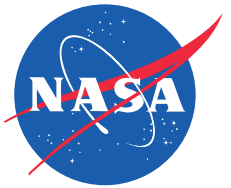


Channel bonding of low-rate links using MPTCP for Airborne Flight Research

Joseph Ishac
Matthew Sargent

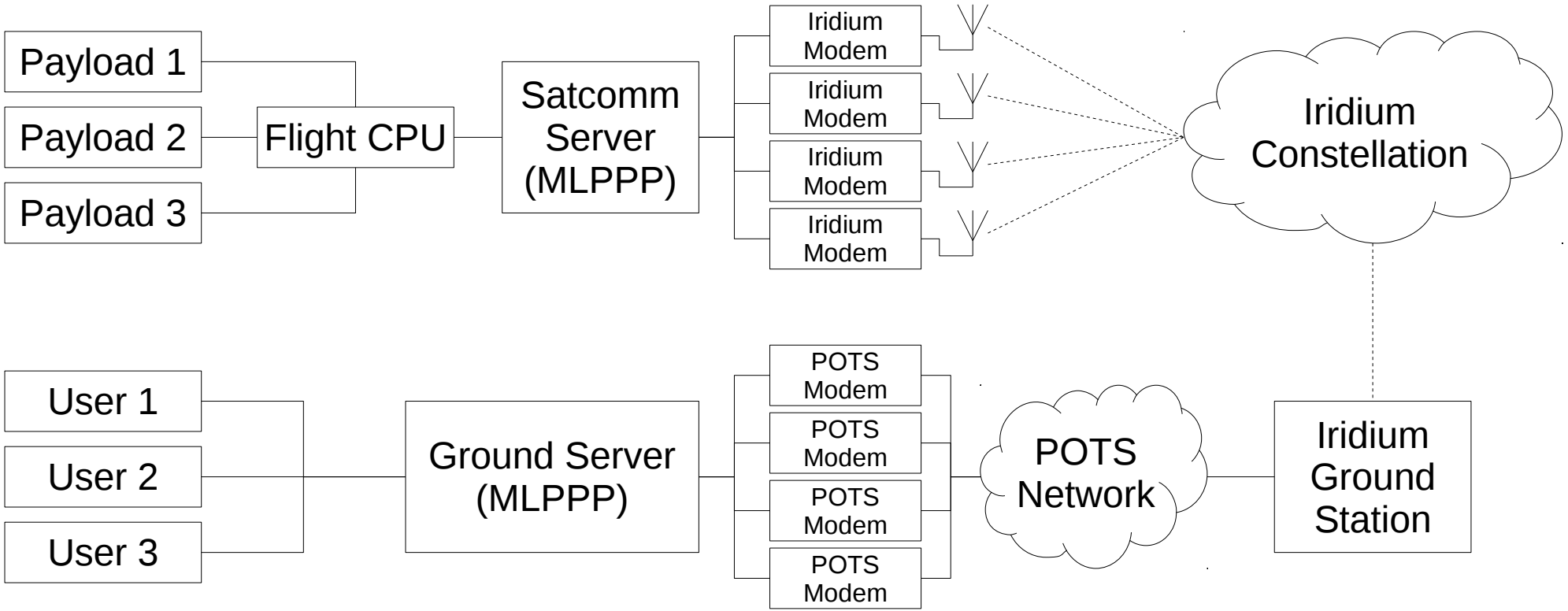
mptcp working group

IETF 98 – Chicago, IL

Quick Background

- Iridium modems are used to communicate to on-board payloads
- Channels are bonded using standard Multi-Link PPP (MLPPP)
- System uses both UDP and TCP
  - TCP/IP performs poorly - Cannot discern losses between links
- Desire to scale the system to even more links (ie: 8, 12)
  - MLPPP breaks down rapidly after 4 in this environment
- Additional Goals
  - Ensure fairness between flows as link conditions degrade
  - Increase reliability in connections

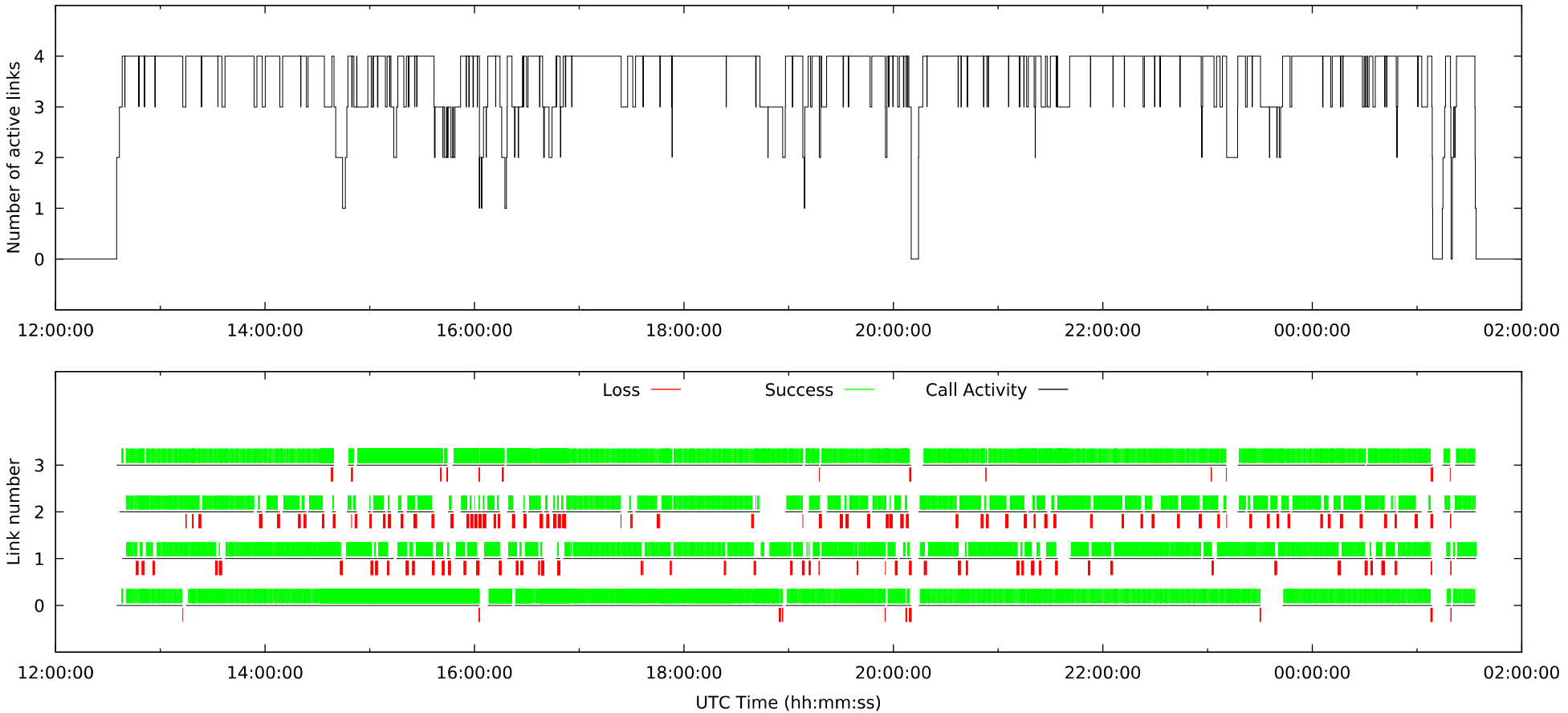
Existing Architecture Illustration



Link Characteristics

- Iridium Modems “go down” fairly often similar to poor cell phone service
  - Degrade: Some information is lost but the call is maintained
  - Drop: Total loss of link, similar to dropping a cell phone call
- The fully operational system is slow by modern standards
  - Each Iridium link is rated at 2.4 Kbit/s or 300 bytes per second
  - Currently 4 channels are used to provide a total of 9.6 Kbit/s
- Round Trip Time (RTT) is very long
  - Roughly 2 seconds for SYNs
  - Roughly 4 seconds for a 500 byte packet

Test Flight November 18

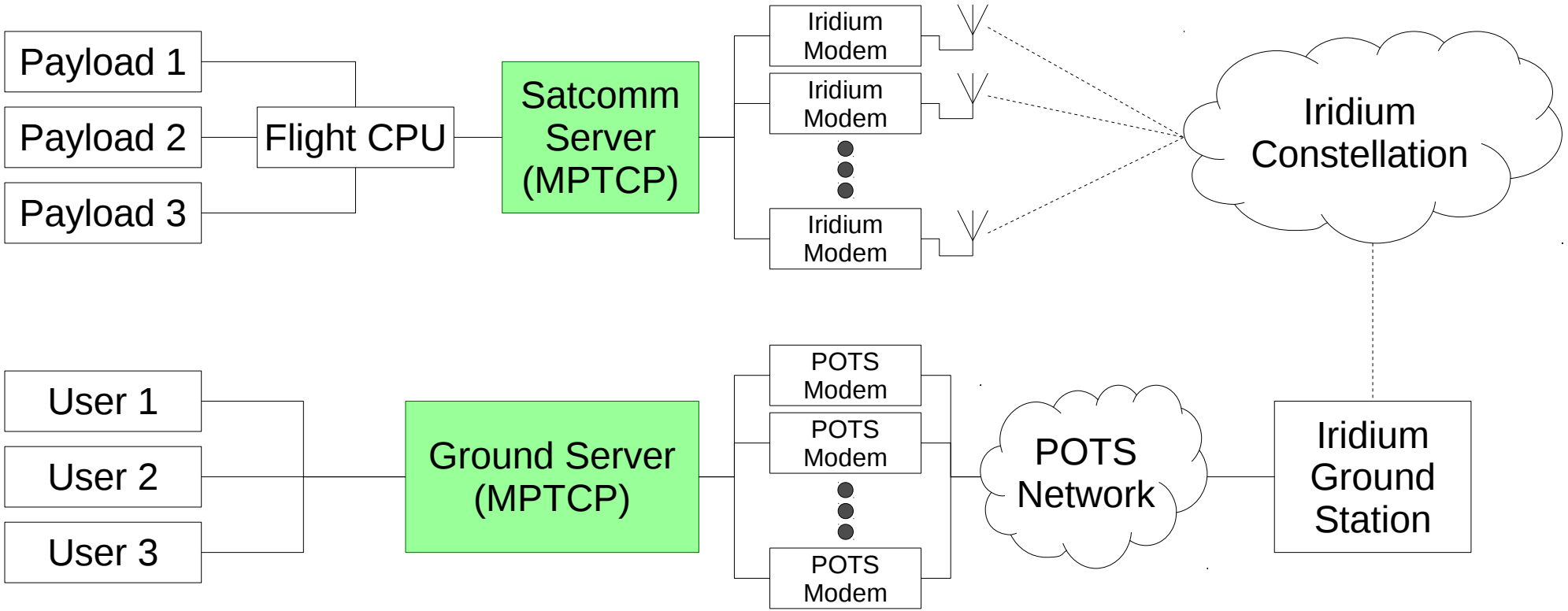


Frequency of Transient Links

- **Flight Duration:** 13 Hours
- **Events that changed the number of active links:** 325
  - 25 changes / hour
- Nearly one fourth of the flight is in a “degraded” state

Number of Active Links	Seconds	Percent
4	35643	76.26%
3	8269	17.69%
2	1969	4.21%
1	235	0.50%
0	624	1.34%

MPTCP Architecture Illustration

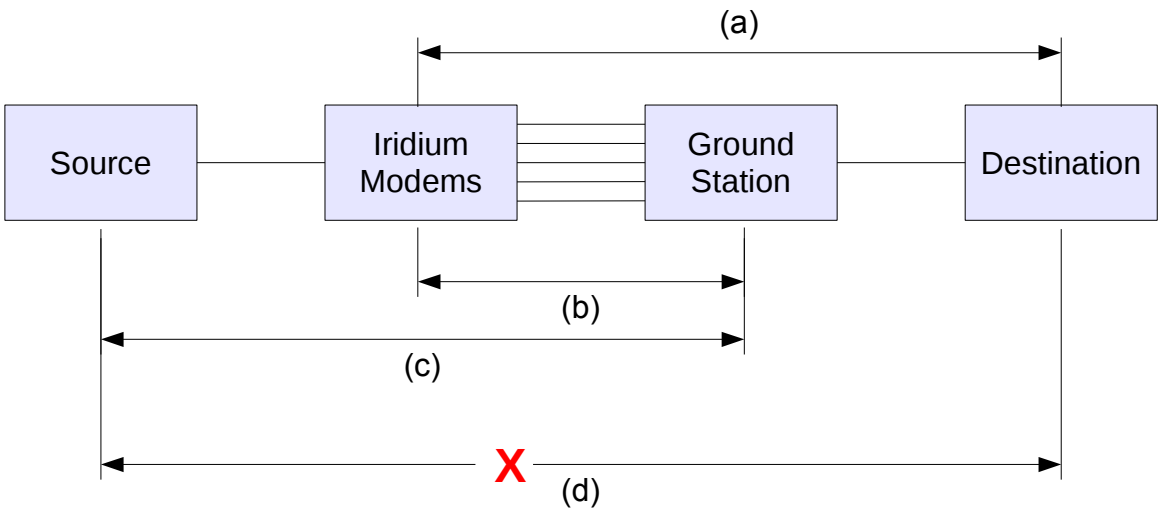


Handling MPTCP Endpoint Limitations

- MPTCP designed to work best when at least one side is directly attached to the point of multiple interfaces (and thus paths)
- Both endpoints must be MPTCP aware

Example, if all nodes have MPTCP enabled, options (a), (b), and (c) all benefit.

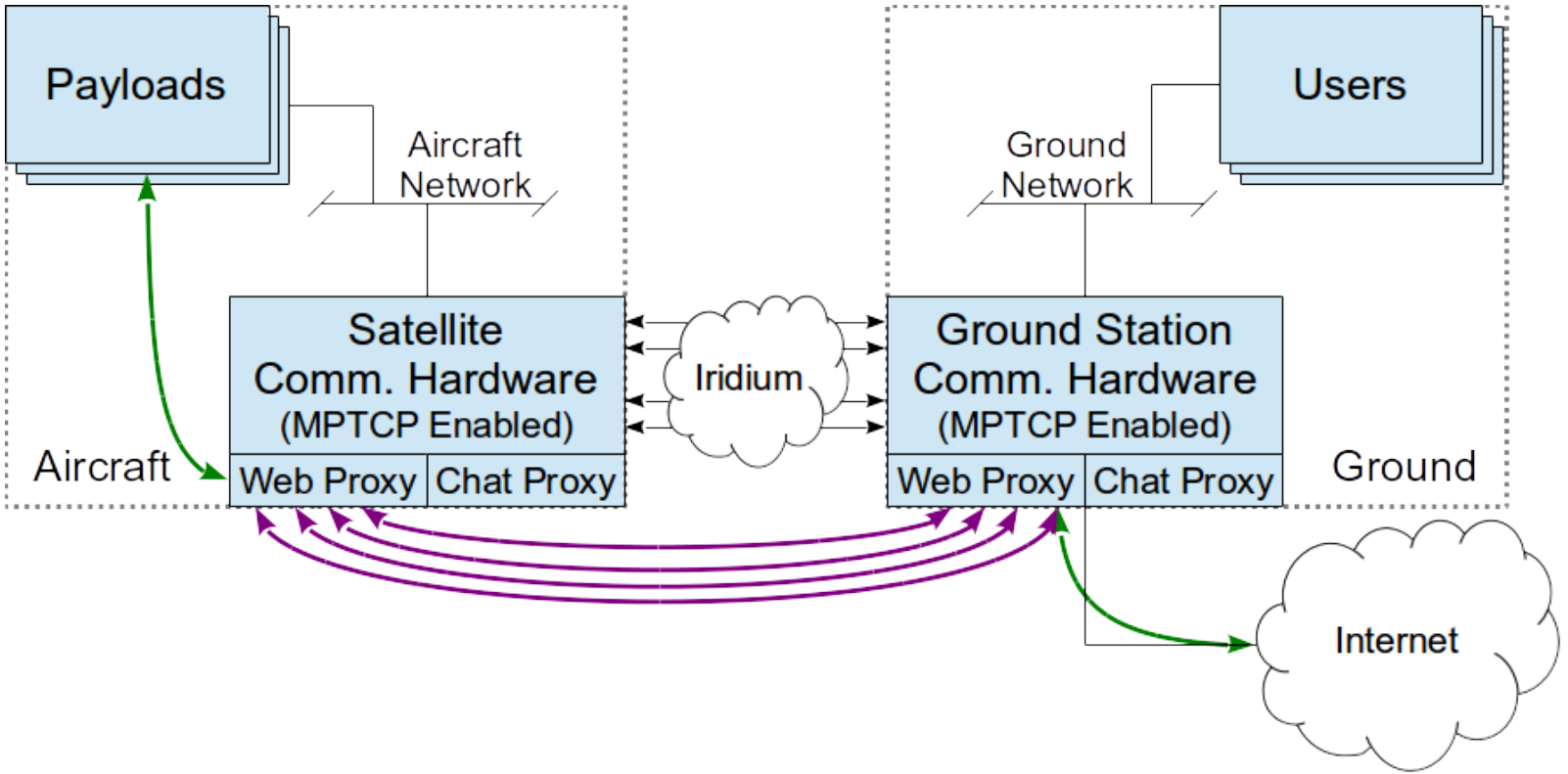
However, option (d) cannot as neither side knows the number of paths.



Service Specific Proxies

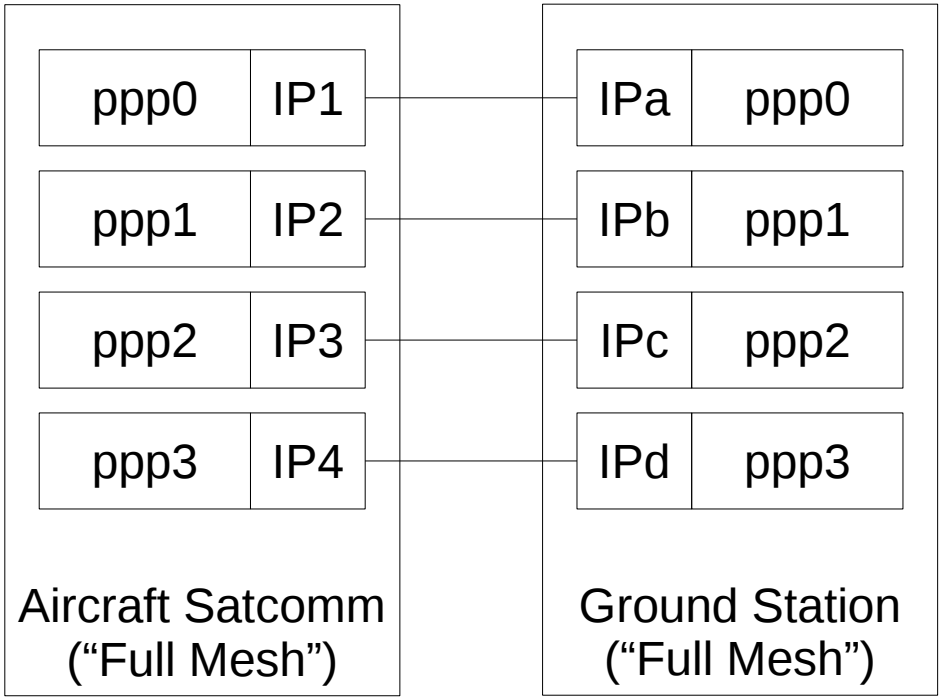
- Used a set of proxies or servers using open source solutions for the types of services in use during flight
  - HTTP proxy (Squid)
  - IRC server (unrealircd) – configured as a chat proxy (a “hub”)
- Installed a proxy on each MPTCP system
  - Flight CPU attached to the Iridium links
  - NASA ground station

MPTCP Flight Configuration



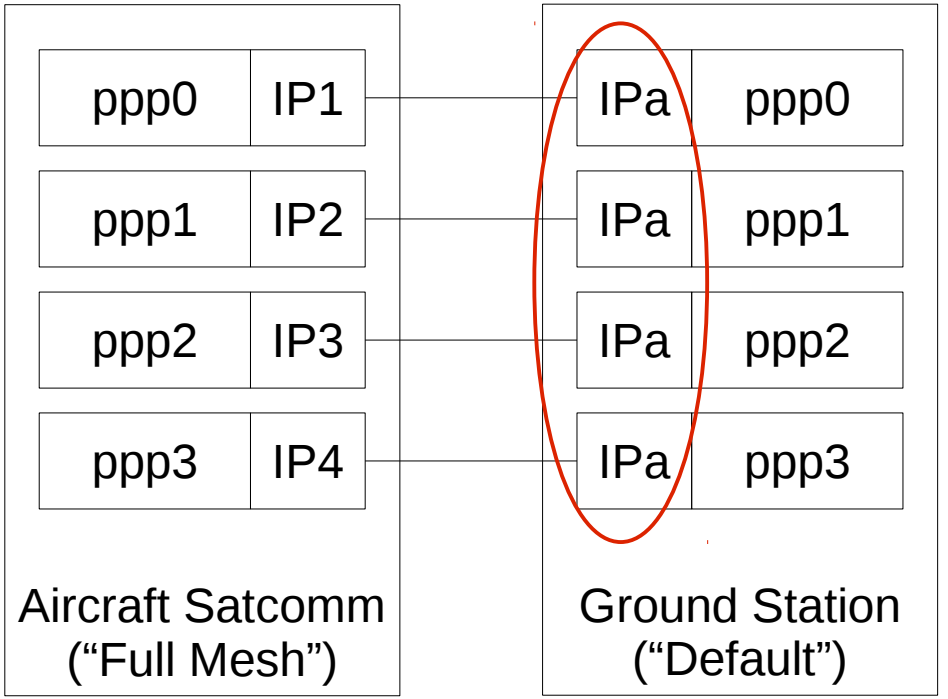
Initial Problems Configuring MPTCP

- First Configuration Attempt
  - IP address for each PPP interface
  - Full Mesh path manager
- Used IPtable rules to limit cross flows (ie: IP1 to IPb)
- Implementation issue limited number of sub-flows to 32
- Complexity in configuration increases rapidly with additional interfaces



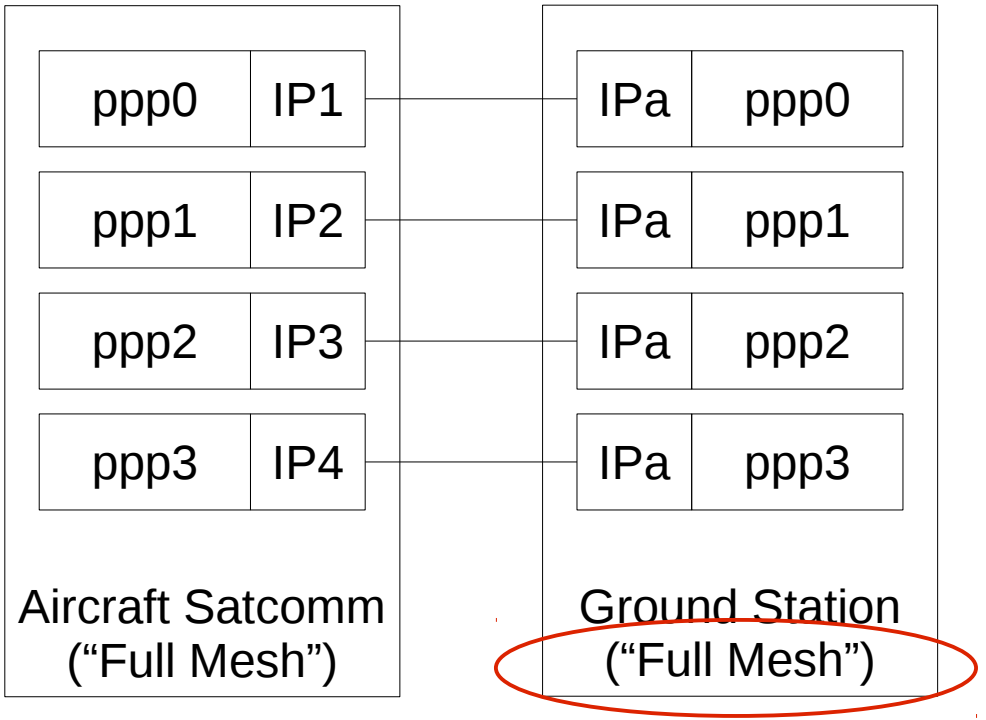
Alternate 1: Using Default Path Manager

- Reduced the number of sub-flows generated by the aircraft
- Works great for connections initiated from the aircraft
- Does not allow MPTCP use from the ground to the aircraft
  - Limited to a single normal TCP connection



Alternate 2: Almost Works

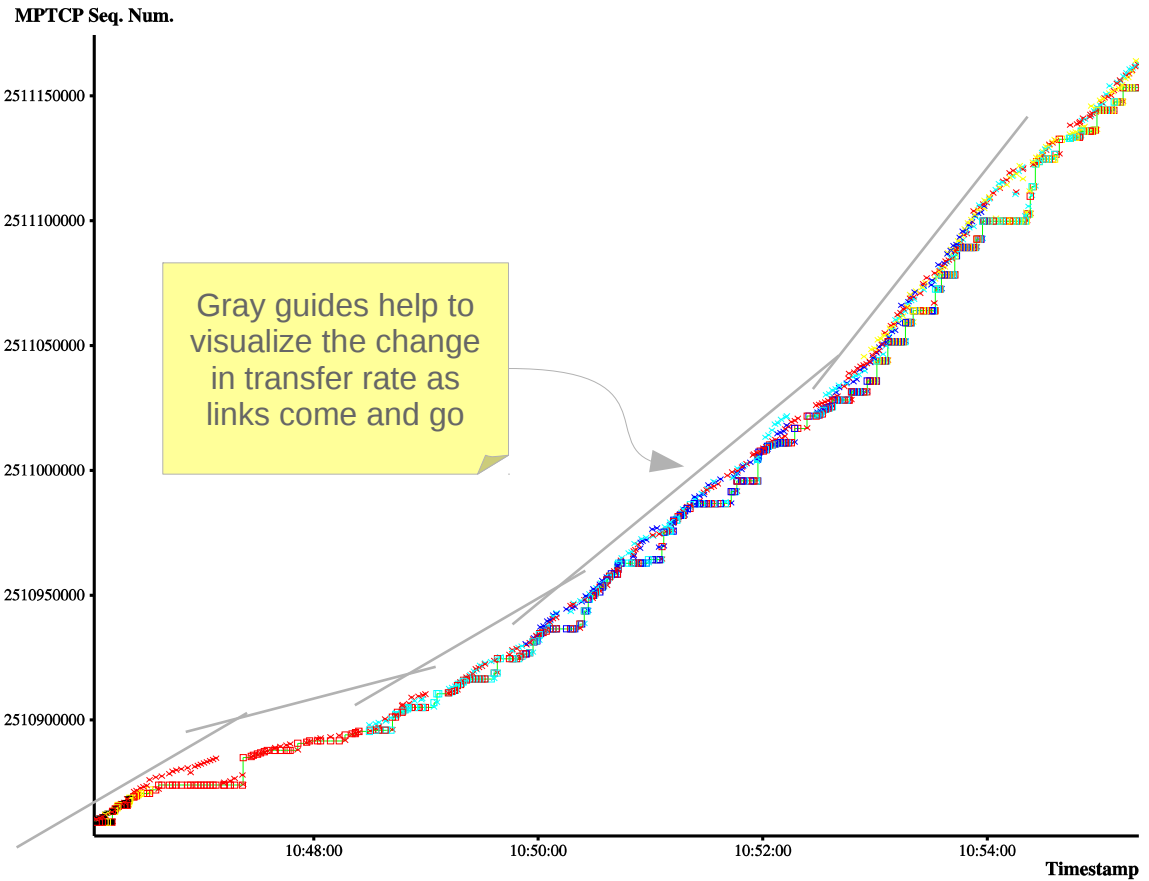
- Changing ground station back to the full mesh scheduler allowed ground to air connections to establish multiple sub-flows
- New Issue: Remove Address for any sub-flow containing IPa would remove ALL sub-flows



MPTCP Implementation Patch

- Patch contributor: Christoph Paasch (Thank you!!)
- Issue:
  - MPTCP would tear down all sub-flows if it encountered a REMOVE_ADDR for the single ground station address
- Solution:
  - Add an option to disable generating REMOVE_ADDR
    - Enabled at the ground station, where only a single address is used
    - Aircraft no longer tears down all other active and healthy sub-flows

Example MPTCP HTTP Connection



Results

- MPTCP did an excellent job of keeping connections active during multiple link transitions
  - Allowed long lived healthy connections
  - Dynamically leveraged the amount of available resources
  - Greatly improves connection stability to the end users
- Service specific proxies worked well in conjunction with MPTCP
  - Not all future services may have easy proxy options (ie: ssh)
- MPTCP is not magic
  - System still suffers if resources are strained (ie: opening 20 TCP connections)

Implementation Observations

- Current implementation has some hard coded limits
  - Cannot use more than 8 addresses, 32 total sub-flows
  - Spec clearly allows for many more
- Implementation will send more than two consecutive ACKs to fit all MPTCP options (Particularly MP_JOIN)
  - Actually beneficial in our case – reduces setup time
  - OK if done prior to any data? No chance to trigger fast retransmit?
  - Need for more TCP option space?
- REMOVE_ADDR behavior

Thank you

Questions?

Backup Slides

Handling UDP Traffic

- MPTCP is TCP/IP specific
  - Needed a solution to “route” UDP traffic across all available PPP links
- Created simple open source program that routes data from payloads and transmits it evenly over all available Iridium channel(s)
  - Use smart queues to store only the latest data from each UDP source
  - Replaces the manually tuned filtering functionality
  - Fairly limits all sources and adjusts dynamically
  - Will throttle all sources when TCP data is present or if channels are lost