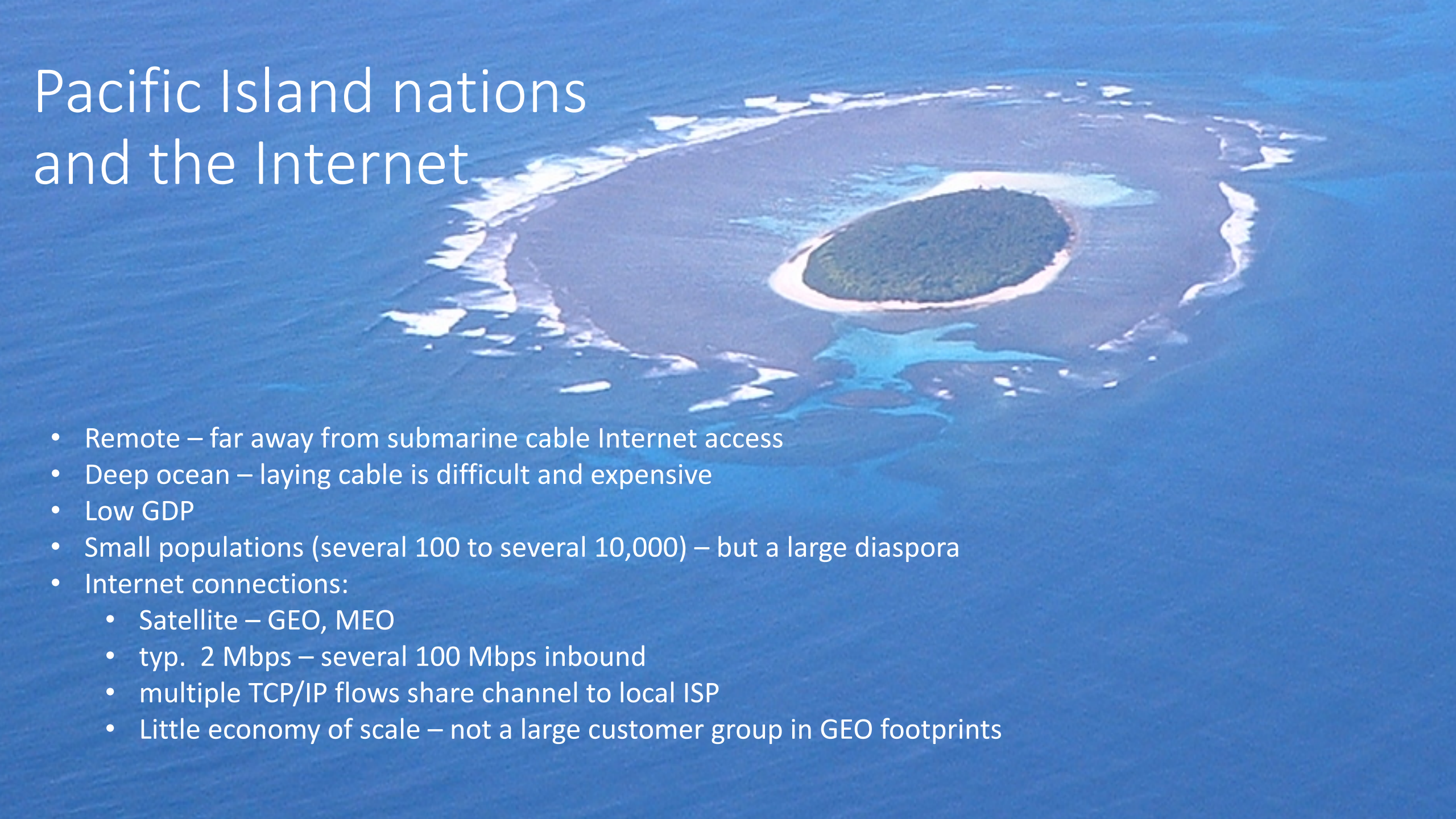
Simulating Satellite Links into Pacific Islands

Ulrich Speidel, Lei Qian, ‘Etuate Cocker (University of Auckland)
Muriel Médard (Massachusetts Institute of Technology)
Péter Vingelmann, Janus Heide (Steinwurf ApS, Denmark)

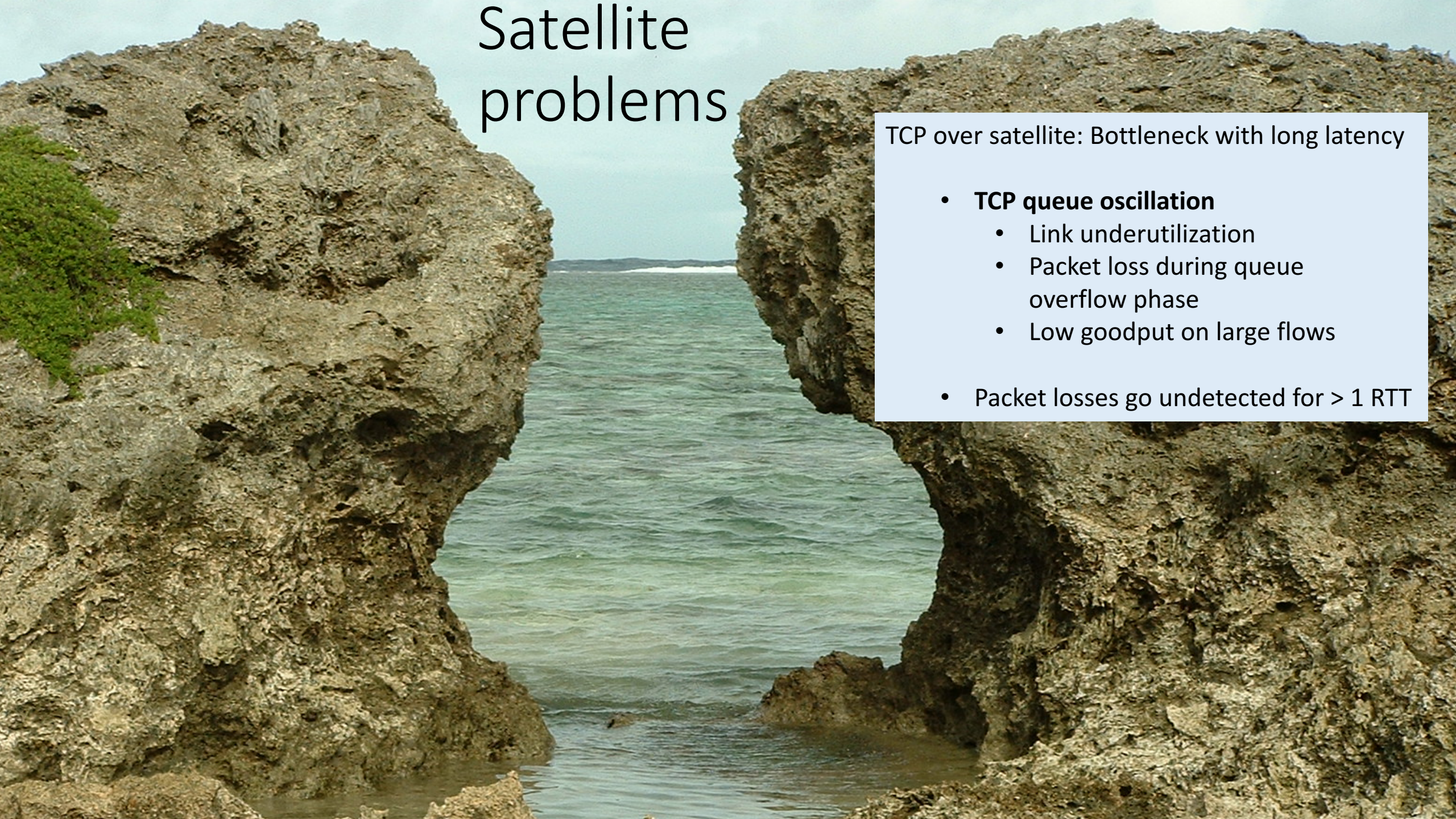
ulrich@cs.auckland.ac.nz

Pacific Island nations and the Internet



- Remote – far away from submarine cable Internet access
- Deep ocean – laying cable is difficult and expensive
- Low GDP
- Small populations (several 100 to several 10,000) – but a large diaspora
- Internet connections:
  - Satellite – GEO, MEO
  - typ. 2 Mbps – several 100 Mbps inbound
  - multiple TCP/IP flows share channel to local ISP
  - Little economy of scale – not a large customer group in GEO footprints

Satellite problems

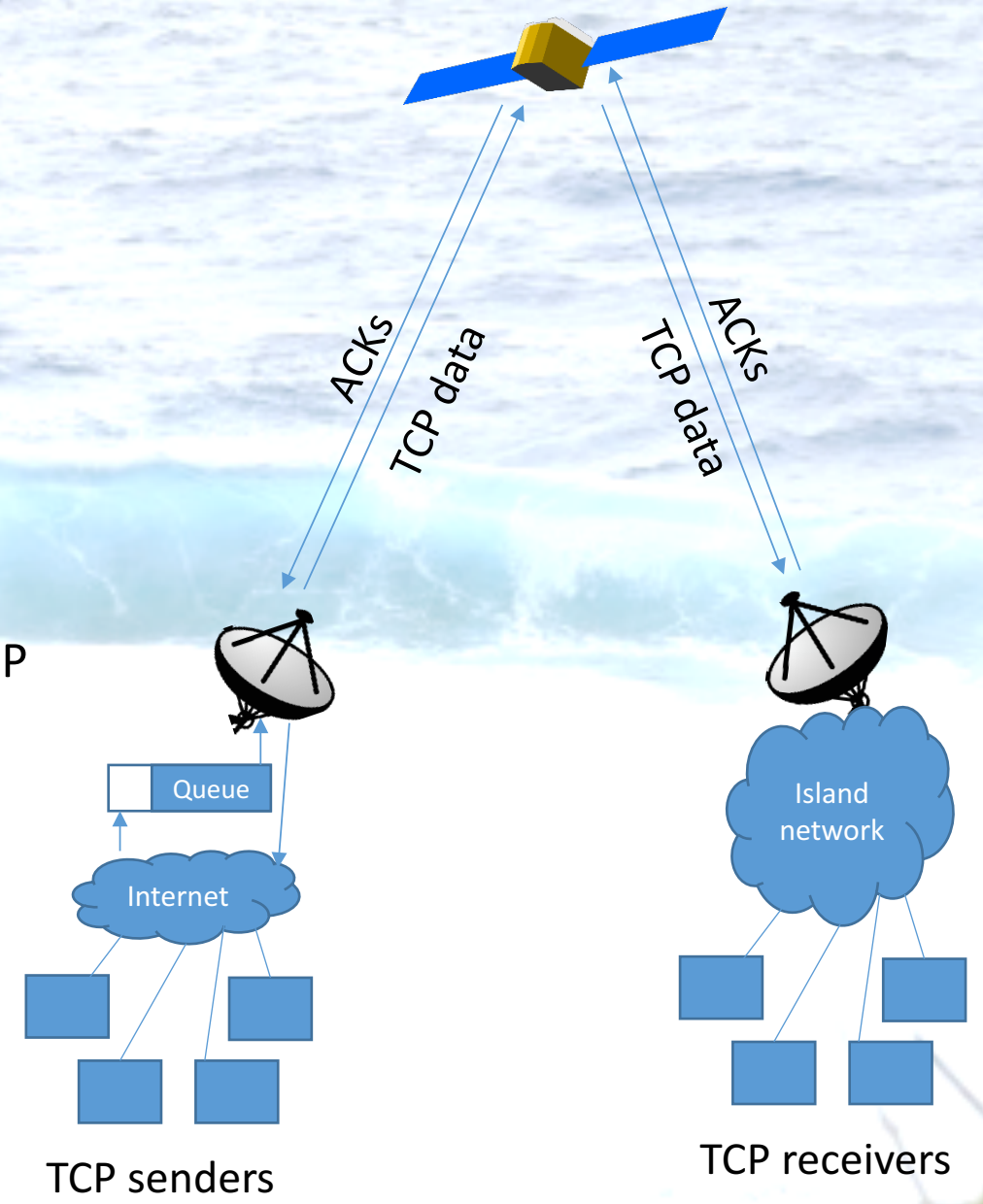


TCP over satellite: Bottleneck with long latency

- **TCP queue oscillation**
  - Link underutilization
  - Packet loss during queue overflow phase
  - Low goodput on large flows
- Packet losses go undetected for > 1 RTT

TCP queue oscillation

- *Multiple* TCP senders remotely send traffic to the sat gate
- Sat link is a *bottleneck*. *Queue* at sat gate acts like a *funnel*.
- TCP sender cannot see queue state directly
- Feedback on queue state goes via the satellite to remote TCP receivers, and from there back to the senders
- Long RTTs: >500 ms on GEO, >125 ms on MEO
- Queue can **oscillate** between empty and overflow
- Complicating factors: TCP slow start, exponential back-off



The four phases of queue oscillation

1. Sat gate queue not full. TCP senders receive ACKs, increase congestion window. Queue builds up.

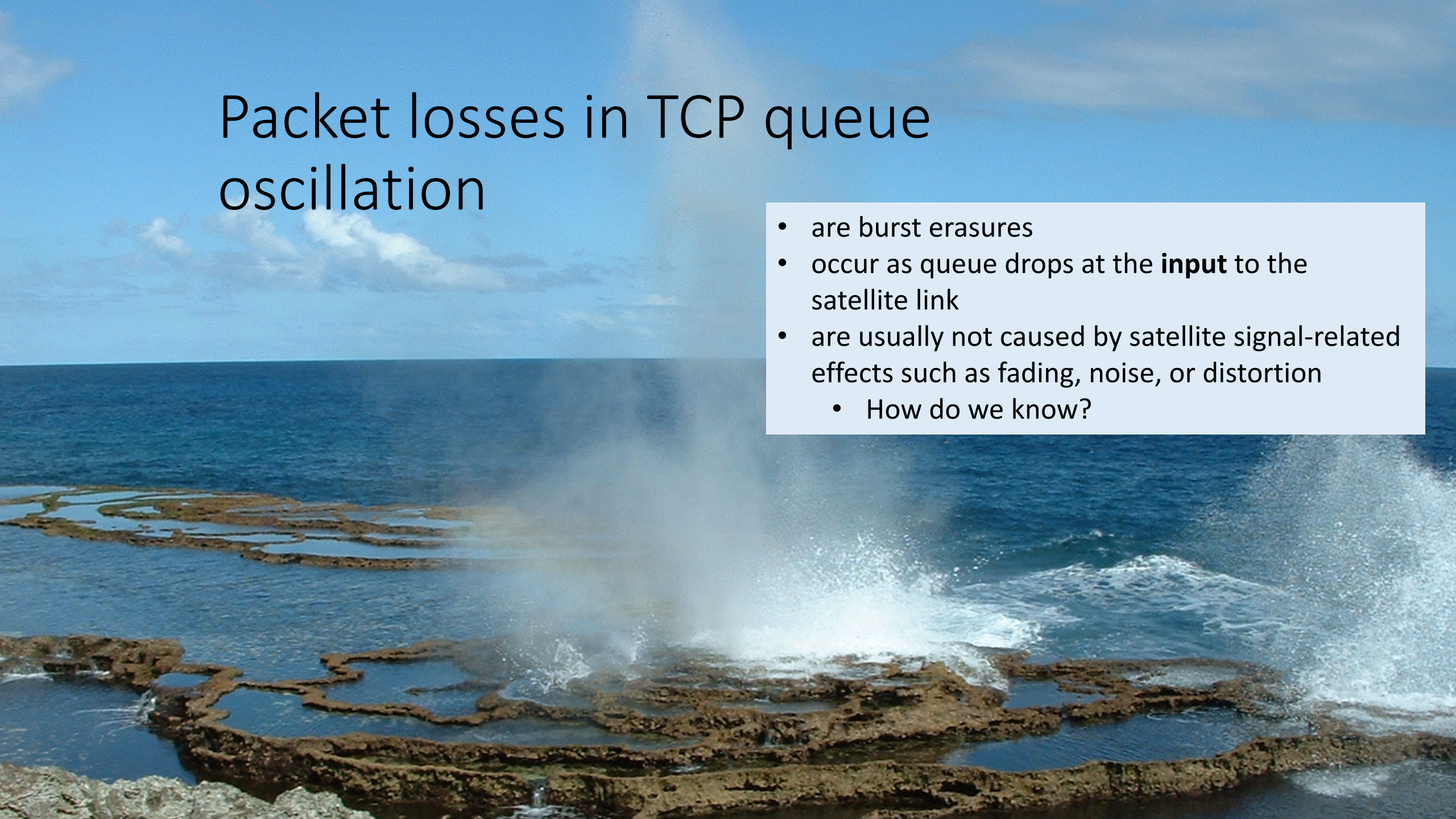
2. Sat gate queue full. New packets arriving are dropped. Senders still receive ACKs and send more data in the direction of the queue. Queue continues to overflow: *burst losses*

3. ACKs from dropped packets become overdue. Senders throttle back. Packet arrival at queue slows to a trickle. Queue drains.

4. Queue clears completely. Link sits idle for part of the time, *link not fully utilised*

Note: Queue oscillation explains the packet loss phenomena on all sat links we studied – we don't need noise or interference

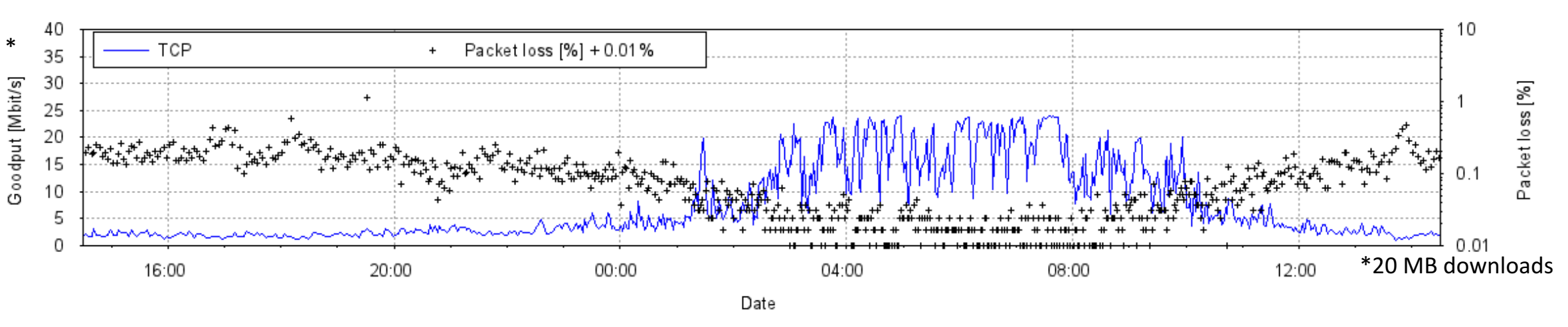
Packet losses in TCP queue oscillation



- are burst erasures
- occur as queue drops at the **input** to the satellite link
- are usually not caused by satellite signal-related effects such as fading, noise, or distortion
  - How do we know?

Packet loss and goodput evidence

Packet loss happens during busy hours in many islands.
This is a sample from Rarotonga, Cook Islands

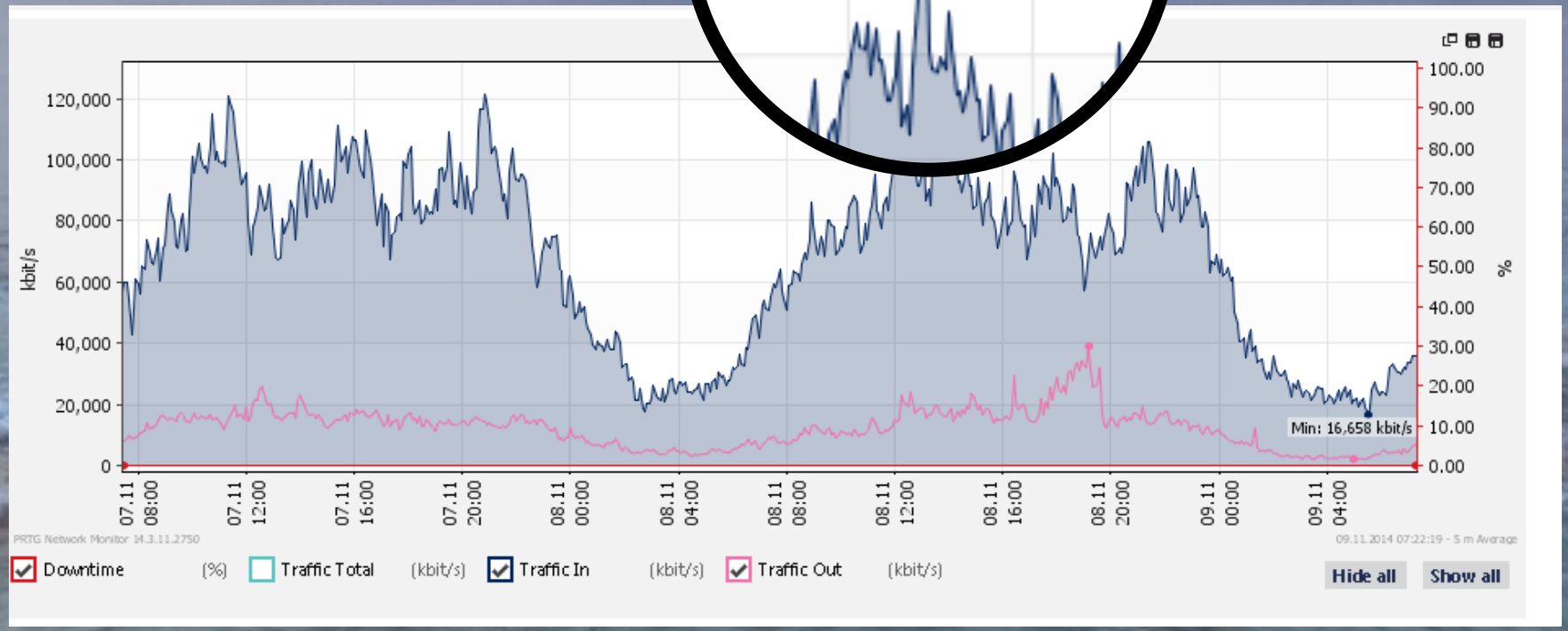


Why? Volume charging!

Link utilisation evidence

Link utilisation of a real ISP on a 160 Mbps MEO connection

- plenty of spare capacity, but…
- users complain of slow downloads

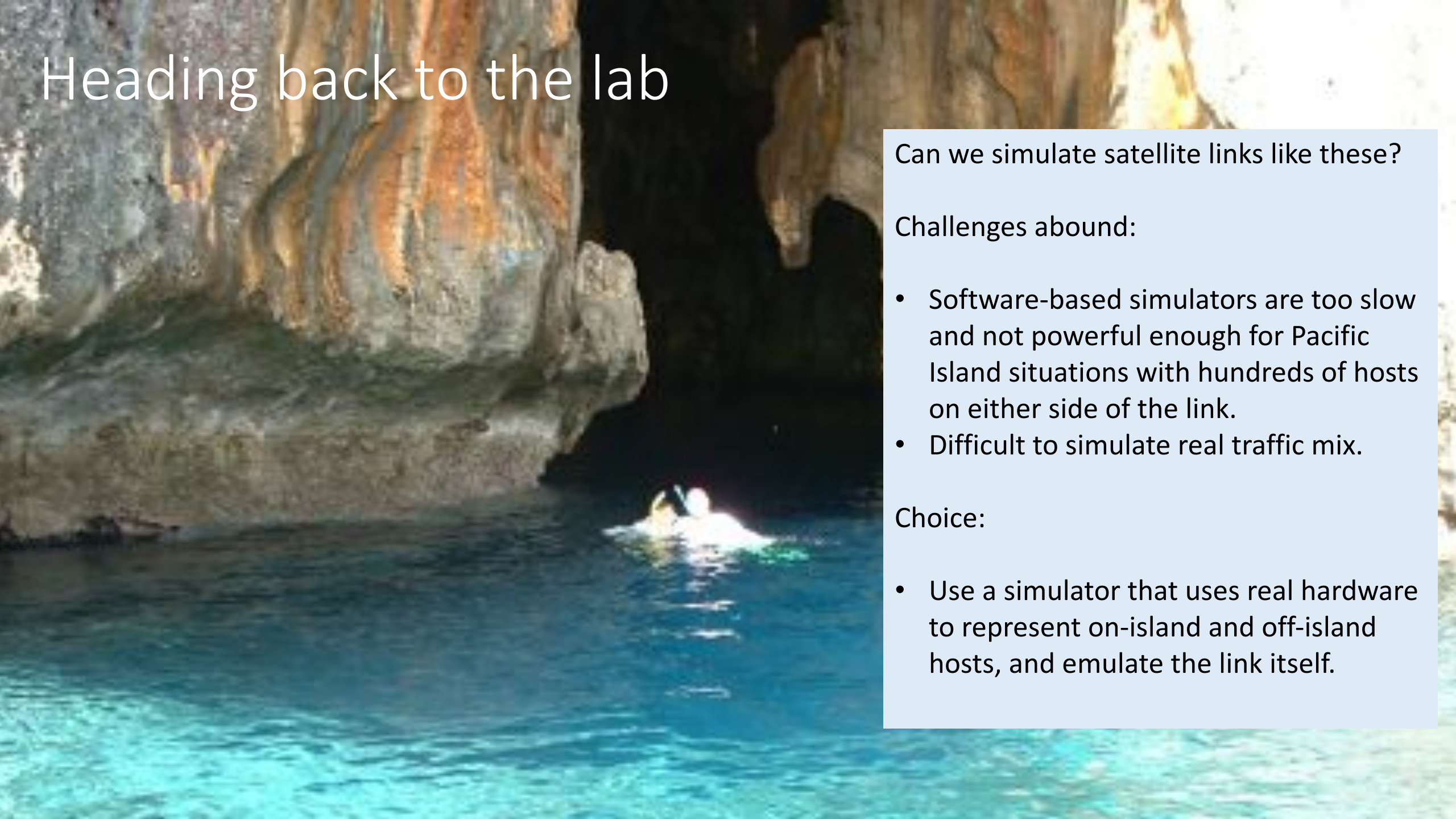


Trying things out on site

The cost of experimenting on site is high:

- Travel
- Dealing with Customs
- Load on production links
- Data retrieval after site visit
- Equipment generally needs to be left on site
- Can’t disconnect an island to try something out!

Heading back to the lab



Can we simulate satellite links like these?

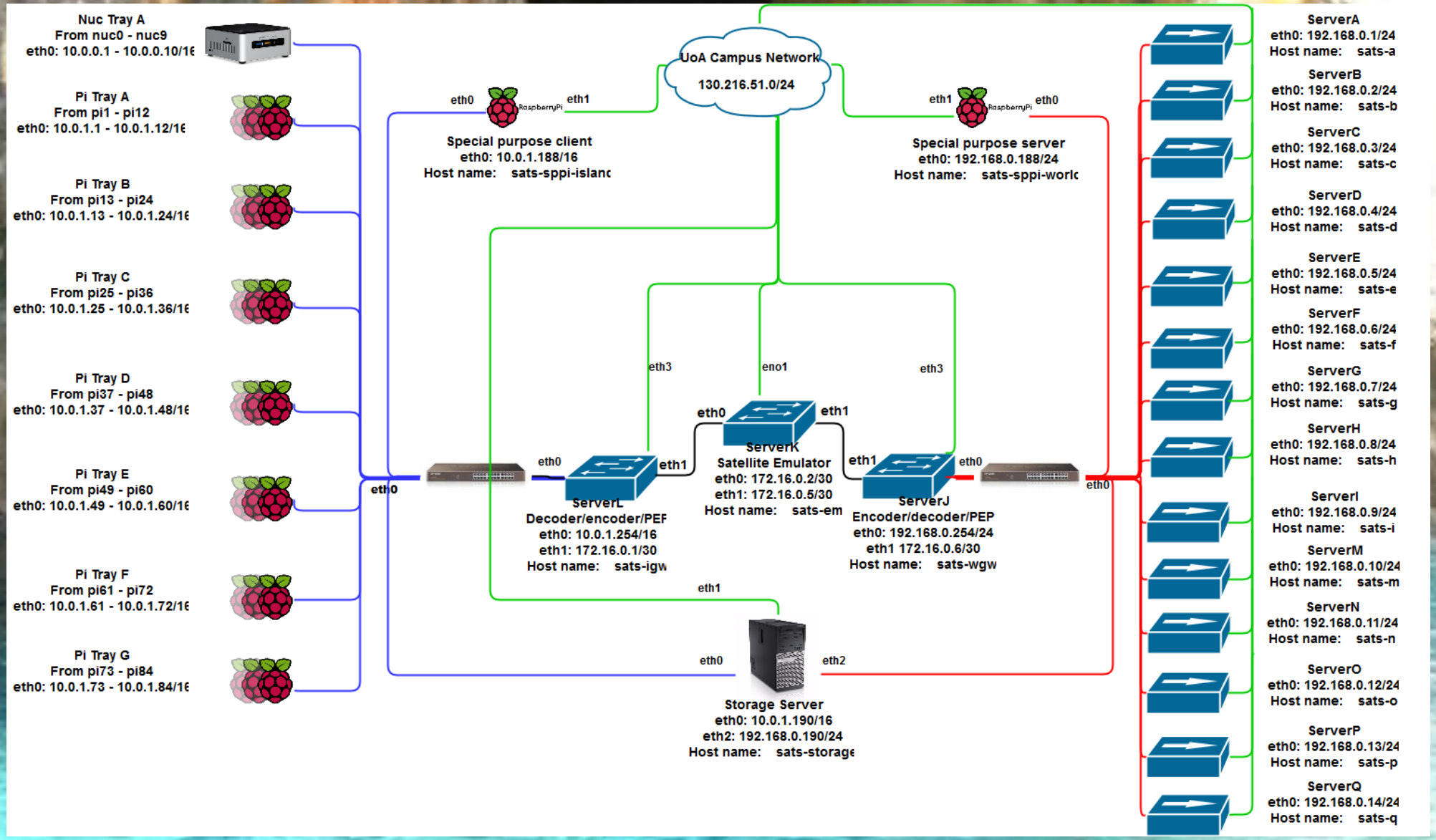
Challenges abound:

- Software-based simulators are too slow and not powerful enough for Pacific Island situations with hundreds of hosts on either side of the link.
- Difficult to simulate real traffic mix.

Choice:

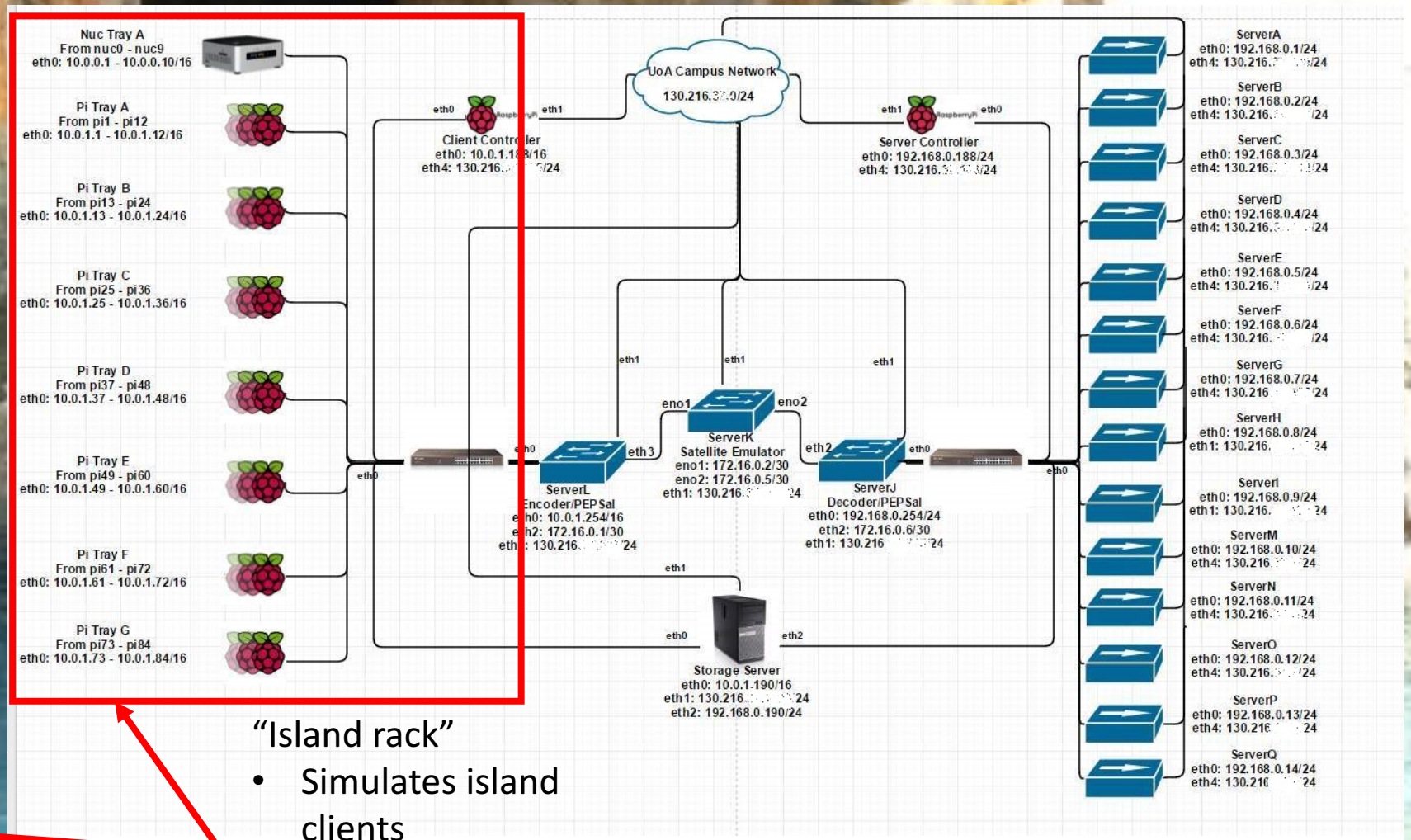
- Use a simulator that uses real hardware to represent on-island and off-island hosts, and emulate the link itself.

The Auckland Satellite Simulator Facility



Simulator hardware

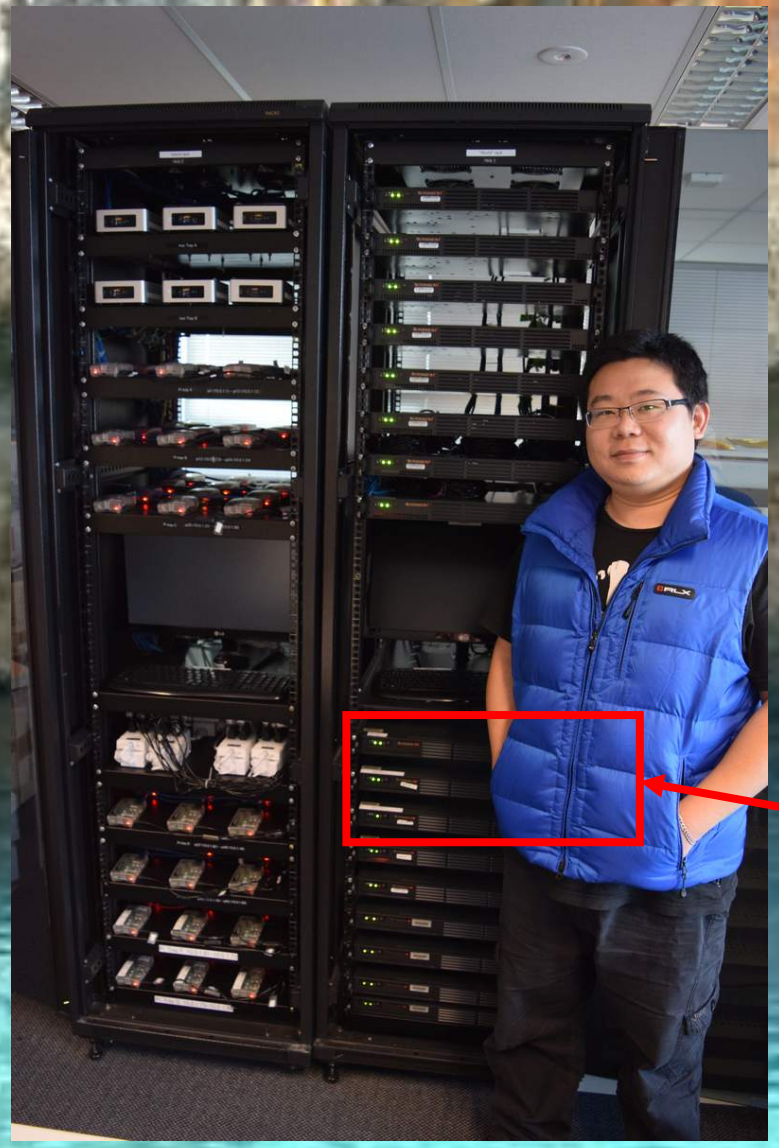
Our satellite link simulator hardware

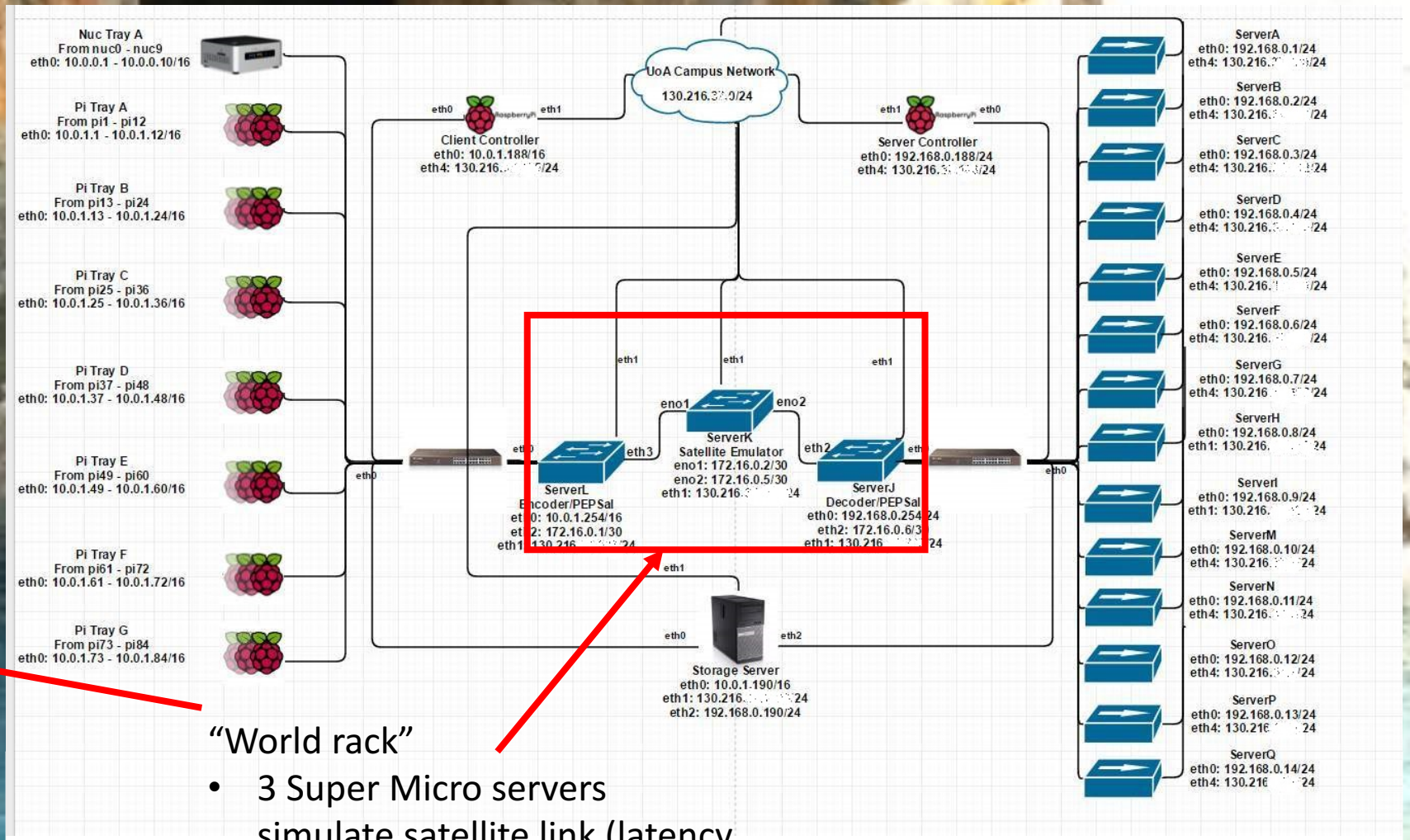


"Island rack"

- Simulates island clients
- 10 Intel NUC (2 trays, up to 6 per tray)
- 84 Raspberry Pi (7 trays, 12 per tray, plus 1 power supply tray

Our satellite link simulator hardware

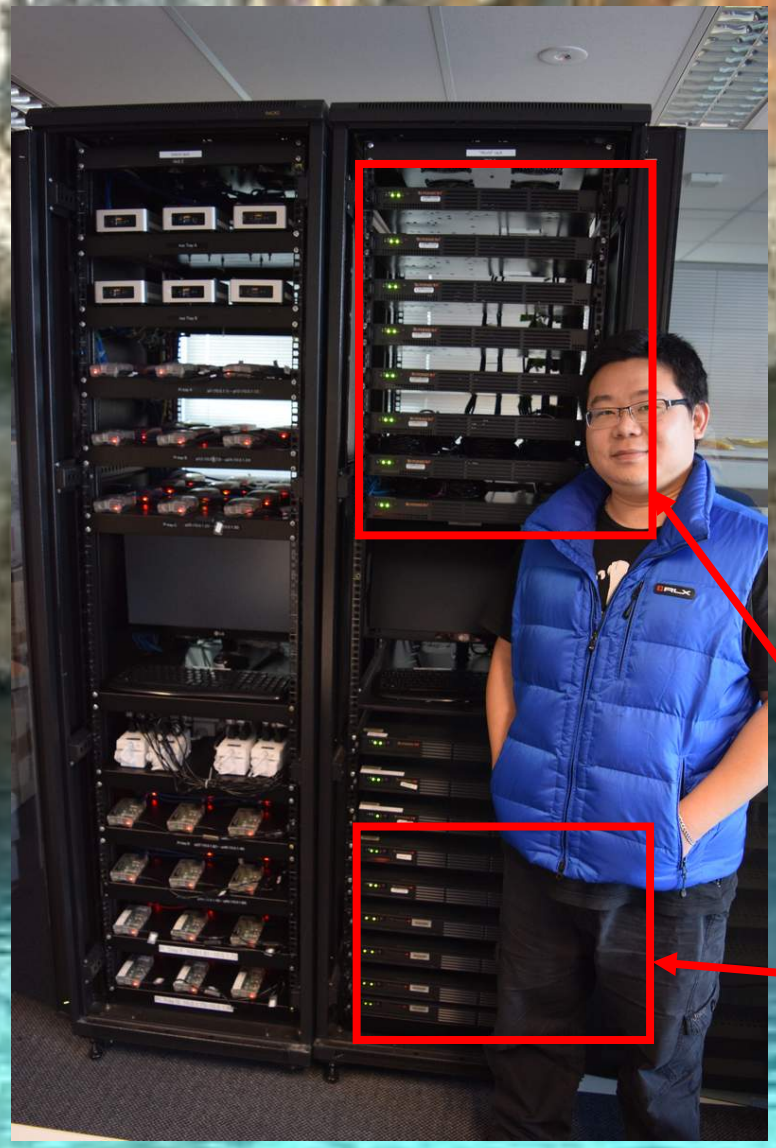


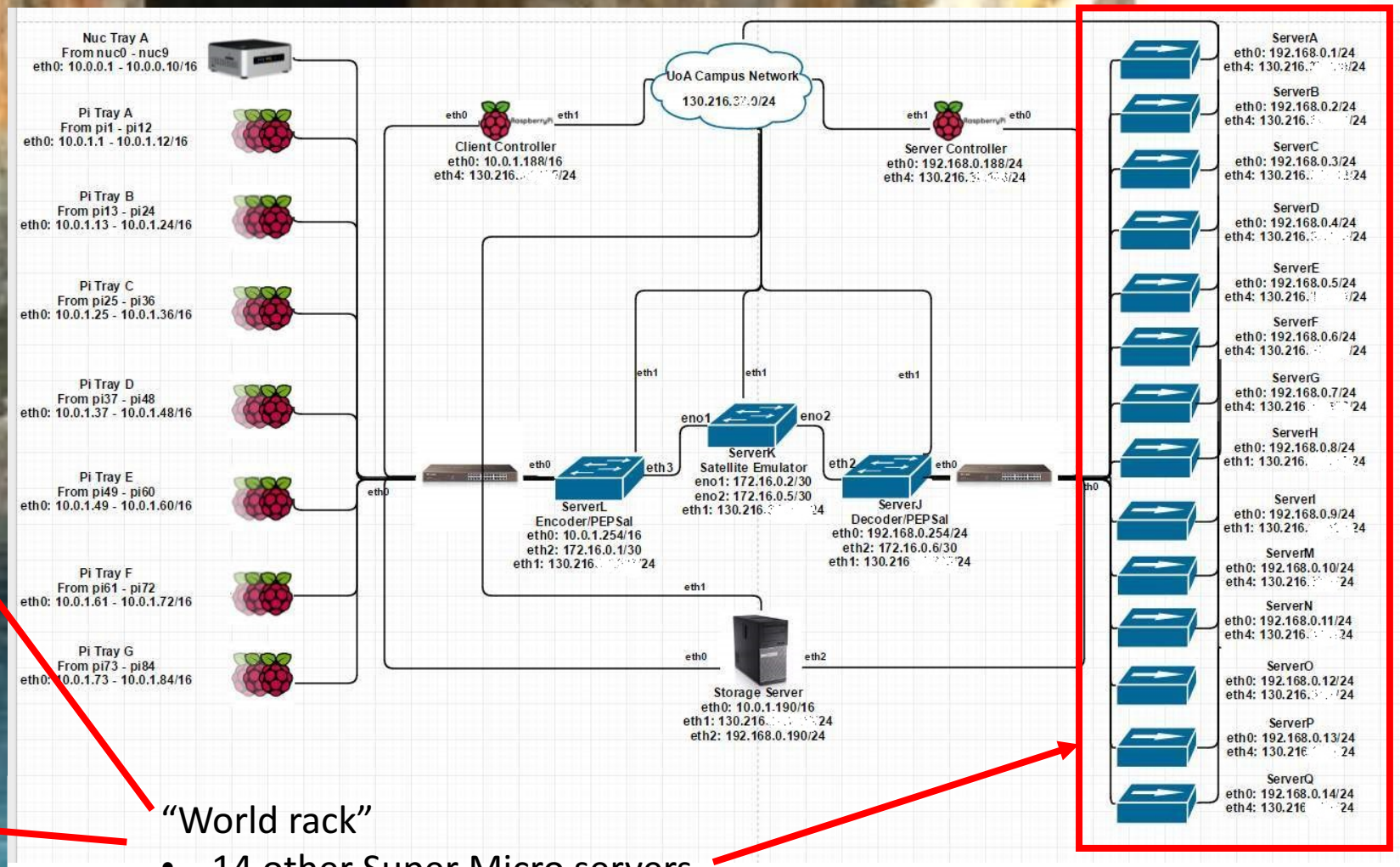


"World rack"

- 3 Super Micro servers simulate satellite link (latency, bandwidth) and PEP or TCP/NC tunnel endpoint

Our satellite link simulator hardware





"World rack"

- 14 other Super Micro servers simulate "servers of the world"

Simulating demand

- Purpose-built client and server software
  - Client software runs on island NUCs and Pis
  - Server software runs on world Super Micros

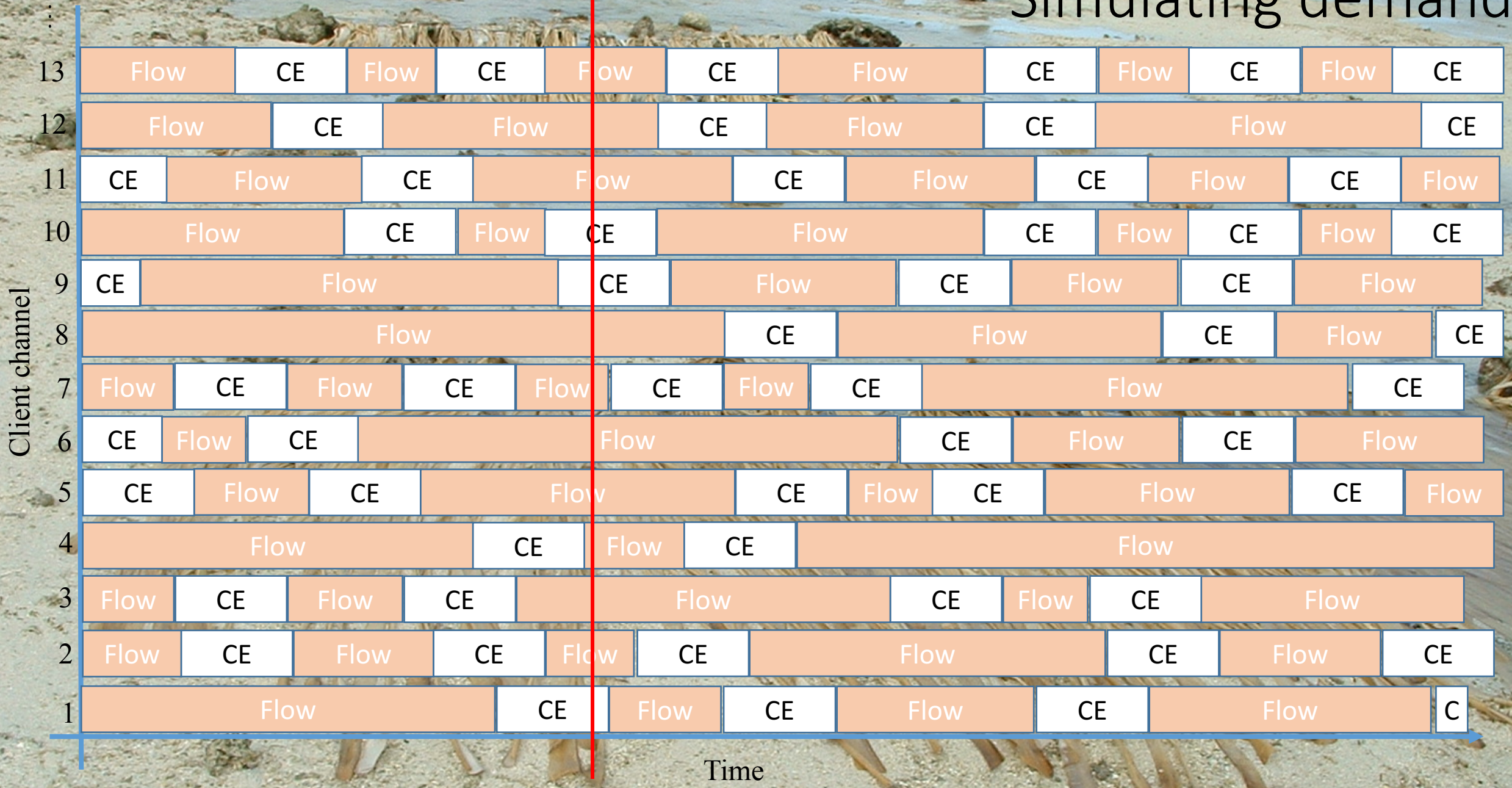
- Each client is configured to run a certain number of *channels*

- Each channel repeats the following procedure for the duration of an experiment
  1. Connect to a randomly chosen world server via TCP
  2. Receive data and await disconnect

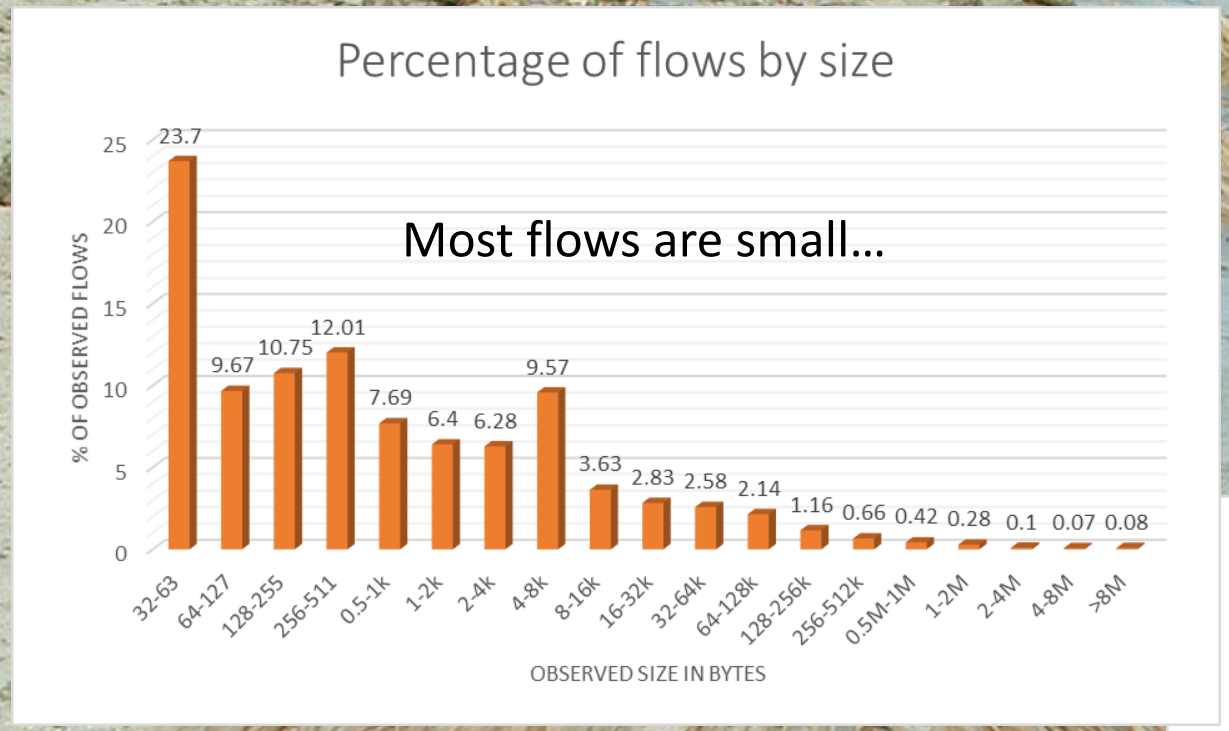
- Each world server:
  1. Accepts incoming connections from client channels
  2. Randomly selects a flow size from a pre-configured distribution
  3. Sends that number of bytes
  4. Disconnects the clients

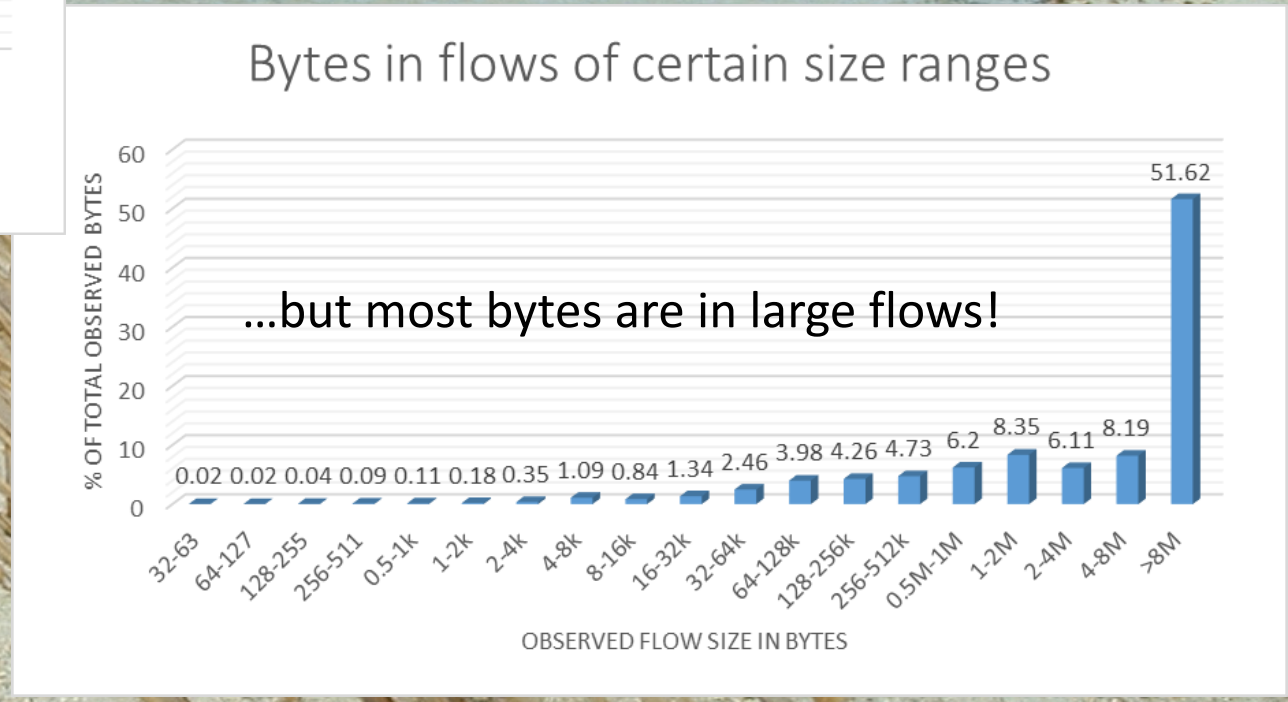
- Total number of channels = demand
  - This lets us control demand

Simulating demand



Simulating demand: Flow sizes





Source: Own collection at Bluesky Cook Islands Rarotonga

Simulating demand: Challenges

Distribution is quite heavy-tailed:

- Experiments with low channel numbers / short durations and random flow size selection from the distribution don't produce the same flow size mean as the distribution
- Flow mix varies during the experiment

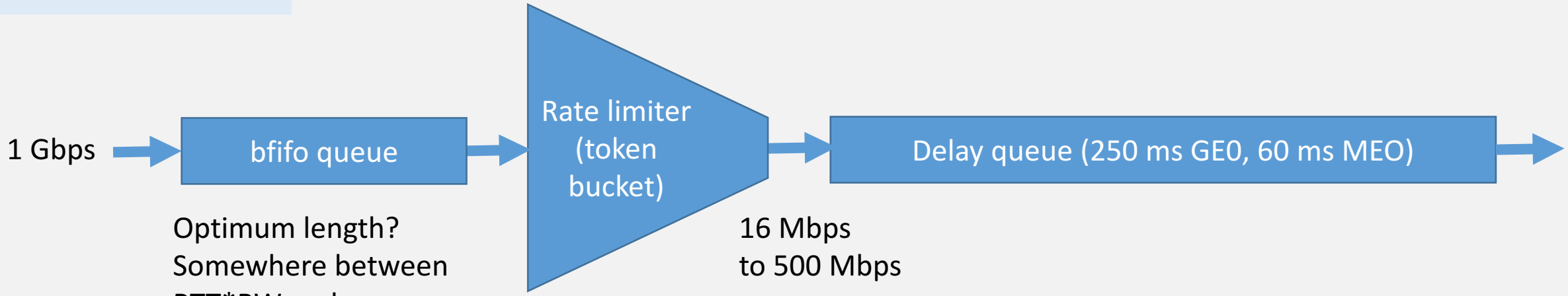
Number of channels is related to, but not equal to the number of parallel flows:

- Connection establishment time means "no flow" on channel
- MEO CE times are much shorter than on GEO, but flows also complete faster (up to a point) because lower RTT lets congestion window open faster.
  - Can't conclude from GEO to MEO and vice versa!

All this happens in real life, too, but experimenters like control – we don't get this here!

Link simulation (simplified)

In each direction:



Experiment procedure

1. Set up “terrestrial” delays on world servers
2. Set up link (delay, rate)
3. Where applicable, set up coding and/or PEP
4. Test all interfaces (pingable, correct RTT?)
5. **Start an iperf3 server on one of the NUCs**
6. Start packet capture on island gateway and then world gateway
7. Reset PMTU cache on world servers (coding changes MTU)
8. Start server on world servers
9. **Ping from special purpose machine on world side to client side** (synchronise capture logs)
10. Compute client channels per client machine and start clients
11. Start a large iperf3 TCP transfer from world side to the NUC (measure goodput of large transfer)
12. Start a rapid 120 s series of pings (every 100 ms)
13. Wait for nominal experiment duration
14. Ping again from special purpose machine (synch logs)
15. Stop clients, servers, iperf3
16. Convert capture files to text and transfer to storage / analysis machine
17. Transfer ping & iperf3 logs to storage / analysis machine
18. Analyse logs

Experiment challenges

- IP fragmentation, TCP segmentation offload along path, aggregation during capture
  - Packets don't appear to match up
  - Need to look at how many bytes were lost
  - Packet loss as a figure in TCP is quite meaningless!
- Need for quality assurance: What happens if individual machines misbehave?
- Timing is critical – correct orchestration is key
- Sat link simulation: tc is nice, but has pitfalls!
- Number of feasible channels per physical client is quite limited (approx. 30-40 with a few tweaks)
- Configuration state can be difficult to ascertain, especially if an experiment needs to be stopped
- Experiments take about 20 minutes each
  - Longer experiments: Better statistical convergence, but also larger log files
  - Higher bandwidth link experiments converge faster, but gains are lost to processing of larger log files.
  - Even so, significant variance in results due to heavy-tailed distribution.

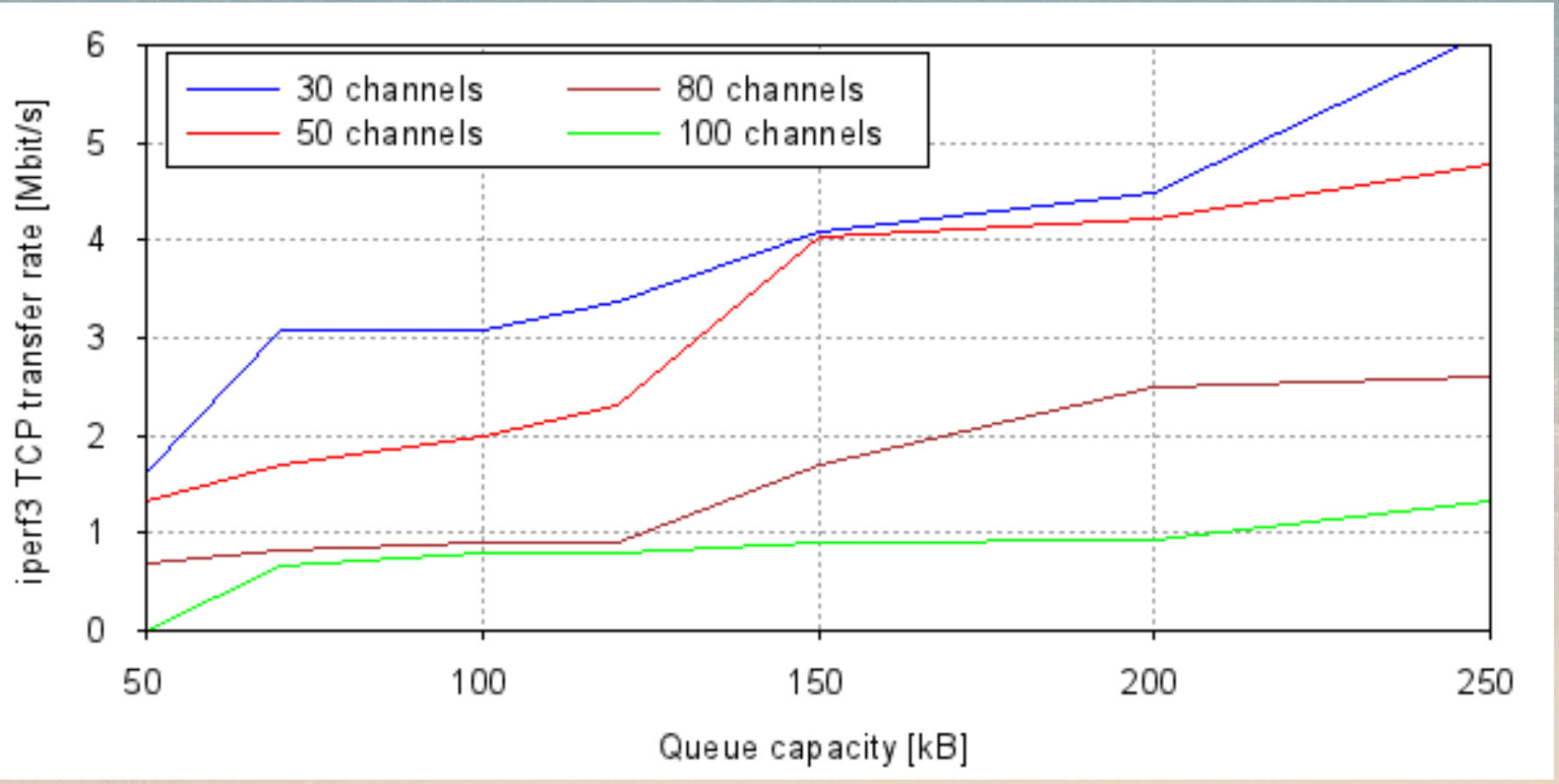
Baselines

- Three major parameters in each experiment: link type (capacity in Mbps and GEO/MEO latency), queue capacity (kB) and demand level (client channels configured)

- Observables
  - Total throughput/goodput (from capture files)
  - RTT/queue sojourn time (from ping logs)
  - Iperf3 TCP transfer rate
  - Average number of concurrent flows/total flows completed
  - Losses (goodput / nominal packet loss)
  - QA observables: number of island/world hosts seen, goodput/loss@host, late start/early finish – check these automatically

Baselines – queue capacity

- More queue capacity means less oscillation but also more latency (bufferbloat)

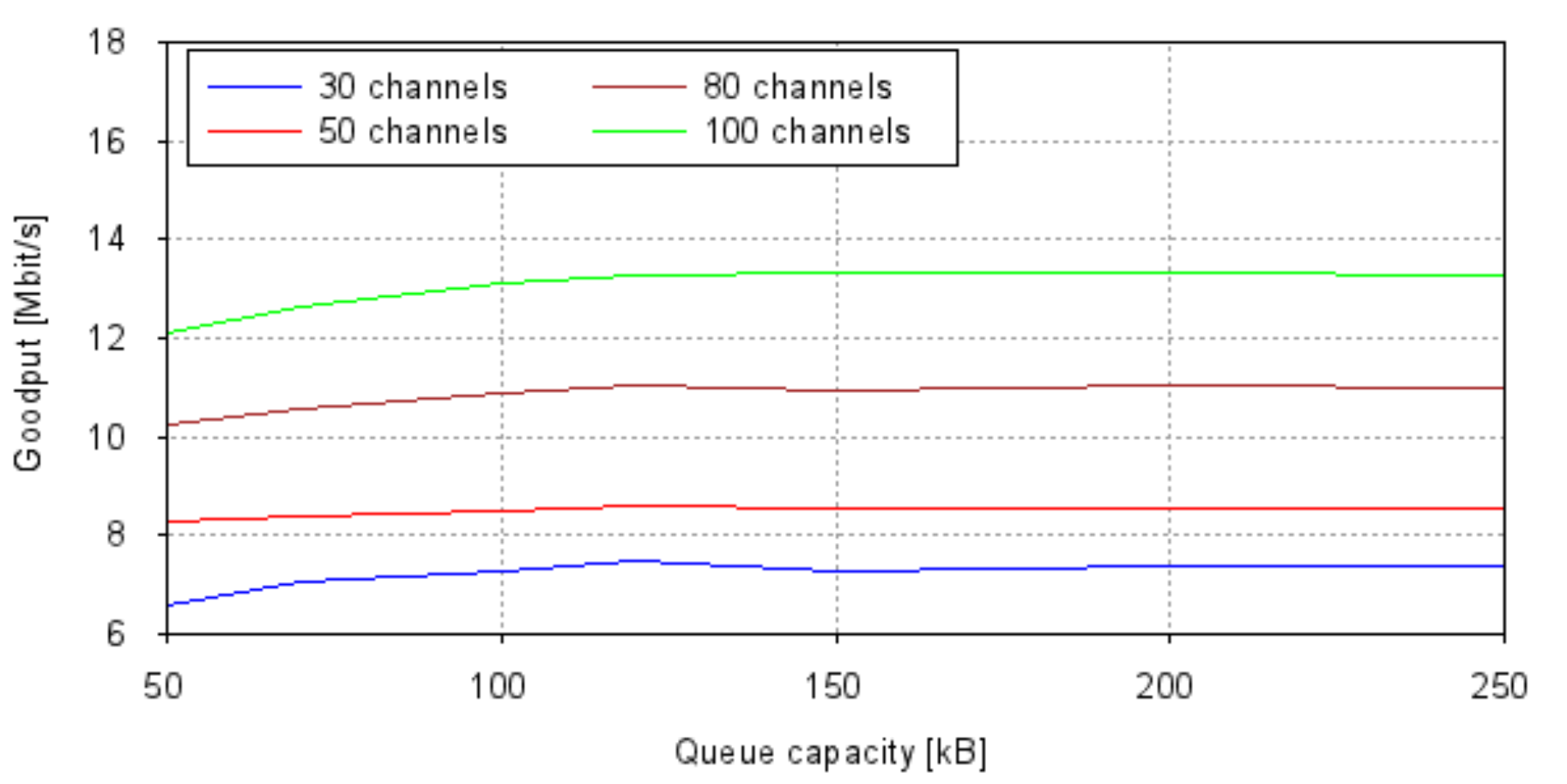


Queue capacity [kB]	Max. queu time [ms]
50	
70	
100	
120	
150	
200	
250	

Example: 16 Mbps sim
GEO satellite link

Baselines – goodput

- More queue capacity does not mean much more goodput – small flows don't benefit

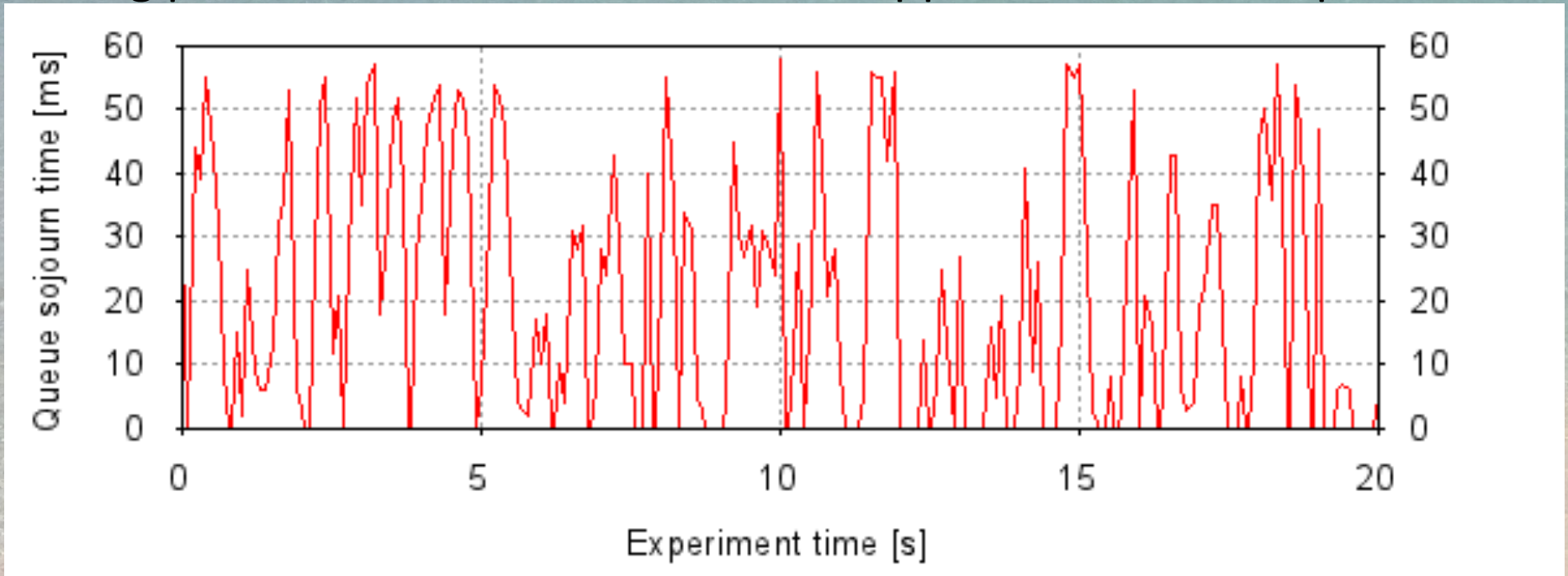


Queue capacity [kB]	Max. queu time [ms]
50	
70	
100	
120	
150	
200	
250	

Example: 16 Mbps sim
GEO satellite link

Baselines – tracking TCP queue oscillation

- Ping packets are small and seldom dropped – RTT shows quick rises and falls: oscillation!



Example: 16 Mbps simu
GEO satellite link, 120 k
50 channels

Pings every 100 ms

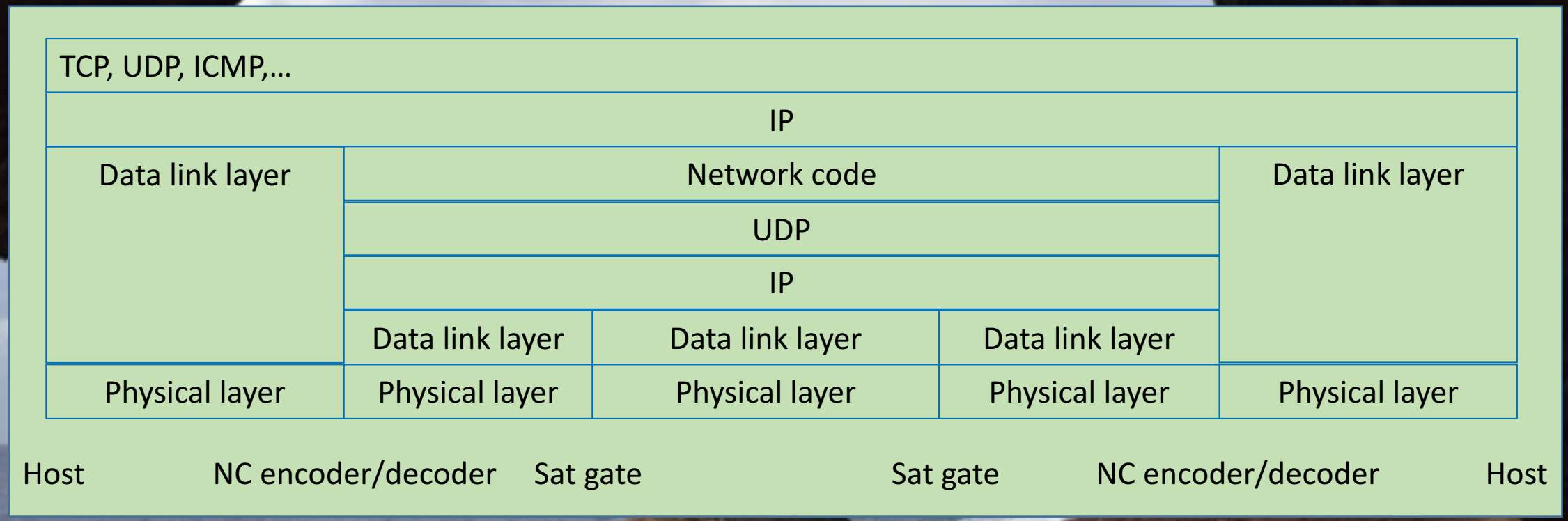
Max. possible sojourn ti

Baselines – conclusion

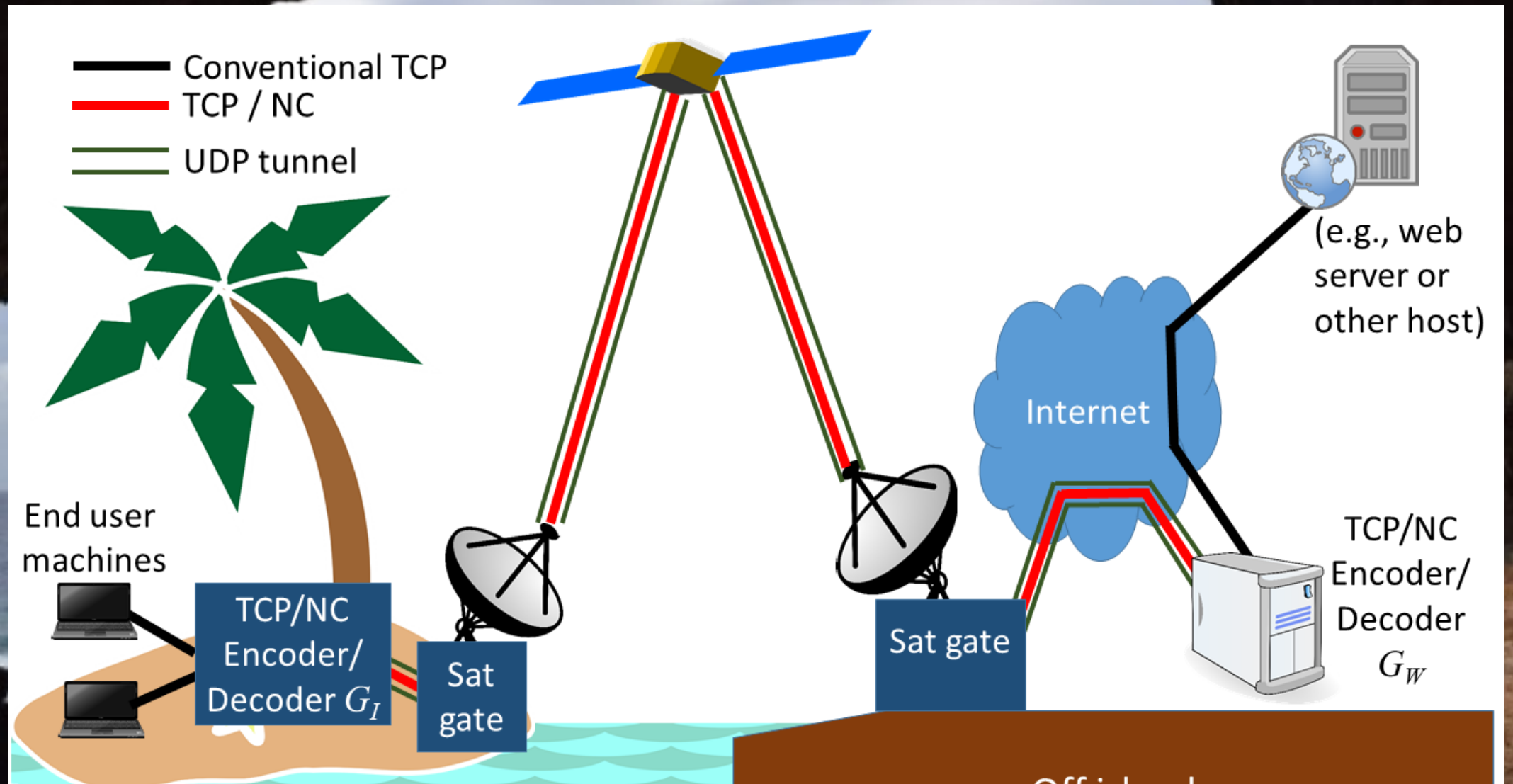
- Baselines let us determine good queue capacities
- From queue oscillation to buffer bloat
- Can see what is actually happening

- Each data point takes ~20 minutes to compute
  - Use “poster cases”: 8, 16, 32, 64 Mbps GEO, 32, 64, 160, 320 Mbps MEO

Coded tunnels



TCP over network coding – basic setup:



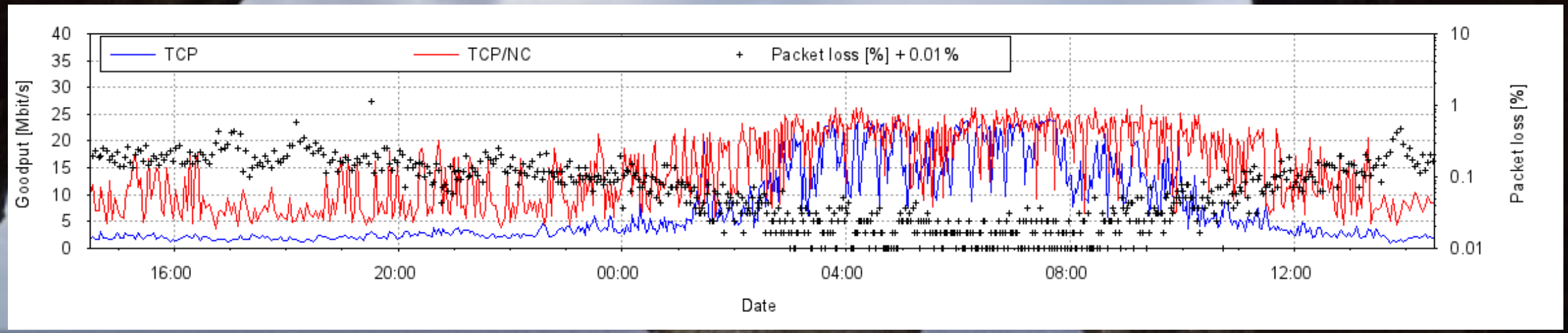
Nuts and bolts: Network coding (TCP/NC)

- Insight: IP packets consist of bytes, which are just binary numbers. Can multiply / add them.
- Can combine N IP packets p_i byte-wise into a "linear combination packet" (network coding) by multiplying each packet with a "random" coefficient and adding the products. The j-th combination packet includes the coefficients $c_{j,i}$ and the sum r_j:

$$c_{1,1}p_1 + c_{1,2}p_2 + c_{1,3}p_3 + \ldots + c_{1,N}p_N = r_1 \rightarrow (c_{1,1}, c_{1,2}, c_{1,3}, \ldots, c_{1,N}, r_1)$$
$$c_{2,1}p_1 + c_{2,2}p_2 + c_{2,3}p_3 + \ldots + c_{2,N}p_N = r_2 \rightarrow (c_{2,1}, c_{2,2}, c_{2,3}, \ldots, c_{2,N}, r_2)$$
$$c_{3,1}p_1 + c_{3,2}p_2 + c_{3,2}p_3 + \ldots + c_{3,N}p_N = r_3 \rightarrow (c_{3,1}, c_{3,2}, c_{3,3}, \ldots, c_{3,N}, r_3)$$

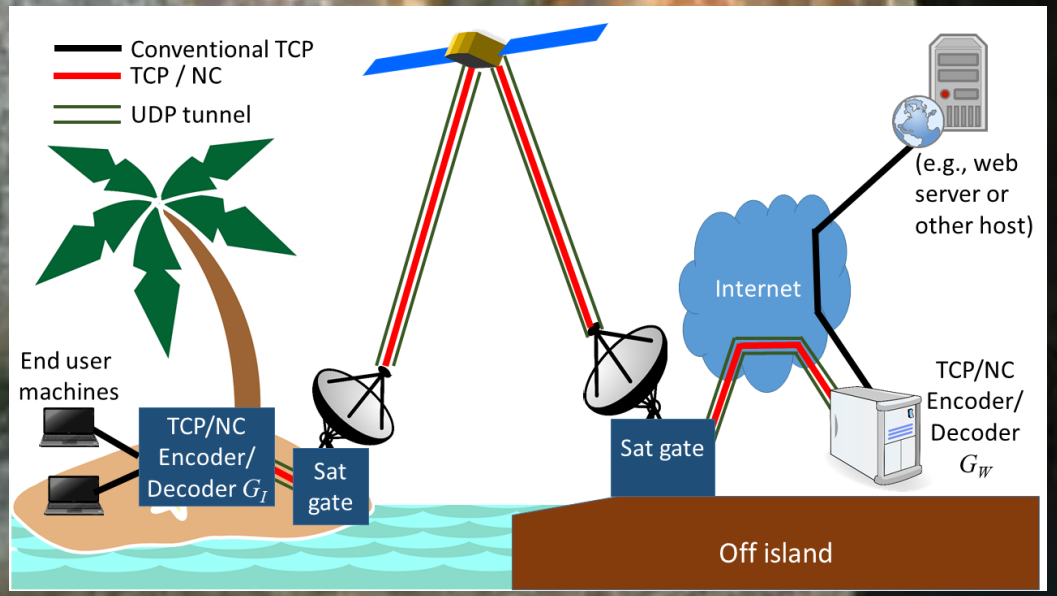
- Instead of sending the N data packets across the satellite link, we send M > N combination packets – essentially, we generate an *overdetermined system of linear equations* whose solution is the set of original packets
- Smart computation: For $i \leq N$, set $c_{i,i}$=1 and all other coefficients zero and add logs rather than multiply
- Receiver solves the system to recover the original packets p_i

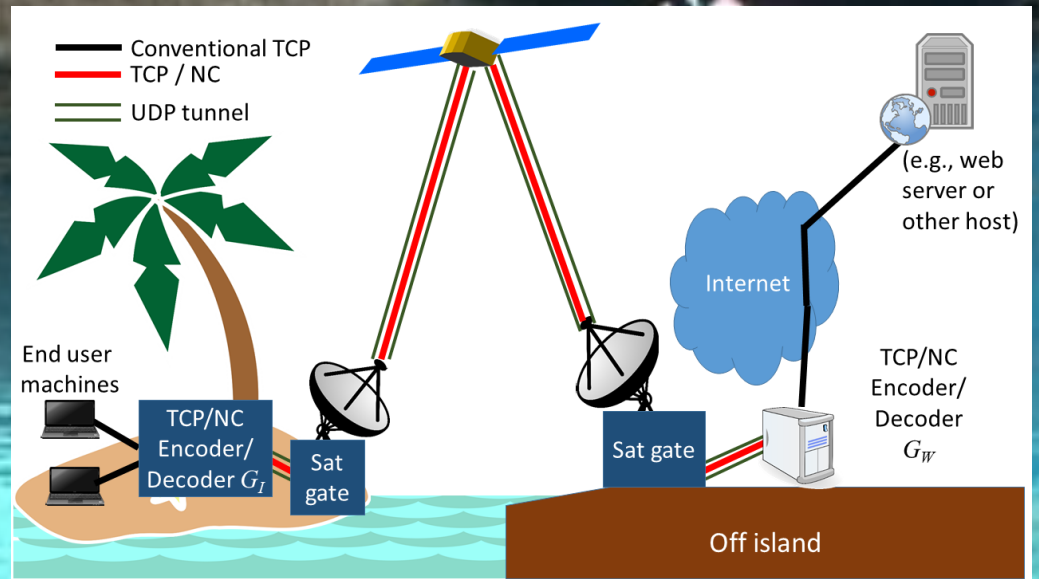
Coded tunnels
Field results



Rarotonga link 29-30 July, 2015

Heading back to the lab again





Can we code an entire island?

- Not that easy - topology is different
- First results show that the extra M-N coded packets get dispersed in our original tunnel – which is good!
- In the whole-of-island coding scenario, the encoder currently sends them immediately after the first N packets.
- That's bad – when we need them most, they hit the overflowing queue with 1 Gbps and no gaps!
- Small flows don't benefit much from coding – no congestion window to open

- Needs work on the encoder
  - Downrating of extra packets
  - Small flow bypass?

Conclusion

Pacific satellite Internet links are a bit unusual

The simulator gives us a powerful tool to investigate them

But: Complex to “drive” – we’re still learning!

Current research focuses on whole-of-island coding – needs work on encoder

Thank you!

ulrich@cs.auckland.ac.nz

Project partners, collaborators and funders

- Bluesky, Telecom Cook Islands Ltd.
- CAIDA, University of California San Diego / San Diego Supercomputer Center
- Information Society Innovation Fund (through APNIC)
- Internet Niue
- Internet NZ
- Massachusetts Institute of Technology
- Pacific Island Chapter of the Internet Society (PICISOC)
- Steinwurf ApS, Denmark
- Tuvalu Communication Corporation
- University of Auckland
- University of Ulm
- Victoria University of Wellington

ulrich@cs.auckland.ac.nz