Mobilizing the Cloud with AIOLOS

Tim Verbelen (tim.verbelen@intec.ugent.be)

iMinds – Ghent University

EclipseCon Europe – 29.10.2014
The Problem...

- Connect all kinds of wearable devices in your applications together ...
Mobile application development now

Suppose the following application:

viewer → logic → storage

sensors
OSGi to the rescue...

We need software modularity!

viewer

logic

sensors

storage

OSGi

OSGi

OSGi

OSGi
AIOLOS

We need transparent remote communication!

viewer → logic → sensors → storage

AIOLOS
OSGi

AIOLOS
OSGi

AIOLOS
OSGi

AIOLOS

iMinds
UNIVERSITEIT GENT
AIOLOS: bundle distribution at runtime

AIOLOS provides:

- Service discovery and binding
- Fast remote service calls
- Runtime (re) deployment
- Service monitoring

http://aiolos.intec.ugent.be
Don't tell me, show me!

- Tardis – an immersive picture viewer
AIOLOS Architecture

- Repository
- Deployment Manager
- Cloud Manager
- Service Monitor
- Node Monitor
- Proxy Manager
- Topology Manager
- Remote Service Admin
Remote Service Admin
Remote Service Admin

- Lightweight implementation of the RemoteServiceAdmin 1.0 spec
- Protocol based on R-OSGi
- Fast serialization with Kryo* and Objenesis**

*https://github.com/EsotericSoftware/kryo
**http://objenesis.org/
How fast is “fast”?
Remote Service Admin

- Export and import a simple Greeting service on localhost
  ```java
  String greet(String name) { return "Hello " + name; }
  ```
- Perform 10000 calls
Remote Service Admin

- Export and import a simple Greeting service on localhost
  
  ```java
  String greet(String name) { return "Hello " + name; ;
  }
  ```

- Perform 10000 calls
Proxy Manager

- Repository
- Deployment Manager
- Cloud Manager
- Service Monitor
- Node Monitor
- Proxy Manager
- Topology Manager
- Remote Service Admin

Deployment

Monitoring

Core
Proxy Manager

- Create a smart proxy for each OSGi service

- Use OSGi Hooks
  - service.EventHook : create proxy before service is registered
  - service.FindHook : hide original service reference

- Implement a ProxyPolicy to select service implementation
Topology Manager

- Platform Manager
  - Repository
  - Deployment Manager
  - Cloud Manager

- Core
  - Service Monitor
  - Node Monitor

- Proxy Manager
  - Topology Manager
  - Remote Service Admin
Topology Manager

- Export local application services
- Import remote services that are requested on the local framework
  - i.e. using service.FindHook
- Discover other nodes in the network
  - i.e. SLP discovery implementation
Monitoring

- Repository
- Deployment Manager
- Cloud Manager

- Service Monitor
- Node Monitor

- Proxy Manager
- Topology Manager
- Remote Service Admin

Deployment
Monitoring
Core
Monitoring

- Node monitoring
  - Access to device parameters (e.g. CPU usage, ...)
  - Implemented by reading /proc/ filesystem on *nix

- Service monitoring
  - Collect method execution information using callbacks from the proxies
  - Available metrics: execution time, size of the arguments, size of the return value, ...
What's the overhead?

There is a generation of kids that have no idea what this is.
Overhead

- Perform 1 million `greet` method calls on local instance with and without proxying/monitoring

Overhead of proxy/monitoring less than 1 us
Deployment Manager
Deployment Manager

- Provides an interface to start/stop application components on a node

- Also provides (stateless) bundle migration:
  - Start component on target node
  - Once started, switch all proxy references
  - Stop component on source node
Repository

Platform Manager
- Repository
- Deployment Manager
- Cloud Manager

Monitoring
- Service Monitor
- Node Monitor

Core
- Proxy Manager
- Topology Manager
- Remote Service Admin
Repository

- Implements the OSGi Repository 1.0 spec
- Makes application bundles available for installation at runtime
- Is used by the Deployment Manager to find and resolve artefacts to be deployed
Cloud Manager
Cloud Manager

- Allows for automatic scaling in a cloud
- Uses Apache jClouds* for cloud interfacing
  - currently tested with Amazon EC2 and OpenStack
- Enables application scaling on a service-level
  - i.e. by using a round-robin ProxyPolicy

*https://jclouds.apache.org/
Use cases
Mobile offloading

- Choose the fastest instance to execute the call
  - Monitor information allows to estimate local/remote time
  - Smart proxy enables fast switching

Component-based cloud scaling

- Scale specific application components based on service monitoring information
  - Scale both VMs and components
  - ProxyPolicy determines load balancing strategy

[S Bohez, T Verbelen, P Simoens, B Dhoedt, “Enabling Component-based Mobile Cloud Computing with the AIOLOS Middleware”, ARM’14, Bordeaux, France]
Thank you!

An OSGi-based platform bridging the gap between mobile devices and the Cloud.

Website http://aiolos.intec.ugent.be
Source https://github.com/aiolos-dev
Repositories http://aiolos.intec.ugent.be/repositories