NASA/TM—2018-219777

Improving Scientific Payload Communication Over Multiple, Low-Capacity Links

Joseph A. Ishac
Glenn Research Center, Cleveland, Ohio

Matthew T. Sargent
Peerless Technologies, Brookpark, Ohio

June 2018

NASA/TM—2018-219777

Improving Scientific Payload Communication Over Multiple, Low-Capacity Links

Joseph A. Ishac
Glenn Research Center, Cleveland, Ohio

Matthew T. Sargent
Peerless Technologies, Brookpark, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

June 2018

Acknowledgments

The authors would like to acknowledge the Earth Science Technology Office for supporting this effort to improve communications for airborne missions. The authors would also like to acknowledge members of the NASA Ames Research Center for supplying key flight hardware and modems and the flight crew members for support of this experimental software during their November flight. Finally, special thanks to colleagues Dennis Iannicca and William Ivancic who assisted and provided valuable input at various points during this effort.

Abstract

Communication is an important part of every airborne science mission. It provides a means for a large group of ground-based scientists to monitor and adjust experiments aboard aircraft that often cannot support a large number of passengers. In some cases, such as unmanned aircraft, communication becomes critical to a successful mission. Existing communication systems require the use of multiple, low-capacity channels, or links, to handle the capacity requirements of the scientists. Current communication solutions attempt to balance performance and reliability, but ultimately suffer from some critical design flaws that limit the ability of the system to handle additional links. This work shows how adding resources to the current system results in decreased performance and reliability. It explores the use of a new communication protocol, MultiPath Transmission Control Protocol (MPTCP), to make better use of multiple communication links and propose a total system capable of scaling to any number of channels efficiently and dynamically. The tools developed to support and measure this solution are also detailed.

1 Introduction

The NASA Airborne Science Program within the Earth Science Division supports missions to gather scientific data from around the world. Researchers rely on satellite links as a communication channel between the aircraft and users on the ground while in flight. Many missions have a choice as to which satellites to use for communication, but missions near the north or south pole are forced to use the Iridium® satellite constellation for communication.

An individual Iridium® modem provides only 2.4 kilobits per second, or about 300 bytes per second, of network capacity. For reference, a typical Ethernet frame of 1500 bytes would take 5 seconds to transmit. Systems typically get around this limitation by installing multiple Iridium® modems and communicating across multiple links simultaneously. Communication over multiple links requires some form of extra overhead and data management, depending on the level of service required and how the information is transmitted over the multiple links. Duplicating or splitting messages would require some mechanism to deduplicate or join logical units to reconstruct the original.

Currently, flights leverage the MultiLink Point-to-Point Protocol (MLPPP) to handle sending data across multiple Iridium® links. MLPPP combines the multiple physical links into one logical link between the aircraft and the ground.

While MLPPP has been used on NASA flights with some amount of success over the years, the following problems have been observed with MLPPP-based systems that leverage Iridium® for communication:

MLPPP links experience fate sharing: Loss of a data fragment on one Iridium® link is logically the same as losing data on all of the available Iridium® links. As individual links may take up to a minute to completely fail, fate sharing can cause minutes-long blackout periods where no data successfully makes it across the links.

MLPPP requires user intervention: In certain cases, MLPPP must be reset manually in order to use all of the available Iridium® links. This reset requires manual monitoring and intervention and part of the reset involves intentionally turning off functioning links, which creates a communication blackout period. Furthermore, traffic sent across Iridium® must be rate-limited in order to avoid overwhelming links with too much traffic. This rate must be adjusted manually in flight as Iridium® links come and go over time.

Queues can add to system latency: End hosts sending data too quickly across Iridium® can cause a queue of packets to build up in the network. As packets are sent across Iridium®, they experience additional delay proportional to the amount of data already enqueued at the link.

MLPPP uses one queue: MLPPP conveniently provides a single logical network path to carry data across multiple links at once. A consequence of this is that all traffic must share a single queue for the MLPPP link where the oldest traffic is sent first. Any new, time critical data must wait for all previous traffic in the queue to be sent first before it can be sent.

The remainder of this report details the design of a new communication system that leverages a new network protocol, the MultiPath Transmission Control Protocol (MPTCP), to address and improve upon the aforementioned problems that have been observed with the current system.

Section 2 begins by defining relevant terminology and network protocols. Section 3 describes the available equipment and how it is configured. Section 4 describes the MLPPP-based communication system and demonstrate the observed problems mentioned in this introduction. Section 5 explores individual system changes and the problems they help to address.

A key success of this effort was taking the lab-developed system and using it to control the communication channel during a test flight in November 2016. Section 6 details the modifications the system requires to fully support a flight and details key insights and results from the test flight. In addition to a successful flight test, a number of deliverables were also generated for both NASA and the public. Section 7 describes these deliverables and their intended audience and use. Section 8 provides some concluding remarks and notes on future work.

Nomenclature

BLOS	beyond line of site
byte	One octet or eight bits
FIFO	first in, first out
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IRC	Internet Relay Chat
LCP	Link Control Protocol
MLPPP	MultiLink Point-to-Point Protocol
MPTCP	MultiPath Transmission Control Protocol
MTU	maximum transmission unit
POTS	plain old telephone service
PPP	Point-to-Point Protocol
RTT	round-trip time
SFQ	Stochastic Fairness Queuing
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTC	Coordinated Universal Time

2 Background

2.1 Earth Science Flight Campaigns

The Earth Science program leverages multiple types of aircraft for their wide range of flight campaigns. In addition, each campaign or mission carries with it a different set of communication requirements, which can be impacted by the type of aircraft being used, the location of the mission, and the types of users or payloads. Most missions leverage some form of satellite communication for beyond line of site (BLOS) communication to researchers or payloads aboard the aircraft.

The type of aircraft plays an important role in the forms of communication that can be supported. Larger aircraft are capable of physically carrying more complex and high rate communication systems, like those needed to communicate to geostationary satellites. However, smaller aircraft are often limited to lighter and smaller solutions. Iridium® uses a simple omnidirectional antenna to communicate and multiple antennas can be leveraged aboard smaller aircraft easily.

The location of the mission also plays an important role. During missions in the Arctic or Antarctic regions, aircraft cannot use geostationary services commonly used for aircraft voice and data communication channels. Regardless of size, once an aircraft is above 72 degrees latitude, geosynchronous satellites are so low in the sky as to be problematic and above 80 degrees latitude they are below Earth's horizon. Currently, the only solution for these deployment scenarios is to use Iridium® satellites that have a polar orbit.

Satellite connectivity allows for improved science in flight. For example, a satellite connection allows scientists aboard the aircraft to obtain the latest weather imagery, or a large team of researchers on the ground to observe and adjust experiments during flight. Some missions leverage unmanned aircraft to perform scientific measurements using a set of installed payloads. In this situation, the communication to the aircraft is critical, as it allows the researchers to command and monitor the experiment and correct any faults that would otherwise go undetected until the aircraft were to return to base.

2.2 Iridium®

The Iridium® satellite constellation is a mesh network of low-Earth orbit satellites that provide global voice and data services to customers through the use of satellite modems. Data services are typically provided by placing a data call from an Iridium® modem to an analog phone modem on the ground and establishing a Point-to-Point Protocol (PPP) link over this call. Data calls provide an average of 2.4 kilobits, or around 300 bytes, of data per second across the Iridium® link.

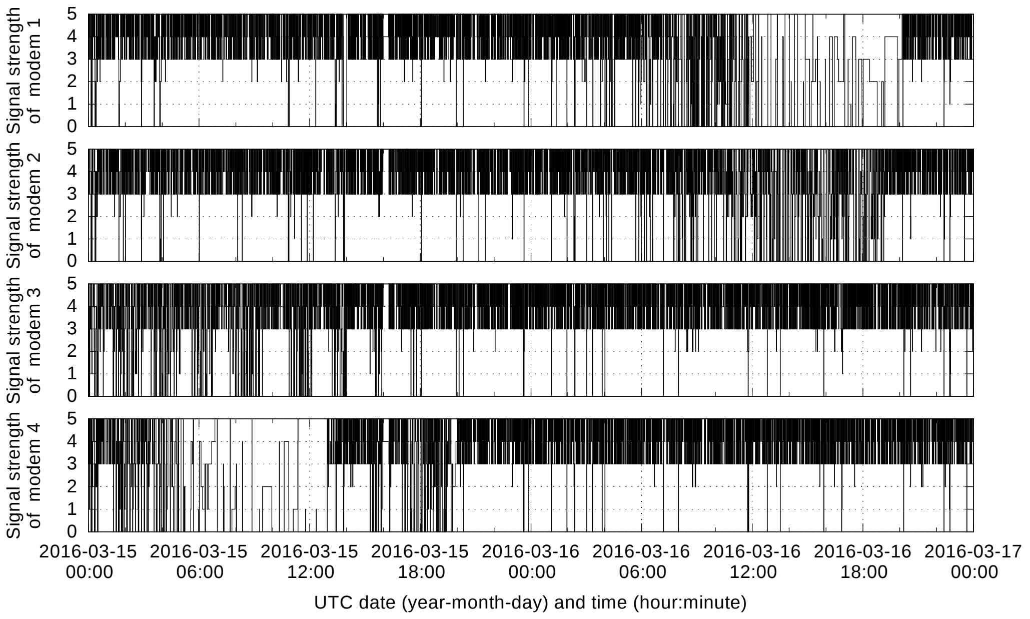

Figure 1. Iridium® signal strength variations over 2 days.

Communication over Iridium® heavily relies on the signal strength between an Iridium® modem and the nearest Iridium® satellite. Iridium® modems report signal strength on a scale of 0 to 5, with 5 being the strongest signal (much like bars on a cellphone). As signal strength degrades, the chance of losing data over Iridium® increases as a lower signal strength coincides with a higher bit error rate.

User-reported problems with Iridium® link quality on airborne missions are a motivating factor of this work. One of the goals is to study the Iridium® constellation and model link quality over time.

In order to study these link conditions, four antennas were mounted on the roof of the laboratory located at the NASA Glenn Research Center in Cleveland, Ohio.

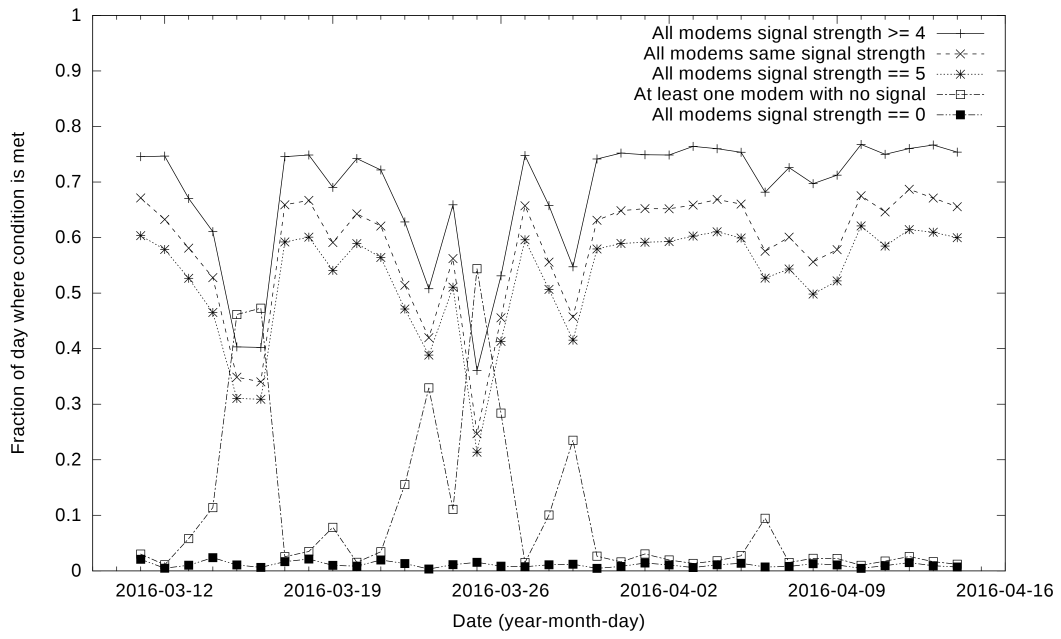

Figure 2. Combined view of Iridium® signal strength.

The Iridium® modems are identical in hardware to those used in flight conditions. The modems were then configured to provide a signal strength value as needed[1] as conditions changed, providing the ability to monitor the signal strength to the Iridium® constellation.

Figure 1 shows the Iridium® signal strength of the four modems individually over an identical 2-day period of time in March of 2016. The figure shows how quickly the signal strength for any particular modem varies, often having over a hundred different values in a given hour. Also note that degraded link states (where the signal strength is less than four) are a common occurrence. Furthermore, how often and which links are in a degraded state changes on a day-to-day basis. Good examples of this can be seen between 06:00 and 12:00 on March 15, where modem 4 is severely degraded, while modem 1 suffers similarly the following day.

Figure 2 shows an expanded view of the measurements in 2016. Instead of showing individual measurements, a series of conditions are plotted with each point representing the fraction of the day for which the condition is met. The best case scenario, where all four modems are at maximum signal strength, only happens for half of the day or 53 percent on average. The next best scenario, where all modems are at a level of 4 or higher, happens 68 percent of the day on average. Several dips are seen throughout the capture, with the left most dip corresponding to the same days shown in figure 1. The sharp decreases in performance are attributed to days when one or two modems seem subject to severe degradation.

[1]By issuing a +CIER command to the modem, it will produce a stream of unsolicited responses containing the receive signal strength indicator level. The message is only produced when the level changes. Furthermore, calls cannot be placed when in this mode of operation.

Several conclusions can be drawn from these results. First, that degraded conditions are not rare, happening at least a third of the day or more. Second, that periods of degradation can be spread out, and can occasionally impact one or more modems very severely. Finally, when monitoring multiple modems at once we find periods of time where a subset of modems is degraded while the remaining modems show a high signal strength. This occurs even though the antennas the modems use are all located in the same location with similar, unobstructed views of the sky. At other times, we observe all of the modems showing similar signal strength for days at a time. Thus, at any point in time there is likely not a single Iridium® signal strength that all modems will experience. The idea of variable signal strength and link availability matches the observations presented in Section 6.

2.3 MultiLink Point-to-Point Protocol

The MultiLink Point-to-Point Protocol (MLPPP) is a method for splitting and recombining data across multiple PPP links simultaneously. By leveraging multiple PPP links, MLPPP provides an amount of capacity equal to the aggregate amount of capacity on the individual PPP links. Further, in cases where propagation delay is a limiting factor, MLPPP attempts to reduce the perceived delay by fragmenting data units across all active links.

Within the kernel, MLPPP will set up a bundle made up of one or more PPP links. The bundle presents itself to the operating system as one network interface and contains a routing rule between the two ends of the MLPPP bundle. As the number of PPP links fluctuates (either new PPP links are established or PPP links fail or are shutdown) MLPPP dynamically tracks the number of available links and controls the flow of data across these links accordingly.

Sending data across all of the underlying PPP links in an MLPPP bundle utilizes a process called fragmentation. When MLPPP receives a packet, it divides the packet into a number of fragments equal to the number of PPP links in the bundle. These fragments are split as evenly as possible based on the original packet size and each fragment carries a sequence number that is used to reassemble the original packet on the other end of the MLPPP bundle. Note that all fragments must be received at the remote end of the MLPPP bundle to reassemble a full packet. If fragments are lost or corrupted, then a packet is lost and any buffered fragments belonging to the lost packet must be discarded.

Individual PPP links are responsible for detecting their own failure through the use of Link Control Protocol (LCP) packets. Each end of the PPP link will send periodic echo request packets to the remote end of the PPP link. Upon receiving an echo request, an echo response will be generated and sent back across the PPP link. PPP links are configured to detect a link failure after N consecutive echo responses have not been received. NASA flights generate echo packets every 30 seconds and set $N = 2$ echoes in a row that must fail before a link is removed. This creates a 1-minute period required to successfully detect a PPP link has failed and remove it from the MLPPP bundle.

2.4 MultiPath Transmission Control Protocol

The Transmission Control Protocol (TCP) is a transport layer protocol that enables two applications to communicate over a reliable byte stream. TCP connections are restricted to communication over a single network path even in cases where multiple network paths exist between the communicating end hosts.

The MultiPath Transmission Control Protocol (MPTCP) is an extension to TCP that enables two applications to communicate over a reliable byte stream while leveraging one or more network paths simultaneously. One example of a device with multiple paths would be a cellphone that has both a WiFi interface as well as a 4G interface. A regular TCP connection could only communicate over one of the available interfaces, but MPTCP can send data using both interfaces at once.

Within the operating system, MPTCP is backwards compatible with TCP and presents the same interface to applications that TCP does. This means that once MPTCP is installed on a machine, all applications on that machine can be capable of leveraging MPTCP for communication.

In addition to being backwards compatible with TCP, MPTCP also has safeguards built into it that allow communications to fall back to regular TCP if needed. The rationale for this fallback is that there may be certain networks that will block MPTCP traffic, but will allow TCP traffic through. On these types of networks, end hosts will attempt to establish communication over MPTCP, but will fall back to regular TCP after several failed attempts.

MPTCP kernel code handles data flow management across all of the available network paths for each ongoing connection. There are two main components to handling the flow of data for an MPTCP connection, the path manager and the data scheduler. The path manager handles establishing and tearing down communication across multiple network paths over the lifetime of a connection. For example, if two hosts were communicating over one PPP link and then a second PPP link becomes available, the path manager would begin to establish a data flow across the newly available path. Likewise, a PPP link that fails will be removed from a connection by the path manager. The data scheduler is responsible for choosing which of the available paths to send data across during an MPTCP connection. Exactly how the scheduler chooses which path to send data across is left up to local policy by the MPTCP specification.

2.5 Differences Between MLPPP and MPTCP

While MLPPP and MPTCP manage data flow across multiple paths in the network, there are key differences that should be understood about these two technologies.

Figure 3 shows how MLPPP and MPTCP exist at different layers of the network stack. MLPPP is a link layer technology, whereas MPTCP is a transport layer technology. Existing at these different layers means that each technology offers a different level of service and capabilities for the traffic being carried.

As a link layer protocol, MLPPP only sends data across one network link between two machines. In this way, an MLPPP bundle performs a similar function to Ethernet. MLPPP simply takes packets from one end of a bundle and sends

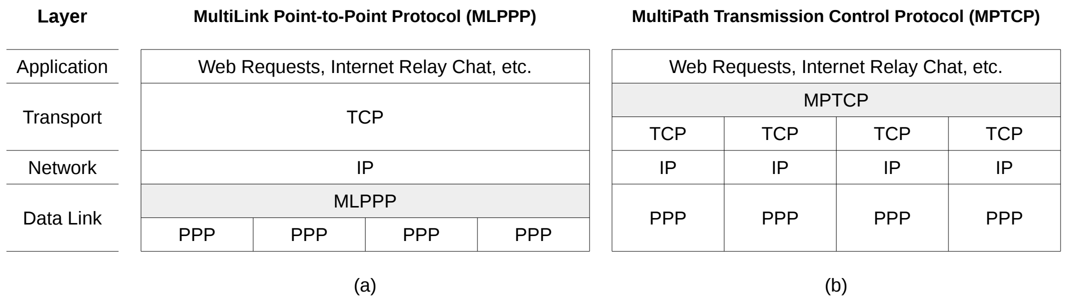
Layer
MultiLink Point-to-Point Protocol (MLPPP)
MultiPath Transmission Control Protocol (MPTCP)
Application
Transport
Network
Data Link
Web Requests, Internet Relay Chat, etc.
TCP
IP
MLPPP
PPP
MPTCP
(a)
(b)

Figure 3. Layer diagram showing (a) MLPPP and (b) MPTCP.

them across the links to the remote end of the bundle. Any data destined for a host not directly connected to either end of the bundle is not a concern of the MLPPP protocol.

MPTCP provides end-to-end connectivity across arbitrary network paths made up of multiple network links. MPTCP does not need to concern itself with exactly which link layer technologies are in use as long as the two hosts have a network path over which to communicate. MPTCP manages data flow across one or more network paths and trusts that the underlying link layer technologies along the communication paths will successfully send its packets between hosts.

The highlighted portions of figure 3 also shows the point in the stack where data is split. MLPPP takes a single data unit and creates fragments across active PPP links, while MPTCP creates multiple TCP subflows, one for each interface.

In addition to carrying data only over a single network link, MLPPP bundles only send a particular piece of information once. MLPPP has no retransmission mechanism or reliability built into the protocol and any piece of data that is corrupted will be lost. MPTCP provides a reliable byte stream between two end hosts. Data will continue to be sent, sometimes across multiple paths, until the data is successfully received and acknowledged by the end host receiving a piece of data.

Since MLPPP and MPTCP exist at different layers of the network stack, it is possible for MPTCP traffic to be sent over top of an MLPPP bundle. We do not explore this method as the underlying problems with MLPPP over Iridium® would also impact MPTCP. In addition, using MLPPP in conjunction with MPTCP would limit MPTCP to one TCP subflow, further negating any benefit.

3 Equipment

This effort relies on test equipment capable of generating MLPPP and MPTCP traffic in order to fully understand how these protocols react to a variety of network conditions. We have two testbeds that each serve similar, but distinct functions.

3.1 Research Testbed

We refer to our first testbed as the research testbed. It consists of two Linux computers, each with a 4-port modem card installed. The two machines are connected with phone lines that run directly between pairs of ports on the modem cards. The two machines are capable of placing calls and negotiating either PPP or MLPPP over the phone lines. Each machine has MPTCP-enabled kernels installed.

The primary use of this testbed is to perform controlled tests of various combinations of network protocols and to build up our understanding of protocol behavior in a lab setting. For example, when first exploring problems with MLPPP over Iridium®, the research testbed allowed for an understanding of MLPPP in a wide variety of network conditions before then understanding MLPPP in an Iridium®-like environment. This helped to establish the expected behavior of MLPPP over Iridium®, which was later verified by performing similar tests on Iridium®.

3.2 Engineering Testbed

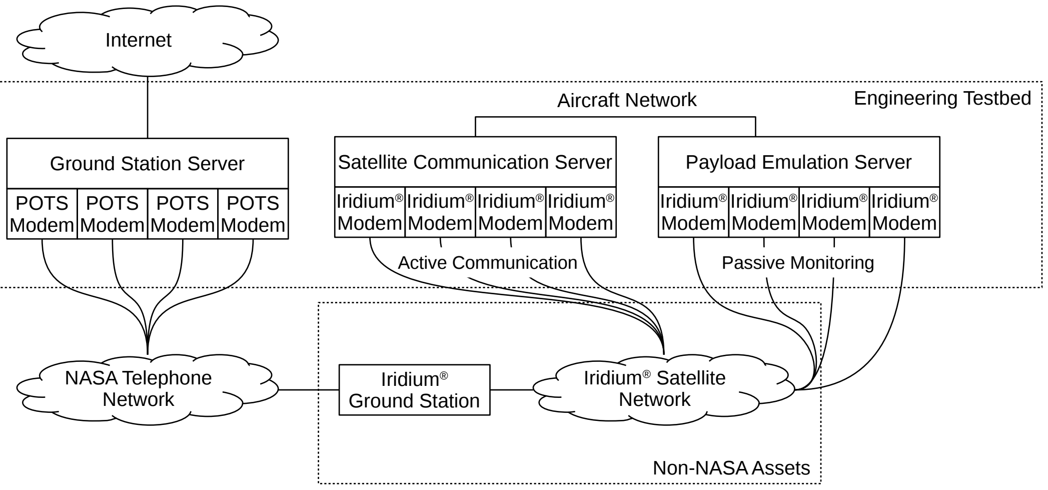

Figure 4. Engineering testbed.

The Iridium®-capable testbed is referred to as the engineering testbed. It contains three separate machines and is illustrated in figure 4. The first is a server with a 4-port analog phone card. This server acts as a ground station to answer calls from Iridium® modems, and thus, becomes the ground side of any MLPPP or PPP interface that is established.

The second machine in the testbed contains a spare piece of flight hardware from Earth Science campaigns. This piece of hardware is another Linux computer that is used in production for communicating over Iridium®. It is attached to four separate Iridium® modems and serves as the air side of all data calls over Iridium® when MLPPP or PPP links are established.

The third machine in the engineering testbed is hooked up to four spare Iridium® modems. This machine serves two main purposes. First, the machine is sometimes used to simulate an instrument on board an aircraft that generates network traffic meant to be carried across Iridium®. The second use of this machine is to monitor the spare Iridium® modems for their signal strength over time.

The engineering testbed allowed for testing of solutions developed in the research testbed and for making adjustments based on what was observed. Sometimes, tests aimed at learning about a particular aspect of the Iridium® constellation were run with the engineering testbed. At other times, the engineering testbed was used to design a replacement system for the MLPPP-based solution that NASA flights rely upon. Given a list of required network traffic that needs support in flight, the engineering testbed can be used to design and test an MPTCP-based system under realistic conditions.

4 Existing Configuration

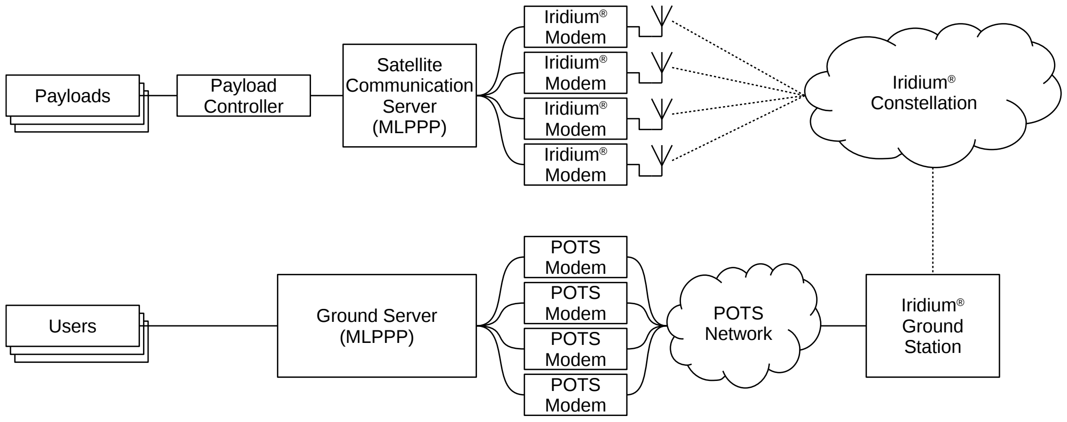

Figure 5. Existing MLPPP configuration used in Earth Science missions.

The existing MLPPP-based configuration is shown in figure 5. A single machine on the aircraft is in charge of interfacing with the satellite systems and establishing a MLPPP bundle. The satellite communication server dials into a NASA-operated ground station over one or more Iridium® links. This bundle serves as the single network path to route traffic between the aircraft and the ground.

Currently, flights send several types of network traffic across the MLPPP bundle. In particular, instruments and payloads on board the aircraft generate User Datagram Protocol (UDP) status packets destined to the ground users and systems. However, payloads do not communicate UDP traffic directly to the ground, but rather to the aircraft's payload controller, which is manually configured to limit the amount of data being sent across the MLPPP bundle.

Regular TCP traffic is also sent across the MLPPP bundle, often to give users on the aircraft the ability to obtain weather imagery from the Internet or provide researchers a means to chat and communicate with the ground crew. TCP traffic is not rate-limited, tuned, or adjusted from the system defaults.

When sending data across the MLPPP bundle, UDP and TCP network traffic are all placed into one first in, first out (FIFO) queue where the oldest traffic is sent across the MLPPP bundle first. Users of the existing system have ranked some traffic with more importance, but no effort to prioritize traffic or provide fairness among the different types of traffic is present in the current system.

4.1 Observed Problems

As mentioned in Section 1, there are certain difficulties that are observed when using the MLPPP-based system for flight communication support. This section dives further into each observed problem and provides a detailed description and visual example of each problem. Section 5 will detail techniques that reduce the impact of each problem. In certain instances, observed problems will be mitigated in such a way that they no longer have any impact on flight communications given the new setup that uses MPTCP.

4.1.1 Fate Sharing

Recall from Section 2.3 that MLPPP bundles will fragment packets by splitting packets up into one fragment per underlying PPP link and send these fragments across the PPP links simultaneously. All of the fragments must successfully be received on the far end of the MLPPP bundle, or else the original packet cannot be recreated and any fragments making up a partial packet must be discarded. A single failure on a PPP link means that all of the other PPP links also experience a failure. This concept of linked components either all succeeding or failing together is termed "fate sharing."

When running MLPPP over multiple Iridium® links, each Iridium® modem may be experiencing a different signal strength than other links in the bundle. Signal strength is correlated with the bit error rate of a link, and a single Iridium® link may experience a loss when others do not. In the flight system, it takes approximately 1 minute to detect a failing link and remove it from the MLPPP bundle. During this time no usable data is successfully sent across the MLPPP bundle.

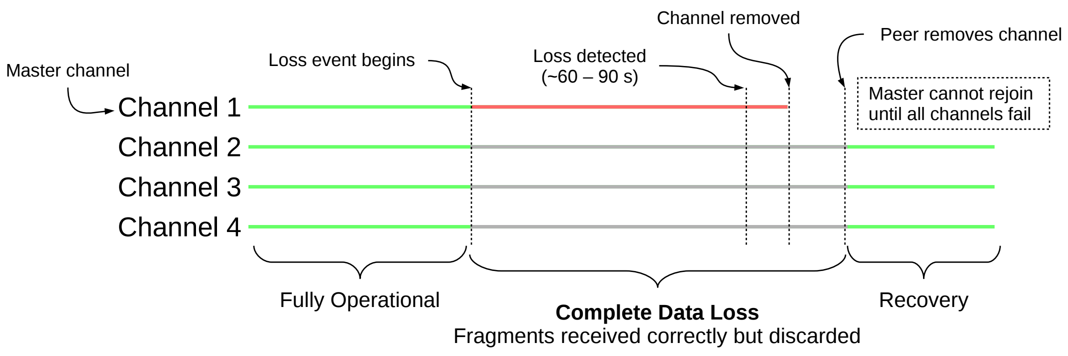

Figure 6. MLPPP fate sharing and master link issues.

Figure 6 shows an example of an MLPPP bundle in the process of losing one of its links. In this example, there are four links in the MLPPP bundle, and one link begins to fail. The remaining three links continue to transmit and receive fragments that are discarded. Once the failing link is removed from the bundle, packets are then fragmented and sent only across the remaining, healthy links and full packets can be reassembled.

Note that when MLPPP experiences a link failure in this manner, no full packets make it across the MLPPP bundle until the failing link is removed. In the flight system, link removal takes approximately 60 seconds to complete. A long loss period like this creates adverse effects for any ongoing TCP connections as these connections will detect loss, timeout, and attempt to retransmit outstanding data. The next timeout period is then doubled as part of TCP's standard exponential backoff procedure. An example of a single link loss and the resulting impact to TCP is described in greater detail in section 5.1.

A loss period of 60 seconds may cause several TCP timeouts and retransmissions to take place, which means that once an MLPPP bundle does recover, TCP will still be sitting idle and waiting for a lengthy timeout to occur before it finally recovers. Users would experience connections with minutes-long periods of delay where the connection is stalled or idle. Such long delays can prompt users to terminate a connection early. Multiple link losses compound this issue further and in some cases will cause the TCP connection to fail completely.

One last issue to take notice of is that increasing the number of Iridium® links in the MLPPP bundle becomes difficult due to the concept of fate sharing. Each additional Iridium® link adds more capacity to the system, but also adds a new failure point that could cause blackout periods as links fail. As such, the current system has limited scalability and serious consideration has to be taken before adding additional links, and thus failure points, to the system.

4.1.2 Single Queue

MLPPP creates one network interface and provides a single network path to route across. By default, this setup also creates one traffic queue for the interface through which all traffic must go. The default queuing rules create one first in, first out (FIFO) queue. This type of queue guarantees that the oldest data in the queue will be the next piece of data to be sent across the MLPPP bundle.

Even if an important, time-sensitive, packet is generated, it will be stuck in the FIFO queue for the MLPPP bundle and will have to wait for all traffic that arrived before it to be sent before the important packet will be sent across the bundle. This may delay the packet enough that its information is no longer timely or useful. Long FIFO queue delays also limit the ability of protocols like TCP to successfully establish new connections in a timely manner.

4.1.3 Oversized Buffers and Excessive Delay

Computer networks often contain heterogeneous links of varying capacity. An end-to-end path through a network has a maximum capacity equal to the bottleneck link, which is the lowest capacity link on the path. Since other links in a path have at least as much capacity as the bottleneck link, it is possible that traffic arrives from higher bandwidth links faster than the bottleneck link can process. When this occurs, outstanding packets will be added to a queue on a router and processed when capacity on the bottleneck link becomes available.

As packets are added to a queue, they will experience additional delay that they would not experience if they were processed immediately. The amount of additional delay is a function of the amount of data already enqueued when a packet arrives and the speed of the bottleneck link. A larger queue size or slower link speed both increase the queuing delay for a packet.

Recall that an Iridium® modem provides 2.4 kilobits per second of available capacity. Since modern networks typically operate on the scale of megabits or gigabits per second, Iridium® is going to be the bottleneck link. This means that if data arrives faster than 2.4 kbit/s at either end of the Iridium® link that a network queue will build up and begin delaying packets.

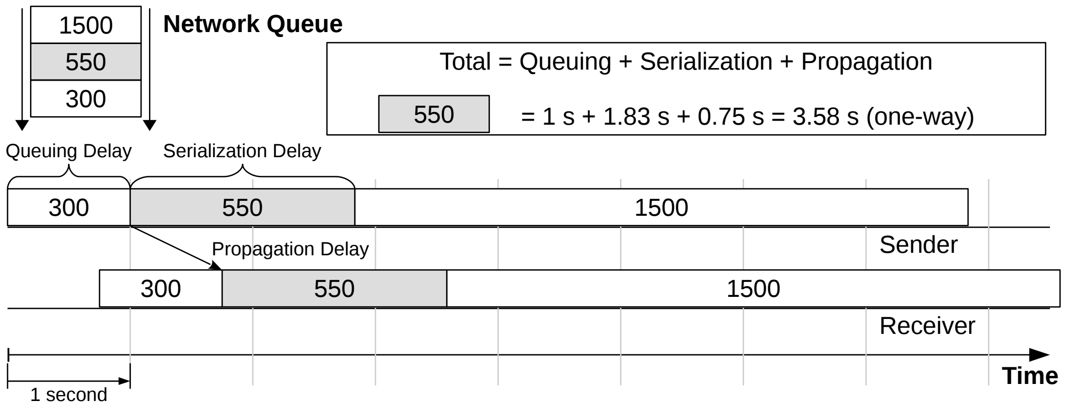

Figure 7. Factors impacting delay over an Iridium® link.

The impact of queuing delay over an Iridium® link can be estimated using the figure of 300 bytes per second. For example, with 600 bytes enqueued, there would be a 2-second delay before transmission of the next data packet could begin. The same figure of 300 bytes per second can be used to estimate the serialization delay, or the delay needed to transmit (and receive) a certain size message. The maximum size of a segment over a PPP link is controlled by the link's maximum transmission unit (MTU). By default, this value is 1500 bytes in modern systems and would result in a delay of 5 seconds. When building a system to handle multiple users over slower links, a large MTU will make the total system seem more unresponsive.

Figure 7 shows the various factors that impact transmission across the Iridium® link. In addition to queuing and serialization delay, time is needed for the signal to propagate through the satellite system and supporting architecture. Furthermore, the Iridium® system is a constellation of satellites and the routing between those satellites is variable, which leads to highly irregular delays. Figure 8 shows the measured propagation delay of different-sized network packets through a single Iridium® link. The figure plots the minimum and average observed round-trip time (RTT) of 35 samples at each network size. The figure also has a fitted linear regression to both the minimum and mean data points. The minimum points intercept the y-axis at roughly 1.5 seconds. Thus, the one-way propagation delay can be estimated to be at least 750 ms or 0.75 seconds through the Iridium® system. While the linear regression of the minimum data points is clearly a good fit to the data, the average RTT values have a much higher spread. A larger sample size would help to identify the outliers that are negatively impacting the data points. However, the high variation would remain.

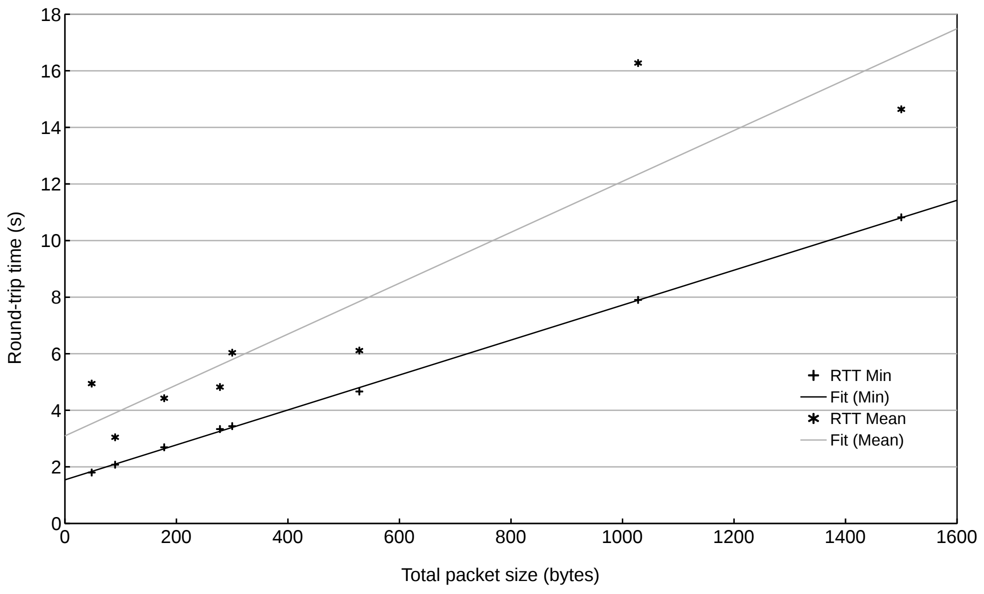

Figure 8. Round-trip propagation delay over an Iridium® link.

When considering all these delay components, the total delay through the network can be quite substantial. If queue size is unregulated, the amount of queuing delay could grow indefinitely and add an excessive amount of delay to network traffic. Queue sizes are not regulated in the current NASA flight system. These excessive delays can have a negative impact on performance, especially when considering the needs of multiple users.

PPP links use a data packet to probe the link and determine if the link is still active. This probe is transmitted like any other data packet and if the response to the probe is not received within a predetermined time frame, then the link is considered to be unresponsive and the call is terminated. NASA currently sets this limit to 60 seconds. Setting the limit longer risks wasting time when a link has indeed failed. Setting the limit shorter risks terminating an otherwise healthy link, especially when faced with excessive or unexpected delays.

Delay is an important factor for many reliable data transmission protocols such as TCP, which measures the delay on a network path and sets timeout values based off of these measurements. These timeouts are used to trigger the retransmission of data in the absence of other feedback from the network. An unregulated queue over Iridium® could inflate these timeouts to be minutes long. These long timeouts would cause exceedingly long delays for end users or cause TCP to fail completely.

Delay is also an important factor for interactive or time-sensitive network traffic. Excessive delay can make data unusable to an application when content is only useful if received within a certain time period of being sent.

4.1.4 User Intervention

Multiple pieces of the current system require manual attention from a user in charge of setting up and maintaining the MLPPP bundle.

Operators recognize the importance of not overwhelming the Iridium® links with too much UDP traffic. Thus, rate limiting is tuned manually prior to each mission to control the flow of UDP traffic. If a rate is picked that is too low, then operators are not making use of all of the available capacity for a given flight. If a rate is set too high, then UDP traffic can overwhelm the MLPPP bundle, delay other traffic, or even cause Iridium® links to fail. Since Iridium® links come and go over time, the amount of capacity available is variable during each flight. Either conservative throttling is used or users must constantly monitor the number of available Iridium® links and adjust their throttling rate accordingly. Neither strategy is ideal as either resources are being wasted or logistical overhead is increased. If a system is tuned for optimal performance and a link fails, the transmission rate must be reduced manually and quickly. If it is not, UDP traffic could overwhelm remaining links and cause them to fail when PPP link probes are not received in time (see section 4.1.3).

A second type of user intervention required on flights is due to the implementation of MLPPP in the Linux kernel. Each MLPPP bundle has what is termed a "master link." Its purpose is to create an MLPPP bundle and manage the addition and removal of PPP links to the bundle. In Linux, the first PPP link that creates an MLPPP bundle will become the master link for that bundle.

If non-master PPP links fail, they will first be removed from a bundle and the overall capacity of the MLPPP bundle will shrink. A removed link can attempt to reestablish itself and, if successful, a reestablished PPP link can be added back into the bundle by the master link. Once part of the bundle, the overall capacity available increases to reflect the total number of PPP links in the bundle.

If a master link fails, its capacity is also removed from an MLPPP bundle, but master links will not attempt to reestablish PPP connectivity. Instead, failed master links will give up their capacity and instead take on a management role for the bundle. As long as at least one PPP link remains that has available capacity in the bundle, the failed master link will never attempt to reclaim its lost capacity. In a system with N links of capacity C, the overall maximum capacity is reduced from $N * C$ to $(N - 1) * C$ when a master link fails. For example, in a system with four Iridium® links, the maximum capacity available is 9.6 kbit/s. Losing the master link limits the maximum capacity to 7.2 kbit/s. To return to a situation where all of the available capacity can be used, either all of the remaining PPP links must fail on their own or a user must choose to tear down the remaining links manually and then reestablish the entire MLPPP bundle from scratch. Neither situation is ideal. In the former, there is a capacity penalty for an indeterminate amount of time. In the latter, tearing down and reestablishing the MLPPP bundle is willfully creating a period with no communication channel to the ground.

Figure 6 shows a visual example of a master link failing. Note that after recovering, only the remaining three links will be used. In this case, by not tearing down the remaining links and resetting the bundle, there is a penalty of 25 percent of the original capacity going forward.

5 Component Improvements

In addition to exploring the use of MPTCP in order to mitigate the observed problems discussed in section 4, changes to the overall system in order to fully support a flight are also identified. This section will detail some component-level changes that were made and how these changes address problems in the existing systems. Section 6 will detail additional system components that enable full support of a flight.

5.1 Eliminate Fate Sharing

Given that Iridium® links change their signal strength often and can fail at any time, fate sharing is a fundamental hurdle that needs to be overcome when shifting away from the MLPPP-based flight system. The first step in designing a new system is to remove MLPPP from the system completely and to treat PPP links as individual network paths. Rather than fragmenting packets across multiple links at once, full packets are sent down each path.

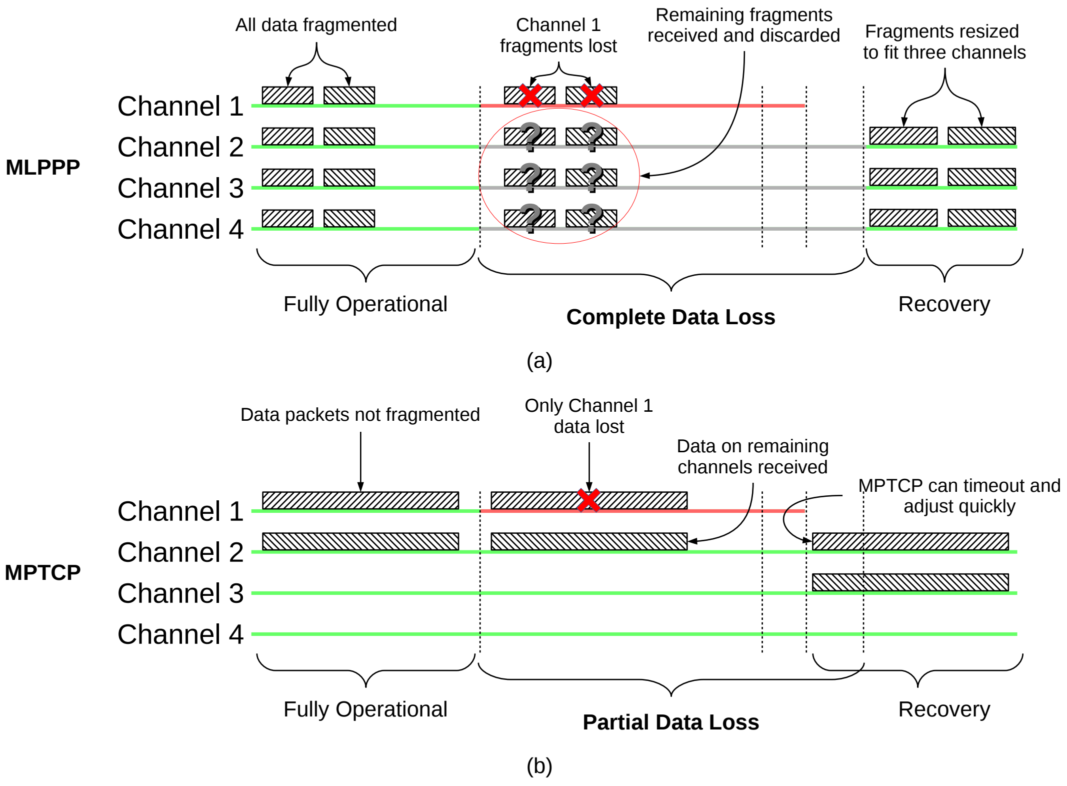

Figure 9. Transmission of data and (a) fate sharing with MLPPP and its (b) elimination with MPTCP.

Figure 9 demonstrates how multiple packets are sent in the old and new systems. In the MLPPP-based system, packets will be fragmented and sent in order across the bundle. In the new system, full packets will be sent at the same time using the different PPP links. Note that if a loss happens on one link, the remaining packets will successfully be received and will still be usable in the new setup.

MPTCP is capable of handling data flows across multiple network paths simultaneously, and each PPP link has been configured to be a separate path in this setup. Rather than fragmenting packets across each link at the link layer, MPTCP takes care of splitting data at the transport layer and sends full packets across each of the PPP links. MPTCP will also keep one control loop per link. Thus, loss or delay on a single link will not have a direct effect on the remaining data flows. With proper configuration, MPTCP will automatically adjust its data flows to use only available links and will shift traffic away from failing or failed links. Failures can be detected even prior to the operating system detecting and removing the link, allowing recovery to happen quickly using the remaining healthy links.

By leveraging MPTCP, fate sharing is eliminated and the blackout periods of complete data loss present with MLPPP and TCP are avoided. In a setup with separate PPP paths, UDP traffic can simply be sent across any available PPP link. The solution detailed in section 6.1 uses a simple round-robin sending strategy to handle UDP traffic.

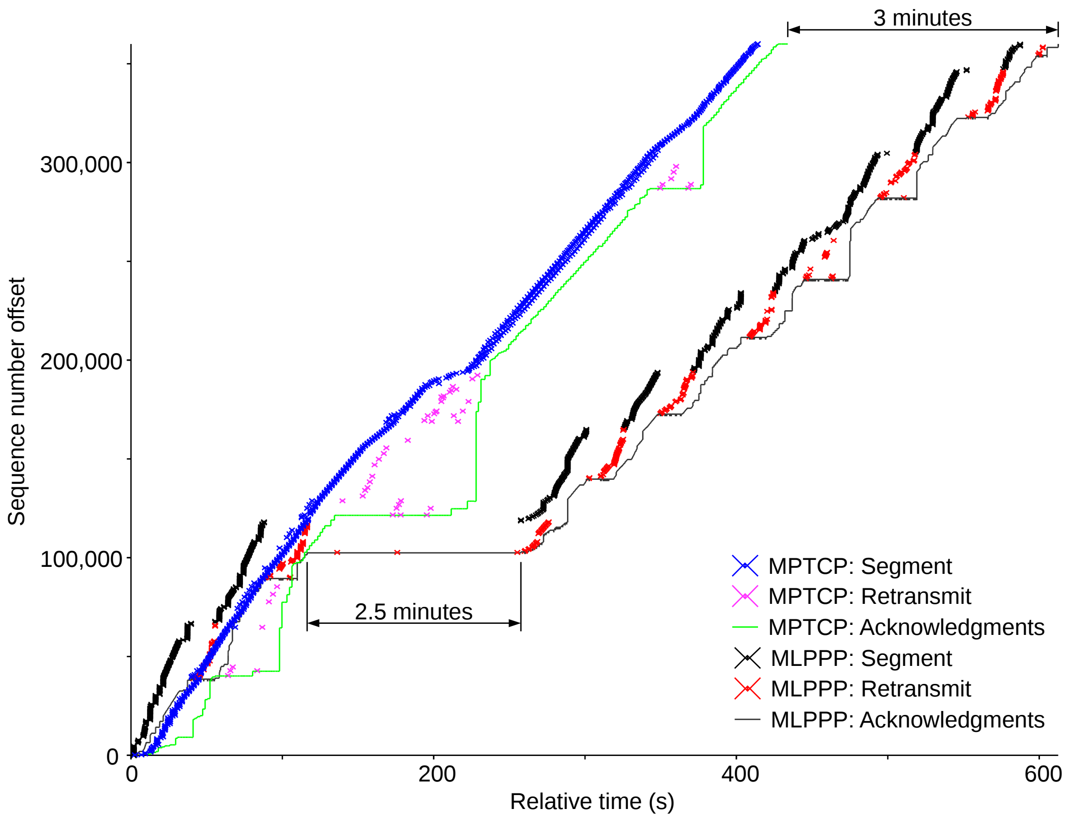

Figure 10. MLPPP and MPTCP single link loss over Iridium®.

Figure 10 contrasts the impact of losing a single Iridium® link on both the MLPPP-based and MPTCP-based systems. In this test, the aircraft acted as a sender, transmitting a single, fixed-sized file to the ground repeatedly. The figure superimposes two sender-side traces taken independently, and thus the axes show relative figures for both time and sequence space. Both traces were over actual Iridium® modems in the engineering testbed. The MPTCP trace was taken with the testbed in the experimental configuration. The MLPPP trace was taken with the testbed in a configuration that matched the current production configuration for Earth Science missions.

For simplicity, all MPTCP subflows are colored blue in figure 10. Likewise, any MPTCP retransmission, regardless of subflow, is colored in magenta. The MPTCP acknowledgment line is green and reflects the acknowledgments over the combined sequence space. The black segments in the figure represent a normal TCP flow in the MLPPP configuration. Retransmissions for this flow are colored in red, and the acknowledgment line is gray.

The loss event for both traces happens shortly after 1 minute. At that time, one link fails and remains unavailable for the remainder of the transfer.

Note that in figure 10 losing a link results in a blackout period lasting nearly 2.5 minutes. Two factors contribute to this long blackout period. The first and direct cause is MLPPP's behavior of complete data loss as the link loss is being detected and eventually repaired. During this period, TCP is timing out and unable to get any data across the MLPPP bundle. The second factor comes as part of TCP's normal behavior to backoff retransmissions with each failed attempt. As a result, even though MLPPP has recovered shortly after TCP's second retransmission, TCP does not detect the recovery until the third attempt, adding over a minute to the recovery time.

Contrast this behavior with MPTCP's performance in figure 10. During the loss period, useful segments are still able to be received on the remaining links and retransmissions begin to migrate from the failing link to healthy ones. The large jump in the MPTCP acknowledgment line seen 230 seconds into the trace is a good visualization of the prompt recovery. MPTCP was able to leverage the data sent on healthy links and continue to transmit new information as it patched the losses. The overall data flow is not disrupted and instead gracefully adapts to the new overall capacity. The net result is that the MPTCP flow finishes 3 minutes quicker than the TCP flow over MLPPP.

5.2 Improve Queues

Splitting the MLPPP bundle into individual PPP links means that the system changes its queuing strategy from a single FIFO queue for the MLPPP bundle to one FIFO queue per PPP link. To force multiple data flows and connections to more fairly share space in the PPP queues, a queuing strategy aimed at making sure flows in the individual PPP queues are never starved out from sending data by any other connection was employed.

Each PPP link installs a Stochastic Fairness Queuing (SFQ) discipline that enforces a round-robin sending pattern among all connections with outstanding traffic waiting to be sent across a PPP link. For example, if two TCP connections, t_a and t_b have data waiting to be sent across a PPP link, one packet from each connection will be sent in an alternating fashion: $t_a t_b t_a t_b$.... If an instrument then begins sending UDP traffic, the SFQ will make sure that the UDP traffic u_a gets a slot in the round-robin sending as well: $t_a t_b u_a t_a t_b u_a \ldots$.

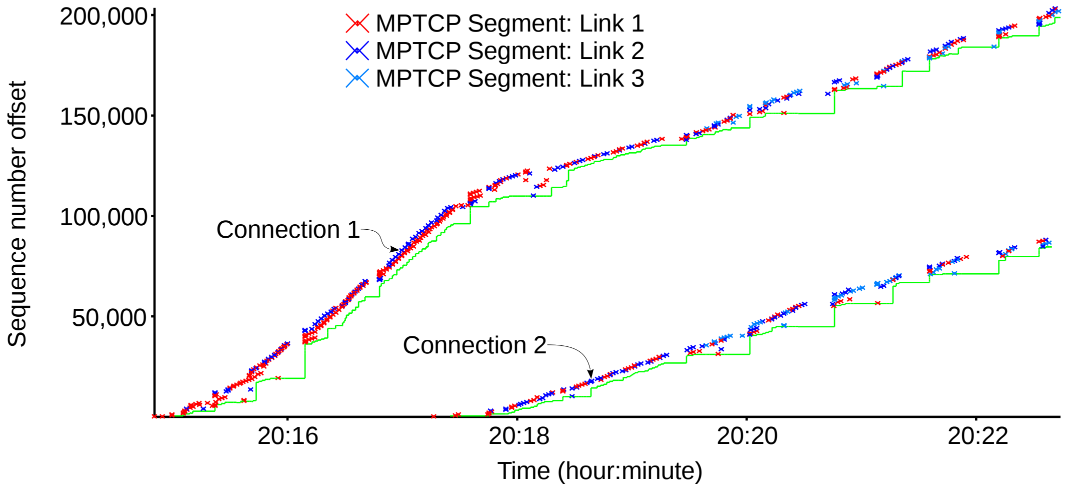

Figure 11. Two MPTCP transfers using Stochastic Fairness Queuing (SFQ).

Figure 11 shows an example of the SFQ rules in the system. At the start of the figure, a single TCP connection is using up all of the capacity in the system. Even with a large amount of data already enqueued and waiting to be sent across the PPP link for the first connection, when the second TCP connection is started, it is able to get its fair share of capacity quickly because of the SFQ rules. This is visualized easily by observing the slopes of both flows. Without these rules, the second TCP connection would have had to wait in the single FIFO queue and would face a delayed startup and reduced capacity as discussed in Section 4.1.2.

5.3 Limit Buffer and Delay Impact

There are two locations where buffers have the biggest impact when using Iridium® for data communications. Building up a large buffer in either location will add delay to all communication across a PPP link. The location of these buffers dictates the amount of control over assigning priorities to traffic and making smarter routing decisions.

The first location to build up a queue is marked "Point A" in figure 12. This location is the PPP interface on the aircraft that controls data flow into the Iridium® modems. If data arrives from instruments and users on the aircraft at a rate faster than Iridium® will handle, then a queue will build up for the PPP interface. In this case, SFQ rules can be employed as detailed in section 5.2 to prevent flows from being starved out completely, but each flow will still be delayed by its own previously enqueued packets.

The second location where a queue may build is marked "Point C" in figure 12. This location is located at the Iridium® ground station, likely at the point where analog phone communication is converted to and from the format needed to send data across the Iridium® satellite constellation. When a call is placed from the

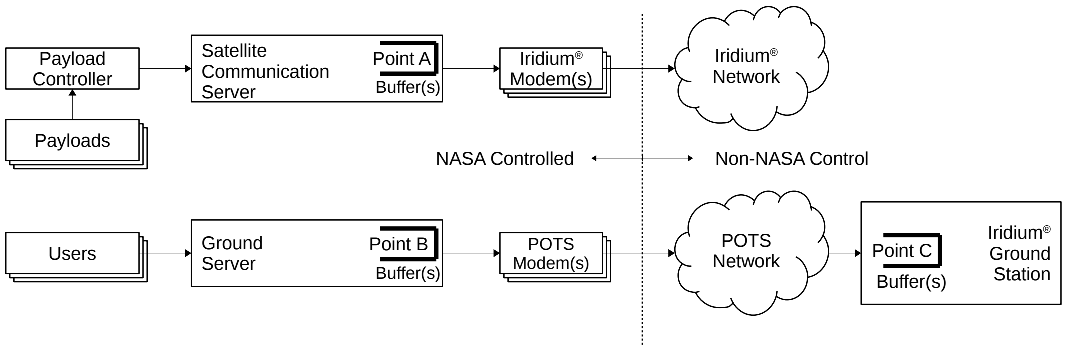

Figure 12. Key buffer and queue locations.

aircraft to the modems on the ground, the Iridium® ground station will split the call into two pieces, the satellite component and the phone component. This split is significant, as the dial-up phone communication occurs at a much higher rate than Iridium®. Based on observed behavior, the Iridium® ground station uses a simple, deep FIFO queue to handle traffic, and a large amount of data can be buffered at the Iridium® ground station, causing significant queuing delays. Since Point C is out of NASA's direct control, additional queuing rules are placed at Point B just prior to transmitting data over the phone lines.

The solution for preventing long queuing delays is the same in both cases. The first step is to deploy rate-limiting rules for PPP interfaces on both the aircraft (Point A) and the ground station (Point B) in order to throttle traffic to match the Iridium® links. The second component is limiting the queue depth for individual flows in the SFQ rules.

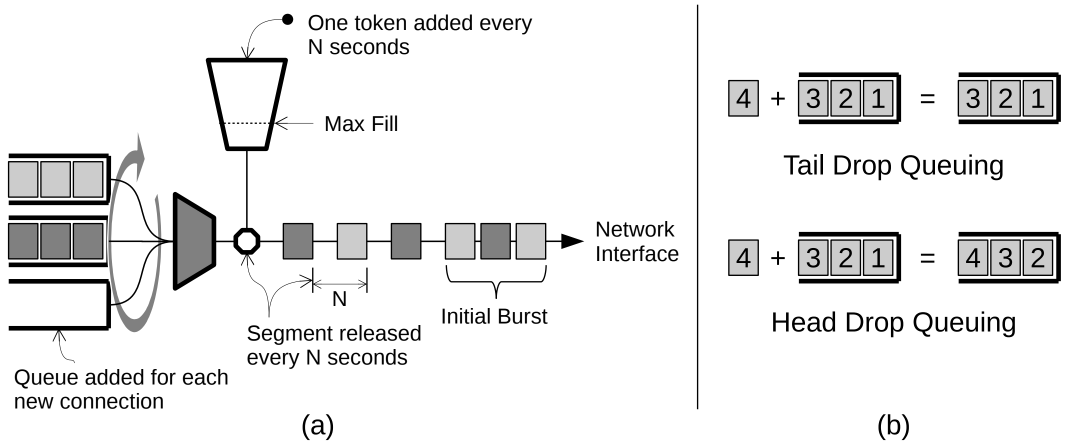

Figure 13. Advanced queuing details. (a) Combined round-robin SFQ solution with token bucket throttling, (b) Tail vs. head drop queuing.

Figure 13(a) illustrates the combined queue and link management solution. A round-robin SFQ is configured with a maximum queue depth of three outstanding packets. The SFQ is also configured to perform head-drop queuing instead of traditional tail-drop queuing. Figure 13(b) illustrates the difference in the two dropping methodologies. With traditional, tail-drop queuing, a new packet is permanently discarded if it arrives when the queue is full. With head-drop queuing, the new packet is saved and the oldest item in the queue is discarded, resulting in only the most recent packets for each flow being saved.

The second component consists of a traditional token bucket to rate limit segments onto the network interface. The token bucket algorithm uses a conceptual bucket to store tokens up to some user-specified maximum. This allows for an initial burst of traffic, followed by a paced stream of segments. In the proposed solution, the burst was limited to 1500 bytes and tokens were generated at 2500 bits/s, a rate just slightly above Iridium®'s expected 2400 bits/s.

This combined solution offers many benefits to the final system. The SFQ ensures that individual flows are prevented from being starved out by competing flows. In addition, a flow generating a large amount of data will only ever get the most recent data through. This has positive impacts to the UDP data transmitted in the system and actually helps TCP to recover more quickly. Limiting the number of packets enqueued per flow caps the additional queuing delay that can be introduced by each connection.

Finally, as the queuing delay is a function of the number of bytes in queue and not the number of packets, the MTU of each PPP link is limited to 550 bytes. This means that a system with one TCP flow with three outstanding segments of full size will experience a maximum of 5.5 seconds of queuing delay. Recall that the default MTU is 1500 bytes, which would lead to one segment generating a similar delay.

5.4 System Scalability

All of the previous improvements in this section contribute to the scalability of the flight communication system.

Section 4.1.1 mentions the scalability problems that are caused by fate sharing when using MLPPP over lossy links. Eliminating fate sharing as mentioned in Section 5.1 improves system scalability. Adding an Iridium® link to the redesigned system no longer increases the risk of having a link failure and causing a blackout period of the MLPPP bundle. Rather, new links can fail and will only cause loss on that individual link. Blackout periods are never a possibility in the new system unless all links fail independently. In that case, no system could possibly provide a communication channel as no satellite links are available for use.

The improvements in Section 5.2 and Section 5.3 allow for fairer handling of a larger number of traffic flows over Iridium®, while reducing the risk of overwhelming the Iridium® links with too much traffic. If new instruments or applications are added to missions, the new traffic flows will be handled gracefully without the need to reconfigure the system.

To address the problems mentioned in Section 4.1.4, the fair queuing and buffer limiting rules have been implemented to automatically adjust themselves as Iridium® links come and go over time. No user intervention is required when the system gains or loses a link and the system will never be in danger of failing due to overwhelming Iridium® with too much traffic. Automation also reduces system management overhead in flight as links no longer have to be manually monitored. Scalability improves as adding Iridium® links to the system will not add any management overhead while in flight.

6 Flight Support and Insights

While understanding which changes at a component level are useful, additional considerations at the system level are needed in order to support mission flights. These flights generate particular types of traffic and have certain preferences and requirements related to how their traffic should be handled. In addition to making changes to fix the underlying problems associated with MLPPP, particular sets of parameters must be chosen for the changes in order to successfully support NASA flights.

This effort supported a 13-hour test flight aboard NASA's Douglas DC–8 aircraft in November 2016 where the MPTCP-based system was used to provide the communication channel over Iridium® during Antarctic flights. This report will detail the support requirements for this flight, explain the total flight system, and present some of the observed results and successes of the test flight.

6.1 Handling UDP Traffic

The NASA flight systems currently transmit UDP data packets from the various payloads to the ground. The payloads are not given a hard sending rate for each mission. Rather, a custom software application is run on the payload controller and manually configured to consume and retransmit the newest packets from various UDP streams at a rate that is configured manually. Since the system is manually tuned, it cannot adjust to the variable Iridium® capacity as links drop out or recover.

The use of MPTCP alters the behavior of TCP, but does not provide a solution for routing UDP packets across the now individual PPP links. Therefore, a solution was required that would route all inbound UDP traffic arriving from the payload controller and transmit them across one of the available PPP links.

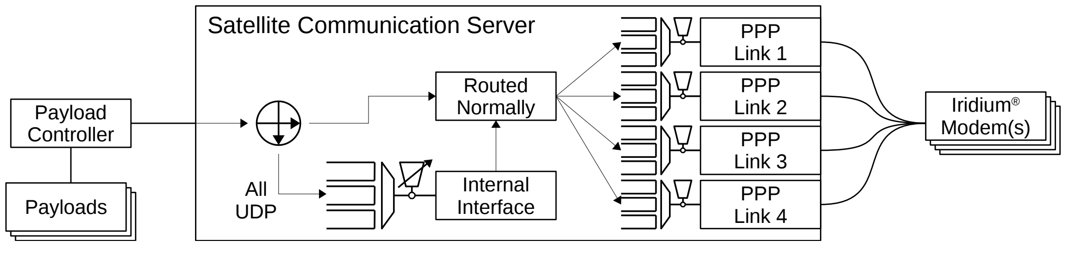

Figure 14. Handling UDP in a fair manner using an additional queue.

Figure 14 shows the final routing solution that handles both TCP and UDP in a fair manner. An additional internal interface is implemented with its own SFQ and token bucket queuing rules. This internal interface is programmed to route data to the next available PPP interface for each segment that the token bucket releases. PPP interfaces that are down are skipped, such that only active interfaces are used.

In addition, the token bucket rate is variable, adjusted automatically as the number of PPP interfaces changes. The rate used is 2500 bits times the number of active links per second. Thus, if all four links were active, the rate would be 10000 bits per second.

This solution effectively bins all UDP traffic into a single competitor against TCP flows and prevents the presence of numerous UDP sources from unfairly suppressing the few other TCP services needed for flight tests. The reuse of the SFQ and token bucket solution also replaces the functionality of the custom software used in the payload controller. The manual selection of rates is no longer necessary, and the head drop nature of the SFQ also provides the same transmission of only the newest data for each UDP source[2]. While the routing of the internal interface is simple, future work could involve the development of more complicated UDP sending strategies, including prioritization of packets.

6.2 Flight Details

There are three primary types of traffic that need to be handled in order to support the flight on the DC–8:

Hypertext Transfer Protocol (HTTP) Users on the aircraft sometimes require information and images from external sources to be displayed for researchers on board. Users expect the ability to issue a web request using HTTP and for the requested item to be successfully retrieved over Iridium®. For example, maps displaying the current weather information in the flight area may be retrieved and displayed for the flight crew and researchers on board the aircraft.

Internet Relay Chat (IRC) While some researchers are on board the aircraft, other users and support staff are located on the ground. These two groups of people expect to be able to communicate using IRC over Iridium®. During the test flight, this communication channel was also leveraged to answer questions for researchers on the aircraft as well as ask for feedback about system performance during the flight.

Instrument Traffic Instruments on the DC–8 will generate status packets that are destined for a server located on the ground. These status packets are helpful to users on the ground who would like a way to monitor instrument statistics in a timely manner.

Out of these three types of traffic, HTTP and IRC both require a reliable transfer of data and are sent using MPTCP in the system. Instrument traffic is sent using UDP.

MPTCP connections require at least one host to be aware of the existence of multiple network interfaces, and therefore multiple paths that can be leveraged for communication. Figure 15 illustrates the requirement to leverage MPTCP if all hosts were MPTCP capable. In this setup, paths (a), (b), and (c) can all leverage MPTCP

[2]Recall that each UDP source will get its own buffer in the SFQ as they originate from unique address port 4-tuple.

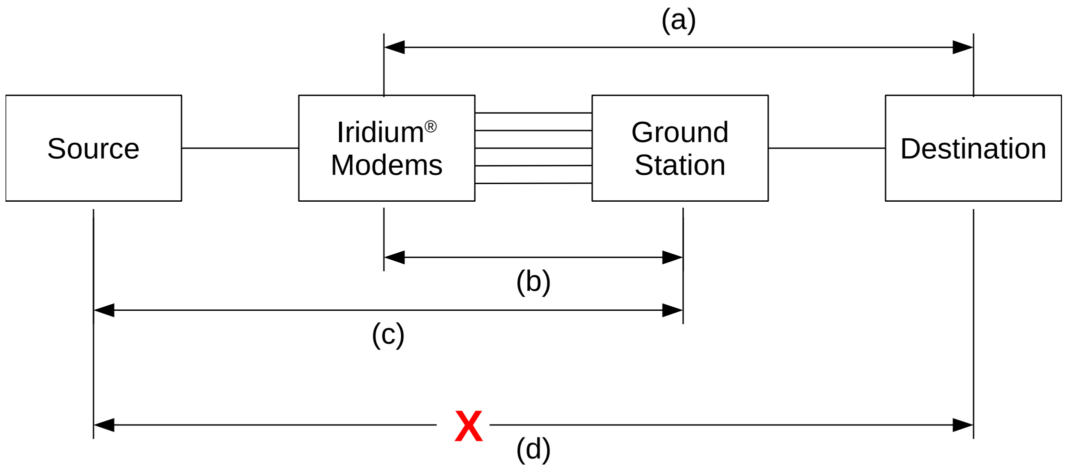

Figure 15. MPTCP endpoint requirements.

properly. However, path (d) cannot, as neither endpoint knows the number of paths available. Furthermore, if the source or destination were not MPTCP capable, then both (a) and (c) are no longer viable options for MPTCP as both endpoints need to be MPTCP capable.[3]

In this setup, there are two machines aware of the number of network paths; the two machines that are connected at both ends of the Iridium® link. First, MPTCP is installed and configured on these two machines so that any reliable connection between these two machines will be performed over MPTCP using all of the available PPP Iridium® links. Next, HTTP proxies are installed on these machines, and the aircraft proxy is configured to forward any request it receives from users on the aircraft to the proxy on the ground. The ground proxy will then go out to the rest of the network and retrieve any requested objects. IRC connectivity is provided in a similar manner. IRC servers are created on the machines at either end of the Iridium® link and configured so that the aircraft server will establish a link with the ground server. The ground server then links up with a network of IRC servers that provide connectivity with users on the ground. The IRC servers establish and maintain their link using MPTCP over Iridium®.

Note that while MPTCP traffic is used over Iridium®, HTTP requests between users on the aircraft and the onboard HTTP proxy are made over regular TCP. Likewise, the ground proxy retrieves objects over regular TCP, but then sends the objects to the aircraft using MPTCP over Iridium®. The same is true for IRC traffic. Figure 16 illustrates the flow of data between TCP and MPTCP that occurs for HTTP and IRC traffic. The use of proxies in this manner is due to the limited access to end hosts on the aircraft and ground. Since MPTCP requires hosts to be MPTCP-aware and have special code installed, the design space was limited to a solution that only involved machines that could be accessed and updated directly. Further, MPTCP can only make use of paths that at least one end host can detect, and the machines directly connected to Iridium® have the ability to detect how many

[3]Note that intermediate nodes do not need to be able to understand MPTCP.

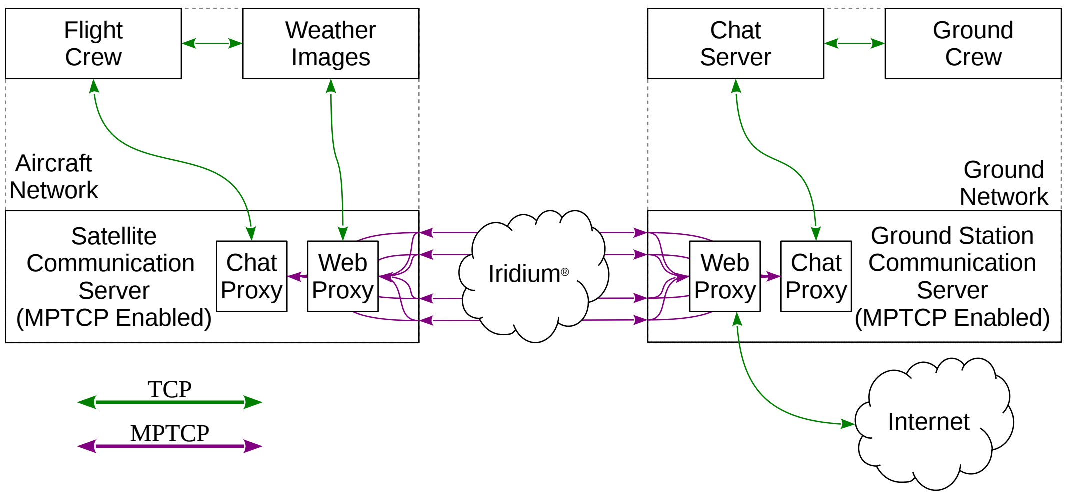

Figure 16. MPTCP and service specific proxies for flight tests.

PPP links are available for communication at any given time.

Instrument traffic is UDP-based and generated at regular intervals throughout the flight. Instruments send their UDP traffic to the satellite communication server connected to the Iridium® links. The UDP packets are then handled as detailed in section 6.1. Once UDP traffic arrives on the ground, packets are sent through a network address translator and then sent toward their final destination.

Other modifications to the system were detailed earlier. In review:

- SFQ rules are installed on individual PPP links as each PPP link is established. The SFQ allows each connection or flow of data to get an equal chance to send data across the link. The SFQ also limits the size of a queue that builds up for each connection or flow to three packets. These packets will be the most recent packets belonging to a flow. Alternating between flows provides fairness among flows and limiting queue size bounds the maximum queuing delay each flow can introduce to the system.

- A token bucket filter rule is installed for individual PPP links that limits the rate data is sent out over the Iridium® link. These rules help prevent large, uncontrollable queues from building up along the network path and limit the additional delay from queuing that traffic experiences.

- Changes to rates and queuing rules are handled automatically by scripts as PPP links come and go. There is no user intervention required to prevent the Iridium® links from being overwhelmed by too much traffic. This improves system scalability as adding modems to the system does not increase the management burden. Furthermore, links do not need to be monitored during flight.

- MPTCP connections maximizes the use of all available PPP links in an automatic way. As PPP links become available, MPTCP connections will begin establishing data flows across the new links. As PPP links fail, MPTCP shifts traffic away from the failing links. As long as an MPTCP connection is able to communicate over at least one link, then a connection will be able to remain active and ongoing. Shifting traffic to and from the various PPP links requires no user intervention. MPTCP allows scalability as additional links will be incorporated into connections automatically.

- Proxy services are configured to connect once PPP links become available without any user intervention. Once proxies establish communication, MPTCP maintains connections as previously described.

All of the system modifications resulted in successful support of HTTP, IRC, and UDP traffic during the 13-hour test flight.

6.2.1 Iridium® Link Quality

Recall from Section 2.2 that Iridium® modems were expected to experience high variance in their signal strength throughout the flight. As signal strength degrades, data calls may be dropped and the number of available PPP links will vary accordingly.

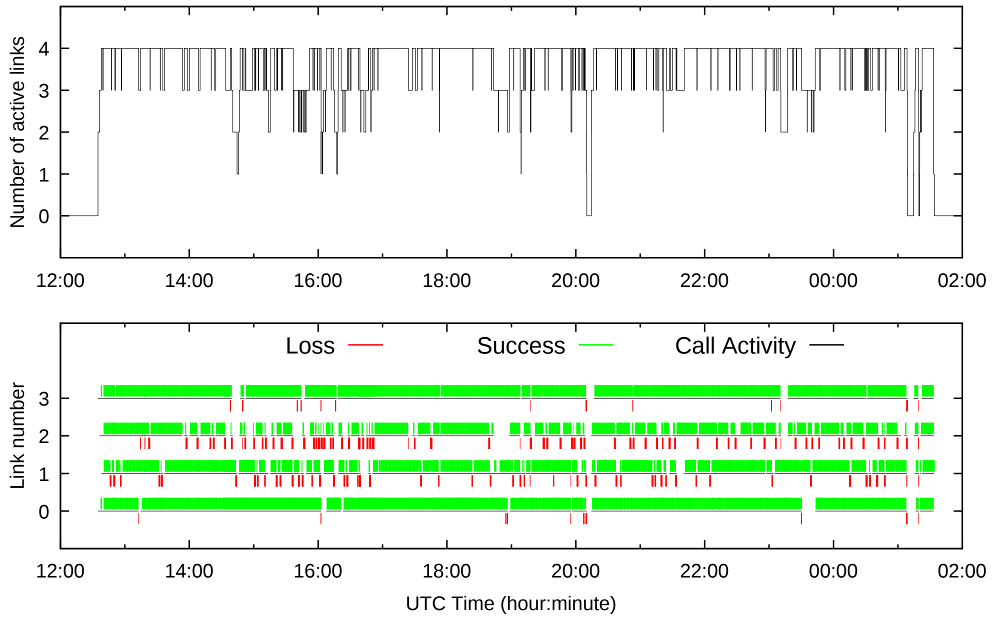

Figure 17. Link transitions, link state, and link activity during the test flight, November 18, 2016.

Figure 17 shows the number of PPP links available during the 13-hour test flight. Note that the total number of links available transitions 25 times per hour on average. The longest time spent in one state was approximately 30 minutes, where all four links were available.

Table 1 shows the total percentage of time spent in a state with a given number of available PPP links. Note that communication between the aircraft and the ground

Table 1. Link state percentages

Number of Active Links	Seconds	Percent
4	35,643	76.26%
3	8,269	17.69%
2	1,969	4.21%
1	235	0.50%
0	624	1.34%

is unavailable for 1.34 percent of the flight when all of the PPP links fail at the same time. The rest of the time there is at least one PPP link available with just under 94 percent of the time spent in a state where three or more links are available.

On the MLPPP-based system, fate sharing conservatively adds 1 minute worth of time where no data is able to make it successfully between the aircraft and ground every time a PPP link fails. As there were 163 decreases in the number of available links, there would have been an additional 2.7 hours of link unavailability on the flight due to MLPPP fate sharing. When compared to the 10 minutes of observed link unavailability with the modified system, the importance of removing MLPPP fate sharing from NASA flights in order to maintain some level of connectivity between the aircraft and the ground becomes very clear.

6.2.2 IRC Connectivity

The main goal of IRC on NASA flights is for the IRC server on the aircraft to stay connected to the IRC server on the ground station. Staying connected maintains a line of text-based communication for users on the aircraft and ground. If the IRC servers are not connected, the aircraft server will attempt to connect to the configured server on the ground. If unsuccessful, the aircraft server will pause before attempting to connect again. Once successful, the IRC connection is maintained using the available Iridium® links. All chat messages from the ground or aircraft networks will be sent across the IRC connection.

Previously, using the MLPPP setup, end users noted the frequent dropout of IRC services. These dropouts would occur naturally and could also be triggered by attempting to utilize the PPP link too heavily. End users would complain about losing IRC when trying to fetch data over HTTP. The former issue is most likely a result of fate-sharing, and the latter an issue with queue depths and fairness.

During the 13-hour flight, the IRC server using MPTCP required only four separate connections even with 163 separate Iridium® link failures. Note that some disconnects are to be expected as there are periods of time where no Iridium® links are available, which was the scenario for two of the disconnects. One connection was terminated due to restarting the IRC server in order to adjust a server configuration. An interesting observation from this test came from the second disconnection. In this instance, the initial MPTCP negotiation failed due to losses over Iridium®, and so MPTCP fell back to using regular TCP due to lack of response. This is the default behavior of MPTCP as this practice would allow a MPTCP host to communicate if MPTCP options were being dropped by a security device such as a firewall. Since the connection had reverted to regular TCP, only one link was being used even though multiple links were available. The connection survived until the one link it was using finally failed. For future tests and flights, it is recommended that MPTCP be configured to be more aggressive and not fall back due to a lack of response, but rather only revert to TCP if the other side explicitly responds without acknowledging the MPTCP option. This alteration is safe for this system, as the MPTCP endpoints are on either side of a point-to-point link.

Another observation from these tests was an additional IRC connection starting while an existing IRC connection was already established. These connections were very cyclical and clearly due to some configuration parameters in the IRC server. The extra connections generate a small amount of traffic, but even small amounts of traffic are significant over an Iridium® system. The configuration changes needed to suppress these extra connections was identified after the flight test, and configuration patches were submitted as a recommendation for all future Earth Science flights.

During the flight, an experiment was conducted where extra HTTP traffic was generated to stress the Iridium® system and test to see if IRC would remain active. The test succeeded and IRC remained online, demonstrating the fairness of the queuing rules for the system. Support for the MPTCP build from Glenn also went smoothly over IRC.

6.2.3 HTTP Requests

HTTP requests were issued during the test flight to retrieve weather map images to display on the aircraft. Over a period of approximately 2 hours, 33 back-to-back HTTP requests were issued from the aircraft. All of the requested images were successfully retrieved using MPTCP connections over Iridium®. Images ranged in size from 107 to 112 kB. Times to retrieve the images ranged from 124 to 425 seconds, with a mean retrieval time of 215 seconds.

The users in charge of retrieving the images reported a positive experience regarding the HTTP requests and gave verbal feedback that connection speeds proceeded at rates proportional to the number of available Iridium® links.

It should be noted that a device on the aircraft was also generating unexpected HTTP requests separate from the valid image requests. The unexpected requests were for objects that could not be retrieved and these requests ultimately failed. However, the unexpected requests still created MPTCP connections over Iridium® that transferred just under 4 KB of data per object when the HTTP proxy transmitted a web page to display a failure message for each object. Thus, these web pages used a small portion of the available capacity each time a request was made. Even with these extra connections using part of the available capacity, valid HTTP requests were able to proceed normally and succeed in a timely manner due to the fair queuing rules.

6.2.4 UDP Instruments

The DC–8 test flight had one UDP instrument sending status packets from the aircraft to the ground. The instrument generated one packet every 5 seconds over the course of the flight. The time stamp of each packet was logged as UDP packets from the instrument on the aircraft were received at the ground side of the Iridium® links. Then the interarrival times between each received packet were calculated. Calculating this value provides insight into the amount of time researchers on the ground would have to wait for fresh data from an instrument to become available.

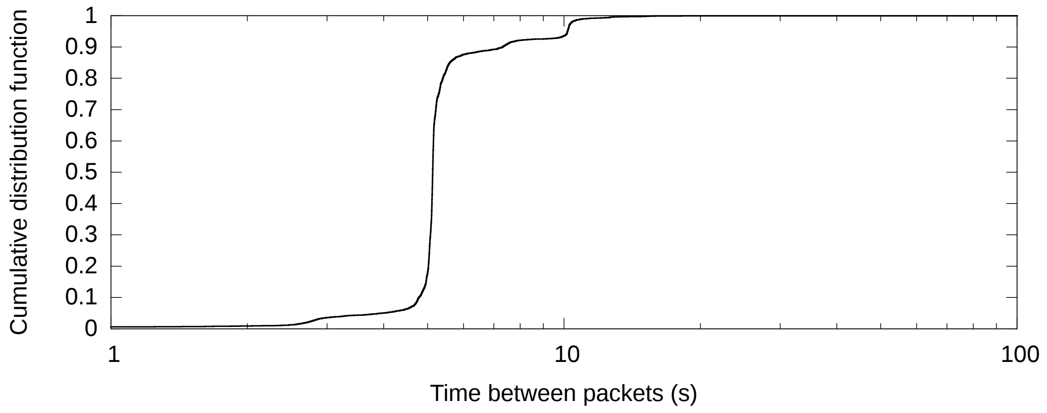

Figure 18. Interarrival times of UDP traffic.

Figure 18 shows the cumulative distribution of interarrival times for instrument packets over the course of the flight. Approximately 99 percent of UDP packets arrive within 11 seconds of each other and the median interarrival time is 5.1 seconds. While not shown on the plot, the maximum time between two UDP packets was 7.8 minutes. This corresponds to a total link outage where no Iridium® links were available for sending. Interarrival times of longer than 5 seconds are either caused by delay from Iridium®, delay from competing traffic over Iridium®, or from packet drops.

Note that figure 18 shows some packet arrivals taking longer than the expected value of 5 seconds. However, there are also packets that arrive within 5 seconds of each other. This is expected behavior as a packet that is delayed will increase the amount of time between itself and the previous packet, but the amount of time between itself and the next packet decreases if the next packet arrives on time. For example, suppose packets are sent at times t_0, t_5, and t_{10} and the packets are received at times t_2, t_7, and t_{12}. This would represent arrivals without any unexpected delay and the interarrival times would be 5 seconds in both cases. If the middle packet were delayed and arrived instead at time t_9, then the first interarrival time would be 7 seconds while the second interarrival time would be 3 seconds. Due to the round-robin sending, it may be the case that two packets sent across different Iridium® links 5 seconds apart may arrive at the ground at around the same time due to delay on one Iridium® link, but not on the other.

Users on the test flight did not report any issues with UDP instrument traffic. These results show UDP performing well in sharing resources with the active IRC and HTTP TCP flows used on the flight.

7 Additional Deliverables

A set of tools was developed to assist in the analysis of MLPPP and MPTCP network traffic. This tooling will be made available for internal use at NASA and a subset of tools will be made available for public release.

7.1 PPP Tooling

PPP links can be configured to log all data sent and received over a modem by using the PPP record option. Certain tooling exists for processing files generated by the use of this option in Linux, but these tools provide limited capabilities and mostly enable a user to display individual packets or fragments sent across a PPP link. To assist in analysis of PPP and MLPPP traffic and support more in-depth analysis, several tools were developed.

The first tool is a graphical user interface, aptly named *ppp_gui*, that can monitor a PPP link and display information about the amount of data sent and received over the last 30 seconds. Additionally, the last several round-trip times for link control traffic are displayed along with a status bar that indicates if a PPP link appears to be healthy, failing, or failed. This tool is meant to assist researchers on flights and users on the ground by providing a visual representation of Iridium® link states during flight. Researchers can quickly see how many links are currently available while in flight. For active links, users can check for abnormalities or signs of an impending link failure by monitoring the displayed link control traffic statistics.

A second tool was developed to compare PPP record files taken at either end of a PPP link. This tool allows tracking of link uptime, propagation delay, and frame losses for a PPP link.

Finally, a tool for analyzing MLPPP bundles was developed. This tool provides a way to track MLPPP bundle state by tracking fragmentation, packet reassembly, and detecting when loss leads to data being discarded on a bundle. The negative impact of fate sharing on an MLPPP bundle with failing links can be better understood with this tool.

7.2 MPTCP Tooling

MPTCP support has been added to certain standard networking tools like Wireshark[4] and tcpdump[5], but this support has largely focused on packet parsing with some minimal understanding of MPTCP packet semantics. The tooling in this work has a larger focus on MPTCP connection analysis and other tools that improve the process of working with MPTCP network packet traces. The MPTCP tooling suite is slated for public release and will be available at https://github.com/nasa/multipath-tcp-tools.

[4]https://www.wireshark.org/

[5]https://www.tcpdump.org/

The first tool, *mptcpplot*, enables users to visualize MPTCP connections by generating time sequence graphs for MPTCP connections in a similar manner that the tool *tcptrace* does for TCP-based connections. Users are presented with a graphical representation of data and acknowledgments being sent across an MPTCP connection and its one or more subflows. As such, *mptcpplot* was a valuable asset that allowed for the study and understanding of MPTCP connection performance and behaviors each time a system-level change was introduced and the impact of the change on network traffic.

The tool *mptcpsplit* enables users to output a list of MPTCP connections contained in a pcap file. The tool is also capable of generating a separate pcap file containing only the packets of a single MPTCP connection. For large pcap files containing many MPTCP connections, this tool allows users to quickly separate out certain connections so that future analysis of an individual connection can be performed quickly.

The last part of the MPTCP tooling suite is the program *mptcpcrunch*. This program lists out statistics about MPTCP connections and their underlying subflows. This tool provides information about both directions of a connection and enables users to see how data is being scheduled across individual subflows as well as to calculate performance metrics on a per subflow basis.

The public release of the MPTCP tooling suite also contains Linux manual pages for each tool mentioned in this section. These manuals contain the full documentation for each tool and detail the exact capabilities of the tools.

7.3 Flight System Configurations

To ease the deployment of MPTCP on NASA flights, scripts were created that will install MPTCP kernels and the required software for flight support on aircraft and ground station machines attached to Iridium®. The installation scripts are on a Universal Serial Bus (USB) memory stick and only require a user to plug the stick into a Linux computer and run a single command. If the script is successful, the computer will be equipped with MPTCP-based communication over Iridium® and will have all of the software required to implement the system described in Section 6.

There are two different sets of installation scripts that handle installation on the ground station and aircraft machines separately. Further, the USB installation script may not be compatible with all of the current NASA flight systems due to hardware limitations. In these cases, custom installation scripts can be developed using the current USB installation scripts as a starting point.

As these scripts are tightly coupled to supporting flight communication on mission flights, they are largely of use only within NASA or other agencies that leverage similar scientific research aircraft.

8 Conclusions

The ability to communicate reliably and effectively with aircraft is a critical component of all airborne science missions. Most flights leverage a form of satellite communication. A commonly used commercial solution is Iridium®, which provides global coverage including the Arctic or Antarctic regions. The capacity of a single Iridium® link is small, and multiple links are used to increase capacity. The reliability of any single Iridium® channel is highly volatile and subject to frequent disconnections. The science community currently uses MLPPP, a networking protocol capable of bonding and handling transmission over multiple links. MLPPP relies on links being robust, and as such the transient nature of Iridium® creates a poor environment where communication blackouts occur frequently. These blackouts are exacerbated by connection oriented protocols such as TCP.

This work describes the use of a new communication protocol, MPTCP, to replace MLPPP. It shows how MPTCP completely eliminates the blackout problem for TCP and instead enables connections to be maintained over many link transitions. The result is long-lived, healthy connections that would have floundered or been terminated with MLPPP. These connections are also more responsive and dynamically able to leverage the current network capacity of the active links. As a result, TCP connections are able to complete a transfer much faster than those over MLPPP. The amount of improvement varies depending on the types of loss events. With a single loss event, MPTCP will finish several minutes faster. A string of losses can easily result in MLPPP failing completely, while MPTCP will still succeed.

In order to build a solution for future airborne science missions, this work outlines a complete communication system for both TCP and UDP, which incorporates the speed limitations of Iridium® and attempts to provide a fair share of resources to payloads and operators. This system-level solution creates a reliable and fair network architecture. Furthermore, the solution automatically scales to the number of users and to the amount of network traffic, with new MPTCP flows quickly obtaining a fair share of the available capacity.

In order to ease the deployment of MPTCP into future airborne science missions, the proposed system only impacts two existing flight components and relies on service specific proxies to provide end-to-end solutions to payload operators and ground users. This work shows that the proxied solution works well for MPTCP, and does not negatively impact the overall performance of the system. Additionally, the two systems impacted by the new software are on either end of the Iridium® link and under direct control of the flight operators. As a result, deployment does not need to rely on all payload operators upgrading.

The solution detailed in this work was configured with a focus on Iridium®, however the solution and architecture can be used wherever multiple links are leveraged. This includes systems that leverage multiple satellite communication providers, or a mix of communication solutions, such as optical, satellite, wired, or wireless. Additional research would be needed for solutions that combine links with vastly different performance capabilities.

This work shows that the proposed system is capable of being fair to multiple TCP and UDP flows. However, there may be times when it is desirable to allow some classes of data to be prioritized over others, especially in cases where resources are limited. In addition to prioritization of data, MPTCP has the potential to duplicate data across paths. While this would be extremely inefficient for all flows over Iridium®, there may be utility in replicating some low-volume flows that are critical in nature. For example, infrequent payload commands can be replicated and received quickly, even if a major loss event were to occur. Since the proposed solution leverages standard developed protocols, it is a first step toward the goal of introducing network priorities and quality of service into the architecture.

Finally, there is discussion in the community about additions to MPTCP that may remove the need for service specific proxies. Future work may benefit from exploring these options, how they apply to this system, and what improvements, if any, they offer to flows.