OSGi Alliance IoT / oneM2M discussion

Christer Larsson
VP EMEA OSGi Alliance
CEO Makewave

2016-03-16, Sophia Antipolis

What is OSGi?

- A modular runtime for the Java Virtual Machine:
  - Modules can be added to or removed from a running framework
  - Module dependencies are enforced at runtime
  - Remote management and monitoring capabilities
- First developed as a home automation platform
  - Lightweight implementations exist
  - Proven on a large variety of hardware configurations
- OSGi is an Open Standards body
  - Specifications are royalty free to implement
  - All API is published under the Apache 2 License

What is the OSGi Alliance

Founded in 1999

Proven, Mature Software Architecture

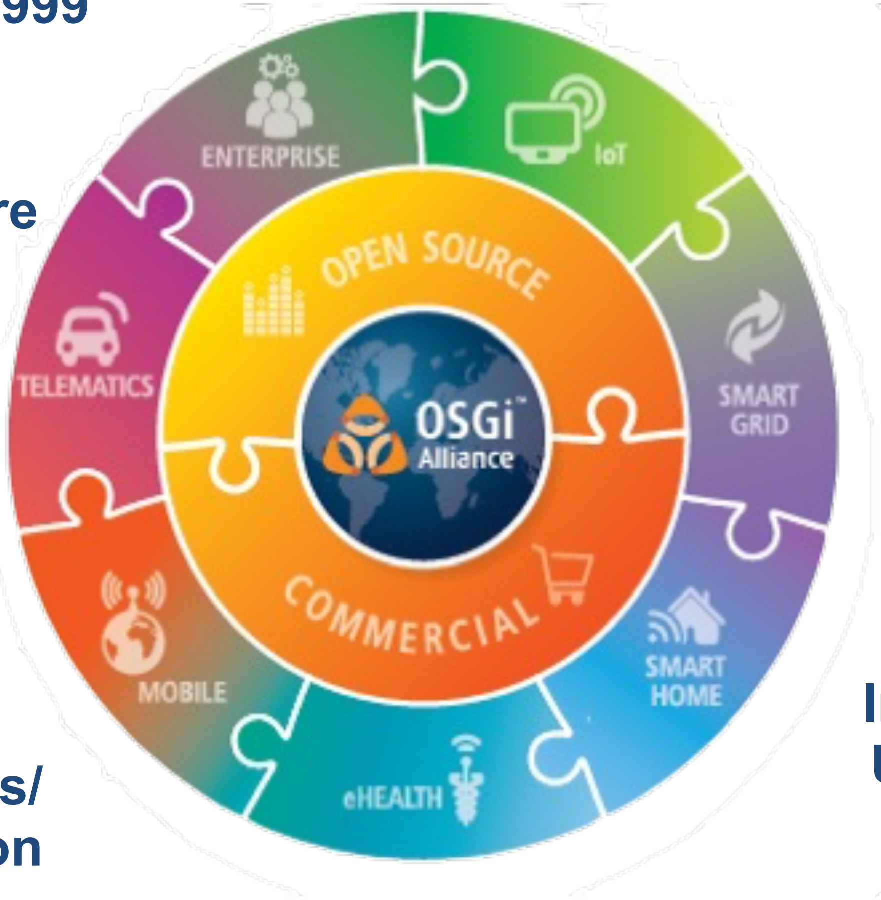
Transparent Development Process

Strategic Partnerships/ Collaboration

Global Ecosystem

Best Practices

Industry & End User Adoption



The OSGi Alliance

- The OSGi Alliance is a global non-profit technology corporation and counts many leading software vendors, telcos and other organizations among its membership. A wide range of open source projects and commercial products use OSGi technology for IoT, cloud and enterprise markets.

- OSGi Alliance members include:

Deutsche Telekom · HUAWEI · Paremus · Adobe · TELECOM ITALIA
NTT · ORACLE · LIFERAY · IBM · orange
ProSyst Bosch Group · software AG · makewave · HITACHI

OSGi Alliance Deliverables

- To foster a valuable cross-industry ecosystem, the OSGi Alliance delivers:
  - Specifications
  - Reference Implementations
  - Test Suites
  - Certifications

We are proud to be a **democratic**, **collaborative**, and **non-profit** organization that is operating in a **fully transparent** environment and **open to everyone** to join and contribute.

More Info

OSGi Alliance
Bishop Ranch 6
2400 Camino Ramon,
Suite 375
San Ramon, CA 94583
USA

Phone: +1 (925) 275-6690
Fax: +1 (925) 275 6691

Email: help@osgi.org

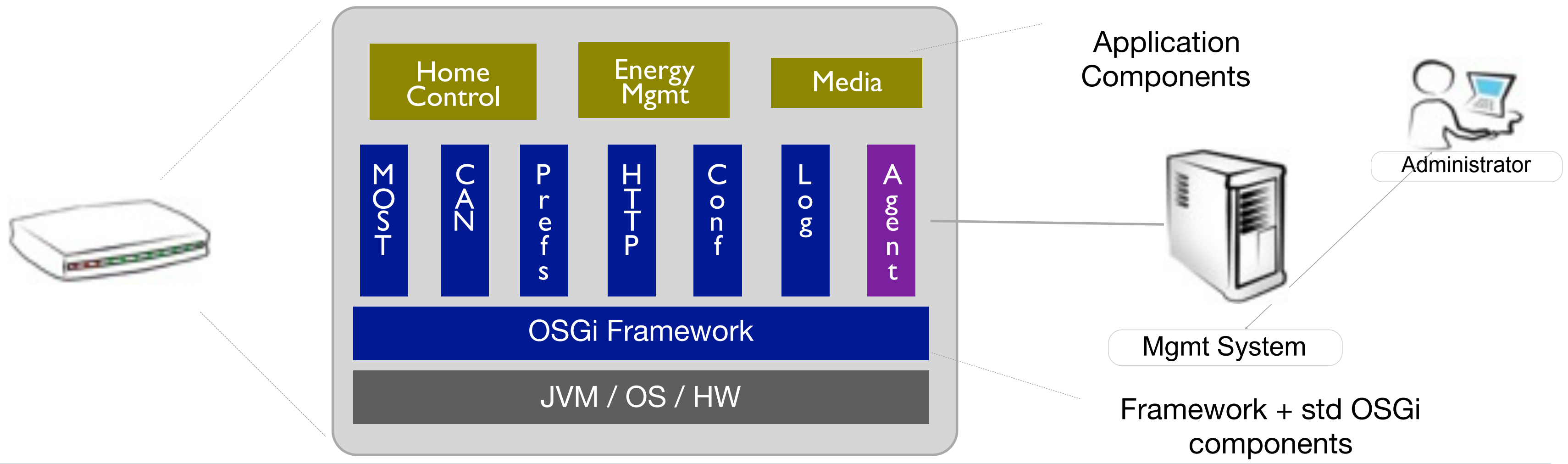
Online: www.osgi.org

Twitter: @OSGiAlliance

LinkedIn:
https://www.linkedin.com/groups/122461

What is the OSGi Service Platform?

- OSGi is a Standardized Software Execution Environment
  - Component based module system defined in Java
  - Open Standard defined by the OSGi Alliance
  - Service oriented & remotely managed (OMA & TR-69)
  - Works like an operating system for small applications called Bundles
  - Ideal for a home gateway, m2m gateway, or similar equipment



Why is OSGi helpful?

- Runtime heterogeneity
  - Not all systems offer the same capabilities (e.g. available memory)
  - Some systems are just totally different! (e.g. MIPS vs x86)
  - OSGi can transparently model these differences using Requirements and Capabilities

- Application complexity
  - As applications grow in size they become more complex and interlinked
  - Managing application dependencies becomes a full-time role
  - OSGi's runtime module enforcement allows complexity to be managed and understood — important from a governance & security perspectives!

Why is OSGi helpful?

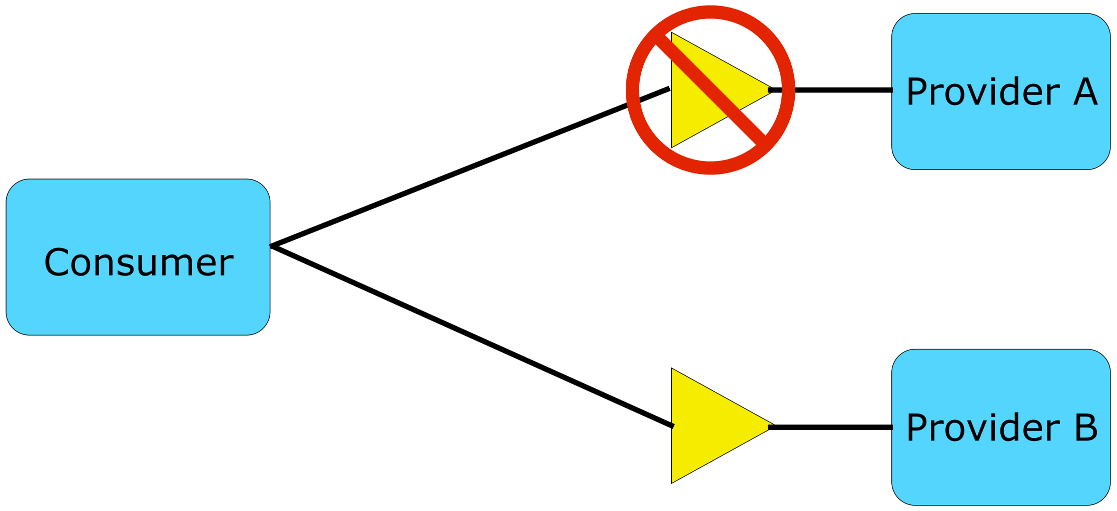
- Remote Management
    - Remote management is built in to the OSGi standards (DMT)
    - Remote Management Protocol is plugable (TR-069, OMA)
    - Management can also be REST based, or using remote calls standard Framework APIs
    - Configuration management is also available,

- Dynamic systems
    - The Dynamic nature of OSGi means that it can easily adapt to changes in its surroundings
    - Naturally address the hierarchical / locality requirements for IoT - Edge, Access, Aggregation & Core - ability to migrate functionality and flow processing between each layer.

The OSGi Service Registry

Most OSGi Users focus on creating modules

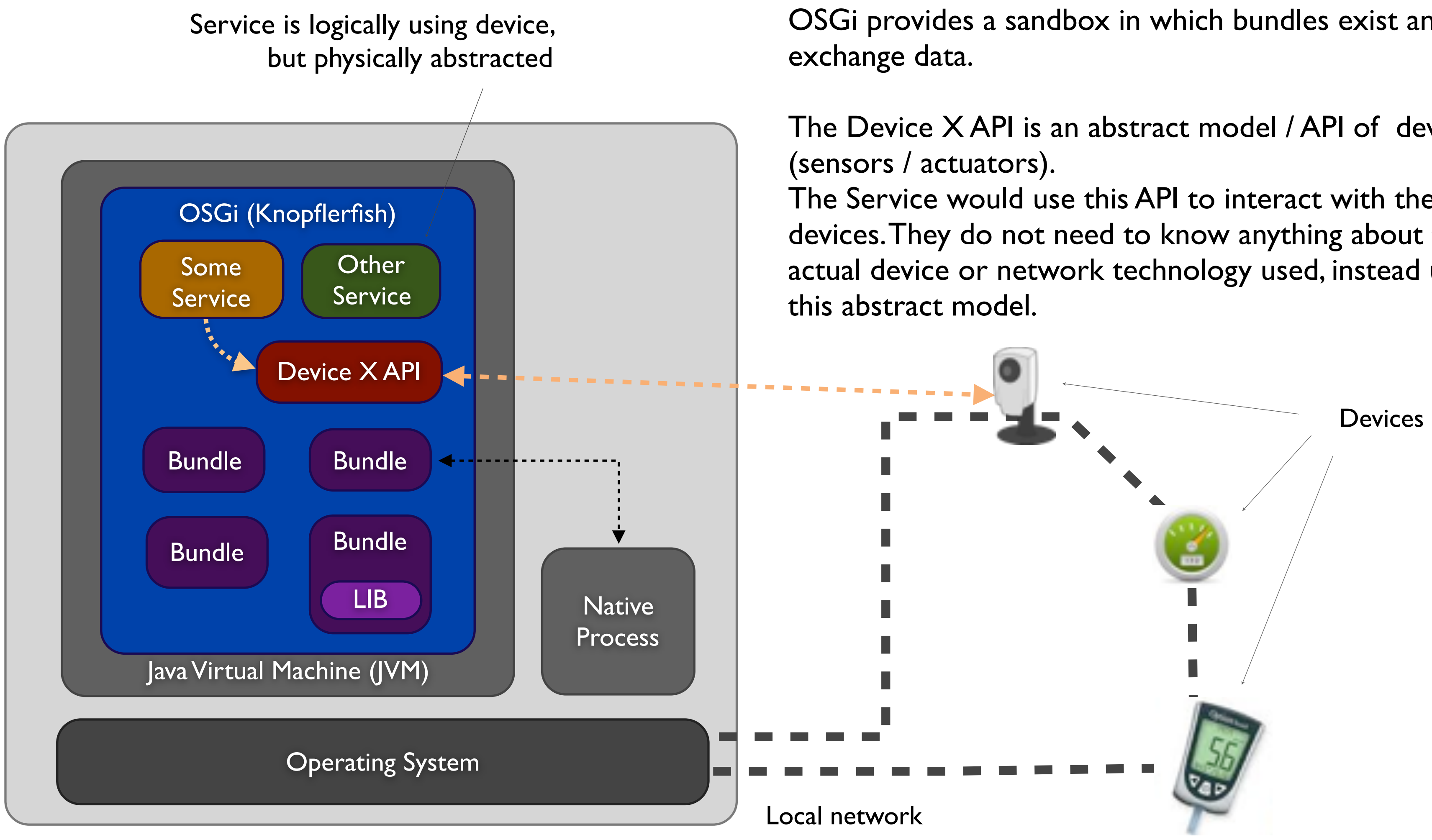
- The real power of OSGi comes from dynamic services
- A lightweight, loosely-coupled way to share implementations



OSGi as an embedded integration platform

OSGi provides a sandbox in which bundles exist and exchange data.

The Device X API is an abstract model / API of devices (sensors / actuators).
The Service would use this API to interact with the devices. They do not need to know anything about the actual device or network technology used, instead use this abstract model.



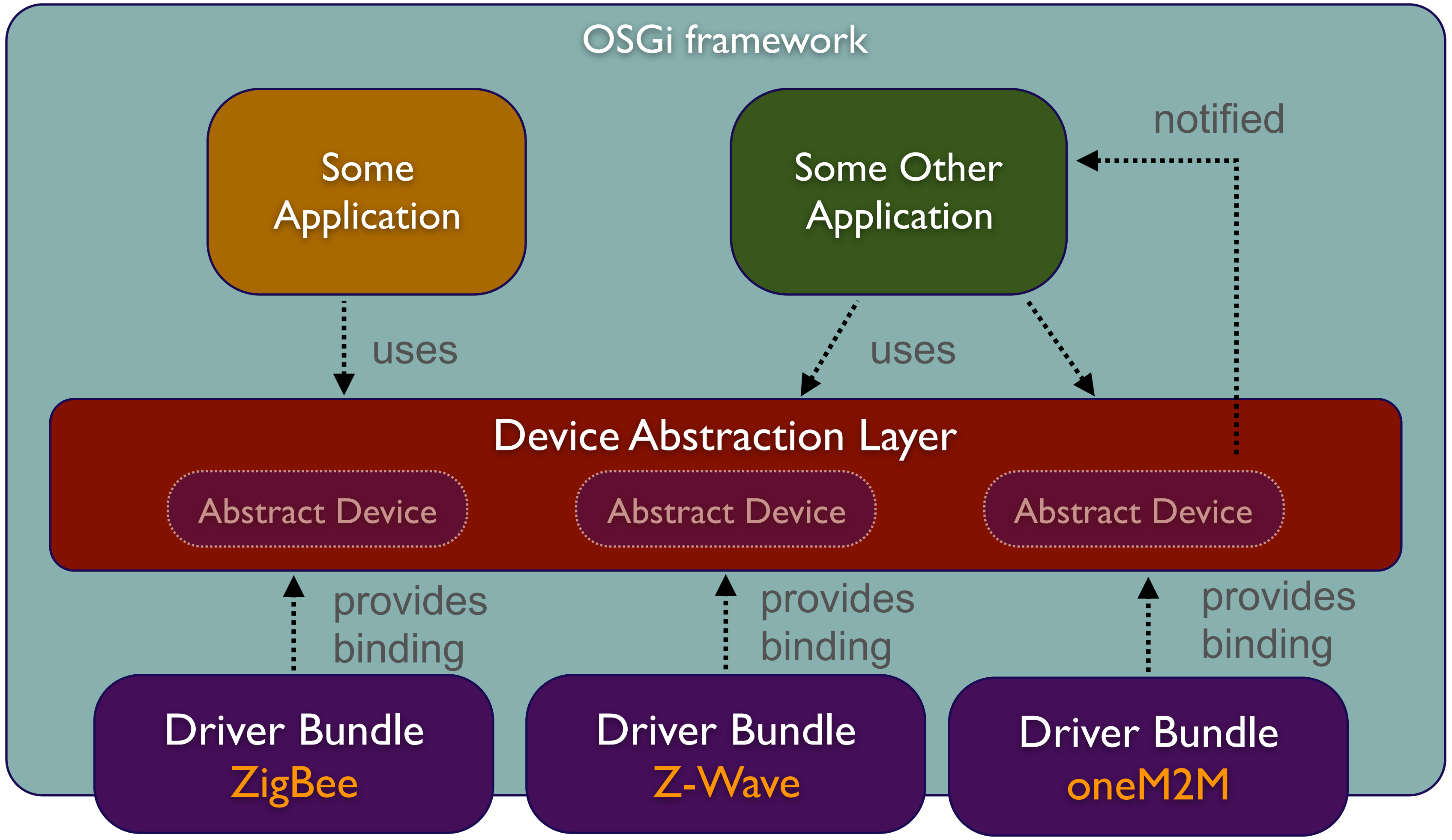
Dynamic OSGi for IoT

- Dynamic services are the perfect way to represent connected devices
    - When discovered or activated then register a service
    - When disconnected or “lost” then unregister the service

- Standard injection frameworks allow components to be notified of the services
    - Components automatically react to new devices
    - Components automatically react to device “failures”
    - Components automatically recover when a device reconnects

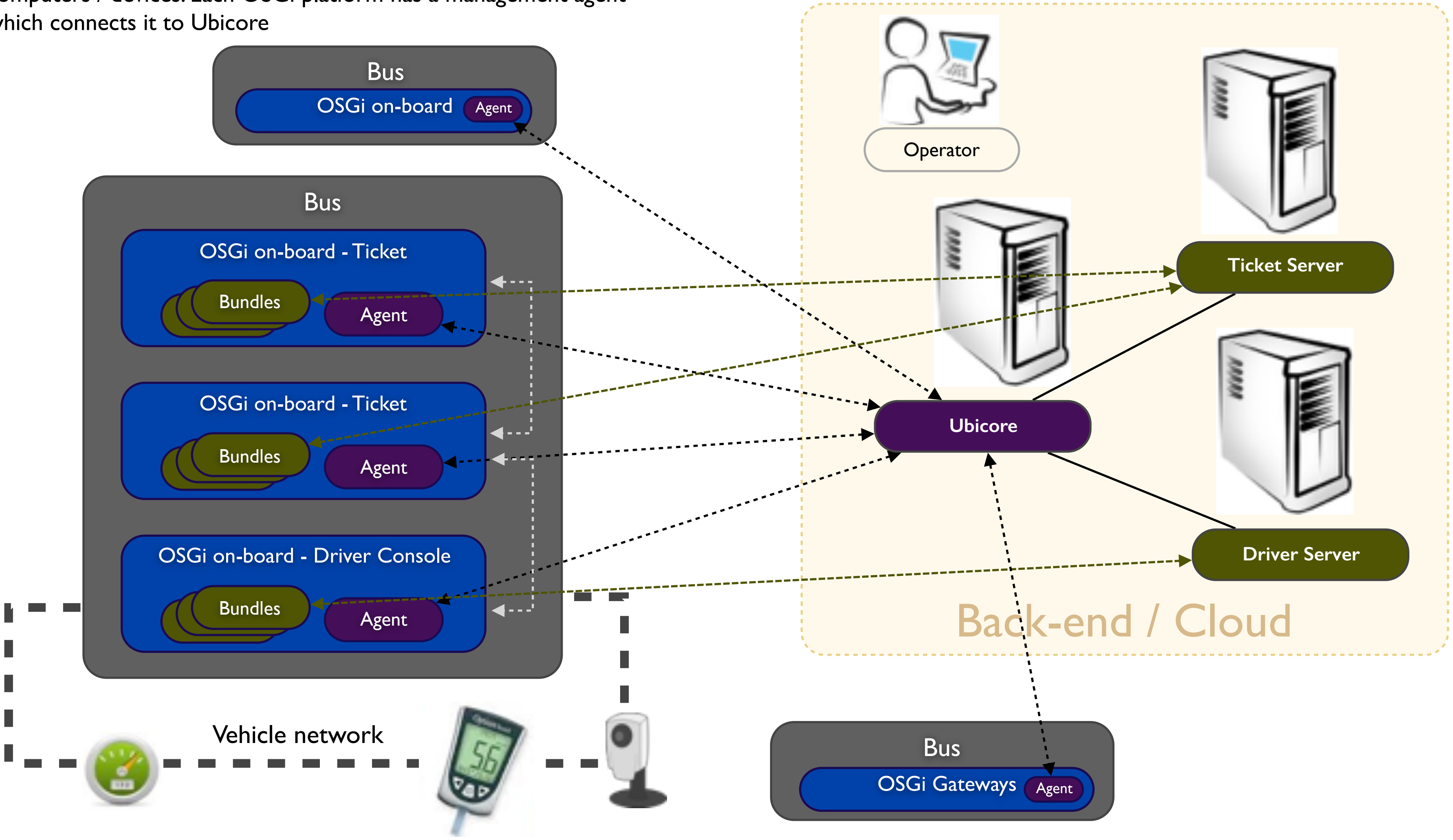
- OSGi services are therefore the perfect way to describe IoT “things”
    - Components are abstracted from the discovery mechanism
    - No need for separate UPnP/mDNS/CoAP multicast handlers

OSGi defines a Device Abstraction Layer



An example of an OSGi IoT Use Case

All buses are equipped with one or more OSGi based on-board computers / devices. Each OSGi platform has a management agent which connects it to Ubicore



Use Case - Bus Fleet System

- Every bus has one or more on-board computers that has an embedded OSGi platform (Knopflerfish)
- The on-board computers are connected to the in-vehicle network (CAN)
- The on-board computers are connected to each other (in-vehicle IP)
- Every on-board computer has an OSGi management agent. The agent is responsible for the connection to its remote management system (Ubicore)
- All business logic is implemented as OSGi bundles. The business logic bundles are all managed via Ubicore.
- The business logic bundles sends / receives data directly to/from its corresponding back-end server. I.e. the IP traffic is not routed via the management server
- The Ubicore server is part of the complete back-end server infrastructure. It is integrated with other parts via a REST API.

Benefits with an OSGi solution

- Clear separation between business logic (implemented as OSGi bundles) and lower lever parts device code (C/C++).
- Uniform architecture - exactly the same business logic bundles can be used regardless of on-board computer architecture (ARM, X86). No need to recompile, or support different versions.
- Remote Managed and Flexible - it is very easy to add, remove or update functionality in the business logic layer over time.
  - E.g. jf a new function is needed 3 months after system deployment just develop the new bundle and deploy it into the system. The mgmt agent will make sure it gets installed on all vehicles.

The OSGi Alliance IoT Expert Group

The OSGi Internet of Things Expert Group (IOTEG) is chartered to define the technical requirements and specifications to tailor and extend the set of OSGi Specifications to address information technology software infrastructure in Internet of Things scenarios.

Dynamic Behaviours in IoT

"IoT systems are fundamentally different from the software systems that we build today. The systems are so large and have so many distributed parts that they will never all be available simultaneously. ***The only way for an IoT system to succeed is to design it so that it can continue to function despite a constantly changing set of catastrophic failures.***

April 2015

IoT systems are constantly changing

- New devices are added all the time
- Existing devices break, are removed, or run out of power
- Network interruptions may partition large groups
- Evolving protocols and standards.

Embracing planned & unplanned changes is the only way to succeed

The IOTEG areas of concern include:

- To support application developers in the creation of IoT services
- Where embedded and cloud environments intersect with endpoint devices
- Data processing and management in IoT gateways and the cloud
- Cross-industry and cross-protocol device connectivity on level of actors/sensors and IoT gateways
- Support the development and deployment of OSGi device abstraction layer and endpoint ontologies
- The virtualization of IoT services
- Connectivity to the cloud for endpoint devices and interoperation with existing management systems and protocols
- Enable and enforce end-to-end IoT security

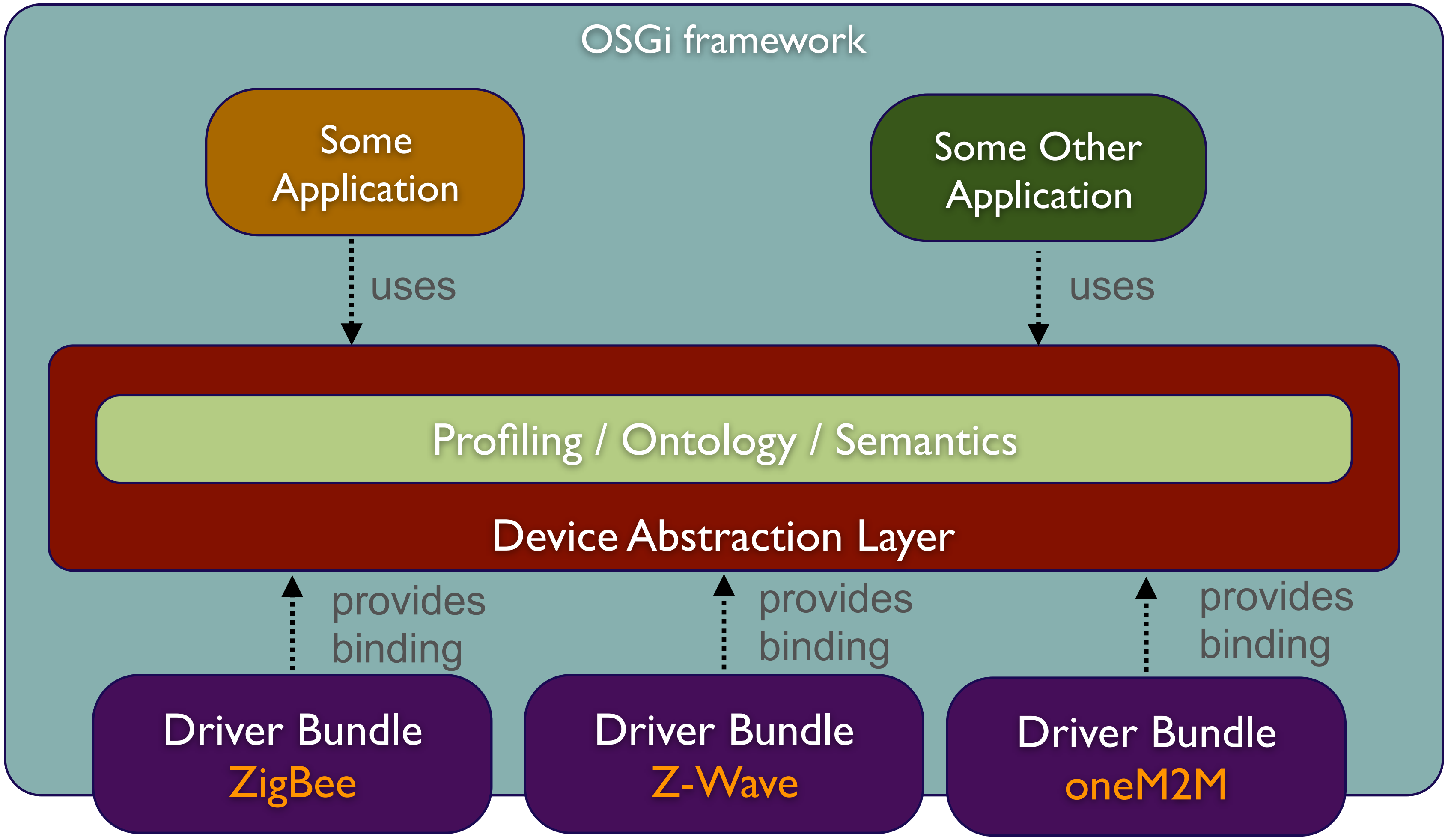
Current IoT EEG EG Activities

- Device Abstraction Layer update
  - Simplified Driver service registration
  - Simpler refinement of devices
- MQTT event handling abstraction
- CoAP resource discovery and access
- Push Streams & Distributed Eventing in progress with EEG.
- Security Best Practices
  - Security test suite

- Future possibilities
  - Software Defined Network management
  - Dynamically controlling Service QoS / Properties.
  - Resource Management
  - Offline store for batch upload?

OSGi and oneM2M

Enhancing the OSGi DAL



Thank You!

Christer Larsson
VP EMEA OSGi Alliance
http://www.osgi.org

CEO Makewave
cl@makewave.com
http://www.makewave.com