



IETF Hackathon

IETF 114
24 July 2022
Philadelphia, Pennsylvania
Remote

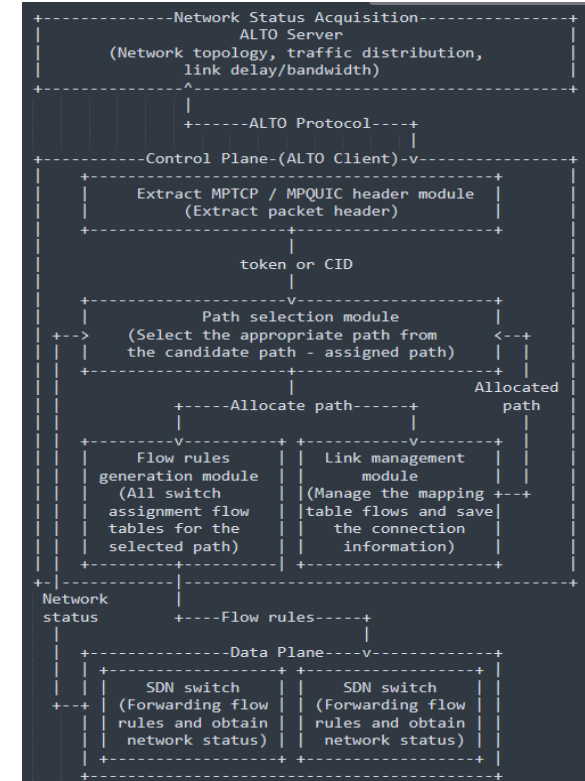
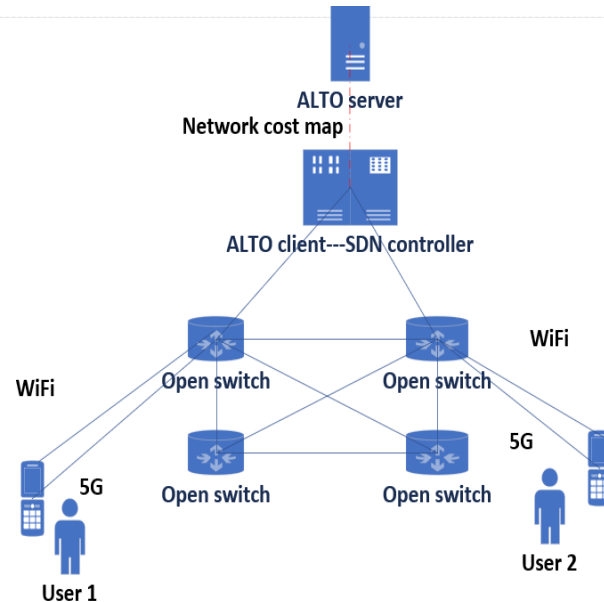


Hackathon Plan

- SDN controller helps host to select the path for traffic optimization in multipath
 - <https://datatracker.ietf.org/doc/draft-xing-alto-sdn-controller-aware-mptcp-mpquic/>
 - Default transmission control mode of MPTCP or MPQUIC in SDN only select one same path every time
- Controller extracts MPTCP or MPQUIC packet header to allocate MPTCP or MPQUIC packet to suitable transmission path according to the network cost indicators by ALTO

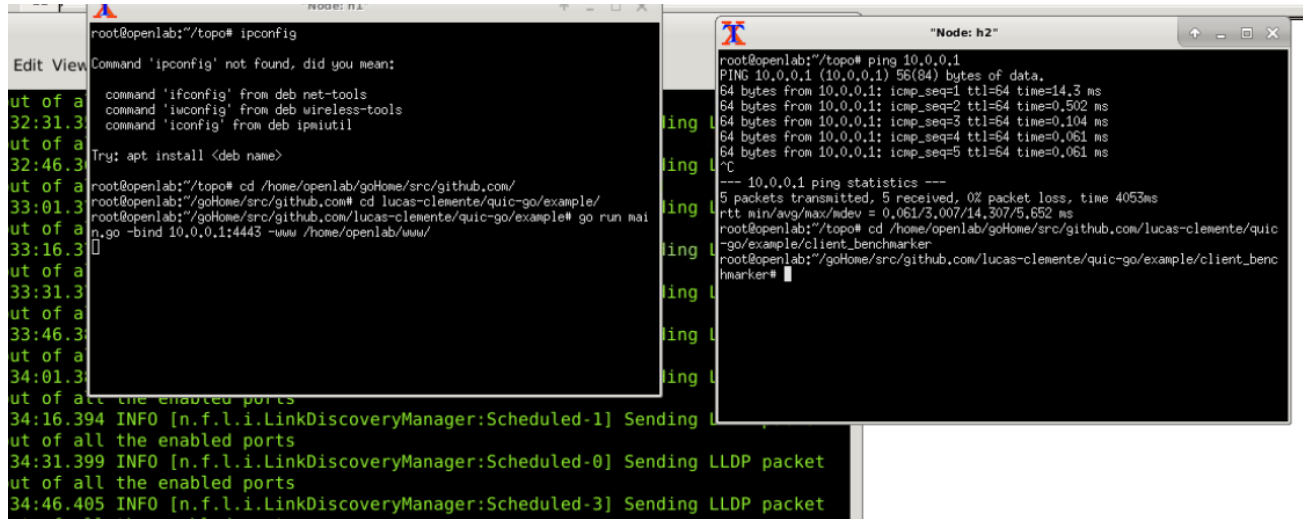
Hackathon Plan

- Implemented SDN-based MPTCP-aware and MPQUIC-aware transmission control using ALTO



Hackathon Plan

- 1.SDN controller(ALTO client): OpenDaylight
- 2.SDN server and client: MPQUIC-go \ MPTCP



The image shows two terminal windows. The left window, titled 'root@openlab:~/topo#', shows the user attempting to run 'ipconfig', which is not found. They then search for it in various Debian repositories and finally install it using 'apt install <deb name>'. The right window, titled 'Node: h2', shows the user running a ping command to 10.0.0.1. The output shows five successful ping requests with varying times, followed by a summary of the statistics: 5 packets transmitted, 5 received, 0% packet loss, and a total time of 4053ms.

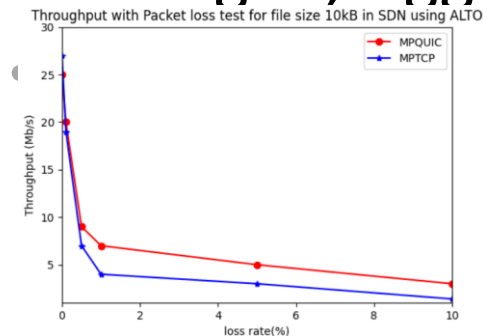
```
root@openlab:~/topo# ipconfig
Command 'ipconfig' not found, did you mean:
  command 'ifconfig' from deb net-tools
  command 'iwconfig' from deb wireless-tools
  command 'iconfig' from deb ipmiutil
Try: apt install <deb name>

root@openlab:~/topo# cd /home/openlab/goHome/src/github.com/
root@openlab:~/goHome/src/github.com# cd lucas-clemente/quic-go/example/
root@openlab:~/goHome/src/github.com/lucas-clemente/quic-go/example# go run mai
n.go -bind 10.0.0.1:4443 -www /home/openlab/www/

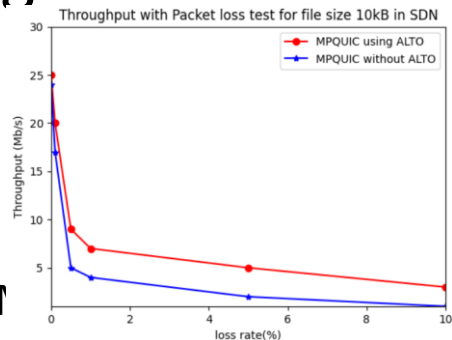
root@openlab:~/topo# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data:
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=14.3 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.502 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.104 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.061 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4053ms
rtt min/avg/max/mdev = 0.061/3.007/14.307/5.652 ms
root@openlab:~/topo# cd /home/openlab/goHome/src/github.com/lucas-clemente/quic
-go/example/client_benchmark
root@openlab:~/goHome/src/github.com/lucas-clemente/quic-go/example/client_benc
hmarker#
```

Hackathon Result

- The throughput of MPQUIC/MPTCP using ALTO is higher than without ALTO in SDN especially in poor network.
- QUIC's new characteristics: 0-RTT connections, Forward error correction, Adaptive congestion control ...
- IPv6 new characteristics: simple header, No-NAT changed, Aggregation-routing ...



SDN-based MPTCP and I



Control

What we learned

- How to deploy ALTO in SDN.
- What is the different for ALTO deployment with SDN in IPv6 and IPv4.

Wrap Up

Conclusion:

The transmission control method proposed improves the throughput using ALTO by about **3 times compared to the default transmission control method.**

References:

- [1] MPQUIC source
<https://github.com/lucas-clemente/quic-go>
- [2] traffic measurement
<http://mahimahi.mit.edu/>

Contact:

xzynet@gmail.com