

# ArduPilot MAVLink telemetry reliability and latency over an Iridium Certus satellite service

Stephen Dade ([stephen@rpanion.com](mailto:stephen@rpanion.com))

January 2026

*With thanks to GroundControl for supplying a Rock-REMOTE UAV OEM modem and Certus airtime for testing*

## Summary

Iridium Certus is a satellite Internet service, with several small-form factor modems available from various manufacturers.

The datarate (22-352 Kbit/sec, depending on service level) in the uplink (modem to Internet direction) makes it suitable for sending telemetry from autonomous vehicles running ArduPilot.

Certus is ideal for operations on unmanned systems in remote locations, where other (land-based) communications systems do not have coverage. The small size of Certus modems (such as the RockREMOTE UAV OEM) and antennas make them suitable for unmanned systems

With the appropriate settings, ArduPilot MAVLink telemetry can be reliably used over a Certus 100 service, with latencies of 600-1600ms.

The addition of an external VPN service degrades the reliability almost completely. If a VPN is required, satellite service providers do have specifically optimised solutions available.

## Definitions

**Latency:** Round-trip time from the ground station (ie Mission Planner) to ArduPilot and back again

**Uplink:** The direction of data from ArduPilot to the ground station. The majority of telemetry data goes in this direction

**Downlink:** The direction of data from the ground station to ArduPilot. Typically consists of vehicle commands

**Direct IP Connection:** When telemetry data is sent over the Public Internet, without a VPN. Requires one side of the connection to have a public IP address.

**MAVLink:** The messaging standard used by ArduPilot for sending and receiving telemetry data.

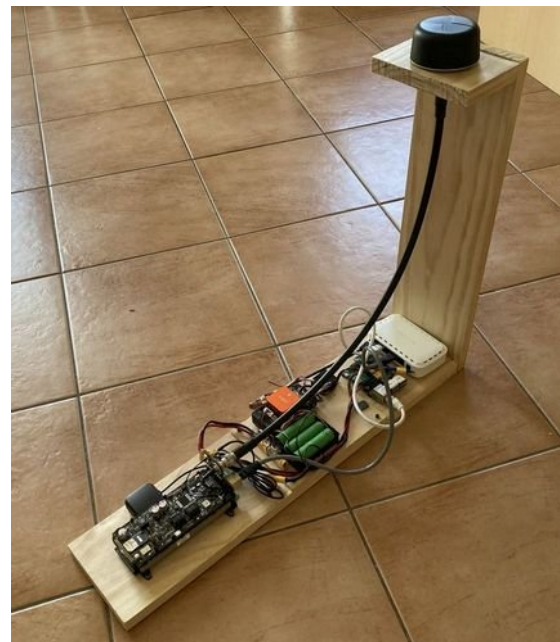
**Outage:** Period of time where no MAVLink packets are received by the Ground Station.

## Setup and Configuration

An AWS EC2 instance, with a public IP, was used as the endpoint (ground station) for all tests. The EC2 instance was located in the nearest region (Sydney) to the test location.

On the “vehicle” side, a portable test rig was made up of:

- CubeOrange+ flight controller, running the latest ArduPilot version.
- CubeNodeEth peripheral as a UART to Ethernet gateway, between the CubeOrange+ and network switch
- Raspberry Pi 3B+ as a UART to Ethernet gateway, capable of running VPN software. It was connected between the CubeOrange+ and network switch. The “mavlink-router” software was used for the UART/Ethernet bridging.
- RockREMOTE UAV OEM modem (with Maxtena M1621HCT-LP-SM antenna), connected to the network switch
- Batteries (12V) for power
- Generic network switch



*Illustration 1: Test Rig Layout*

The RockREMOTE UAV OEM was configured as a DHCP server for the test rig network.

For baseline testing, the RockREMOTE UAV OEM was disconnected and the vehicle network was connected to a FTTP Internet connection.

The below diagram (Illustration 2) details the network layout connectivity, where blue is IP (Ethernet or fibre), green is UART and grey is satellite.

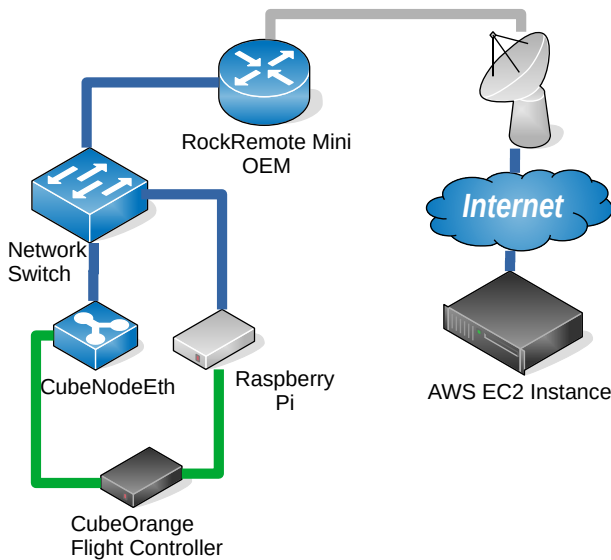


Illustration 2: Network Layout

All tests were run for 600 seconds. Both TCP and UDP connection types were tested.

On the Iridium Certus side, a Certus 100 service was used, giving a maximum of 88Kbps downlink and 22Kbps uplink.

## Methodology

For logging the required data, the “mavlinklinktester” (MLLT) was developed. This tool automatically logged the latency, data transfer rate and outages of a MAVLink connection, generating detailed statistics and logs for further analysis.

MLLT is open-source and available from <https://github.com/stephendade/mavlinklinktester>

## Baseline Testing

### Baseline Direct Tests

For an initial benchmark of a “best case” connection, measurements of a direct connection between the CubeOrange, Raspberry Pi and AWS server for made, using a FTTP Internet connection. Two sets of tests were run, using the CubeNodeEth for the first set and the Raspberry Pi for second set. This is shown in the table below:

Connection Type	CubeNodeEth	CubeNodeEth	Raspberry Pi	Raspberry Pi
Protocol	TCP	UDP	TCP	UDP
Packets received	55935	57603	58227	56573
Bytes received	1894201	1950654	1987354	1911066
Dropped packets	0.00%	0.00%	0.00%	0.00%
Misordered packets	0.00%	0.00%	0.00%	0.00%
Mean latency	19ms	19ms	70ms	60ms
Median latency	19ms	19ms	53ms	37ms
Outage time	0.00%	0.00%	0.00%	0.00%

As expected, the link quality was very good. The latency was much higher when going via the Raspberry Pi, however.

Going to a higher baudrate (57600 to 921600) on the UART connection between the CubeOrange and Raspberry Pi gave a much lower latency, as shown in the below table:

Connection Type	Raspberry Pi	Raspberry Pi
Protocol	TCP	UDP
Packets received	58022	58002
Bytes received	1952554	1951856
Dropped packets	0.00%	0.00%
Misordered packets	0.00%	0.00%
Mean latency	22ms	18ms
Median latency	20ms	18ms
Outage time	0.00%	0.00%

### Baseline VPN Tests

In many cases, a VPN is required to connect the Flight Controller to Ground Station. For example, if public IP addresses are not available or if the MAVLink stream is required to be encrypted.

Two popular VPN services are Zerotier and Wireguard.

In Wireguard’s case, the Wireguard server was set up on the same AWS server that the tests were run from.

The tests were run again using these two services (running on the Raspberry Pi):

VPN	Wireguard	Wireguard	Zerotier	Zerotier
Protocol	TCP	UDP	TCP	UDP
Packets received	58004	58009	57961	58011
Bytes received	1951363	1951856	1949914	1954145
Dropped packets	0.00%	0.00%	0.00%	0.00%
Misordered packets	0.00%	0.00%	0.00%	0.00%
Mean latency	22ms	19ms	22ms	19ms
Median latency	20ms	19ms	20ms	19ms
Outage time	0.00%	0.00%	0.00%	0.00%

As expected, the VPN services did not change the overall latency or reliability, compared to the direct tests.

Over reliable high-bandwidth connections, there is no appreciable difference between TCP and UDP, or VPN vs non-VPN configurations. However, if running via a companion computer, the UART connection to the flight controller must be set to 921600 or higher to ensure low latency.

### Lowering the Streamrates

By default, Ground Station software (such as Mission Planner, MAVProxy and QGroundControl) will request telemetry to be streamed at a rate of 4Hz (known as the “streamrate”). This can be configured higher or lower to suit user requirements

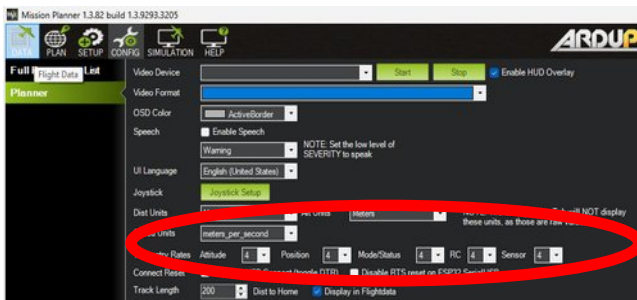


Illustration 3: Streamrate configuration in

### Mission Planner

Noting that the Certus 100 service maximum uplink datarate is 22 Kbit/sec (2.72 Kbyte/sec) and the average datarate from the Baseline tests was 3.25 Kbyte/sec, the telemetry streamrate should be lowered to fit the Certus 100 datarate.

The Baseline tests were run again at 2Hz streamrates, using the Raspberry Pi via both VPN and non-VPN connections.

Averaging over these tests:

- 51 pkts/sec (1689 bytes/sec) are expected at a 2Hz streamrate
- 97 pkts/sec (3253 bytes/sec) are expected at a 4Hz streamrate

Thus a 2Hz streamrate will comfortably fit within the datarate limits of a Certus 100 service.

Note the 2Hz streamrate is not exactly half of the 4Hz streamrate. This is due to some telemetry fields being transmitted at independent rates (such as the HEARTBEAT message is always sent at 1Hz) or message responses to GCS commands (such as the TIMESYNC message used for ping measurements, sent from MLLT at 2Hz).

## Satellite Testing

Initial testing with the modem in a (suburban) backyard, with the antenna 1m off the ground gave very poor results. The modem was then moved to the roof, which gave much better results.



Illustration 4: Test Rig mounted on the roof

It is critical that the antenna is mounted at a typical roof height for the area. Tree and building blockages greatly affect performance.

### Satellite Direct Tests

Initial testing again used the CubeNodeEth and Raspberry Pi on a direct connection to the AWS server, at 2Hz streamrates. The results are in the below table:

Connection Type	CubeNodeEth	CubeNodeEth	Raspberry Pi	Raspberry Pi
Protocol	TCP	UDP	TCP	UDP
Packets received	29864	27827	30571	28346
Bytes received	1004522	933872	1024249	948111
Dropped packets	0.00%	7.00%	0.00%	8.00%
Misordered packets	0.00%	3.00%	0.00%	1.00%
Mean latency	903ms	895ms	1451ms	827ms
Median latency	846ms	861ms	1432ms	787ms
Outage time	1.00%	1.00%	0.00%	1.00%

The latency differences between the CubeNodeEth and Raspberry Pi over a TCP connection were unable to be explained. It is possible that it was simply due to the local conditions are the time of the test or a difference in how TCP connections are handled between the two devices.

The outage periods of 1% represent 6 seconds of outage every 10 minutes. The logfiles collected from MLLT showed that outages were typically 1-2 seconds, with none exceeding 4 seconds. This was in both uplink and downlink directions.

The optimal settings are to run at a 2Hz streamrate over a TCP or UDP connection.

### Satellite VPN Tests

It should be noted that using a VPN over a Certus link is *not* recommended by service providers.

The tests were run again (at a 2Hz streamrate) using the Zerotier and Wireguard VPN services running on the Raspberry Pi.

Connection Type	Zerotier	Zerotier	Wireguard	Wireguard
Protocol	TCP	UDP	TCP	UDP
Packets received	0	4721	29105	13234
Bytes received	0	157667	974431	444412
Dropped packets	N/A	57.00%	1.00%	54.00%
Misordered packets	N/A	6.00%	1.00%	3.00%
Mean latency	N/A	3466ms	1685ms	3500ms
Median latency	N/A	3494ms	1570ms	3446ms
Outage time	N/A	61.00%	12.00%	5.00%

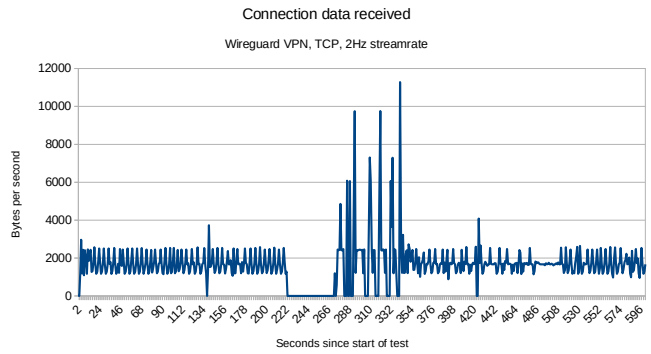
A TCP connection via Zerotier was unable to be established. Also the latencies were significantly (more than double for UDP) higher.

However, this does not tell the full story, as the “dropped packets” can be unreliable for high loss rates. This is due to the MAVLink packet sequence being an 8 bit number, so large (>256) packet gaps can be difficult to discern.

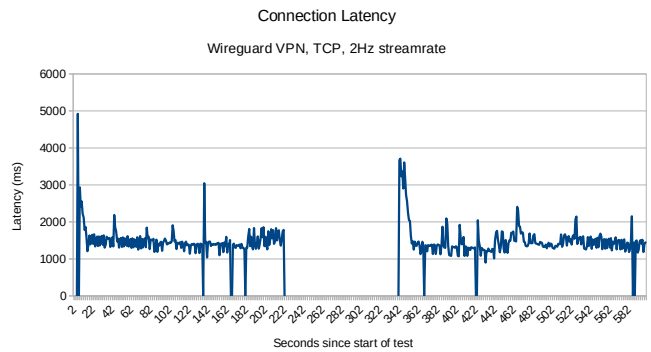
Showing a comparison with the expected packet numbers from the Baseline testing gave a more realistic picture.

Connection Type	Zerotier	Zerotier	Wireguard	Wireguard
Protocol	TCP	UDP	TCP	UDP
Packets received	0	4721	29105	13234
Expected Packets received	30329	30329	30329	30329
Percent packet loss	100.00%	85.00%	4.00%	56.00%

Whilst a 4% packets loss and 12% outage time for the Wireguard TCP configuration may seem low enough to work in some cases, the graphs tell a different story:



The above data throughput graph shows a 48 second outage in telemetry. Due to TCP re-sends, the packets are queued and sent later – as seen by the later spikes in data rate (270-340 seconds).



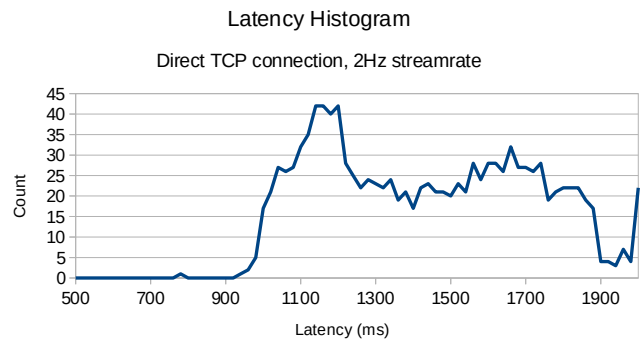
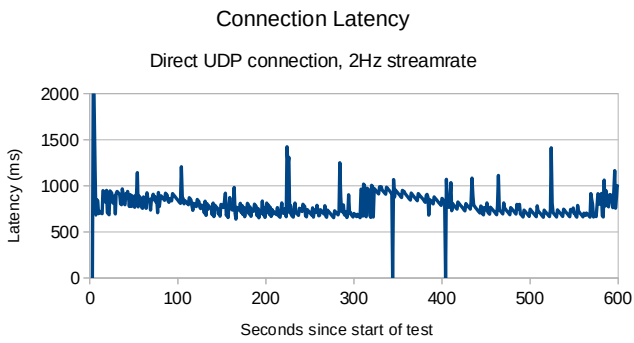
The above latency graph shows a 115 second outage in the latency, which is the downlink (ground station to vehicle) direction. This represents an almost 2 minute outage of not being able to send commands to the vehicle.

VPNs are not recommended over a Certus connection. Some service providers do offer VPN services optimised for Certus, which will be more reliable.

### Notes About Satellite Latency

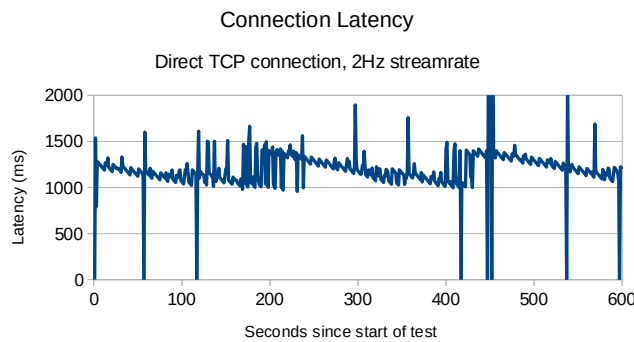
The Iridium Certus satellite constellation consists of 66 satellites in low-Earth orbit. Therefore a user needs to be handed over from satellite to satellite as they pass overhead. Typically, handover occurs every 5-10 minutes.

Looking at a graph of latency over the direct TCP and UDP tests show the following:



A (direct) UDP connection has a much smaller variation of latency, of between 650-1000ms. This gives a 400ms variation total. A TCP connection, on the other hand, has a spread of 1000-1900ms, so a 900ms variation.

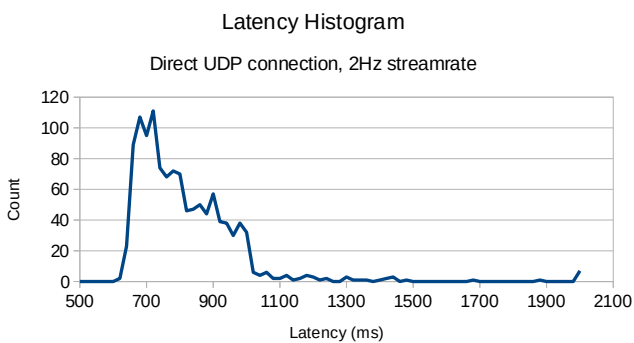
The expected round-trip latency over a Certus connection is 650-1000ms over UDP and 1000-1900ms over TCP.



## Recommended ArduPilot Configuration

The latency tends to follow a saw-tooth pattern, with a 300-500 millisecond variation. The regular sudden increase *could* be indicative of a satellite handover.

Putting this data into a histogram gives a clearer idea of the spread of latencies:



Based on the results, a recommended hardware and software configuration can be given.

On the hardware side, the flight controller running ArduPilot needs an Ethernet interface to connect with the RockREMOTE UAV OEM. This could be:

- Flight controller with a native Ethernet interface, such as the CubeRed
- Serial to Ethernet interface (via PPP), such as a CubeNodeEth or Botblox DroneNet
- Single-Board Computer (such as a Raspberry Pi or NVIDIA Jetson) running a MAVLink router between it's UART and Ethernet ports.

On the software side, ensure the IP addresses between the RockREMOTE UAV OEM and other networked hardware are on the same subnet (192.168.250.XXX by default). Note the RockREMOTE UAV OEM's DHCP server is disabled by default.

The ArduPilot streamrates should be set to 2Hz (see <https://ardupilot.org/dev/docs/mavlink-requesting-data.html>) and ArduPilot be configured to block any streamrate changes from the GCS.

## Conclusions

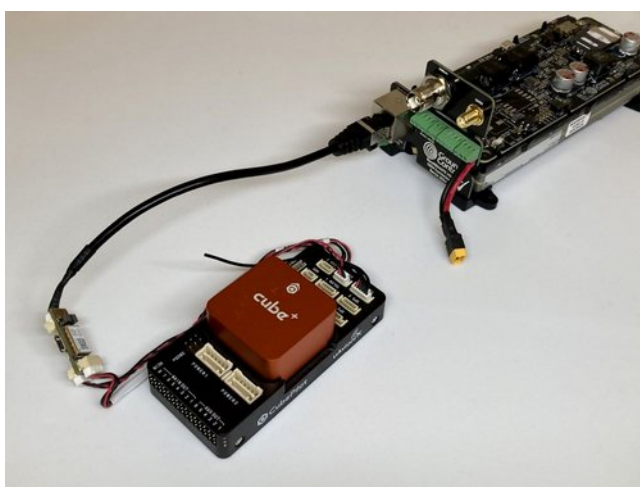
Using a Certus 100 service (and associated modem, such as the RockREMOTE UAV OEM) for MAVLink telemetry is reliable and practical, particularly in remote loca-

tions where other communications options are not available.

For maximum reliability, using a streamrate of 2Hz via a TCP or UDP connection is recommended. Usage of an external VPN service is not recommended and can cause extreme packet loss and unreliability.

During ground-based testing it is important that the antenna is location at the typical roof or vegetation height (of higher) of the local area. Blockages of the sky will make the Certus link quite unreliable.

The typical round-trip latency of a Certus 100 service is 650-1900ms, depending on the exact configuration (UDP has a lower latency than TCP) and the local conditions.



*Illustration 5: Connection between a Flight Controller running ArduPilot and the RockREMOTE UAV OEM*