SCION Performance Analysis: Bittorrent over SCION

+ Short HSR Deployment Update

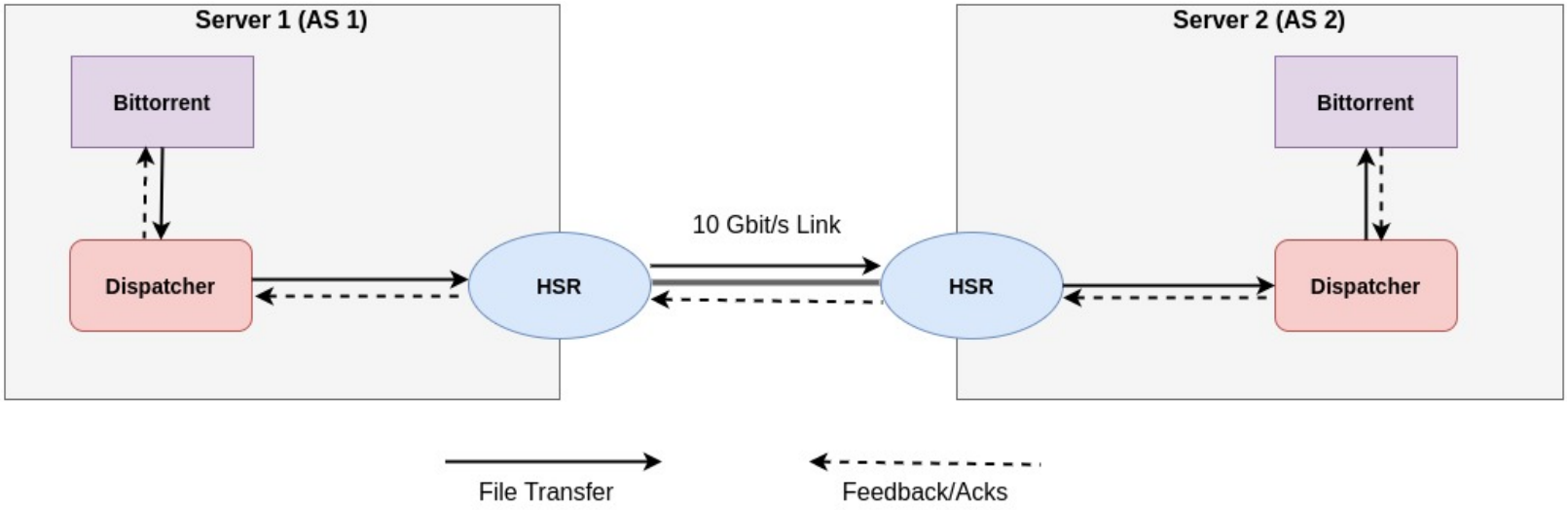
Marten Gartner, Martin Koppehel, David Hausheer, Thorben Krüger

2021

HSR Deployment Status & Next Steps

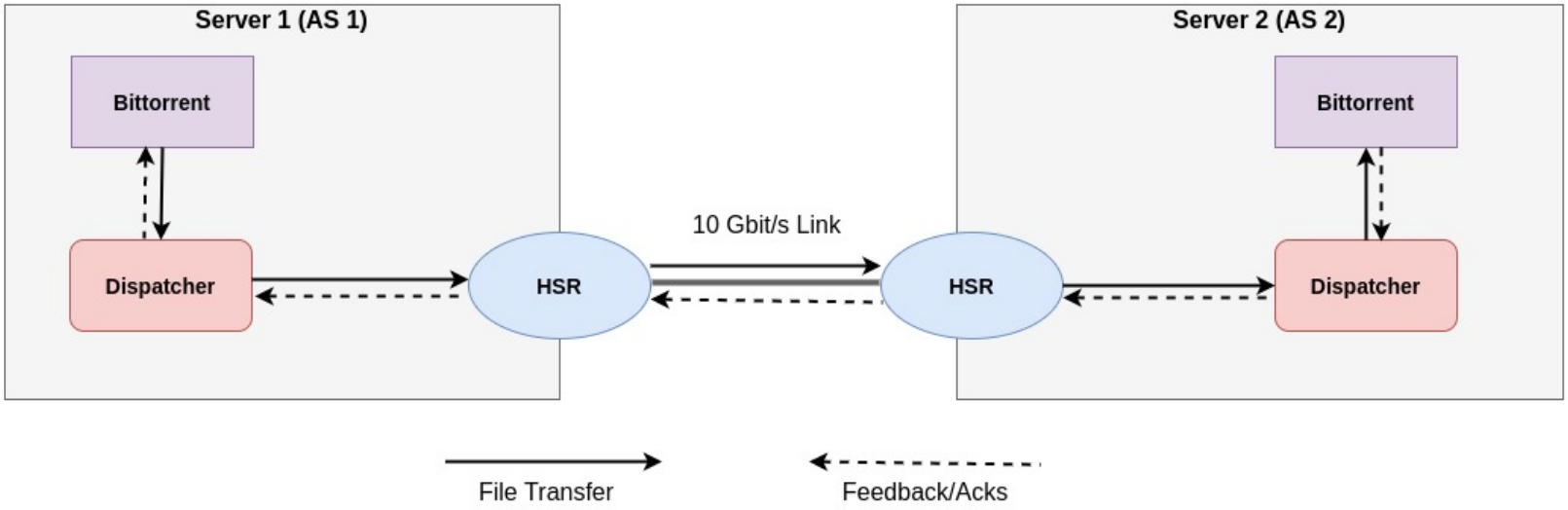
- 2x User ASes in Magdeburg
- 2x Servers in GEANT (PAR, HAM)
- 1x Server at SWITCH (GVA)
    - Missing SR-IOV Support => Add a separate mgmt interface to deploy HSR as PF (or use another machine)
- Next: Check further SCIONLab machines to deploy HSR
- Documentation: Complete installation guide in progress...

Bittorrent over SCION - Setup



- Target: Achieve 10Gbit/s bandwidth with Bittorrent over SCION
- Results: Achieved ~2.5Gbit/s bandwidth with HSR and Jumbo Frames
- With Bittorrent over TCP and parallel downloads ~8Gbit/s possible

Bottleneck: Dispatcher?



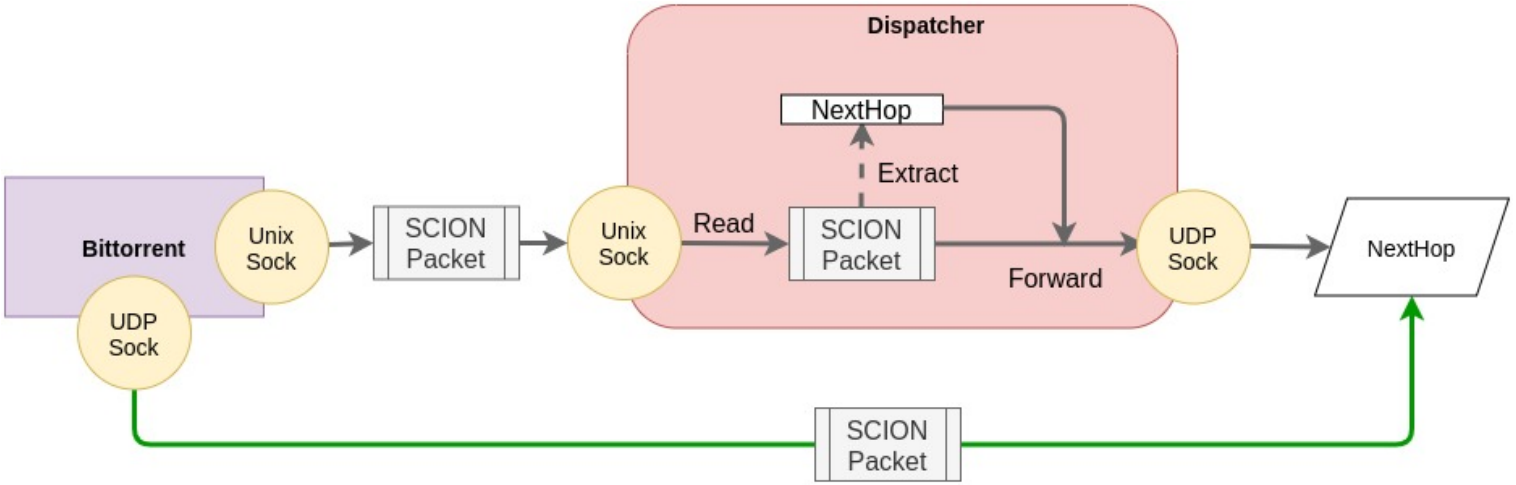
- Dispatcher handles all incoming/outgoing SCION packets
- Each packet goes through UDP Socket and Unix Socket
- In Summary: Each Packet goes 4x through linux kernel

Improving the Dispatcher?

- First idea was to bypass the dispatcher, like hercules does
- This approach only works for particular apps, which is absolutely fine
- Consequently: Adapt the dispatcher to improve all SCION apps

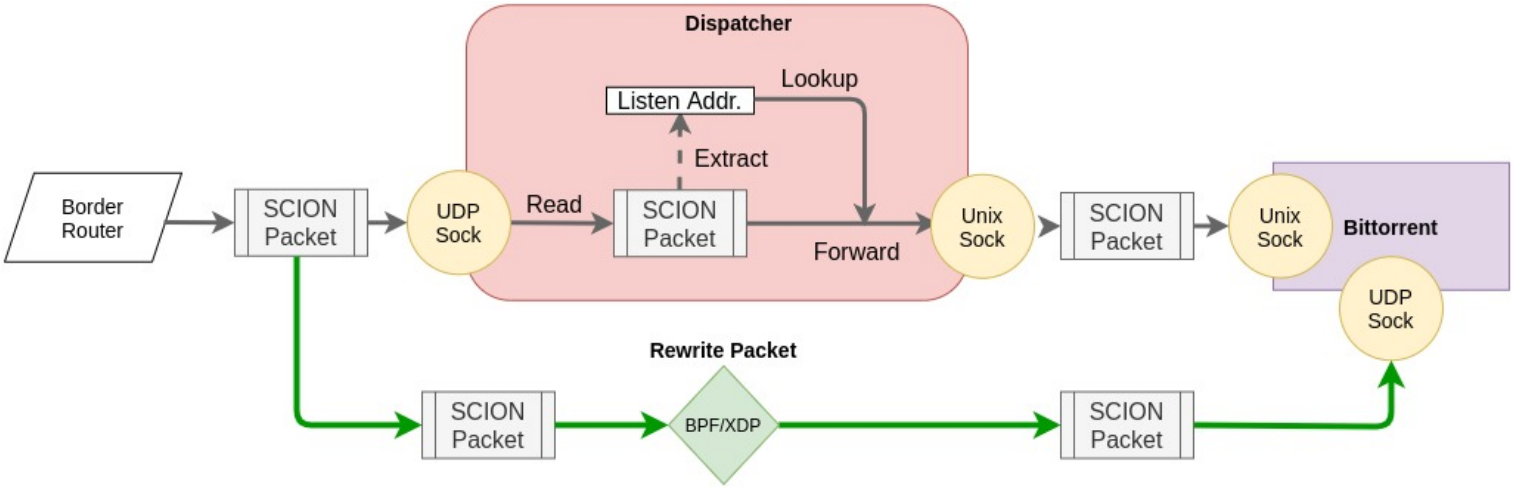
- Approach: Improve only the forwarding part
  - Receiver: Let apps open "normal" UDP ports and rewrite the incoming packets using BPF/XDP
  - Sender: Send SCION packets to nexthop underlay address (prob. border router)

Sender side: Use NextHop



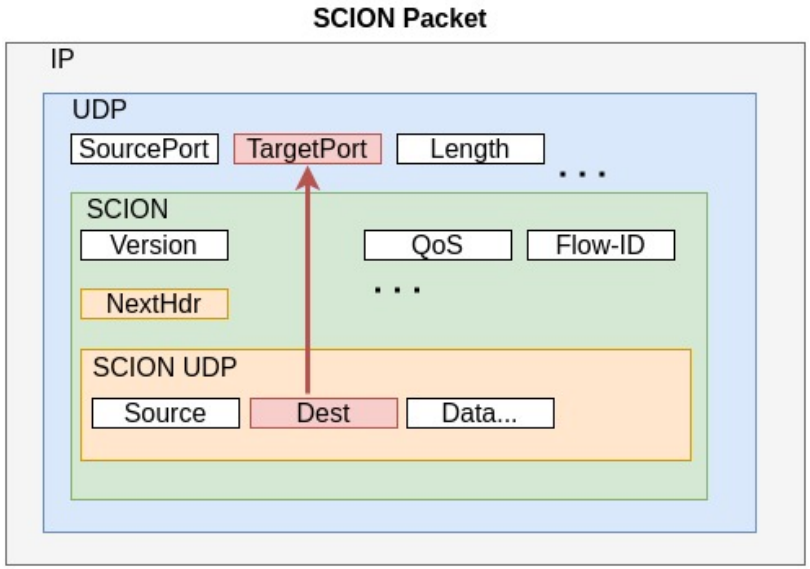
- SCION path contains nexthop field, which is used by the dispatcher
- The app can send to the nexthop directly

Receiver side: UDP Ports + BPF



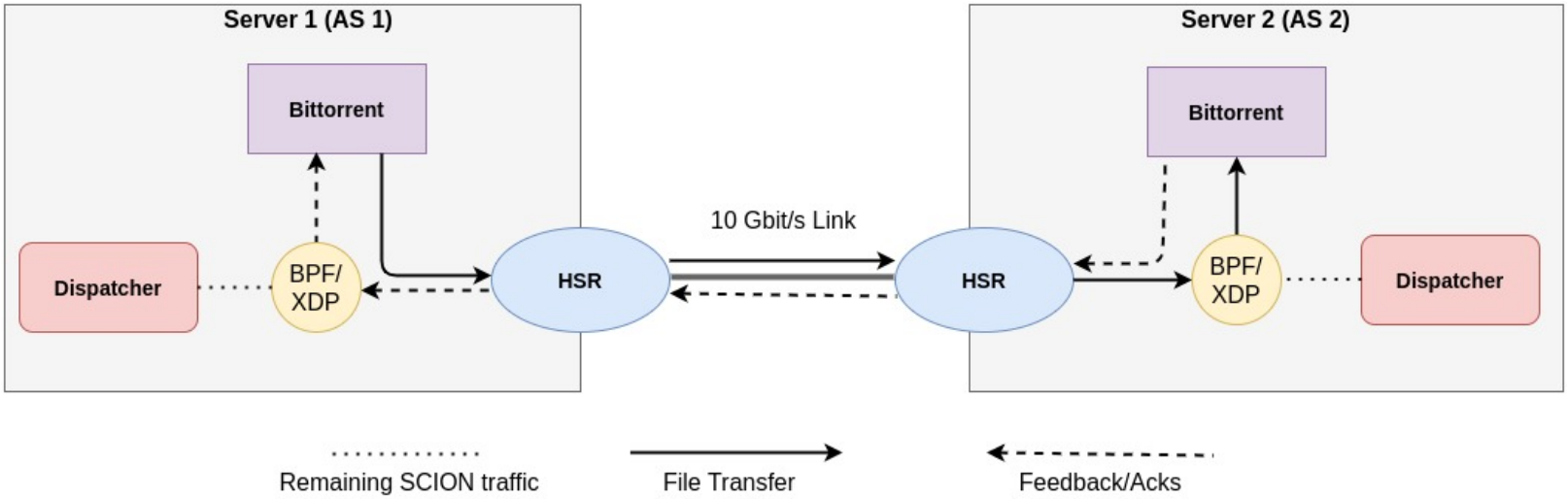
- App listens on UDP Port wrapped by SCION PacketConn
- All incoming packets with the target SCION L4 UDP Port are rewritten
- Only 1x through Kernel on each machine

Receiver side: Packet rewriting



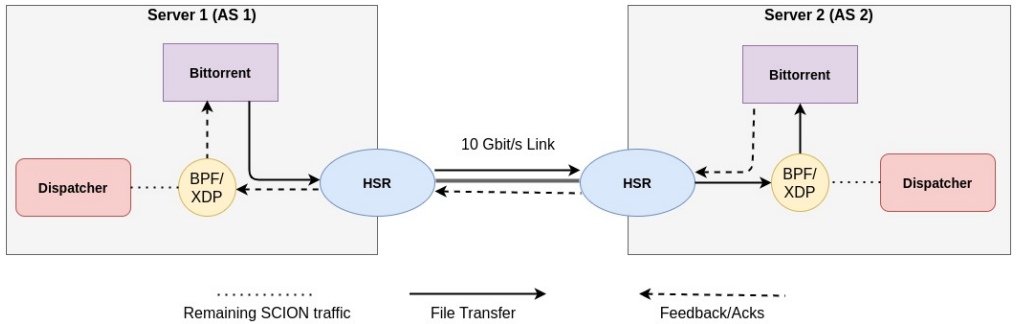
- BPF app opens packets containing the UDP TargetPort of the dispatcher
- Lookup L4 SCION UDP dest, write into UDP TargetPort

New SCION Data Flow



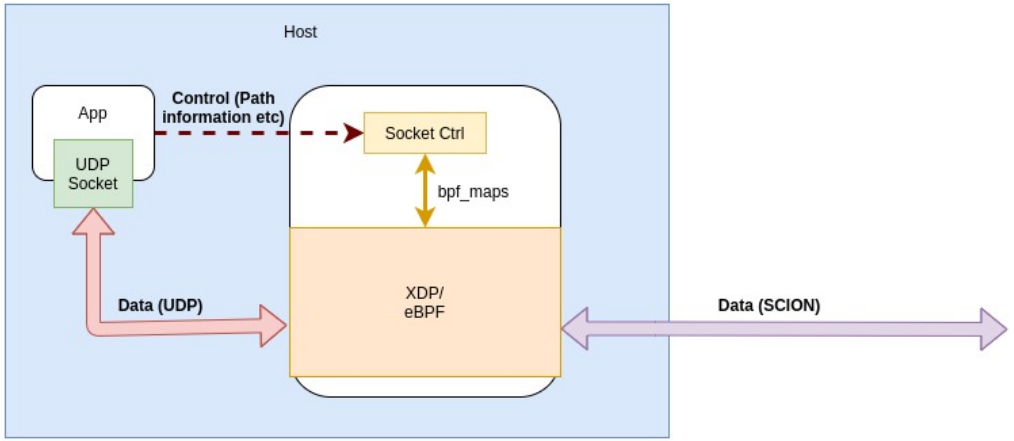
- Only Bittorrent specific SCION packets are changed
- Remaining SCION traffic remains untouched

SCION eBPF: Next tasks



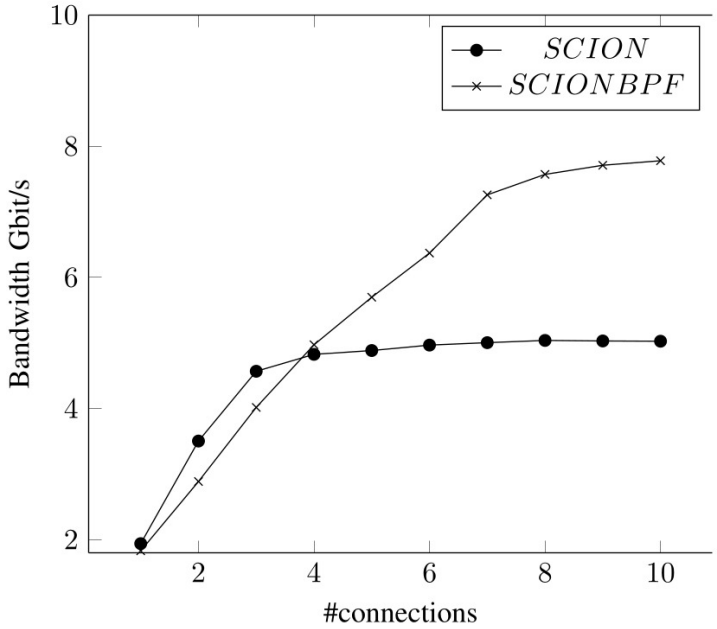
- Only forward communication, backwards must be implemented
- Implement API to be called from applications to listen on UDP ports
- Analyze (and if possible improve) SCION/UDP & SCION/QUIC performance
- Compare approach to current upstream implementation (performance, resource usage etc)

Idea for future: SCION eBPF Socket



- Remove the SCION packet de/encapsulation to the kernel
- Why? Something slows down this in the SCION implementation (blocking go routines)
- Could support different programming languages

First Results: Raw SCION/UDP



- Measurements done on 10Gbit/s link with HSR, Jumbo Frames and simple bandwidth tester

Next Steps:

- Fix SCION/QUIC packet sending (handshake error)
- Test with Multipath Bittorrent over SCION
- Continue with more experiments

- Investigate why single connections are limited to max 2Gbit/s
- Support multiple registered apps instead of hard coded port range
- Aim for bringing this prototype upstream once its a working application

Notes:

- Related issue from Anapaya for removing the dispatcher: https://github.com/scionproto/scion/issues/3961

- Code changes for SCION can be found here: https://github.com/martenwallewein/scion/tree/feature/udp-listen-dial
- Code changes for SCION-Apps can be found here: https://github.com/martenwallewein/scion-apps/tree/feature/listen-udp
- XDP code is currently in a private repo, because SCION packet parsing was adapted from hercules

Thanks for your attention!