Data Sharing Services in a Space Information Network

dr. Anders Fongen
Norwegian Defence University College, Cyber Defence Academy, Lillehammer
email: anders@fongen.no

EMERGING 2023, Porto, Portugal, September 2023

Presenter's bio

Anders Fongen

- Associate Professor, Norwegian Defence University College
- Field of research: Distributed Systems, Networking security
- PhD in Distributed Systems, Univ. of Sunderland, UK, 2004
- Career history
  - 7 years in military engineering education (Associate Professor)
  - 10 years in defence research (Chief Scientist)
  - 8 years in civilian college (Associate Professor)
  - 11 years in oil industry
  - 6 years in electronics industry

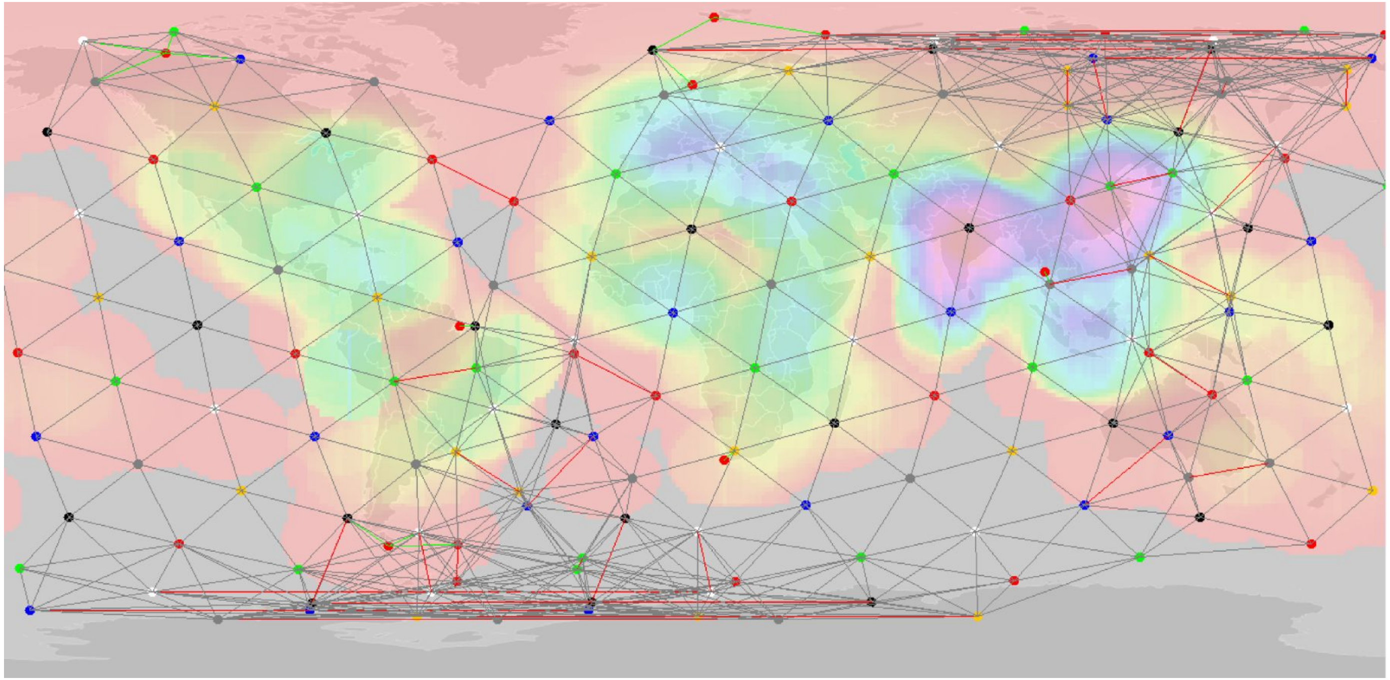
Introduction

- The evolution of satellite communication?
  - Application services (“Cloud computing in space”)
  - Higher system complexity (larger state space)
- What are the advantages?
  - Very low latency (as low as 3 ms)
  - Global coverage
- Interesting properties of a Low Earth Orbit (LEO) system:
  - Predictability of positions, links, routes and workload
  - Long idle periods (due to inhabited surface) mixed with traffic peaks
- Viewed as a problem of *Distributed Computing*
  - *having a set of distinct properties*

What is a SIN (Space Information Network)?

- A collection of communicating LEO satellites
- Able to serve terrestrial/airborne client
  - Communication services (e.g., IP transport, VoIP, Publish-Subscribe comm.)
  - Discovery Services (DNS, Service Brokering...)
  - Storage Services (Content Distribution Network, caching, session states)
  - **Application Services** (Collaborating editing, Situational awareness ...)
- Resource constrained / disadvantaged
- **Predictable workload and link availability**
- “Mobile” system: Stationary clients, mobile infrastructure
- Rapid hand-over of client connection and *client state*

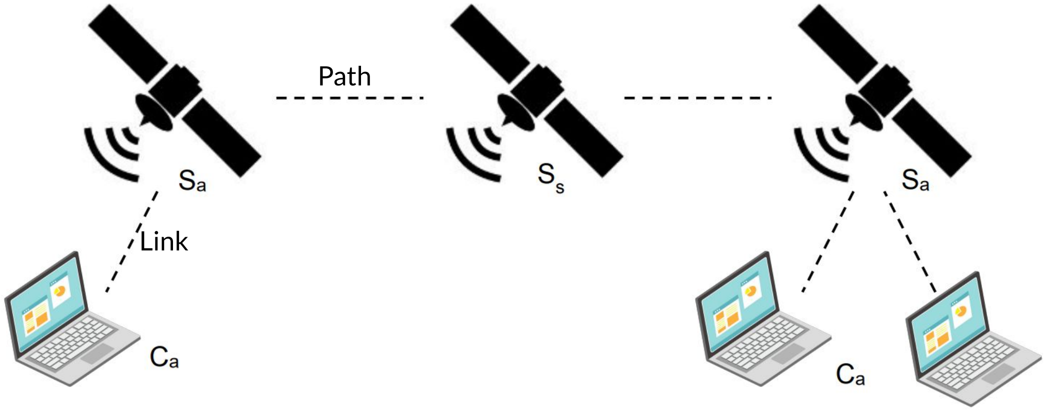
Population “heat map” from satellite footprint



Data sharing in N-layers constellation

Problems:

- Access method
  - Shared memory
  - Service interface
- Sharing semantics
  - Protection, transactions
  - Update ordering
  - Update notification
- Handover management
  - New service endpoint
  - Migration of data
- Relative position *Sa-Ss*
  - Minimum access cost



Ca - Application Client
Sa - Application Server
Ss - Sharing Server

Shared data: access methods

1. Access like a *memory cell*
   a. Abstract and “beautiful”
   b. Lacks protection from race conditions (need separate mutexes)
   c. Lacks update ordering, update notification
   d. No error handling
2. Access through a *service interface*
   a. Slower, need interface stub, parameter serialization, etc.
   b. Offers meaningful abstraction, synchronization, protection and notifications
   c. Meaningful error handling

We choose alt. 2 (Distribution transparency was never a good idea)

Mobility properties

- Handover (approx every 15 min.)
  - Requires *Ca* to find a new *Sa* (link connection)
  - May advice all *Sa* to find *one* new *Ss* (can be planned)
  - Requires all *update listeners* to be updated (listener group dynamics)
- Migration of *Ss* shared data during handover
  - Simplest solution: Migrate all data to new *Ss* between service invocation
  - Scalable solution: Migrate data element *on demand*
    - Why? Because the shared data elements are accessed with different frequencies
    - -> **Scale Free Distribution**

Methods for shared data management

1. Keep one copy of shared data in a stable and reachable location (e.g., on the ground)
    - defeats the purpose of a SIN
2. Copy entire shared data to the oncoming satellite
    - reduces access latency, but creates unnecessary network traffic
    - creates uneven workload of satellites and links (due to population distribution)
3. Copy shared data elements to oncoming satellite **on demand**
    - creates a balance between access latency and copying traffic

On demand migration: Scale Free Distribution

Access operations to shared data elements are assumed to follow a *scale-free distribution*,
where a few elements are often accessed, others less often.
Inversely proportional to their *rank*.

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

Where a is given a value so that

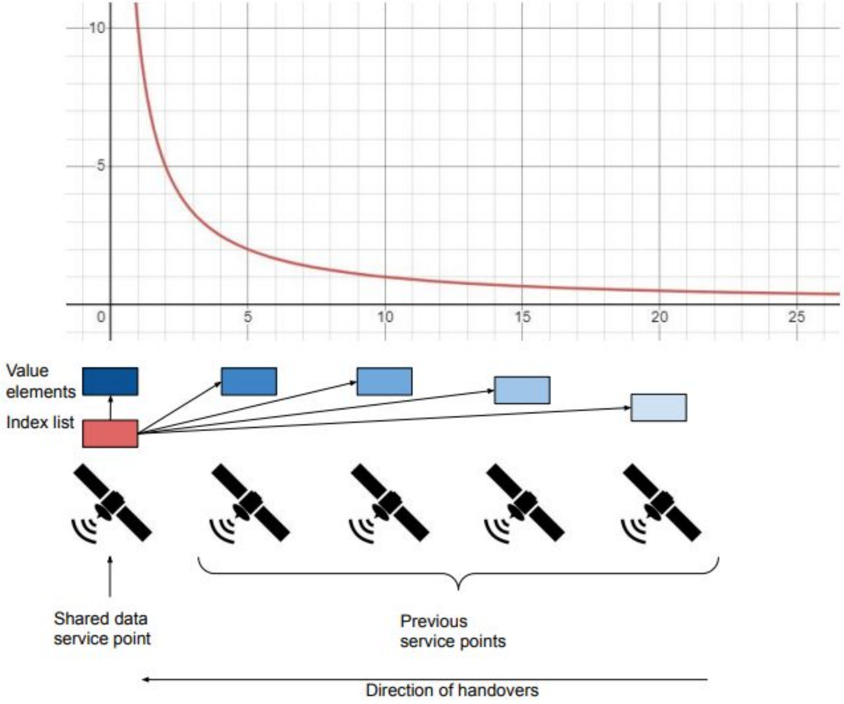
$$\sum \frac{a}{r} = 1$$

On-demand copying of shared data elements will reduce the number of copy operations.

We will arrange the shared data as a hash table of pointers to named-value data elements. The pointer value identifies both *satellite* and *memory address.*

Only the list of pointers are proactively migrated, the shared value data is migrated on demand.

Shared data, distributed by access frequency



Performance of on-demand migration

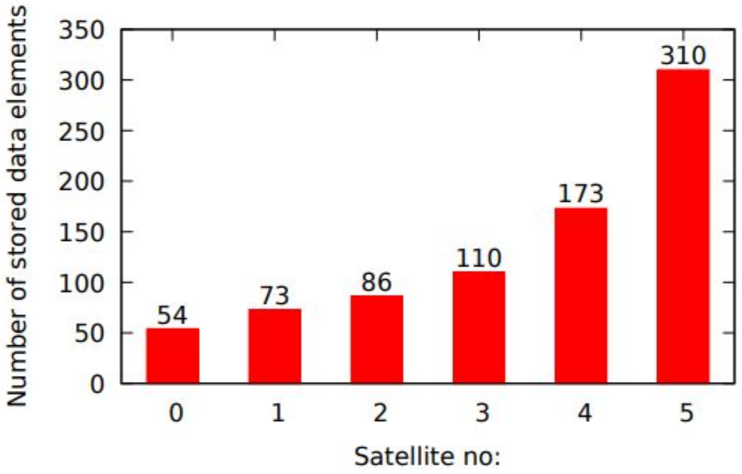
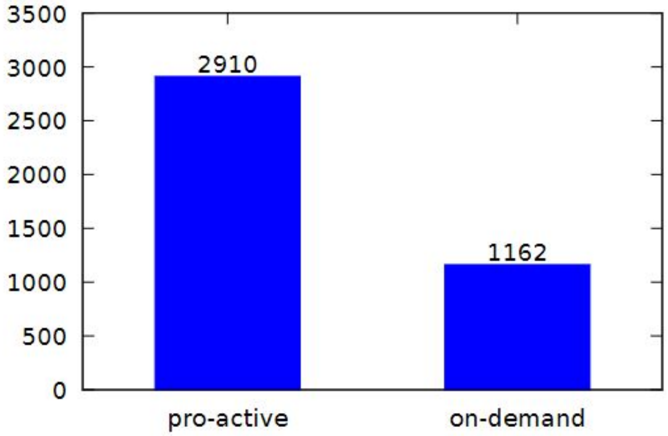


Figure 4. The distribution of shared data elements after 5 handover operations.

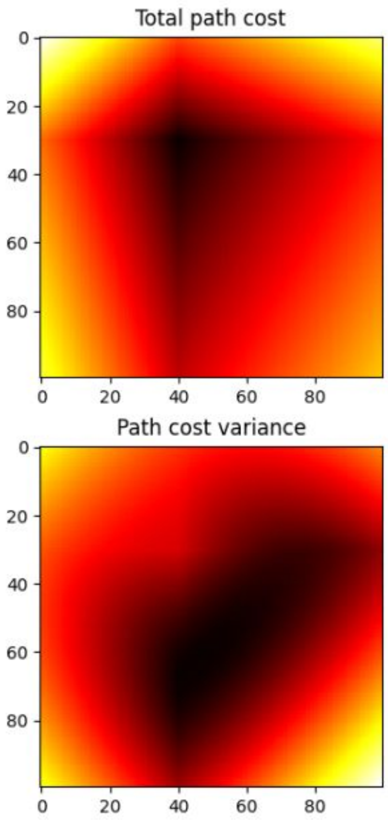


Best location for the *Ss* instance

The best location for the *Ss* is where

- the total path cost for all *Sa* is the minimum
- the variance of path cost between the *Sa* is the lowest

The two heat maps shown here shows these values for a group of 5 *Sa* (with different access frequencies to the *Ss*) and 100x100 possible locations of *Ss*.



Conclusion

The problem: **How to best organize shared data in a SIN, given the problem of frequent handovers?**

- Shared data should be exposed through a service interface, to maintain useful semantics for protection and update ordering.
- Elements of shared data are assumed to be accessed according to a **scale-free distribution**
- During a handover, the *index list* are migrated, not the entire value set
  - then, the values are migrated **on demand**.
- On-demand migration of value elements generate **60% less** network traffic.
- The best location for the *Ss* is a solvable problem

Thank you for your attention, any questions?