Transfer of Session State Between Satellites in a Space Information Network

Anders Fongen
Norwegian Defence University College (FHS)
Lillehammer, Norway
email: anders@fongen.no

***Abstract*—In a Space Information Network (SIN), there will be frequent handovers of service connections between ground-based clients and orbiting satellites above. In the case where the state of the client sessions are represented in the satellite computers, the handover operation becomes considerably more complicated and expensive, and several methods for the transfer of client session state are investigated in this article.**

***Keywords*—LEO satellites; space information networks; session state; state transfer; mobile computing.**

I. Introduction

The term *Space Information Network* (SIN) describes a set of satellites that cooperatively offer services for information processing and sharing, as well as traditional communication services. SIN is regarded as a natural evolution of satellite services, coming from radio mirrors in geostationary orbit and Low Earth Orbit (LEO) constellation for communication services (e.g., Iridium) [1], [2].

In a series of previous publications, different aspects of SIN operation (architecture [3], security [4], cache management [5] and routing [6]) have been addressed. This article will focus on the management of *session state* as a client engages the SIN in information processing tasks. During handover, the existing state of the session/dialog needs to be made accessible to the next satellite. This can be formulated as a *process migration* problem, well known in the field of Distributed Computing.

While the general problem of stateful process migration has no practical solution, a SIN offers properties which, when taken into account, can reduce the general problem into something for which a practical solution can be found. The two most important properties are:

1) The predictability of satellites' relative position at any time.
2) The uneven density distribution of the world's population.

The perspective of the presented analysis is that from Distributed Computing. Technical and physical properties of satellites, related to energy management, beamforming, modulation, coding, jamming resistance etc., are not taken into account.

The remainder of the paper is organized as follows: Section II provides a discussion on the organization and representation of session state. Section III discusses how Docker Swarms and Kubernetes can provide a framework for service code management. Section IV explains briefly the software simulation model used for the experiment. Section V analyses how session state representation can be handled during a handover operation. Section VI shows the paging arrangement evaluated in the experiment, while Section VII discusses some remaining issues in the results. Section VIII sums up the text and identifies remaining research problems.

II. Representation of session state

During a dialog between a service provider and a client, there always exists a context within which the next transaction will be interpreted. The representation of this context is called the *session state*, which is a collection of several data elements in the service provider and the client (CPU register content, user and system space memory values). Together, these elements allow the dialogue to form a coherent chain of transactions, without relying on the CPU instruction pointer (i.e., the transactions can all start at a common execution entry point).

In the well understood field of Web Services, the state is most often represented by the content of the Application Protocol Data Unit (A-PDU) (i.e., the content of the URL, HTML page and cookies), as well as a *session object* in the server storage. Since the HTTP protocol is stateless, every transaction has the same starting point, which reduces the problem of state maintenance considerably. Besides the ability for the web browser to maintain dialog state, many programming environments offer the server to maintain a session object which is preserved across service invocations, sometimes even across server incarnations ("restarts").

This model can be ported to a SIN environment, where all satellites of the constellation offer the same service, and the session object is made accessible to the new service provider during the handover operation. The session object can be stored in one dedicated satellite or on the ground to which there exist a route, distributed across a small set of satellites, or moved to the new serving satellite in its entirety.

Although these methods seems viable, they should be evaluated in terms of their communication requirements. A SIN will always be under-provisioned with regard to the Earth's population, so a close look at scalability properties is essential to the analysis.

III. Replication of service code

Following a handover operation, the client expects to find the same set of services in the newly connected satellite. The service code must be deployed and maintained in every satellite, which is not expected to consume much communication resources. The handover operation resembles what would be called *Process Migration* in the Distributed Systems literature.

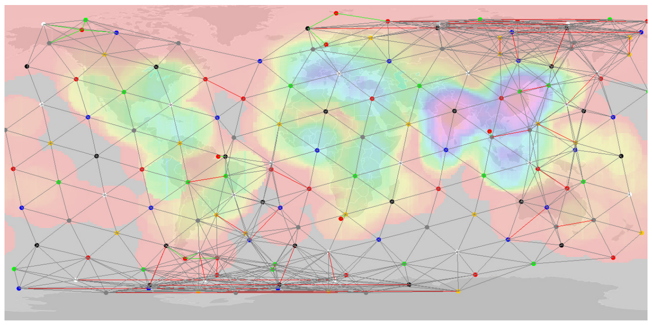
Figure 1. Screenshot from the satellite constellation model.

The Docker architecture has been studied for this purpose. Even though Docker was not designed for a SIN environment, the *Docker Swarm* [7] platform has appealing properties with regard to SIN operation.

In a Docker Swarm, every participant is an endpoint for a given service. All participants cooperate through an *overlay network* and distribute incoming service requests for the purpose of load balancing and failure recovery. An arrangement of a SIN as a Docker Swarm will allow clients all over the world to compete for the same computing resource pool, regardless of their location. Idle resources in satellites momentarily flying over inhabited areas may be employed to offload busy satellites. The price for this is the traffic volume in the overlay network, which will compete for the communication capacity in the constellation. Please observe that a Docker Swarm only supports *stateless* services.

IV. The software model

The results presented in this article are based on a software simulation of a satellite constellation. A screenshot from the model is shown in Figure 1. The constellation consists of 150 satellites at 500 km altitude.

The colored backdrop in the figure indicates the population density inside the satellite footprint at a given location, based on gridded population data from NASA [8]. This data set has also been used to calculate the graph in Figure 2, discussed in Section V-D.

V. Strategies for session state maintenance

The maintenance and transport of session state between satellites during a handover operation can be divided into three different techniques:

1) Keep the session object separate from the service component, e.g. in a computer on the Earth's surface, while the service client provides a reference to its existence through, e.g., a cookie value.
2) Move the entire session object proactively from the outgoing satellite to the oncoming satellite during the handover operation.
3) Move elements of the session object from former satellites to current satellite *on demand*, so that the session object is effectively distributed across a trail of past connected satellites.

These alternatives will now be discussed in more detail.

A. Separate and stable session object

Keeping the session object in the same location contradicts the entire idea of using satellites as units for information processing and sharing, and reduces the constellation to an ordinary communication network for information routing in a multi-hop infrastructure. This alternative is therefore abandoned for not meeting the required objectives.

B. Proactively moving entire session object

From a functional perspective, the proactive moving of the entire session object will meet the requirement to offer access to the session object elements at any time, with the shortest possible access latency. From a non-functional perspective, this alternative will consume unnecessary many resources and will not scale well. The reason for this is twofold: (1) The people of the Earth are concentrated in small regions, and a group of clients are likely to employ a small fraction of the satellite constellation, which will have to carry a majority of the session objects while the rest of the satellites are idling. (2) The links between satellites in these densely populated areas will be disproportionately loaded with traffic related to session object traffic, while other links in the constellation will be left unemployed.

These two observations both cause the SIN to scale poorly, due to the ineffective workload distribution of both storage capacity and link capacity. This alternative is taken into account in the traffic simulation results shown later, mostly for the sake of being a baseline for comparison.

C. Move session object elements on demand

The third alternative up for discussion is to defer the movement of session object elements and fetch them from past satellite connections on demand. Access operations to the session object elements are not expected to be uniformly distributed across the elements, but follow a *Scale Free Distribution* (SFD) [9], also known as Zipf's law. According to SFD, the frequency (f) of accesses to an element is inversely proportional to the *rank* (r) of the element. Applied to this use case, SFD predicts that the most frequently used element will be accessed twice as often as the second most frequently used element, three times more often than the third most frequently access element, and so on. Mathematically, this may be expressed as

$$f = \frac{a}{r} \tag{1}$$

where a is given a value so that

$$\sum \frac{a}{r} = 1 \tag{2}$$

Assuming SFD, the elements of a session object are then expected to form a "wake" along the row of past connected satellites, where the least frequently used elements are stored in the earliest connected satellite. Frequently accessed elements may be accessed faster because their access path is shorter. This arrangement is quite similar to the concept of Virtual

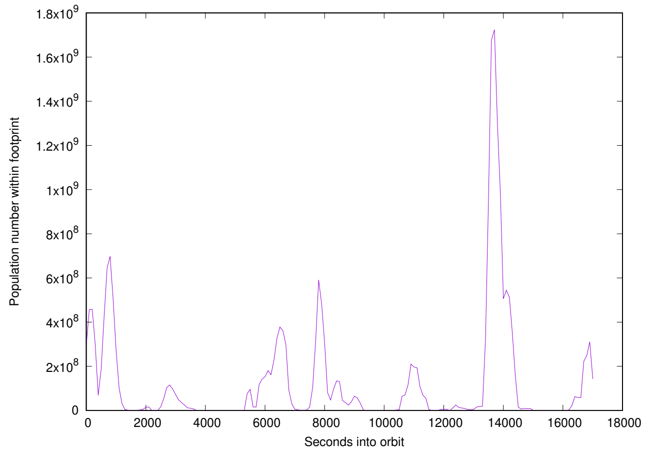

Figure 2. The population number inside the footprint of a satellite during three subsequent orbits.

Memory in operative system design, where the placement of a data element in a *storage hierarchy* is decided by its access frequency. The implementation of this arrangement is inspired by the OS design of Virtual Memory.

D. Employment of idle satellites

The distribution of the Earth's population density is highly uneven. The graph in Figure 2 shows the population size inside the satellite footprint over three consecutive orbits. The population size represents an estimate for the satellite's workload at that location. As shown in the figure, the "rush hour peaks" are so brief that most busy satellites will probably have an idle neighbor, or at least a much less busy one. This observation suggests that a distributed representation of the session object will utilise idle resources better than a session object kept in only one busy satellite. A simulation based evaluation of this scheme will be presented later in the article.

VI. Session state management based on Paging

Memory Paging in Distributed Systems, as it is known from Operating System design, is an on-demand replication technique. It is well understood under the name *Distributed Virtual Memory* [10]. For the purpose of on-demand replication of session state, the technique will be investigated with the following requirements in mind:

- Consistency and convergence
- Latency of page fault handling
- Volume of generated traffic

In this proposed solution, each service in a satellite will maintain their own *Page Table* (PT) for the storage used to keep the session state. Each entry in the PT will contain a reference to a *Page* stored in a *Page Frame* (PF) in one satellite. Once a page is to be retrieved, the PT is inspected to see if it is stored locally. If it is, the content is returned to the caller. Otherwise, the page is located in a distant PF and copied to a local PF (by the data routing service using the inter-satellite links), then removed from the distant PF. The PT is then updated and the page content returned to the caller. The resulting distribution of pages based on this arrangement is shown in Figure 3.

If the page in question does not exist, it is created in a local PF and referred to in the PT. During handover, the new satellite will inherit the PT, but not the content of the PFs. Subsequent page requests will therefore cause the frequently used pages to be moved first, while other pages may never be moved, and will be deleted by the end of the session.

The distribution of pages has been simulated with these parameters:

- 5 handover operations, involving 6 satellites
- 1000 accesses to the session state object to each satellite
- PT has 1000 entries (for page references)
- Pages are accessed according to SFD

The resulting distribution is shown in Figure 4. The sum of all numbers (806) shows that far from every page in the session state object were ever accessed, and existing pages remaining in PF of previous satellites indicate that they were never accessed since that satellite's time of service.

TABLE I
Distribution of pages across satellites before each handover.

satellite total	0	1	2	3	4	5
336	336					
499	172	327				
608	118	168	322			
704	91	113	157	343		
763	67	94	108	164	330	
806	54	73	86	110	173	310

A. Scalability properties

The number of hops for pages being moved through a path of satellites is chosen as an indicator of the scalability properties of the chosen session state management arrangement. Therefore, the number of single hop movements of pages will be analysed under the scenario described in the previous section. The resulting number from the proactive and the on-demand replication will be compared and reported.

The distribution of the session state pages across the current and past satellites was measured just before a handover operation, after 1000 access operations. The numbers are shown in Table I. From these numbers, it is possible to calculate the total number of page movements across inter-satellite links during the scenario of 5 handover operations with 1000 access operation between each.

For the proactive page movement method, the total number of page movements is 2910 (the sum of the 5 first numbers in the "total" column). For the on-demand method, the total number is 1162. This means that the on-demand method consumes only 40 % of the communication capacity required by the proactive method. These numbers are also shown in Figure 5.

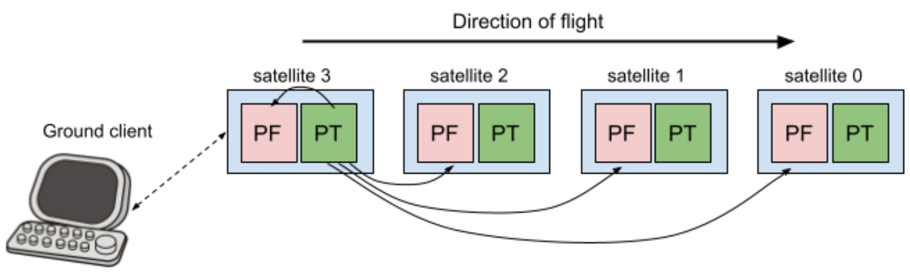

Figure 3. Pages are distributed in PFs along the trail of satellites, and referenced from a single PT.

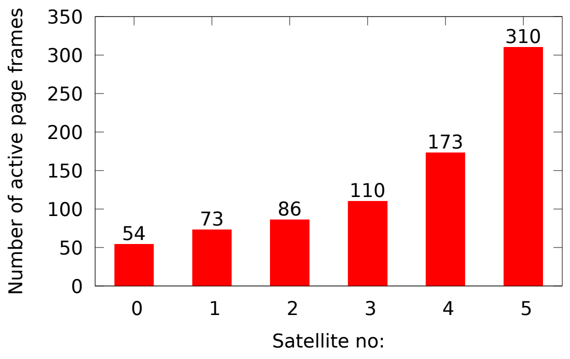

Figure 4. The distribution of session state pages after 5 handover operations.

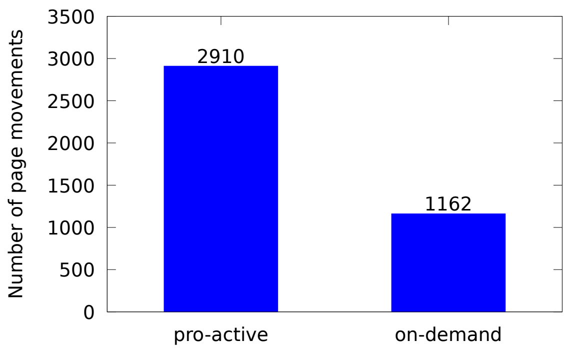

Figure 5. The number of session state page movements for pro-active and on-demand method

VII. Discussion

The results from the simulation experiment support the assumption that an on-demand paging arrangement saves a lot of communication resources, even though page retrievals through several hops may increase the access latency.

A. *Inclusion of population density*

The population density should be taken into account for our scalability analysis, as pointed out in Section V-D. The population density data was included in the early stage of the experiment and showed the expected results. On the other hand, the results are highly variable depending on the chosen location on the Earth, so an average value will not provide useful information, and are not included in the article.

B. *Session state as a file/memory system*

The data elements of a session state object have varying sizes and do not fit well with the arrangement of pages with fixed size. Two possible improvements on this matter are:

- Provide access to the session state pages through a file API. On a Linux computer, this requires the construction of a *Block Device Driver*, which involves kernel module programming. This option is left for future investigation.
- Provide access as a virtual memory block. This solution requires modification to the virtual memory management as well as access to the PT and related structures inside the CPU, and involves modifications of the OS kernel itself. This option is therefore considered infeasible.

VIII. Conclusion

In the presented article, the problem related to stateful service components in a cloud of satellite-based services providers has been analysed. The proposed solution has been evaluated and found to be sound and effective. Remaining problems include byte-level access to session state pages, and solution to possible satellite faults.

References

[1] S. Briatore, N. Garzaniti, and A. Golkar, "Towards the internet for space: Bringing cloud computing to space systems," in *36th International Communications Satellite Systems Conference (ICSSC 2018)*, 2018, pp. 1–5.
[2] L. Bai, T. de Cola, Q. Yu, and W. Zhang, "Space information networks," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 8–9, 2019.
[3] A. Fongen, "Application services in space information networks," in *CYBER 2021*. Barcelona, Spain: IARIA, Oct 2021, pp. 113–117.
[4] A. Fongen, "Trust management in space information networks," in *SECURWARE 2021*. Athens, Greece: IARIA, Nov 2021, pp. 14–18.
[5] A. Fongen, "Cooperative caching in space information networks," in *INTERNET 2022*. Vienna, Italy: IARIA, May 20212, pp. 1–5.
[6] A. Fongen, "Population-based routing in leo satellite networks," in *MOBILITY 2022*. Porto, Portugal: IARIA, June 2022, pp. 1–4.
[7] A. Modak, S. D. Chaudhary, P. S. Paygude, and S. R. Ldate, "Techniques to secure data on cloud: Docker swarm or kubernetes?" in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 7–12.
[8] "Gridded population of the world v.4.11," [Online; retrieved 09-Nov-2022]. [Online]. Available: https://sedac.ciesin.columbia.edu/data/collection/gpw-v4/sets/browse
[9] A.-L. Barabasi, *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume Books, April 2003.
[10] C. Morin and I. Puaut, "A survey of recoverable distributed shared virtual memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 9, pp. 959–969, 1997.