The Virtual OSGi Framework

Jan S. Rellermeyer

“Inaugural Talk”

Invited Researcher of the OSGi Alliance

Systems Group
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
Virtual OSGi? VOSGi?

- Unintended name clash
- VOSGi is work by Stephane Frenot et. al.
- Share services among virtual gateways

[Y. Royon, S. Frenot: *Un environnement multi-utilisateurs orienté service*. In: CFSE 2006]
Remember EclipseCon 2007…

- R-OSGi
- Originally motivated by embedded systems
- Service Discovery via SLP

[R.-OSGi: Distributed Applications Through Software Modularization. In: Middleware 2007]
R-OSGi Today

- Point to point
- Explicit connections
- Service Discovery is an optional part
- Closer to OSGi
  - RemoteServiceReferences known after lease exchange
  - Proxy is generated when the service is retrieved by a client
  - More feasible for remote services on the server side.
  - Gives “hints” where to connect to.

Gives “hints” where to connect to.

Already synchronized

Transparently “import” the remote service into the local framework
R-OSGi as an ECF Remote Service Provider

- ECF API on top of R-OSGi

- Thereby, non-transparent access on top of a transparent service approach
- E.g., asynchronous service invocation
Motivation: Distributed Service Registry

• Original R-OSGi
  ♦ Service Discovery

• R-OSGi 1.0.0.RCs
  ♦ Pair wise joined Service Registry
  ♦ Still service registry and remote service registry

• ECF Generic Provider
  ♦ Server and DSOs

• Other possibility: Unified service registry for both local and remote services

“The network is the service registry”

“Remote Service Registry = union of connected service registries”

“Centralized service registry”
Motivation: Tool for “Orthogonal Distribution”

How does it work?

Proxy Bundle

Service

Channel Multiplexer

Channel Endpoint

Channel Endpoint

Channel Endpoint
The Problem of State

- Failover
- Load balancing
- Real Services are not always stateless.

Could’t state be preserved?
Service replicas instead of just copies?
Motivation: Sensor Nodes as OSGi Services

• Tmote Sky
  - TI MSP430F1611 microcontroller at up to 8 MHz
  - 10k SRAM, 48k Flash + 1024k serial storage
  - 250kbps 2.4 GHz Chipcon CC2420 IEEE 802.15.4 Wireless Transceiver

• Cannot even run an OS
  - Runs TinyOS

• But it can be an R-OSGi service…

Demo

Extending the Idea of OSGi Services

Why should a remote OSGi service have to be written in Java?

- C/C++
- CLDC
- Over Bluetooth, …

Would be nice to have this for local services as well

The Virtual OSGi Framework

What about consuming services

• The consumer has to be an OSGi framework
• But, …
• Couldn’t it be

The Virtual OSGi Framework
The Virtual OSGi Framework

• OSGi on the cloud
  ✷ Have a network full of machines running OSGi
  ✷ Don’t care where they are
  ✷ Don’t care where bundles are installed
  ✷ Don’t care where services are provided
  ✷ Access them from anywhere

• Fluid OSGi
  ✷ Have a replica where you need it
  ✷ Read any / write any
Architecture

- Unifying local and remote services
- As an extension, non-invasive against the framework
Architecture

• Unifying local and remote services
• As an extension, non-invasive against the framework

Equivalent for a peer on the cloud
Virtualized Module Layer

- The Virtual Framework runs as a bundle on the host framework
- Virtual Bundles are installed on the host framework
- Virtual Bundles are started on the virtual Framework
Virtual Bundles

• Installation of the bundle
  • Install on the host framework
  • Pass back a VirtualBundle instead of the host framework’s Bundle implementation

• Starting the bundle
  • Called through a VirtualBundle
  • get the Activator from the host framework
  • Call it with a VirtualBundleContext
  • Handle the virtual state of the bundle within the virtual framework
  • Subtile: ensure BundleID consistency
Distributed Registries

• Centralized registries are replaced by a distributed registry
• Prototype system: kind of DHT
  - Can store pointers to bundles
    ▪ Supports constraints
  - Can store pointers to services + attributes
    ▪ Supports filters
Challenges

- Mapping the class space model to the DHT

- `getAllServices` becomes a very expensive operation

- Maintaining replicas of DHT nodes

- Scalability?
  - Can it scale to massively distributed systems?
  - Can it scale to the diameter of the internet?
OceanStore?

Service Replication

• Fluid Replication
  ◦ Place a replica of the service where ever it is needed
• Preserve the state between service replicas
• Prototype: Communication model through the DHT
• Coordinator nodes
  ◦ For update propagation
  ◦ For using different consistency levels within the same virtual framework
PRACTI?

- Partial Replication
- Arbitrary Consistency
- Topology Independence

Replication of Services

• What is state?
  - Model: Only services have state
  - State is contained in fields

• Capturing state?
  - Update propagation
  - Arbitrary consistency

• Goal: Transparent replication
  - Run with every OSGi Service
  - Requires no changes
OpenTerracotta?

• Transparent clustering
• Load time instrumentation
• Distributed locking

[http://www.terracotta.org]
Instrumentation

• Symbolic Execution

• Instrumentation to capture fields

• Also does distributed locking, distributed thread coordination

• Also used for service migration

But we also handle thread migration.

Find out where state is accessed/changed

P2P update propagation through group communication

Seamless parallelization

Can be considered as a temporal replication
What we have now…
Coordination overhead

- Coordinating all the replicas
- Affects scalability

Duality between Service Replicas and Remote Services
Outlook: Autonomous Controller

[Ramon Küpfner, Dario Simone; starting soon]
Outlook: Language-independent OSGi

• Soon a prototype for Barrelfish (new operating system at ETH)
  - Use the kernel-provided IPC model
  - Provide an application model (derived from OSGi)
  - Optimize for interactions within the same language
  - Provide generic type mappings for heterogeneous apps

[with Simon Peter, Adrian Schüpbach, Andrew Baumann, Timothy Roscoe]
Conclusions

• The Virtual OSGi Framework
  - Unifies local and remote services
  - Makes a (dynamic) group of machines appear as a single OSGi framework
  - Allows replication of services for load balancing or to increase failure resilience
  - Runs as a bundle on top of every framework
  - Uses the host framework for module layer operations
  - Intercepts/extends certain operations on the virtualization layer
  - Can relocate bundles/services
  - OSGi on the cluster/cloud
Welcome to the virtual world!

• Questions?