SOFTWARE-DEFINED PAYLOAD CONCEPTS FOR SMALLSAT COMMUNICATION CONSTELLATIONS

Kari Seppänen, Tapio Suihko, Marko Höyhtyä, Jani Suomalainen, Ijaz Ahmad, Antti Anttonen,
VTT Technical Research Centre of Finland, P.O. Box 1000, FIN-02044 VTT, Finland,
Email: {firstname.lastname}@vtt.fi

Abstract

Satellite communications development is driven currently by two large disruptions: 1) Integration of satellite and terrestrial networks together, mainly advanced by 3GPP standardization and its Non-Terrestrial Networking (NTN) work items. 2) Emergence of large satellite constellations to provide low delay and high-capacity connections globally. These constellations typically consist of hundreds or even thousands of small satellites. In addition to broadband systems, also machine-type communications via satellites is increasing since satellites can provide connections anywhere without the need to build terrestrial infrastructure that is not always possible nor economically feasible.

In order to provide services to very heterogeneous and all the time increasing customer base without yet knowing in detail their future needs, one needs to be able to adapt and update the satellites and satellite constellations over their lifetime. One way to do the update is to launch satellites with better functionalities as part of the constellation. The other option is to update the satellites themselves with the Software-Defined Payloads (SDP). This enables updating the software over-the-air and start using software-defined networking capabilities to integrate satellites tightly as part of 5G/6G networks. Software-defined satellites have payloads and on-board processors that can be reconfigured using commands from ground stations. Software-defined capabilities in the satellite can include beamforming antennas, routing processors, and demodulation capabilities. These payloads make it possible to adapt frequencies and transmission powers and steer antenna beams according to current end user needs. Furthermore, as the processing power of SmallSat platforms increases, it is possible to place some part of the network functions on orbit as well as to provide edge computing capacity for, e.g., task offloading.

The concepts expand the cyber-attack surface and set demands for new security solutions and architecture. For instance, opening of systems for software updates exposes satellites for the integrity violations and added control communications increases availability related risks. Security solutions—from software-defined security, software segregation, and security protocols to hardware and virtualized platform security, and to reactive and learning defences—impose costs and face unique challenges within the space domain.

In this paper, we will make a short review about the current state-of-the-art in software-defined payloads, network softwarization, and network function virtualization in order to identify the potential concepts and technologies for constrained SmallSat platforms. We further intend to include sustainability aspects by introducing SDP concepts which enable simultaneous communications and sensing capability. Moreover, we will provide an overview of SmallSat software frameworks and consider how those can be integrated with communications related SDP that would be managed and orchestrated by integrated terrestrial/non-terrestrial management system. We will discuss cybersecurity aspects as well, pointing potential vulnerabilities that software-defined payloads and remote update possibilities may have - and how to protect systems accordingly.

Finally, we will consider the performance and efficiency of the proposed solutions in order to find out what kind of network functions and Multi-access Edge Computing (MEC) functionality can be distributed to SmallSat platforms. Furthermore, we will consider how the software development pipeline should be arranged to support updating software and creating new services in such SmallSat constellations.

SmallSat constellations and 5G NTN

Satellite communications field is in a strong transformation phase currently. A large number of New Space players are developing and launching small satellites to low Earth orbit (LEO) to provide broadband and machine-type communication services globally [1]. These commercial players are both challenging and complementing traditional satellite operators that are providing services with large satellites from the geostationary orbit (GEO). The emergence of small satellites and their constellations enable improved satellite services in remote areas where terrestrial and GEO services cannot fulfil the capacity demands nor have coverage. Megaconstellations including hundreds or even thousands of satellites have become viable solutions due to rapid development in miniaturization, launching services, and the ever-increasing need for communication capacity. Even though the number of satellites in a single constellation is high, costs are not increasing so rapidly due to the use of serial production and recent progress in reusing launch equipment.

Many of the currently existing constellations such as Starlink are using proprietary technology which limits interoperability between different satellite systems or between satellites and mobile terrestrial networks. Therefore, 3GPP standardization actively develops NTN technology in order to enable 1) direct access from handhelds to satellites and 2) satellites as backhaul to the terrestrial 5G and coming 6G systems [2]. The work started in Rel. 15 in 2017 by defining channel models and deployment scenarios. Rel. 16 considered New Radio (NR) functionalities needed to support LEO and GEO satellites. For example, the Doppler effect caused by rapid movement of a LEO satellite and its compensation are addressed. Latest work in Rel. 17 covers Quality of Service (QoS) aspects of direct access and backhauling, mobility management, and the core network architecture. Standardized interfaces and protocols enable better interoperability across terrestrial, airborne, and space platforms with reduced costs while aiming to wider adoption of NTN-enabled services in the future.

Sustainability is one of the main targets in 5G/6G research and development work and satellites need to play their role. One must ensure space safety, use renewable energy and limited spectrum resources efficiently, and overall improve longevity and recycling of the satellites. Satellites support also access equality enabling satellite services e.g., in the Arctic or remote villages anywhere. A practical example of how satellites already now can support remote work is to use so-called tactical bubbles [3] or local networks and connect with satellite backhaul to outside world. While the ongoing densification of small satellite constellations is expected to improve communication performance, the increasing collision risk with other satellites, as well as space debris, must be handled via a sophisticated space traffic management framework. An important enabler is to be able to detect smaller space debris in a timely manner. This can be achieved using a separate ground-based radar system or a space-based approach by utilizing inter-satellite communication signals [4].

Small satellite constellations will play a key role in future 5G and 6G networks but cannot be yet used with their full potential. This is especially true when considering communication services offered by nanosatellites [5]. While small satellites are cheaper, their limitations to perform complicated operations must be carefully considered. In this direction, different functional splits of the overall network control operations and their locations play a key role. The proper functional splits should encounter the satellite constellation topologies, local computing capability, and resulted end-to-end delays against traffic requirements. Further required developments include implementation of chips with standardized 5G NTN features, adoption of higher frequency bands and more efficient use of spectrum, and development of dedicated hardware for adaptable satellite payloads. A good target is to have software-defined architecture in order to be able to update the satellites and constellations to future needs over the air – and to integrate spaceborne, airborne, and terrestrial systems seamlessly together. This will eventually lead to multi-layer satellite systems that can flexibly support operations in the air, sea, and ground across the globe.

Edge computing (EC) is considered as an essential enabler for many 5G and 6G use cases such as Industry 4.0, IoT, and low-latency reliable communication applications. The space segment is developing in rapid pace in this field with “data centers in orbit” concepts like LEOcloud and OrbitsEdge. While these concepts will most likely find their first applications in, e.g., processing remote sensing data, it is quite obvious that such EC technologies offer large potential for mobile EC applications. 5G SmallSats with integrated MEC functionality would not only provide connectivity

for remote areas but they can also enable many of the new services and applications that rely on low-latency EC. Such integration would tie the satellite segment even tighter with terrestrial software-defined and virtualized communication and computing environment, which would necessitate the development of space aware integrated network orchestration mechanisms.

Software-defined satellite concepts

Various types of radio transceivers form the main assets in a communications satellite. Radio signals serve for communication and, actively or passively, for remote sensing in Earth observation, for example. Since radio payloads have evolved from invariable bent-pipe systems to regenerative transponders and increasing flexibility by using Software-Defined Radios (SDR), phased-array antennas and beamforming, the payloads have been called “software-defined”. But the technological flexibility extends further towards modularization of satellite components with standard interfaces, e.g. CubeSats, virtualization of the infrastructure, structuring of applications as a collection of autonomous microservices, and orchestration and management of the containerized applications and services. Further, satellite edge computing raises cloud-computing capabilities and an Information Technology (IT) service environment to the space. This can be applied for radio transmission optimization, or for onboard analysis of observed data, like hyperspectral images, by using Machine Learning (ML) techniques, for example.

Thus, software-defined payloads that can be adapted to future needs over-the-air can be used to enhance both communications and remote sensing satellite services. In some special cases both can be done using the same radio equipment with so-called joint communications and sensing capabilities. Functionalities that can be adapted include: (1) configuring the channels in the downlink beam by sampling and digitizing the uplink signal into channels that are flexibly allocated; (2) Changing dynamically the coverage areas to provide communication or sensing capabilities to defined areas (requires re-steering of antennas); (3) Updating routing tables and protocols to enable e.g., multi-layer connections. Finally, we can foresee that standardized software-defined satellite systems and software-defined ground systems could be efficiently integrated together. This will enable end users to seamlessly connect to different satellites or base stations and access services anywhere. However, this requires tighter collaboration between current satellite system and terrestrial system manufacturers and operators.

When considering the deployment of mainstream terrestrial IT infrastructure in small satellites several environmental conditions and constraints related to energy and compute resources need to be taken into account while assessing the feasibility of a planned satellite mission. These conditions and constraints include:

- Physical dimensions: satellite payload size, shape, and weight
- Power generation, storage, and distribution: dependence on solar panels
- Power consumption by the computing, memory and storage resources: sets limits on the available computing power
- Time-critical operations: hard deterministic response times requires real-time platforms
- Radio communications: limited bandwidth, long delays, intermittent connectivity, and interference
- Space weather in the forms of radiation and charging (e.g.,, magnetic storms), varying surface temperatures (±150 ºC), vacuum
- Terrestrial weather affecting electromagnetic wave propagation (clouds, rain, and snow) in communications and Earth observation
- Mechanical shocks and vibration: during launch and ejection from the rocket
- Space dust and debris (also self-inflicted): incurs collision hazards and equipment loss
- In-orbit lifetime: limits the mission duration (for example, with CubeSats typically shorter than 3–5 years)

- Security: malicious software attacks, jamming, etc.

Software frameworks and virtualization technologies

As was discussed in the previous section, a typical small satellite environment is characterized by low size, weight, and power (SWaP), various risks of equipment failure, and the need for fast reaction to events. These constraints and requirements call for power efficient and fault tolerant HW and SW platforms with real-time properties.

Realtime operating systems that have been designed for embedded systems and thus have been popular in small satellites include eCos (Embedded Configurable Operating System), freeRTOS, Rodos, RTEMS (Real-Time Executive for Multiprocessor Systems), RTLinux, µClinux, and VxWorks. Though, Linux (with vanilla kernel and U-Boot) will do if about 10 ms response time is sufficient. Then harder real-time operations, e.g., regarding the Attitude Determination and Control System (ADCS) may be delegated to specialized microprocessors.

The required reliability and high availability can be tackled by relying on cold, hot, and warm standby recovery mechanisms, based on HW and SW redundancy. This represents rudimentary forms of infrastructure abstraction and resource pooling, which are also the foundations of more general virtualization technologies, like hypervisors, containers, unikernels, and microVMs. Hypervisors (aka Virtual Machine Monitors) run virtual machines on bare metal (meaning that there is no operating system between the virtualization software and the hardware, e.g., KVM and Xen) or they may be hosted by an underlying operating system (e.g., QEMU, VirtualBox, VMware Workstation). Containers are more light-weight user-space instances that use OS-level virtualization (e.g., Docker and FreeBSD Jails). Further, in the urge for decreasing footprint, unikernels are single address space images in which a minimal set of OS functions is compiled as libraries, and which thus do not need any hosting OS. Finally, MicroVMs (e.g., Firecracker on top of KVM) aim at offering the rapidity and isolation benefits of containers and virtual machines, respectively. The image size is typically ca. 10 % of a Docker image.

The above-mentioned server-based solutions are complemented by the serverless architecture, which is based on remote launching and execution of small single purpose stateless functions on demand. This concept, also called as Function-as-a-Service (FaaS) is proposed for the LEO edge in [6]. Open source serverless frameworks include OpenFaaS, Knative, OpenWhisk, and Fission. More recently, there has been discussion about challenges in implementing scalable low-latency distributed services with stateless functions. For example, transactional stateless functions usually utilize external database to store or update the current state of the service and while this makes handling failovers and load-balancing much easier, this also tends to increase latency considerably. Thus, to overcome these problems, there are new proposals about stateful functions but most of these developments are still at early phases [7, 8].

EC with LEO satellites pose also problems related to frequent satellite handovers such as state migration where the current state of service or offloaded computation must be transferred to new MEC resource. These problems are similar to vehicular EC problems but, e.g., limitations in intersatellite communications can make it more challenging. Currently quite common stateless microservices and more recent stateless functions that rely on storing service or application state to an external database are also problematic with state migration – state migration would be easy if the database is located at ground segment but that increases latency in storing each transaction, on the other hand, using thin streamlined local database would transfer the migration problem to database level. Furthermore, such arrangement would invalidate some of the reciliency benefits of having the current state stored in (resilient) database.

Furthermore, starting and stopping services due to, e.g., non-identical service needs in different geographical areas, would need some consideration: starting and stopping classical VMs and containers take significant amount of time and, if loaded from ground segment, consume valuable communication resources. One solution would be keeping the services running all the time but that would be wasteful with limited SmallSat resources. MicroVMs provide some solutions to this with startup times of range of 10 ms or so, and with considerably smaller VM image size. For some type

of services, it might be worthwhile to consider lighter weight distributed, resilient computing solutions based on, e.g., Actor Model [9, 10].

Table 1 lists some Unikernel implementations along with the supported programming languages and virtualization environments (compiled from [11, 12]).

Table 1: Unikernel implementations [11, 12]

Unikernel	Programming languages	Supporting platforms
ClickOS	C++	Xen
Clive	Go	Xen, KVM
EggOS	Go	bare metal
HalVM	Haskell	Xen
HermitCore	C, C ++, Fortran, Go	Xen, KVM, x86_64
IncludeOS	C++	KVM, VirtualBox
LING	Erlang	Xen
MirageOS	OCaml	Xen, FreeBSD, POSIX
Nanos	C, C++, Go, Java, Scala, Rust...	Xen, KVM
OSv	Java, C, C++, Node.js	Xen, KVM, VMware, VirtualBox
Rumprun	C, C++, Erlang, Go, Java, Javascript, Python	Xen, KVM, bare metal
RustyHermit	Rust, C, C++, Go, Fortran	Xen, KVM, x86_64
UniKraft	C, C++, Go, Python, WASM, Lua	Xen, KVM

The above-mentioned virtualization solutions alone, without a management and orchestration system, do not provide the resilience and reliability properties required in a satellite system context. The deployment of containerized workloads among the compute resources in a satellite, in a satellite constellation, or in the whole satellite system, including the ground segment, can be automated by orchestrators like Kubernetes (also called as K8s) or Nomad, for example. One of the lightweight versions of K8s, called K3s [13], has been planned for use in satellite clusters in a joint effort of Hypergiant LLC, SUSE RGS, and DoD PlatformONE [14]. Here Ansible play would serve as a continuous integration and continuous delivery (CI/CD) solution while using Raspberry Pi SBCs for the HW platforms. Alternatively, Nvidia Jetson Nano, or similar, could be used for the SBCs. It should be noted that all too common careless CI/CD practices should be avoided in satellite context and integration of security assurance in the process (DevSecOps) is essential.

In addition to the general-purpose ICT solutions for the software-defined payloads, there are also open-source satellite SW frameworks targeted for traditional satellite platform operations and exploration missions, with support for standardized Consultative Committee for Space Data Systems (CCSDS) telemetry and command messages (for example, see [15]). Such frameworks, of which many have actual flight heritage, include CORDET Framework, CubedOS, KubOS, NanoSat MO Framework (NMF), NASA/Core Flight System (CFS), and NASA F' (F Prime). The NASA's frameworks, for example, can be easily experimented with on small form factor SBCs, like Raspberry PIs. However, these frameworks do not natively support the above-mentioned virtualization techniques.

Cybersecurity for software in the orbit

Challenges

Software-defined concepts increase the security risks of satellites by bringing vulnerabilities that are common in software related cyber-attacks, such as malicious software, integrity violations, backdoors, and misconfiguration of software. Consequences from these attacks may lead, e.g., to unavailability of satellite services or communications, spoofed situational information, leakage of information belonging to the satellite operator or payload provider, or destruction of satellites. Satellites are alluring targets for advanced adversaries as they can give clear monetary benefits to extortionists, and as hostile nations or competitors may have the interest to decapitate them.

Cybersecurity attacks threaten different phases of the life cycle of software. The fundamental challenges for software-originated threats include the difficulty to determine the trustworthiness of software as supply chains are long, complex and susceptible to attacks. Similarly, proprietary and open-source solutions are mixed, code development is error-prone; and code review and assurance is hard. In the space context, software updates to critical satellite platforms take time, thus, unverified software cannot be uploaded as it may make satellites completely unusable [16]. This longer verification time in turn means that software patches are sent in delay and the adversary has some time between the publication of a security vulnerability and patching of it. Zero-day vulnerabilities are also possible for advanced adversaries. Software tailored for satellites is not used everywhere and hence won't be so well security tested.

Software-defined concepts provide flexibility but also complicate the control and, hence, often require automation, AI, and new design principles, like intent-based networking, that make configuration easier. However, this complexity opens a door for new security vulnerabilities. It is more difficult to verify that complex policies and control solutions are error-free and work correctly also in hostile situations.

Different security challenges, attack vectors, and vulnerabilities are highlighted in Table 2 alongside potential countermeasures. The table aims to pinpoint the unique challenges within the space domain.

Table 2. Software-related security challenges with satellites and mitigations

Security challenges	Impact of the space domain	Security approaches
Malicious supply chain for software	Niche satellite-specific software is less tested and less mature.	Trust and integrity verifications before deployment and in the orbit. Software signing infrastructure.
Misbehaving tenant software	Resource limitations in satellites may limit controls.	Access control-based segregation of software. Security monitoring to detect attacks. Security architecture distributing intensive processing to the ground.
Software vulnerabilities	Updates take time due to longer verification of satellite-tailored programs. Software, reconfigurability, and automation often complicate design.	Software architecture based on mature products minimizes patching. Software-defined security as a concept to make satellites' security easier to update.
Denial of control signals	Jamming against ground-to-satellite communications	Architectural resilience; protocol robustness and security; monitoring and responses on the ground. Routing security for inter-satellite communications.
Control channel integrity and confidentiality	Security handshakes may impair latency that is already significant with satellites.	Communication security. Protocols minimizing overhead.
Vulnerabilities in systems on ground	Intrusions in the ground stations, connected systems, or communication terminals	Security monitoring approaches, e.g., AI-based threat detection. Develop satellite system-specific security technologies.
Vulnerabilities in satellite platforms	Optimization of platforms and operating systems may impact security features	Security-by-design

Solutions

Segregation of software (tenants), which are responsible for handling assets of different organizations, or have different responsibilities within the satellite, is an important principle. Access control from two different perspectives: a) hardware, firmware, and b) operating system, virtualization and network slicing levels is needed. However, different software components often share resources, such as network services or memory, that open attack paths. Reactive security controls are needed to detect misbehaviour. Security monitoring and software integrity verifications before deployment and at run-time are necessary.

The integrated satellite-terrestrial architecture will adopt 3GPP solutions, but also require adaptations and approaches that are tailored for satellite infrastructures. Communication security

can be, thus, in part based on 3GPP's approaches. Different software-defined concepts typically require their own application layer solutions to protect signalling and software delivery. Authentication infrastructure and security protocols versions, such as the current version of the standardized Transport Layer Security (TLS 1.3), with minimized overhead are, hence, necessary.

Software-defined security [17,18] is a concept where security functions are implemented in software and hardware, and where control and data flows are decoupled. Essentially, software-defined security relies on the virtualization of software functions and the centralization of security intelligence. The approach has the potential to make satellites more easily updatable to respond to new emerging threats. However, it may also introduce additional overhead and complexity due to added virtualization layer and control communications. Security solutions are restricted by the capabilities and resources in the satellite platforms. For instance, processing-intense solutions like security monitoring and AI-based threat detection are limited by memory, processing, energy, and available channel capacity. Consequently, many security functions locate in the ground systems. Other limitations emerge from satellite hardware and software platforms (e.g., FPGA, Linux, virtualization) that may be optimized for use in space. Satellites are still quite specialized devices and there is a lack of advanced security products that are tailored for them, which need further research.

Summary

In this paper, we discussed about proving 5G and 6G communication and edge computing services by means of low orbit small satellites. The interoperability and integration with terrestrial mobile networks would necessitate adoption of 5G NTN while in-orbit MEC would extend the coverage of demanding 5G applications to remote areas. The key-enablers in this development would be advanced software defined SmallSats that would not only provide flexible communications payload but also virtualized computing resources for virtualized network functions and EC tasks. However, LEO SmallSats pose various limitations that should be considered when virtualization and orchestration mechanisms are selected. The current mainstream cloud computing solutions are not necessarily ideal in this environment as they are designed for static computer centre surroundings. Thus, we revied some alternative solutions that could be better suited with SmallSat constraints. Finally, we took an overview of security issues with software defined satellites and the importance of proper security procedures in developing and deploying software components in this context cannot be overemphasised as the failures can lead even to loss of satellite.

References

[1] M. Höyhtyä, M. Corici, S. Covaci and M. Guta, "5G and beyond for new space: Vision and research challenges," in *Proc. ICSSC*, Oct. 2019.

[2] F. Rinaldi et al., "Non-terrestrial networks in 5G & beyond: A survey," *IEEE Access*, vol. 8, pp. 165178–165200, Sep. 2020.

[3] M. Heikkilä et al., "Field trial with tactical bubbles for mission critical communications," *Trans. Emerg. Telecommun. Tech.*, 2021.

[4] A. Anttonen et al., "Space debris detection over intersatellite communications signals," Acta Astronatica, 2021.

[5] N. Saeed et al., "CubeSat communications: Recent advances and future challenges," IEEE Commun. Surveys & Tut., 2020.

[6] T. Pfandzelter, J. Hasenburg, and D. Bermbach, "Towards a Computing Platform for the LEO Edge," in Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '21). Association for Computing Machinery, New York, NY, USA, 43–48.

[7] M. de Heus, K. Psarakis, M. Fragkoulis, and A. Katsifodimos, "Distributed transactions on serverless stateful functions," in Proceedings of the 15th ACM international conference on distributed and event-based systems, 2021, pp. 31–42.

[8] C. Puliafito, C. Cicconetti, M. Conti, E. Mingozzi, and A. Passarella, "Stateful function as a service at the edge," *Computer*, vol. 55, no. 9, pp. 54–64, 2022.

[9] R. Hetzel, T. Kärkkäinen, and J. Ott, "MActor: Stateful serverless at the edge," in Proceedings of the 1st workshop on serverless mobile networking for 6G

communications, 2021, pp. 1–6.
[10] S. N. Srirama, F. M. S. Dick, and M. Adhikari, “Akka framework based on the actor model for executing distributed fog computing applications,” Future Generation Computer Systems, vol. 117, pp. 439–452, 2021.
[11] S. Chen and M. Zhou, “Evolving Container to Unikernel for Edge Computing and Applications in Process Industry,” Processes. 2021; 9(2):351.
[12] Awesome Unikernels. [Online]. Available: https://github.com/uniqernel/awesome-unikernels
[13] K3s, Lightweight Kubernetes. [Online]. Available: https://k3s.io/
[14] Hypergiant and SUSE RGS, taking Kubernetes to the final frontier, 2021. [Online]. Available: https://www.suse.com/c/hypergiant-and-suse-rgs-taking-kubernetes-to-the-final-frontier/
[15] D. J. F. Miranda, "A Comparative Survey on Flight Software Frameworks for ‘New Space’ Nanosatellite Missions," Journal of Aerospace Technology and Management, vol. 11, 2019.
[16] R. Merriam. (2016) Software update destroys 286$ million Japanese satellite. [Online]. Available: https://hackaday.com/2016/05/02/software-update-destroys-286-million-japanese-satellite/
[17] Al-Ayyoub, Mahmoud, et al. "Sdsecurity: A software defined security experimental framework." IEEE international conference on communication workshop (ICCW). IEEE, 2015.
[18] Iqbal, Waseem, et al. "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security." IEEE Internet of Things Journal, 2020.