SpaceOps-2023, ID # 345

Moon, Mars, and Beyond: How to Enable Interplanetary Communications

Joy Fasnacht[a], Ed Meletyan[b], Kathy O'Donnell[c], Jonathon Fraker[d], Tom Johnson[e]

[a,b,c] *Amazon Web Services, Space Products Solutions Architecture, Aerospace and Satellite Team, joyf@amazon.com, emelet@amazon.com, klodon@amazon.com*
[e,f] Amazon Web Services, Space Services Team, jfraker@amazon.com, spacetom@amazon.com

Abstract

Designs of Amazon Web Services (AWS) infrastructure and algorithms are effective at planetary scale and these designs and methods AWS uses can tackle the challenges of interplanetary communication. Depending on their relative orbits, communications between Earth and Mars can take between 4 and 24 minutes. Scientists estimate humanity will have sustainable exploration on the moon by the end of the decade, and colonies on Mars by 2050. To sustain communication to outposts, a multi-hop system with authentication, storage, redundancy, and guarantees will be required.

Acronyms/Abbreviations
AWS – Amazon Web Services
AZ – Availability Zone
EC2 – Amazon Compute Cloud
S3 – Simple Storage Service

1. Introduction

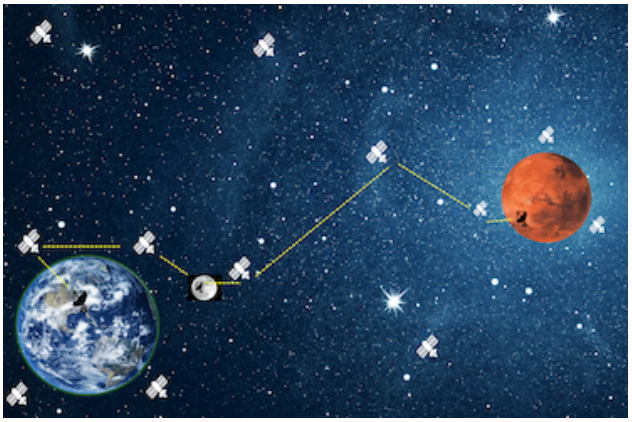
This topic explores design methods used at planetary scale in AWS global networking, storage, and compute infrastructure, as well as drawing parallels from history to explore concepts for re-use in interplanetary communication. We describe commonly used patterns in AWS technology that supports global scale, and detail patterns for future consideration.

2. Humanity's Voyage to Space

We can learn from the history of exploration on Earth, and take lessons from the early explorers on their ocean voyages. To survive these voyages, crews had to take with them everything that they needed. They needed to be self-sufficient, and took along a lot of spare equipment, especially for mission-critical items.

Space has some stark difference to those early voyages. Our modern-day exploration vessels – spacecraft – will need to handle these challenges. These vessels, and ultimately human inhabited outposts, will have navigation, sensor, life support, communications, and many other technologies that require localized compute and storage in large enough quantities to process the terabytes or petabytes of data generated daily. Mission operators will require access to scalable and durable infrastructure that stores and delivers time sensitive information when they need it. Durability will include both hardening aspects as well as redundancy. Specific use cases include:

- Streaming video and data from exploration missions;
- Enabling the physical and mental health of our modern voyagers;
- Video, entertainment, and chat; and
- Lowering the recovery time for emergency situations.



3. A Day in the Life of a Space Packet

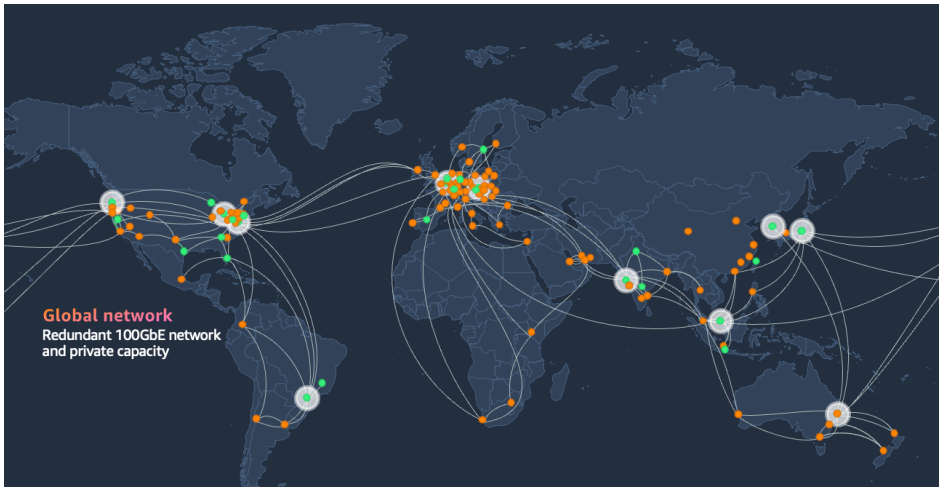
Terrestrial networking is largely two-dimensional and static, with known transmission speeds and reliability in various transmission mediums. Networking in space is in three dimensions, with communication nodes moving and routes dynamically changing, and obviously it is very big out there.

One thing we can learn from history is the idea of island hopping. Enclaves in the Earth's oceans provided our sea-going explorers with waypoints where communication could be gathered and sent on or consumed. Space networking will require this as well. Communication relays in orbit, on the moon, beyond the moon, and in Mars' orbit will enable that most ubiquitous of use cases, streaming a re-run of "The Martian" to a human habitation on Mars.

4. AWS Infrastructure Solutions

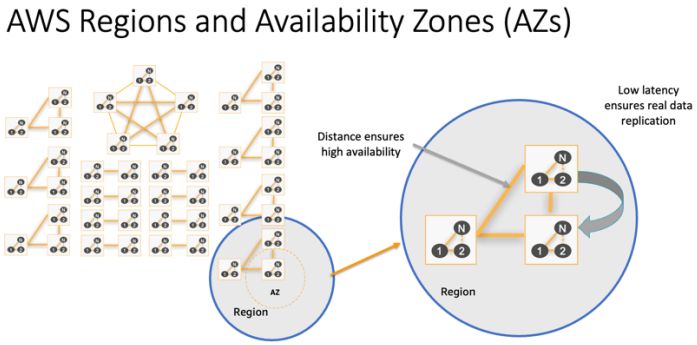
AWS has developed key tenets to building out terrestrial infrastructure, a process we started in 2006 that continues today. AWS internal infrastructure emphasizes horizontal scaling versus vertical, and an emphasis on the idea that "everything fails" (Dr. Werner Vogels, Chief Technology Officer[1]), requiring automation, self-diagnosis and healing, and fungibility.

Terrestrially, AWS has millions of worldwide active customer accounts, operating across the planet using services from our 31 global Regions, each of which spans multiple Availability Zones (AZs). AWS delivers the highest earthbound network availability of any cloud provider. Each region is fully isolated and comprised of multiple AZs, which are fully isolated partitions of our infrastructure. In addition, AWS control planes and management console are distributed across regions, and include regional API endpoints, which are designed to operate securely for at least 24 hours if isolated from the global control plane functions.



AWS Regions have low latency, low packet loss, and high overall network quality because of a fully redundant 100 GbE fiber network backbone providing many terabits of capacity between Regions.

AWS has horizontally scaled capacity using the concept of an AZ – AZs consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities. Availability Zones exist on isolated fault lines, flood plains, networks, and electrical grids to substantially reduce the chance of simultaneous failure, and provide the resiliency of performing real-time data replication with the reliability of multiple physical locations.



5. Applications of AWS Infrastructure Best Practices

So how can we take these tenets of horizontal scaling, redundancy, extreme automation and self-healing from AWS, and apply them to interplanetary communications? Relays in space will need to be composed of fleets of communication satellites, capable of acting together as a single entity, in a similar fashion to how multiple data centers look to our terrestrial customers like a single AZ. Data transfers to a relay can take advantage of technologies like ***packet bundling, increased packet size, and increased error correction*** to reduce handshake overhead, and Artificial Intelligence algorithms like ***reinforcement learning*** can contribute to ***continuous smart routing*** on each node. Modern advancements in precision navigation such as ***quantum entangled clocks*** will allow each node in a fleet to know precisely where they are in time and space relative to neighbors in the overall network.

When designing AWS terrestrial networking, compute and storage, AWS insists on treating each item in infrastructure as a fungible, replaceable resource, which makes the use of ***Infrastructure as Code*** critical. Similarly, fleets of communication nodes in space will eventually be viewed as disposable resources; self-correcting where possible, but where not, quickly replaceable by a steady re-supply chain of new nodes being trucked to space

through supply lines. Automation will be perfected to a high degree of predictable certainty, and documented best practices used in highly distributed computing such as dependency isolation [2], constant work and self-healing [3], and load shedding [4] will be adapted for interplanetary communications.

- ***Dependency Isolation*** - In distributed systems, queue times can increase abruptly when there's an increase in request processing latency. There is little that an application can do if the slowdown is out of its control. However, the application can make sure that its APIs that don't depend on the database or requests that could have been served out of a cache still work even when the dependency is slow. This concept can be extrapolated to space network processing.

- ***Constant Work*** – AWS Network Load Balancers run on AWS Hyperplane, an internal service that is embedded in the Amazon Elastic Compute Cloud (EC2) network. AWS Hyperplane integrates customer changes into a configuration file that is stored in an Amazon S3 (Simple Storage Service) object-store bucket, and Hyperplane nodes fetch this configuration from Amazon S3 every few seconds. The AWS Hyperplane nodes then process and load this configuration file. This happens even if nothing has changed. Even if the configuration is completely identical to what it was the last time, the nodes process and load the latest copy anyway. Effectively, the system is always processing and loading the maximum number of configuration changes. Whether one load balancer changed or hundreds, it behaves the same. Additionally, the configuration is also sized to its maximum size right from the beginning. Even when we activate a new Region and there are only a handful of Network Load Balancers active, the configuration file is still as big as it will ever be. There are dummy configuration "slots" waiting to be filled with customer configuration. There is anti-fragility in this design. If AWS Hyperplane nodes are lost, the amount of work in the system goes down, not up. There are fewer requests to Amazon S3, instead of more attempts in a workflow.

- ***Load Shedding*** - At AWS we avoid overload by designing our systems to scale proactively, before they encounter overload situations. However, protecting systems involves protection in layers. This begins with automatic scaling, but also includes mechanisms to shed excess load gracefully, the ability to monitor those mechanisms, and most importantly, continuous testing. We use the following methods in building services at AWS for load shedding: 1/ Understand the cost of dropping requests; 2/ Triage and prioritize requests; 3/ Use timeout hints in requests; 4/ Finish started work; 5/ Look at request duration when managing internal queues; 5/ Protect against overload in lower layers such as the OS; 6/ Protect in layers (defense in depth).

6. Learnings

- Best practices from terrestrial AWS cloud-scale designs will apply in space
- Horizontal scaling and resource fungibility will be key
- Infrastructure as Code concepts will carry over to Software Defined Space Networking
- Robust comms will enable strong human advancement in space!

References

[1] – AWS re:Invent Keynote 2021 with Dr. Werner Vogels - https://www.youtube.com/watch?v=8_Xs8Ik0h1w
[2] - https://aws.amazon.com/builders-library/dependency-isolation/ - AWS Builder's Library
[3]- https://aws.amazon.com/builders-library/reliability-and-constant-work - AWS Builder's Library
[4] - https://aws.amazon.com/builders-library/using-load-shedding-to-avoid-overload/ - AWS Builder's Library