MobilityFirst
GENI
Tutorial
Initial Setup

• Requirement:
  • Have a GENI Portal Account

• [https://portal.geni.net/](https://portal.geni.net/)

• Join the GENI project for the tutorial

• Tools -> Wireless Account Setup -> Enable

• You can use your credentials to access ORBIT resources
Tutorial Program

- MobilityFirst Introduction
- ORBIT Overview
- Tutorial:
  - Exercise 1: Simple MobilityFirst Network Deployment and Test.
  - Exercise 2: Measuring Performance of a MobilityFirst Router
  - Exercise 3: Socket Programming using New MobilityFirst NetAPI
Tutorial Program

• MobilityFirst Introduction
• ORBIT Overview
• Tutorial:
  • Exercise 1: Simple MobilityFirst Network Deployment and Test.
  • Exercise 2: Measuring Performance of a MobilityFirst Router
  • Exercise 3: Socket Programming using New MobilityFirst NetAPI
MobilityFirst: Motivations

Historic shift from PC’s to mobile computing and embedded devices...

- ~4B cell phones vs. ~1B PC’s in 2010
- Mobile data growing exponentially – Cisco white paper predicts 3.6 Exabytes by 2014, significantly exceeding wired Internet traffic
- Sensor/IoT/V2V just starting, ~5-10B units by 2020
MobilityFirst: Name-Address Separation

- Separation of names (ID) from network addresses (NA)
- Globally unique name (GUID) for network attached objects
  - User name, device ID, content, context, AS name, and so on
  - Multiple domain-specific naming services
- Global Name Resolution Service for GUID <-> NA mapping
- Hybrid GUID/NA approach
  - Both name/address headers in PDU
  - “Fast path” when NA is available
  - GUID resolution, late binding option
MobilityFirst: Protocol Stack

- Service ID (SID) specifies specific processing or delivery to be applied.
- GUID based network header.
- Hybrid GUID/NA approach.
- Dynamic GUID <-> NA resolution.

MobilityFirst Packet

<table>
<thead>
<tr>
<th>SID</th>
<th>Dst_GUID</th>
<th>Dst_NA</th>
<th>Src_GUID</th>
<th>Src_NA</th>
<th>DATA</th>
</tr>
</thead>
</table>

App 1 || App 2 || App 3
---|---|---|---|---|---|
E2E Transport 1 | E2E Transport 2 | E2E Transport 3 |

GUID Service Layer

GSTAR Routing | MF Inter-Domain

Hop-by-Hop Block Transfer | Switching Option

Link Layer 1 (802.11) | Link Layer 2 (LTE) | Link Layer 3 (Ethernet)
MobilityFirst: Global Name Resolution Service (GNRS)

- Fast GNRS implementation (Dmap) based on DHT between routers
  - GNRS entries (GUID <-> NA) stored at Router Addr = hash(GUID)
  - Results in distributed in-network directory with fast access (~100 ms)
MobilityFirst: Routing (GSTAR)

- Storage aware (CNF, generalized DTN) routing exploits in-network storage to deal with varying link quality and disconnection.
- Routing algorithm adapts seamlessly from switching (good path) to store-and-forward (poor link BW/short disconnection) to DTN (longer disconnections).
- Storage can have benefits for wired networks as well.
MobilityFirst: Network API

- **Service Abstractions**
  - Direct Addressability for All Network Principals.
- **Multi-Point Addressability.**
  - send(Group1, data)
  - get(A)
- **En-Route Storage and Compute.**
## MobilityFirst: Network API

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>open, close</strong></td>
<td>- <code>open(profile, [profile-options], [source-GUID])</code>&lt;br&gt;- Allocate the appropriate resources given the profile of the communication specified by the program.</td>
</tr>
<tr>
<td><strong>send, recv</strong></td>
<td>- <code>send(destination-GUID, data, [service-options])</code>&lt;br&gt;- <code>recv(source-GUID, buffer, [GUID-set])</code>&lt;br&gt;- Name based message exchange.&lt;br&gt;- By use of options ability to request set of specific network services.&lt;br&gt;- Per message destination GUID.</td>
</tr>
<tr>
<td><strong>attach, detach</strong></td>
<td>- <code>attach(GUID-set)</code>&lt;br&gt;- Management of network presence and reachability.</td>
</tr>
<tr>
<td><strong>get, post, exec</strong></td>
<td>- <code>get(content-GUID, request, buffer, [svc-opts])</code>&lt;br&gt;- Exploit the additional information on the type of network object represented by the GUID.&lt;br&gt;- Allows the client network stack to select the best transport and allocate adequate resources. By use of options ability to request set of specific network services.</td>
</tr>
</tbody>
</table>
MobilityFirst: Protocol Example 1

Dual Homing Service

Send data file to “John Smith22’s laptop”, SID= 129 (multihoming – all interfaces)

Query to GNRS
MobilityFirst: Protocol Example 2

Handling Disconnection (Store-and-Forward mobility service example)

Send data file to “John Smith22’s laptop”, SID= 11 (unicast-mobile delivery)
MobilityFirst: Prototype

Network Stack:
• C++ software level implementation that uses the pcap library to intercept and inject packets.
• API available for C/C++ and JAVA programs.
• Implements anager with support for simple migration policies (e.g. “use wifi”)

Router:
• Click based router implementation.
• Hop-by-Hop reliable transmission.
• Implements Generalized Storage Aware Routing (GSTAR) routing protocol.

GNRS:
• 2 different implementations: DMap and Auspice
• low latency ~50 ms average and ~100 ms at the 95th percentile
Tutorial Program

- MobilityFirst Introduction
- ORBIT Overview
- Tutorial:
  - Exercise 1: Simple MobilityFirst Network Deployment and Test.
  - Exercise 2: Measuring Performance of a MobilityFirst Router
  - Exercise 3: Socket Programming using New MobilityFirst NetAPI
ORBIT Overview

VPN Gateway to Wide-Area Testbed

Gigabit backbone

Front-end Servers

Data switch

Application Servers
(User applications/
Delay nodes/
Mobility Controllers /
Mobile Nodes)

Control switch

Back-end servers

80 ft (20 nodes)

70 ft (20 nodes)

RF/Spectrum Measurements

Interference Sources

Internet VPN Gateway /
Firewall
ORBIT Radio Node (Version 3 & 4)

- Core 2 Quad with Q35 Express chipset
- 4 GB DDR2
- 2 x Gigabit Ethernet ports
- PCI-Express X16
- Mini-PCI socket
- 8 x USB 2.0
- 2 x COM

- Core 2 Duo with GM45 chipset
- 8 GB DDR3
- 2 x Gigabit Ethernet ports
- PCI-Express X16
- PCI Express mini socket
- Mini-PCI socket
- 8 x USB 2.0
- 2 x COM
ORBIT Grid
Experimental readings at one location:

CINR = 29  RSSI = -51
OMF Overview

OMF, a framework for

Controlling Experiments
- Systematic description
  - Resources
  - Tasks
  - Measurements
→ Reproducibility
  (within & across testbeds)

Managing Testbed
- abstraction for many resource types
- Optimise temporal & spatial use
→ Lower setup & Operation cost

OMF Control

OMF Management

OMF Measurement

Researchers

Results

Experiment Description

Federated Testbeds

OMF Resources
Tutorial Program

• MobilityFirst Introduction
• ORBIT Overview
• Tutorial:
  • Exercise 1: Simple MobilityFirst Network Deployment and Test.
  • Exercise 2: Measuring Performance of a MobilityFirst Router
  • Exercise 3: Socket Programming using New MobilityFirst NetAPI
MobilityFirst Tutorial

- All the tutorials are available at:
  - http://geni.orbit-lab.org/wiki/Tutorials/oMF
Exercise 1: Objective

- Setup a basic MobilityFirst network composed of:
  - 2 MF routers
  - 2 clients
  - 1 GNRS

- Generate traffic through a ping-like application
Exercise 1: Design/Setup

• ORBIT
  • Log into grid console using ssh (for simplicity do this in 3 windows, required throughout the exercises)
  • Load the MobilityFirst image on the nodes assigned to you (using your group ID instead of XX):
    • `omf load -i 'mf-release-latest.ndz' -t system:topo:mf-groupXX`
  • If you see the following, you are good to go:

```
INFO exp: ----------------------------
INFO exp: Imaging Process Done
INFO exp: 4 nodes successfully imaged - Topology saved in '/tmp/pxe_slice-2014-10-15t02.10.16.594-04.00-tope-success.rb'
INFO exp: ----------------------------
INFO EXPERIMENT DONE: Event triggered. Starting the associated tasks.
INFO NodeHandler:
INFO NodeHandler: Shutting down experiment, please wait...
INFO NodeHandler:
INFO NodeHandler: Shutdown flag is set - Turning Off the resources
INFO run: Experiment pxe_slice-2014-10-15t02.10.16.594-04.00 finished after 1:50
```
Exercise 1: Design/Setup

• Software and experiment control in the ORBIT testbed automated using the OMF framework, OMF control script written in Ruby
  • Application Definition (path, description, parameters)
    • MF-Router
    • MF-HostStack
    • MF-GNRS
  • Topology/Groups definition (use single statements to set configuration on nodes belonging to the group)
    • Router
    • Host
Exercise 1: Execution

• Turn the assigned nodes on:
  • omf tell –a on –t system:topo:imaged

• Download the exercise script into your grid console:
  • wget www.winlab.rutgers.edu/~bronzino/downloads/orbit/exercise1.rb

• Execute the exercise:
  • Omf exec exercise1.rb

• If you see this line you can test the network as follows:
  
  INFO exp: Bringing up routers...
  INFO exp: Request from Experiment Script: Wait for 5s....
  INFO exp: Bringing up host stacks...
  INFO exp: Access the nodes to run a program
  INFO exp: Request from Experiment Script: Wait for 10000s....
Exercise 1: Test the Network

- In the two other terminals you opened at the beginning, ssh in to the client nodes: ssh root@nodex-y
  - x-y for the server is the one with GUID 102, the client is with GUID 101

In the server’s terminal:
- mfping –s –m 102 -0 101

In the client’s terminal:
- mfping -c -m 101 -o 102 -n 10
Exercise 1: Finish

- Kill the *mfping* server using Ctrl-C on the corresponding node.
- On the grid's console running the experiment script, interrupt the experiment using the Ctrl-C key combination.
Tutorial Program

• MobilityFirst Introduction
• ORBIT Overview
• Tutorial:
  • Exercise 1: Simple MobilityFirst Network Deployment and Test.
  • Exercise 2: Measuring Performance of a MobilityFirst Router
  • Exercise 3: Socket Programming using New MobilityFirst NetAPI
Exercise 2: Design/Setup

• Setup a basic MobilityFirst network composed of:
  • 2 MF routers
  • 2 clients
  • 1 GNRS
Exercise 2: Design/Setup

• Setting up the “OML-Enabled Monitor on Router’s Application”

• Generate traffic between 2 hosts

• Measure key performance metrics like throughput and latency

• Monitor periodically queries the router through a socket control port

• Extract the statistical results using OML-enabled monitor for MobilityFirst routers
Exercise 2: Execution

• Download the exercise script into your grid console:
  • `wget www.winlab.rutgers.edu/~bronzino/downloads/orbit/exercise2.rb`

• Execute the exercise:
  • `omf exec exercise2.rb`

• If you see this line you can test the network as follows (like exercise 1):

```
INFO exp: Bringing up routers...
INFO exp: Request from Experiment Script: Wait for 5s....
INFO exp: Bringing up host stacks...
INFO exp: Access the nodes to run a program
INFO exp: Request from Experiment Script: Wait for 10000s....
```
Exercise 2: Execution

- ssh to node with GUID 102 (ssh root@nodex-y) and type in:
  - mfping -s -m 102 -o 101

- ssh to node with GUID 101 (ssh root@nodex-y) and type in:
  - mfping -c -m 101 -o 102 -n 10

- Now to retrieve the data the routers have reported, in your browser type in:
  - http://oml.orbit-lab.org:5054/result/dumDatabase?expID= <your_exp_ID>
Exercise 2: Finish

- Kill the `mfping` server using Ctrl-C on the corresponding node.

- On the grid's console running the experiment script, interrupt the experiment using the Ctrl-C key combination.
More Info @

mobilityfirst.winlab.rutgers.edu
www.orbit-lab.org
www.geni.net