ORBIT Radio Grid Emulator

Dr. Maximilian Ott  (max@winlab.rutgers.edu)
Orbit as Service

• Should be:
  – Easy to use
  – Reliable
  – Useful

• Not characteristic of academic projects, but the rule for commercial projects
  – Instead of telling you about our latest research
  – … asking for your help and expertise
ORBIT Testbed: Radio Grid
10000 Feet View

Experiment
Life-cycle

Infrastructure
Experiment Life-Cycle

- Definition
- Scheduling (not yet)
- Setup
- Execution/Measurements
- Analysis
Definition ...

- **Static**
  - Application-based (NS later)
  - Classes of nodes
    - Set of apps and their parameters
  - “Coloring” of actual nodes
    - Now: explicit
    - Near-term: libraries of scenarios
    - Long-term: Specifying topology
- Measurements
  - What and how to measure
Architecture

User's Workstation

Operator Console
Application Description
Code Generator
Execution Harness

Grid Controller

DB
Management Station

Install
Upload

Agent
Spawn
Control

Deployed Application
Measurement FW

Node

Experiment Package
Store

max@winlab.rutgers.edu
Application Description

{Parameters}
{<type, value range, constraints>}

Application

{Measurements}
{<tuples>
Experiment definition

A simple 2x2 grid with diagonal pairs of sender/receivers on separate channels.

- **Project**: ref=winlab:wp3
- **Templates**
  - **Comment**: define templates for nodes with similar configurations (this may be included from another definition for reuse)
  - **Node**
    | id  | name  | description | param | applications |
    |-----|-------|-------------|-------|--------------|
    | 1   | sender| Nodes which send a stream of packets |       |              |
    | 2   | receiver | Nodes which receive a stream of packets |       |              |
- **Topology**
  - **Node**
    | name  | template | Comment | set-param |
    |-------|----------|---------|-----------|
    | 1     | sender   | customize template for this instance | set-param name... |
    | 2     | sender   |          | set-param name... |
    | 3     | receiver |          |           |
    | 4     | receiver |          |           |

- **Experimenter**
OML – Orbit Measurement Library

• Main result of experiment are often measurements
• Collection of measurements in distributed environments is hard
• How can we make that task simpler?
• … and get the experimenter off the testbed quickly
OML ...

Application

Filter

DB

DB

Off-line
Usage Scenarios

• Bare Hardware
  – User provides system image

• Standard OS package
  – Pick from provided library of pre-packaged images
  – User provides applications

• Standard experimental package
  – Re-run previous experiments
  – Customize existing experiments
    • Topology
    • Algorithm
    • Parameters
Life-cycle: Setup

- Power-on
- Safe boot (net)
- Image disk
- Reboot into local image
- Install additional packages
- Ready

~ 5 minutes for 400 nodes!
Defining Dynamic Behavior (Ideal)

foreach

max@winlab.rutgers.edu
Defining Dynamic Behavior

if error

if timeout
Alternative: State Machine

On Error

On Timeout
Single Testbed View

Logical View

Physical View
Dynamic Behavior through Rules

• On *//*/status is UP do {...}
• WhenAll *//*/status is UP do {...}
• When *//*/status:UNKNOWN every 5000 do {...}
... Definition

• Dynamic
  – Explicit:
    • Controlled sequence of parameter changes
  – Implicit (hard to capture upfront):
    • Application internal
    • Reactive, adaptive
# Application definition

```
<application xmlns="schema.orbit-lab.org/application/05022004">
  <id>orbit::admin::nodeAgent</id>
  <name>NodeAgent</name>
  <id>nodeAgent</id>
  <currentVersion>0.0.1</currentVersion>

  <organization>
    <shortDescription>Node Agent</shortDescription>
    <description>A node agent is active on every experiment node and communicates through semantic multicast with all other agents in the system. It can control all the resources on a node and can react to commands from the management agent to control the life cycle of experiments on the respective node.</description>
  </organization>

  <url>http://apps.orbit-lab.org/nodeAgent/</url>

  <properties>
    <property>
      <name>channel_id</name>
      <mnemonic>c</mnemonic>
      <type>int</type>
      <dynamic>yes</dynamic>
      <constraints>???</constraints>
    </property>
    <property>
      <name>payload_length</name>
      <mnemonic>l</mnemonic>
      <type>int</type>
      <dynamic>yes</dynamic>
      <constraints>???</constraints>
    </property>
  </properties>

  <measurements>
    <measurement>
      <id>group1</id>
      <metric>(2)</metric>
    </measurement>
    <measurement>
      <id>group2</id>
      <metric>(1)</metric>
    </measurement>
    <measurement>
      <id>group3</id>
      <metric>(1)</metric>
    </measurement>
  </measurements>
</application>
```
Defining dynamic behavior (Filter)

```xml
<node id="receiver">
  <name>Receiver</name>
  <description>Nodes which receive a stream of packets</description>
  <applications>
    <application refid="orbit:admin:nodeAgent">
      <measurements>
        <measurement refid="group1">
          <filter refid="average" time="100sec"/>
        </measurement>
        <measurement refid="group2">
          <filter refid="min-max" time="1min"/>
        </measurement>
      </measurements>
    </application>
  </applications>
</node>
```