SWIM - Flight data distribution using Web Services and publish/subscribe technologies

Antonio Strano

astrano@sesm.it

Dario Di Crescenzo

ddicrescenzo@sesm.it

OMG's Workshop on Real-Time Embedded and Enterprise-Scale Time-Critical Systems - May 24 - 26, 2010 - Arlington, VA USA

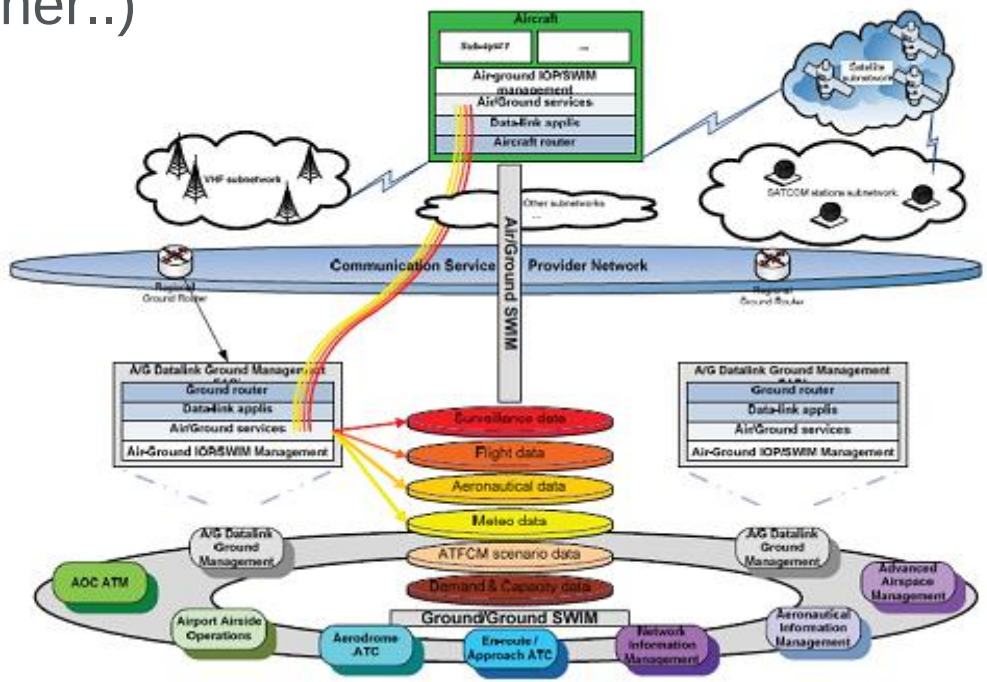
Outline

- SWIM concept
- SWIM-SUIT project
  - Objectives
  - Design Principles / requirements
  - SWIM-SUIT Prototype architecture overview
  - Testing & Validation
- Test Scenarios
- Some test results

SWIM concept (simplified)

- SWIM *(System Wide Information Management)* aims to establish a seamless interoperability among heterogeneous ATM stakeholders
  - common data representation
  - coherent view on current ATM information (e.g. Flight Data, Aeronautical Data, Weather..)

- SWIM may be seen as a common data/service bus on which systems having to interoperate are “connected”



SWIM-SUIT project objectives

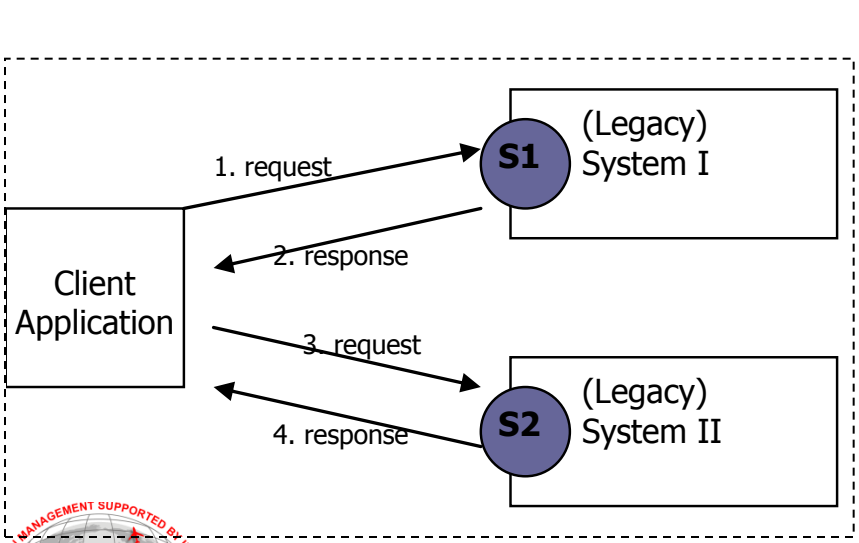
- Analyse the (SESAR) SWIM concept defining (high level) requirements
- Identify the right technologies
- Design and Implement the first SWIM prototype
- Validate the SWIM prototype capabilities in order to demonstrate the feasibility of the SWIM concept
- Identify the lessons learnt to support the SESAR activities

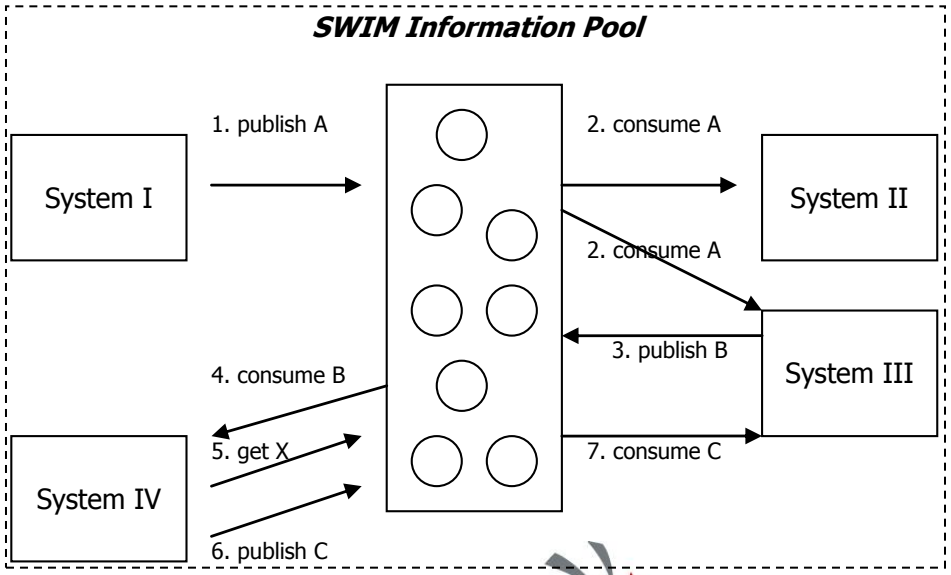
Design Principles / Requirements (1/2)

- Following SESAR definitions the prototype :
  - has been designed using a domain based approach (Flight, Surveillance, etc. – like a data model based systems integration)
  - has been implemented using a standard based approach
    - well known data and information models (e.g. ICOG2 for the flight data domain)
    - standard technologies (Web Services, EJB, DDS)
  - decouples external adapters/systems from internal knowledge about the SWIM implementation

Design Principles / Requirements (2/2)

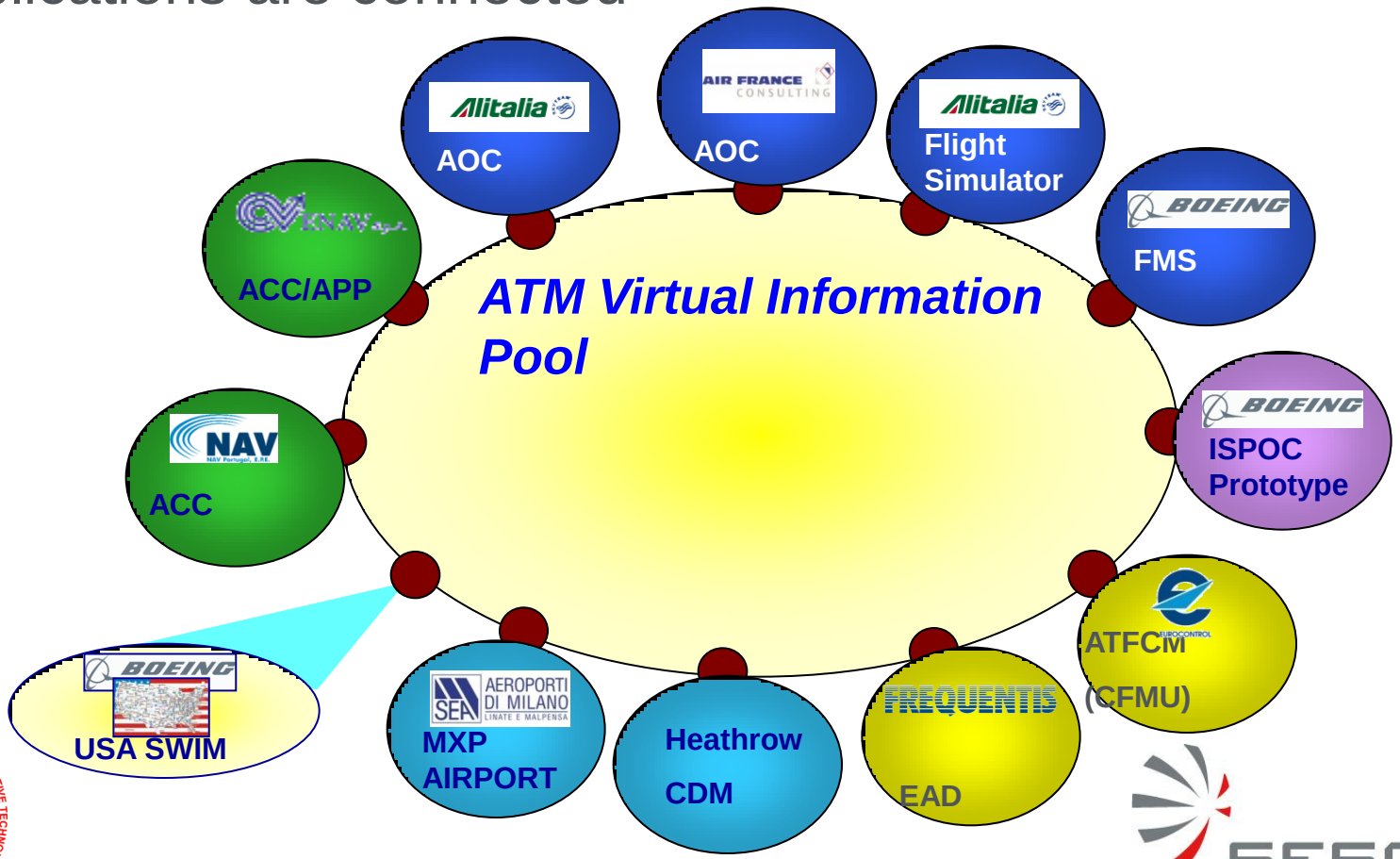
- Request/Reply
  - Supported by Web Services (basically a synchronous access)
- Publish/Subscribe
  - Supported by JMS/DDS (asynchronous access)





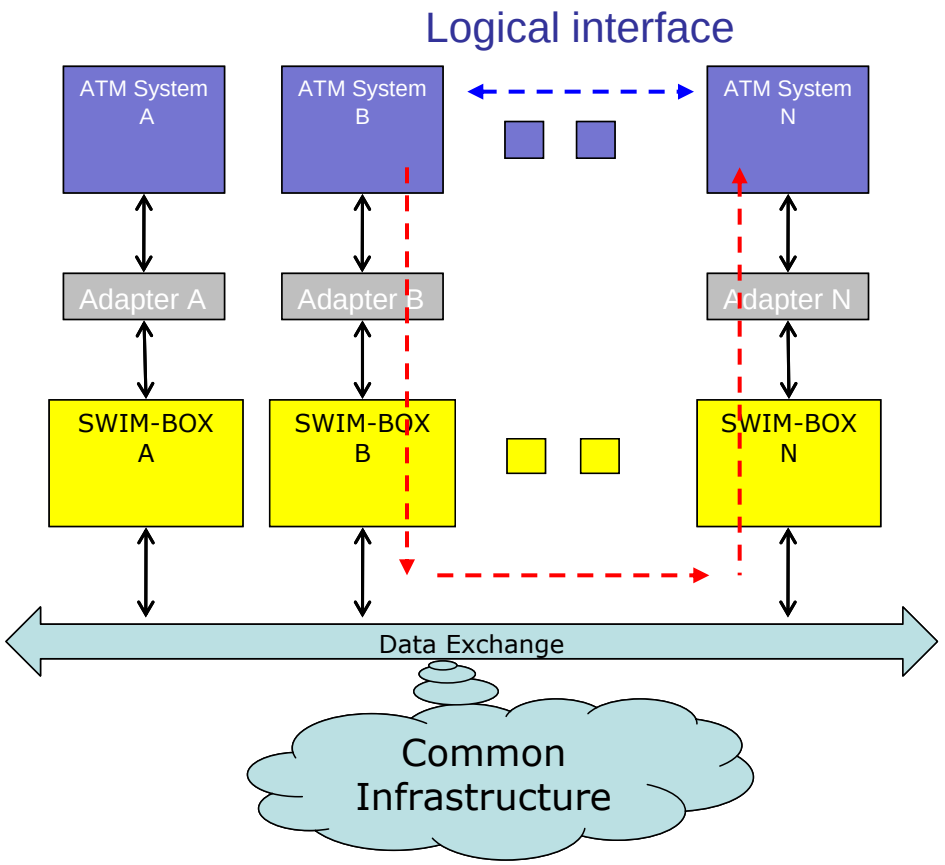
SWIM-SUIT prototype

- The prototype (named “SWIM-BOX”) has been conceived as a sort of common data/service bus on which legacy applications are connected

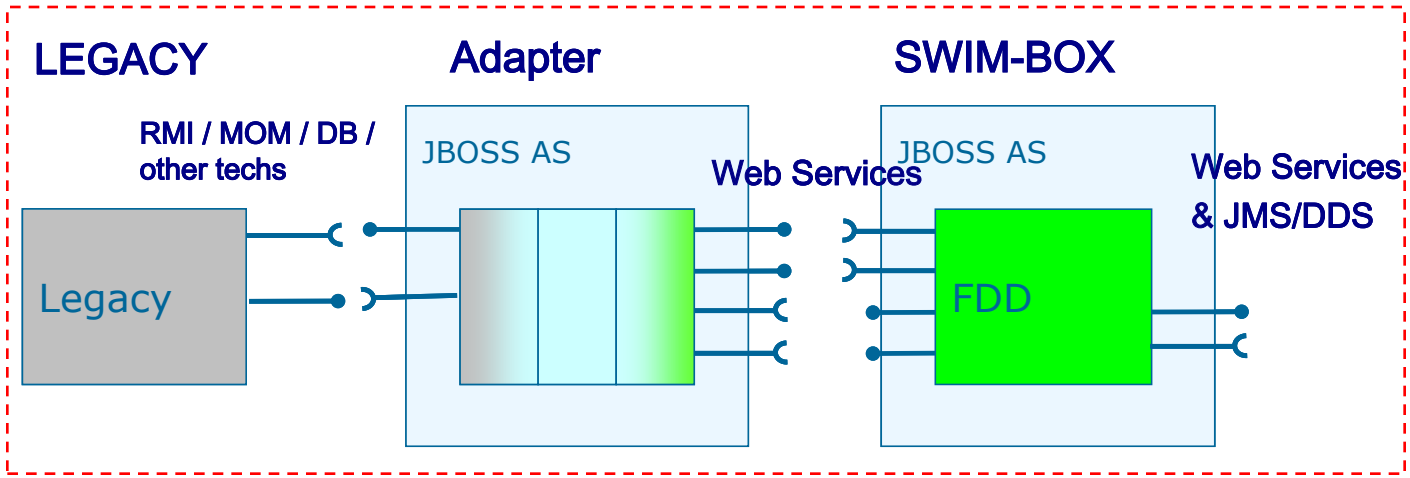


SWIM-SUIT Prototype architecture 1/3

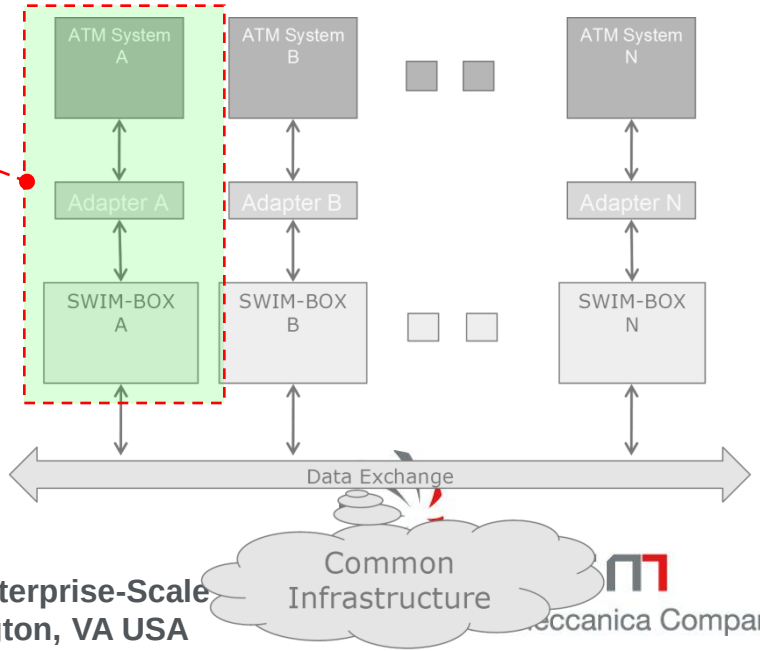
- The Adapter translates data/services among (pre SWIM-SUIT) Legacy and SWIM-BOX
- The SWIM BOX acts as a mediator between different legacy systems enabling the end-to-end communication (req/reply and pub/sub)
- Each Adapter might be further decomposed in different adapters dedicated to the served “Data Domains” (Flight, Surveillance, etc)



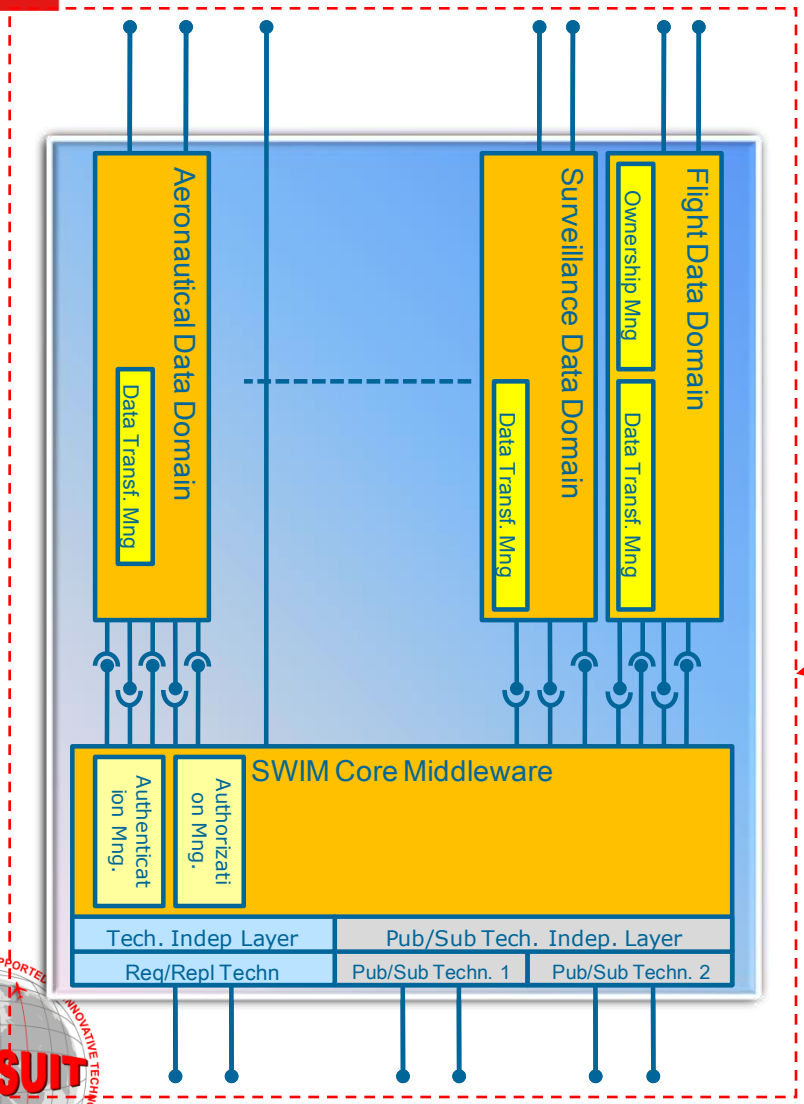
SWIM-SUIT Prototype architecture 2/3

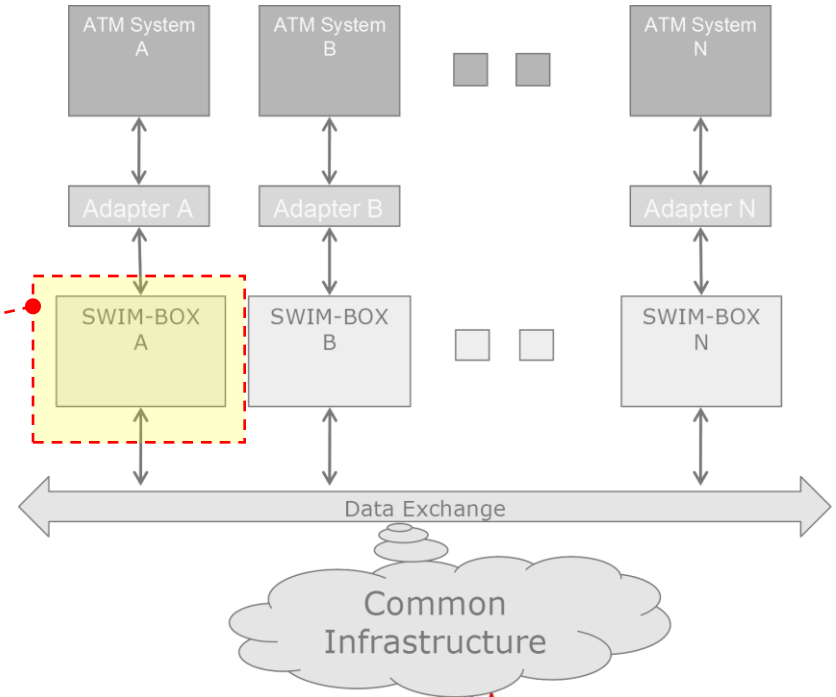


- Currently just the WS technology is supported at adapter/swim-box interface but J2EE enables to expose this interface also using different technologies (e.g. CORBA, MoM, etc)



SWIM-SUIT Prototype architecture 3/3





SWIM Data Domains

- Offers services for the specific domain (e.g. create, update and publish flight object) and offer some extra management and utility (e.g. filtering) services
- Defines a “standard” data representation and translate it in a flexible format (XML in the prototype)
- On a domain basis, manages the roles taking into account a subset of information
- Provides facilities for consuming services exposed by the adapters\legacies through the SWIM-BOX
- For the prototype three Data Domains have been implemented:
  - *FDD (Flight Data Domain)* : ICOG2 data & information standard (providing specific extensions)
  - *SDD (Surveillance Data Domain)* : ASTERIX Category 62 data standard (binary & XML representations)
  - *AID (Aeronautical Information Service Domain)* : Aeronautical Information Exchange Model (AIXM)

SWIM-BOX Core

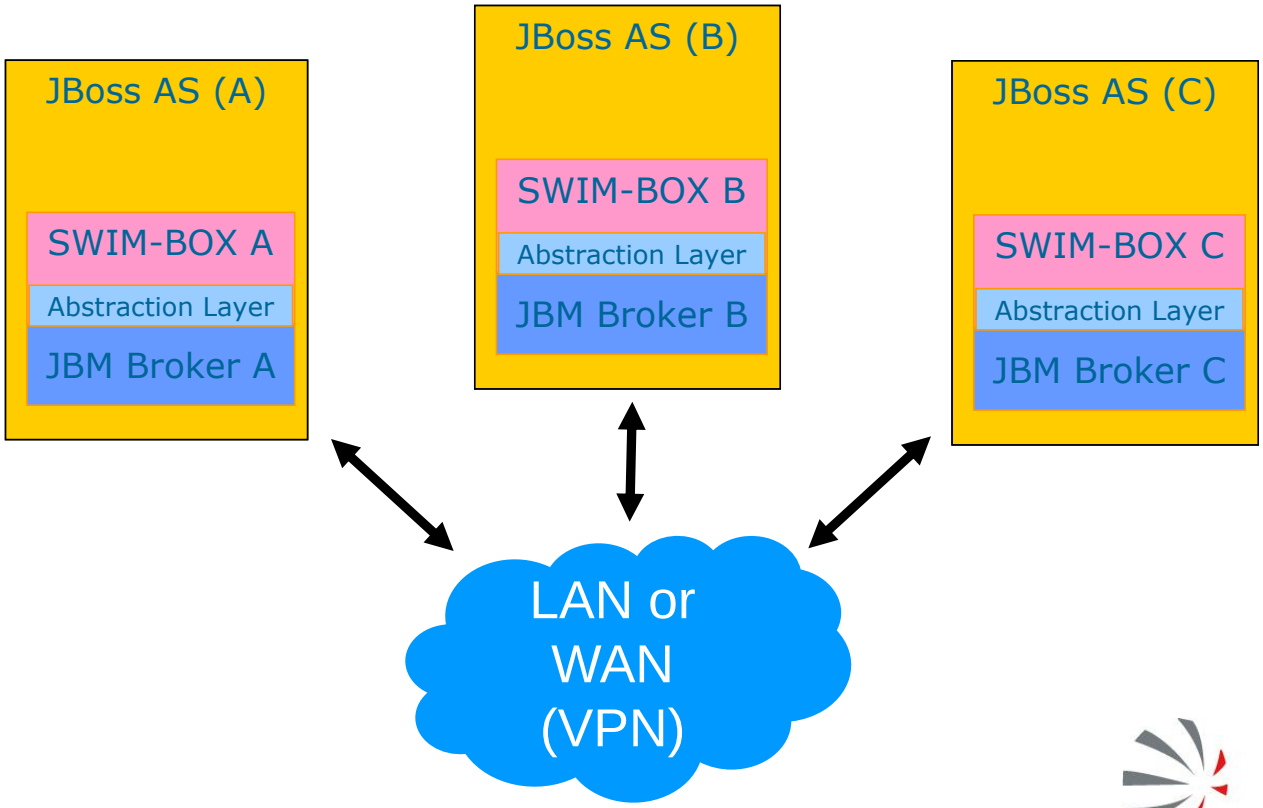
- It is loosely impacted by changes in the data representation and by changes in the services exposed in the SWIM Data Domains (it acts as much as possible as a transport layer)
- Two main components :
  - *SWIM-SUIT DataStore* : provides services for sharing data across the SWIM-BOX network (distributed, persistent and transactional cache)
  - *SWIM-SUIT PublishSubscribe* : provides services required for the pub/sub pattern (subscribe, publish, content based filtering, etc)

SWIM-BOX Core - PubSub

- SWIM middleware should be able to minimize impact of technology change
- In order to shield domains components from such technology change, a thin PubSub technology abstraction layer has been introduced
- The technology dependent layer has been implemented using JMS and DDS (multiple products)
  - JMS and DDS implementations provide different QoS support
  - RBAC (Role Based Access Control) & XML encryption
  - Content based filtering
  - Asynchronous (data listeners) and Synchronous (read/take) subscription styles
- Interoperability between different DDS implementations – on going activity

JMS clustered configuration

- JMS: *JBoss Messaging* in a clustered configuration has been set up
  - Uses *JGroups* protocol on multicast to synchronize the JBM brokers
  - Uses TCP connections among JBM brokers to exchange messages
  - At “application” level, the message broker is always “local”



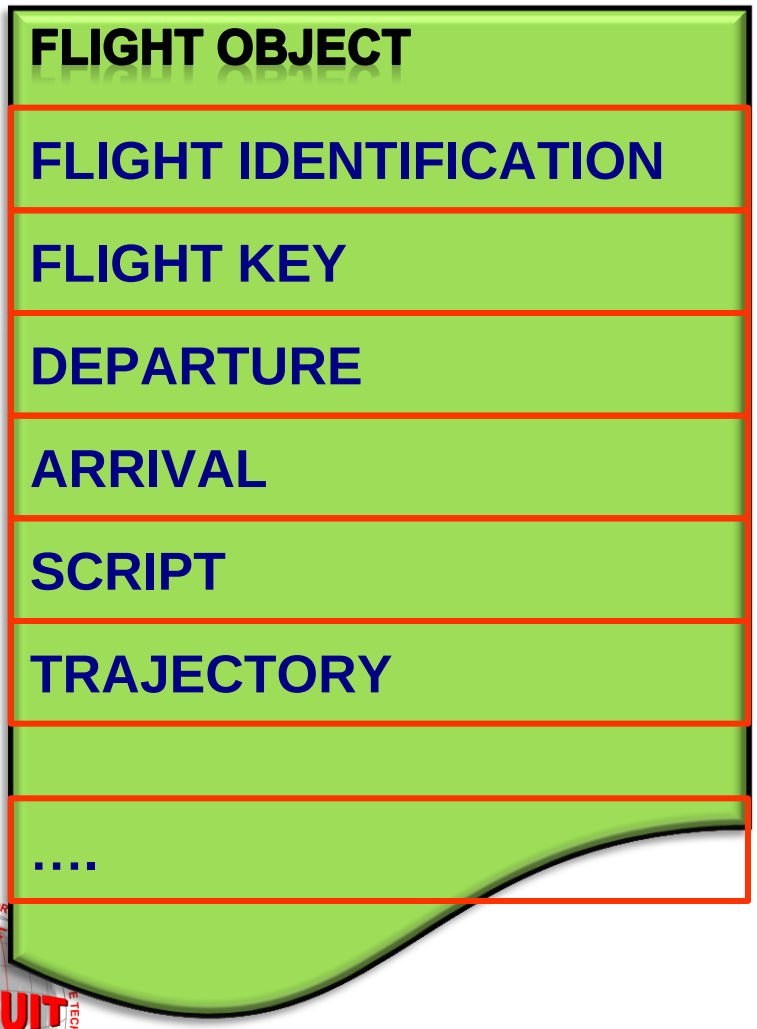
Testing & Validation activities

- Pseudo Operative Tests (Legacy Systems involved)
  - Suite of operative scenarios (e.g. management of the airport arrival sequence) to validate the prototype capabilities and to demonstrate the feasibility of the SWIM concept
  - WAN
- Performance Tests (no Legacy Systems involved) – on going
  - Automatic test suite to evaluate the prototype behavior (e.g. average response time, data loss, throughput, etc.) building specific workloads (e.g. service invocation rate, data publication rate, number of published data, number of flight data, etc.)
  - LAN & WAN

Test cases

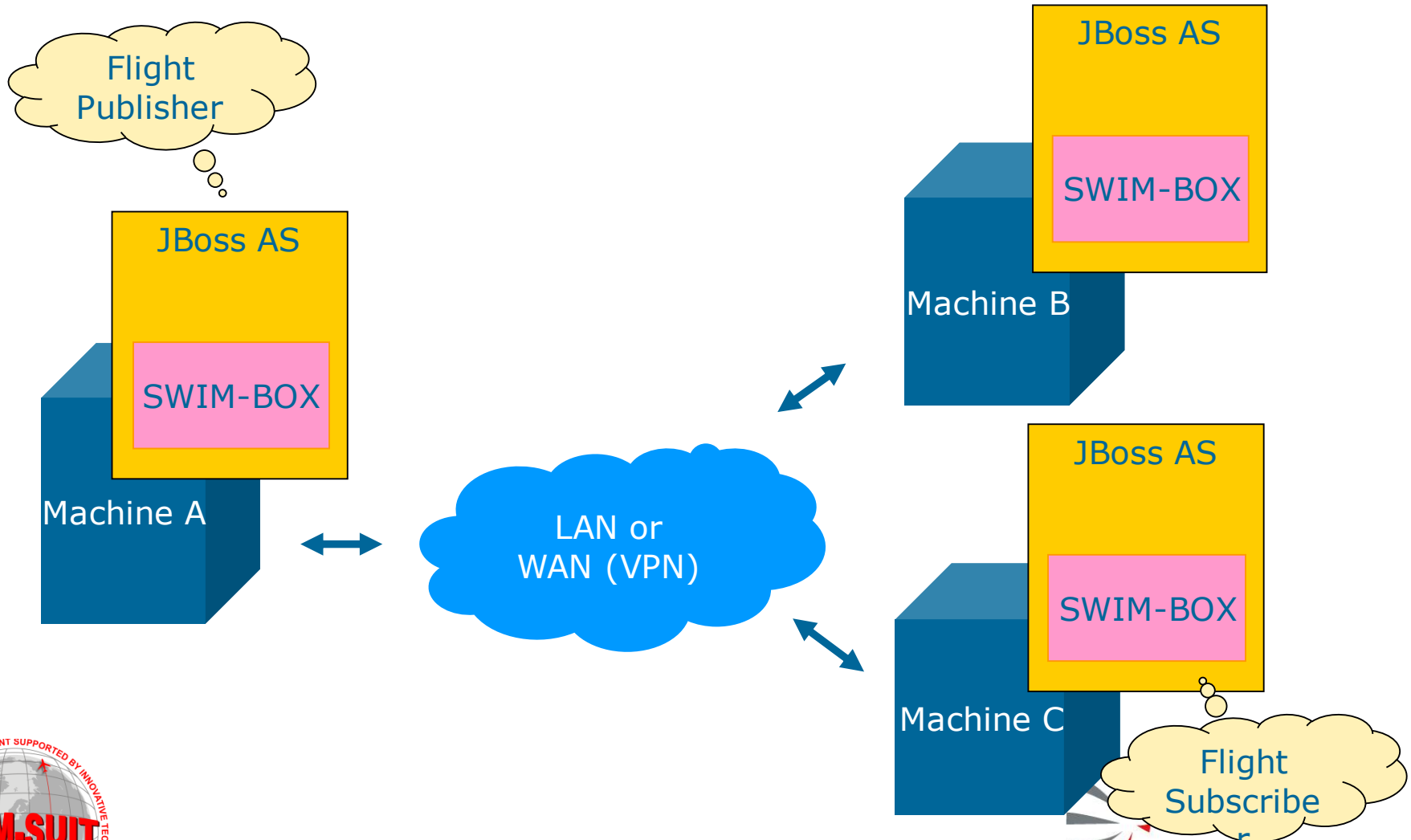
- Tests are organized on a functional basis
  - Test and evaluate performances when creating a number N of Flight Objects (i.e. flight clusters) every X ms with different technologies (DDS, JMS) – each creation implies an amount of data sent on the wire of circa 60KB
  - Test and evaluate performances when requesting N times an update of a Flight Object every X ms with different technologies (DDS, JMS) – each update request triggers a new distribution of a number of flight clusters - each update implies an amount of data sent on the wire of circa 30KB

The ICOG2 Flight Object



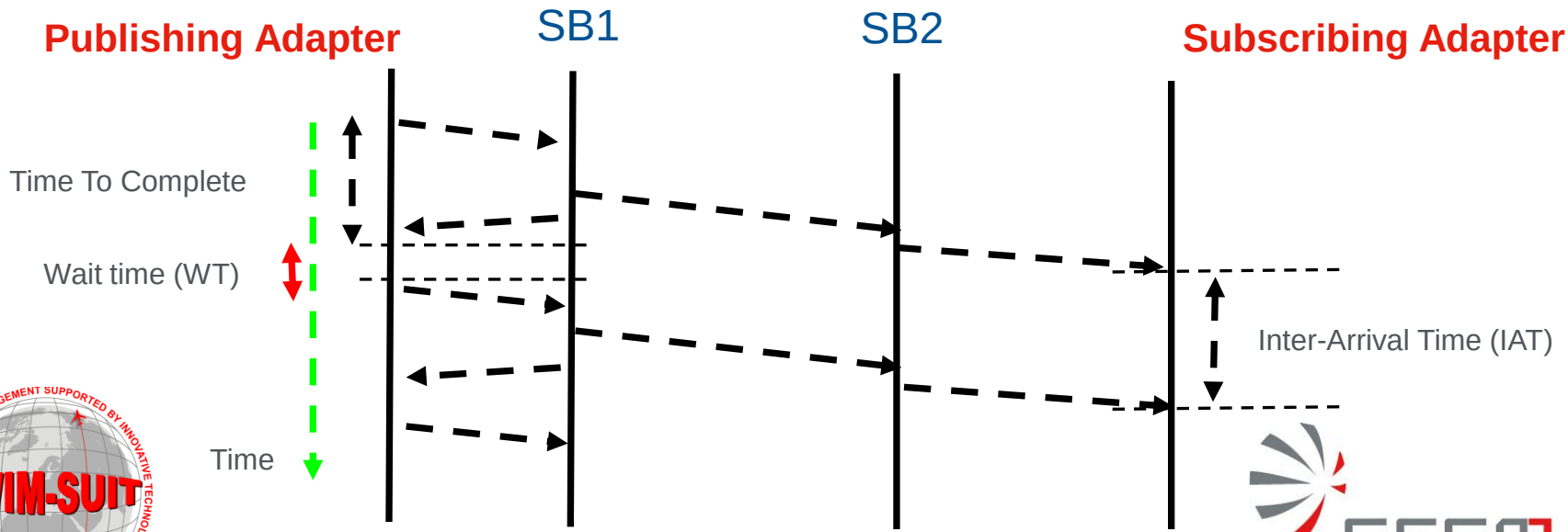
- It can be seen as a single entity comprising different information (flight object clusters) related to a flight
- Information are “clusterised” in self-consistent XML parts (Departure, Arrival, Trajectory, etc)
- When having to share/update information to the interested parties, just clusters are distributed

Tests deployment view

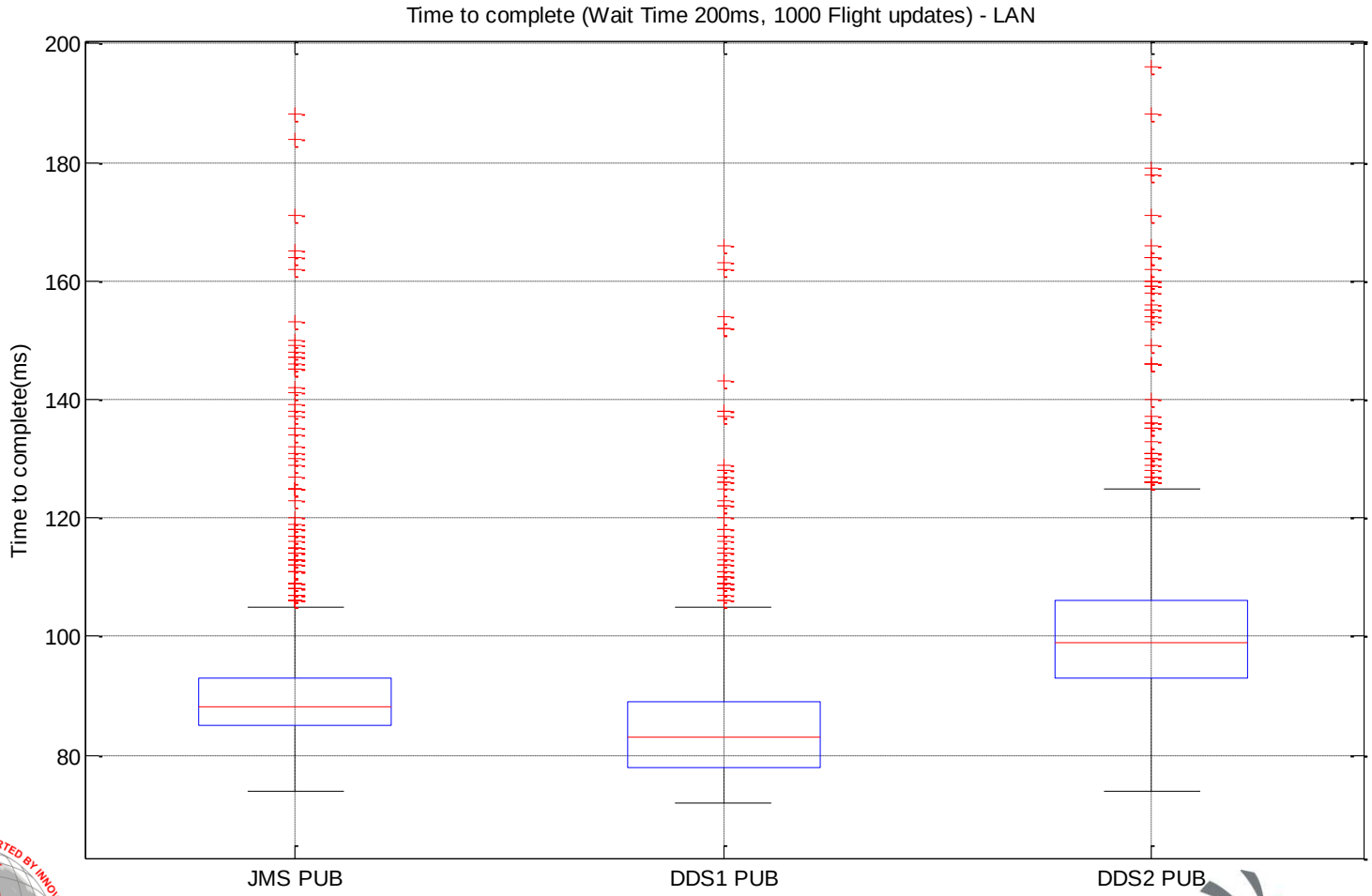


What we measure

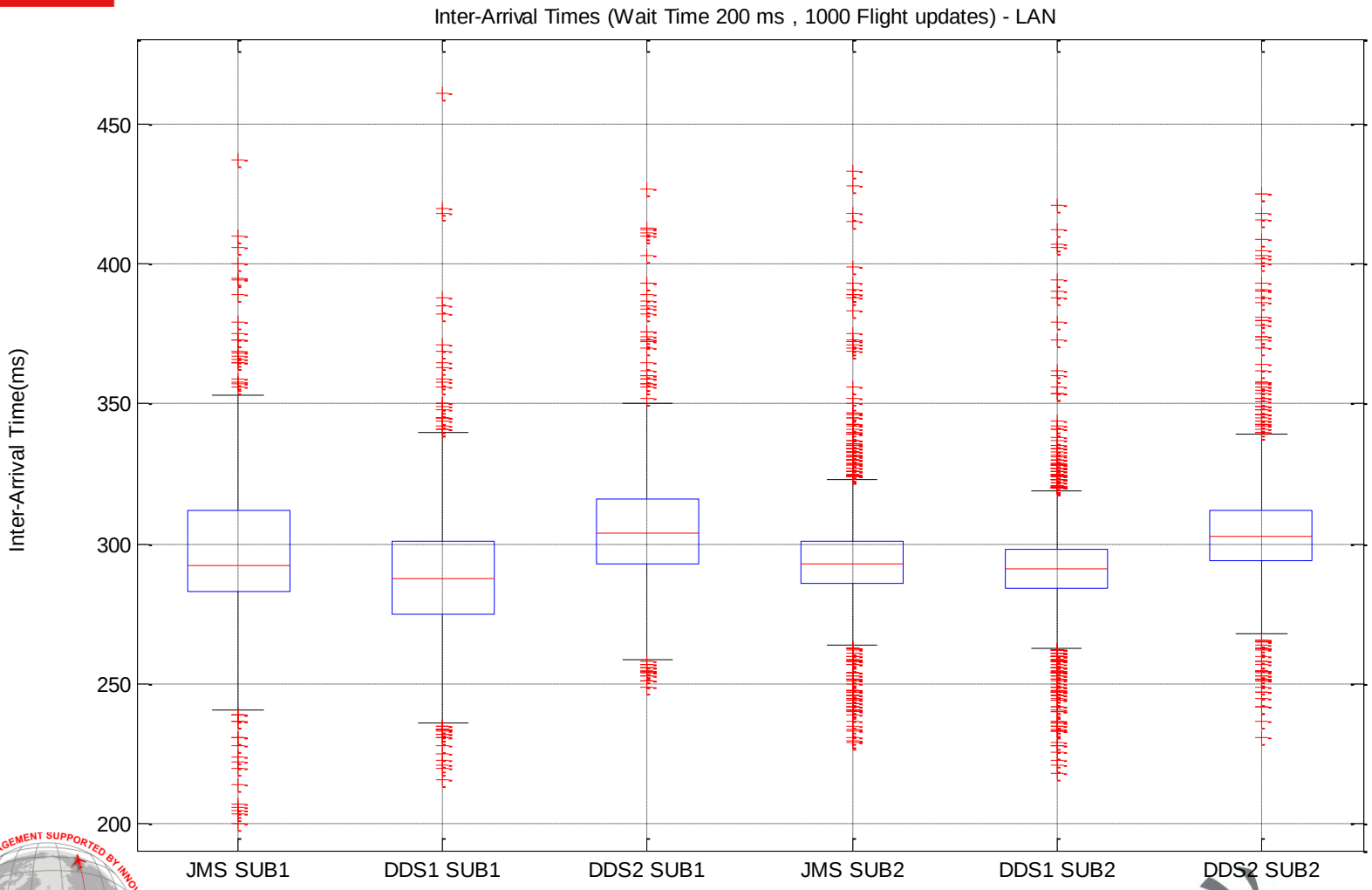
- Measures are performed at client side (“Adapter” level in our architecture):
  - On the publisher, the time to complete each create or update operation is measured
  - On the subscribers, the time among two subsequent notifications (Inter-Arrival Time) is measured
- On-going work activities to develop an infrastructure to measure (without modify the current prototype implementation) the time spread across different architecture layers (Adapter, Data Domain, Core Service, etc)



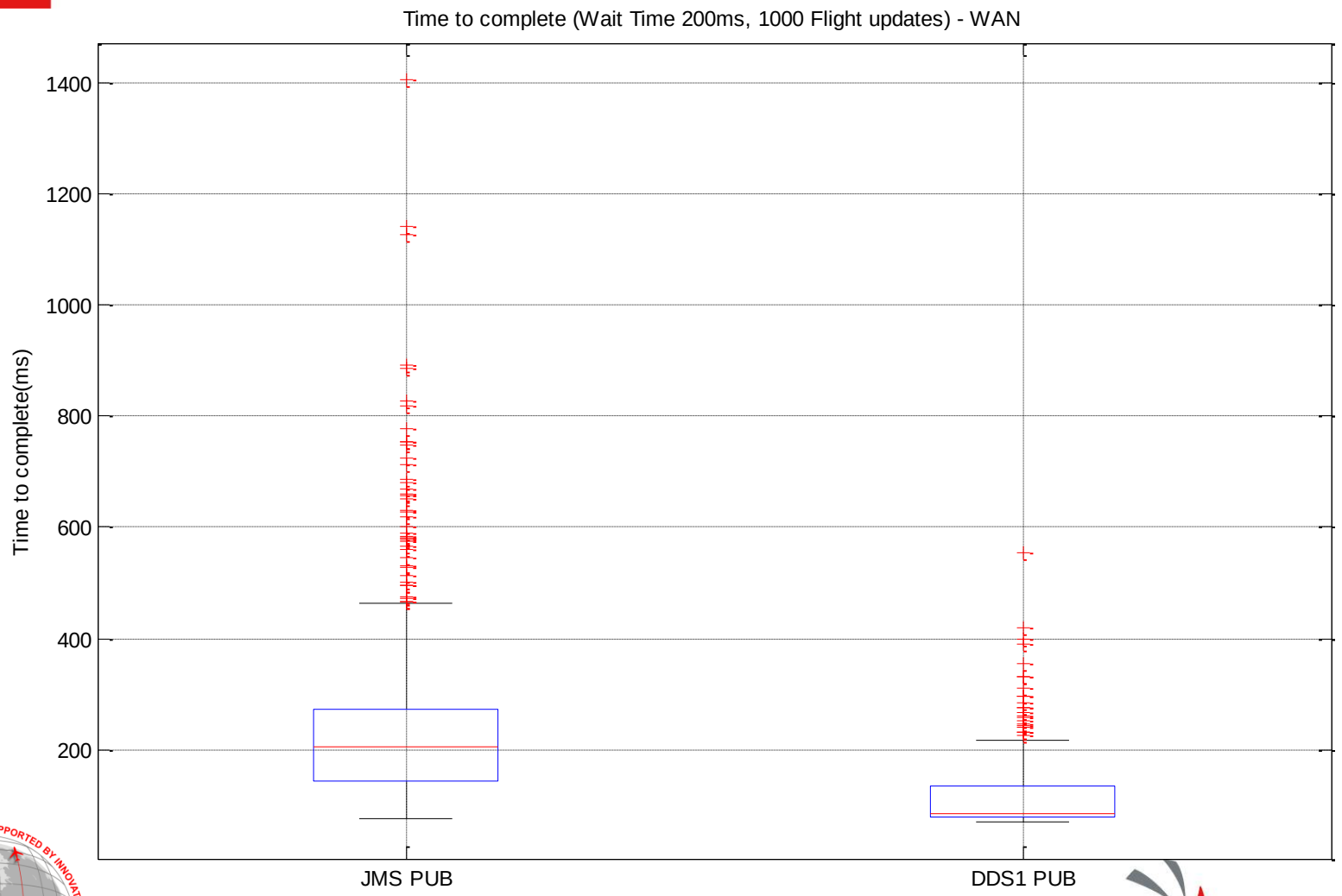
Test case 2 – FO update on LAN (Publisher)



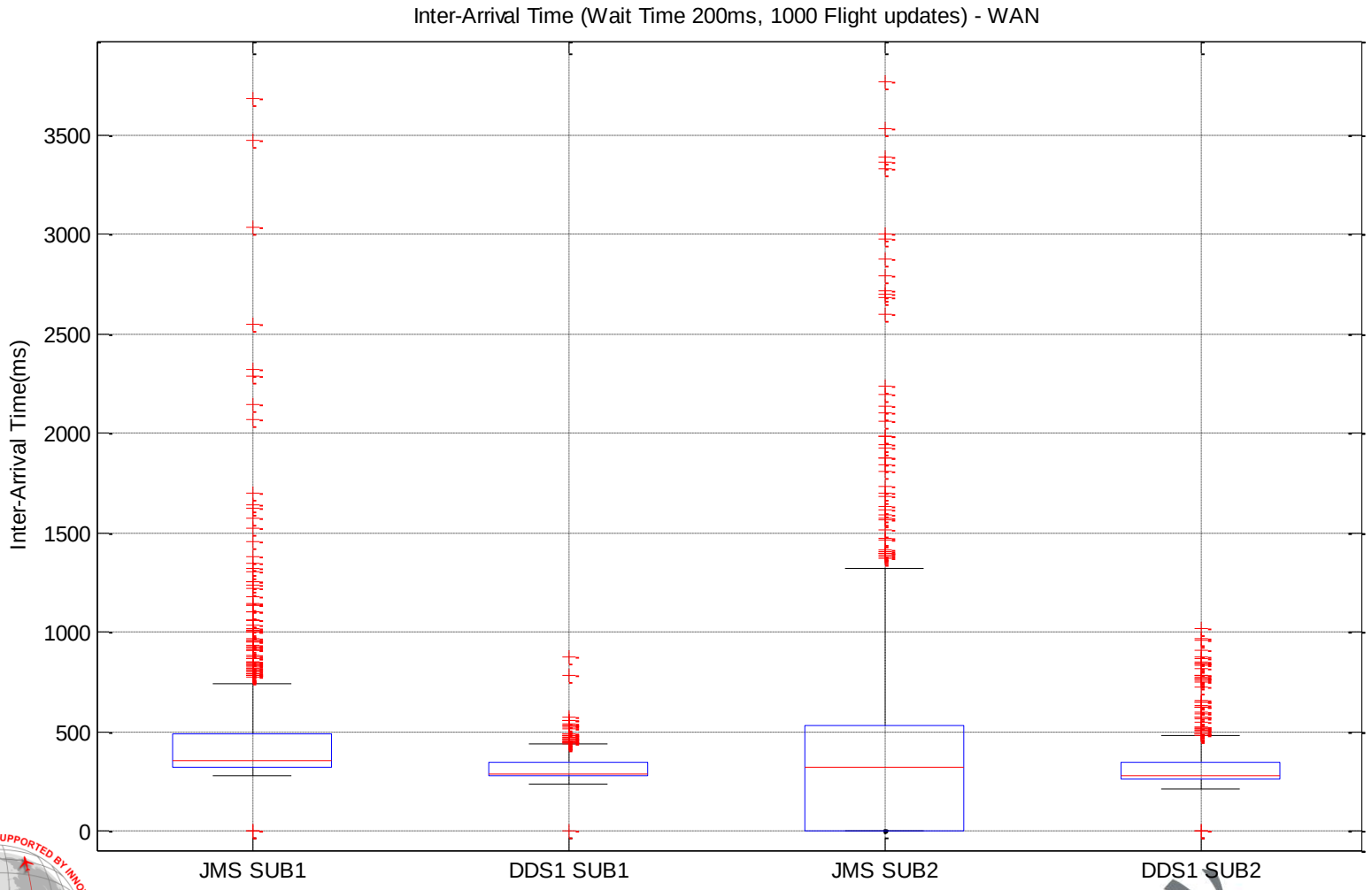
Test case 2 – FO update on LAN (Subscribers)



Test case 2 – FO update on WAN (Publisher)



Test case 2 – FO update on WAN (Subscribers)



Some practical experiences (1/2)

- What actually came out from our hands on experience when using both technologies?
  - DDS provides a wide set of capabilities (wrt JMS) which better support the development activities (e.g. content based filtering, partitioning) and the system integrators activities (e.g. QoS tuning)
  - Different DDS implementations are interoperable
  - but…when moved in WAN environment much more effort have been spent on DDS to determine a “good” WAN configuration

Some practical experiences (2/2)

- What could help developers/system integrators life (from our point of view)?
  - An open (at least semi-automatic) test suite allowing to estimate, in user defined context/scenarios, optimal configuration parameters (e.g. buffer sizes, latency buckets etc..)
  - Asking for more, a strategy/tool for on-line determining such parameters (where applicable) thus supporting system automatic re-configuration according to context changes (e.g. bandwidth, network load, etc..)

Questions?

FINMECCANICA

http://www.swim-suit.aero/swimsuit/

SESM
A Finmeccanica Company