Mobile Cloud Networking: Hurtle, Cyclops, Gatekeeper

Prof-Dr. Thomas Michael Bohnert (TMB)
Challenges

• Deliver your software as a service?
• How to compose existing services?
• How deliver and maintain reliability?
• How to monetise your software?
The Challenge

How to offer your software as a service?”

- Automate the life-cycle management of your service, from deployment to disposal
- Recursive service composition
- Designed for Cloud-Native Applications
- Designed for Cross-Domain Orchestration
Implementation

Dependencies:
- OpenShift
- OpenStack
→ Abstracted through the Cloud Controller

Standard:
- OCCI
Key Components of Hurtle

- **Service Manager (SM)**: receives requests for new tenant service instances
  - [https://github.com/icclab/hurtle_sm](https://github.com/icclab/hurtle_sm)
- **Service Orchestrator (SO)**: manages the lifecycle of a tenant service instance
  - Sample [https://github.com/icclab/hurtle_sample_so](https://github.com/icclab/hurtle_sample_so)
- **CloudController (CC)**: manages and abstracts underlying resources and SOs
  - [https://github.com/icclab/hurtle_cc_api](https://github.com/icclab/hurtle_cc_api)
Features

- Complete **orchestration of your software** lifecycle
  
  Easy implementation of your **service API** - See [how to write your Hurtle Service](#)

- Guided implementation of your **service manager**

  Many languages supported including Python, Java, Perl, PHP, Demo applications available

- Scalable **runtime management**

  Complete **end-to-end logging** of your software

- Integration with [OpenStack](#), [ICCLab's Joyent SDC contribs](#)

- Handle potential **incidents of your software**,

  On-Going Integration with [ICCLab's Watchtower (Cloud Incident Management)](#)

- Leverages **Open Cloud Standards** ([OCCI](#), [OpenStack](#)), Multi-dc/multi-region support

- **Bill for your software** and services,

  Integration with [ICCLab's Cyclops (Rating, charging & Billing)](#)
Roadmap

- More examples including the cloud native Zurmo implementation from ICCLab

- Enhanced **workload placement**, dynamic policy-based

- Support for **docker** registry deployed containers: OpenShift v3

- **Runtime updates** to service and resource topologies

- **CI** and **CD** support
  - safe monitored dynamic service updates

- **TOSCA** support

- Support for **VMware** and **CloudStack**

- **User interface** to visualise resource and services relationships

- Additional external service endpoint protocol support
The Challenge

How to monetize your service?

- Provide a complete rating, charging, and billing service
- Able to deal with multi-domain/multi-provider service compositions
- Able to deal with dynamics inherent to metered cloud services (pay-as-you-go)
- Itself to be provided as a service – VAS for cloud operators
Key Components

**Gatekeeper:** simple authentication/authorization micro-service

**Event bus:** rabbitmq based service for collecting key events, including SLA violations

**udr-microservice:** ‘usage’ data collection, transformation and storage + UDR generation

**rc-microservice:** rule based rating engine - rate generation, and CDR generation and storage

**billing-microservice:** CDR aggregation and bill generation (pull based), discounts, penalties, coupon processing, VAT rules, etc.
Technology Landscape

Codebase mostly written in Java + Frontend written in Java & Angularjs

Gatekeeper code written in go!

Database: Influxdb (tsdb)

Rule engine: drools

Scheduler: will be replaced by in-house scheduler

REST interface developed using restlet framework

Message broker: Rabbitmq

Inter-microservice line message format: json
Roadmap

● Data collection failure tracking and recovery mechanism
  ○ Keeping track of failed collection periods
  ○ Lazy recovery attempts to fill usage data for missing timeline entries

● Light-weight marketplace in dashboard
  ○ for proof of concept and demonstrations
  ○ ISV /app developer view - revenue reports, deployments tracking and metrics visualization
Links

HURTLE

GitHub: https://github.com/icclab/hurtle
• Architecture
• Implementation
• Write your own service

Website: http://hurtle.it/, Twitter: @hurtle_it, Mailing List: icclab-hurtle@dornbirn.zhaw.ch

Advanced Service Composition: https://www.youtube.com/watch?v=03YiBT3lM9s

CYCLOPS

All about RCB and CYCLOPS
http://blog.zhaw.ch/icclab/category/research-approach/themes/rating-charging-billing/

GitHub:
Questions

http://blog.zhaw.ch/icclab/
Hurtle & the ICCLab
Implementation in Practice
Runtime Module

Automatic alarm creation for each new service provisioning, with callback to the service orchestrator. Monasca as technology.
Write your own, easily!

- To create a new service, write a **Service Definition** and a **Service Bundle**
  - [Service Definition](#)
  - [Service Bundle](#)
    - Service Orchestrator: Your service’s logic
    - Service Manifest: Your service dependencies
    - Heat Template: The resources your service needs

- Testing is easy
  - Service Def. is an executable python app
  - run it, then send OCCI requests, e.g.
    - `curl -X POST http://localhost:8888/example/ 
      -H 'Category: example; scheme=http://schemas.hurtle.it/occi/sm#';
      -H 'class="kind"';
      -H 'content-type: text/occi';
      -H 'x-tenant-name: YOUR_TENANT_NAME';
      -H 'x-auth-token: YOUR_KEYSTONE_TOKEN'`