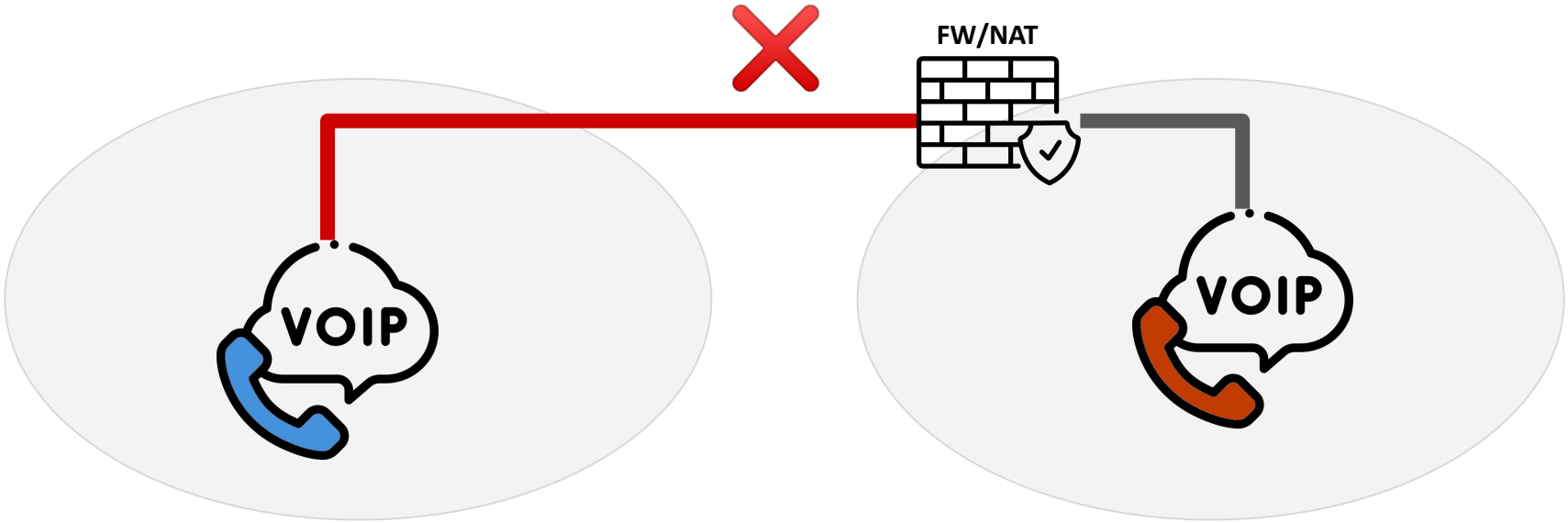
Images: flaticon.com



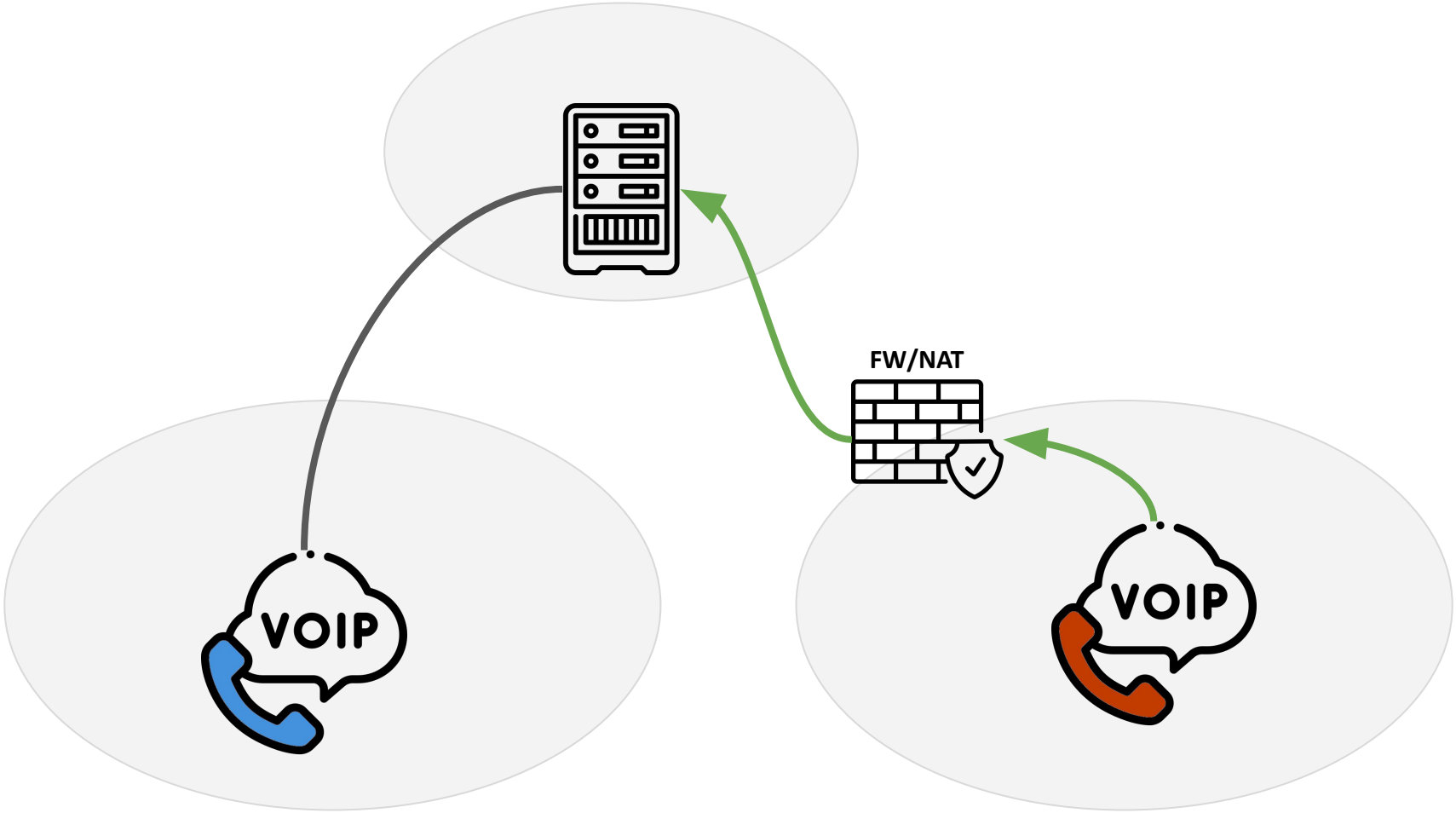
Images: flaticon.com



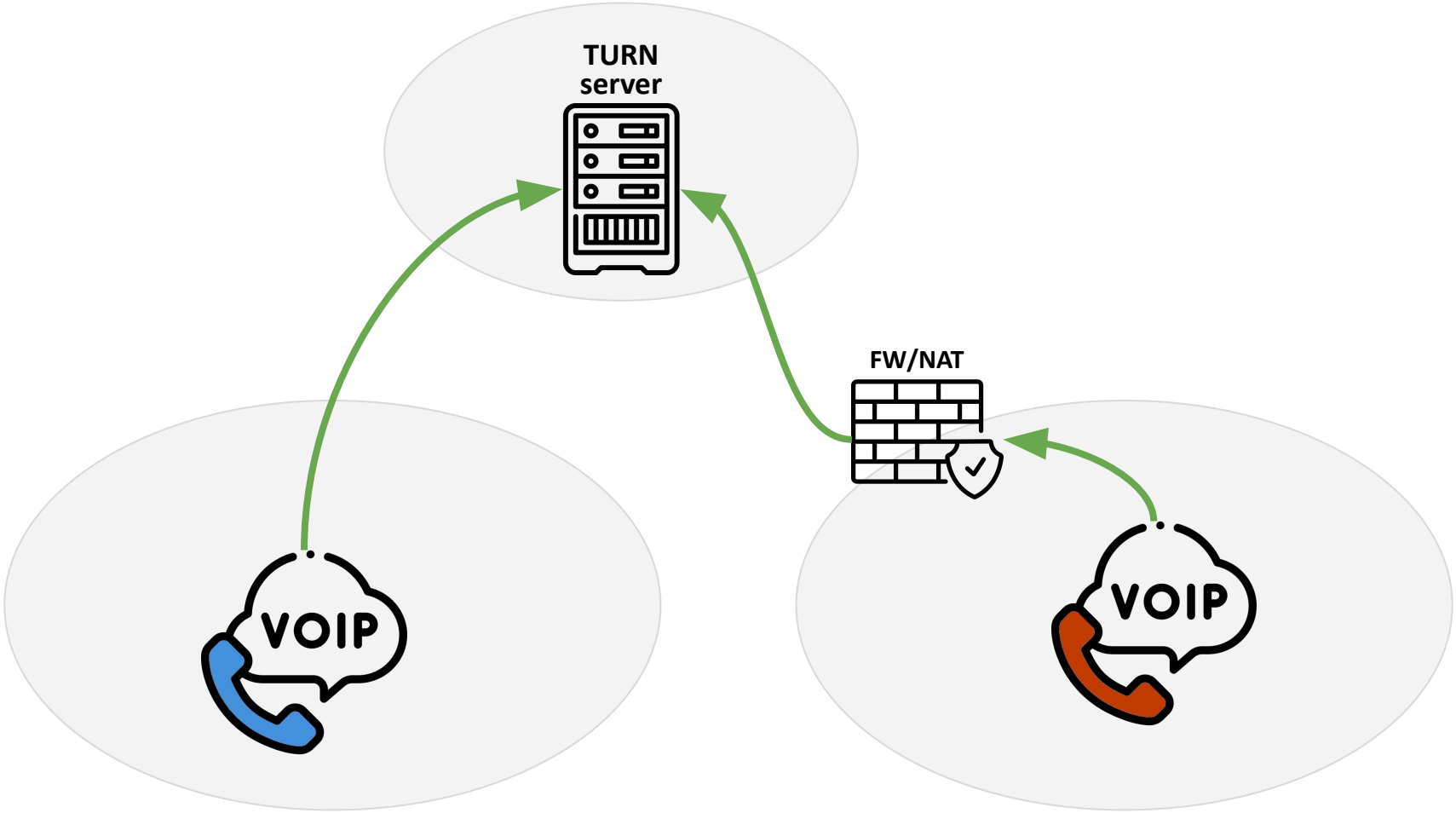
Images: flaticon.com



Images: flaticon.com

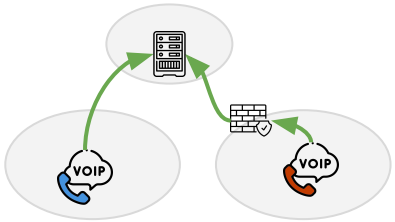


Images: flaticon.com



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, …

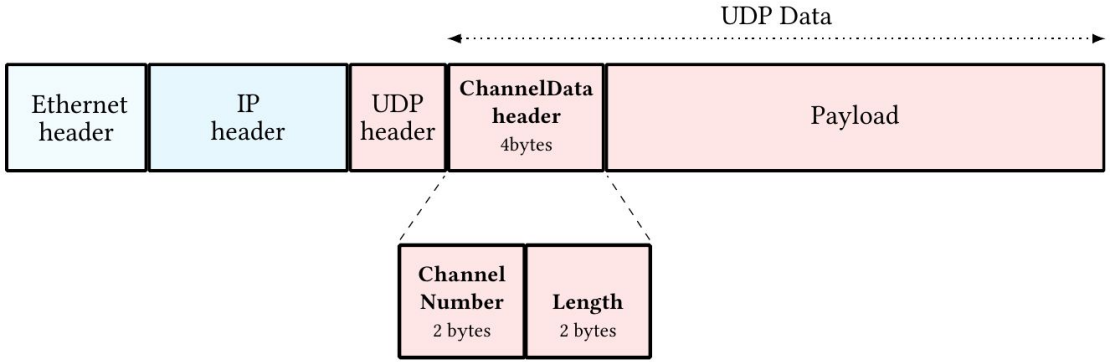
- Resource and network intensive
  - requires large bandwidth and extremely low latency
  - small UDP packets

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**

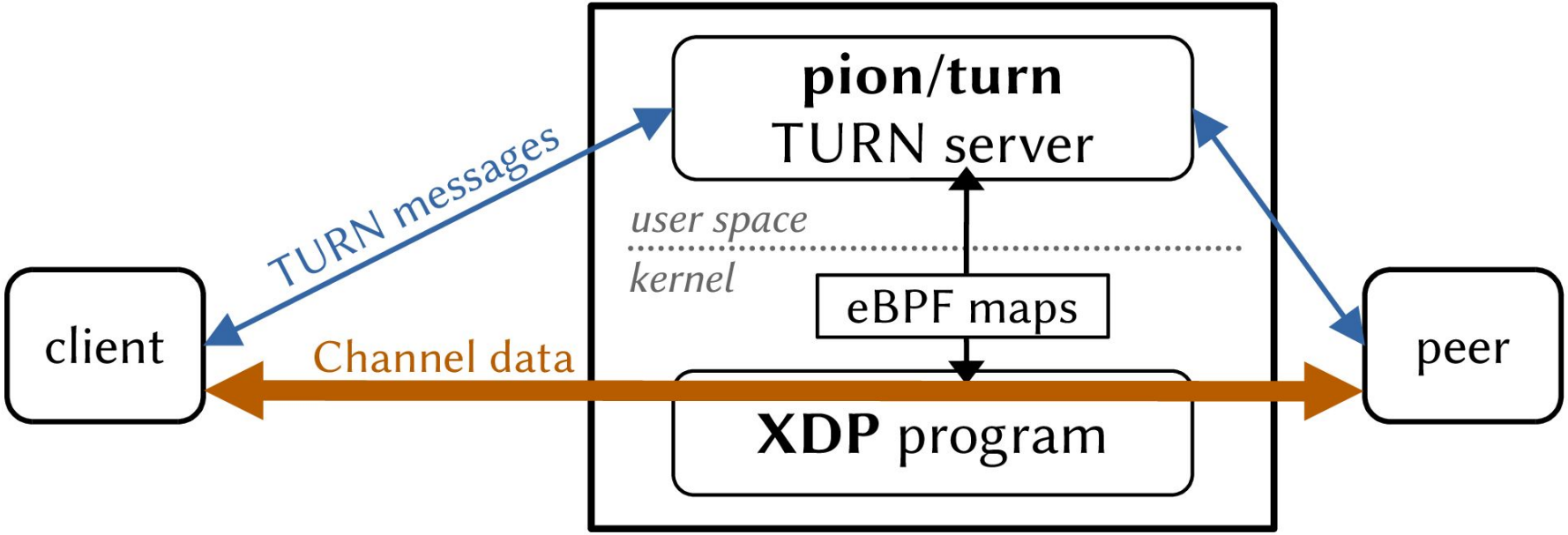
TURN Channels

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

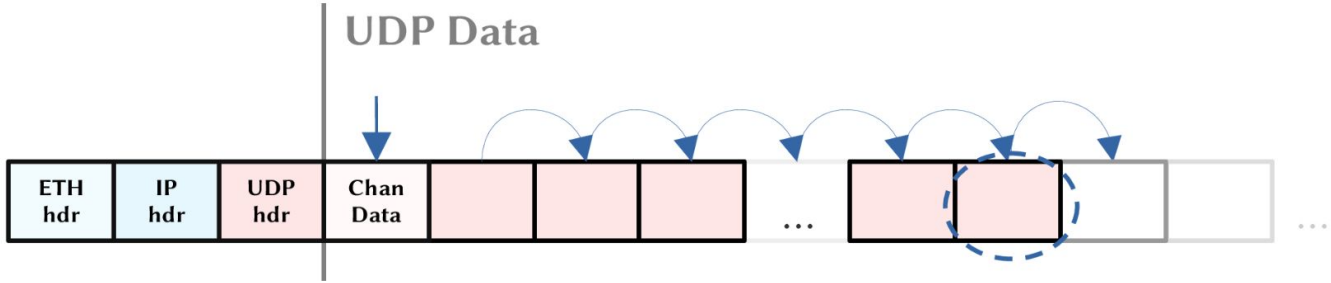


- **bulk of TURN traffic is channel data**

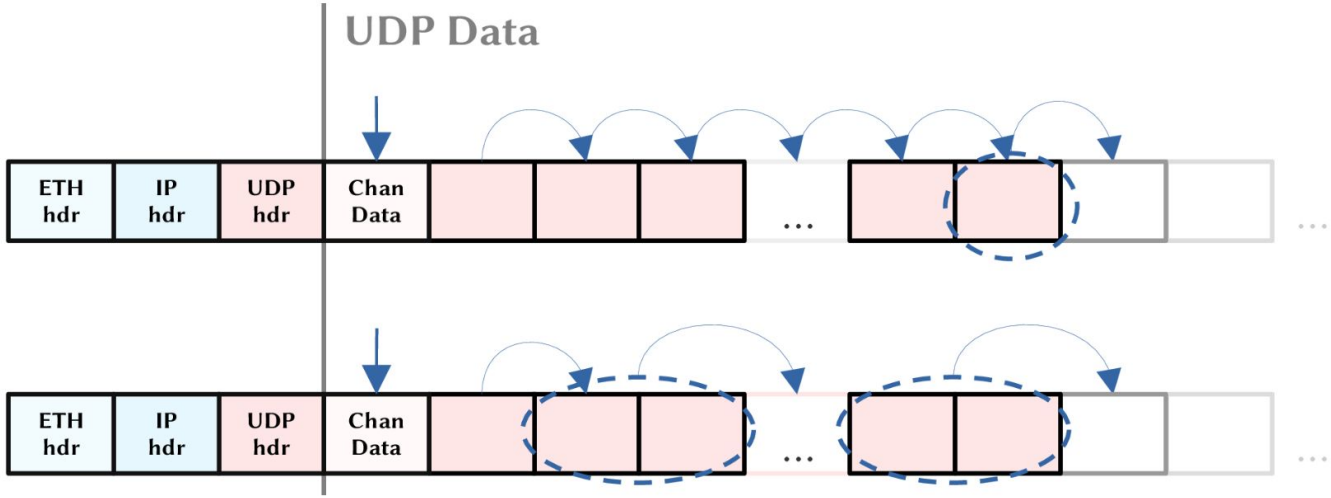
Architecture/Implementation



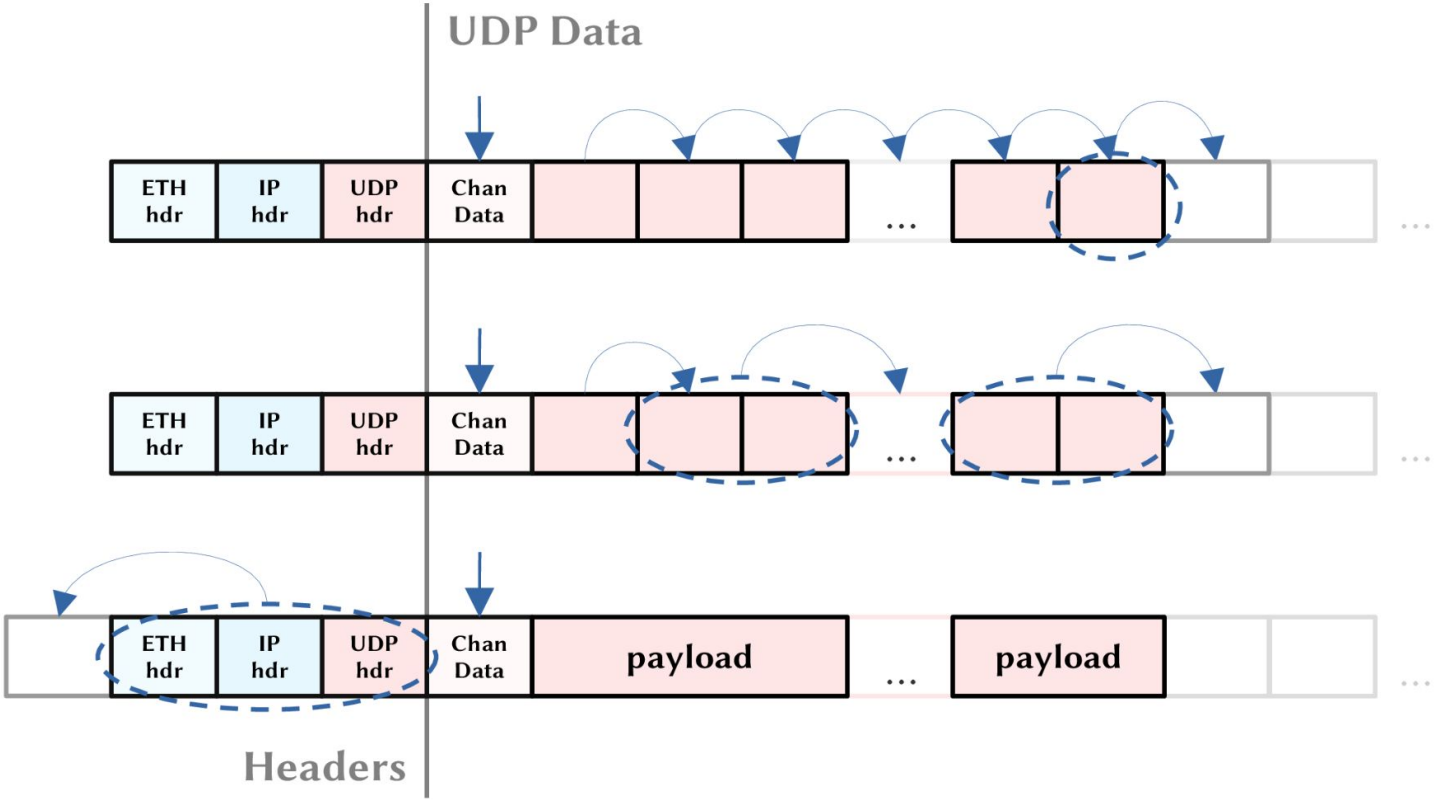
Challenge: Add/remove ChannelData header



Challenge: Add/remove ChannelData header



Challenge: Add/remove ChannelData header



Challenge: Update UDP checksum

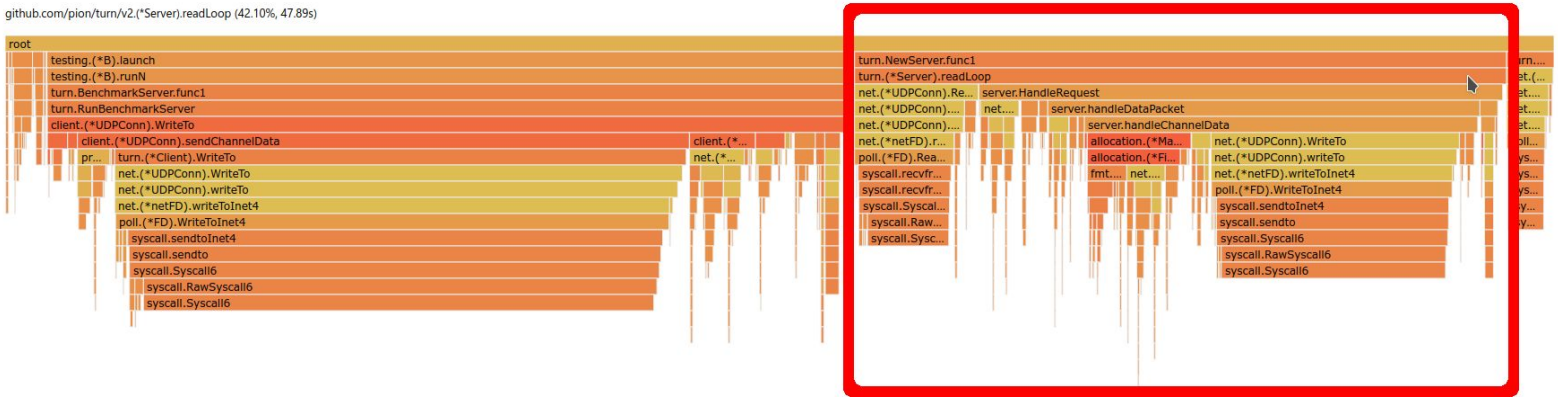
- Add/remove ChannelData header invalidates the UDP checksum
  - UDP checksum: pseudo header + UDP data

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()`

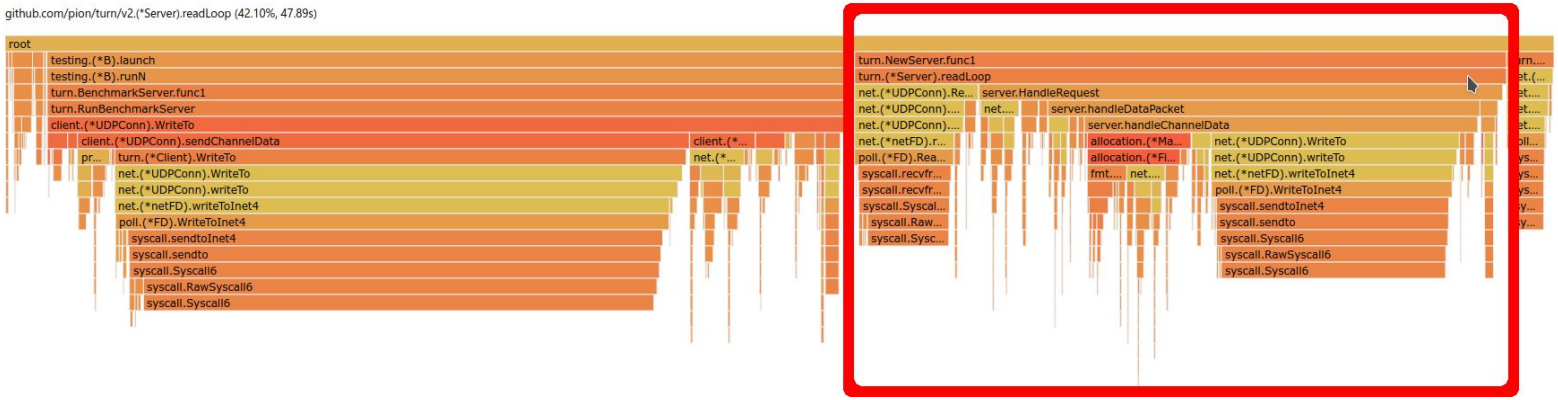
Evaluation: Internal TURN server performance

No offload

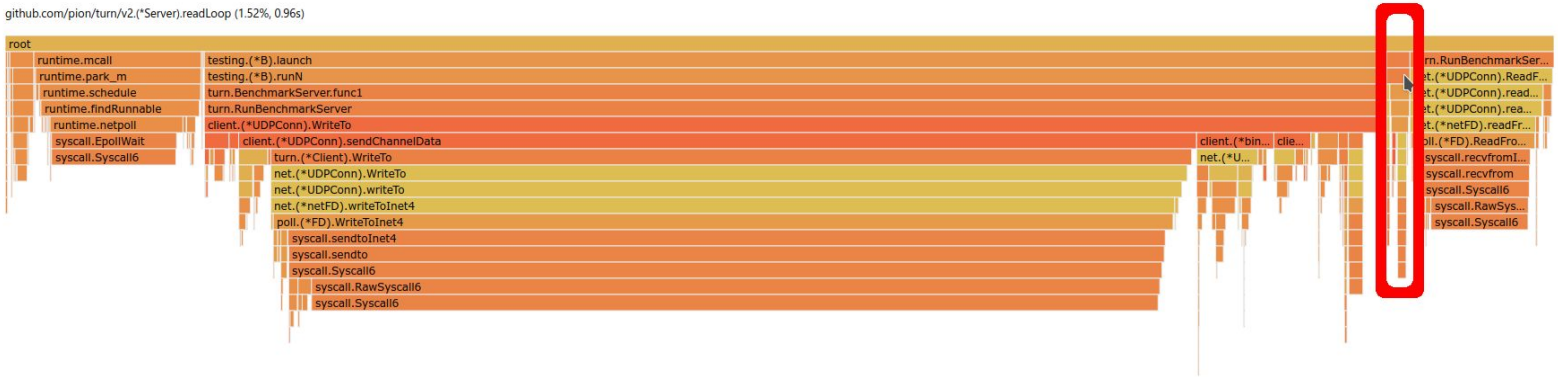


Evaluation: Internal TURN server performance

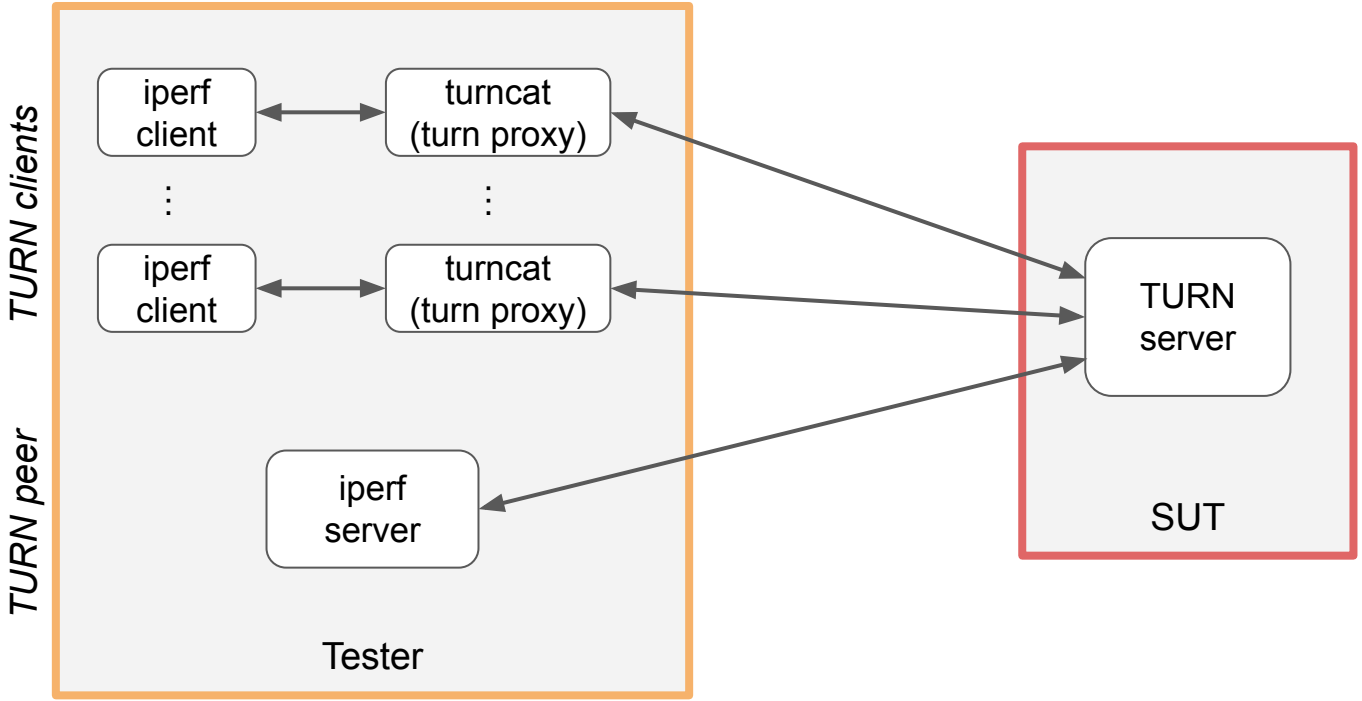
No offload



XDP offload



Evaluation: End-to-end testbed

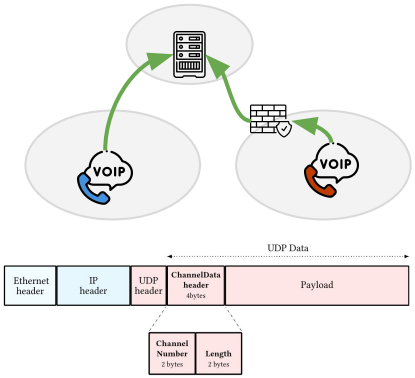


Evaluation: End-to-end results

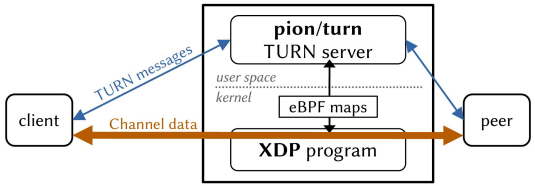
Implementation	Throughput [kpps]	Latency [ms] avg	Latency [ms] std dev	Jitter [ms]
1 thread	68.5	0.1795	0.2261	0.0265
4 threads	94.8	0.2492	0.3731	0.0351
XDP	134.3	0.0852	0.0994	0.0217
baseline	134.3	0.0386	0.0132	0.0086

Summary

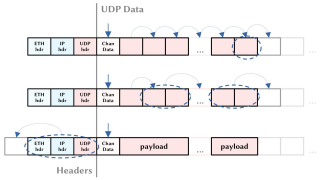
TURN



eBPF/XDP offload architecture



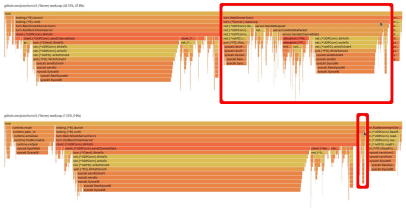
Challenges



UDP checksum

Monitoring

Results



Implementation	Throughput [kpps]	Latency [ms] avg	Latency [ms] std dev	Jitter [ms]
1 thread	68.5	0.1795	0.2261	0.0265
4 threads	94.8	0.2492	0.3731	0.0351
XDP	134.3	0.0852	0.0994	0.0217
baseline	134.3	0.0386	0.0132	0.0086

Code and artifacts available at

: [l7mp/turn/tree/server-ebpf-offload]