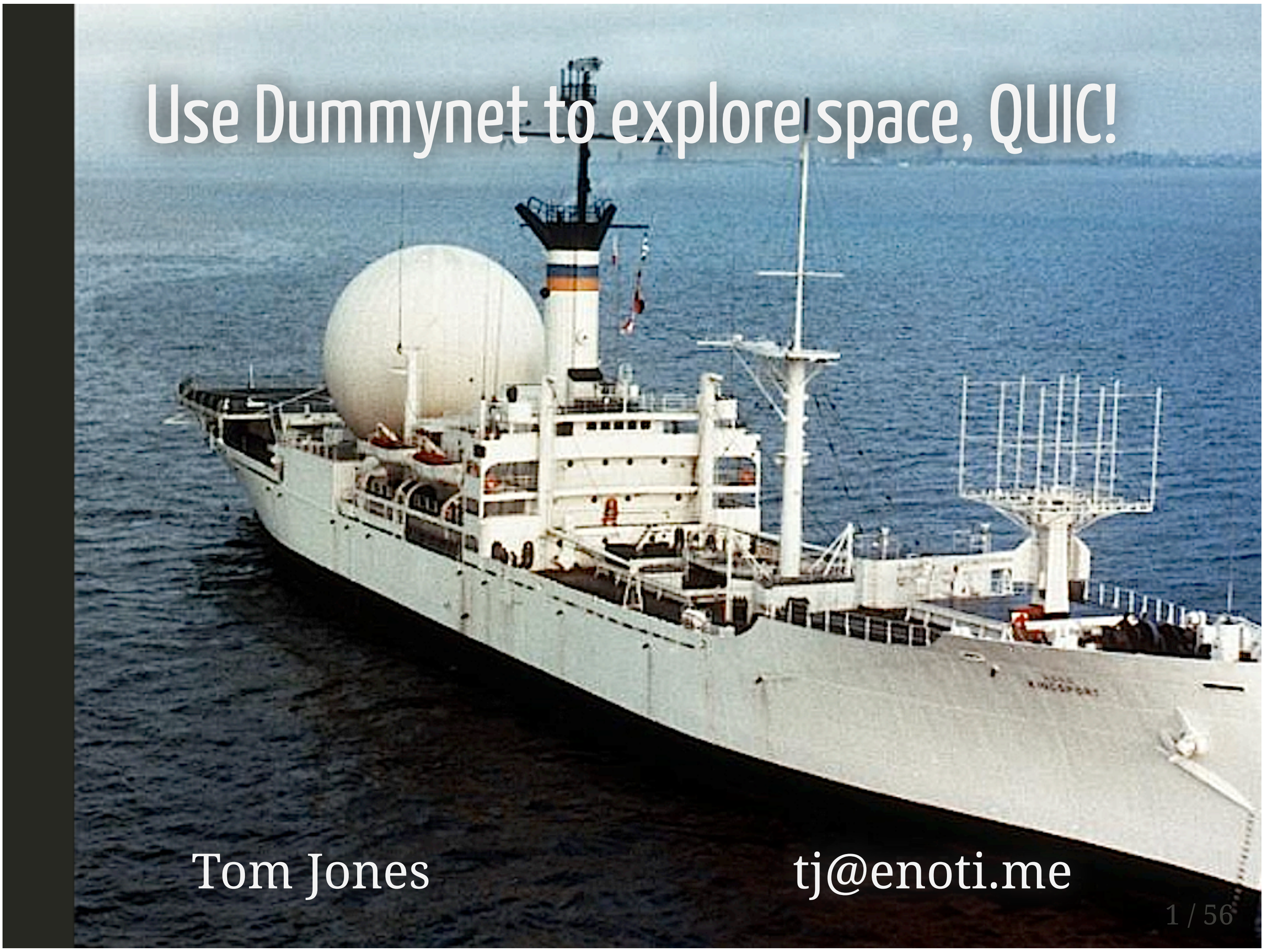
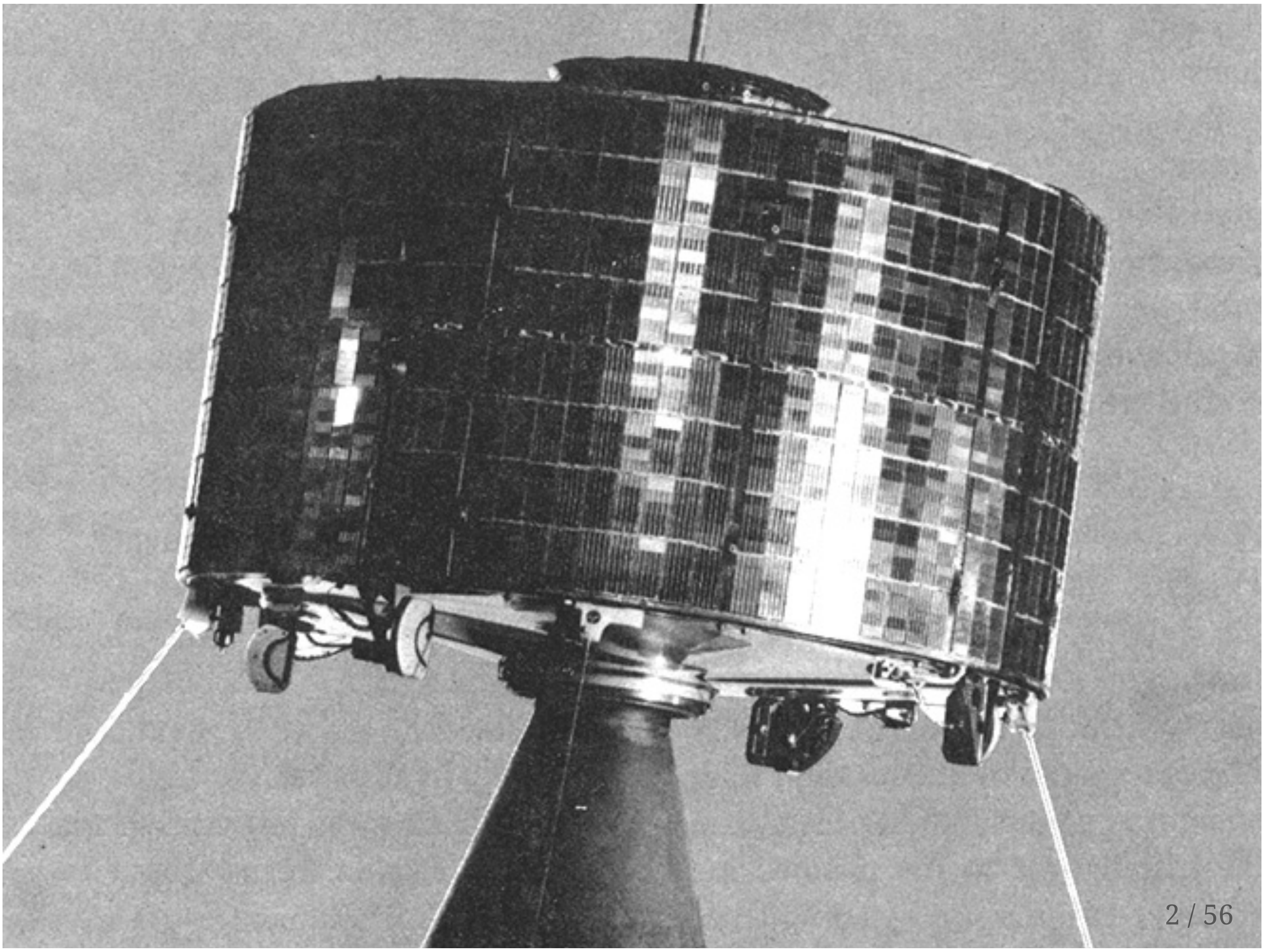
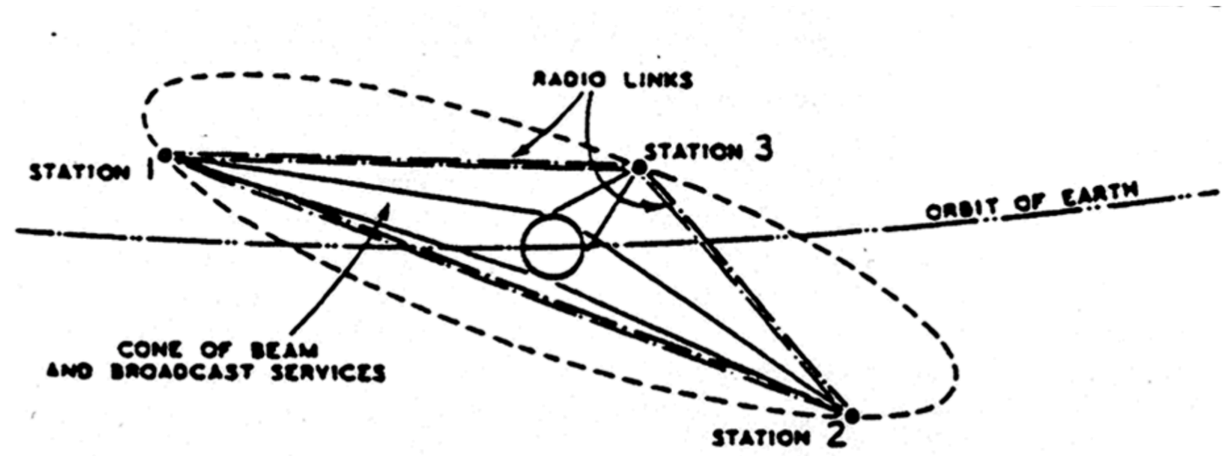
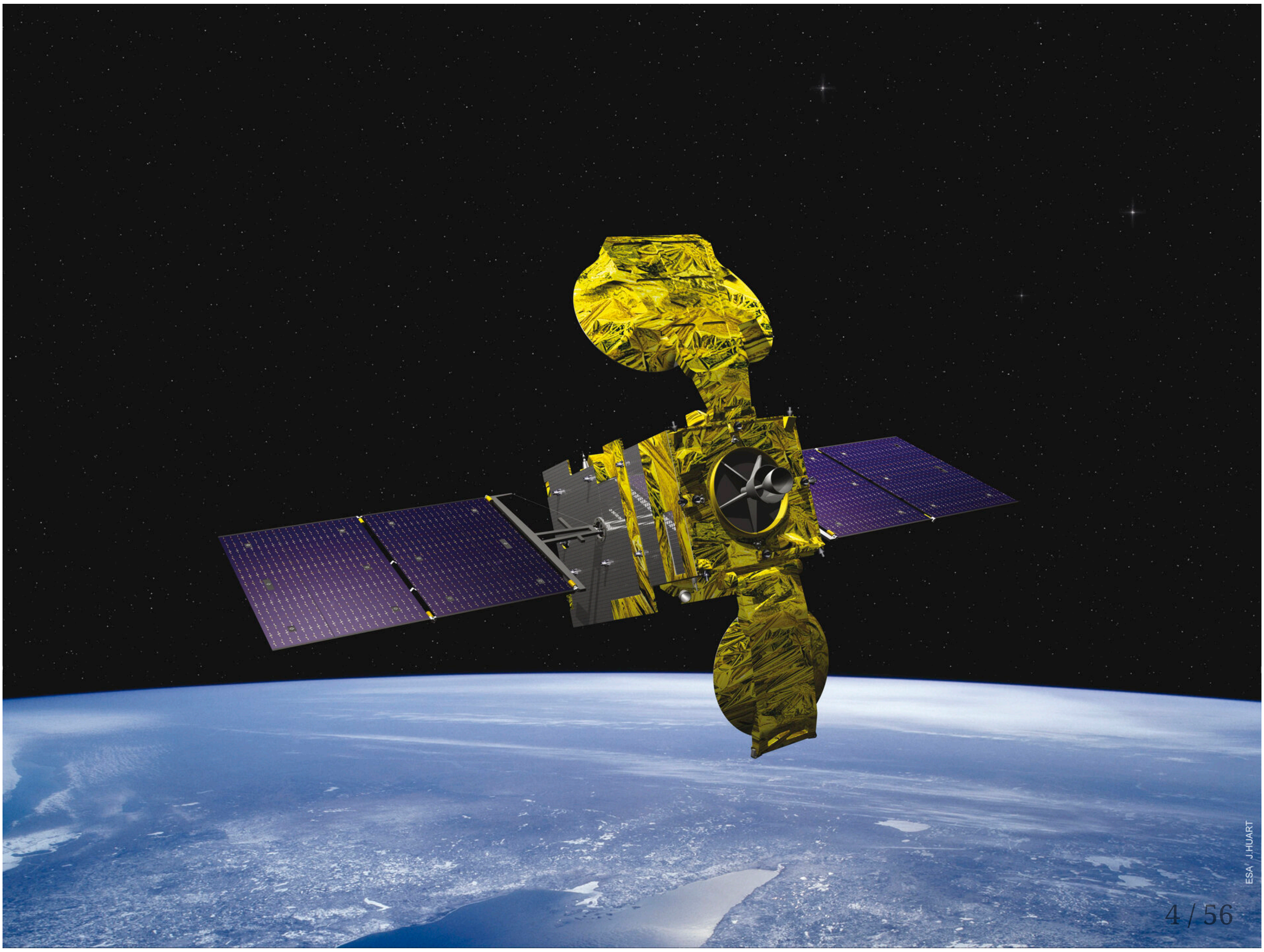

Use Dummynet to explore space, QUIC!
Tom Jones
tj@enoti.me
1 / 56


2 / 56

Three altitudes of orbit are used for satellite networks



- LEO (ISS, hubble, starlink, iridiumn and others)
- MEO (GPS, O3B)
- GEO (all the vsat services you can buy today)


4 / 56

whoami

- Internet Engineer
  - I do Internet and Transport Protocol design and implementation
  - And I write Standards in the IETF (to blame for RFC8304 and RFC8899)
  - I like to hack on the FreeBSD Network stack
  - I try to make the Internet better
- One eigth of BSDNow hosting team
- For the last few years I have been working on making sure QUIC works well in satellite networks
- Some relevant current standards work:
  - draft-jones-transport-for-sattellite
  - draft-kuhn-quic-0rtt-bdp
  - draft-fairhurst-quic-ack-scaling

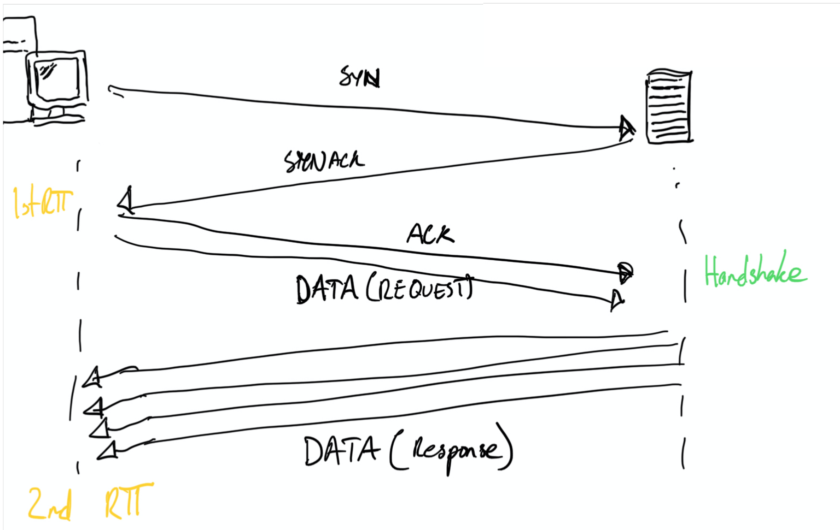
This Talk

- This is an introduction to using `dummynet`
- Wrapped in the state of the art of transport protocol design
- with lots of digressions into Internet engineering

Transport protocols and the web

- HTTP runs the Web
- We first had http0.9, then http1.0 and finally http1.1
- http1 had issues with connection set up latency
  - to improve page loads browsers make many (sometimes 6) http connections to a server
- In 2012 google released spdy, which evolved in the IETF into http2
- H2 added
  - Multistreaming to http
  - 0RTT connections with TCP Fast Open
  - Flow control
  - when it helped, it helped a lot, and when it didn't, it hurt
- **Practically all web traffic has been http over TCP to date**

Speeding up HTTP



Things in GEO orbit are really far away

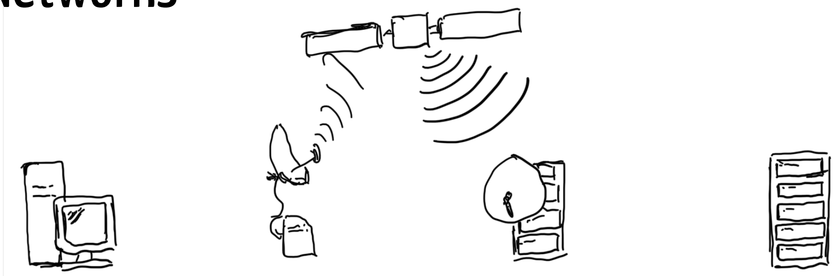
RFC2488

```
Many communications satellites are located at Geostationary Orbit
(GSO) with an altitude of approximately 36,000 km [Sta94].  At this
altitude the orbit period is the same as the Earth's rotation period.
Therefore, each ground station is always able to "see" the orbiting
satellite at the same position in the sky.  The propagation time for
a radio signal to travel twice that distance (corresponding to a
ground station directly below the satellite) is 239.6 milliseconds
(ms) [Mar78].  For ground stations at the edge of the view area of
the satellite, the distance traveled is 2 x 41,756 km for a total
propagation delay of 279.0 ms [Mar78].  These delays are for one
ground station-to-satellite-to-ground station route (or "hop").
Therefore, the propagation delay for a message and the corresponding
reply (one round-trip time or RTT) could be at least 558 ms.
```

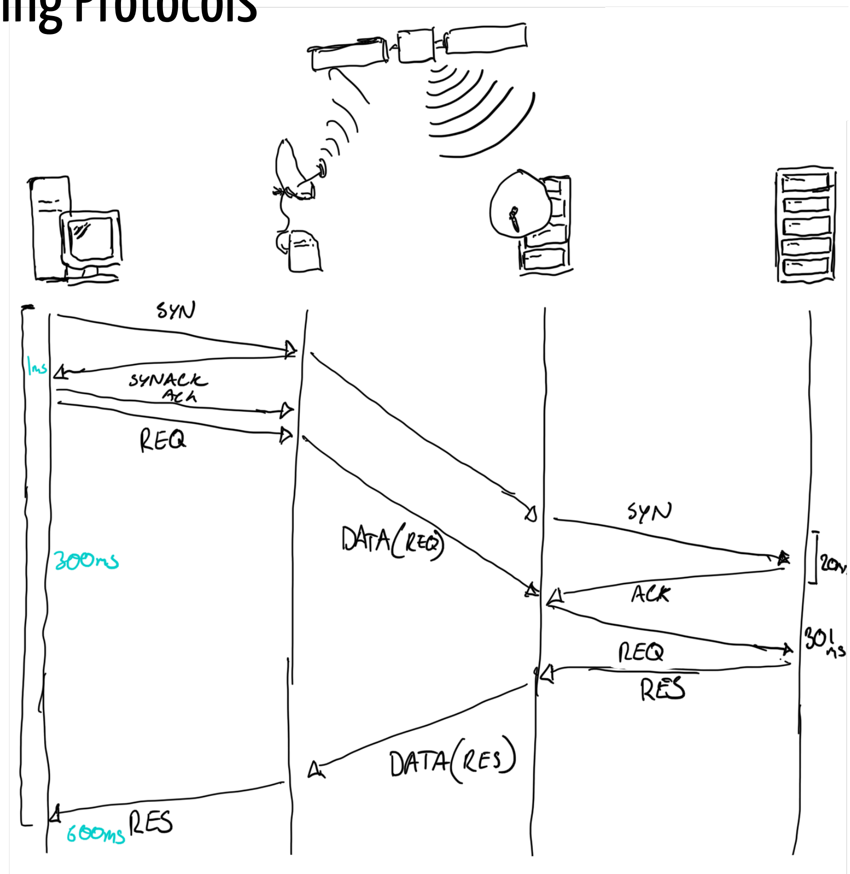
Why does the delay hurt so much?

- Anything that requires a round trip will have greatly inflated time steps
- Reno Based congestion control grows as a function of the RTT
- Auto tuned buffers grow as a function of the RTT

Satellite Networks



Accelerating Protocols



11 / 56

QUIC

- QUIC is a next generation transport protocol published by the IETF in 2021
  - The transport protocol is defined by RFC9000 (QUIC Transport)
  - and fully across RFC8999 (invariants), RFC9001 (TLS) and RFC9002 (congestion control)
  - work on the protocol continues in the IETF QUIC working group on extensions and a 'fast' process to QUICv2

12 / 56

QUIC

- UDP used as a substrate
- Multistreaming
- Transport protocol fully authenticated and encrypted on the wire
- 0RTT connection resumption
- native connection migration and load balancing
- the protocol enables modern congestion control and loss recovery mechanisms
- deliberately design to resist ossification
    - greases fields to make finger printing hard
- implementations in userspace, with many open source implementations available in lots of languages
- one kernel ready implementation (msquic from Microsoft)

13 / 56

As a network operator that means

- **A LOT** more UDP traffic
  - that messes with fields to make it hard to pin down
- minimum metadata is available to measure network health
- interception boxes "don't work", so no
  - MSS rewrites
  - HTTP proxies
  - the end of Performance Enhancing Proxies
- less visibility into your network traffic

I think you just have to accept this new reality

Impetus

In 2019 there was a lot of fear about QUIC in the GEO satellite community.

The perception was that TCP without a PEP on a GEO link is not enjoyable to use. These satellite cost a lot of money, if the world moves to quic did we just fling that money into space?

The QUIC standardisation process needed to include satellite viewpoints.

Satellite and QUIC

- QUIC can't be accelerated (see authenticated and encrypted)
- DNS acceleration is still possible (until we see wide DOH deployment)
- Is this the end for GEO satellite service?


16 / 56

Our hook up

- 10/2 GEO Internet service from Avanti
- on a 'engineering link'
- reality 8.5Mbit/s down, 1.5Mbit/s up
- typically ~610ms of delay (ping google.com)
- delay varies up to 8 seconds (8000ms, try playing quake on that!)
- We have a limited SLA, at some point we can 'use up' our Internet allowance and then science stops

Testing protocols

- because we have a limited service, most testing of protocol changes needs to happen in emulation
- emulation allows us to model networks we don't have too!
- For network emulation, dummynet on FreeBSD is the best™ choice

```
NAME
     dummynet - traffic shaper, bandwidth manager and delay emulator

DESCRIPTION
     The dummynet system facility permits the control of traffic going through
     the various network interfaces, by applying bandwidth and queue size
     limitations, implementing different scheduling and queue management
     policies, and emulating delays and losses.

     The user interface for dummynet is implemented by the ipfw(8) utility, so
     please refer to the ipfw(8) manpage for a complete description of the
     dummynet capabilities and how to use it.

...

SEE ALSO
     setsockopt(2), if_bridge(4), ip(4), ipfw(8), sysctl(8)

HISTORY
     The dummynet facility was initially implemented as a testing tool for TCP
     congestion control by Luigi Rizzo <luigi@iet.unipi.it>, as described on
     ACM Computer Communication Review, Jan.97 issue.  Later it has been
     modified to work at the IP and bridging levels, integrated with the
     ipfw(4) packet filter, and extended to support multiple queueing and
     scheduling policies.
```

Dummynet

- **traffic shaper**
  - limiting the bandwidth that can be consumed by certain applications
- **bandwidth manager**
  - process of measuring and controlling the communications on a network link, to avoid filling the link to capacity or overfilling the link
- **delay emulator**
  - the act of introducing a device to a test network (typically in a lab environment) that alters packet flow in such a way as to mimic the behavior of networks

Dummynet History

- First proposed in 1997 by Luigi Rizzo
- bridging
- ipfw integration
- Packet scheduling and AQM
- SCTP NAT
- MAC layer emulation

Dummynet

The dummynet interface to ipfw is via the pipe mechanism:

From ipfw there are two interfaces to dummynet:

- pipes
- queues

We add a pipe rule to our ipfw rule set, packets that match vanish into the pipe and pop back out at some later time.

Dummynet

http://info.iet.unipi.it/~luigi/dummynet/ example:

```
simulate an ADSL link to the moon:

ipfw add pipe 3 out
ipfw add pipe 4 in
ipfw pipe 3 config bw 128Kbit/s queue 10 delay 1000ms
ipfw pipe 4 config bw 640Kbit/s queue 30 delay 1000ms
```

22 / 56

Characterising networks

When we talk about computer networks we tend to label them with 4 properties:

- **Delay:** is the amount of time it takes for a packet to propagate through the network
- **Bandwidth:** is the number of packets a second a network can process
- **Buffering:** is the networks ability to accommodate bursts of traffic
  - too little and throughput suffers (actual transmitted packets)
  - too much and latency suffers (delay increases)
- **Packet loss:** anything other than random packet loss on a link is going to make it unenjoyable to use

Characterising networks: Performing measurements

- When we perform measurements we need to take multiple measurements across the expected use and work from an average.
- Before looking at anything new, use an existing thing to set a baseline to compare to
- You environment will have its own peculiarities, test your measurements against your intuition and understanding of design and configurations limitations
  - if you are getting 11Gbit/s on your home network might be measuring localhost
  - if your ping is 28ms you aren't using the satellite link
  - if everything matches your expectations be suspicious

24 / 56

Characterising Networks: Delay

- We measure delay in seconds, for almost all situations milliseconds (ms) or 1000th of a second is precise enough without being silly
- We measure the delay of a network using plain old `ping`
- Delay has a large impact on anything that needs to feel interactive, too much or too much delay variation and it becomes really difficult to predict what will happen (try using ssh over 4G while downloading and see!)
- With a good number of samples we can get an idea of the networks average, min, max and variation
- Delay will vary with packet scheduling in hardware, link layer loss and buffer occupancy.
- Real networks have diurnal patterns of use, to get a real picture of a network it can be important to measure across each hour of the day for a long period to detect patterns.

Characterising Networks: Delay

```
$ ping -c 100 eurobsdcon.org
ping eurobsdcon.org (5.9.139.66): 56 data bytes
64 bytes from 5.9.139.66: icmp_seq=0 ttl=54 time=42.269 ms
64 bytes from 5.9.139.66: icmp_seq=1 ttl=54 time=44.200 ms
64 bytes from 5.9.139.66: icmp_seq=2 ttl=54 time=115.496 ms
64 bytes from 5.9.139.66: icmp_seq=3 ttl=54 time=44.003 ms
64 bytes from 5.9.139.66: icmp_seq=4 ttl=54 time=42.426 ms
64 bytes from 5.9.139.66: icmp_seq=5 ttl=54 time=42.451 ms
64 bytes from 5.9.139.66: icmp_seq=6 ttl=54 time=42.417 ms
64 bytes from 5.9.139.66: icmp_seq=7 ttl=54 time=43.907 ms
64 bytes from 5.9.139.66: icmp_seq=8 ttl=54 time=44.183 ms
64 bytes from 5.9.139.66: icmp_seq=9 ttl=54 time=44.570 ms
64 bytes from 5.9.139.66: icmp_seq=10 ttl=54 time=43.833 ms
64 bytes from 5.9.139.66: icmp_seq=11 ttl=54 time=43.356 ms
...
64 bytes from 5.9.139.66: icmp_seq=97 ttl=54 time=44.246 ms
64 bytes from 5.9.139.66: icmp_seq=98 ttl=54 time=45.132 ms
64 bytes from 5.9.139.66: icmp_seq=99 ttl=54 time=44.137 ms
--- eurobsdcon.org ping statistics ---
100 packets transmitted, 100 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 41.982/45.220/115.496/8.470 ms
```

26 / 56

Characterising Networks: Bandwidth

- We measure bandwidth in bits per second, we normally manage a number in the millions (Mega) or billions (Giga)
- There are lots of tools for measuring capacity (netperf, iperf3, …)
- I ♡ iperf3
  - it can benchmark TCP, UDP and SCTP
  - can report information in JSON
  - offers single shot server modes for testing
  - frustratingly it defaults to measuring from client to server

27 / 56

Characterising Networks: Bandwidth

```
$ iperf3 -c iperf3.eurobsdcon.org -R
Connecting to host iperf3.eurobsdcon.org, port 5201
Reverse mode, remote host iperf3.eurobsdcon.org is sending
[  7] local 192.168.1.115 port 62263 connected to 79.2.17.13 port 5201
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec  3.07 MBytes  25.7 Mbits/sec
[  7]   1.00-2.00   sec  3.48 MBytes  29.2 Mbits/sec
[  7]   2.00-3.00   sec  3.49 MBytes  29.2 Mbits/sec
[  7]   3.00-4.00   sec  3.48 MBytes  29.2 Mbits/sec
[  7]   4.00-5.00   sec  3.48 MBytes  29.2 Mbits/sec
[  7]   5.00-6.00   sec  3.20 MBytes  26.8 Mbits/sec
[  7]   6.00-7.00   sec  3.44 MBytes  28.8 Mbits/sec
[  7]   7.00-8.00   sec  3.48 MBytes  29.2 Mbits/sec
[  7]   8.00-9.00   sec  3.29 MBytes  27.6 Mbits/sec
[  7]   9.00-10.00  sec  3.48 MBytes  29.2 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-10.05  sec  34.6 MBytes  28.9 Mbits/sec   32             sender
[  7]   0.00-10.00  sec  33.9 MBytes  28.4 Mbits/sec                  receiver

iperf Done.
```

28 / 56

Characterising Networks: Bandwidth

```
$ iperf3 -c iperf3.eurobsdcon.org
Connecting to host iperf3.eurobsdcon.org, port 5201
[  7] local 192.168.1.115 port 62268 connected to 79.2.17.13 port 5201
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec   128 KBytes  1.04 Mbits/sec
[  7]   1.00-2.00   sec  3.51 KBytes  28.8 Kbits/sec
[  7]   2.00-3.00   sec   339 KBytes  2.79 Mbits/sec
[  7]   3.00-4.00   sec   396 KBytes  3.25 Mbits/sec
[  7]   4.00-5.00   sec   393 KBytes  3.22 Mbits/sec
[  7]   5.00-6.00   sec   399 KBytes  3.27 Mbits/sec
[  7]   6.00-7.01   sec   201 KBytes  1.64 Mbits/sec
[  7]   7.01-8.00   sec   344 KBytes  2.83 Mbits/sec
[  7]   8.00-9.00   sec   331 KBytes  2.70 Mbits/sec
[  7]   9.00-10.00  sec   427 KBytes  3.51 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-10.00  sec  2.89 MBytes  2.43 Mbits/sec                  sender
[  7]   0.00-10.18  sec  2.86 MBytes  2.36 Mbits/sec                  receiver

iperf Done.
```

Characterising Networks: Bandwidth

- For UDP measurements iperf3 tries to send at a rate and sees what happens
- Default rate is 1 Mbit/s

```
$ iperf3 -c iperf3.eurobsdcon.org -u
Connecting to host iperf3.eurobsdcon.org, port 5201
[  7] local 192.168.1.115 port 58001 connected to 79.217.13 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  7]   0.00-1.00   sec   129 KBytes  1.05 Mbits/sec  91
...
[  7]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec  90
[  7]   9.00-10.00  sec   129 KBytes  1.05 Mbits/sec  91


- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-10.00  sec  1.25 MBytes  1.05 Mbits/sec  0.000 ms  0/906 (0%)  sender
[  7]   0.00-10.03  sec  1.25 MBytes  1.05 Mbits/sec  0.527 ms  0/906 (0%)  receiver

iperf Done.
```

- You probably want iperf3 to send at a higher rate (-b flag)
- You need to get the client and server logs and compare the rate of traffic that actually arrived

Characterising Networks: Bandwidth

```
$ iperf3 -c iperf3.eurobsdcon.org -u  -b 10M --get-server-output
Connecting to host iperf3.eurobsdcon.org, port 5201
[  7] local 192.168.1.115 port 61050 connected to 79.2.17.13 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  7]   0.00-1.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  7]   1.00-2.00   sec  1.19 MBytes  9.99 Mbits/sec  863
[  7]   2.00-3.00   sec  1.19 MBytes  10.0 Mbits/sec  864
[  7]   3.00-4.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  7]   4.00-5.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  7]   5.00-6.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  7]   6.00-7.00   sec  1.19 MBytes  10.0 Mbits/sec  864
[  7]   7.00-8.00   sec  1.19 MBytes  10.0 Mbits/sec  863
[  7]   8.00-9.00   sec  1.19 MBytes  9.99 Mbits/sec  863
[  7]   9.00-10.00  sec  1.19 MBytes  10.0 Mbits/sec  863
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-10.00  sec  11.9 MBytes  10.0 Mbits/sec  0.000 ms  0/8632 (0%)  sender
[  7]   0.00-11.15  sec  4.51 MBytes  3.39 Mbits/sec  3.357 ms  5368/8632 (62%)  receive

...more...
```

31 / 56

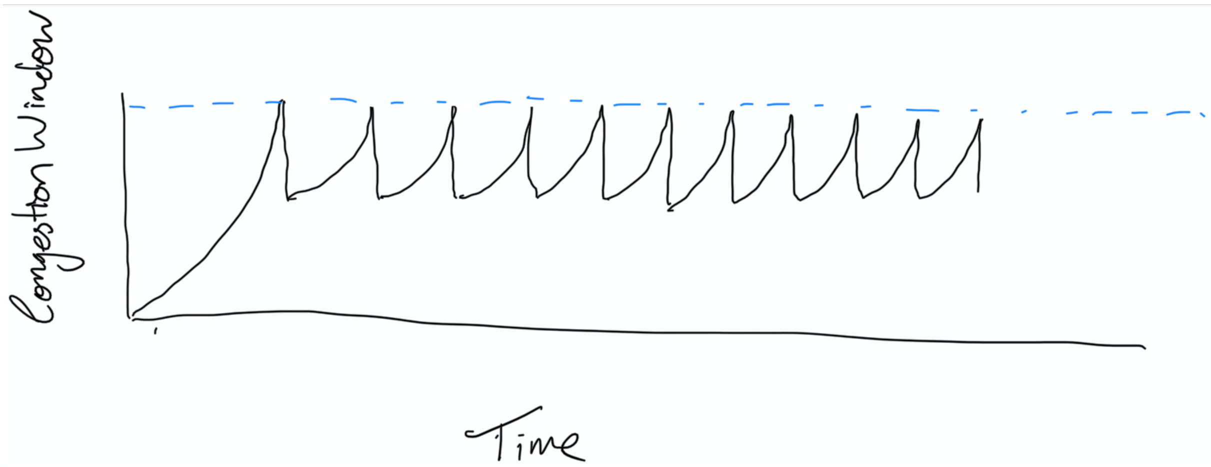
Characterising Networks: Bandwidth

```
...continued...
Server output:
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 79.2.12.119, port 62307
[  5] local 10.0.4.5 port 5201 connected to 79.2.12.119 port 61050
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   348 KBytes  2.85 Mbits/sec  2.190 ms  0/246 (0%)
[  5]   1.00-2.00   sec   421 KBytes  3.45 Mbits/sec  2.115 ms  46/344 (13%)
[  5]   2.00-3.00   sec   424 KBytes  3.48 Mbits/sec  3.787 ms  691/991 (70%)
[  5]   3.00-4.00   sec   424 KBytes  3.48 Mbits/sec  2.857 ms  553/853 (65%)
[  5]   4.00-5.00   sec   419 KBytes  3.43 Mbits/sec  2.242 ms  489/785 (62%)
[  5]   5.00-6.00   sec   420 KBytes  3.44 Mbits/sec  2.077 ms  636/933 (68%)
[  5]   6.00-7.00   sec   414 KBytes  3.39 Mbits/sec  2.487 ms  543/836 (65%)
[  5]   7.00-8.00   sec   419 KBytes  3.43 Mbits/sec  4.396 ms  621/917 (68%)
[  5]   8.00-9.00   sec   419 KBytes  3.43 Mbits/sec  2.197 ms  540/836 (65%)
[  5]   9.00-10.00  sec   423 KBytes  3.46 Mbits/sec  2.679 ms  578/877 (66%)
[  5]  10.00-11.00  sec   421 KBytes  3.45 Mbits/sec  3.789 ms  580/878 (66%)
[  5]  11.00-11.15  sec  63.6 KBytes  3.39 Mbits/sec  3.357 ms  91/136 (67%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-11.15  sec  4.51 MBytes  3.39 Mbits/sec  3.357 ms  5368/8632 (62%)  receive


iperf Done.
```

32 / 56

Characterising Networks: Buffering

- The most complicated question you'll see today:
  - **"How much buffering do I need?"**
- Networks have to buffer packets
- The number of packets buffered depend on what the link is trying to do
- Buffers help accommodate buffer overshoot and improve performance
- Too much buffering conversely increases latency and reduces performance
  - yes performance had two different means there



Characterising Networks: Bandwidth Delay Product

- To calculate how large buffers need to be for a connection we have to work with the bandwidth and the delay
- We talk about the Bandwidth delay product (BDP)
- To fill the network each RTT we send 1 BDP of data
- Sender and receiver has to be able to buffer this much
- BDP for satellite networks are unreasonable

34 / 56

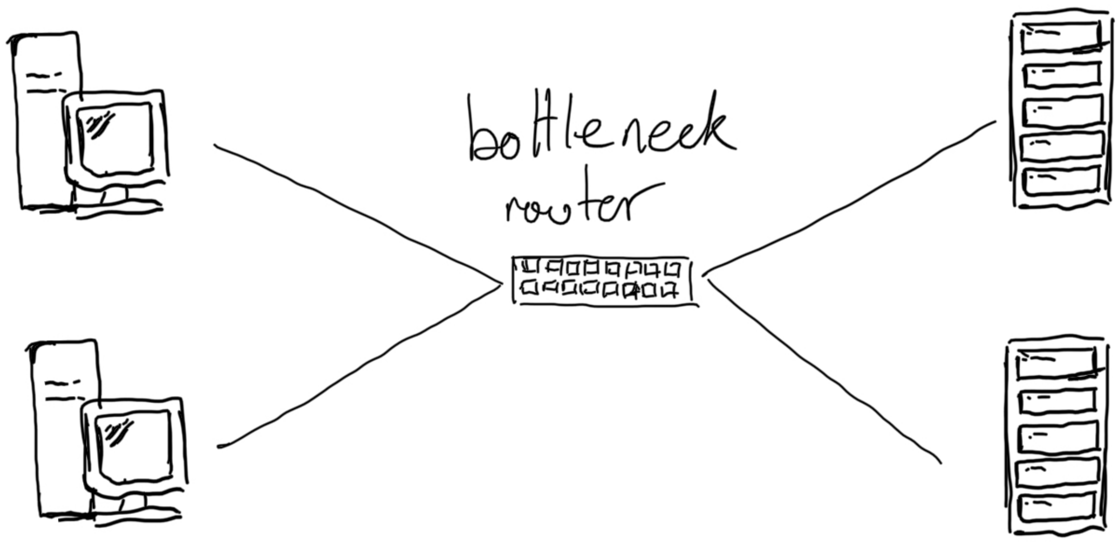
Typical Network Characteristics

```
Network      Delay   Down    Up      BDP (Down)

DSL           20ms   90Mbit  10Mbit    225 KB
4G            50ms   30Mbit  10Mbit    180 KB
Data Center    5ms   1Gbit   1Gbit     256 KB
Data Center    5ms   10Gbit  10Gbit   2560 KB

VSAT (today)650ms   10Mbit  2Mbit     812 KB
VSAT (soon) 650ms   50Mbit  10MBit   4000 KB
VSAT (lab)  650ms   250Mbit 10MBit  20312 KB
```

qalc (from libqalculate) is great for doing mixed unit calculations

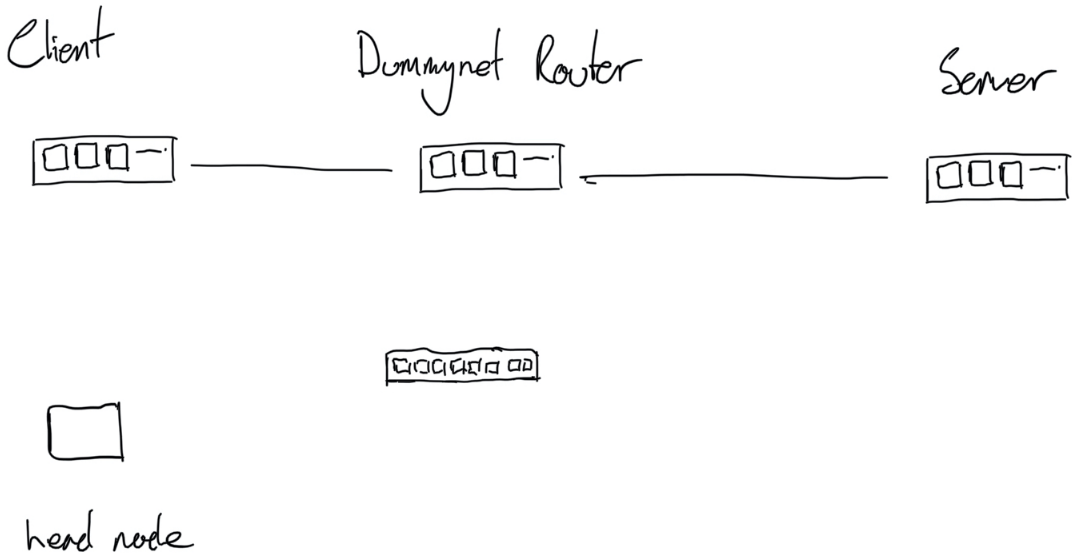
```
$ qalc 10Mbit/s times 650ms to kilobytes
(10 × (megabit / second)) × (650 × millisecond) = 812.5 kilobytes
```

Building Test Networks

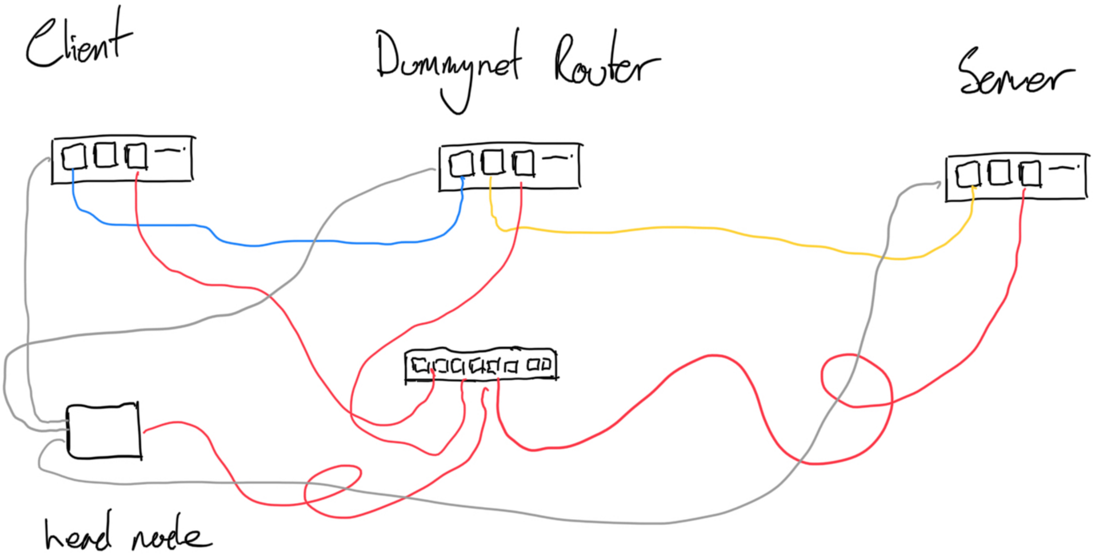


- virtual machines have problems
- hard computers

Building Test Networks

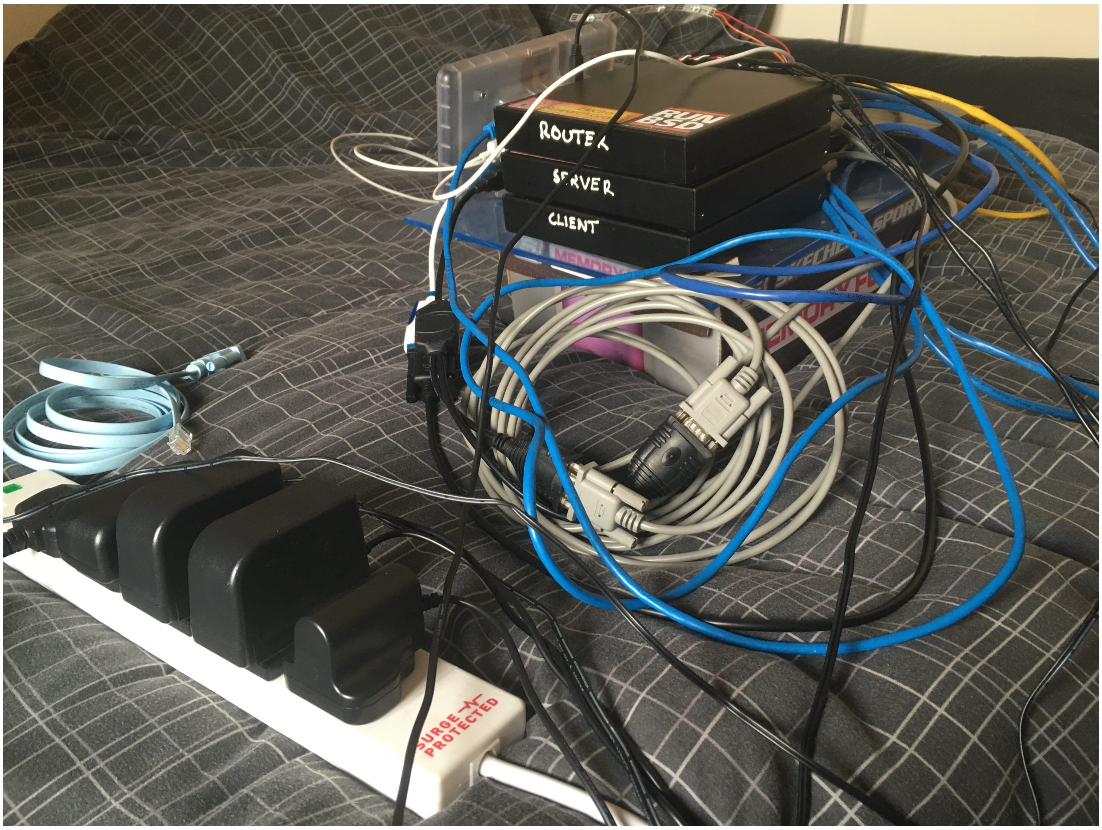


Building Test Networks



38 / 56

Building Test Networks



39 / 56

Config: router

rc.conf

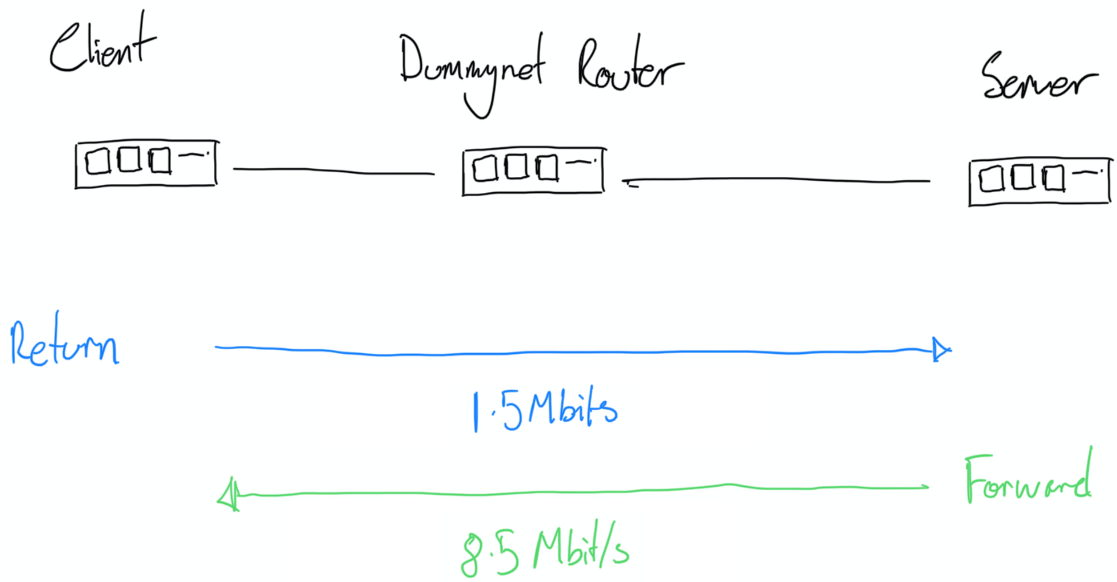
```
hostname="quicsat-router"

ifconfig_igb0="inet 192.168.19.33/24"
defaultrouter="192.168.19.1"
ifconfig_igb0_ipv6="inet6 accept_rtadv"
sshd_enable="YES"

gateway_enable="YES"
ipv6_gateway_enable="YES"

ifconfig_igb1="inet 10.0.1.1/24"
ifconfig_igb2="inet 10.0.2.1/24"

ifconfig_igb1_ipv6="inet6 fd00:1:0:0:1::1/64 no_dad"
ifconfig_igb2_ipv6="inet6 fd00:2:0:0:1::1/64 no_dad"

static_routes="clientv4 serverv4 clientv6 serverv6"
route_clientv4="-inet 10.0.1.0/24 10.0.1.1"
route_serverv4="-inet 10.0.2.0/24 10.0.2.1"
route_clientv6="-inet6 fd00:1:0:0:1::0/56 fd00:1:0:0:1::1"
route_serverv6="-inet6 fd00:2:0:0:1::0/56 fd00:2:0:0:1::1"

#Firewall
firewall_enable="YES"
firewall_script="/etc/ipfw.rules"
```

Config: router

ipfw.rules

```
#!/bin/sh

#Flush out thye list
ipfw -q -f flush

#set rules command refx
cmd="ipfw -q add"

# interfaces
controlif=igb0
clientif=igb1
serverif=igb2

# No restrictions on Loopback Interface
$cmd 00010 allow all from any to any via lo0
$cmd 00101 check-state

$cmd 000200 allow all from any to any via $controlif

$cmd 1000 pipe 1 ip from 10.0.1.0/24 to any via $clientif
ipfw -q pipe 1 config delay 300ms bw 1500Kbit/s
$cmd 1000 pipe 2 ip from 10.0.2.0/24 to any via $serverif
ipfw -q pipe 2 config delay 300ms bw 8500Kbit/s queue 640KB

#allow everything
$cmd 50000 allow all from any to any via any
```

Building Test Networks

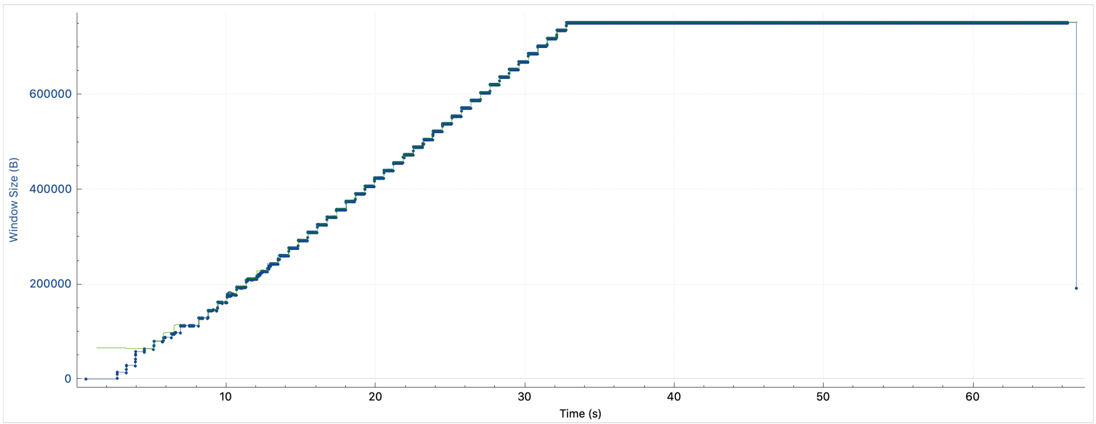


42 / 56

Config: running experiments

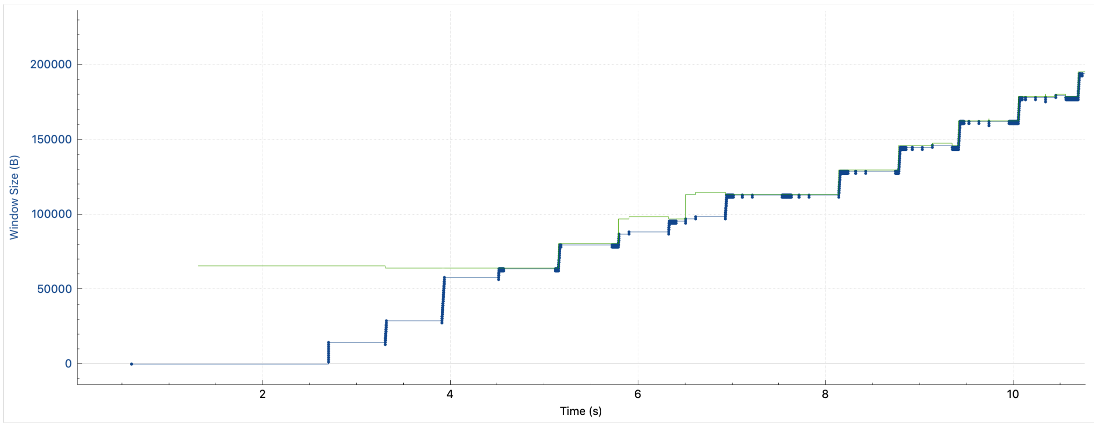
```
#!/bin/sh
# conf bottleneck

case $1 in
    ref)
        # $ qalc 8.5Mbit/s times 600ms to kilobytes
        # (8.5 * (megabit / second)) * (600 * millisecond) = 637.5 kilobytes
        echo reference scenario
        ipfw pipe 1 config delay 300ms bw 1500Kbit/s
        ipfw pipe 2 config delay 300ms bw 8500Kbit/s queue 640KB
        ;;
    small)
        echo small scenario
        ipfw pipe 1 config delay 325ms bw 2000Kbit/s queue 160KB
        ipfw pipe 2 config delay 325ms bw 10000Kbit/s queue 810KB
        ;;
    medium)
        echo medium scenario
        ipfw pipe 1 config delay 325ms bw 10000Kbit/s queue 810KB
        ipfw pipe 2 config delay 325ms bw 50000Kbit/s queue 4062KB
        ;;
    *)
        echo "usage ./setupnetwork [ref|small|medium]"
        exit
        ;;
esac
```

Discoveries in Testbed



44 / 56

TCP Flow control



45 / 56

Config: buffer tuning

Fixing the TCP RWND:

```
$ iperf3 -c localhost -w 10M
Connecting to host iperf3.eurobsdcon.org, port 5201
iperf3: error - unable to set socket buffer size: No buffer space available
```

10MB queue sizes for dummynet

`client` and `server`:

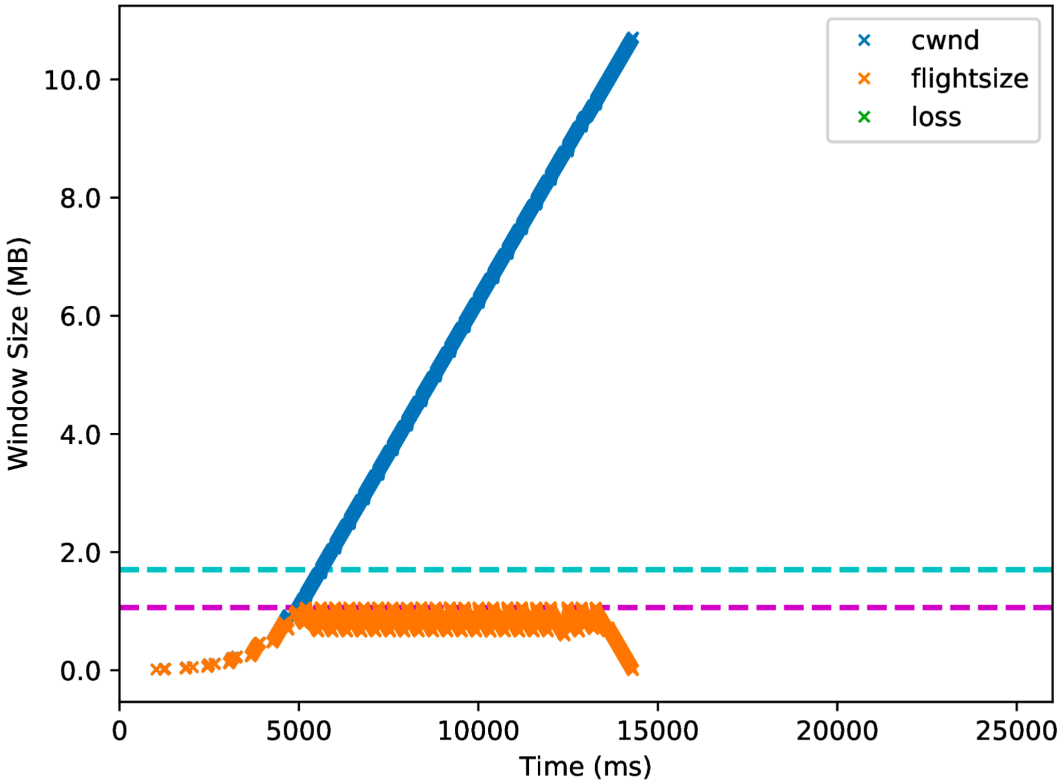
```
kern.ipc.maxsockbuf=209715200
```

`router`:

```
net.inet.ip.dummynet.pipe_byte_limit=10485760
```

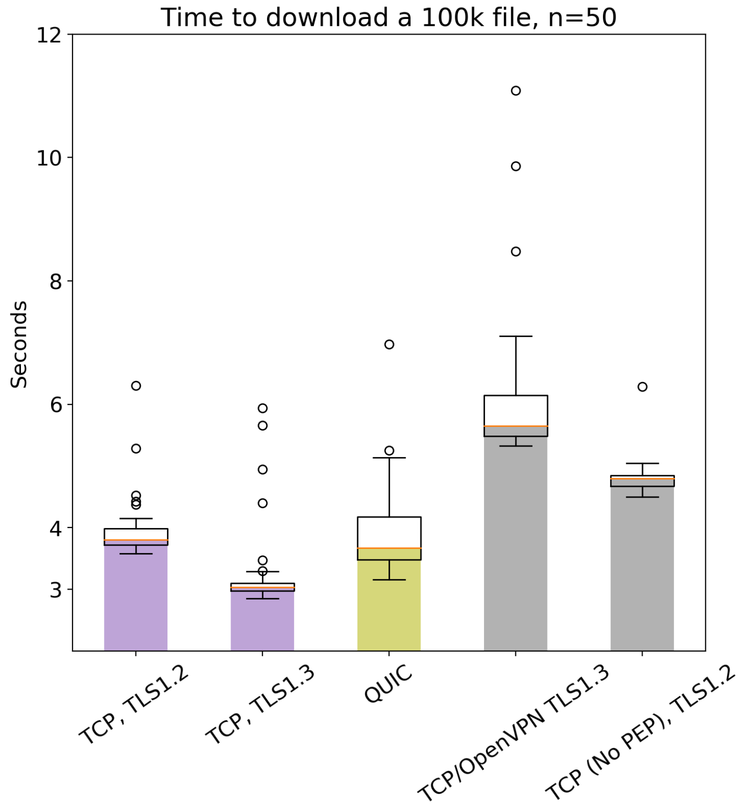
46 / 56

It is difficult to get QUIC up to speed!



47 / 56

QUIC start up vs TCP vs TCP without a PEP



48 / 56

Limitations of Dummynet

- Time in virtual machines is too weird for accurate delay emulation, without VNET support this makes testsutie integration really hard
- Dummynet is feeling its years
  - Dummynet can be configured to shape up to about 4Gigabit
- TLEM is a proposed design for Terrabit network emulation

49 / 56

Future of Dummynet

VNET

- support landed by kp@ in 2021
- this enables the use of dummynet in FreeBSD and other test suites!

pf

- there has been pf support for dummynet in other operating systems (including Mac OS) for a long time
- support is now being ported from pfsense to FreeBSD (funded by Netgate)
  - https://reviews.freebsd.org/D31904
  - this will probably be mfc'd

There is a WIP high performance rewrite, watch the mailing lists for info

Questions?

thanks for listening

Colophon

- These slides were written in markdown and composed in vim and
- presented using remarkjs
- put together with some hacky shell scripts
- They were presented in the firefox web browser.

extra additional slides

Config: client

rc.conf

```
hostname="quicsat-client"

ifconfig_igb0="inet 137.50.19.31/24"
defaultrouter="137.50.19.1"

#ifconfig_igb1_ipv6="inet6 fd00:1:0:0:2::1/64 no_dad"
#ipv6_defaultrouter="fd00:1:0:0:1::1"

ifconfig_igb1="inet 10.0.1.2/24"
static_routes="router server"
route_router="-net 10.0.1.0/24 10.0.1.1"
route_server="-net 10.0.2.0/24 10.0.1.1"

sshd_enable="YES"
```

Config: server

rc.conf

```
hostname="quicsat-server"
ifconfig_igb0_ipv6="inet6 accept_rtadv"
sshd_enable="YES"
ntpdate_enable="YES"
ntpd_enable="YES"

ifconfig_igb0="inet 192.168.50.19.32/24"
defaultrouter="192.168.19.1"

ifconfig_igb1_ipv6="inet6 fd00:2:0:0:2::1/64 no_dad"
ipv6_defaultrouter="fd00:2:0:0:1::1"

ifconfig_igb1="inet 10.0.2.2/24"
static_routes="router client"
route_router="-net 10.0.2.0/24 10.0.2.1"
route_client="-net 10.0.1.0/24 10.0.2.1"
```

Config: running experiments

```
#!/bin/sh

DELAY=5

loss()
{
    echo 1 percent loss
    ipfw pipe 2 config delay 300ms bw 8500Kbit/s queue 640KB plr 0.01
}

noloss()
{
    echo no loss
    ipfw pipe 2 config delay 300ms bw 8500Kbit/s queue 640KB plr 0
}

quit ()
{
    echo
    echo clearing loss
    ipfw pipe 2 config delay 300ms bw 8500Kbit/s queue 640KB plr 0
    exit
}

if [ "$1" == "loss" ]
then
    loss
    exit
fi
```