



RFID Middleware: Ensuring Qualities of Service for the Internet of Objects

Olivier Liechti

Sun Microsystems, Inc.

Dominique Guinard

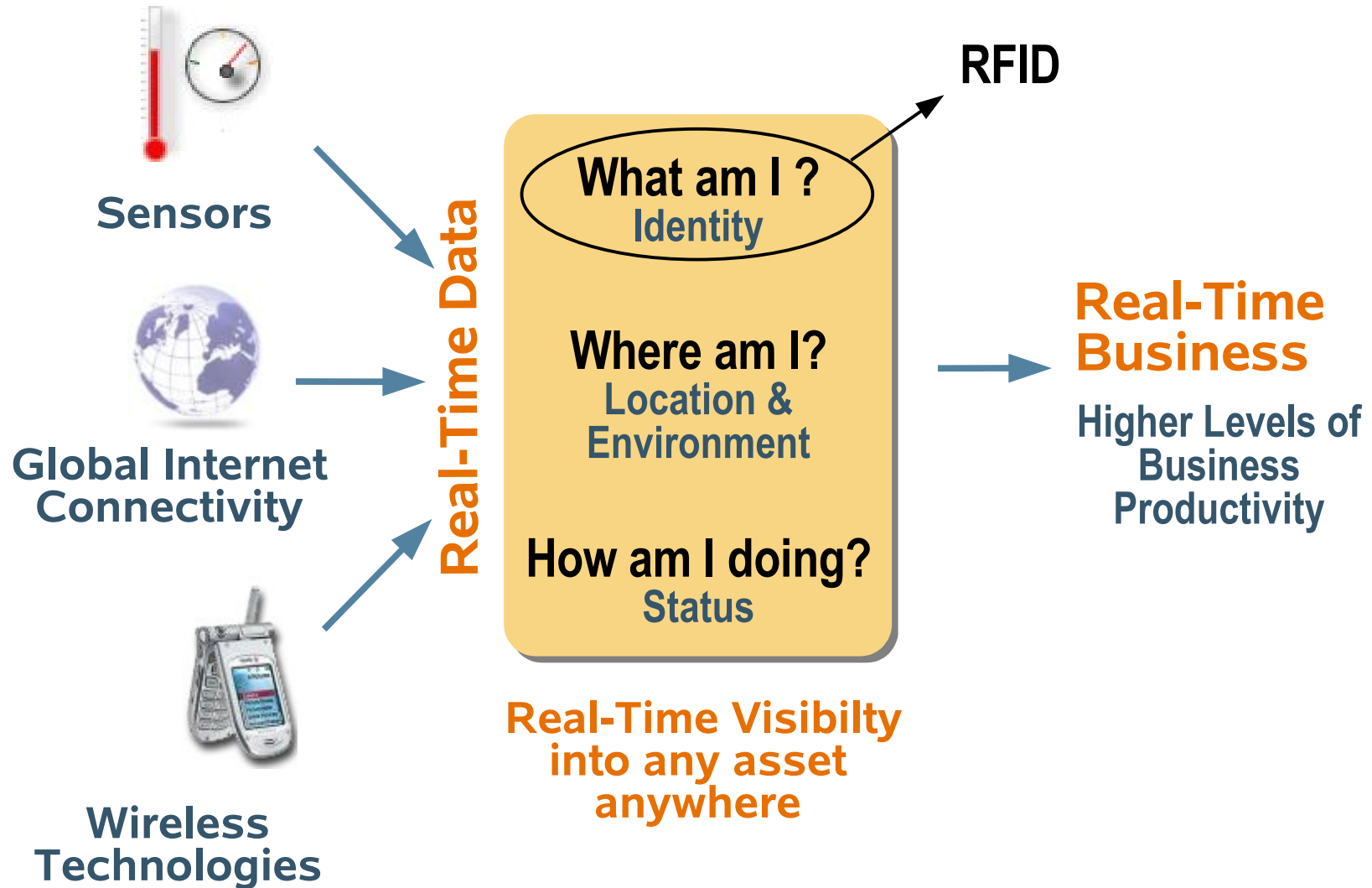
University of Fribourg



Agenda

- Vision
- RFID Deployment Challenges
- RFID Middleware
 - > Requirements & features
 - > Architecture
 - > Current trends
- Sample application: RFIDLocator

Vision: the Aware Enterprise



Deploying RFID Poses Challenges...

- Physics
 - > What Tags, What Readers?
 - > Regulations, Standards etc.
- Infrastructure Issues
 - > Data Management
 - > Connectivity Issues
 - > Remote Management
 - > Security and Access Control
- Integration Issues
- Leveraging the Data (Business Intelligence)

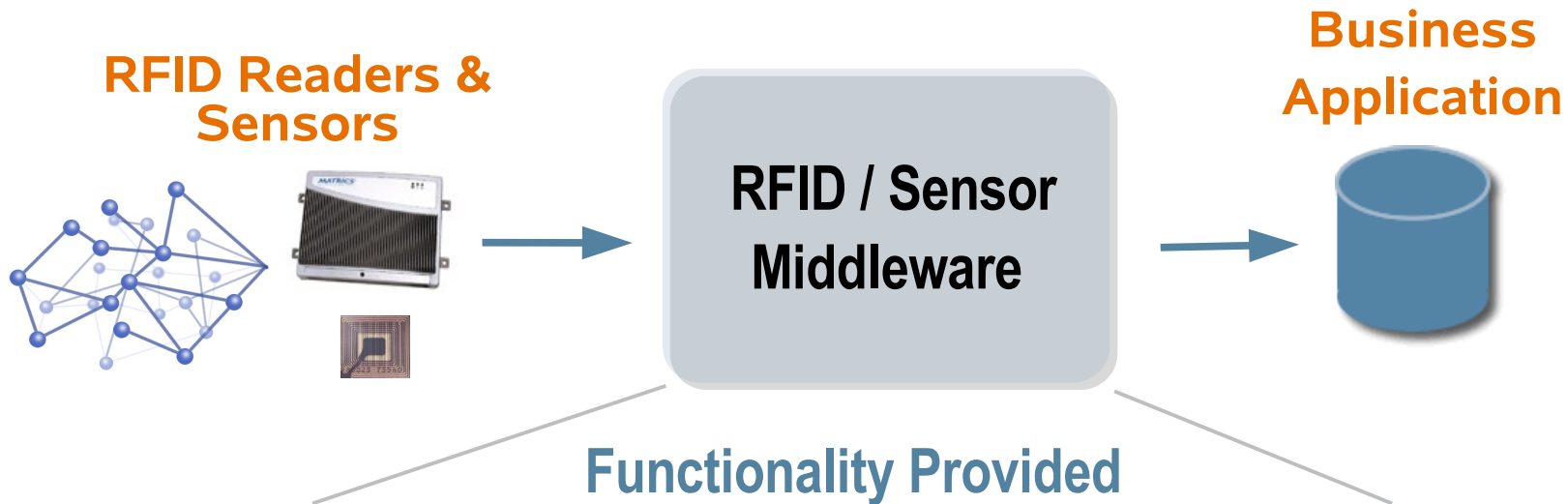
How much RFID Data will We See?



or

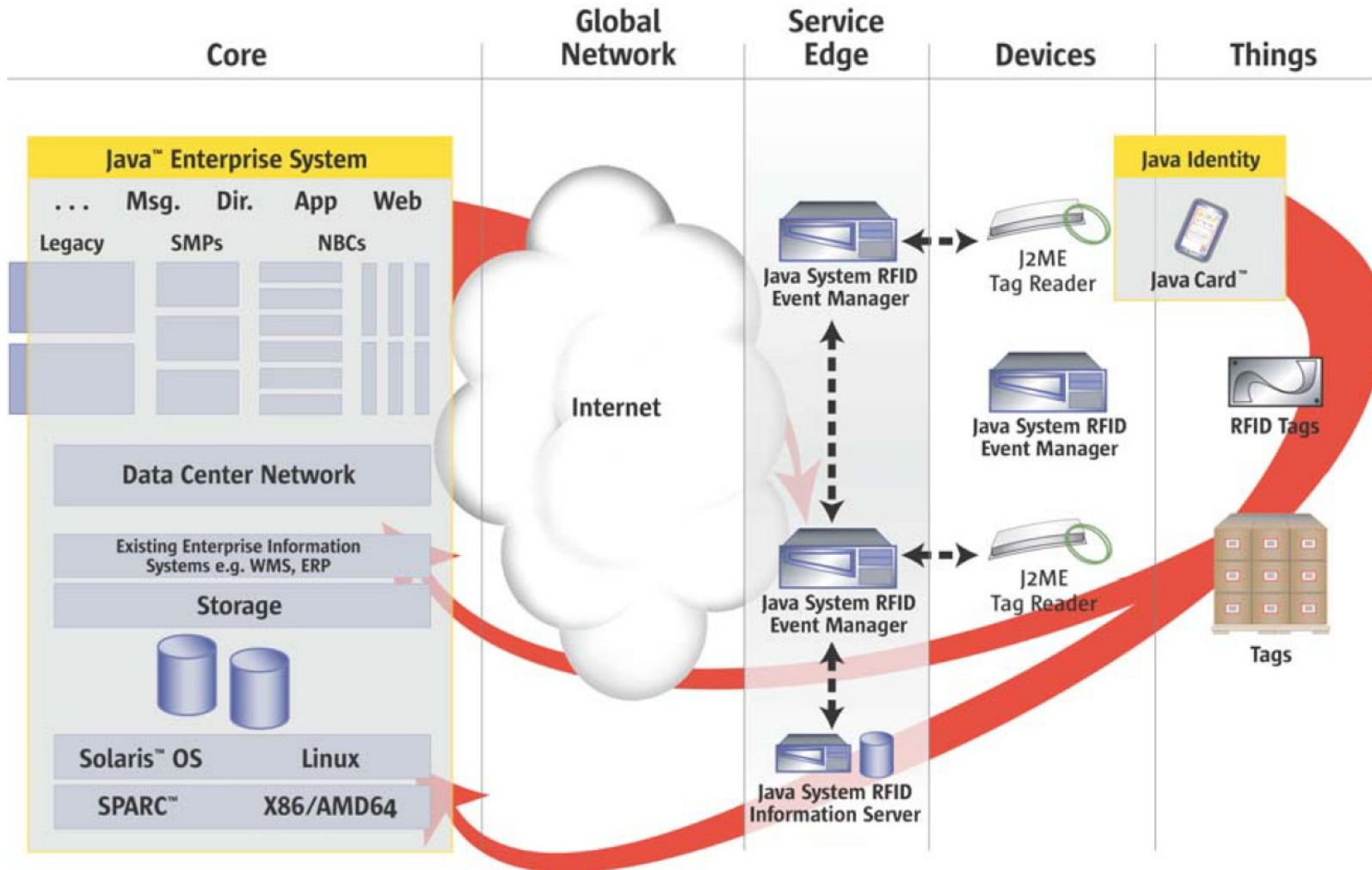


RFID Middleware Functionality



1. Data smoothing and validation
2. Integration of data across multiple sensors
3. Information lookup in internal or external databases,
4. Data buffering and aggregation over time
5. System Management performance and health

RFID Middleware Architecture



Sun Java System RFID Software

SJS RFID Event Manager



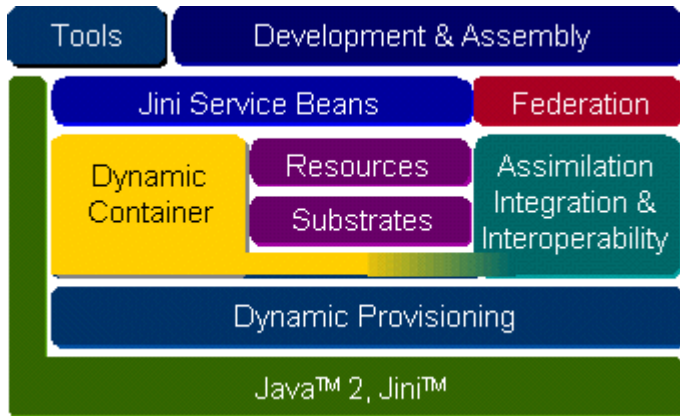
- Purpose: Event Data Collector
- Supports EPC Gen 2, ISO, other tags (Active/Passive) and sensors
- Self-healing, fault tolerant, automatic
- Central Monitoring console, Remotely Administrable (JMX, SNMP)
- Java based (API for extension)

SJS RFID Information Server



- Purpose: Store and Retrieve Item Information and Track & Trace
- J2EE based Java Enterprise System
- RDBMS (Oracle)
- Extensible schema
- Client Interfaces: XML/Http, JMS, RMI (Java Client Library provided).

Sun's Sensor Middleware Technology

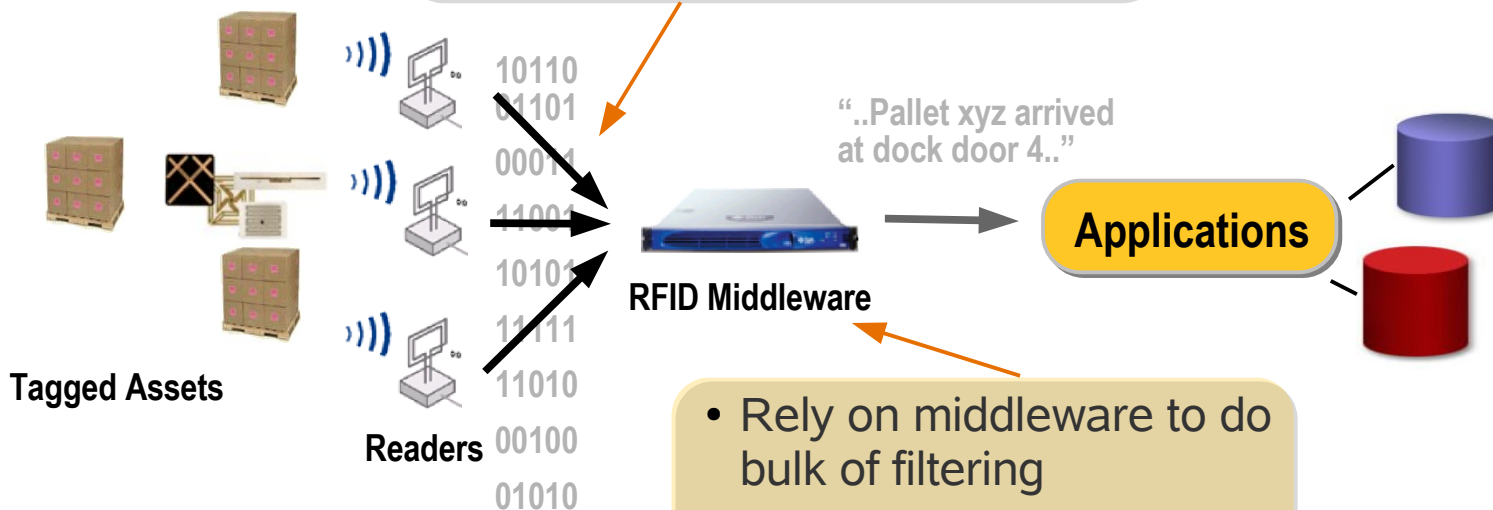


RIO Container

- RIO - a dynamic adaptive network architecture built using Jini technology mechanisms
- Key Constructs include
 - > Dynamic Provisioning
 - > Policy-based and Quality of Service (QoS) mechanisms
 - > Jini Service Beans
 - > Event Dispatch and Discovery
 - > Peer-to-Peer Event Model
 - > Federation

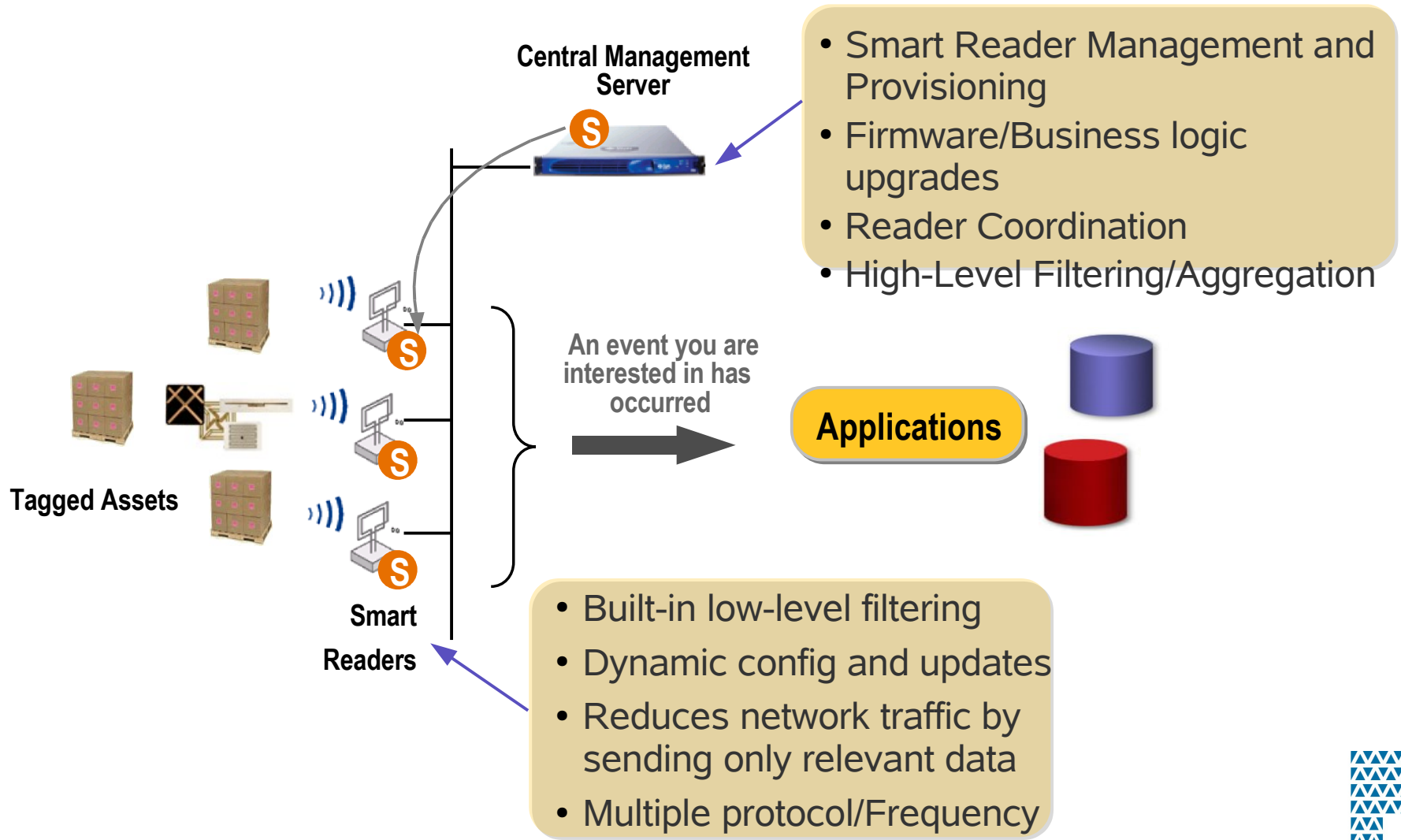
Basic Readers

- No intelligence besides reading and forwarding data
- Little or no filtering of data
- Floods the network with data
- Cheap, Single protocol/frequency

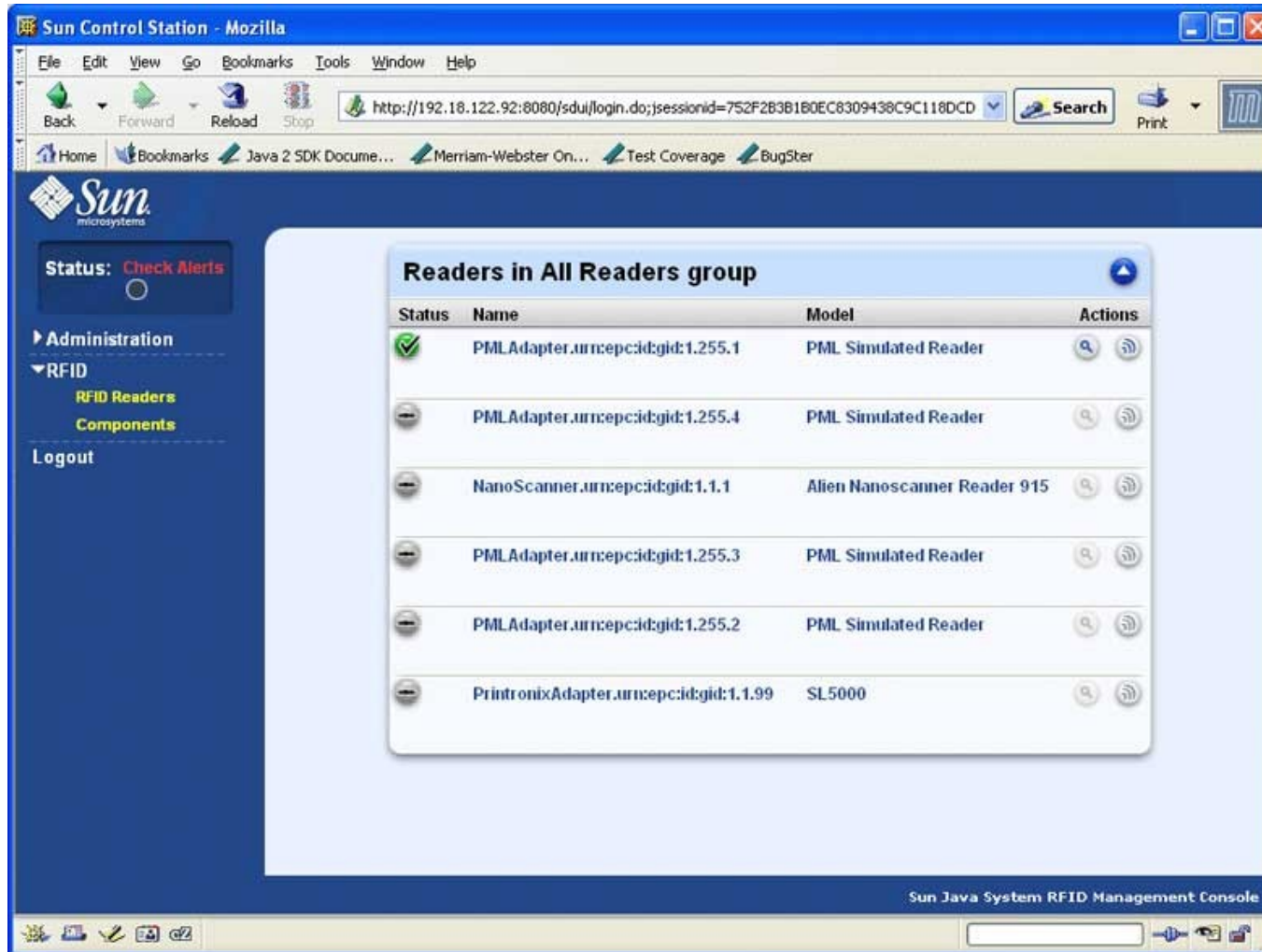


- Rely on middleware to do bulk of filtering
- RFID Middleware could become point of failure; need robust middleware

Smart Readers



Reader Management

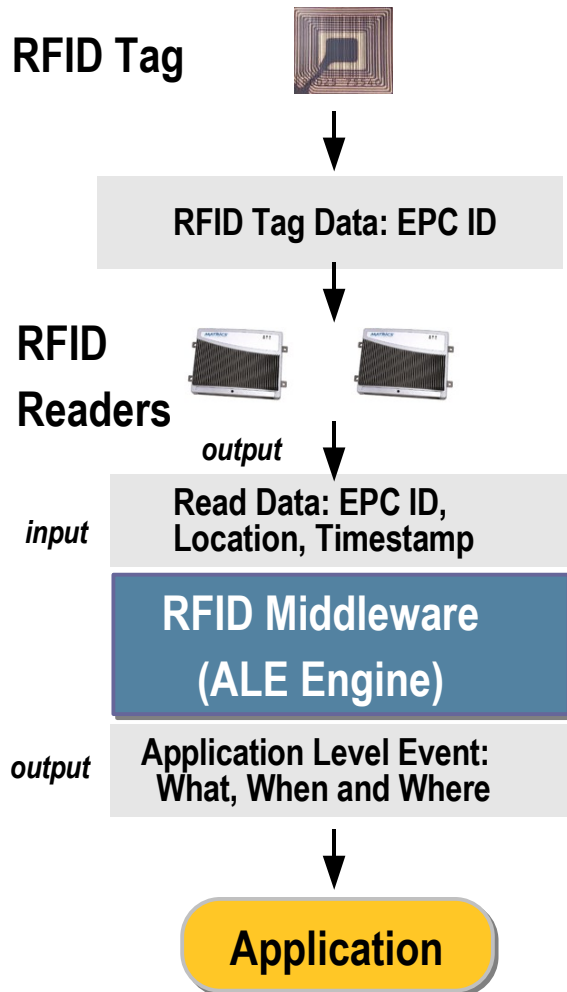


Readers in All Readers group

Status	Name	Model	Actions
✓	PMLAdapter.urn:epc:id:gid:1.255.1	PML Simulated Reader	🔍 🔄
⊖	PMLAdapter.urn:epc:id:gid:1.255.4	PML Simulated Reader	🔍 🔄
⊖	NanoScanner.urn:epc:id:gid:1.1.1	Alien Nanoscanner Reader 915	🔍 🔄
⊖	PMLAdapter.urn:epc:id:gid:1.255.3	PML Simulated Reader	🔍 🔄
⊖	PMLAdapter.urn:epc:id:gid:1.255.2	PML Simulated Reader	🔍 🔄
⊖	PrinttronixAdapter.urn:epc:id:gid:1.1.99	SL5000	🔍 🔄

Sun Java System RFID Management Console

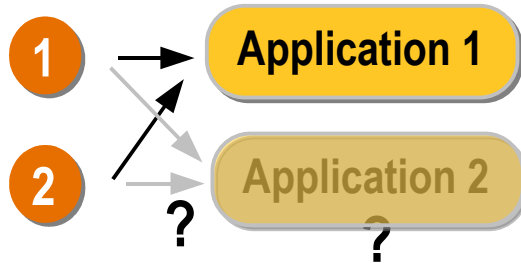
What is ALE?



- ALE stands for Application Level Events
- ALE is a SW specification for the filtering and collection of RFID data being defined and ratified by EPCglobal
- ALE enables the aggregation and translation of individual reader events into events meaningful to applications

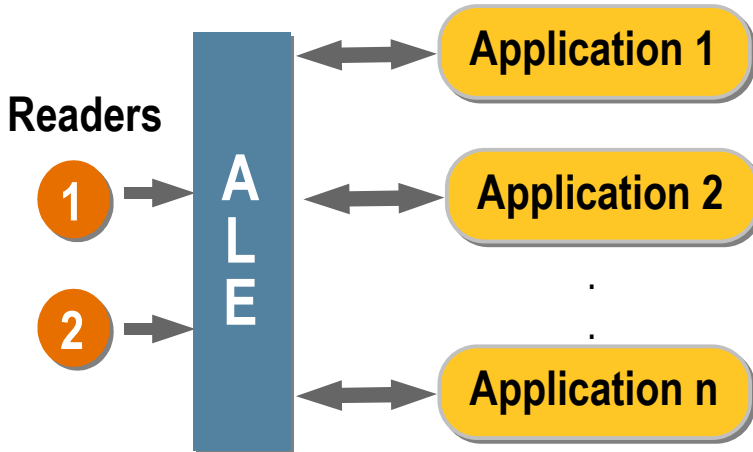
Why ALE is Important?

Readers



An Application-centric approach on incorporate RFID data is not scalable

What if you need a new application to access data from your existing readers?

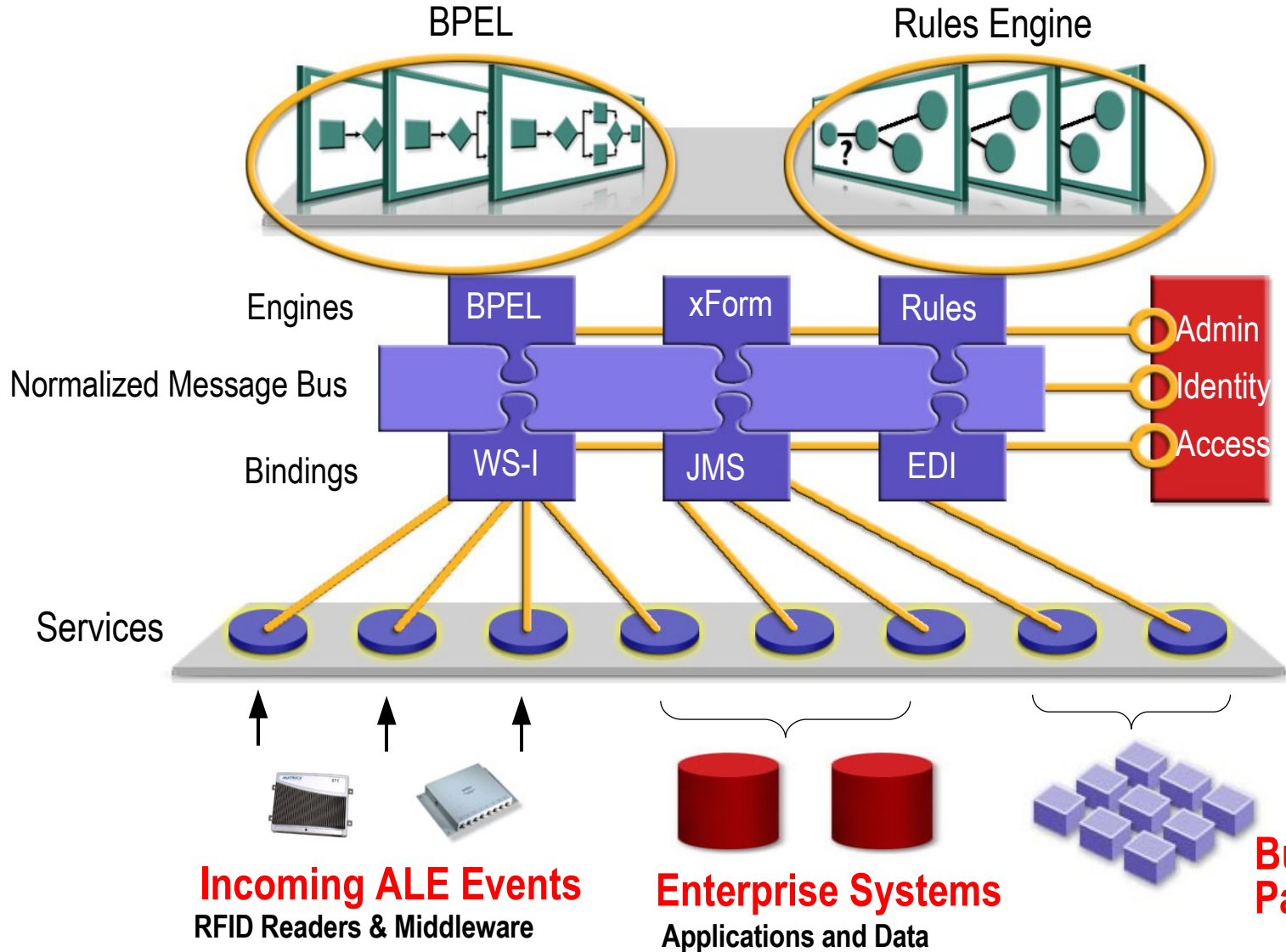


ALE-based middleware enables a more scalable approach to integrating RFID

In this example, with the same set of readers

- Application 1 can request RFID reads only when an object enters or leaves a door
- Application 2 can request RFID reads every 10 seconds for inventory tracking
- Application 3 can request all RFID reads whenever they happen

RFID as a SOA Service



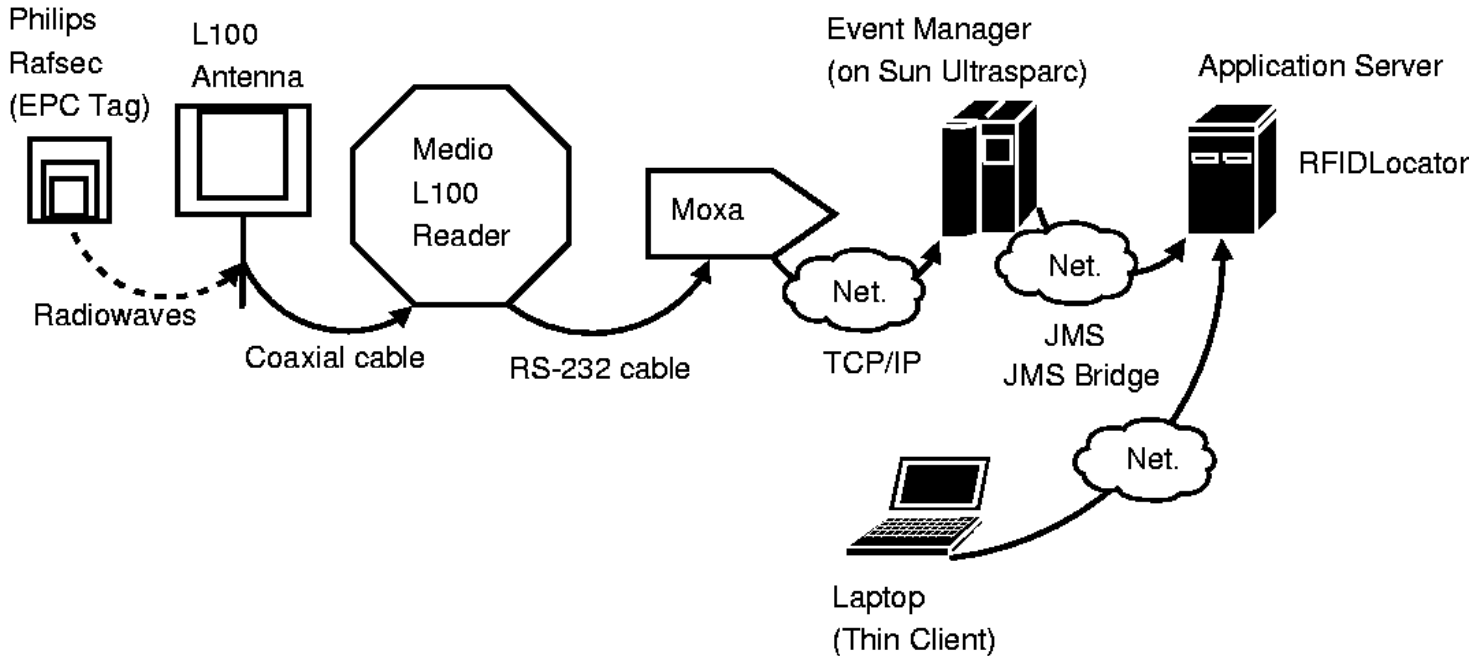
Example: RFIDLocator



- Goal:
 - > track assets within a predefined area (e.g. a building)
 - > put RFID tags on physical objects & deploy readers
 - > offer a user interface to query the location of assets
- Example 1 - the Data Center:
 - > One RFID transponder on each hardware asset
 - > Fixed RFID readers at the entrance of the server room
- Example 2 - the Hospital:
 - > One RFID transponder on each medical file
 - > Mixed use of fixed and handheld readers

System architecture

RFIDLocator is a J2EE application deployed in Sun Java System Application Server



Tags
EPC Philips RAFSEC



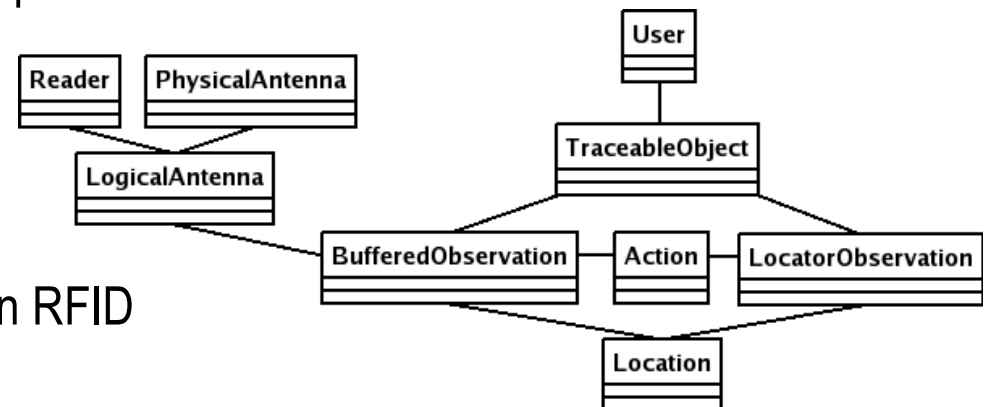
RFID Reader
Medio L100



Converter
Moxa Nport Express

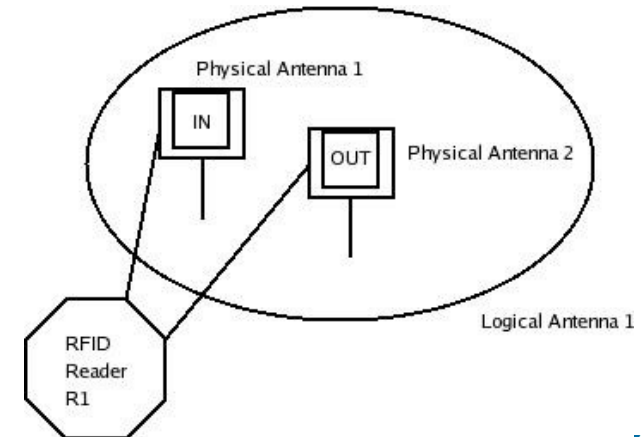
Object model

- **Reader**: models an RFID reader (aka Sensor).
- **PhysicalAntenna**: a hardware component able to capture RFID events.
- **LogicalAntenna**: Groups 1..n PhysicalAntennae.
- **LocatorObservation** and **BufferedObservation**: results of an RFID event.
- **TraceableObject**: models assets equipped of an RFID tag and traced by the application.
- **Location**: a “place” within the area controlled by the RFIDLocator.
- **Action**: action assigned to the RFID events: either **IN** or **OUT** of the Location.



Example

- One RFID reader incapable of detecting the motion direction.
- Two PhysicalAntennae: one reports IN observations, the other OUT observations.
- One LogicalAntenna aggregating both PhysicalAntennae.
- The LogicalAntenna reports LocatorObservations attached to either IN or OUT actions for “room RM.3-113”.
- This decision relies on the Solver (algorithm) of the LogicalAntenna.



Summary

- Do not ignore physics testing
- Push RFID/Sensor data processing to the edge
- Don't assume the Network will always be available
- Plan for remote management; IT support will not always be available
- Pay attention to a Reliable and Scalable architecture that can accommodate RFID and Sensor data
- RFIDLocator's website: www.gmipsoft.com/rfid