Simulating satellite Internet performance on a small island

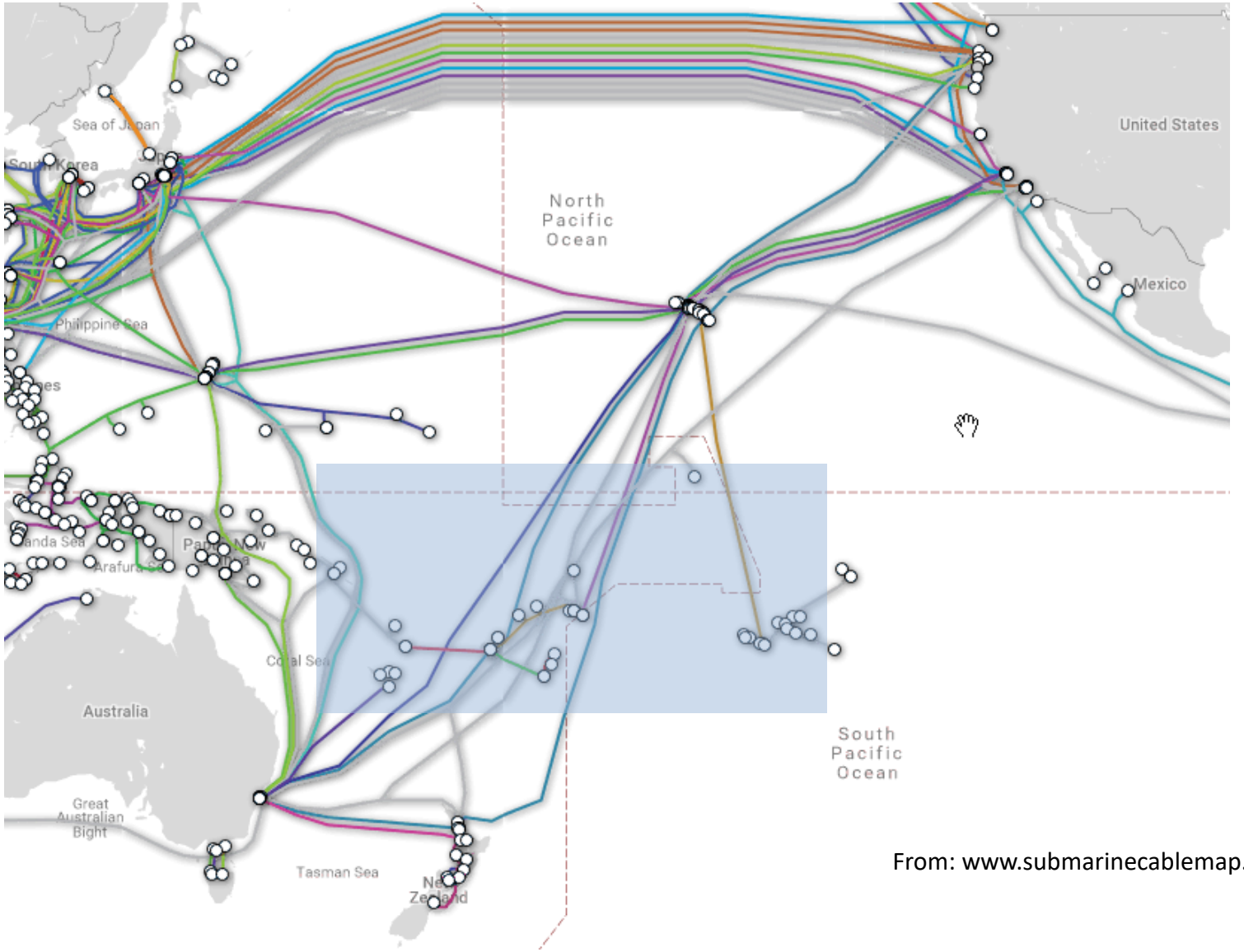
Ulrich Speidel
Lei Qian

Department of Computer Science

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

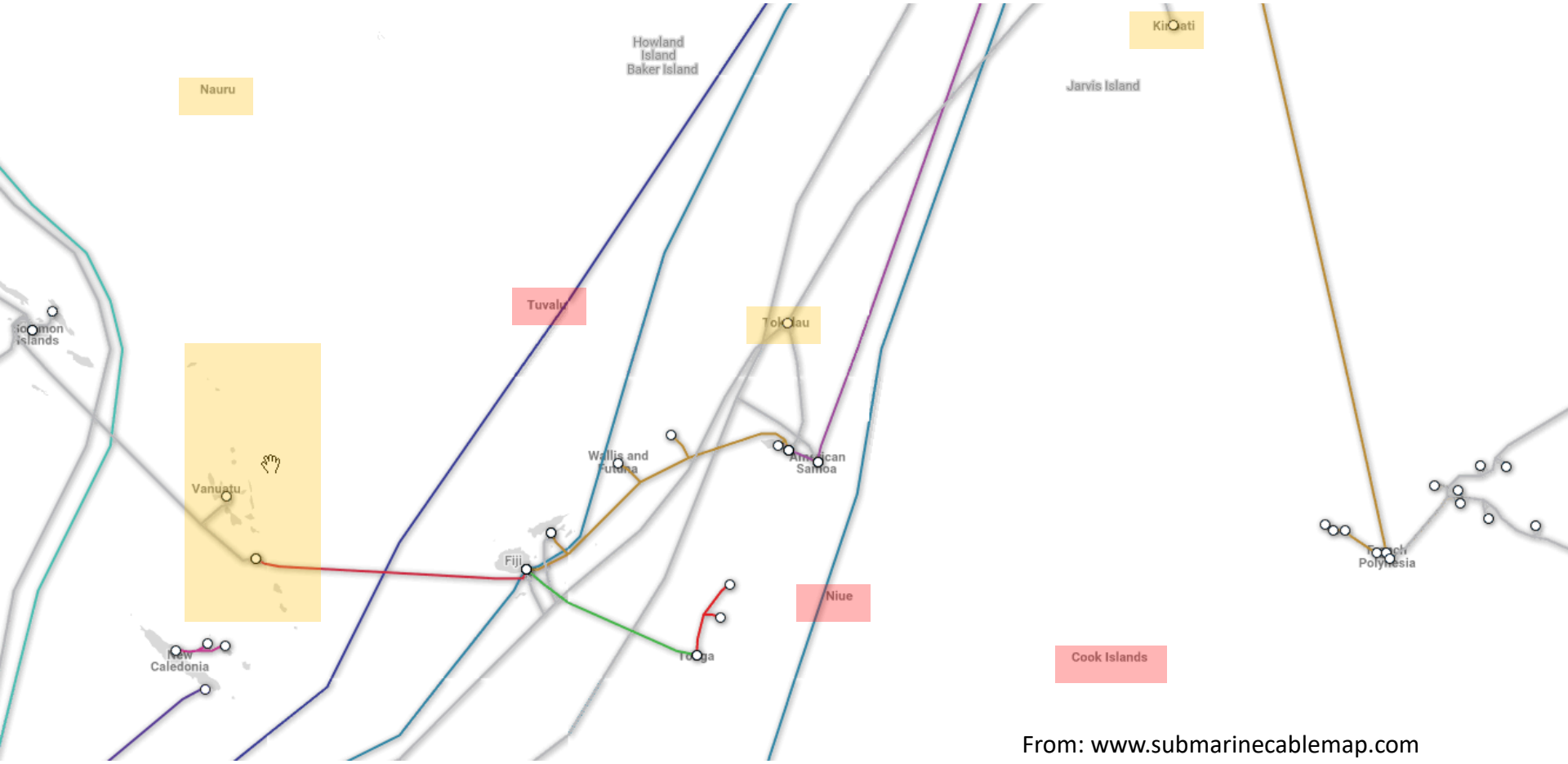
August 2018

Fibre connectivity in the Pacific



From: www.submarinecablemap.com

Fibre connectivity in the Pacific



From: www.submarinecablemap.com

Satellite Internet on small islands

- Satellite connections used on islands
  - long latency
    - 500 ms RTT to Internet on GEO, 120+ ms on MEO (typ. < 100 ms on cable)
  - low bandwidth
    - typ. 1 Mbps GEO – several hundred Mbps MEO (typ. >= 1 Gbps on cable)
- Poor TCP performance
  - small transfers such as small emails or browsing are often OK
  - large files (attachments, pictures, software) cannot be transferred in reasonable time
- Often have both a lot of packet loss and link underutilization
- Increasing digital divide! Can we do something about this?

Motivation for the present work

- Many island satellite links suffer from link underutilisation even when demand is heavy
- Would like to learn how to utilise precious narrowband satellite link bandwidth more efficiently
- Would like to improve quality of service on satellite links

Cause of link underutilisation

- Greedy individual TCP connection tries to send as much as possible by increasing congestion window size.
- TCP congestion control algorithm will slash window size once congestion at the satellite link input queue is detected.
- Long delay/RTT on narrowband satellite link leads to TCP senders accelerating and backing off in a cyclic fashion
- Multiple TCP senders synchronise into cycle
- The link underutilisation is caused by **queue oscillation!**

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 see noise or interference

Potential solutions

- Performance enhancing proxy (PEP)
  - TCP connection splitting
  - partially solves long RTT problem
  - Creates potentially tricky failure modes
- Coding-based solutions
  - Encode/decode packets instead of just forwarding
  - Linear coding
  - Forward error correction/concealment

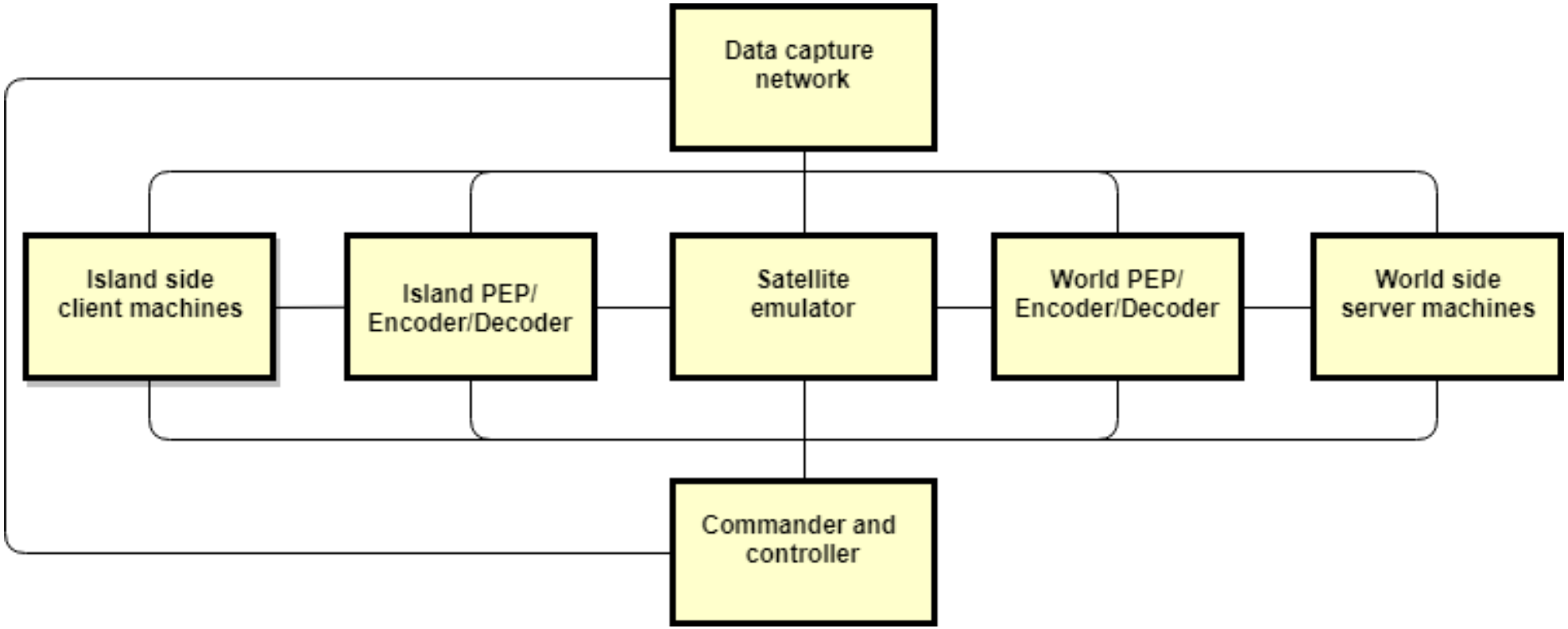
Research questions

- Can we overcome link underutilisation by coding packets?
- How do we make more efficient use of precious low link bandwidth?
- Do we need to encode/decode all or just some of the packets?
- ***In this talk: How do we demonstrate what is happening on a link with and / or without coding, PEP, etc.?***

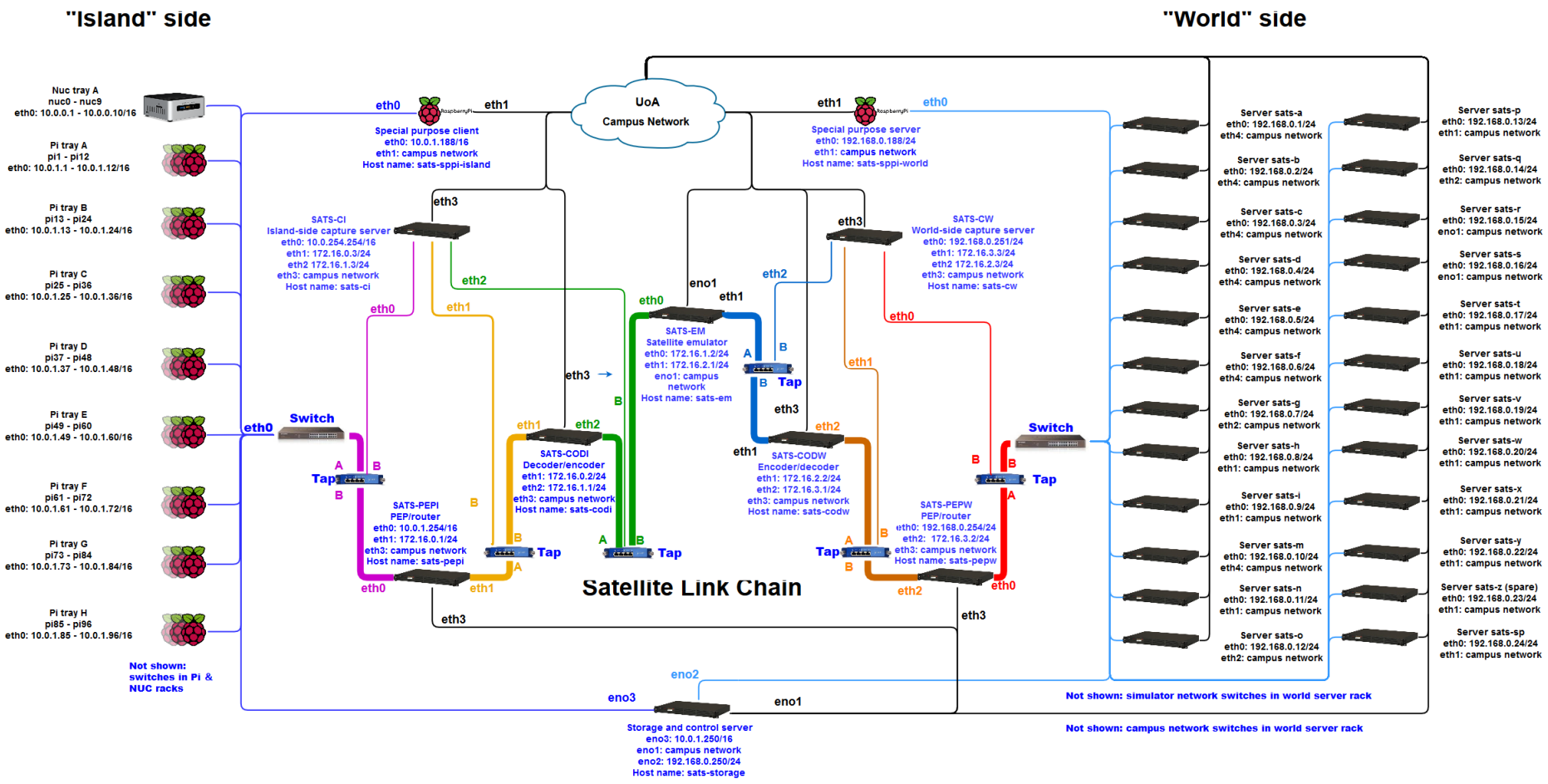
Simulation of real world traffic

- Why simulate in hardware?
  - Impossible to achieve all the criteria below on a real link
  - Pure software simulators not powerful/realistic enough
- Link type
  - Want to simulate GEO/MEO delay, jitter, different bandwidths, queue capacities
- Demand level
  - Need enough traffic generation while controlling the demand
- Demand profile
  - Want realistic flow distributions: mix of different flow sizes
- Terrestrial latency profile
  - Want configurable latency distribution between world-side gateway and different parts of the world

Simulation topology



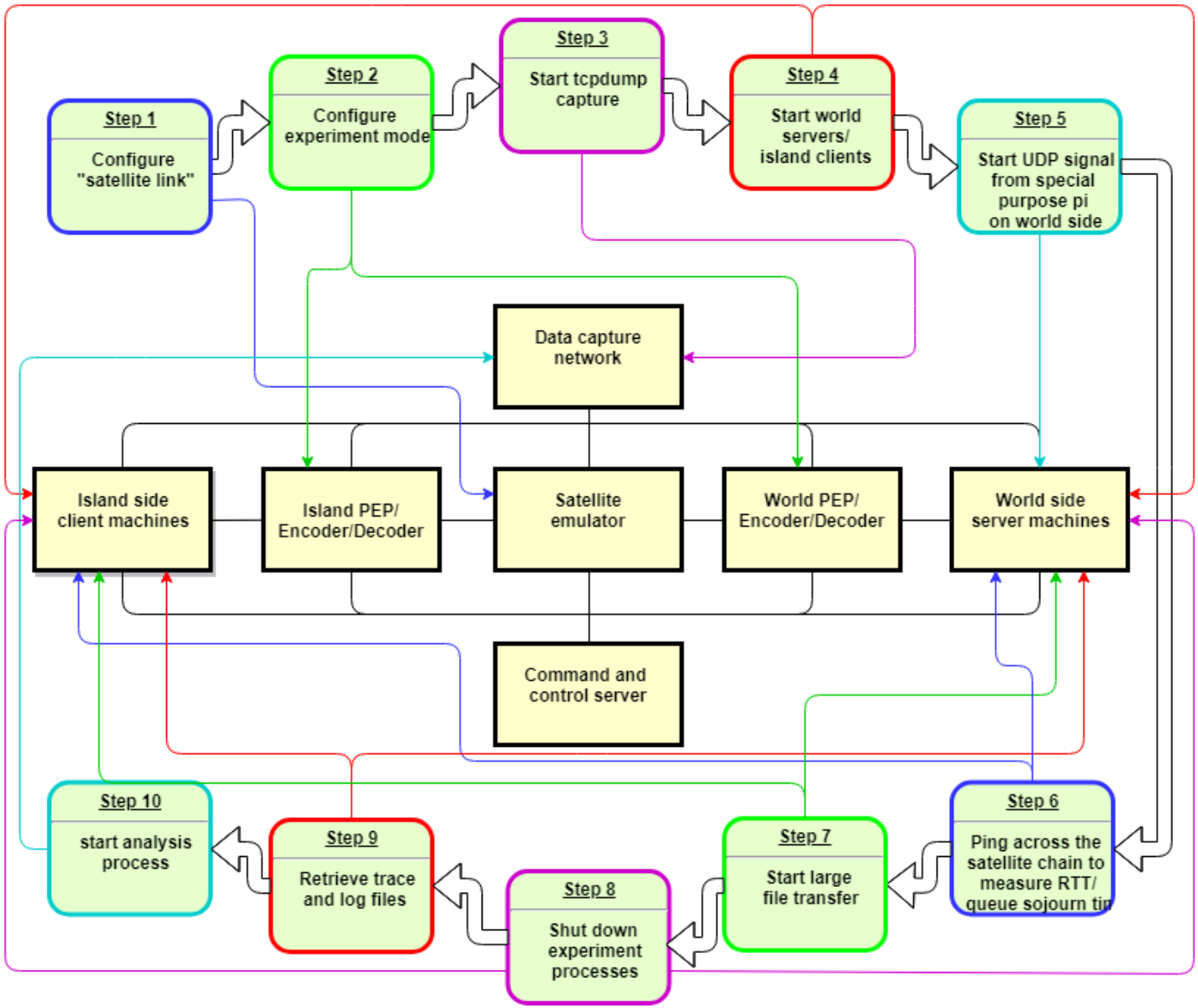
Simulation topology



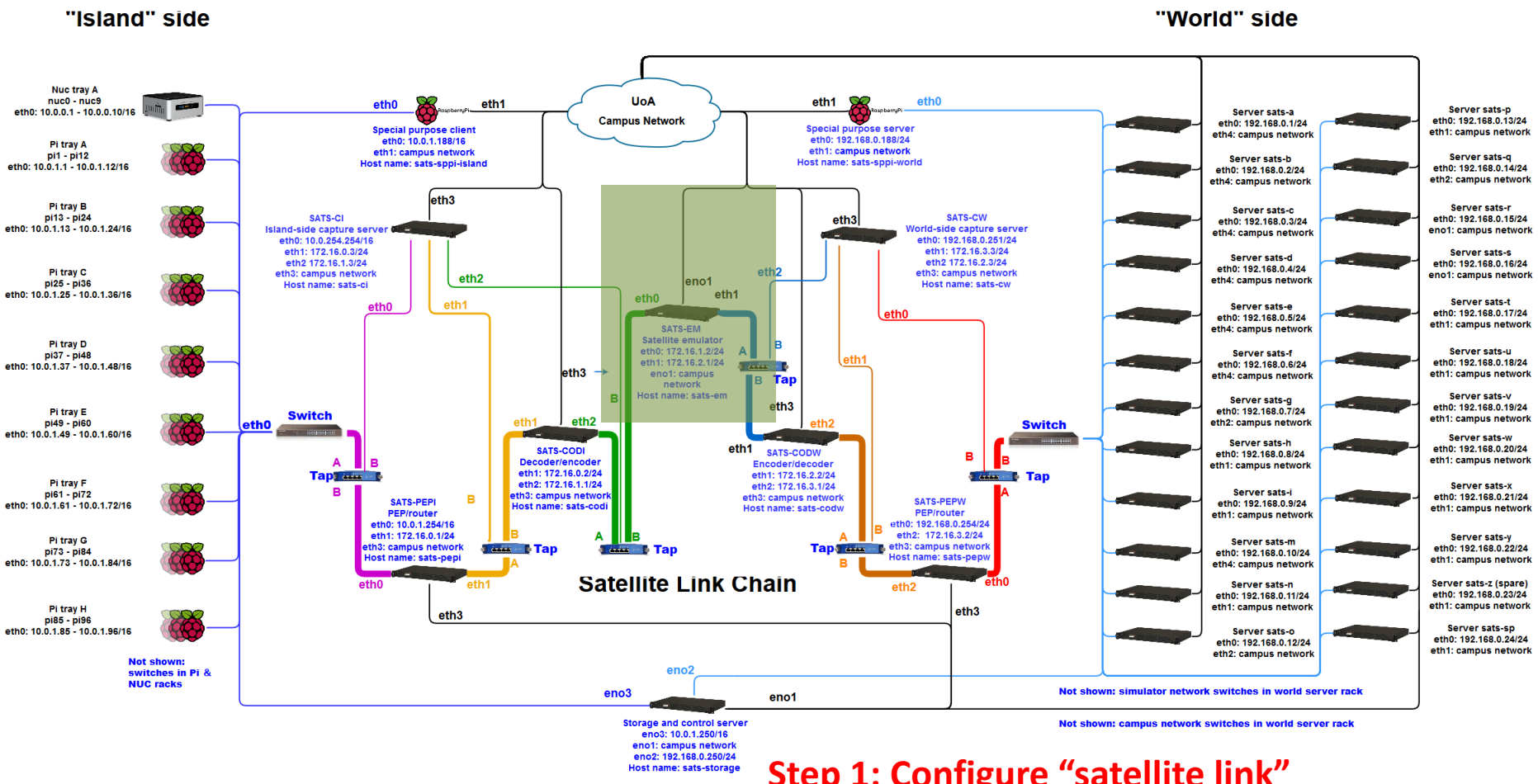
Simulation configuration

- Purpose-built software
  - Purpose-developed client/server software
  - Able to fulfill controlled demand level requirements
- Standard tools
  - Reachability, traffic dumping, remote control and traffic injection
  - Standard tool sets in Linux. E.g., ping, tcpdump, ssh/pssh, iperf3
- Experiment control and analysis software
  - Scripting of experiment and post experiment analysis
  - Shell scripting and PHP

Simplified experiment procedure

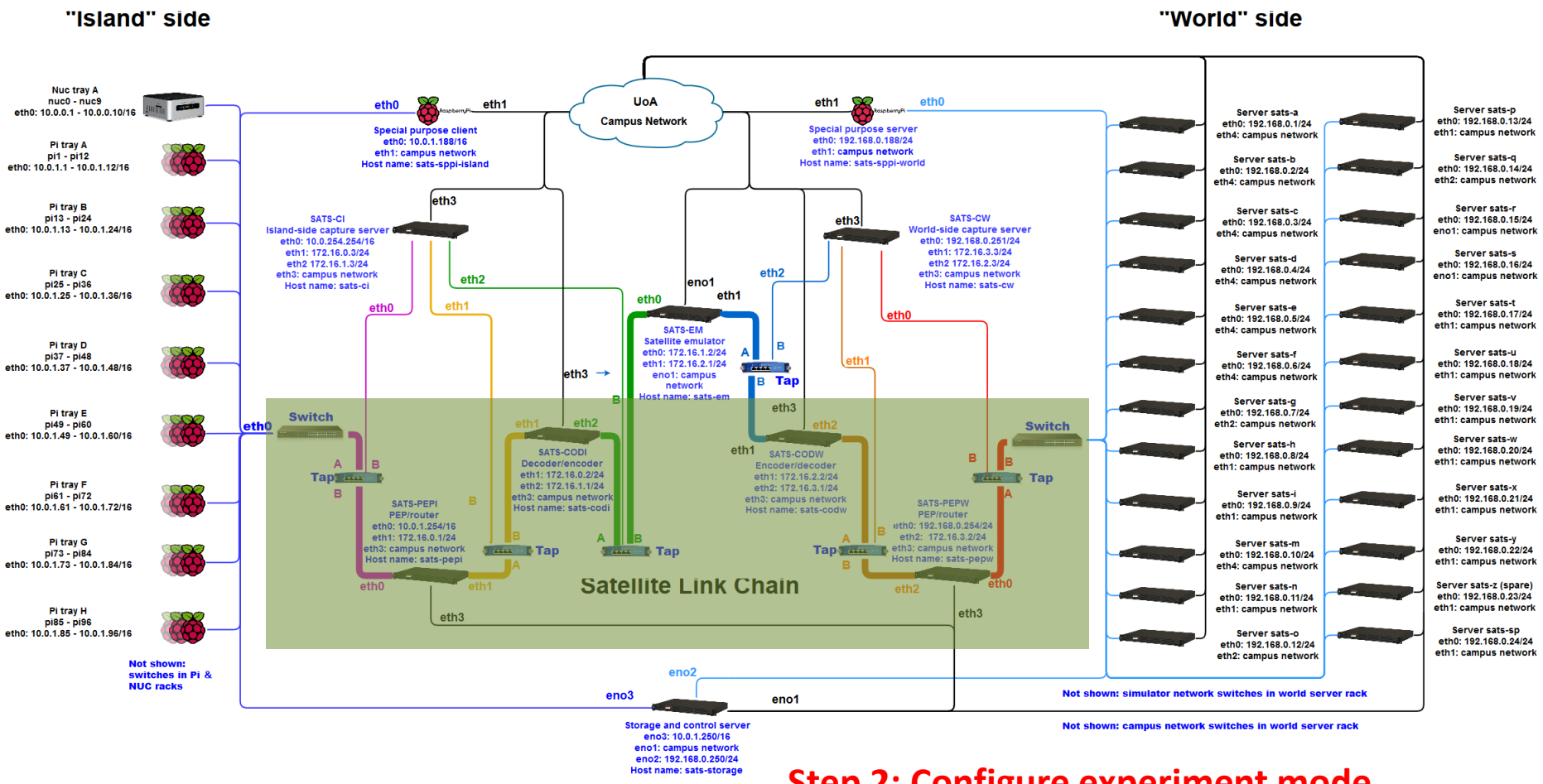


Experiment step 1



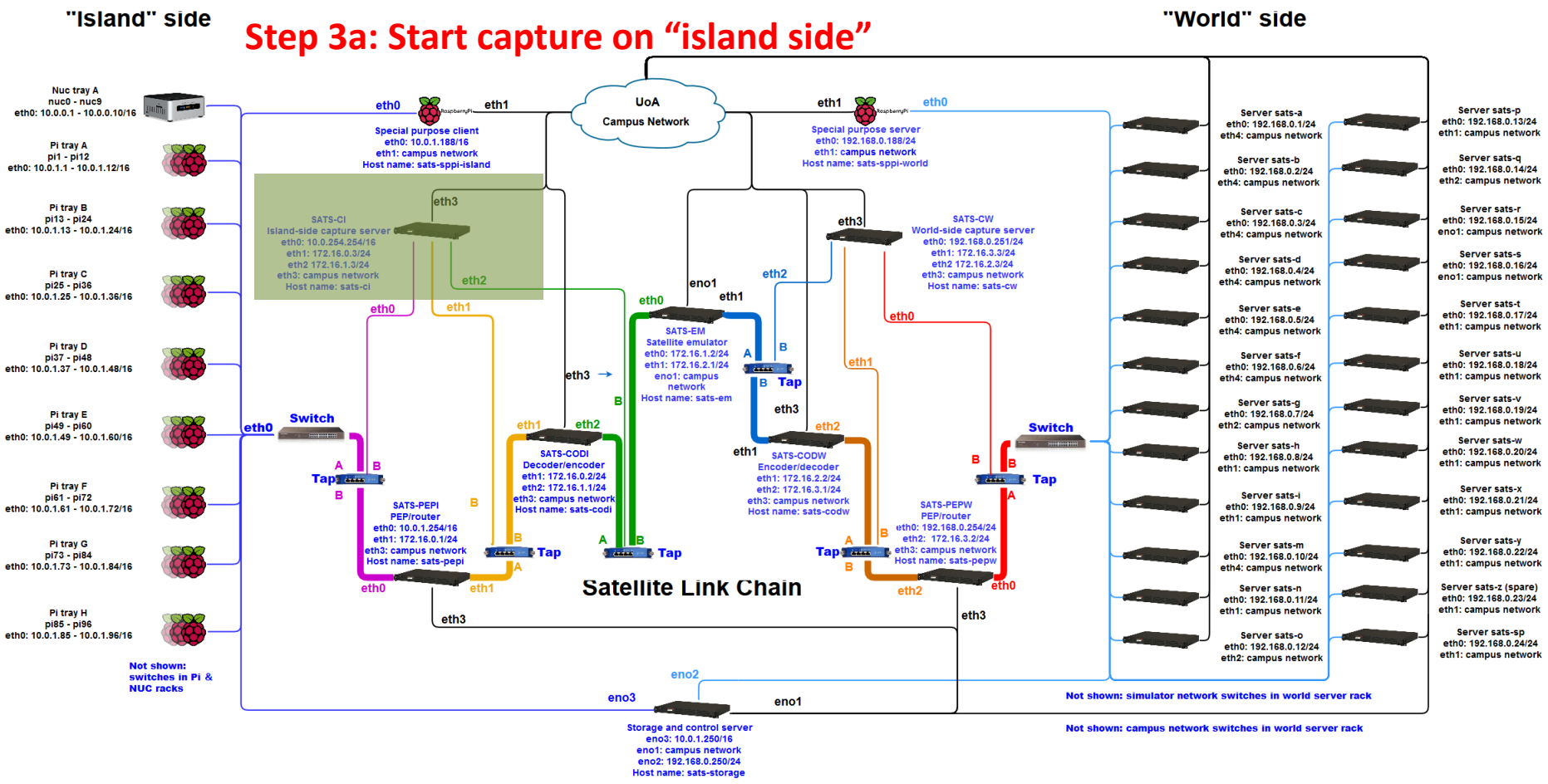
Step 1: Configure “satellite link”

Experiment step 2

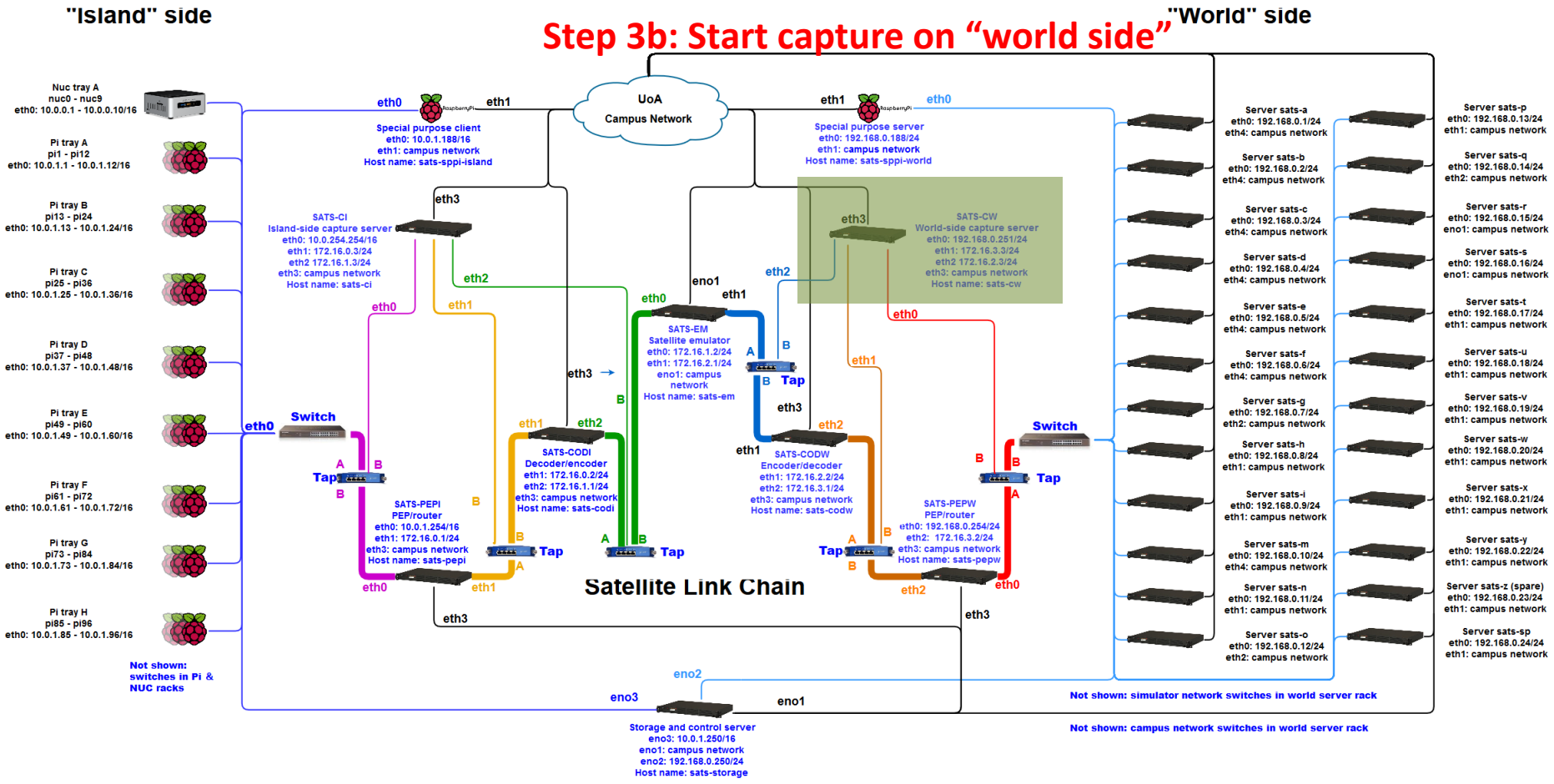


Step 2: Configure experiment mode

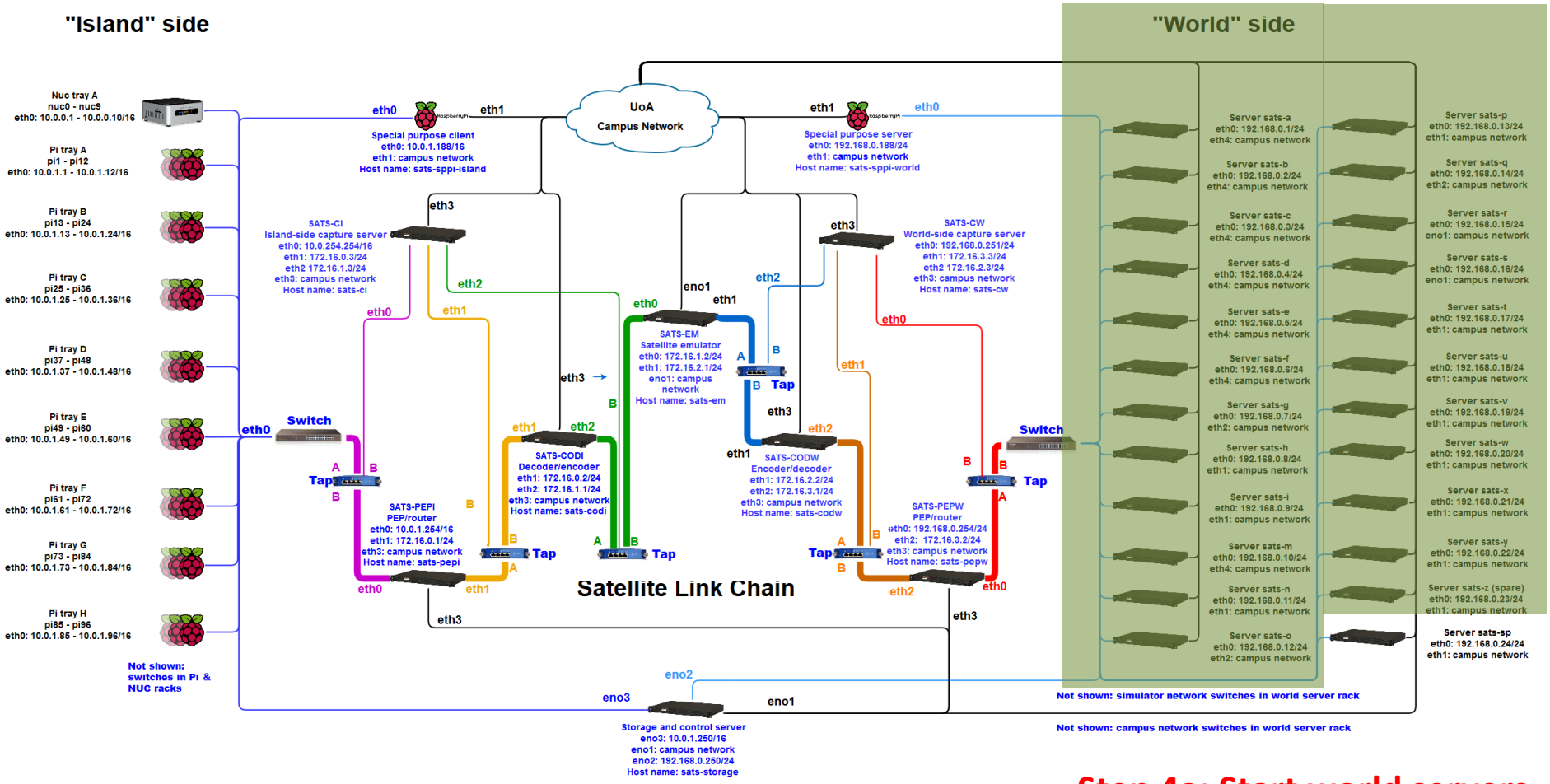
Experiment step 3a



Experiment step 3b

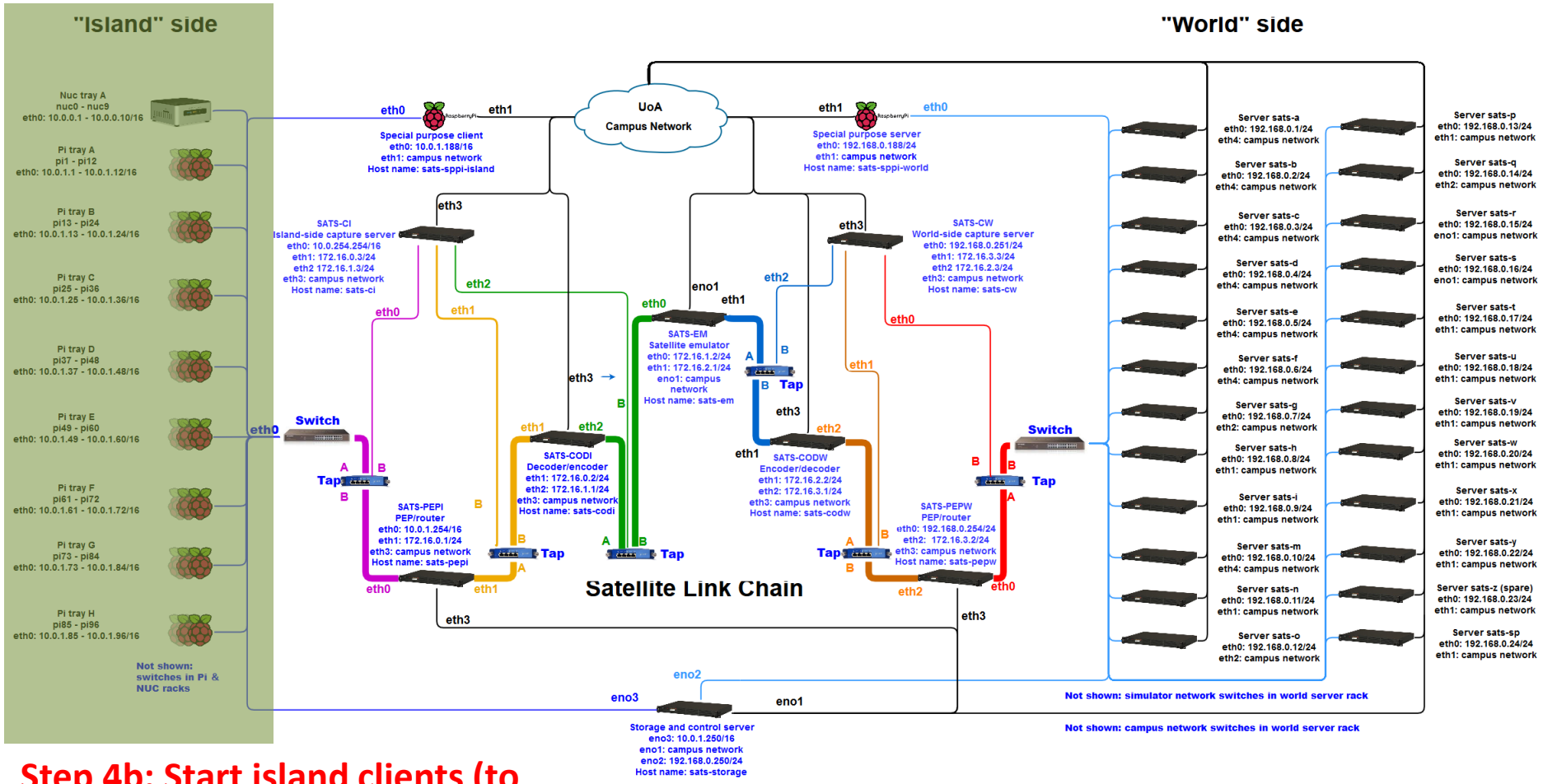


Experiment step 4a



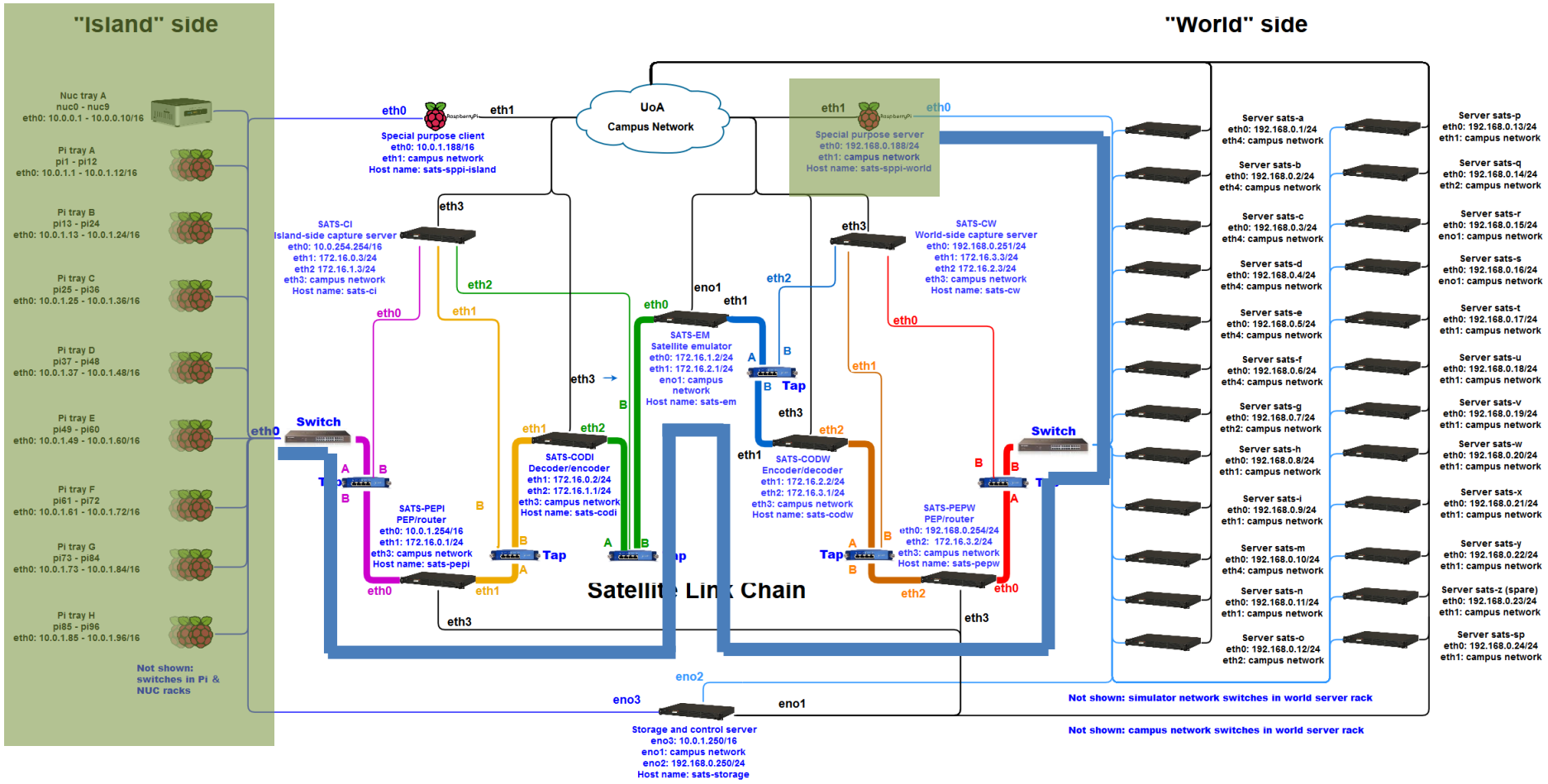
Step 4a: Start world servers (to provide background traffic)

Experiment step 4b



Step 4b: Start island clients (to create the demand for background traffic)

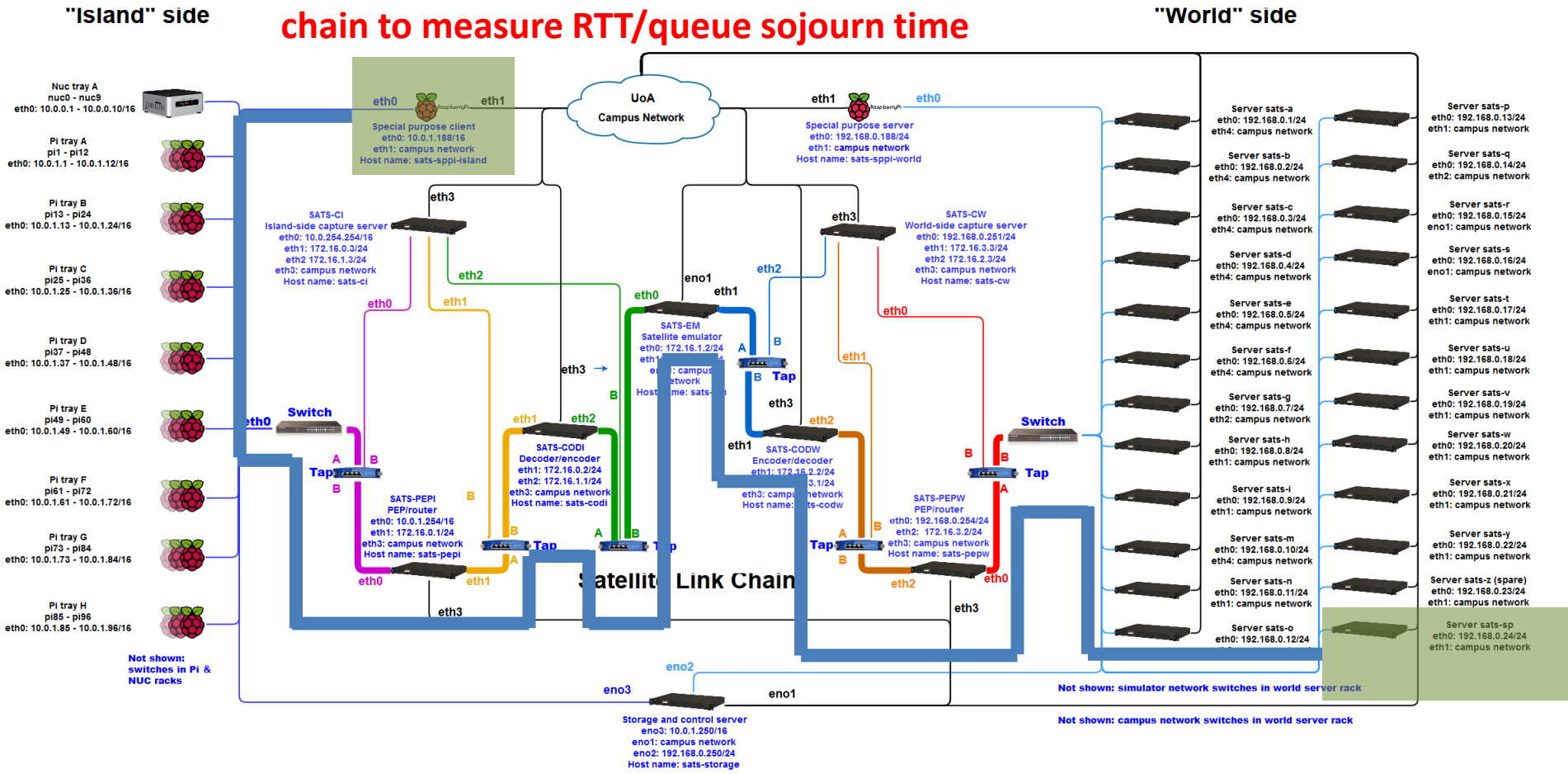
Experiment step 5



Step 5: Send UDP start signal from world special purpose Pi to island side ("experiment start whistle")

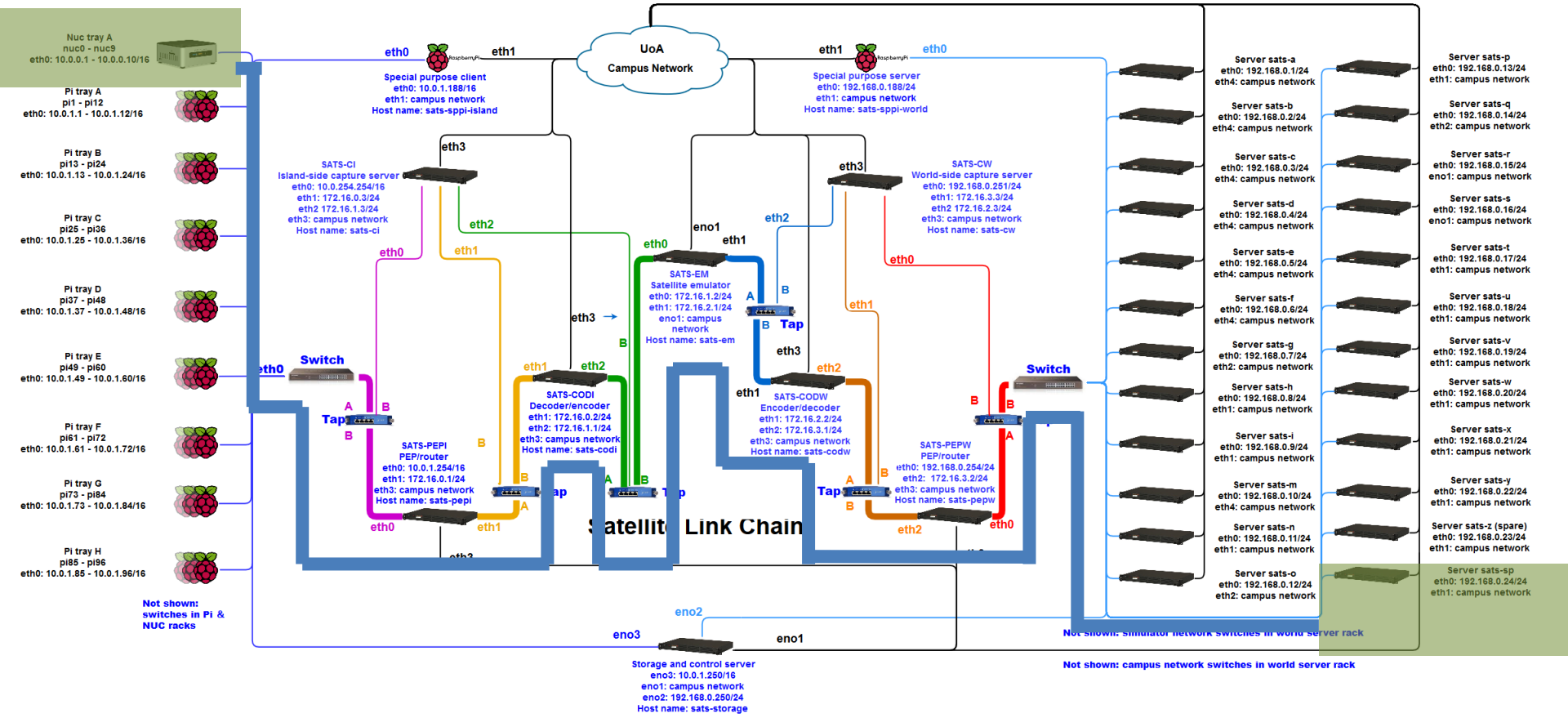
Experiment step 6

Step 6: Start to ping every 100 ms across the satellite chain to measure RTT/queue sojourn time

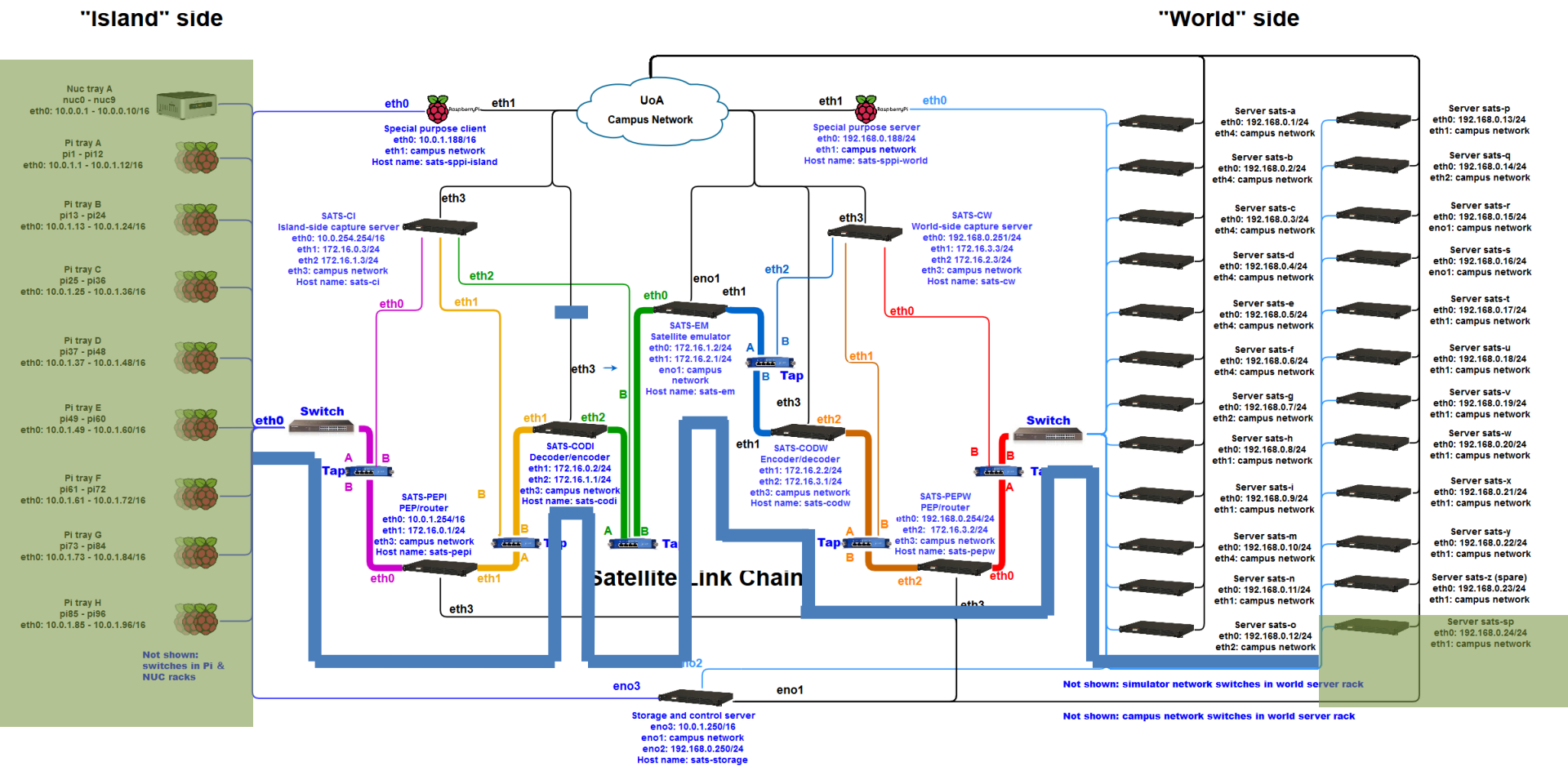


Experiment step 7

Step 7: Start large file transfer from sats-sp to one of the NUCs

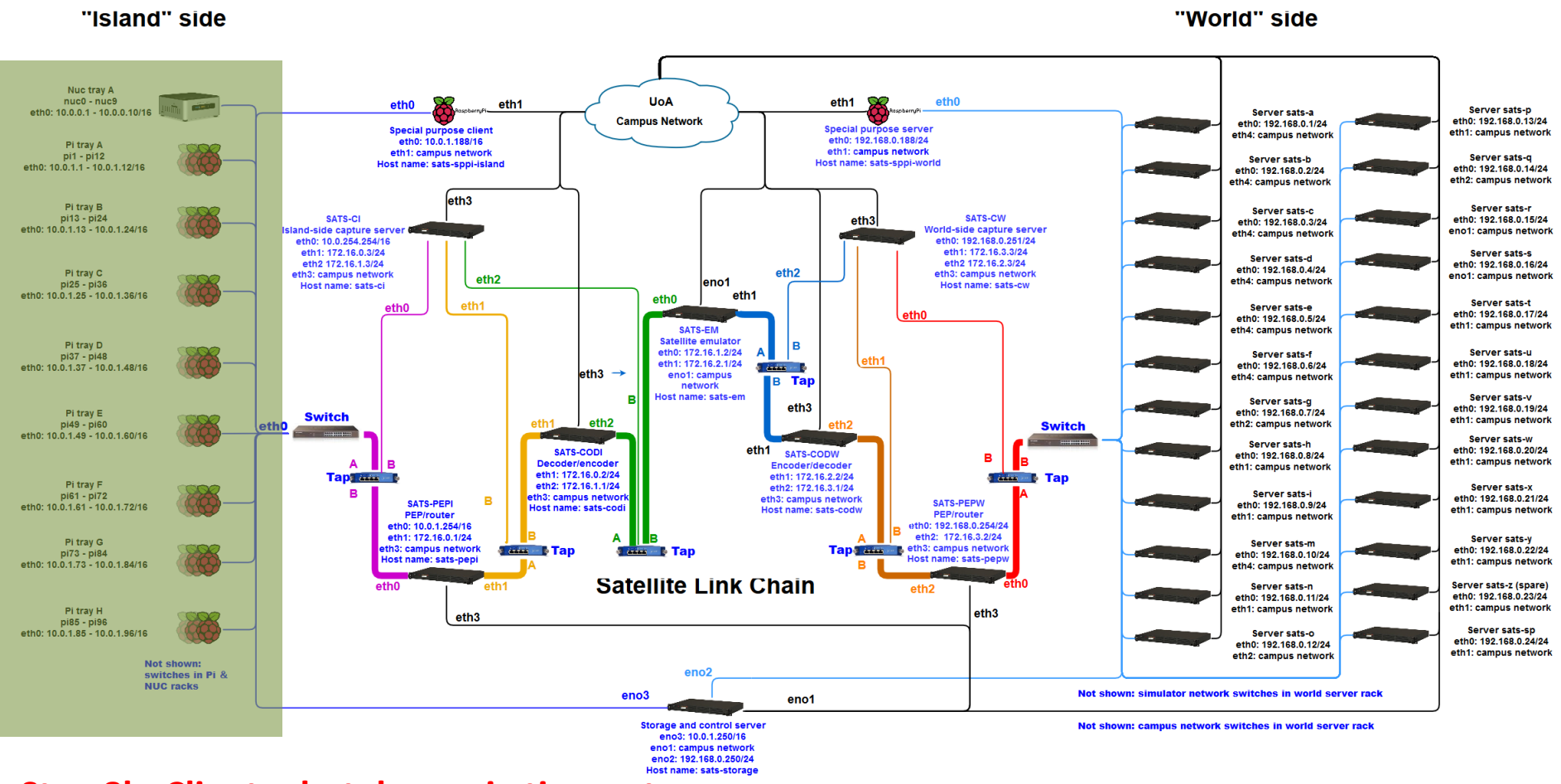


Experiment step 8a



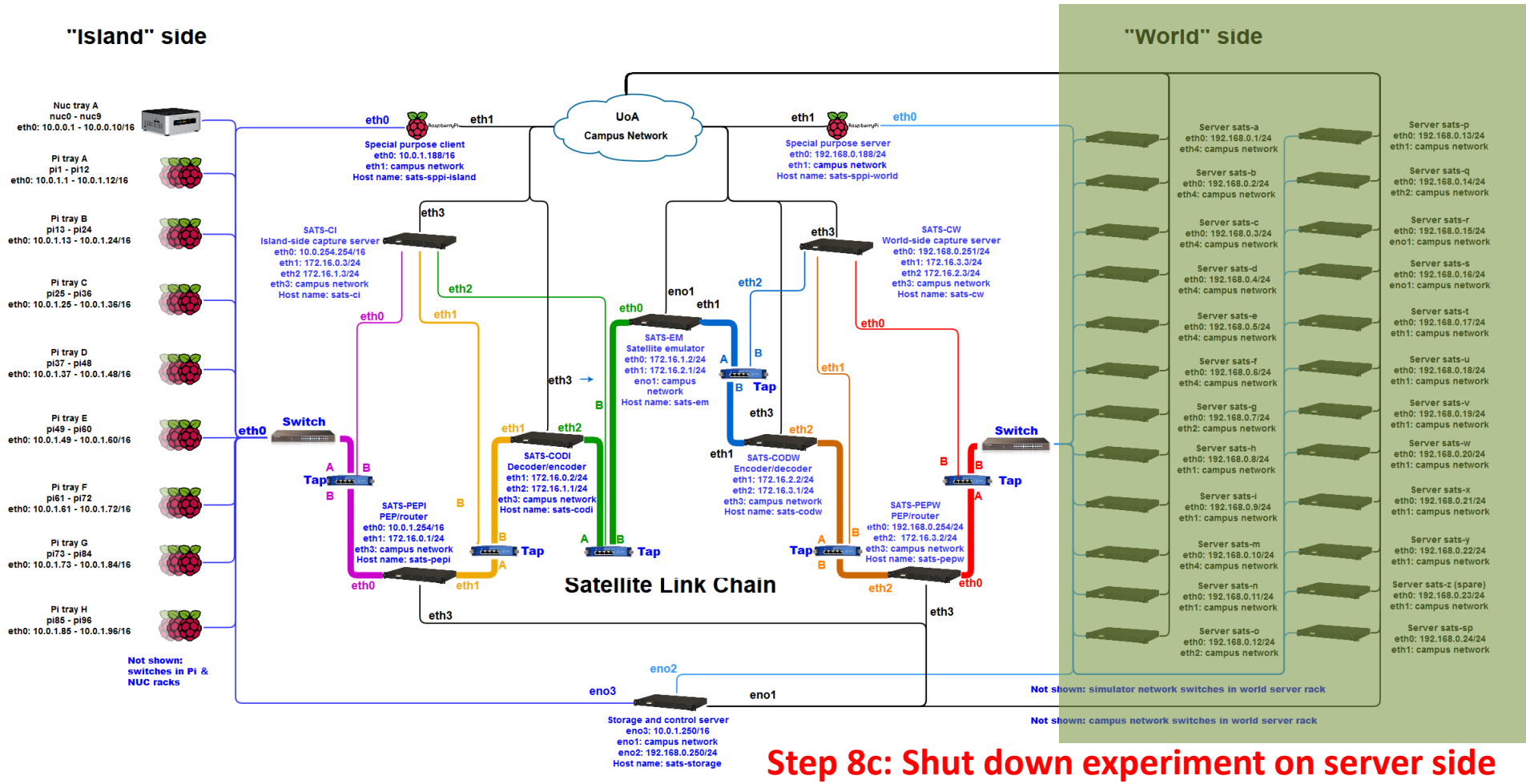
Step 8a: After 30-600 seconds after start signal, send UDP stop signal from world special purpose Pi to island side ("experiment stop whistle")

Experiment step 8b



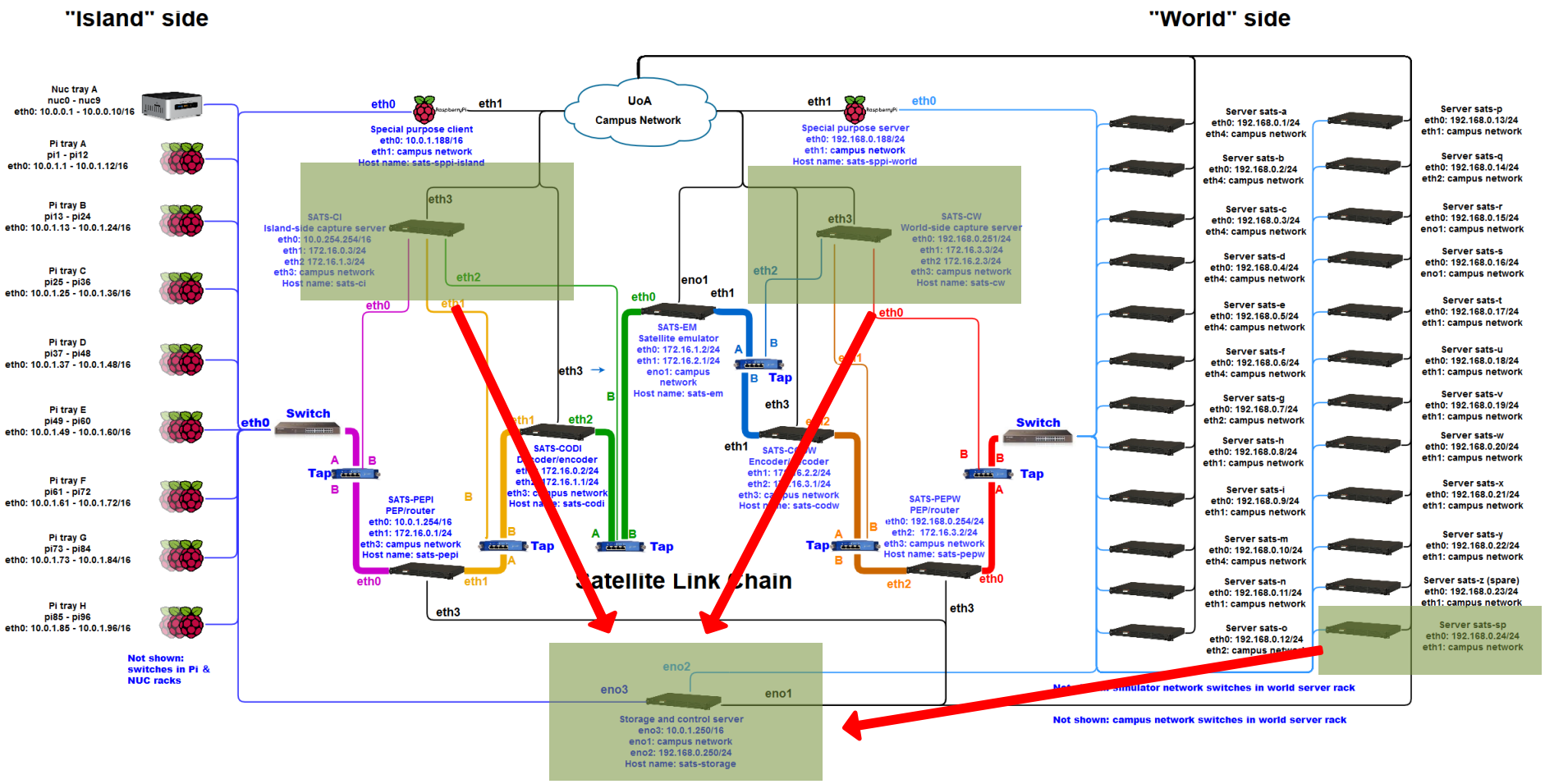
Step 8b: Clients shut down via timeout

Experiment step 8c



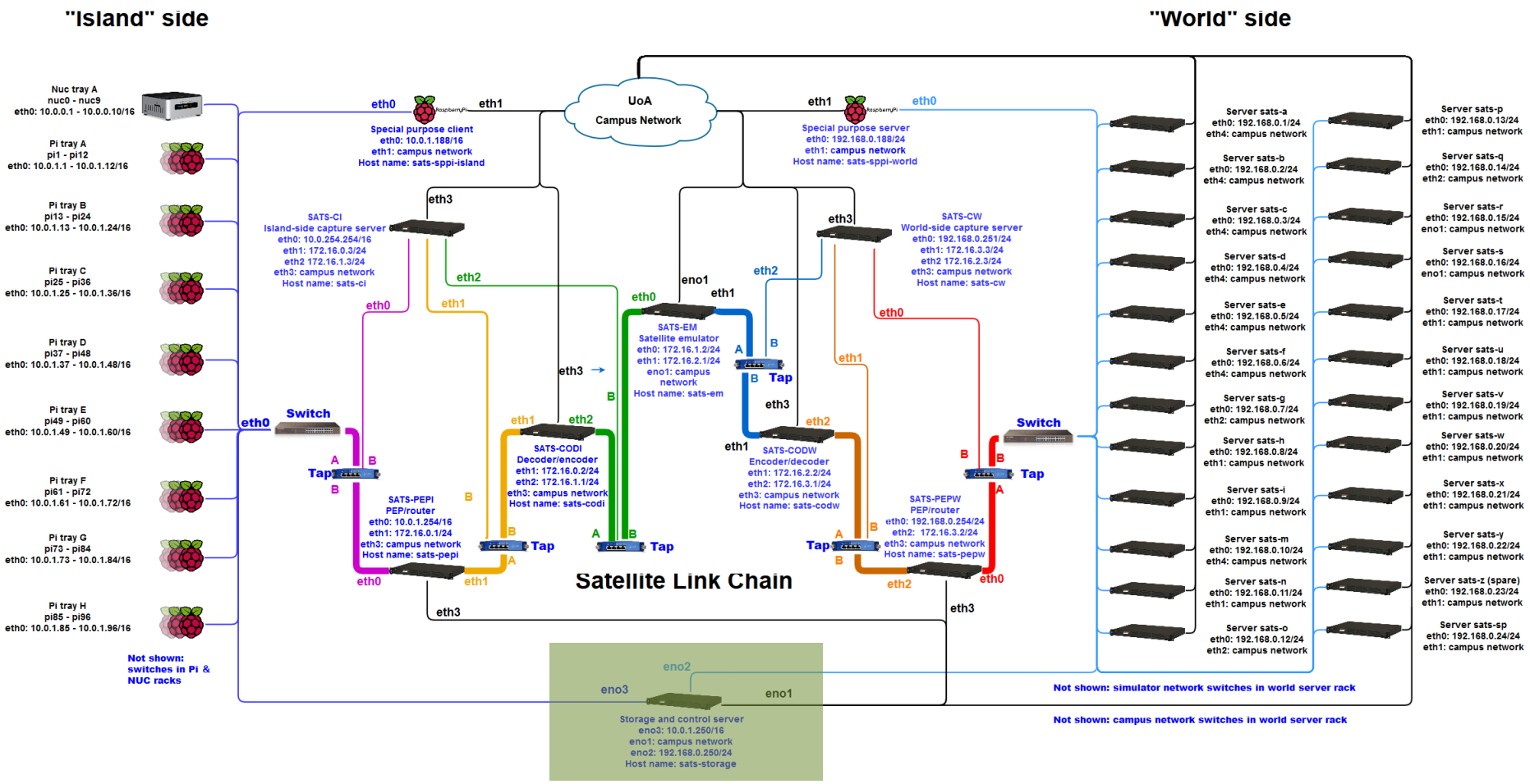
Step 8c: Shut down experiment on server side

Experiment step 9



Step 9: Retrieve trace and log files

Experiment step 10

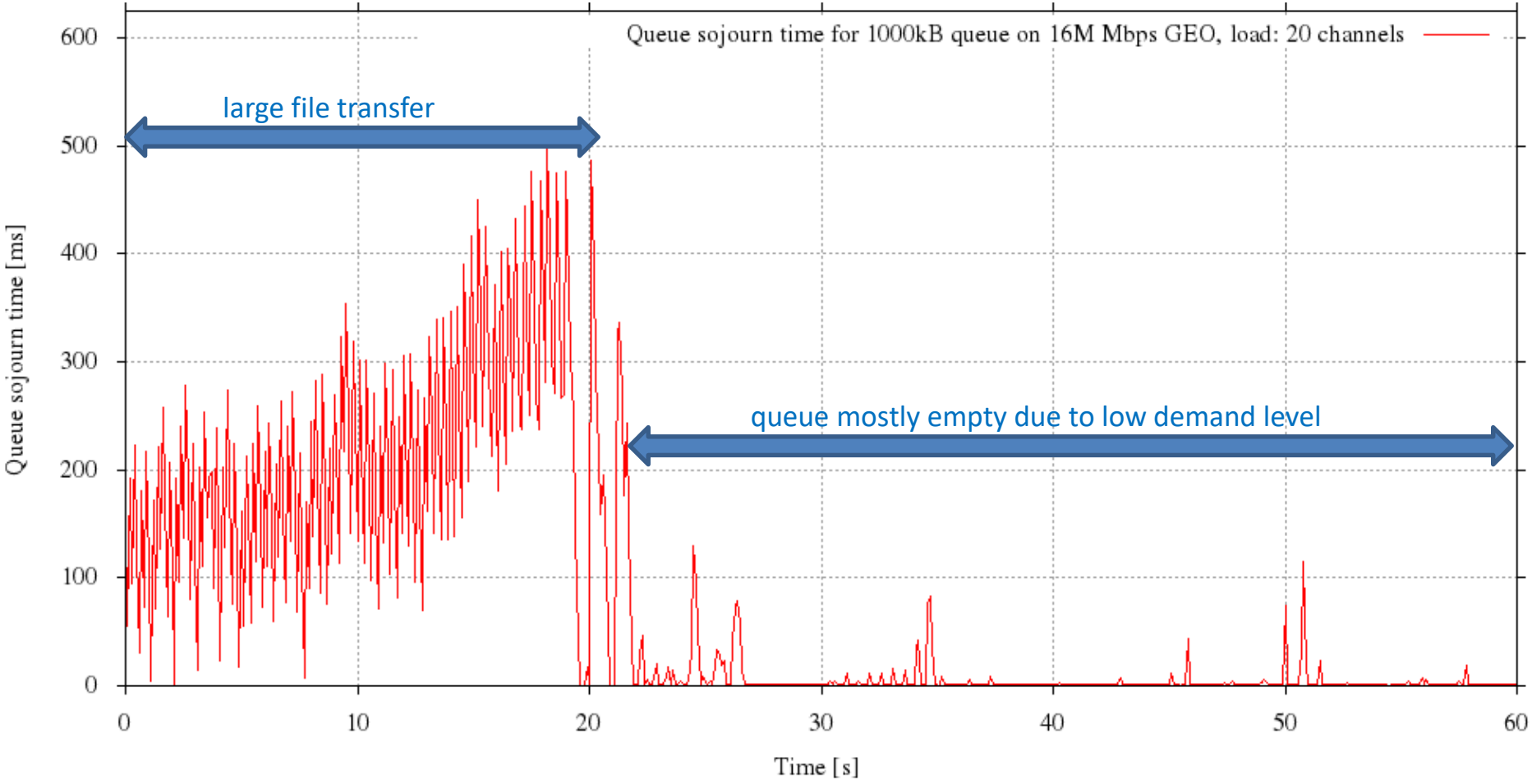


Step 10: Start analysis process

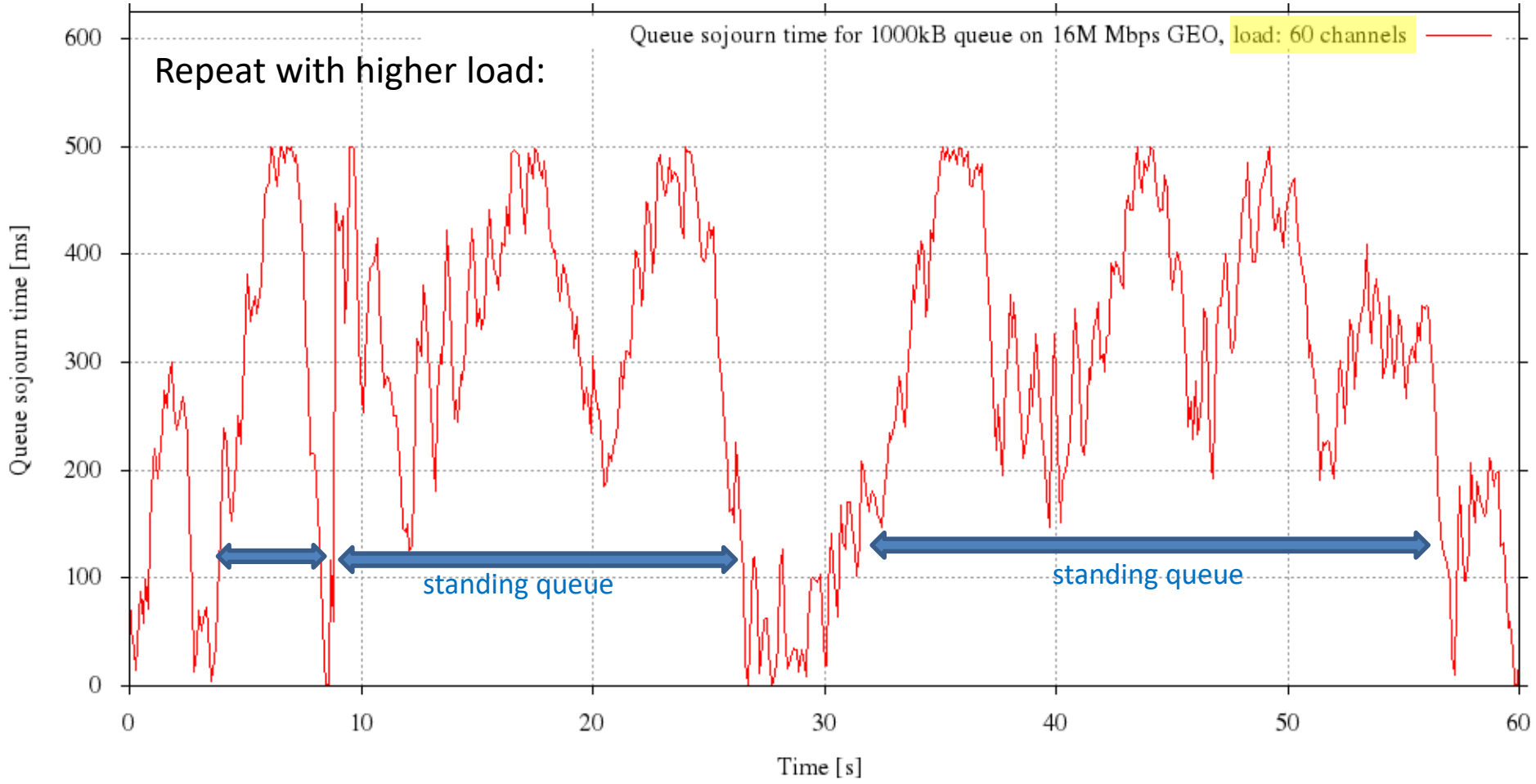
Experiment – quality assurance

- Pre-experiment checks
  - Link setup confirmation
  - Reachability test (are all RTTs what we expect?)
  - Link test (bandwidth + max. queue sojourn time)
- Post-experiment checks
  - Experiment running time verification (Start and stop signals present in traces?)
  - Active node verification (Did all expected nodes participate? Did they share the load?)

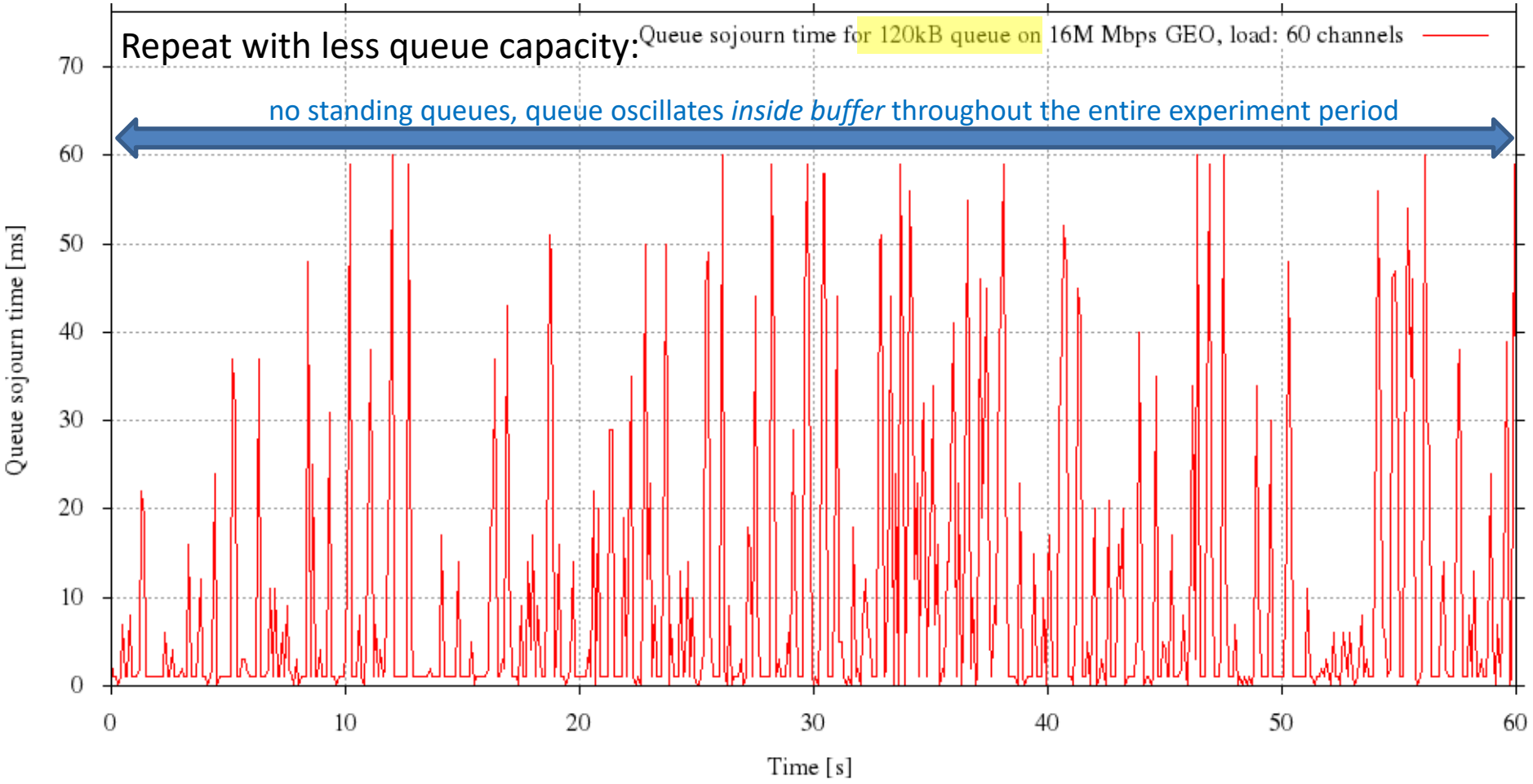
Selected result 1 of 4



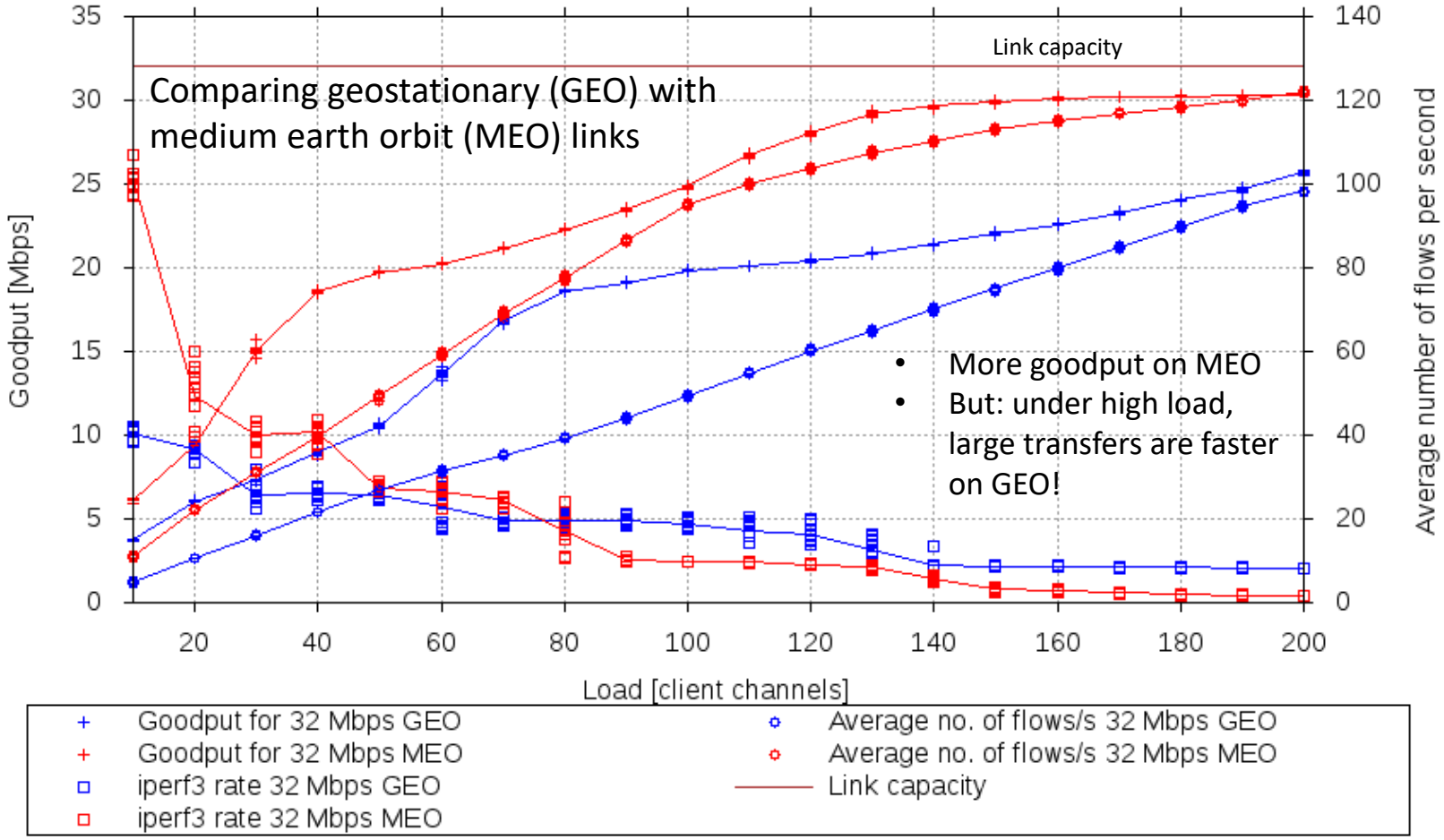
Selected result 2 of 4



Selected result 3 of 4



Selected result 4 of 4



Conclusion and future work

- Hardware based simulation of traffic is feasible.
- Can replicate TCP queue oscillation and show dependence on queue capacity and load.
- Small and short flows present a serious problem on satellite links.
- Currently trying to work on forward error correction coding across the input queue. Hoping for better link utilisation and overall goodput improvement.
- Need to strike balance between redundancy, goodput gain and spare capacity on coded links.

Acknowledgements

- Thanks for valuable comments to Nevil Brownlee, Brian Carpenter and many more.

- Thanks to APNIC and Internet NZ for supporting this project with multiple grants.

Questions ?