Design elements for a Space Information Network Operating System

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

ICSEA 2023, Valencia, Spain, October 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

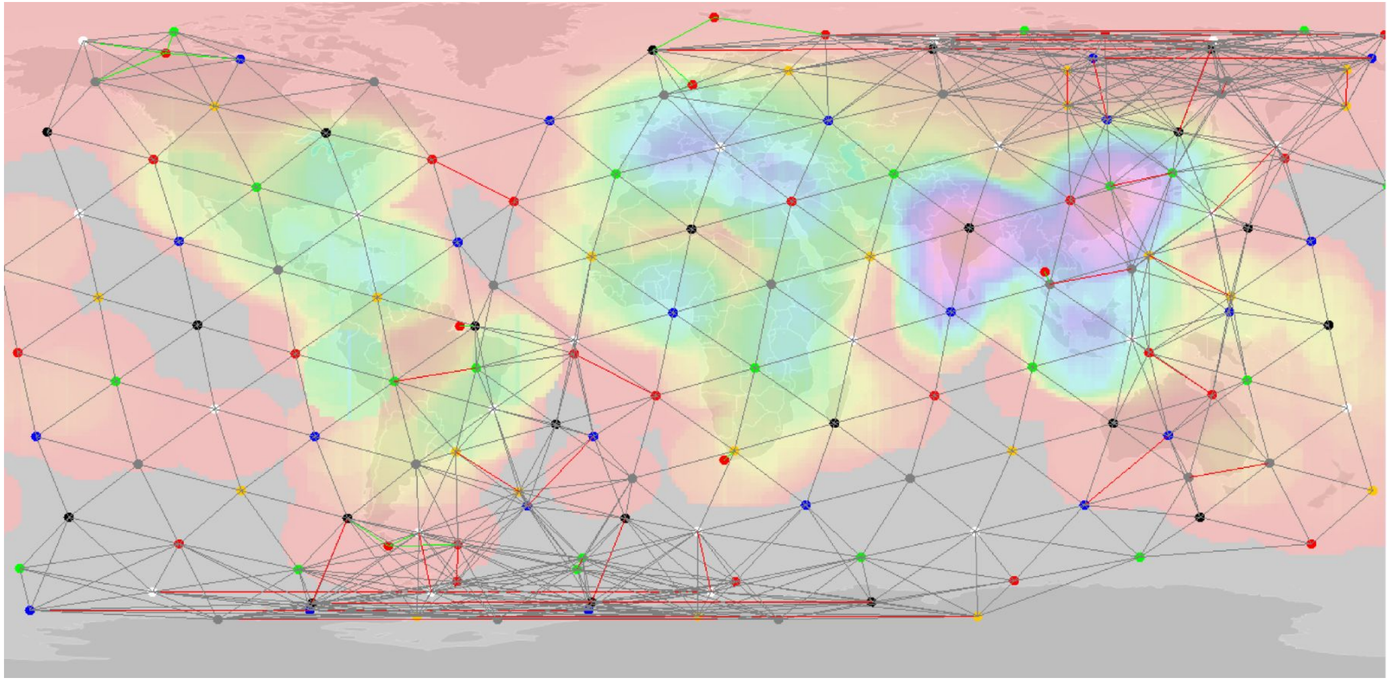
Introduction

- 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



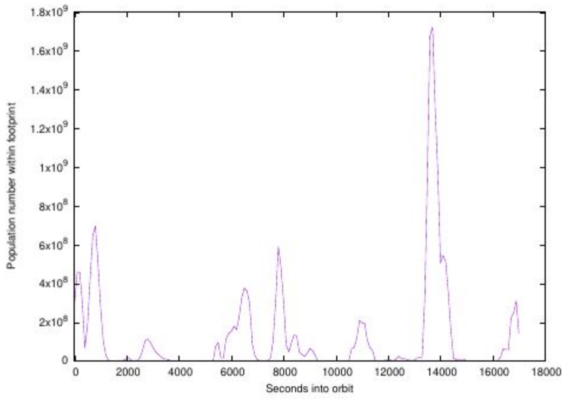
Why is a distinct middleware/OS needed?

- A SIN is distributed and mobile in its very core
  - basic MW/OS services must be “Mobility-aware”
  - even server layers must conduct handovers
  - resource discovery, invocation and migration is a formidable problem
- Mobility and resource management affects many interfaces
  - container <-> component
  - client <-> container
  - container <-> resource management
- A set of software services should provide life-cycle management for components and containers (e.g. Docker)

Which are the distinguishing design factors?

- N-layer structure
  - Service providers need to be replicated
  - Loop-free graph (DAG)
  - Frequently rebuild of the invocation tree
- Handover operations
  - Surface nodes (client and service providers) are stationary
  - First tier of service providers must be visible to ground client (frequent handover)
  - Links between satellites may require handover if path becomes too long.
- Stateful migration
  - Make “session object” accessible for appointed node after migration

Which are the distinguishing... ?



- Link and load predictability
  - Link availability and link budget can be estimated
  - Offered load can be estimated based on population statistics
  - *Fewer discovery protocols needed*
- Fail-over arrangement
  - Fail detection and fail-over should be conducted in the *Management Plane*, to relieve the clients from uncertain fail detection
- Security and trust management
  - "traditional" PKI certificate management has too high comm. requirements, authentication and authorization control should be done in one round trip

SIN-OS components

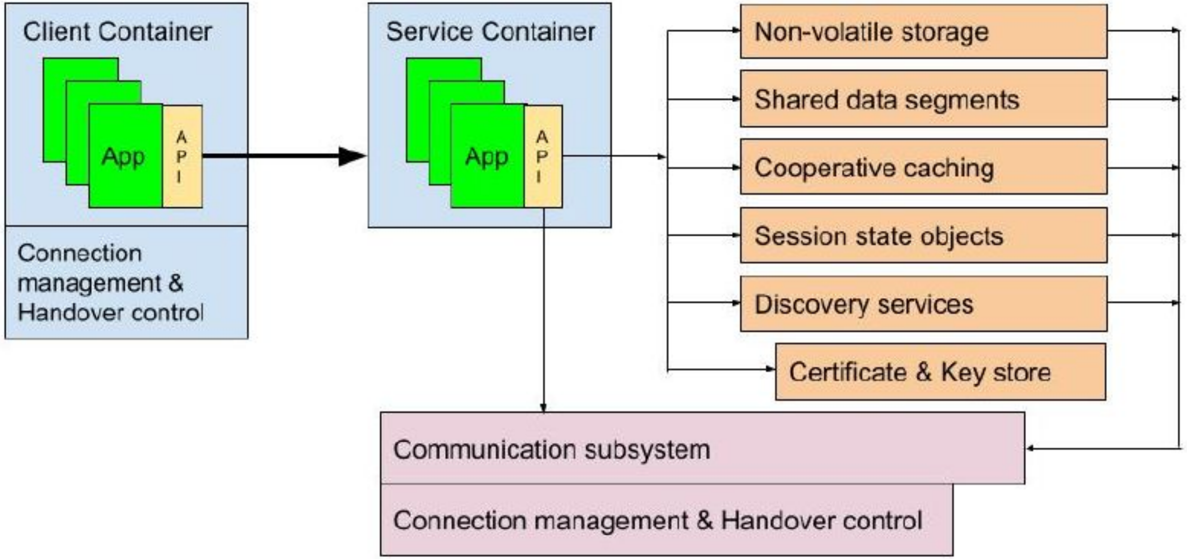


Figure 2. The components of a SIN-OS and their relations

Essential services in a SIN-OS

- Non-volatile storage
  - Files, OODBMS, RDMS, tuplespace. Distributed
- Shared data segments
  - Provides transaction protection, update ordering semantics, update notification ….
  - Clients must migrate in a synchronous manner
- Cooperative caching
  - Sharing immutable objects, coming from, e.g., lookup/discovery services
- Session state objects
  - Keeping session state variables accessible across handover operations
- Discovery services
  - Satellite positions can be predicted, but not the location of services
- Certificate & key store
  - Certificates likely to be different from X.509, with simpler validation methods

see: thinkmind.org

API collections

- Client API
  - Invokes services in satellite host SIN-OS (not service container)
  - Methods used for app *management*, others for *invocation*
  - *uploadApp, StartApp, ConnectApp, invokeService, requestHandover*
- Container API
  - Offered by the SIN-OS to the container
  - Resource allocation, life-cycle management
  - *loadApp*, *startApp, suspendApp, destoyApp*, *executeHandover*
- Component API
  - Access to SIN-OS services for storage, communication, synchronization etc.
  - Callback methods for life-cycle management

Conclusion

The problem: **How could the characteristic problems in a SIN be solved by a well organized middleware/operating system?**

- A SIN exhibits distinct problem due to the orbital cycle and predictable offered load (from surface clients)
- A SIN should provide a runtime environment for application components with proper separation and resource management, as well as the usual set of services for the execution (storage, communication,synchronization etc.)
- The application components are likely to be executed in a container environment, with a well defined API offered by the SIN-OS
- A suggested set of services and API have been presented in the paper.

Thank you for your attention, any questions?