Payload Data Delivery Solutions and Optimisation for Ground Segment as a Service Operations

Giovanni Pandolfi Bortoletto[a*], Giovanni Baccichet[a], Frédéric Manteau[a], Giovanni Antonio Peru[a], Giovanni Zanotti[a]

[a] *Leaf Space S.p.A., via Cavour 2, Lomazzo, Como, 22074, Italy*
** Corresponding Author*

Abstract

As satellite constellations grow in scale and diversity, efficient payload data delivery has become a critical bottleneck in Ground Segment as a Service (GSaaS) architectures. This paper investigates various payload data delivery methods and presents optimisation strategies tailored to operational requirements and mission profiles.

We systematically analyse multiple delivery paradigms including real-time streaming protocols (TCP/IP, UDP), standard batch transfer mechanisms (S3/object storage, FTP/SFTP), chunked transfer architectures enabling progressive delivery, and advanced edge-based file transfer solutions leveraging ground-to-satellite return links for enhanced control and error recovery. Each method is evaluated across critical performance metrics: end-to-end latency, throughput efficiency, packet loss tolerance, Automatic Repeat reQuest (ARQ) capability, scalability under multi-tenant operations, and infrastructure complexity. Our methodology combines theoretical performance modelling with empirical data from operational Leaf Space ground station networks, providing quantitative benchmarks for each delivery approach.

The research presents use-case-specific optimisation strategies that balance latency requirements against delivery reliability guarantees. We address diverse operational scenarios including near-real-time mission-critical telemetry requiring sub-second delivery, high-volume Earth observation data prioritising throughput over latency, congested or low-quality link environments demanding resilient retry mechanisms, and multi-tenant operations with competing Quality of Service requirements.

Expected outcomes include practical decision frameworks for selecting optimal delivery architectures based on mission constraints, protocol-level optimisation techniques (TCP window tuning, chunking strategies, parallel transfers), and strategies for mitigating both space segment and ground station infrastructure limitations. These findings directly apply to GSaaS providers, satellite operators, and system integrators seeking to maximise data delivery performance while minimising operational costs and complexity.

Keywords: *Ground Segment, GSaaS, Data Delivery*

Acronyms/Abbreviations

ARQ	Automatic Repeat reQuest
ConOps	Concept of Operations
EO	Earth Observation
GS	Ground Segment
GSaaS	Ground Segment as a Service
IoT	Internet of Things
LEO	Low Earth Orbit
RF	Radio Frequency
RTT	Round-Trip Time

1. Introduction

As the space market continues to expand, the number of operational satellites keeps increasing, with an ever-growing number of operators deploying satellites in constellations. Having multiple assets in orbit that are operated in clusters with centralised and unified methods is essential for efficiently scaling up operations.

There are indeed many different companies that have built satellite fleets with various mission objectives, which benefit greatly from the distributed nature of the constellation.

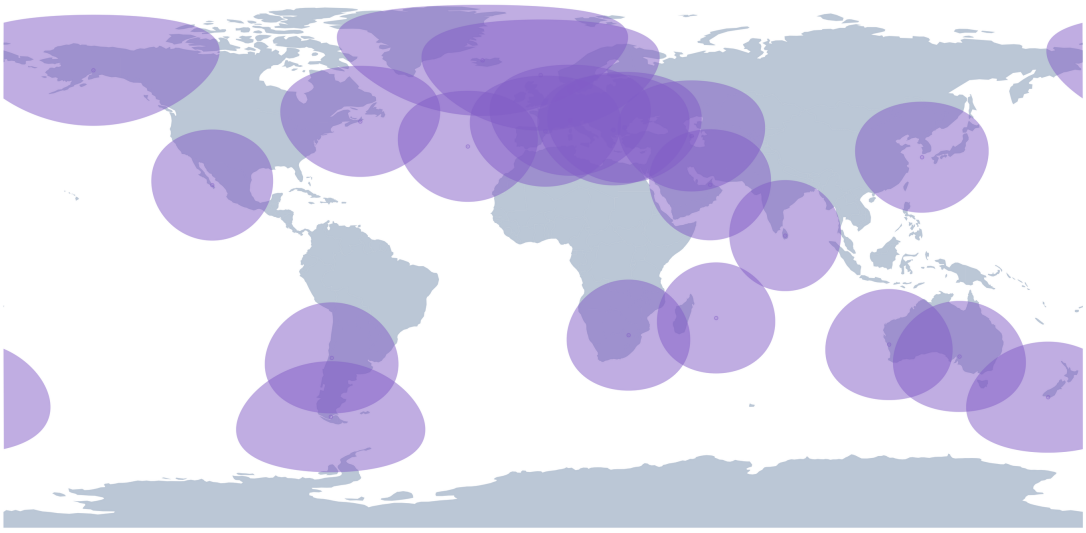

Figure 1: coverage map of the different ground stations in Leaf Line network.

Among these we find many Earth Observation missions, typically using either optical or radar payloads that, thanks to a dedicated orbital constellation design, achieve very reduced revisit periods of their target areas. Another field is given by telecommunication services, with satellite internet or mobile telephony provision being a driving business. Similar considerations hold also for IoT missions, where the need to maximise the continuity of service is driving the need for multiple assets working in harmony.

In all the scenarios described above, the data are arguably the most important output of the mission, hence the data delivery from the ground station to the mission control centre is a key aspect that shall not be underestimated.

Looking down at what occurs on the ground, the trend of the last decade in the space market is to transition more and more from mission-dedicated ground segments towards the exploitation of Ground Segment as a Service (GSaaS) providers, that provide a distributed network of ground stations over the globe in a shared resources paradigm. While this concept provides many benefits in terms of costs, complexity and development and integration times, there may be a reduction in configurability and the need to comply with a strict and fixed data delivery method.

One of the established actors among GSaaS providers is Leaf Space, which through their Leaf Line service, makes their whole ground station network available to satellites operators, exploiting in-house developed scheduling software [1]. Figure **1** shows the ground stations in Leaf Line network on the globe, each one with the coverage it provides as seen by a LEO satellite.

The goal of this paper is to show the different solutions that Leaf Line users can choose to avoid being constrained to a specific data delivery method. In such a way the user can exploit the flexibility provided by the solutions put in place to make sure that the compliance with their mission requirements is met.

After this introduction, the second section will provide an in-depth overview of all the available data delivery methods, presenting each advantage and disadvantage. The third section will introduce multiple mission scenarios, together with the related mission requirements and then will present the best delivery method selected. Finally, Section 4 will wrap up the most relevant outcomes of the analysis, extracting a possible decision pattern based on the users' requirements, before introducing future developments for further improvement in the delivery methods.

2. Data delivery methods

As stated, the focus of this paper is on data delivery methods, concerning the transfer of the demodulated payload from the ground station to the end user. In this section we want to provide an overview of the available delivery methods that users can exploit to collect their payload data. The comparison between the methods will be performed

looking at both theoretical and operational data, providing qualitative and, when available, quantitative values.

The available methods are the following:

- Real-time stream, through a dedicated TCP or UDP connection.
- Chunked file delivery, where the transmission of data occurs progressively through a set of chunks with given size that are accumulated and sent as soon as they are filled.
- Batch file delivery via S3 buckets or via dedicated FTP/SFTP streams.
- Robust file delivery, where checking mechanisms are used to ensure all the parts of a file are correctly retrieved and in case retransmission of the missing or corrupted parts is required via the RF uplink.

The main differences lie in the overall data latency and in the required throughput of the system.

The real-time delivery is the first method presented here, and it can be achieved through different transport layer protocols, each one with its advantages and disadvantages. Currently the two possible options here are TCP and UDP.

The typical distinction between these two protocols is considering the associated performance, where TCP is the more reliable of the two, at the cost of higher overheads and latency, being a connection-oriented protocol. The connection is first established with a three-way handshake and then, due to its ACK-based mechanisms that provide automatic retransmission of data segments that are lost, the delivery of data is ensured to be successful and ordered even in lossy and disturbed channels. However, the reliability of the link adds some downsides, due to the required overhead of the protocol, the congestion control and its operativity. Indeed, any issue with packet loss, duplication, or out-of-order delivery requires retransmission, adding latency in the delivery.

The UDP protocol on the contrary is completely devoted to throughput maximisation, without including connection negotiation, data order guarantee, error correction or any ARQ mechanism. In such a way, the protocol overhead is minimised, and the timeliness is enhanced by significantly lower latencies and absence of retransmission delays.

The throughput obtained by the real-time connection depends on the performance of the available internet connection between the two nodes, but in the case of TCP it is also influenced by the buffer window size, whose value directly defines the amount of total delay, and by the congestion control, which modulates the data rate to prevent channel issues. On the other hand, the throughput of UDP is effectively bounded by the physical layer alone, providing potentially much higher data rates with extremely low latency.

Regardless of the chosen protocol, real-time transfer proves useful in specific scenarios, in which the data need to be delivered with virtually zero latency. The TCP protocol could be preferred for a scenario without extremely high data rates, while UDP in the remaining cases.

The second method presented is the chunked file delivery, which sits roughly in between the real-time and the batch file delivery. Indeed, its working principle is to aggregate the incoming data into small files (the chunks) and send them to a remote storage location as soon as they reach a given tuneable size. For several reasons it combines the advantages of both methods.

First, having a progressive delivery of data, it lets the user achieve a reduced value of latency, which can be tuned by selecting a proper chunk size. Furthermore, it brings also the advantage of having the data delivered in a file to a storage, hence without the issue of losing valuable data due to temporary disconnection problems of a real-time stream.

When the latency is not a concern, the batch file delivery is a reasonable solution, that does not require any real-time stream acceptance mechanism and provides the full payload data in a single file. This means that, contrary to the chunked delivery the data are promptly ready for post-processing and/or analysis. The simplest solution for the actual file transmission is for the user to host an S3 bucket to which the GS uploads the file directly through a dedicated API. Another solution is instead to use FTP/SFTP protocols to instantiate a dedicated transfer session. The choice between the two is mostly a matter of user preference and infrastructure,

since no specific advantage is present in one way or the other.

The last delivery method of interest here is a robust file delivery strategy that employs mechanisms to ensure the correctness of data reception, by exploiting two RF channels, typically an X-band downlink and an S-band uplink. The basic architecture in this case comprises hosting in a virtualised environment on the GS server a container provided by the user, which is fed with the incoming blocks of data. The user's container is then in charge of verifying the correctness of the received data and, in case needed, generating the telecommands to require retransmission through the RF link. Such telecommands are directly forwarded to the modem in order to be uplinked and eventually receive the required missing or corrupted portions of the file, ensuring proper ARQ capabilities. The protocol used finally to deliver the aggregated file can be as in the previous case, with an S3 bucket or via SFTP.

3. Operational test cases

In this section, the goal is to present four different mission scenarios and, after listing the mission and related data transmission requirements, provide a justification of the best delivery method to comply with them.

The first scenario considers a telecommunication mission, whose main goal is to stream high data-rates to a gateway node, passing through the ground station. Such a setup could resemble a constellation mission to provide network connectivity via satellite to remote terminals, for either space-based mobile telephony or IoT. The main requirements for a data delivery architecture in this case are the following.

- Handle high data-rate – Considering that the satellite is aggregating streams from multiple users, the overall bandwidth to downlink reaches large values, easily as high as several hundreds of Mbps.

- Zero latency – Given that the provided service is to relay data in real-time, the latency is an extremely relevant performance index, for which a near zero value is crucial, potentially at the expense of link reliability.

A real-time stream is certainly the only viable approach, with the provided requirements, due to the zero-latency constraint. The main trade-off then is between the TCP and the UDP protocols for the transport layer. As suggested in the previous section, the driving mechanism here is the balance between throughput and the reliability. The TCP protocol indeed, is usually more limited than the UDP. Its maximum achievable throughput depends on the round-trip time (RTT) and on the buffer window size at both the sender and receiver ends. Considering the standard configuration of a Linux-based machine for TCP window size and congestion control mechanism, with a non-congested link bandwidth of 1 Gbps, and an RTT of 250 ms, which can be considered as a worst-case value for a cross-globe internet connection, the maximum theoretical throughput reaches 134 Mbps. However, with a proper tuning of the window size parameter, much larger throughputs can be achieved. For example, with a 24 MB window size at both ends, the expected peak approaches 800 Mbps. Increasing the window over 64 MB, the 1 Gbps value can be reached, practically hitting the boundary given by the internet connection. The only issue is that, even with optimised congestion-control algorithms (like the BBR algorithm, specifically designed for high-bandwidth-delay scenarios), the peak speed is reached only after 3-5 seconds from the start, meaning that short- duration connections will never reach high speed. On the contrary, given that no congestion mechanism is present in UDP, the RTT doesn't affect its throughput, and the full speed is reached from the beginning. In this case the maximum value is directly bounded by the internet connection.

With these considerations in mind the best solution for the mentioned telecommunication mission would be surely to use TCP if the required datarate remains contained, since the impact of the congestion mechanism will not introduce much delay in reaching the top speed, while still providing a highly reliable link. If the rates start reaching values that require larger buffer windows, and the delay in reaching full speed cannot be accepted, the UDP is probably the best way forward.

The second scenario of interest is considering a mission for environmental disaster monitoring, whose main driver is to download the acquired data aggregated as large files, keeping a non-zero but still reduced latency. In particular, the goal in this case is to limit the latency to 10 seconds at most, in order to promptly process incoming data and raise warnings as soon as possible. For this case, the throughput is

not a real driver, but mainly a direct consequence of the delivery method. What matters is instead the capability to handle large volumes with a limited latency from data reception by the GS to the delivery for the post-processing. The best choice for this case is to go for a chunked file delivery, which permits to limit the latency and to handle large file transfers. Also here, similar to the TCP scenario, a proper tuning of the chunk size is required, in order to meet the latency requirement with a limited number of chunks. Indeed, the larger the chunk size, the larger the latency will be, but the fewer the chunks will be, hence providing a lower overhead.

The tuning can be performed by considering that the latency of a single chunk is composed of three main contributions: the time it takes to aggregate the chunk at the RF-link datarate, the time it takes to deliver it from the GS server to the final end-user destination and a fixed latency value introduced by the preparation processes to instantiate each delivery stream. With the knowledge of the incoming datarate (assumed as 125 Mbps), the delivery throughput (225 Mbps) and the fixed latency (500 ms), it is possible to derive the chunk size that satisfies the required latency constraint of 10 seconds, i.e. ~95 MB.

Considering instead the case of a scientific mission that requires consistent download of large volumes of data without any latency requirement, the real-time or chunk-based delivery methods are not strictly needed. Indeed, in this case the post-processing analyses are performed on multiple samples collected across different passes. As a consequence, having all the data delivered to a remote location with an S3 bucket or through SFTP is advantageous. First, it doesn't require activity on the receiver side, other than putting in place the bucket (passive) or keeping one or multiple SSH sessions open. Secondly, it provides the user directly with the data in an aggregated file, which can be useful for the collection and storage of the mission products.

For this delivery method, throughput does not significantly impact the overall performance. The upload of the file (either to the S3 bucket or via SFTP) will start at the end of the contact, so the actual rate will just determine how long it will take to complete the delivery. As latency is not a concern for this scenario, there are no particular reasons to push for large throughputs. One additional advantage of working on a single file aggregated after the end of the pass, is the possibility to exploit parallelisation of the stream, which cannot be achieved with the real-time or chunked based deliveries.

As highlighted, there are two possible methods for the file delivery, through S3 buckets or via SFTP protocol. Both alternatives use the TCP protocol for the stream, and it is possible to improve the overall throughput of the upload, exploiting parallelisation, hence instantiating multiple TCP streams. Moreover, both methods instantiate encrypted connections, so that end-to-end security is preserved. As said in the previous section, the choice among the two alternatives is mostly a matter of user preference and infrastructure availability.

The last case taken into consideration here is a similar scenario, where an EO mission generates large volumes without real latency concerns, but with sensitive and high-value data. In this case, all the considerations done for the previous scenario still apply, hence a file-based delivery system is still appropriate. However, the nature of the data requires ensuring that every single byte is delivered without corruption, which can be obtained through the last delivery method presented. The delivery is hence passing through an intermediate stage, in which the incoming data from the ground modem are passed to the user's containerised software running on the GS server. This is done to achieve the ARQ capability, verifying integrity and generating telecommands for the retransmission of missing or corrupted portions via RF. It is important to highlight how this crucial passage is done at the edge of the delivery chain, the GS, allowing to benefit from both the high-throughput and low-latency of the cabled connection between the modem and the server and the relatively large computational resources available. This procedure runs in real-time during the pass to allow direct feedback to the spacecraft and everything is happening at the edge, while the accepted portions are aggregated in the meantime to be finally delivered via S3 or SFTP. This means that all the advantages of file-based delivery are still present, hence not requiring direct operations from the mission control centre during the pass.

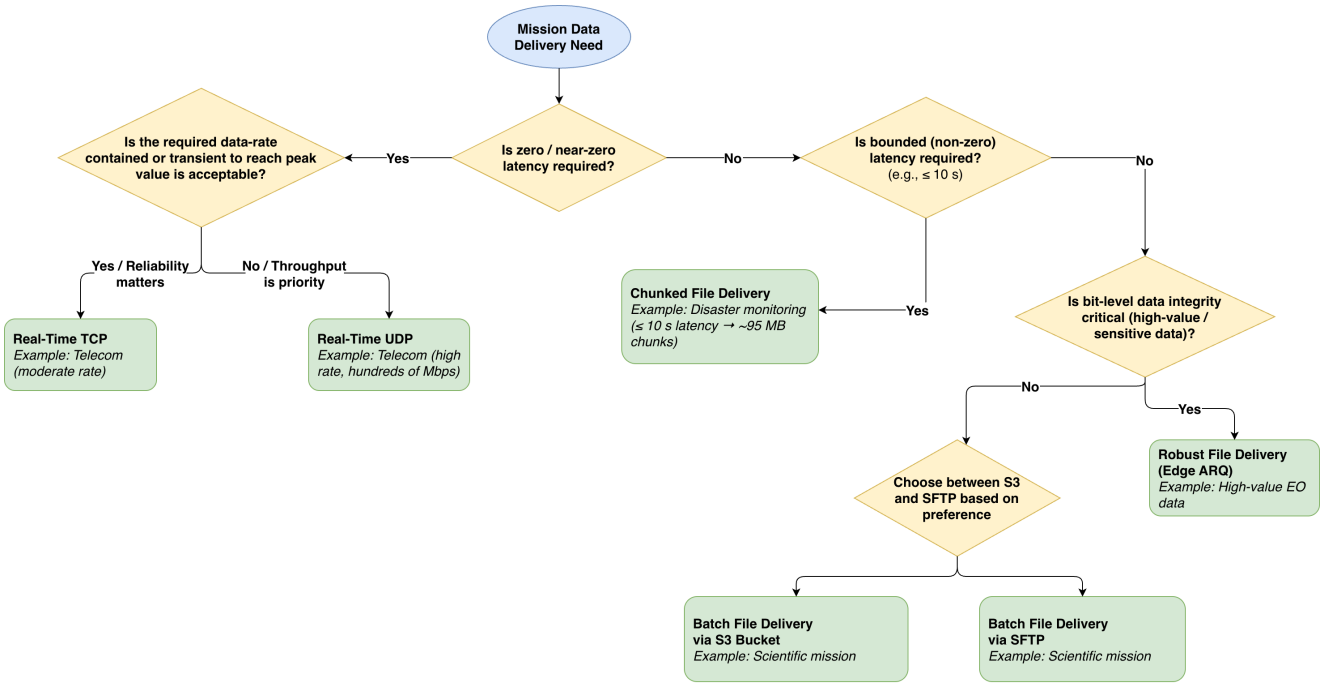

Figure 2: decision diagram for the data delivery method.

4. Discussion and future works

In the previous sections we have first presented and then described in real operational scenarios the four data delivery methods that Leaf Line users can choose from. The available options are each one suited to different sets of requirements from different mission ConOps. To summarise, the telecommunication scenario demonstrated that real-time streaming is the only viable path when near-zero latency is mandatory. Within this category, TCP offers a reliable and ordered strategy that suits moderate data-rate missions, while UDP becomes necessary when the required throughput exceeds what TCP congestion control can sustain without unacceptable transient delays. The disaster-monitoring scenario showed that chunked file delivery provides an effective compromise. Indeed, by tuning the chunk size as a function of incoming data-rate, delivery throughput and fixed processing latency, the user can tune the end-to-end latency to a target value, while retaining the resilience of file-based transfers against temporary connectivity losses. The scientific-mission scenario confirmed that, when latency is not a concern, batch file delivery via S3 or SFTP remains the simplest and most efficient approach, with the choice between the two governed primarily by the user's preferences and infrastructural availability. Finally, the Earth observation scenario showed how the robust file delivery method, exploiting an edge-deployed user container and RF-based ARQ, can guarantee bit-level data integrity for high-value payloads without requiring any real-time involvement of the mission control centre.

From these cases a general decision pattern emerges. The first discriminator is whether the mission demands near-zero latency: if so, a real-time stream (TCP or UDP) is the only option, and the secondary choice depends on the balance between throughput demands and link reliability. If the mission tolerates a bounded but non-zero latency, chunked delivery should be considered, with the chunk size tuned to the specific latency target and available link parameters. When no latency constraint exists, file-based batch delivery is preferred for its operational simplicity, with the S3-versus-SFTP trade-off resolved by user preference. On the other hand, if bit-level integrity of the delivered data is critical, the robust delivery method with edge ARQ should be added on top of the file-based approach. This hierarchical decision framework, schematised by Figure 2, that emerges from the operational scenarios presented in Section 3, can serve as a practical reference for users to have their data delivery architecture comply with mission and operational requirements.

Looking ahead, several protocol-level developments may further improve payload data delivery performance. The QUIC transport protocol [2], which combines the reliability of TCP with reduced connection-establishment overhead and increased built-in encryption, is a strong candidate to complement the current real-time alternatives, particularly for missions that require both low latency and secure delivery. Additionally, the Saratoga protocol [3], originally designed for high-speed data transfer over lossy links, presents an interesting alternative for real-time delivery, improving throughput in scenarios where traditional TCP tuning reaches its limits. For file-based transfers instead, the CCSDS File Delivery Protocol (CFDP) [4] offers a robust framework with a high heritage for reliable file transfer over delay-tolerant links. Its integration into the GSaaS delivery chain could standardise the interface between spacecraft and ground-segment file management.

Future work will focus on the evaluation and integration of these protocols within the Leaf Line architecture, together with the extension of the presented analysis to evaluate scenarios where these new methods could outperform the current architecture.

References

[1] Zucchelli, Enrico, et al. "Automatic Scheduling for a Ground Segment as a Service Platform Dedicated to Small Satellites." (2018).

[2] Yang, Siyu, Hewu Li, and Qian Wu. "Performance analysis of QUIC protocol in integrated satellites and terrestrial networks." 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, 2018.

[3] Wood, Lloyd, et al. "Saratoga: a Delay-Tolerant Networking convergence layer with efficient link utilization." 2007 International Workshop on Satellite and Space Communications. IEEE, 2007.

[4] Pavale, Sanjay, E. Unnikrishnan, and P. Laksminarasimhan. "Design, implementation and performance evaluation of CCSDS CFDP protocol." 2010 IEEE International Conference on Computational Intelligence and Computing Research. IEEE, 2010.