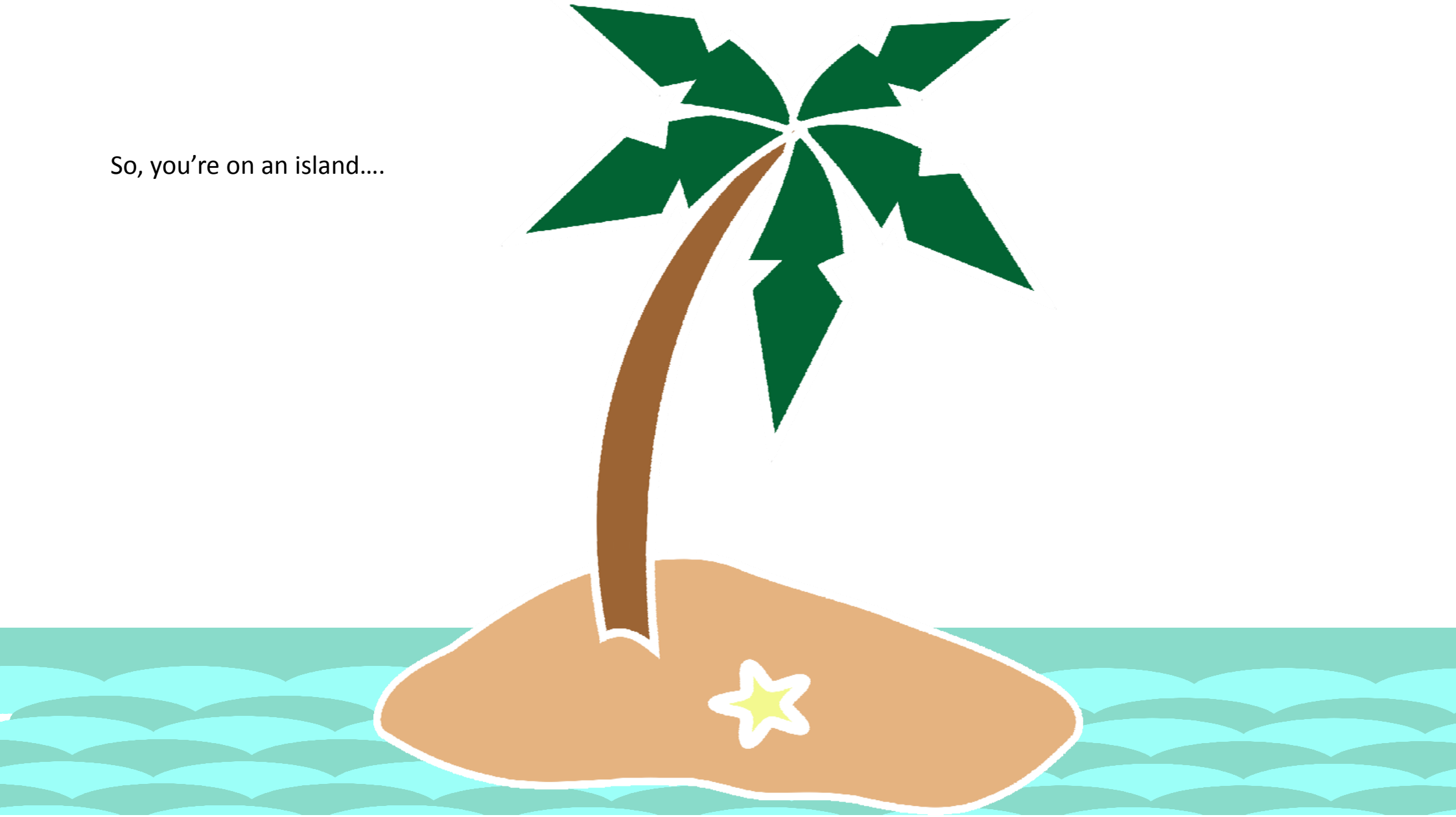
Can network coding bridge the digital divide in Pacific Island countries?

Ulrich Speidel, Lei Qian, ‘Etuate Cocker, Péter Vingelmann, Janus Heide, Muriel Médard

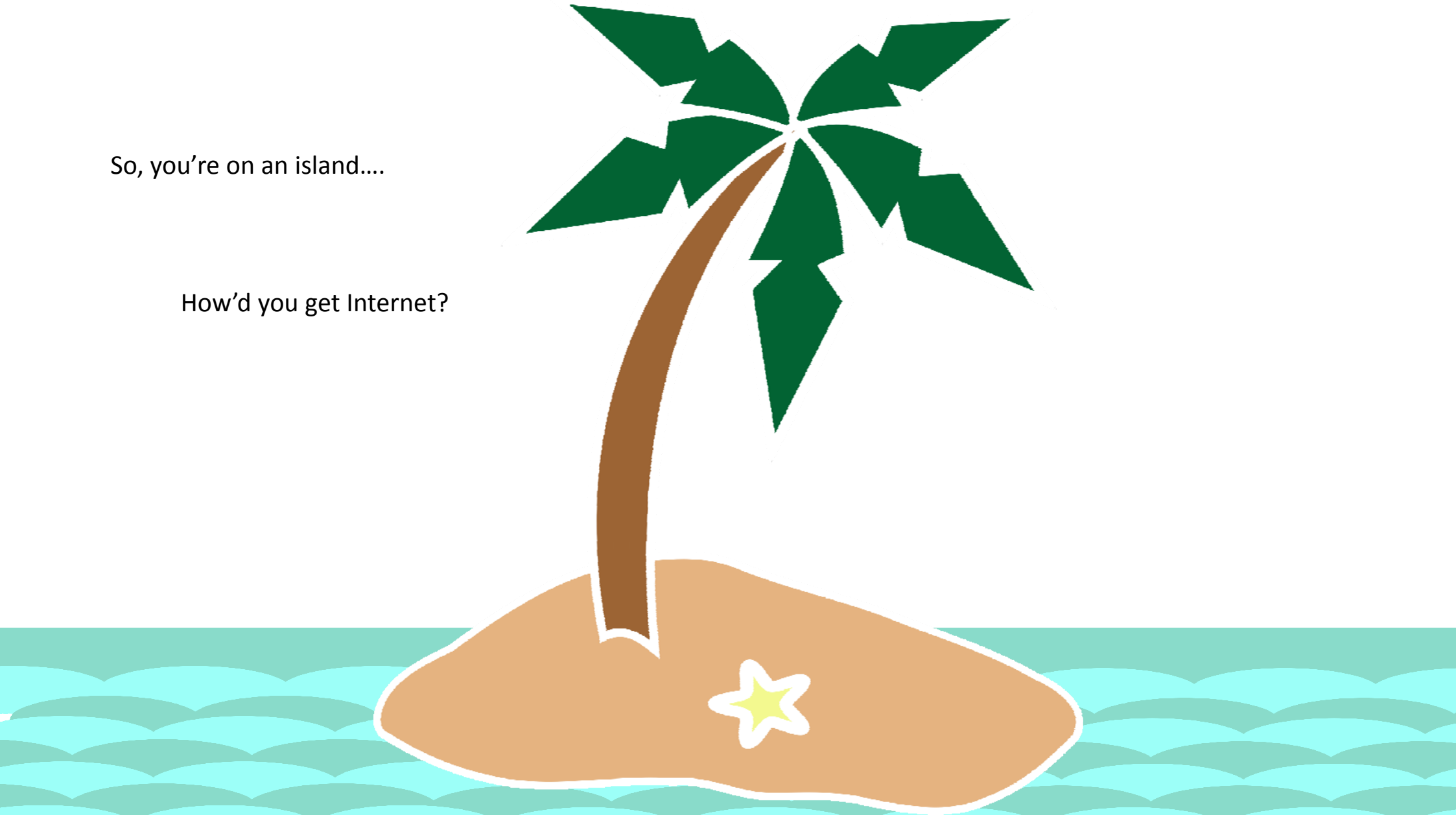
ulrich@cs.auckland.ac.nz

So, you’re on an island….



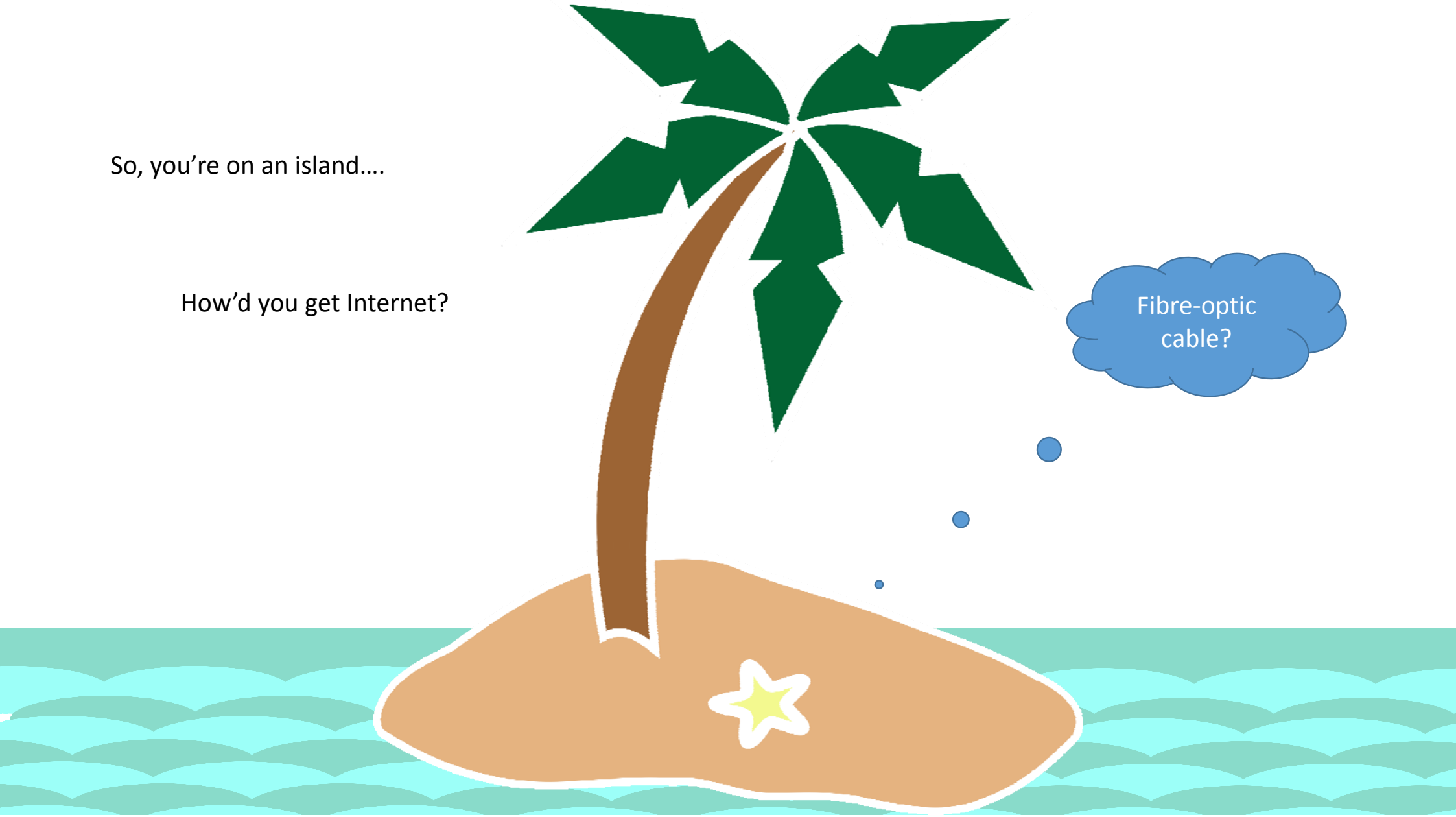
So, you're on an island….

How'd you get Internet?



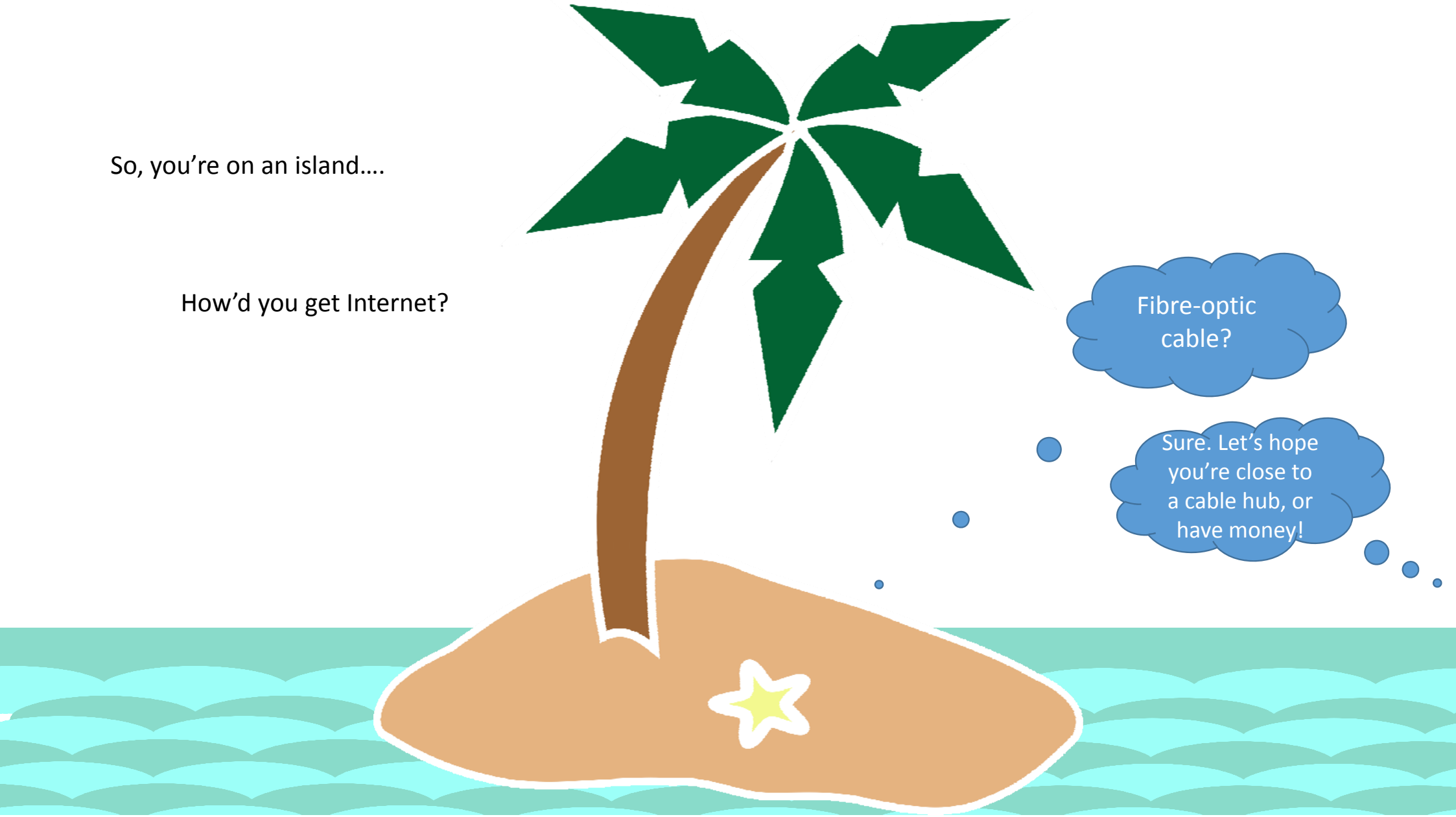
So, you're on an island….

How'd you get Internet?



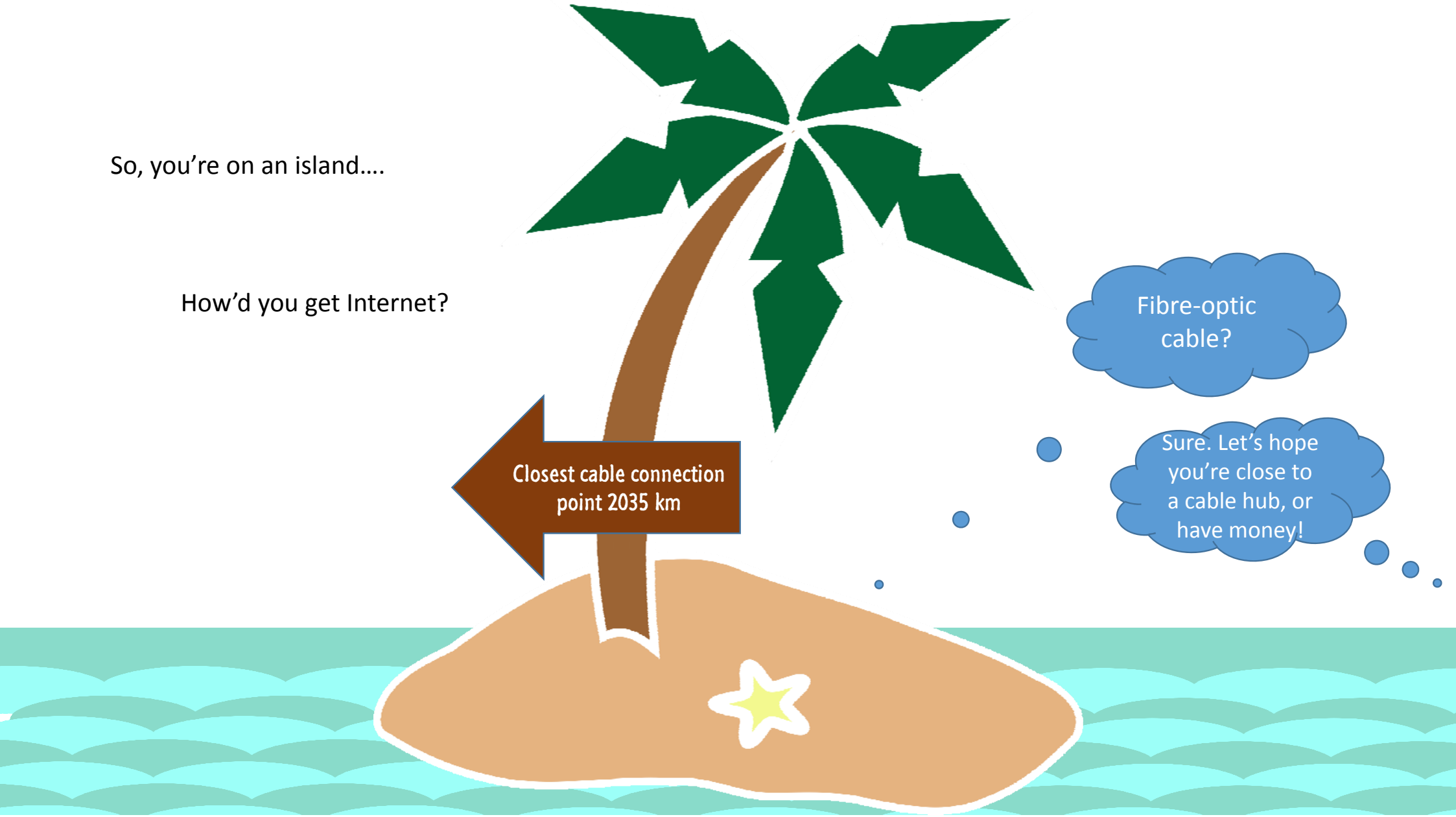
So, you're on an island….

How'd you get Internet?



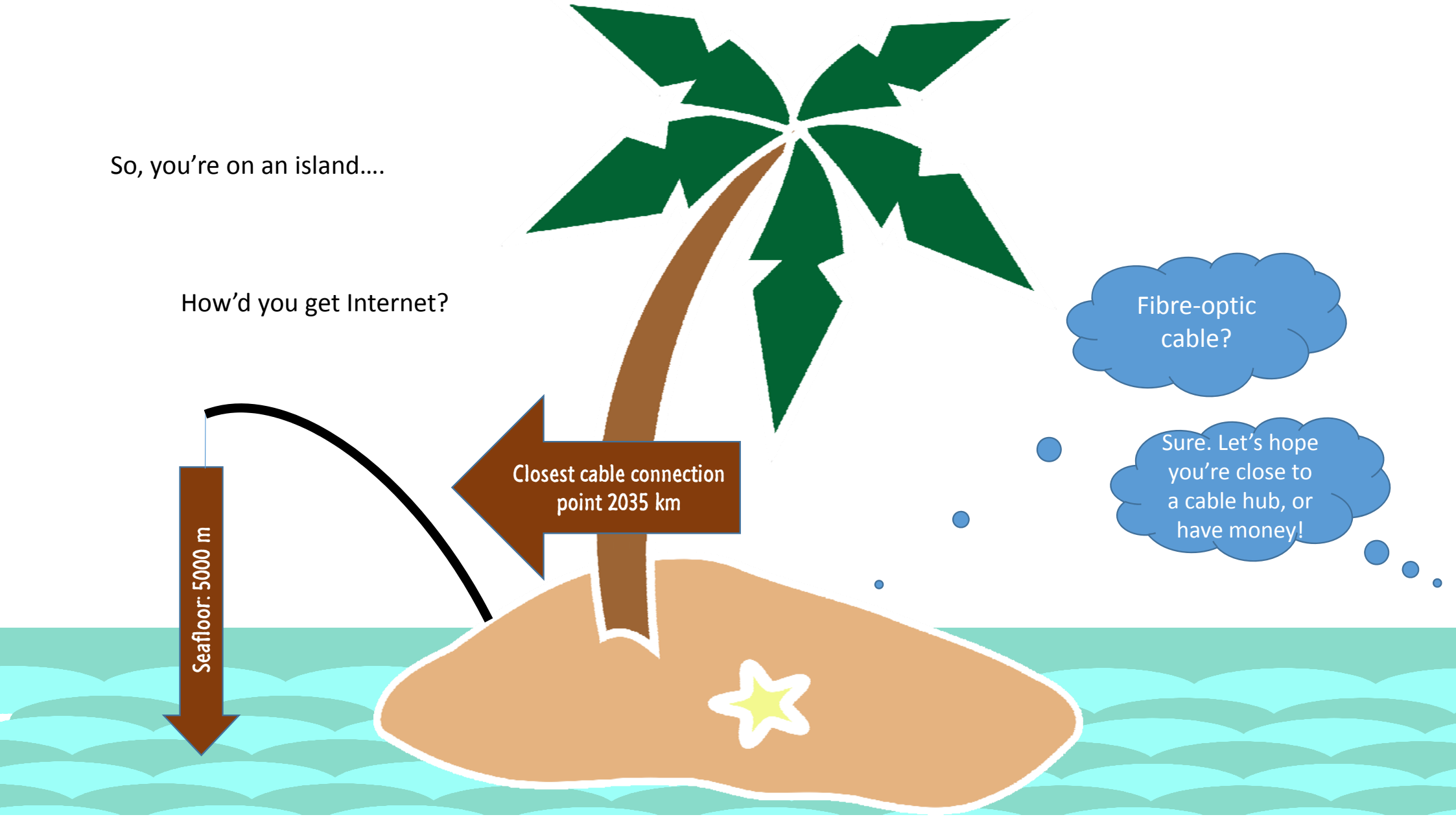
So, you're on an island….

How'd you get Internet?



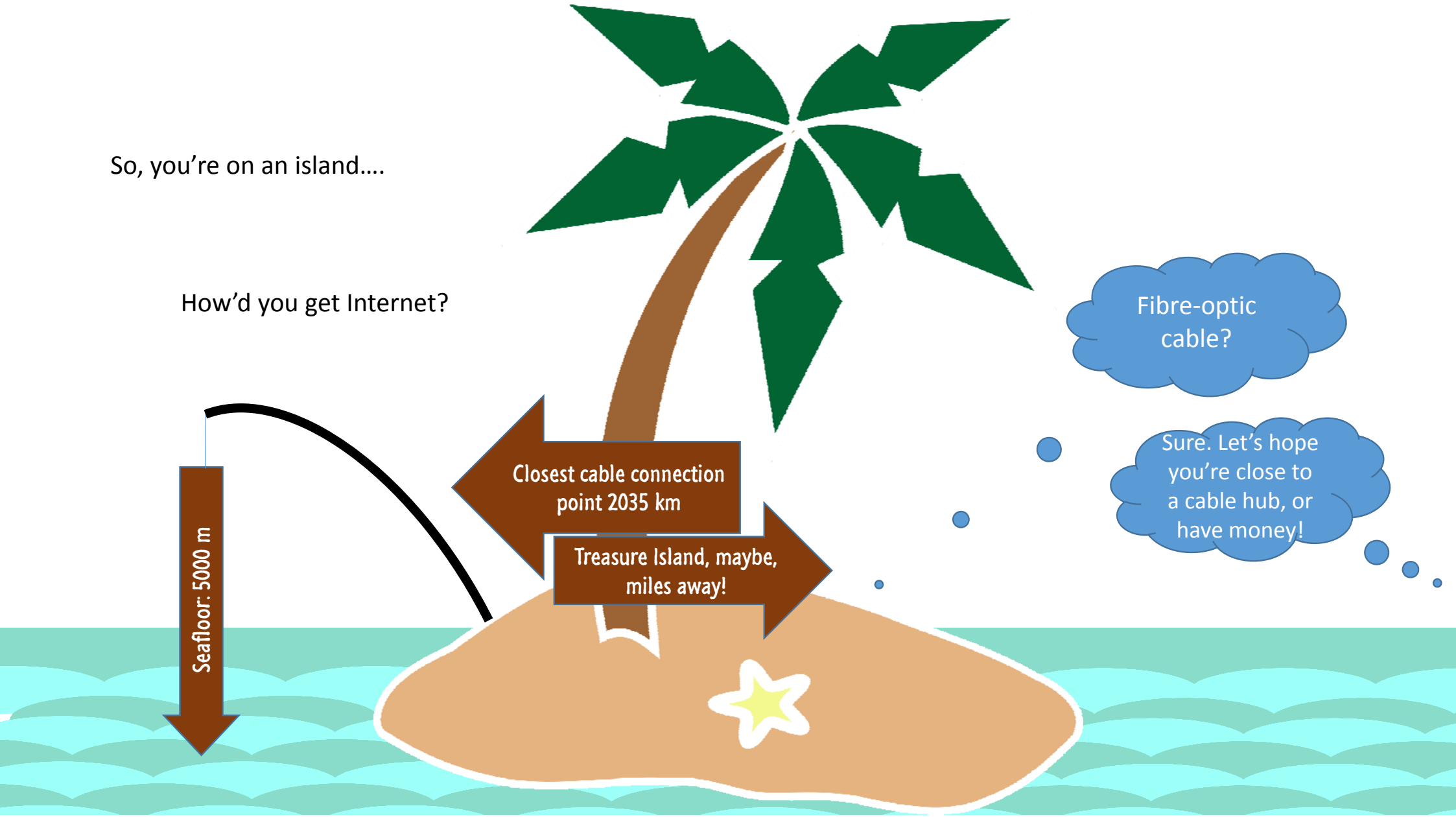
So, you're on an island….

How'd you get Internet?



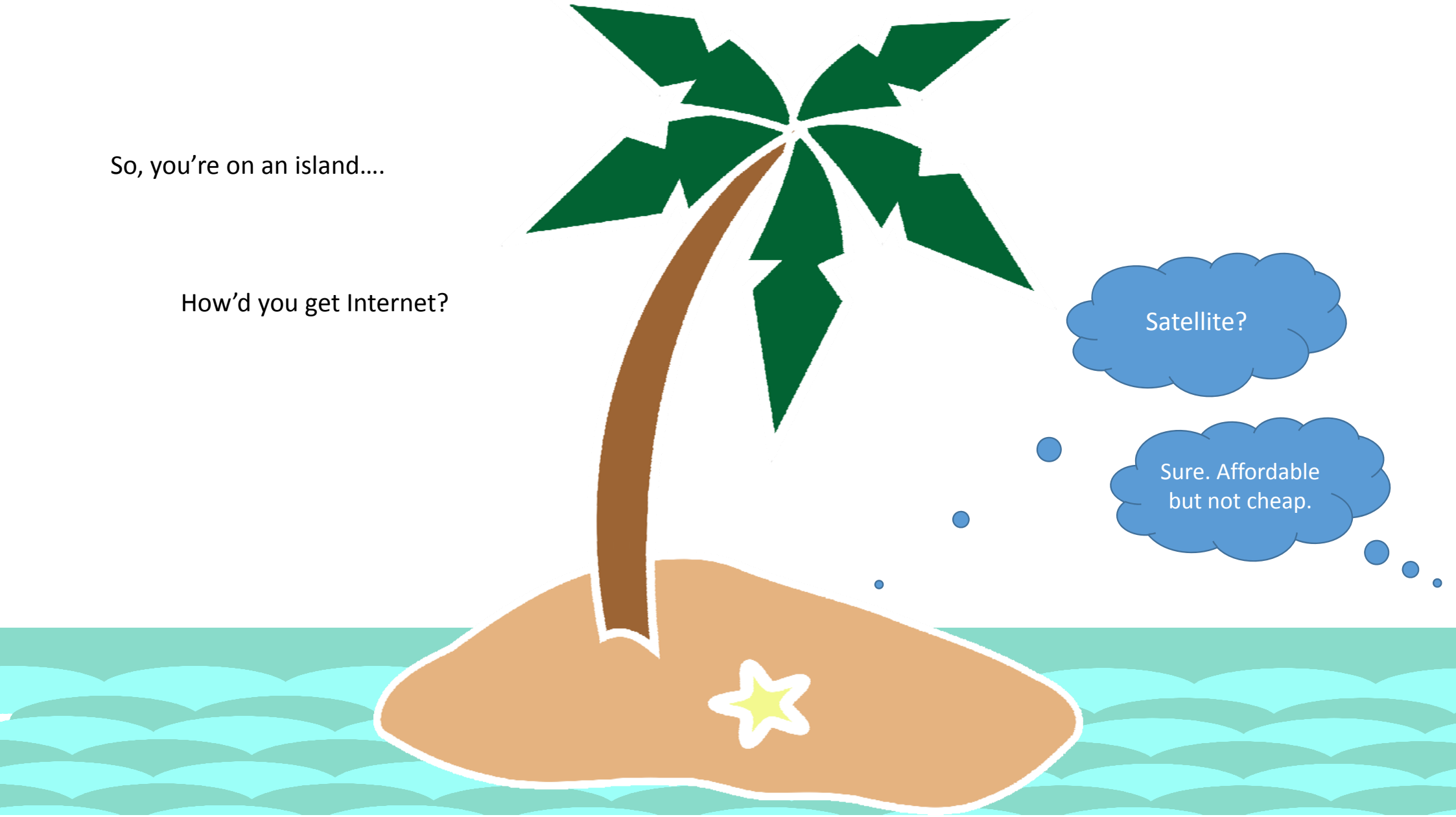
So, you're on an island….

How'd you get Internet?



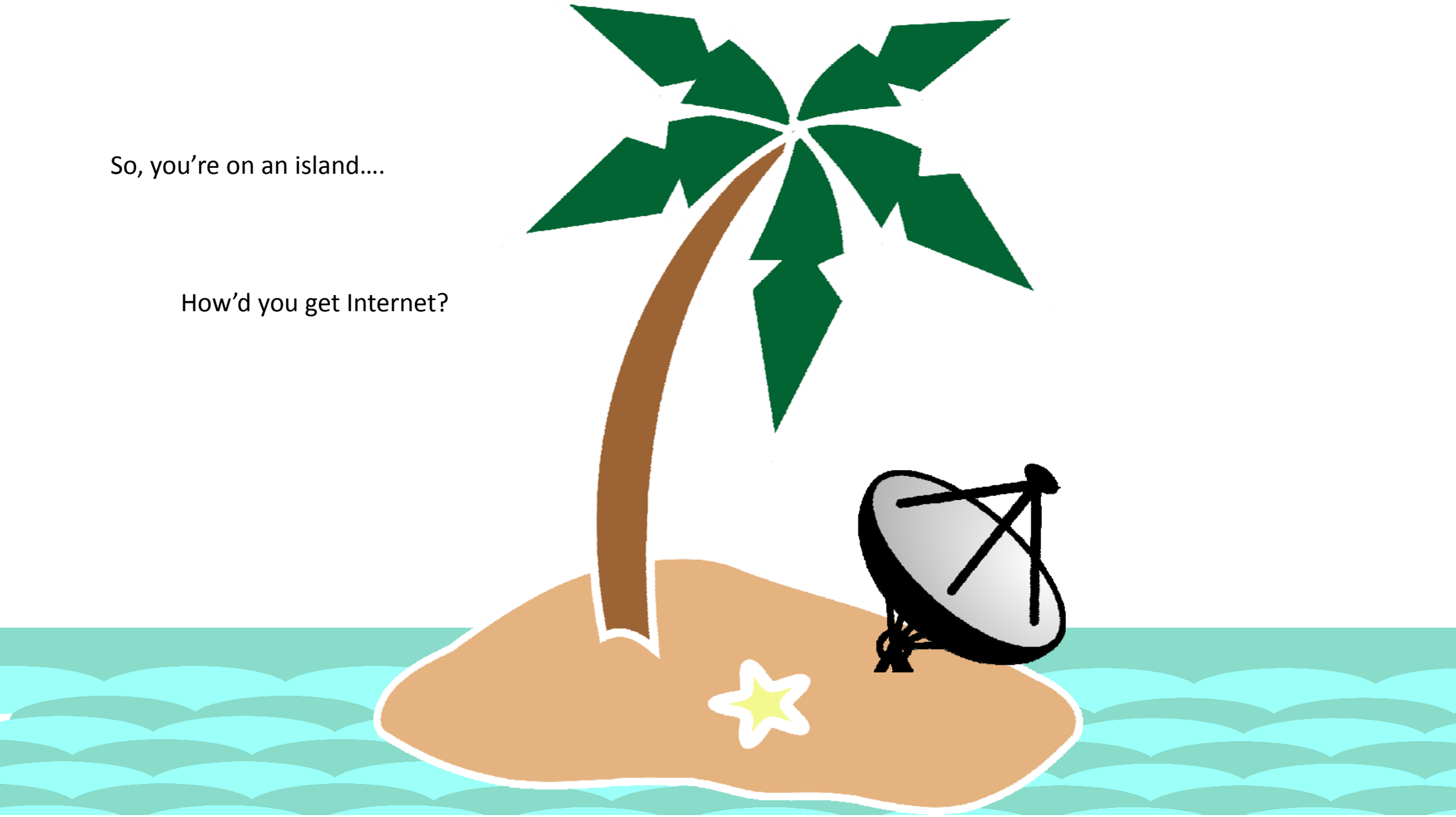
So, you're on an island….

How'd you get Internet?



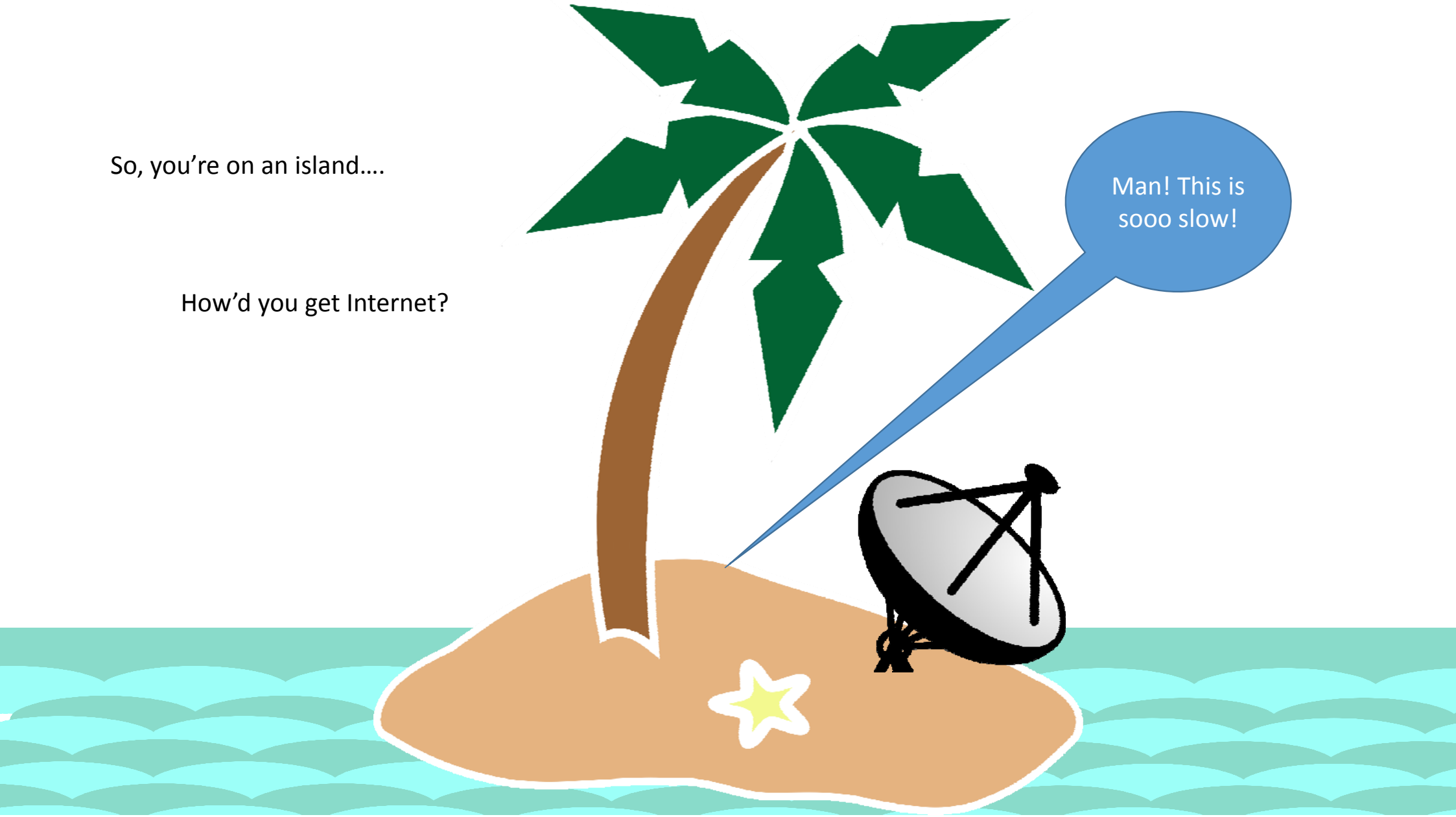
So, you're on an island….

How'd you get Internet?



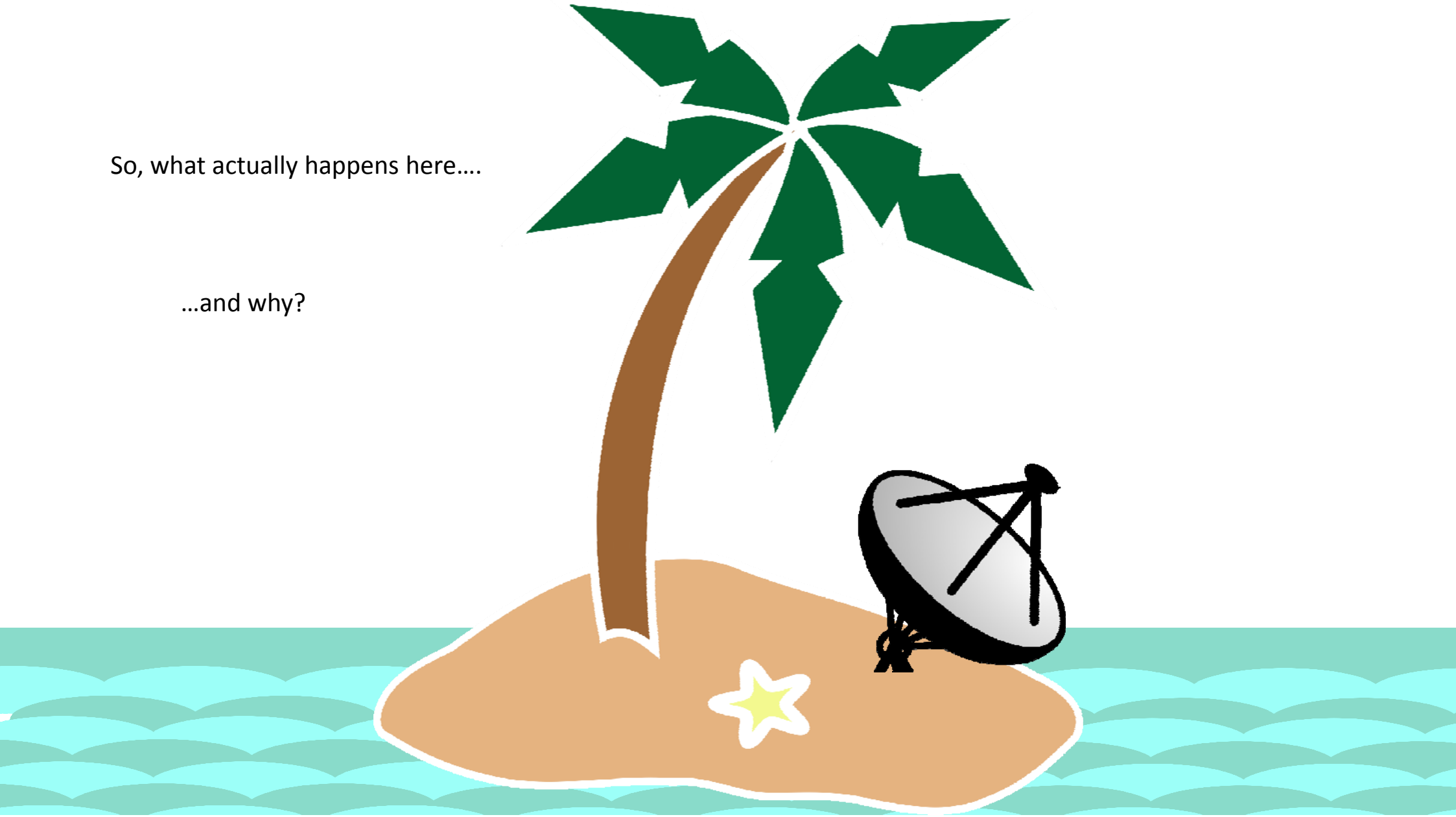
So, you're on an island….

How'd you get Internet?



So, what actually happens here….

…and why?



Satellite links are not born equal

Geostationary (~500 ms one-way latency)…

Satellite links are not born equal

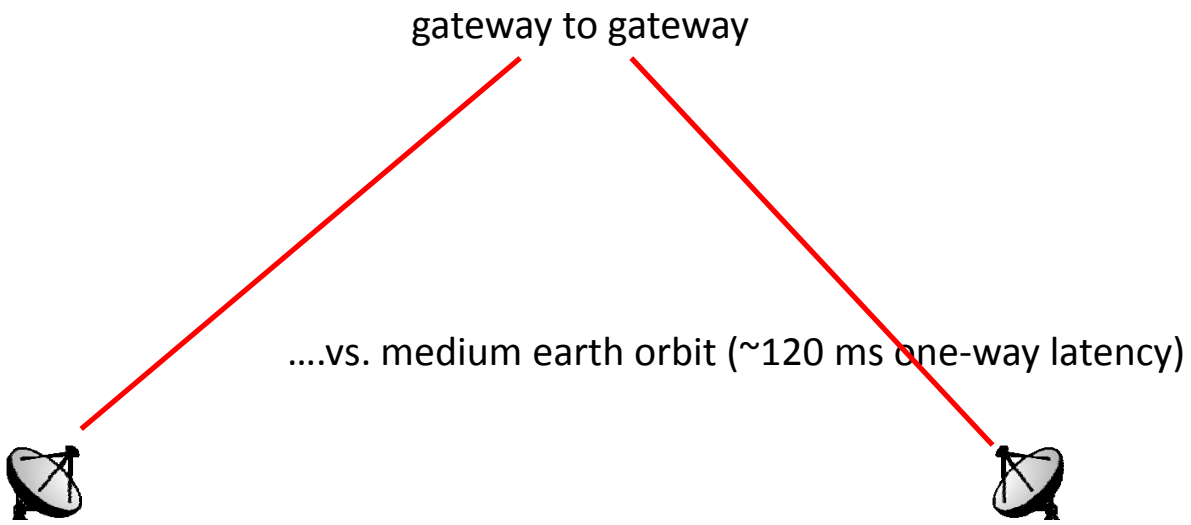
Geostationary (~500 ms one-way latency)…

Satellite links are not born equal

….vs. medium earth orbit (~120 ms one-way latency)

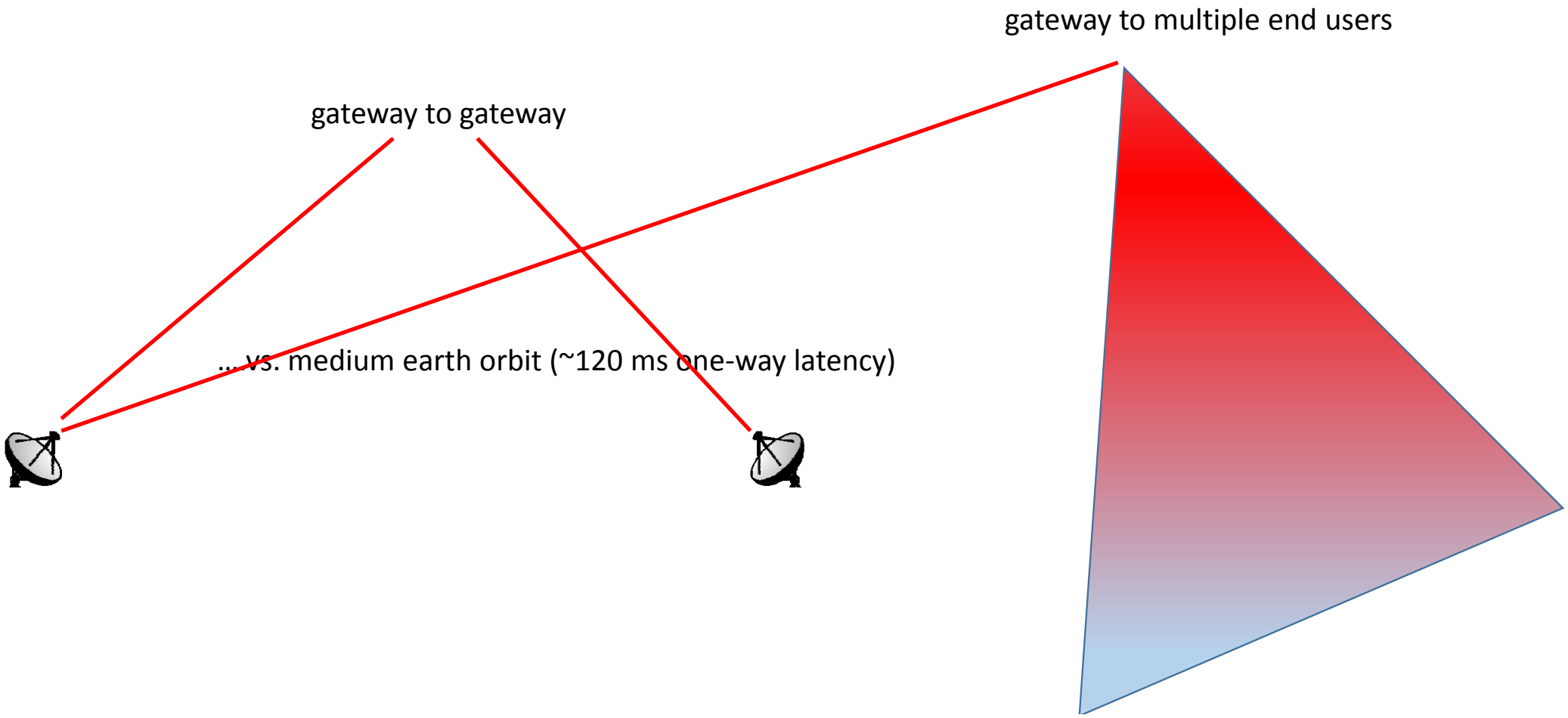
Geostationary (~500 ms one-way latency)…

Satellite links are not born equal



Geostationary (~500 ms one-way latency)…

Satellite links are not born equal



Geostationary (~500 ms one-way latency)…

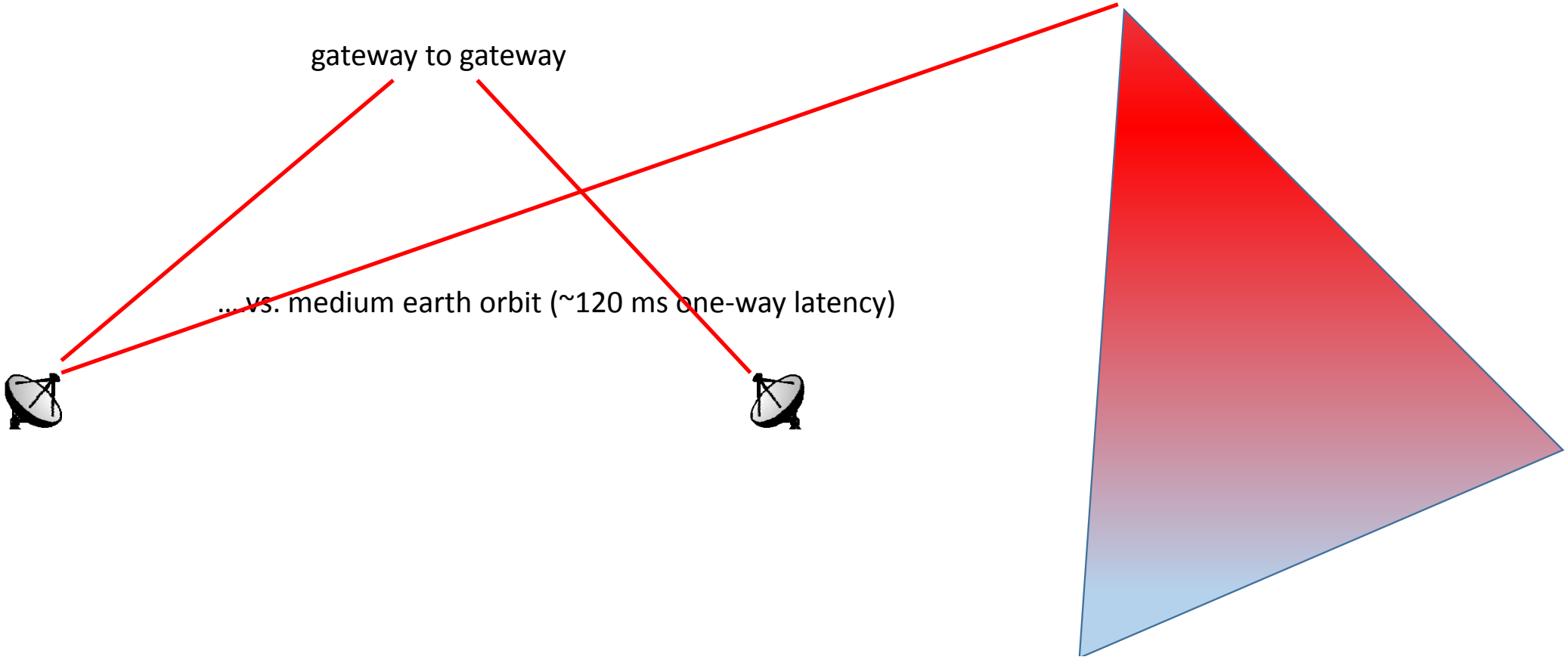
Satellite links are not born equal

narrowband

gateway to multiple end users

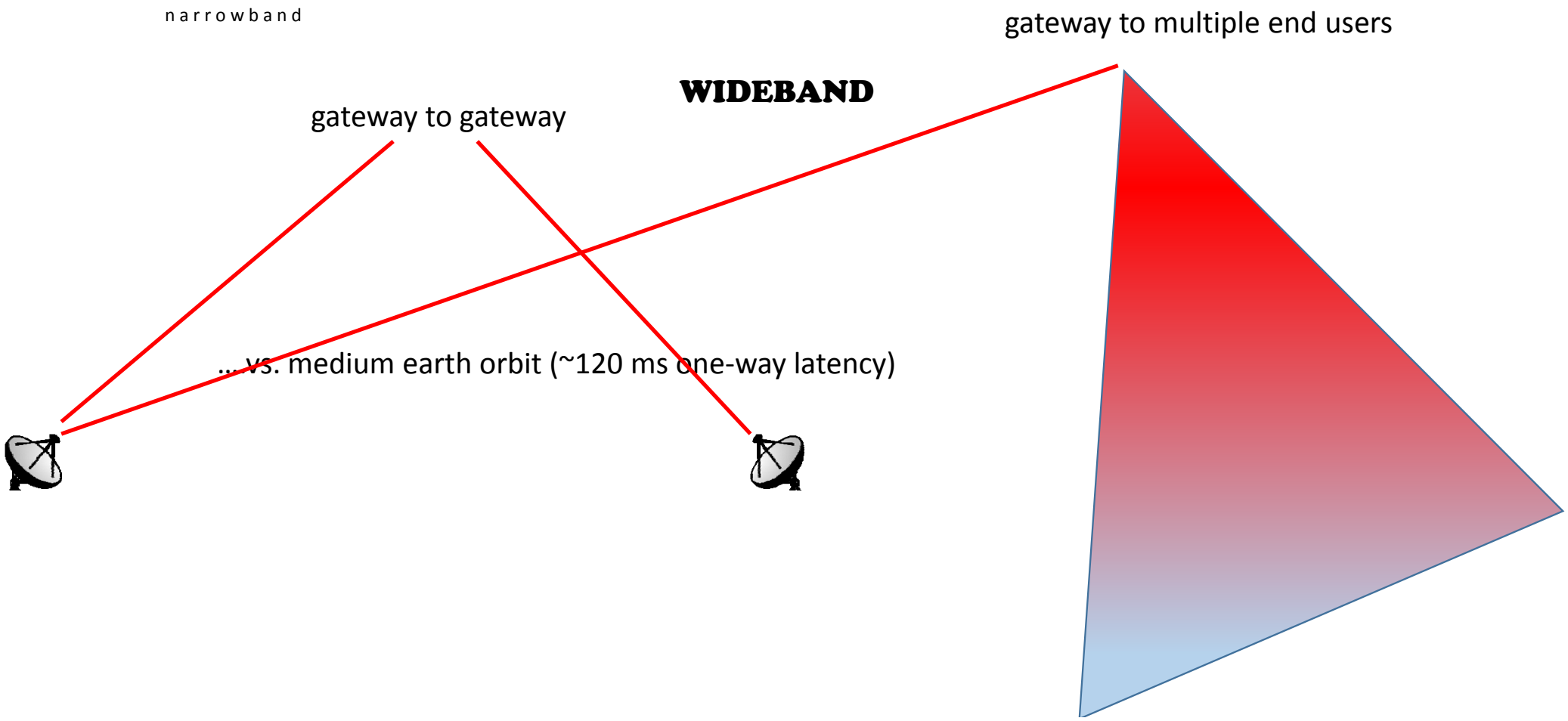
gateway to gateway

…vs. medium earth orbit (~120 ms one-way latency)



Geostationary (~500 ms one-way latency)…

Satellite links are not born equal



Geostationary (~500 ms one-way latency)…

Satellite links are not born equal

narrowband

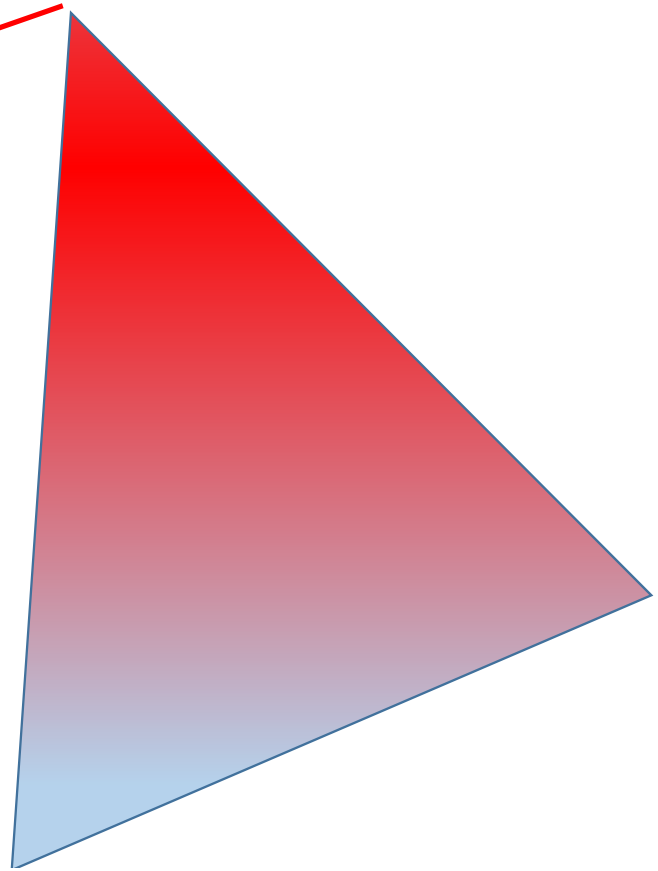
gateway to multiple end users

WIDEBAND

gateway to gateway

…vs. medium earth orbit (~120 ms one-way latency)

bandwidth shared by just a few flows



Geostationary (~500 ms one-way latency)…

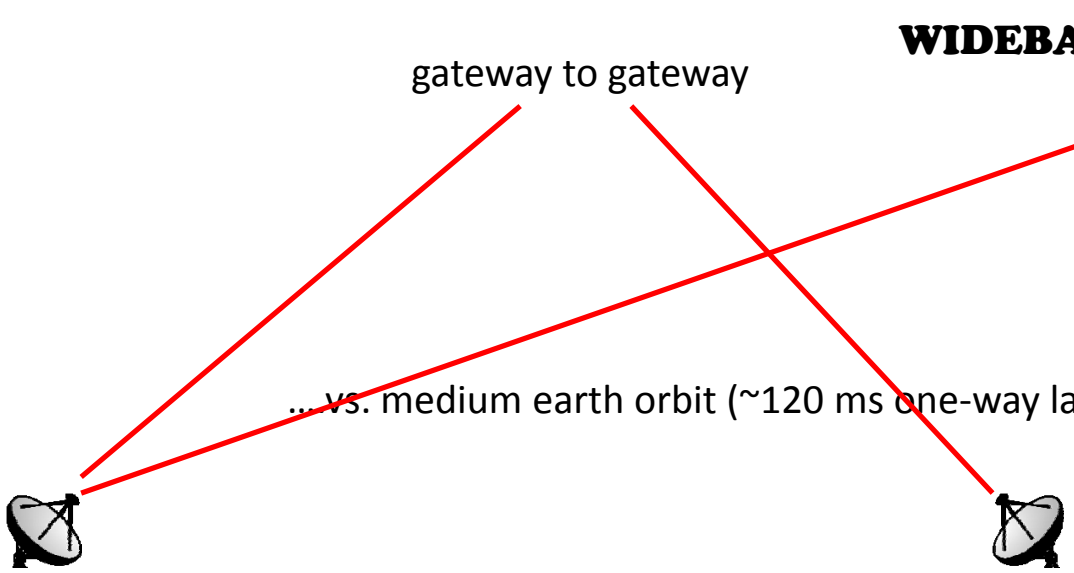
Satellite links are not born equal

narrowband

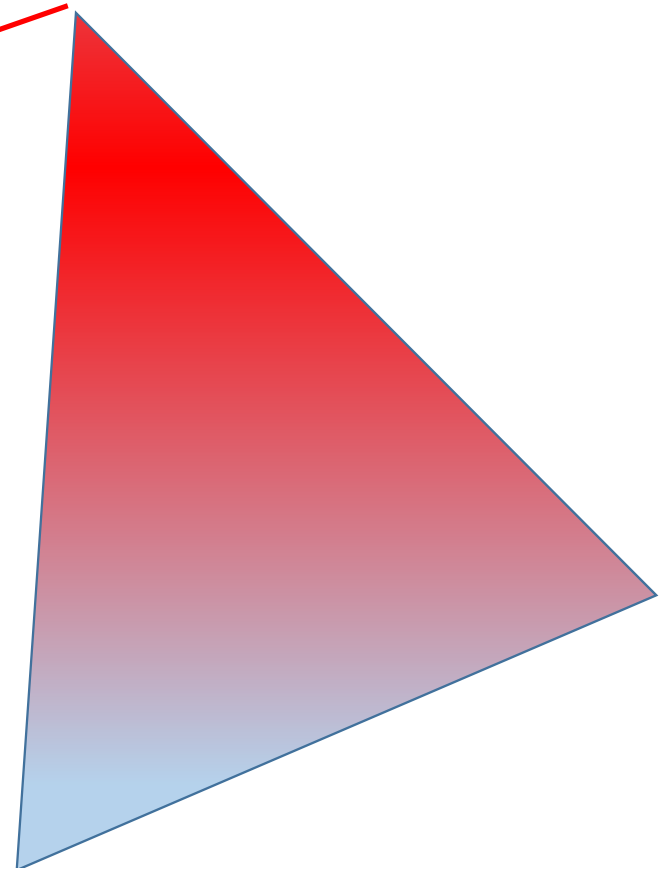
gateway to multiple end users

WIDEBAND

gateway to gateway



…vs. medium earth orbit (~120 ms one-way latency)



bandwidth shared by just a few flows

bandwidth shared by a large number of flows

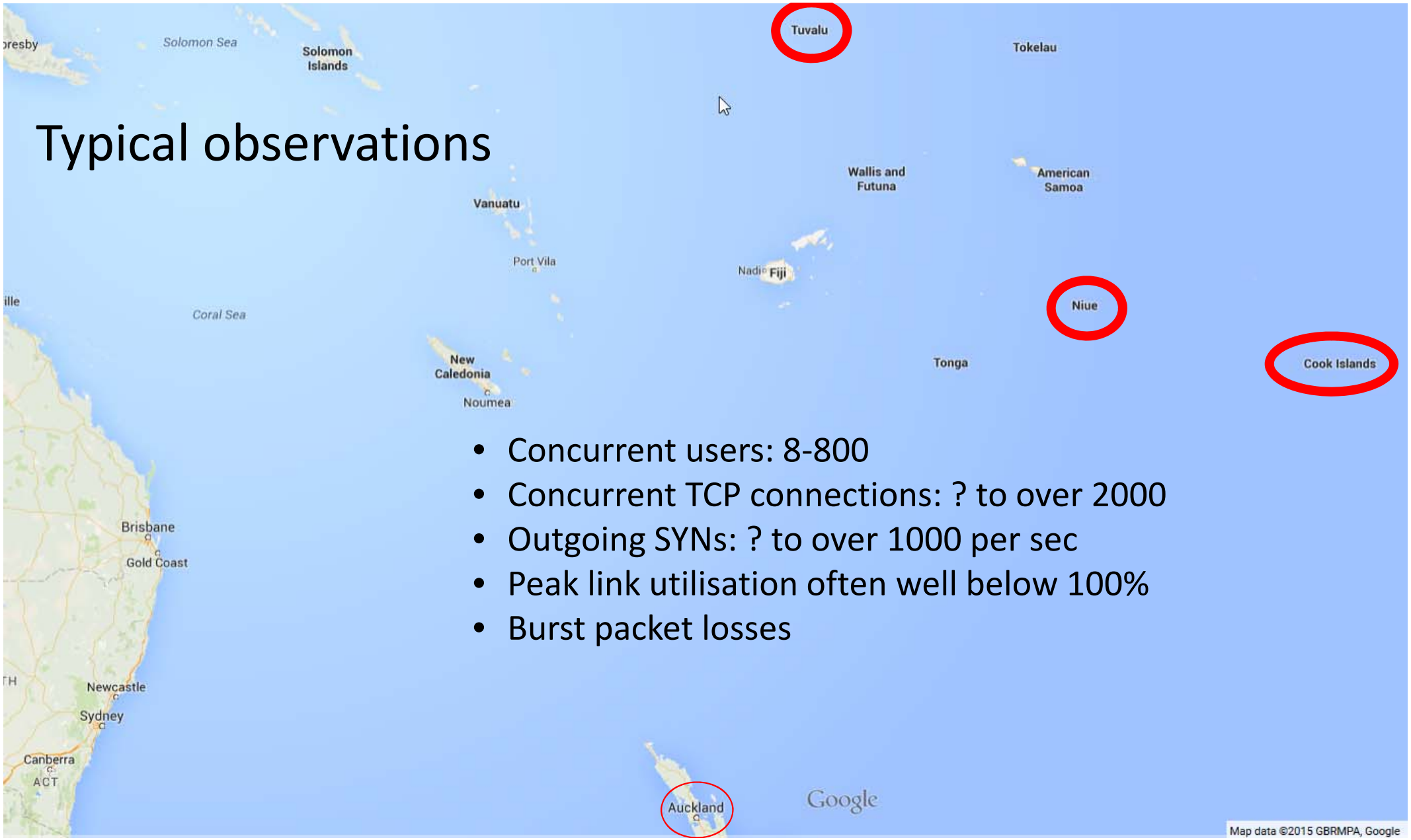
Our partners in the Pacific

- Bluesky Cook Islands
- Internet Niue
- Tuvalu Telecom
- PICISOC

Typical setups

- GEO link from off-island gateway to gateway on island
- MEO link to Rarotonga and Aitutaki in the Cook Islands
- “Pipe model”: All IP traffic to/from the island shares the same channel
- Provisioned bandwidths (inbound traffic): 8-332 Mbps

Typical observations



- Concurrent users: 8-800
- Concurrent TCP connections: ? to over 2000
- Outgoing SYNs: ? to over 1000 per sec
- Peak link utilisation often well below 100%
- Burst packet losses

Question time!

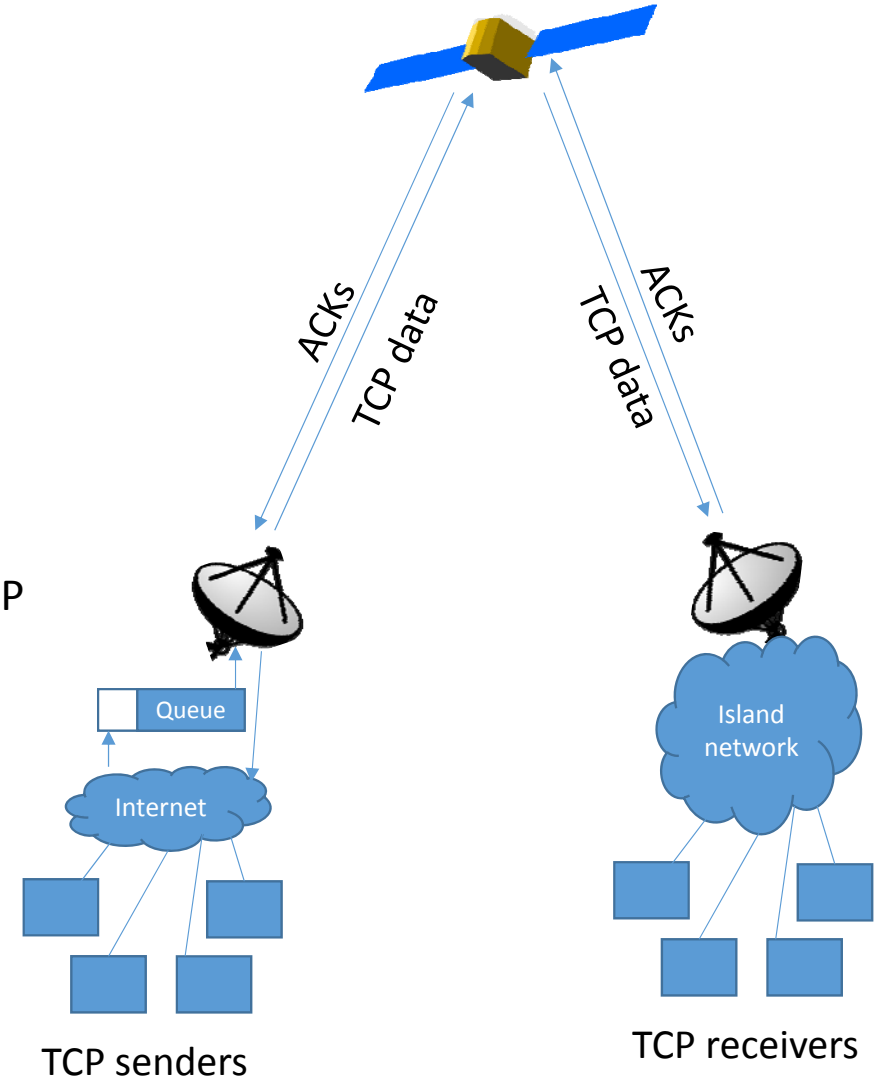
- What's the first thing that comes to mind when you hear the term "packet loss"?

Question time!

- What's the first thing that comes to mind when you hear the term "packet loss"?
  - "Noise", "fading", "bit errors", "symbol errors"?
    - you're probably an electronic engineer by training
  - "Congestion" or "queue drop"
    - you're probably a networker
- Note that we're dealing with both packet loss and link underutilisation here!
- Networkers will recognise this as a sign of…

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 delays: >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*

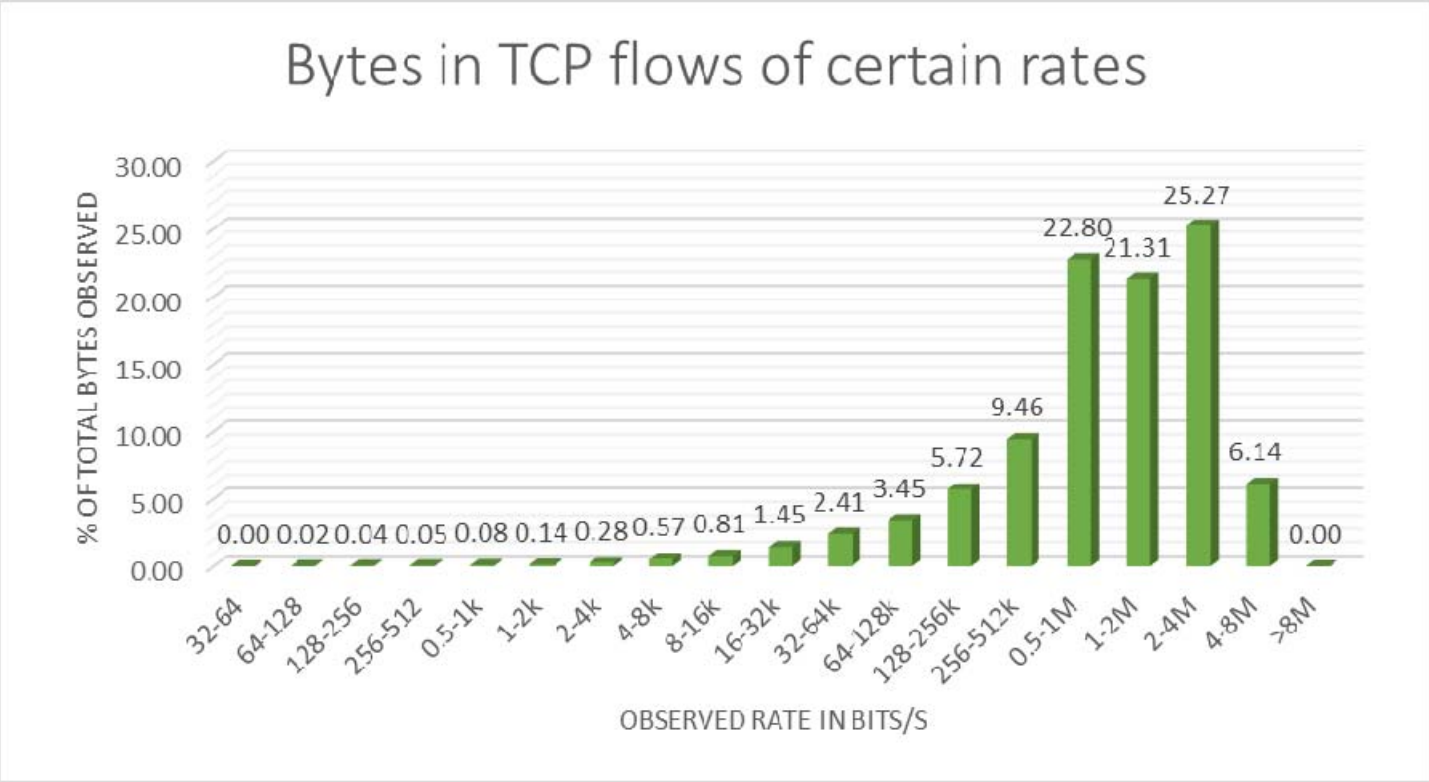
3. ACKs from dropped packets become overdue. Senders throttle back. Packet arrival at queue slows to a trickle. Queue drains.

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

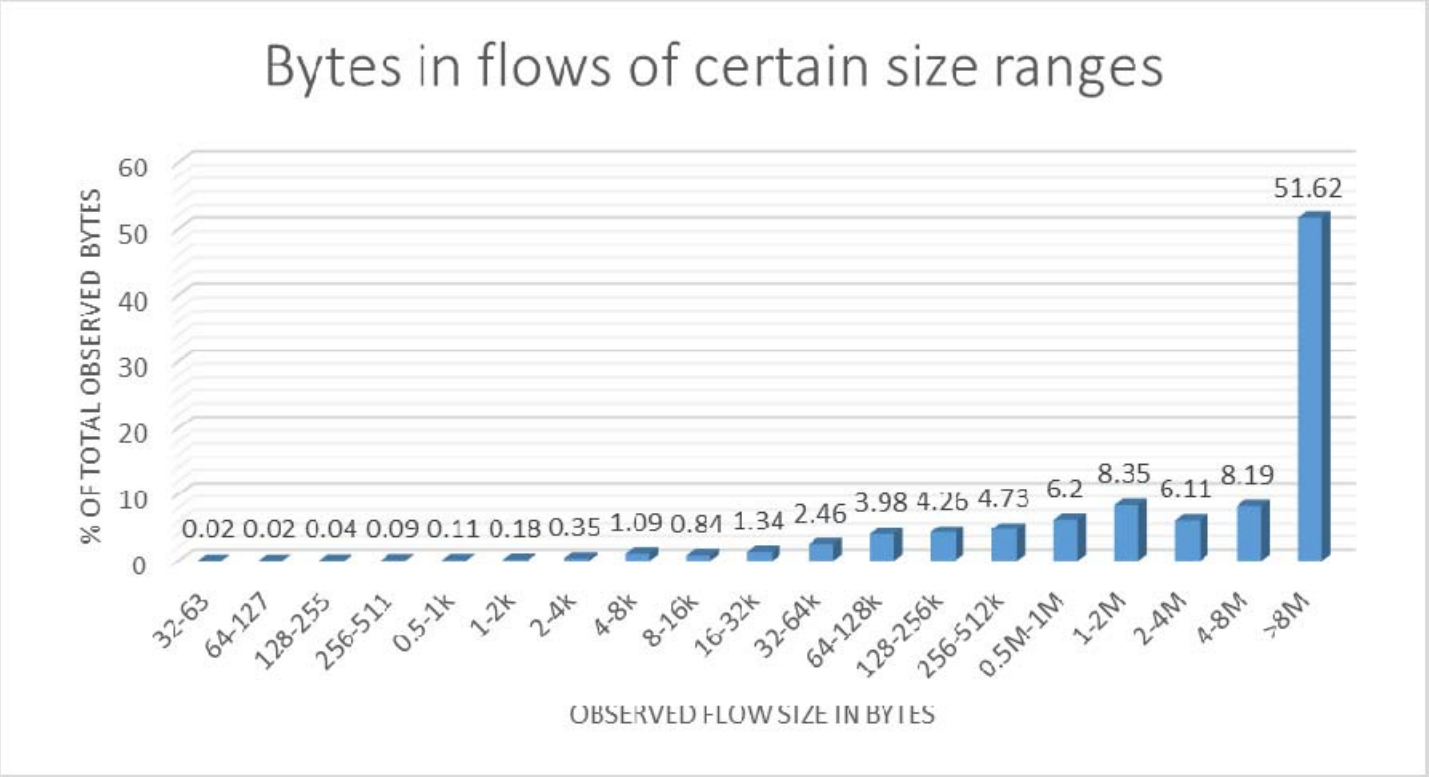
Effect of queue oscillation on TCP flows

- This is at peak time on a connection that's not seeing full capacity utilization
- No TCP flows above 8 Mbps
- Capacity allows *much* faster flows
- >90% of data bytes are in TCP flows whose average rate is <4 Mbps
- But: ISP's don't necessarily see that
- Why?



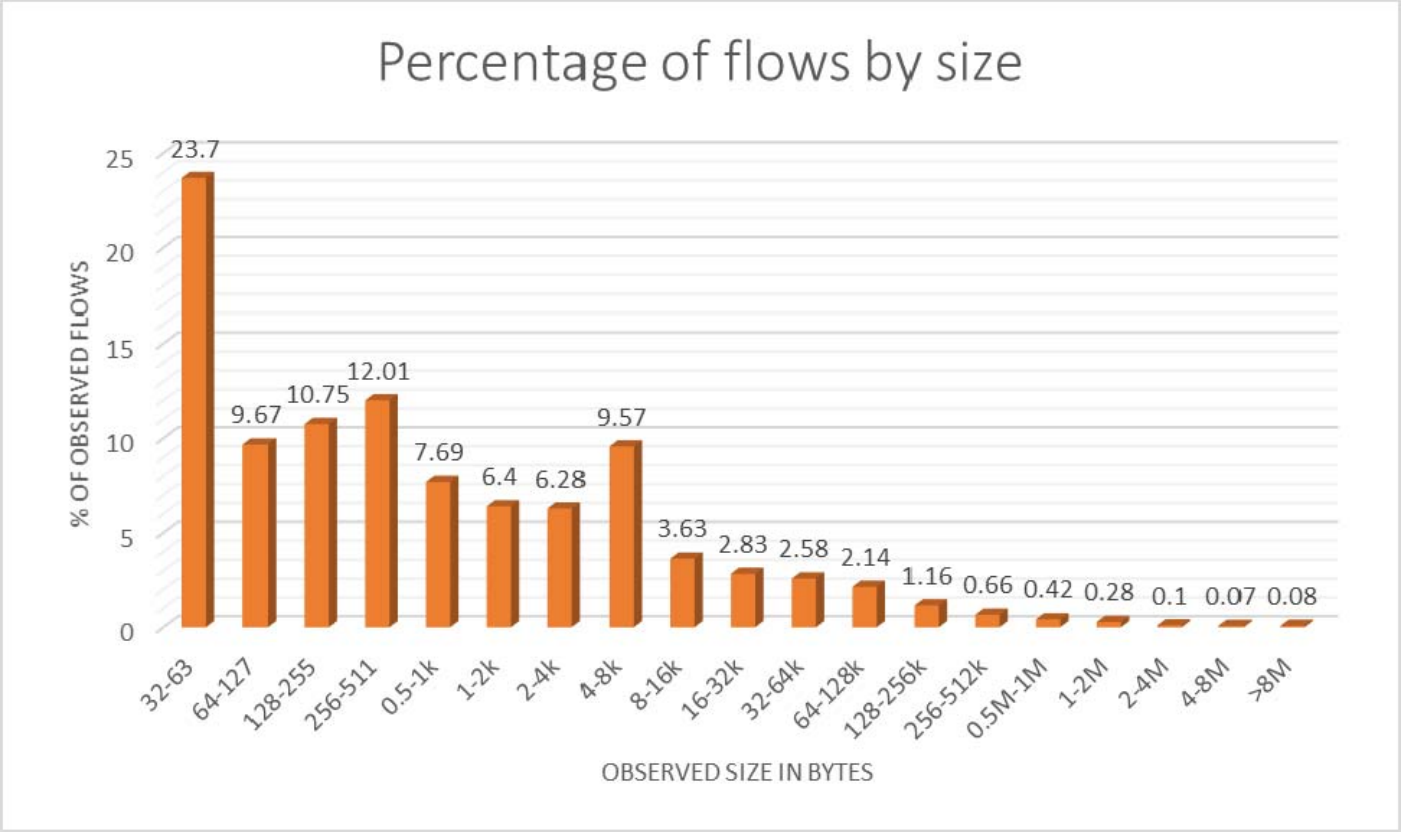
Effect of queue oscillation on TCP flows

- Over half of the bytes are in the largest flows
- Since most bytes experience low data rates…
- But again:
  ISP's don't necessarily see that
- Why?



Why ISP's don't see Queue Oscillation

- Most flows are really short!
- Typical testing:
  - Ping: short flows
  - Loading web sites: short flows (most elements are small)
- So: User's can surf and get their mail
  - If something appears slow: users blame server, not ISP)
  - Users don't complain to ISP!
- Short flows don't experience queue oscillation: user experience dominated by RTT delay, not rate



What's the practical effect of queue oscillation?

- Mail headers and e-mail messages load quickly – but large attachments don't

- Web browsing pages with only text or small elements is fast – but download of software and larger documents (e.g., PDFs, movies) isn't

Common misconception

- “The download time depends mainly on the data rate”
- This is true for large downloads only!
- Example: 160 ms RTT, 300 Mbps MEO sat link, assume no server delay
  - 5 kB download averaging a rate of 4 Mbps takes 170 ms – 94% RTT
    - This is what users experience in web browsing
    - Half the data rate takes this up to 180 ms only
  - 10 MB download averaging a rate of 4 Mbps takes 20.16 s - 0.8% RTT
    - If we only achieve half the data rate here, we need to wait over 40 seconds!
- TCP slow start should let larger flows make better use of the capacity
- **But**: This isn’t happening here: shorter flows between 4 and 32kB in size achieve peak data rates around 70% faster than flows of 1MB and more

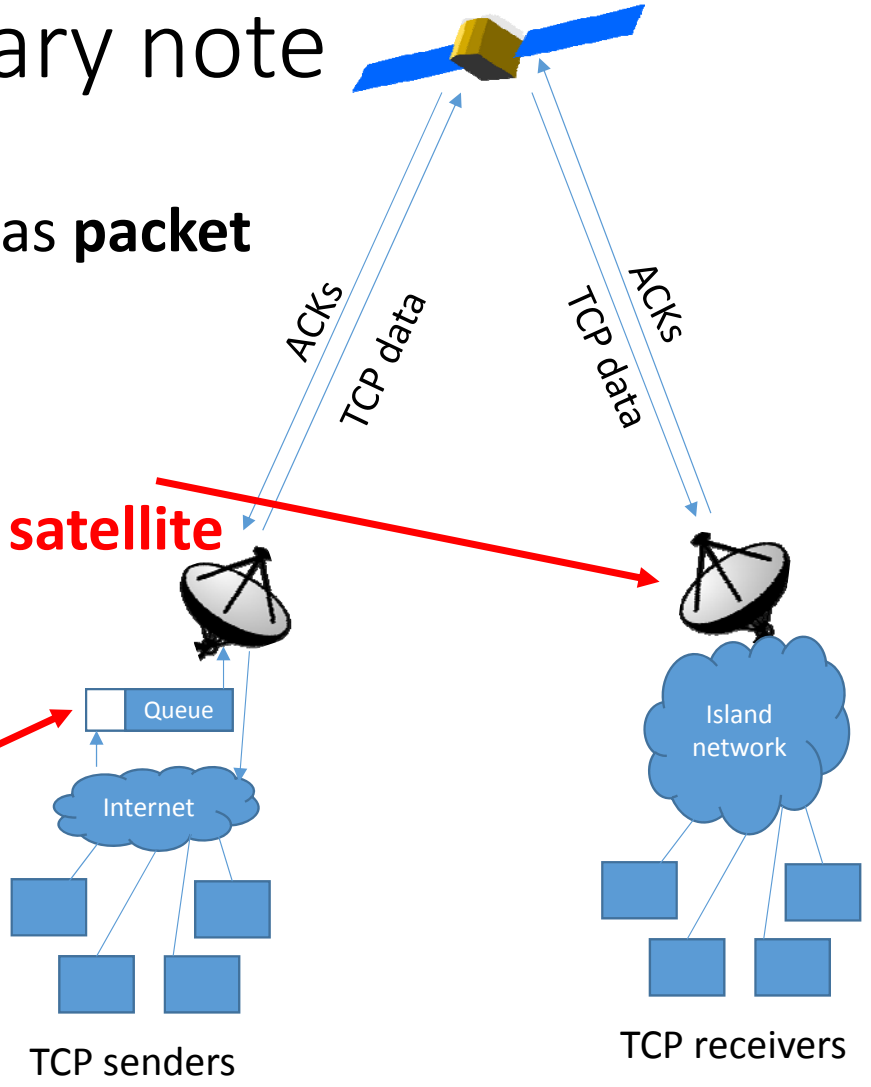
Queue oscillation doesn’t always happen

In order for queue oscillation to occur, we need:

1. A capacity bottleneck
2. Latency
3. A large number of TCP senders
4. Sufficient demand
5. At least some TCP flows that are long enough to be subject to congestion control
   - Flows that complete before the first ACK reaches the sender cannot oscillate
   - For satellite links, this can affect flows of many kB in size

Possible solutions – preliminary note

- Reminder: TCP queue oscillation manifests as **packet loss + link underutilisation**
- Packet loss caused by low/distorted signal and noise/interference occurs **at the satellite receiver. This is not what we see here.**
- Packet loss in the case of queue oscillation occurs **at the input queue**, not at the sat receiver



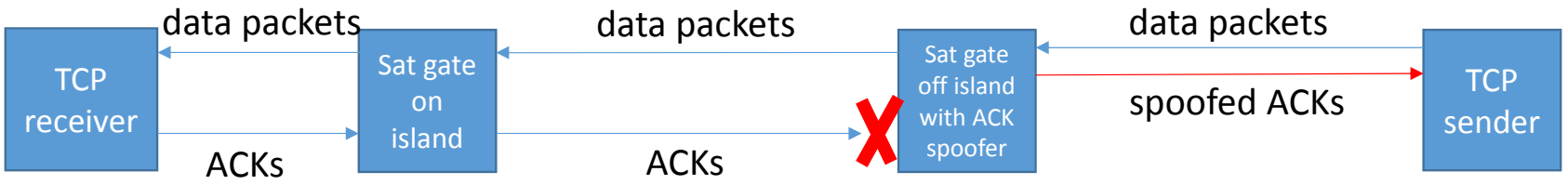
Possible solutions

- Performance Enhancing Proxies (PEPs)
  - Pure ACK spoofers
  - Full connection splitters

- Forward error correction across multiple packets: network coding

ACK-spoofing PEPs

- PEP ACKs incoming data packets to sender
- PEP forwards & caches data packets *without modifying sequence numbers*
- Absorbs ACKs from receiver or retransmits packets after ACK timeout



- PEP interferes with connection, but doesn't fully split it
  - Data packets/ACKs at sender and receiver use the same sequence numbers

Connection-splitting PEPs

- PEP pretends to be the server when dealing with the host that initiates the connection
- PEP terminates the connection and opens a separate connection to the server
- Data between the connections at the PEP travels through a pipe at the application layer

- PEP splits the connection – violates end-to-end principle
  - Connection A and B use different packets and sequence numbers

PEPs

- Have been around for a while but either aren't used in the Pacific, or if used, don't seem to work that well
- Literature indicates that PEPs work well for a small number of parallel connections, but there are few studies looking at hundreds or thousands of parallel connections
- Connection A potentially still sees a bottleneck with long latency
- Main effect is to bring the RTT down (a little?) for the TCP senders
- Can also have connection-splitting PEPs at both sat gates and use TCP variants for long latencies (Hybla, H-TCP etc.) between sat gates. However, these tend to be aimed at long fat pipes, not long narrow ones.

Solution: TCP over network coding (TCP/NC)

- Rethink: Need a way in which we can let the sat gate drop data without causing mayhem in TCP
- Insight: packets consist of bytes, which are just binary numbers. Can multiply / add them.
- Can combine g IP packets p_i byte-wise into a "linear combination packet" 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:

$$c_{1,1}p_1 + c_{1,2}p_2 + c_{1,3}p_3 + \ldots + c_{1,g}p_g = r_1 \quad \rightarrow \quad (c_{1,1}, c_{1,2}, c_{1,3}, \ldots, c_{1,g}, r_1)$$

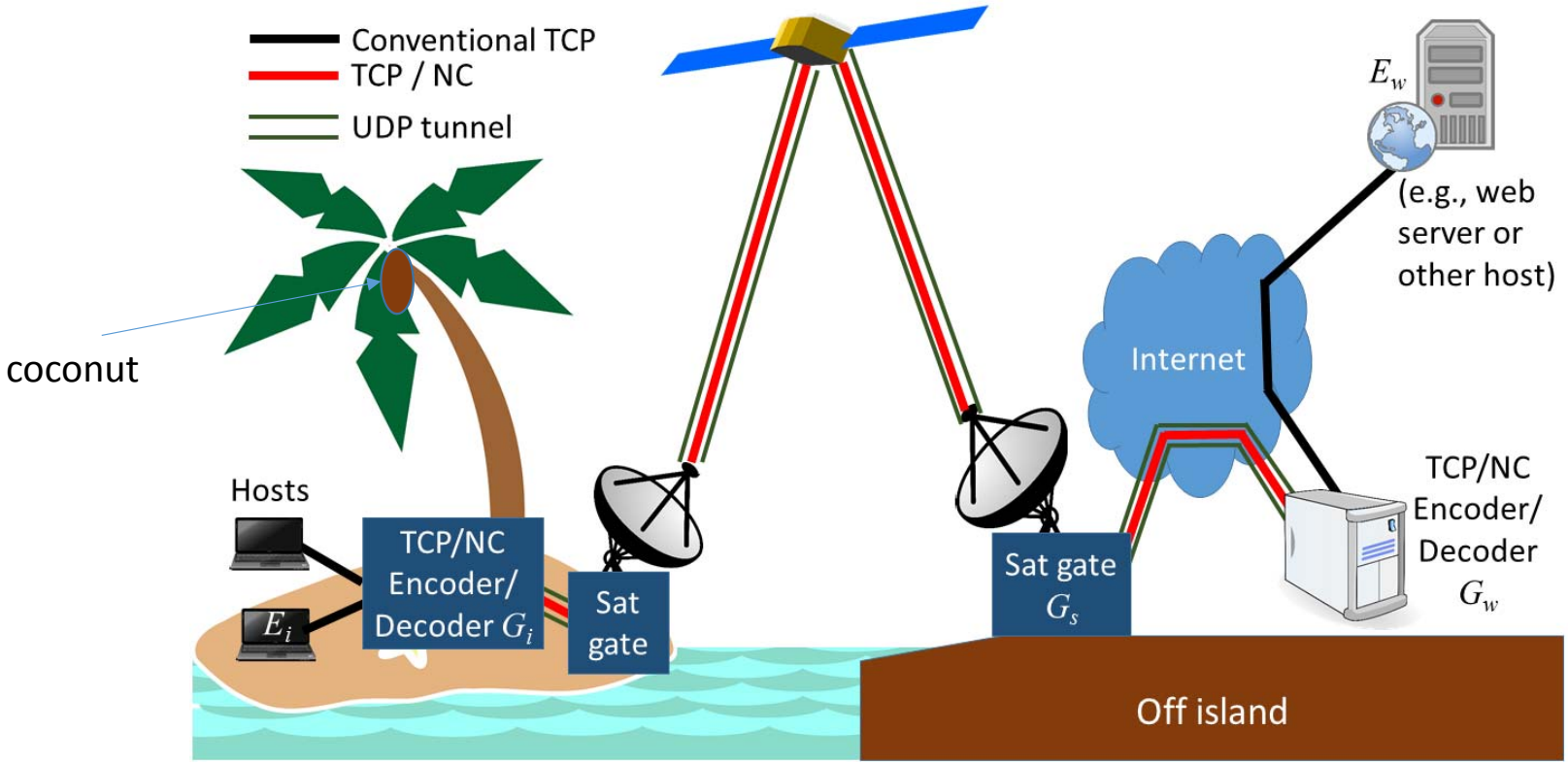
$$c_{2,1}p_1 + c_{2,2}p_2 + c_{2,3}p_3 + \ldots + c_{2,g}p_g = r_2 \quad \rightarrow \quad (c_{2,1}, c_{2,2}, c_{2,3}, \ldots, c_{2,g}, r_2)$$

$$c_{3,1}p_1 + c_{3,2}p_2 + c_{3,2}p_3 + \ldots + c_{3,g}p_g = r_3 \quad \rightarrow \quad (c_{3,1}, c_{3,2}, c_{3,3}, \ldots, c_{3,g}, r_3)$$

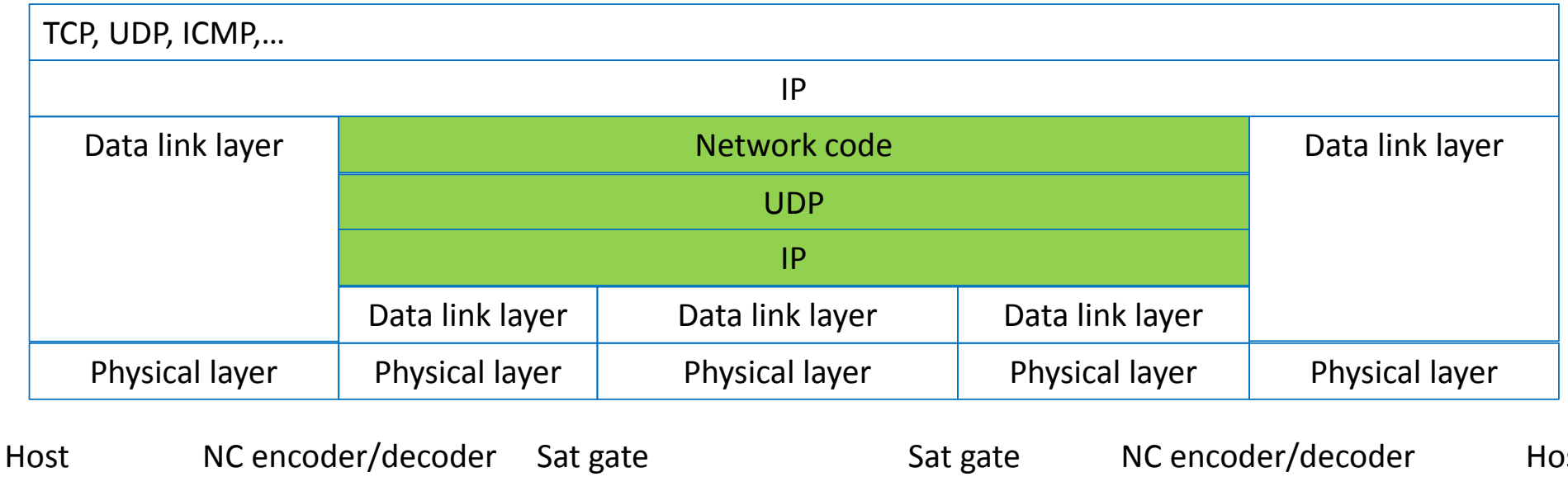
… …

- Generate and send $N > g$ combination packets across the satellite link instead of the g original IP packets.
- This generates an overdetermined system of linear equations whose solution are the original data packets p_i. Receiver solves the system to recover the p_i

TCP/NC in a tunnel setup



TCP/NC tunnel protocol stack

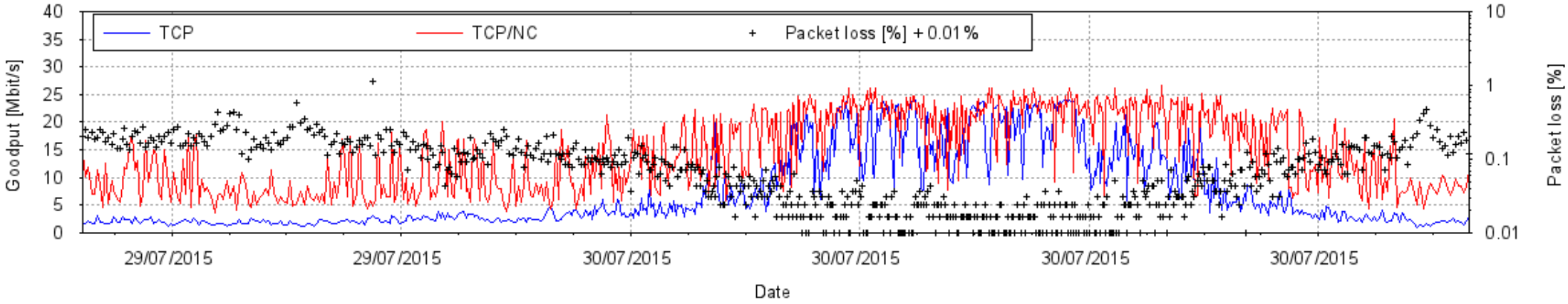


Benefit

- Sat gate can drop up to N-g of the packets without TCP seeing any packet loss
- Amount of data going across the sat link is either almost the same as unencoded, or takes up what would have been spare capacity anyway on links with queue oscillation
- Receiver almost never has to wait for missing data – TCP can communicate faster
- Technical effort (cost) involved is much lower than the equivalent in extra satellite bandwidth or a cable
- End users / content providers don't need to upgrade their computers
- Larger end users can use network coded TCP for their networks without involving their ISP

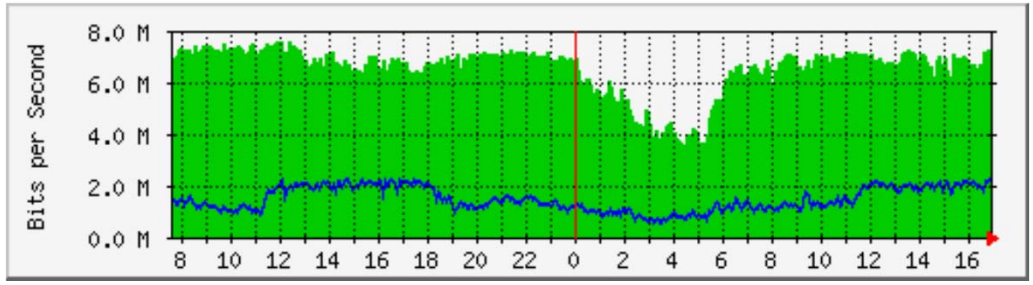
Rarotonga

- O3b satellite connection
- Typical peak time link utilisation around 50%
- TCP/NC encoders/decoders in Avarua and Auckland
- g=30, N variable based on feedback
- TCP/NC running alongside standard TCP
  - Would probably need less overhead if all connections were coded



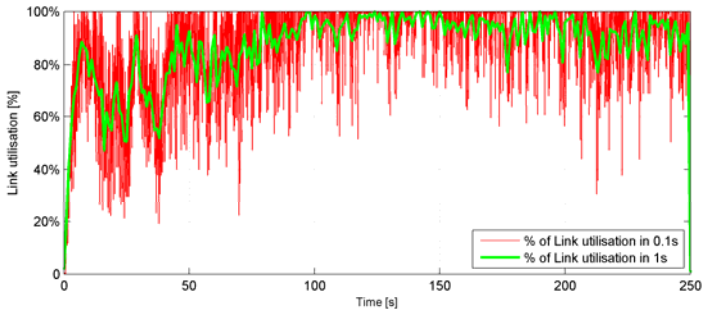
Niue scenario (2015)

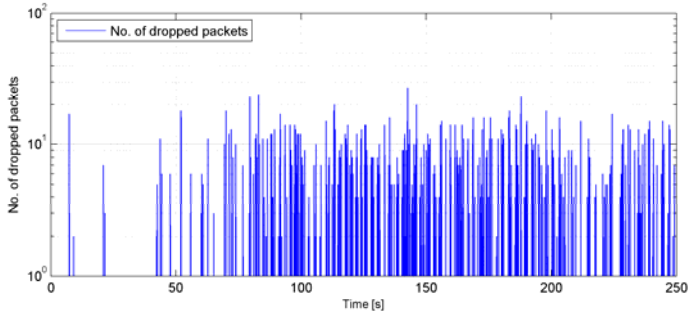
- Connection via geostationary satellite with 8 Mbps downlink
- Very high link utilisation – sustained use of around 7.4 - 7.6 Mbps during the day



Graphic courtesy
Internet Niue

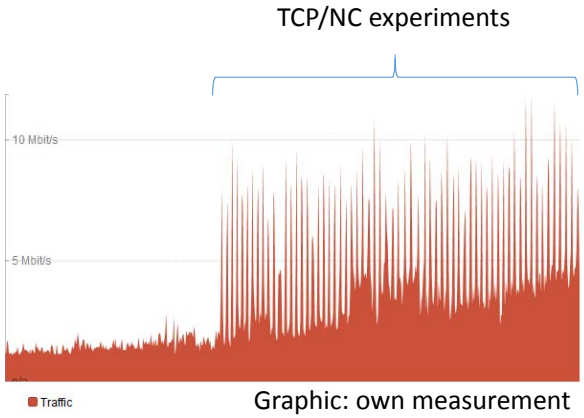
- When sending data to Niue, we see some packet loss
  - Queue overflows but never drains!
- Almost all of the traffic on the link is goodput
  - Can simulate this quite well, too:



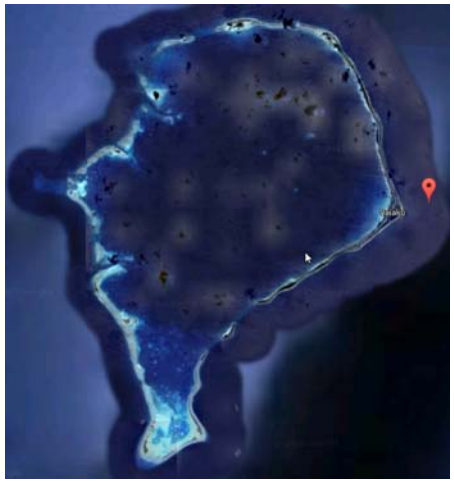


- Non-adaptive TCP/NC buys us goodput of over 2 Mbps on a single connection – **but:** at the expense of other traffic!

Funafuti, Tuvalu

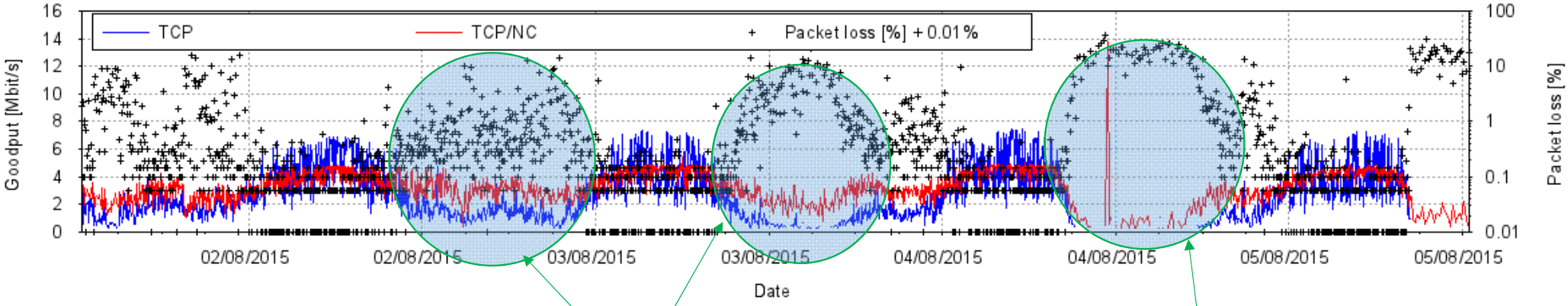


Graphic: own measurement



- GEO downlink (~16 Mbps)
- TCP supported by SilverPeak accelerator
- Very low link utilisation below 25%

- TCP/NC (g=30, adaptive) does not use the SilverPeak



TCP/NC maintains steady goodput with 1-10% packet loss

TCP/NC can complete some downloads even with >10% packet loss

Open questions and progress

- Links shared with legacy TCP cause burst errors that necessitate high NC overhead (high N-g). What would happen if *all* traffic to an island were encoded? Could we get away with less overhead?

- Could network coding work better than PEPs?

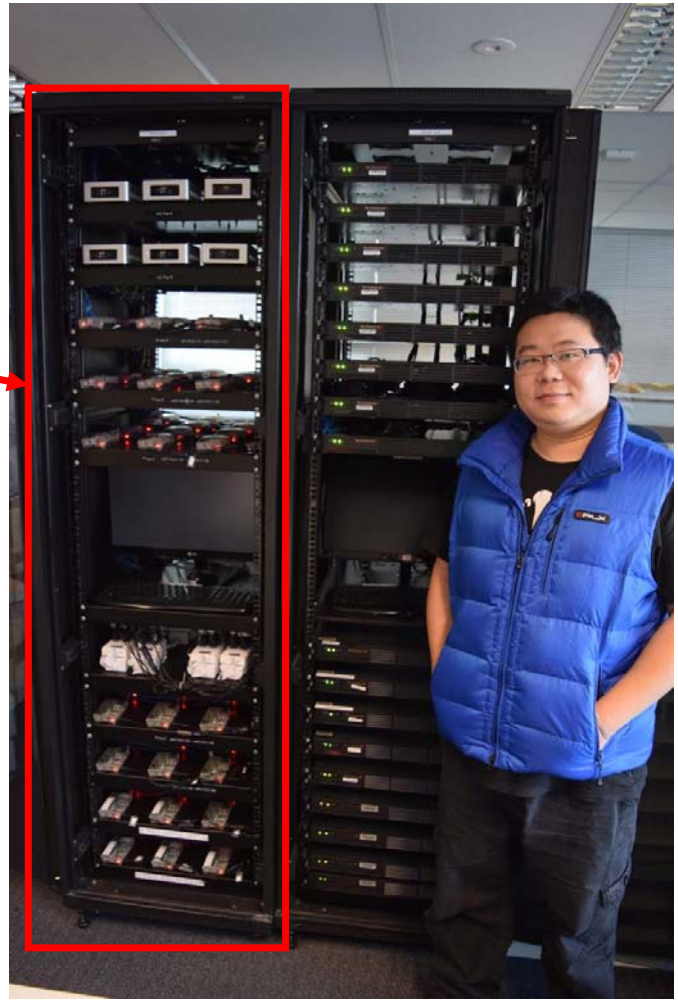
- Current work: Simulating satellite connections with and without TCP/NC and PEPs

Our satellite link simulator

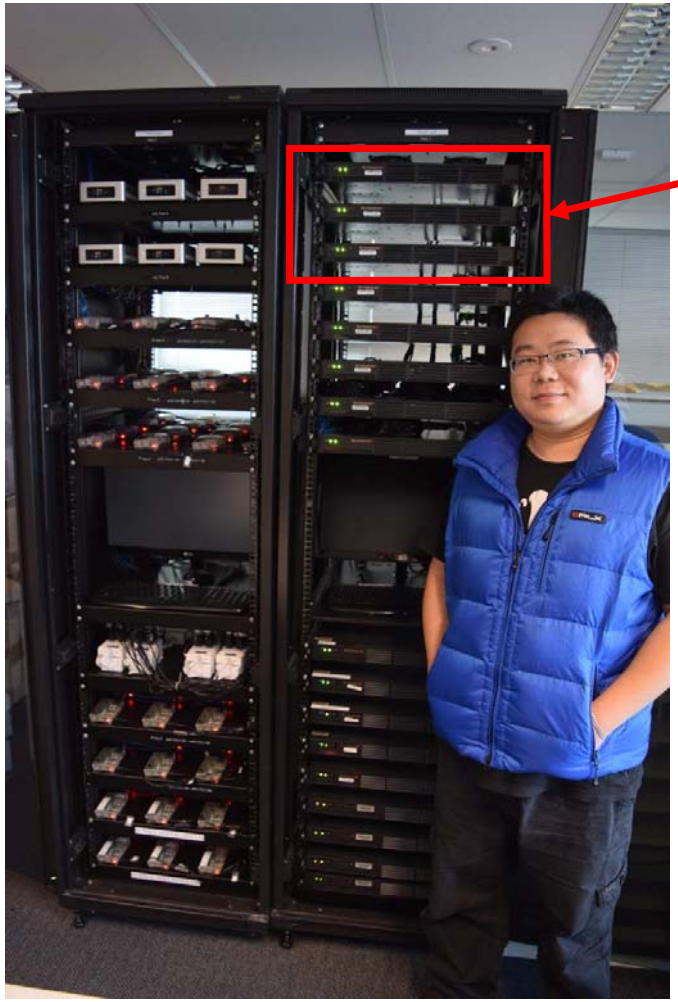
Our satellite link simulator

"Island rack"

- Simulates island clients
- 10 Intel NUC
- 84 Raspberry Pi



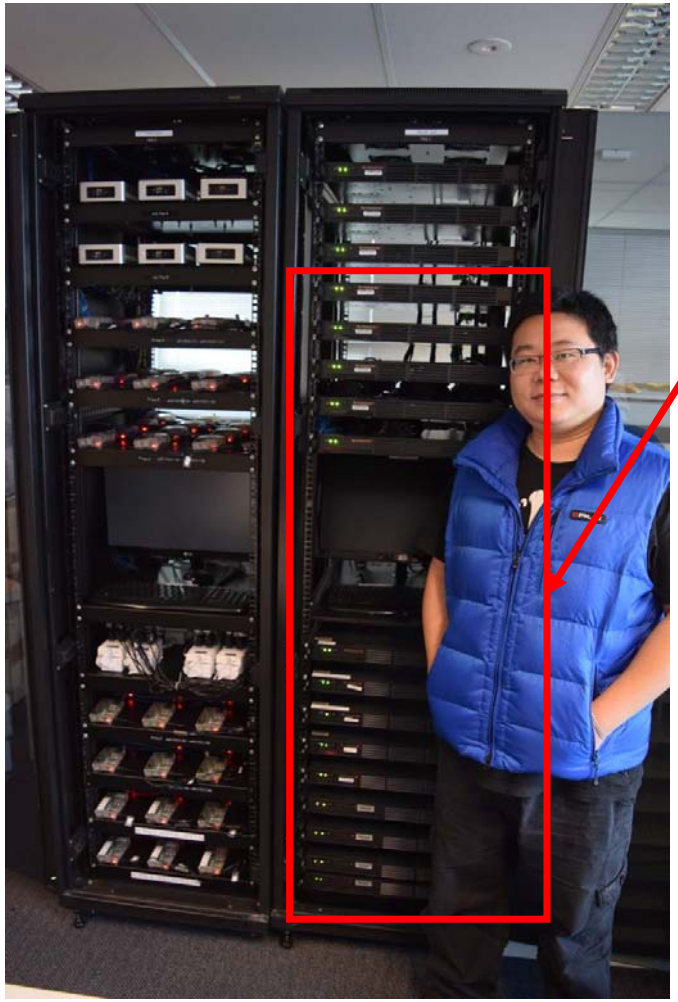
Our satellite link simulator



"World rack"

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

Our satellite link simulator

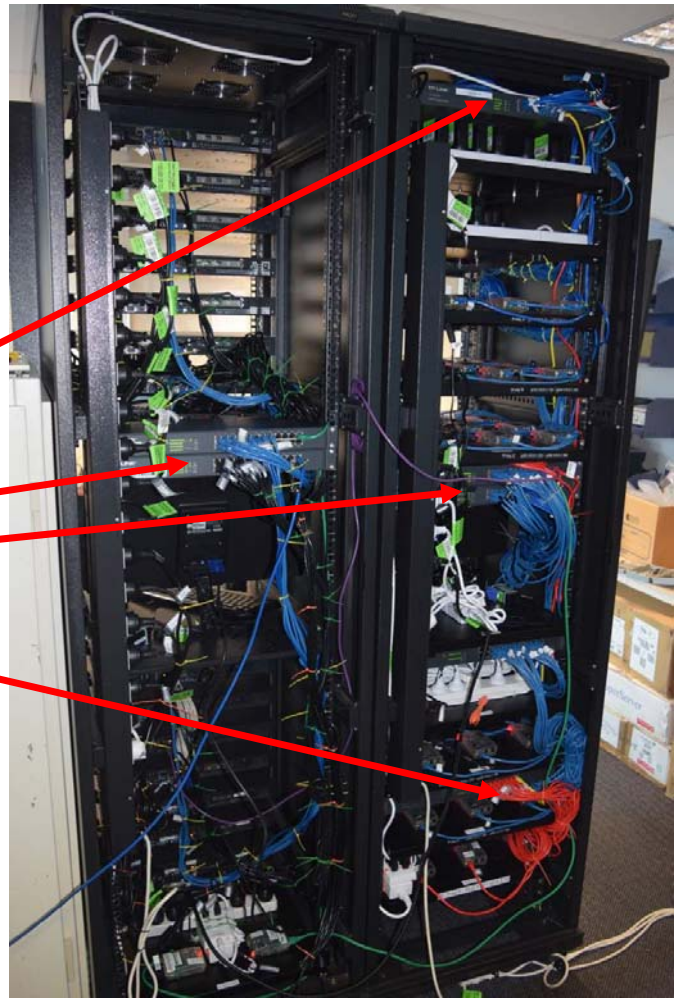


"World rack"

- 14 other Supermicro servers simulate "servers of the world"

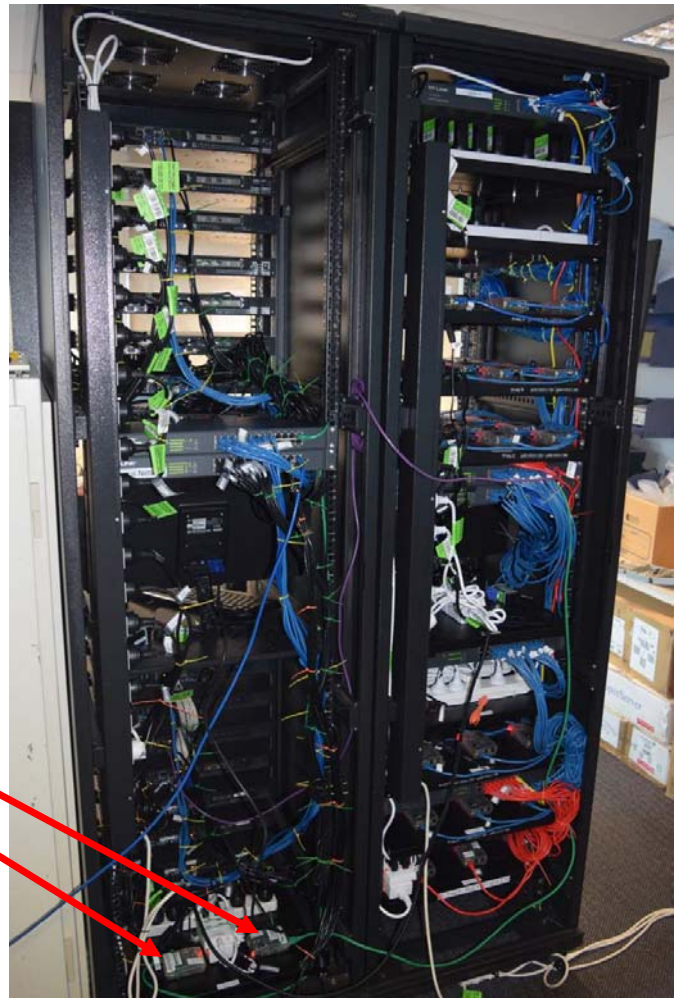
Our satellite link simulator

A number of GBE switches tie the “island” and “world” networks together



Our satellite link simulator

Two dedicated Raspberry Pis act as command and control gateways on each side.



Satellite link simulator client software

- Clients run on the “island side” (Pis and NUCs) of the simulator
- Each client can be configured with multiple “channels”
- Each channel runs at most one TCP client socket at any time
- Each channel operates as follows:
  - Client creates a TCP client socket for the channel
  - Connect to a randomly selected server on the “world side”
  - Socket receives data from the server on the world side and records how many bytes were received
  - Server disconnects the client
  - This process repeats immediately

Satellite link simulator server software

- Servers run on the “world side” (Supermicros) of the simulator
- Servers listen for incoming connections and accept them
- For each incoming connection:
  - The server randomly selects a number of bytes from a real-life flow-size distribution (configurable)
  - Server sends that number of bytes less overhead to the client
  - Server disconnects the client

Conclusions

- TCP/NC
  - Works well under low to moderate queue oscillation
  - Not so much a matter of “How many times faster?” but more of “How much of my bandwidth can I claw back?”
  - Noticeable benefit in Rarotonga and Tuvalu
  - Niue simply didn’t have enough bandwidth deployed – TCP/NC gains there squeezed out standard TCP (current situation is more like Aituitaki)
  - No benefit in low demand conditions (Aitutaki in mid-2015)
  - Transition requires a bit of network planning

- Simulator: Ongoing work
  - Trying to max out a client platform is an unusual task – load balancing between over 20 client channels works well but above that, it gets difficult
  - Will allow us to clarify whether whole-of-island coding brings benefits (we think so)
  - Will allow comparative studies of TCP/NC, PEPs
  - Will allow us to predict when queue oscillation is likely to occur

- Last but not least: Can we beat the throughput of the coconut telegraph?

Thank you!

Project partners, collaborators and funders

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