MobilityFirst
GENI
Tutorial

Initial Setup

- Requirement:
    - Have a GENI Portal Account
- 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~~

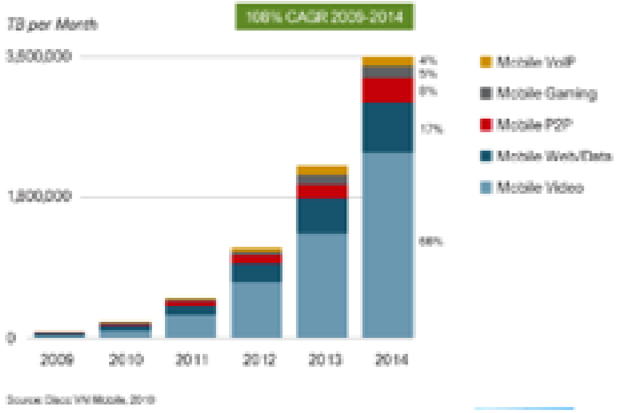
Tutorial Program

- 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~~

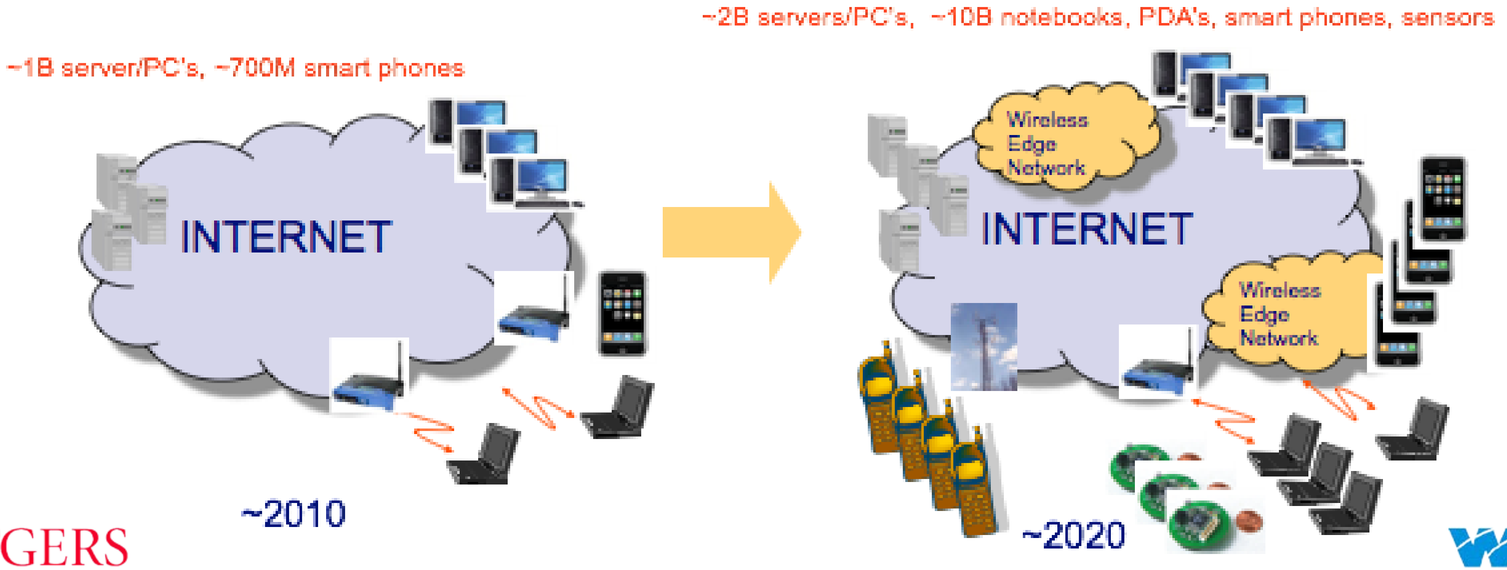
MobilityFirst: Motivations

Historic shift from PC's to mobile computing and embedded devices…

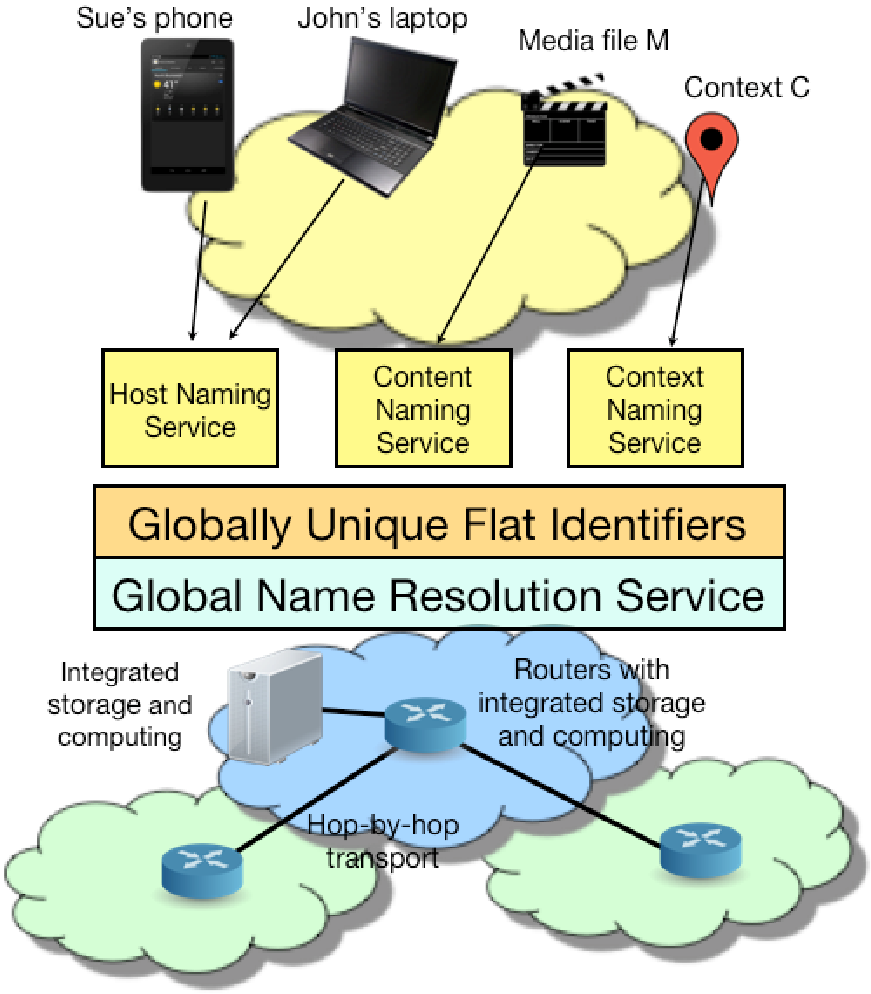
- ~4 B cell phones vs. ~1B PC's in 2010
- Mobile data growing exponentially – Cisco white paper predicts 3.6 Exabytes by 2014, significantly exceeding wired Internet traffic
- Sensor/IoT/V2V just starting, ~5-10B units by 2020



Source: Cisco VNI Mobile, 2010

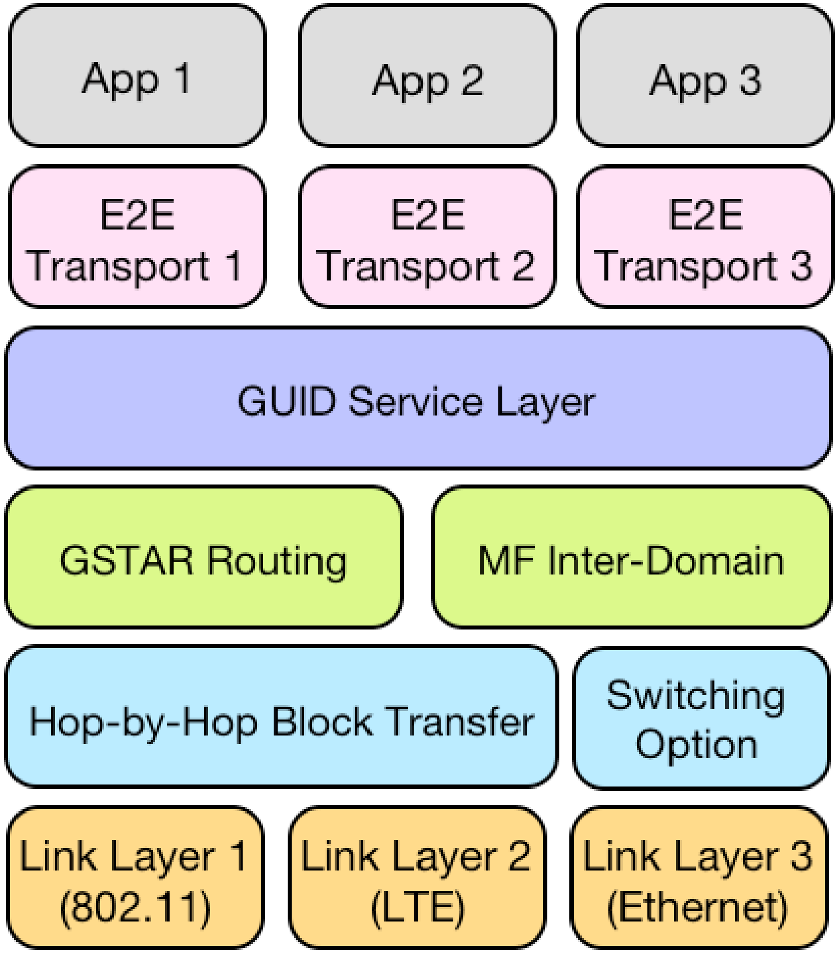

~1B server/PC's, ~700M smart phones
INTERNET
~2010
~2B servers/PC's, ~10B notebooks, PDA's, smart phones, sensors
Wireless Edge Network
INTERNET
Wireless Edge Network
~2020

MobilityFirst: Name-Address Separation


Sue's phone
John's laptop
Media file M
Context C
Host Naming Service
Content Naming Service
Context Naming Service
Globally Unique Flat Identifiers
Global Name Resolution Service
Integrated storage and computing
Routers with integrated storage and computing
Hop-by-hop transport

- 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



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)

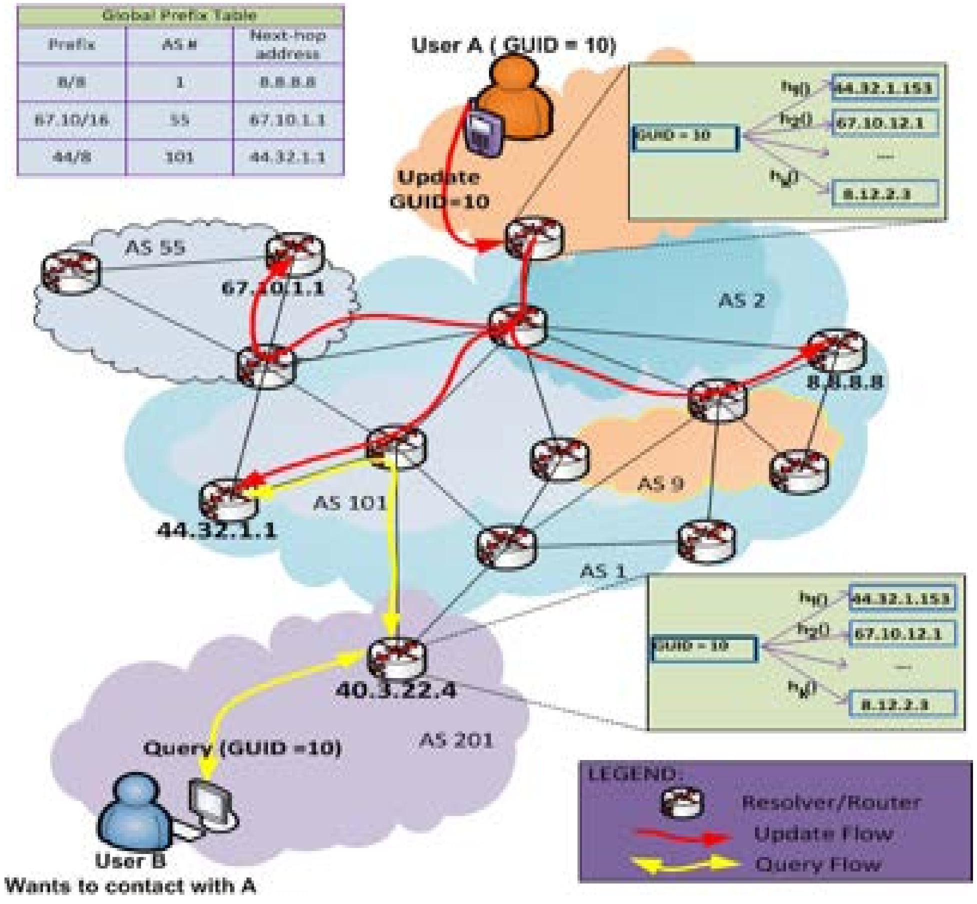
Link Layer 3 (Ethernet)

MobilityFirst Packet

SID	Dst_GUID	Dst_NA	Src_GUID	Src_NA	DATA

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

MobilityFirst: Global Name Resolution Service (GNRS)

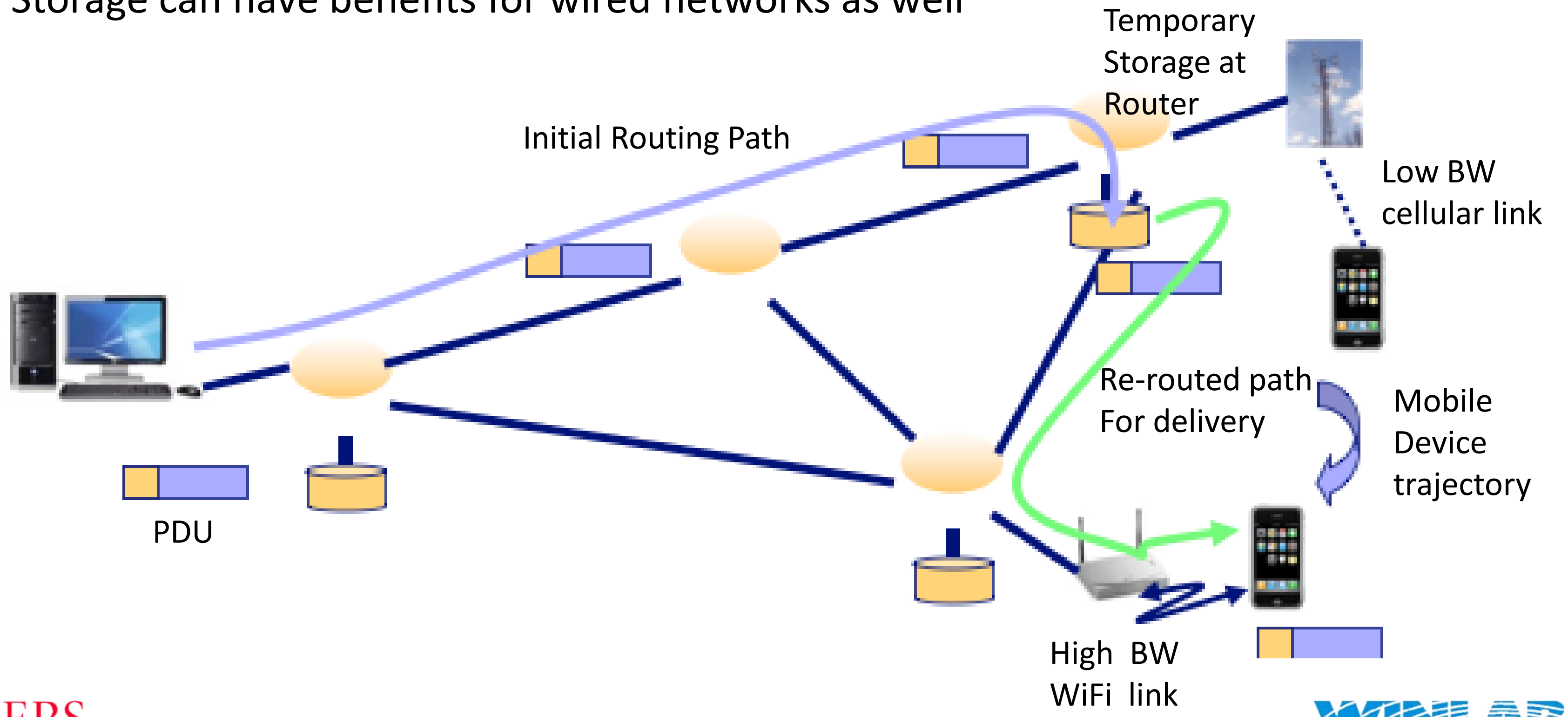


Global Prefix Table		
Prefix	AS #	Next-hop address
8/8	1	8.8.8.8
67.10/16	55	67.10.1.1
44/8	101	44.32.1.1

- 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)

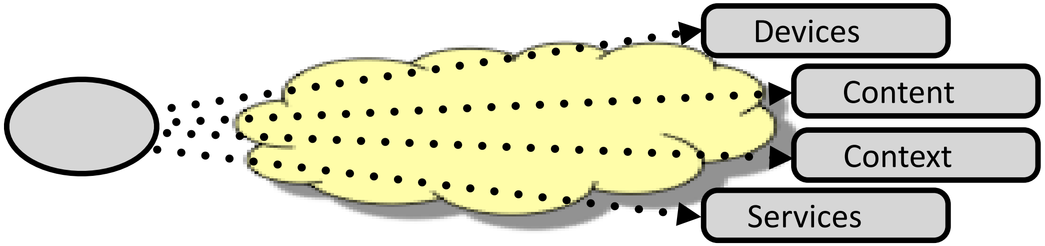
MobilityFirst: Routing (GSTAR)

- Storage aware (CNF, generalized DTN) routing exploits in-network storage to deal with varying link quality and disconnection
- Routing algorithm adapts seamlessly adapts 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

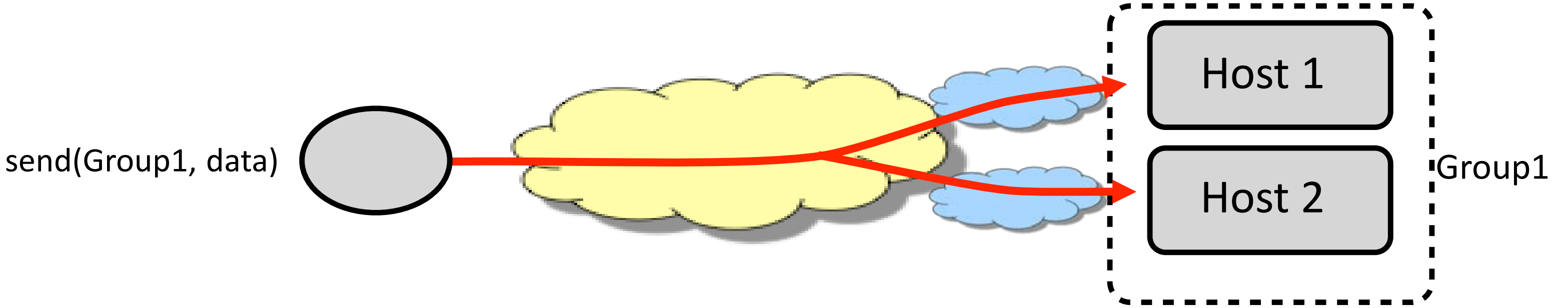


MobilityFirst: Network API

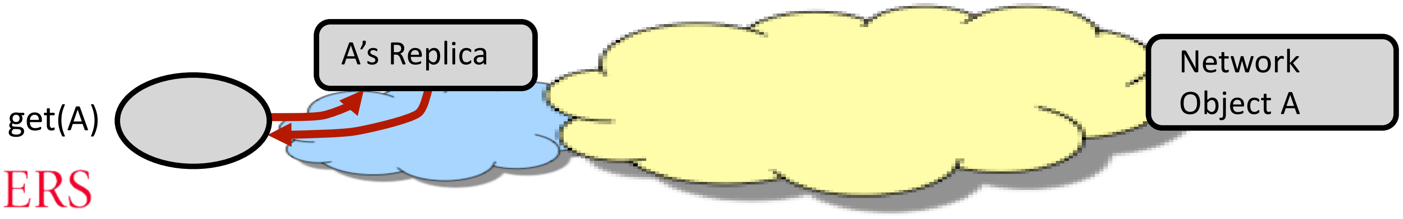
- Service Abstractions
  - Direct Addressability for All Network Principals.



  - Multi-Point Addressability.



  - En-Route Storage and Compute.

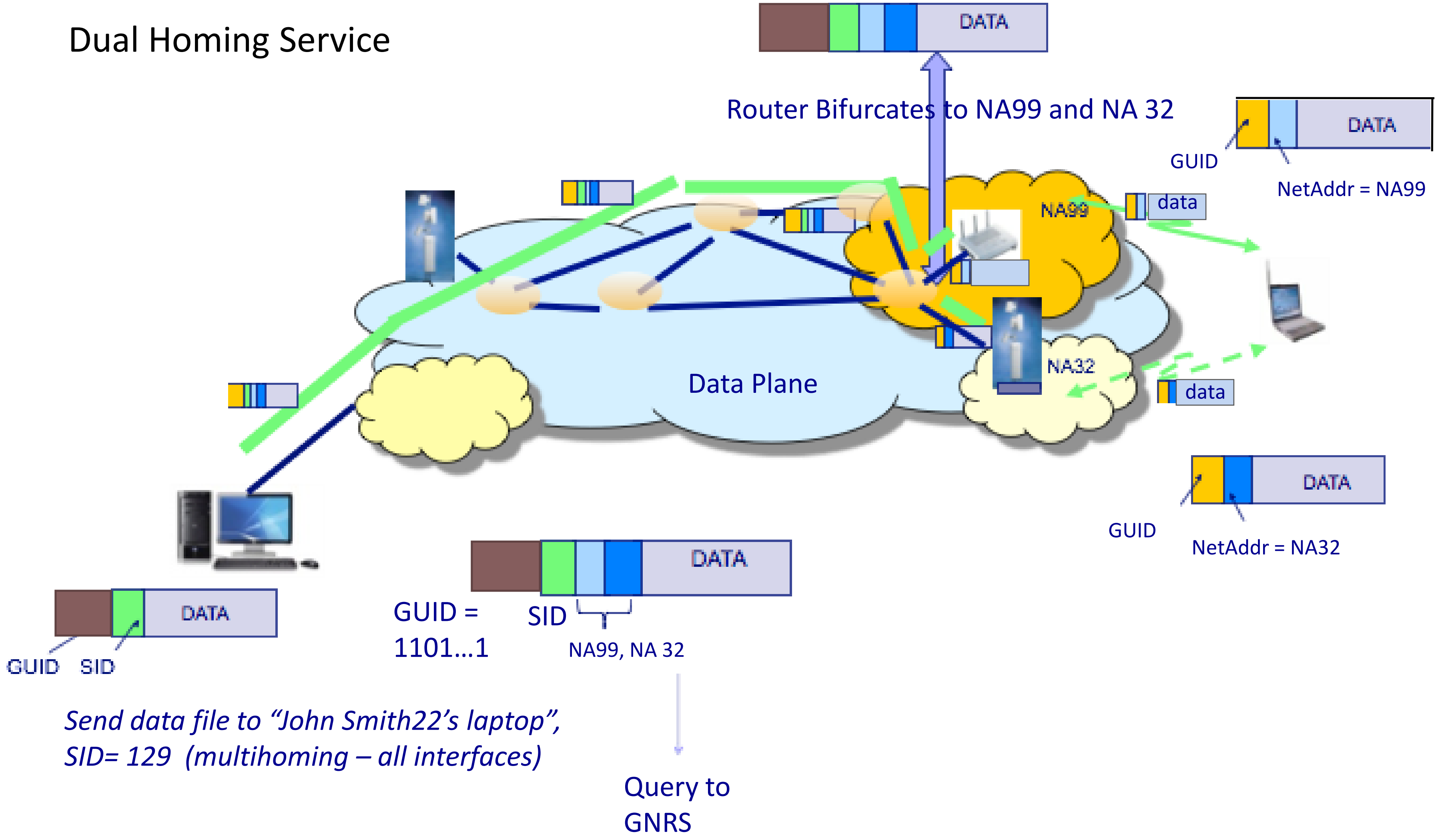


MobilityFirst: Network API

open, close	- *open(profile, [profile-options], [source-GUID])* - Allocate the appropriate resources given the profile of the communication specified by the program.
send, recv	- *send(destination-GUID, data, [service-options])* - *recv(source-GUID, buffer, [GUID-set])* - Name based message exchange. - By use of options ability to request set of specific network services. - Per message destination GUID.
attach, detach	- *attach(GUID-set)* - Management of network presence and reachability.
get, post, exec	- *get(content-GUID, request, buffer, [svc-opts])* - Exploit the additional information on the type of network object represented by the GUID. - 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.

MobilityFirst: Protocol Example 1

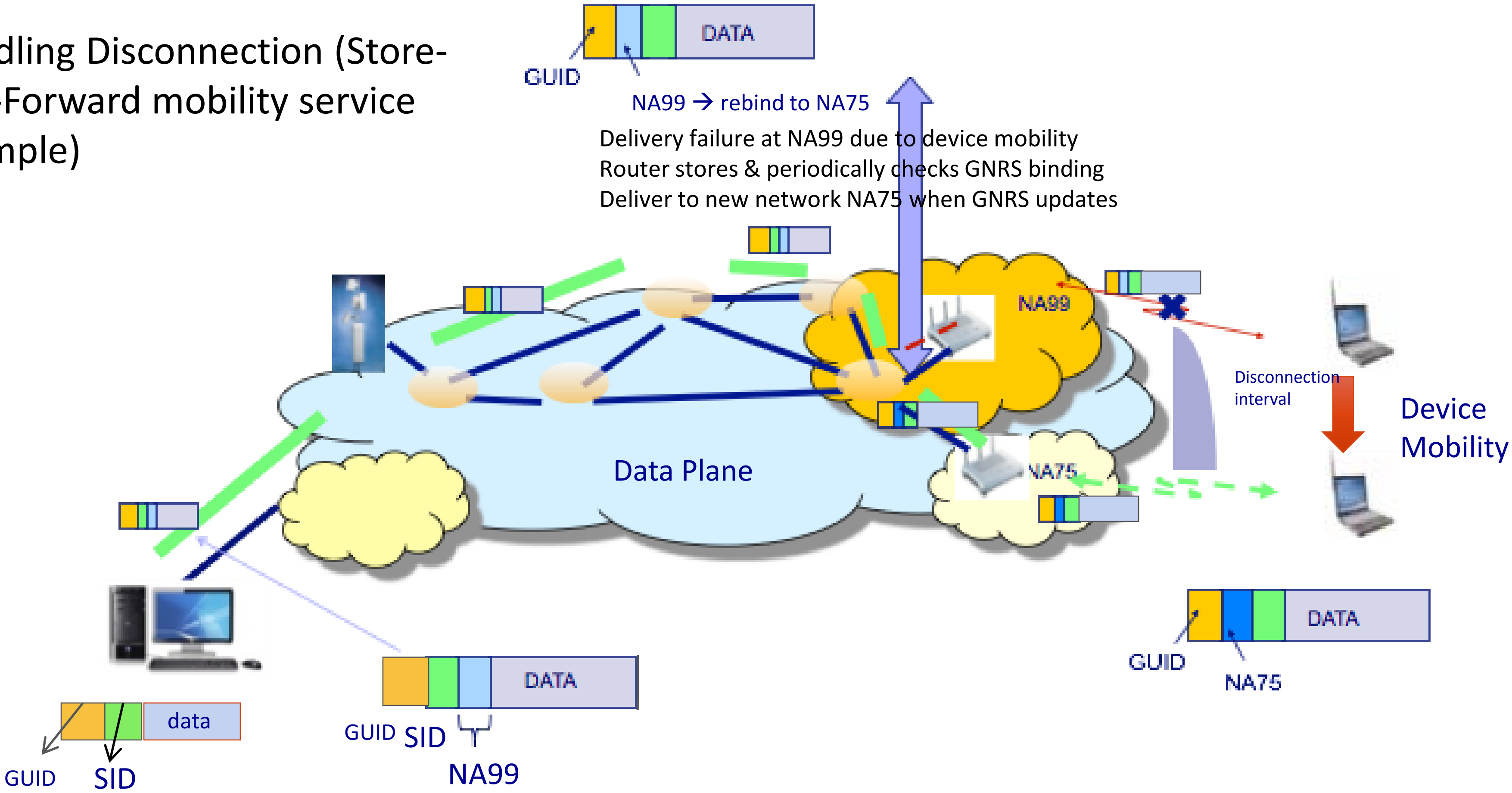
Dual Homing Service



Send data file to "John Smith22's laptop",
SID= 129 (multihoming – all interfaces)

MobilityFirst: Protocol Example 2

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



NA99 → rebind to NA75

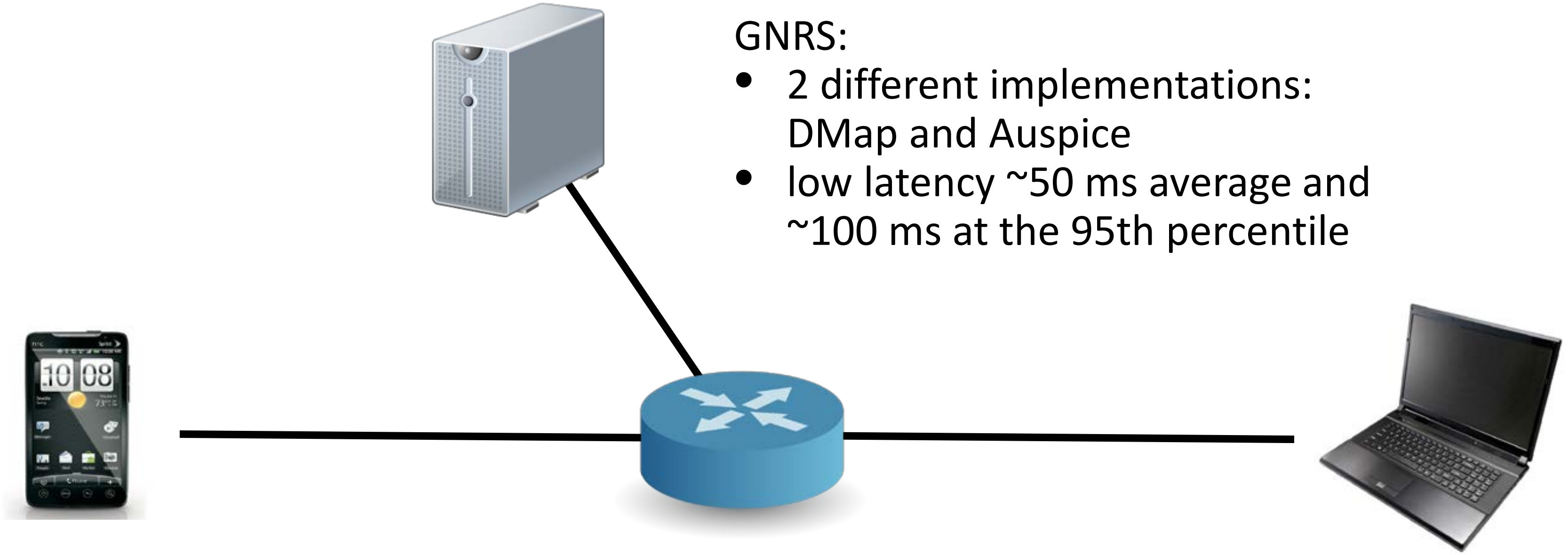
Delivery failure at NA99 due to device mobility
Router stores & periodically checks GNRS binding
Deliver to new network NA75 when GNRS updates

Disconnection interval

Device Mobility

Send data file to "John Smith22's laptop",
SID= 11 (unicast-mobile delivery)

MobilityFirst: Prototype



GNRS:

- 2 different implementations: DMap and Auspice
- low latency ~50 ms average and ~100 ms at the 95th percentile

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”)

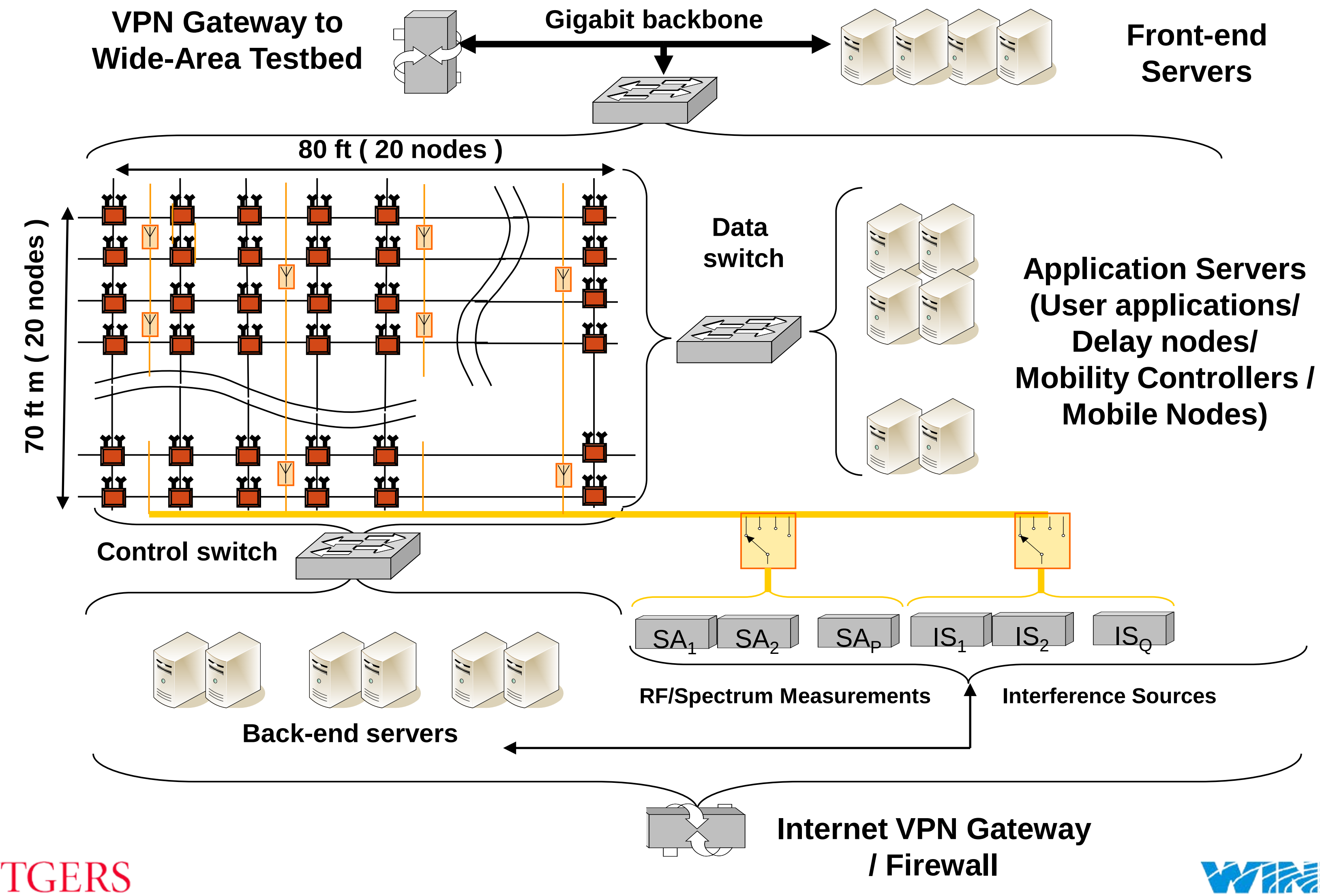
Router:

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

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

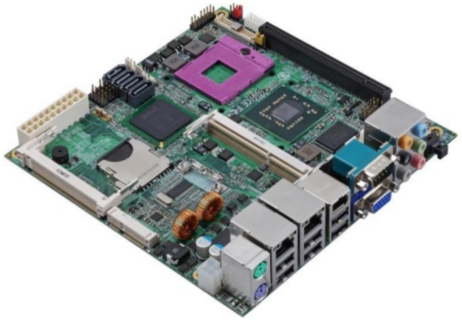


RUTGERS

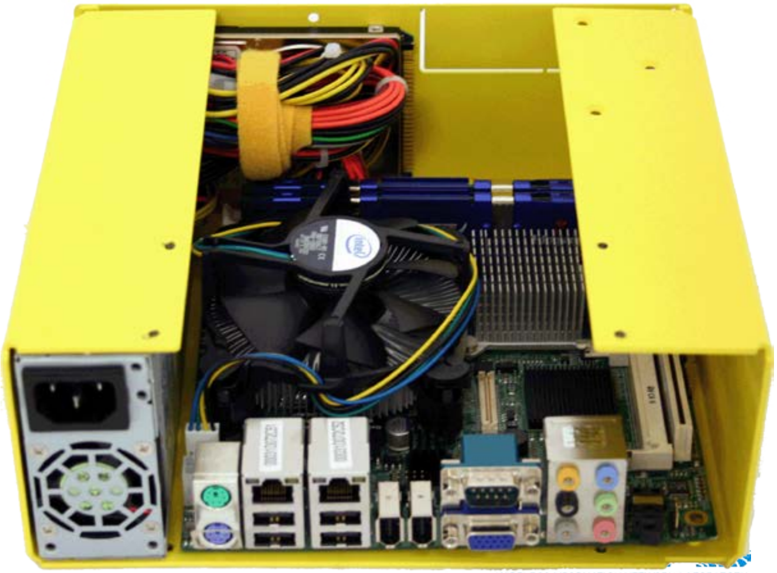
WINLAB
WIRELESS INFORMATION NETWORK LABORATORY

ORBIT Radio Node (Version 3 & 4)

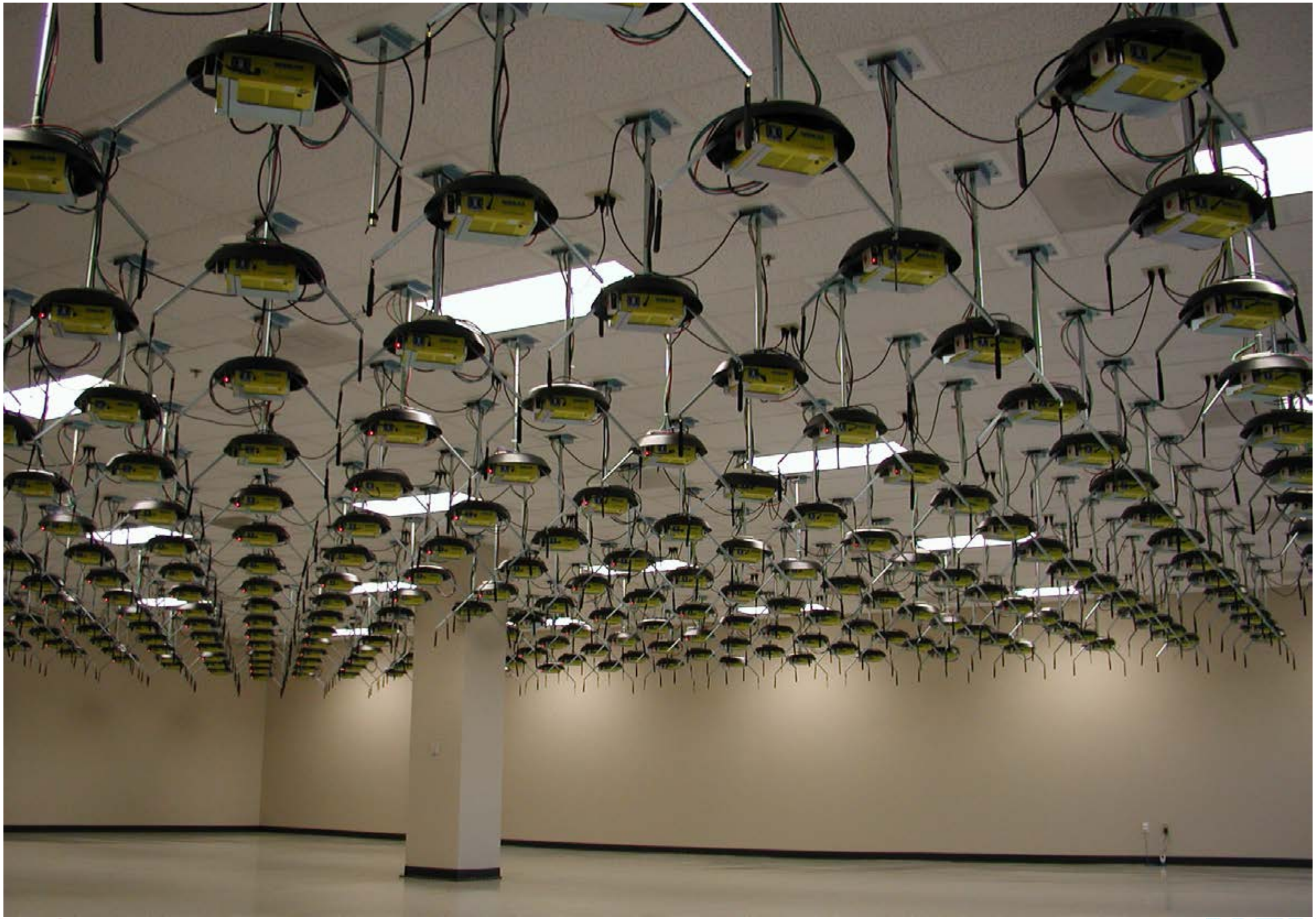
- Core 2 Quad with Q35 Express chipset
- 4 GB DDR2
- 2 x Gigabit Ethernet ports
- PCI-Express X16
- Mini-PCI socket
- 8 x USB 2.0
- 2 x COM



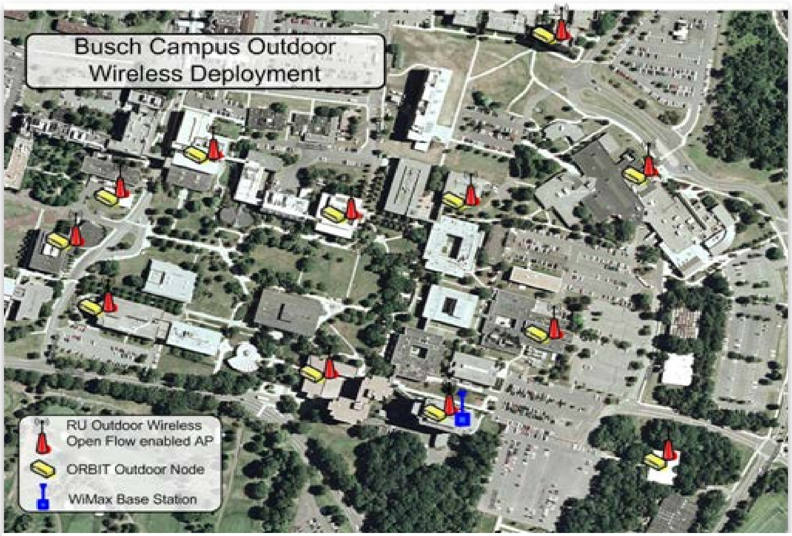
- Core 2 Duo with GM45 chipset
- 8 GB DDR3
- 2 x Gigabit Ethernet ports
- PCI-Express X16
- PCI Express mini socket
- Mini-PCI socket
- 8 x USB 2.0
- 2 x COM



ORBIT Grid

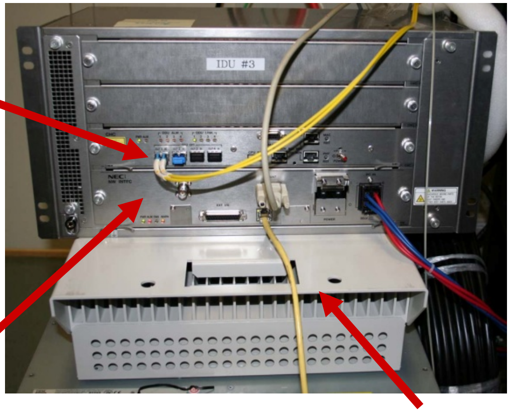


ORBIT Outdoor Infrastructure


Busch Campus Outdoor Wireless Deployment
RU Outdoor Wireless Open Flow enabled AP
ORBIT Outdoor Node
WiMax Base Station

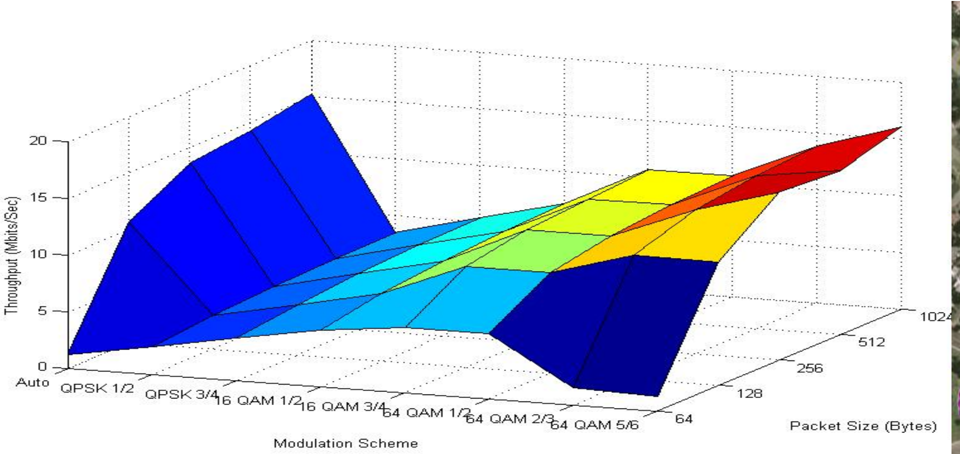
RF Module (sector)

Base Module



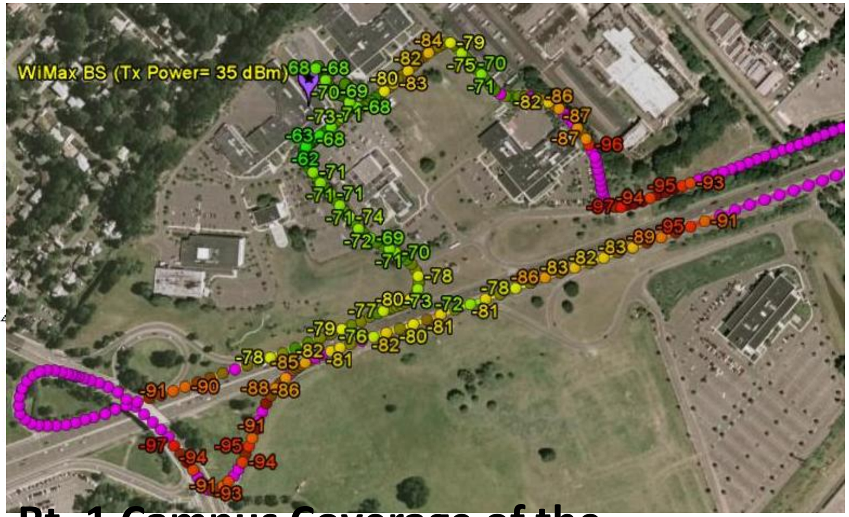
Outdoor Unit (ODU)

Omni-directional antenna
(elev. < 6ft above roof!)



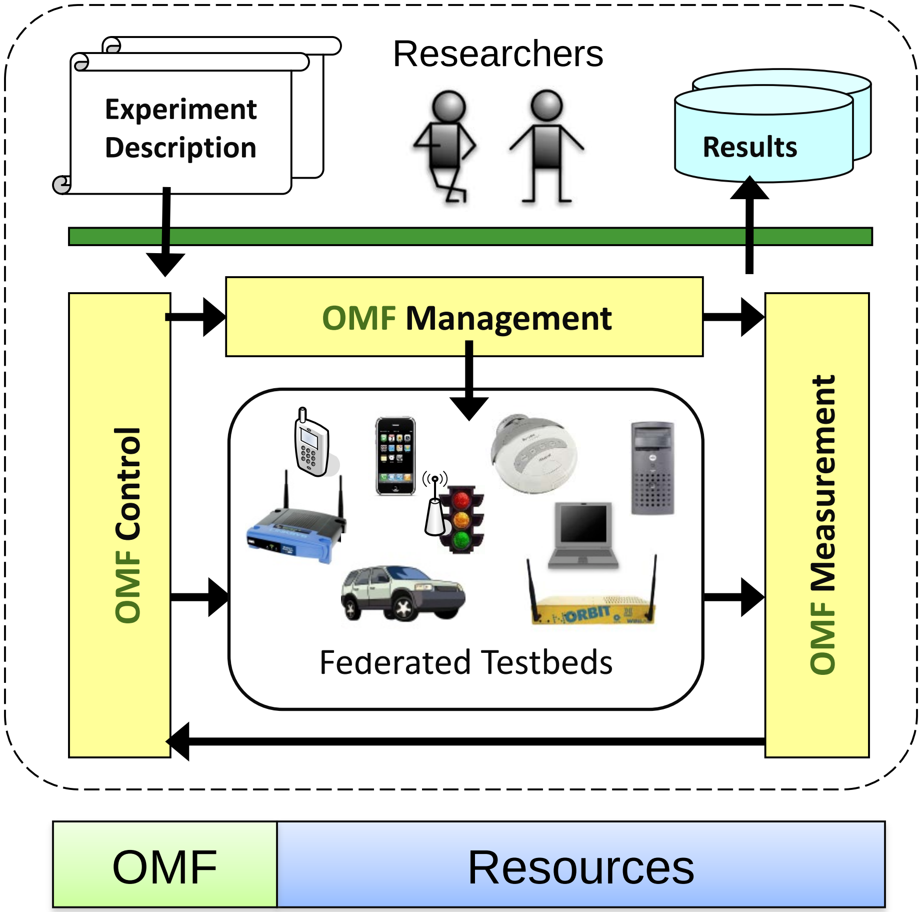
Experimental readings at one location

CINR = 29 RSSI = -51



Rt. 1 Campus Coverage of the WiMAX base station

OMF Overview



Experiment Description

Researchers

Results

OMF Management

OMF Control

OMF Measurement

Federated Testbeds

OMF

Resources

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

Tutorial Program

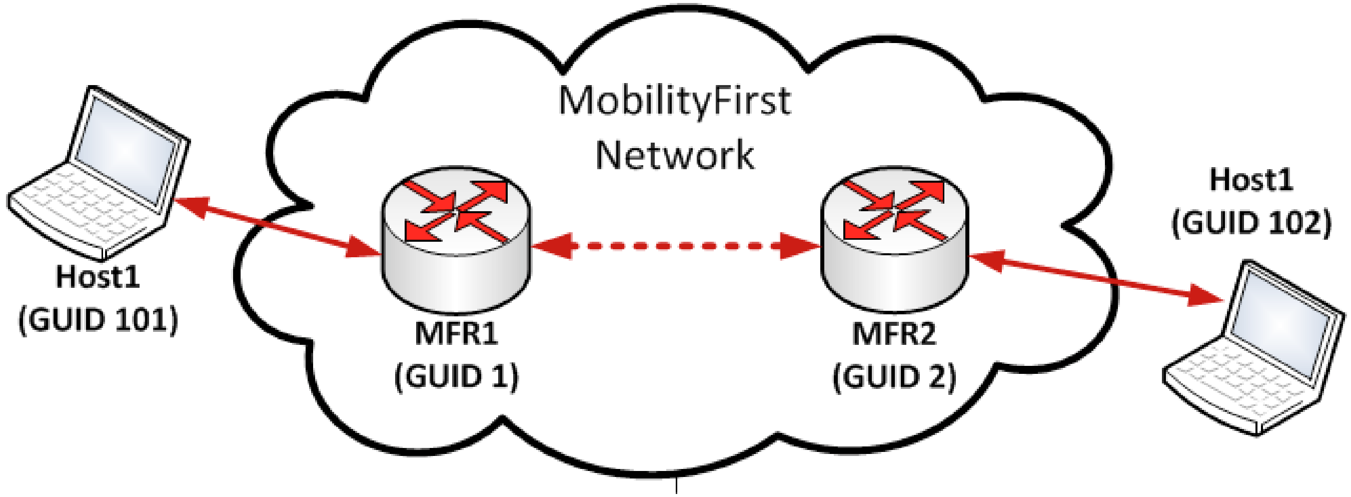
- MobilityFirst Introduction
- ORBIT Overview
- Tutorial:
  - Exercise 1: Simple MobilityFirst Network Deployment and Test.
  - Exercise 2: Measuring Performance of a MobilityFirst Router
  - ~~Exercise 3: Socket Programming using New MobilityFirst NetAPI~~

MobilityFirst Tutorial

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

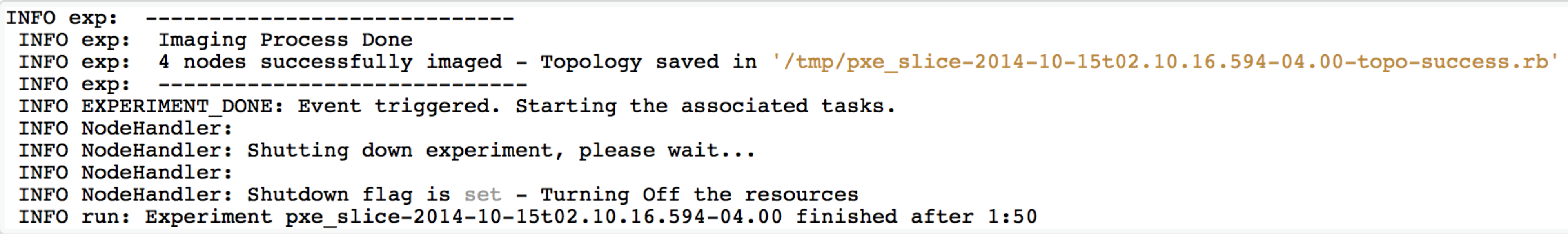
Exercise 1: Objective

- 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-topo-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

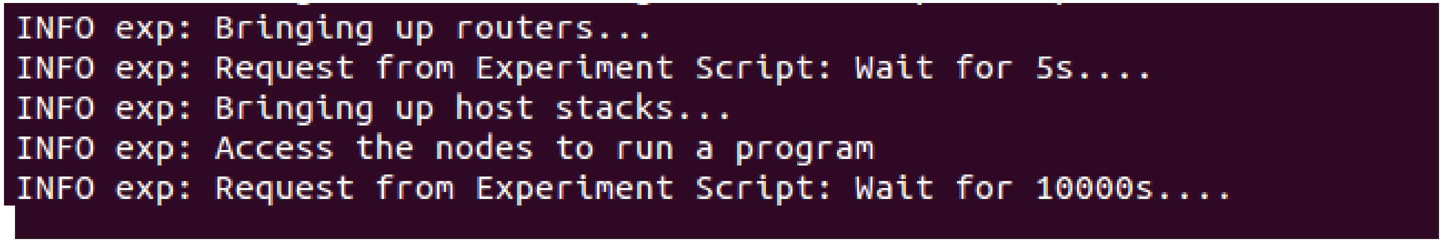
- Software and experiment control in the ORBIT testbed automated using the OMF framework, OMF control script written in Ruby
  - Application Definition (path, description, parameters)
    - MF-Router
    - MF-HostStack
    - MF-GNRS
  - Topology/Groups definition(use single statements to set configuration 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

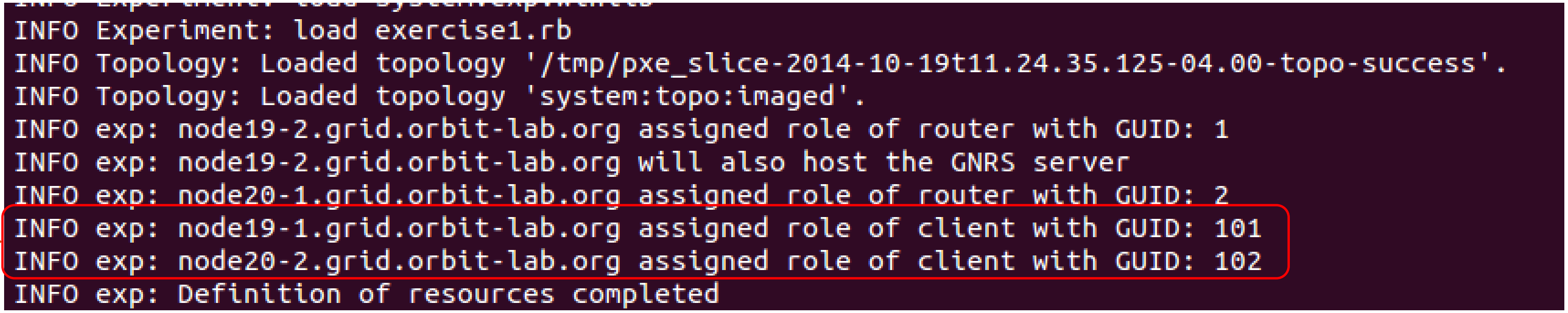
- Execute the exercise:
  - Omf exec exercise1.rb
- If you see this line you can test the network as follows:



```
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

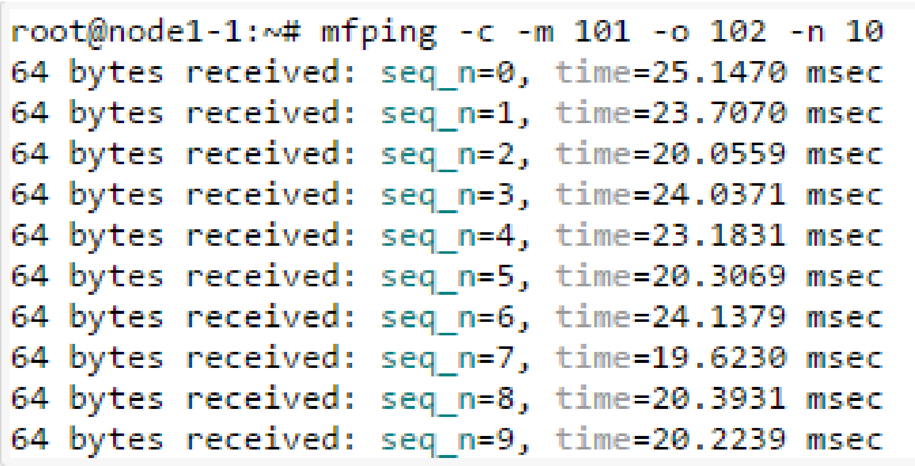
- In the two other terminals you opened at the beginning, ssh in to the client nodes: ssh root@nodex-y
  - x-y for the server is the one with GUID 102, the client is with GUID 101



```
INFO Experiment: load exercise1.rb
INFO Topology: Loaded topology '/tmp/pxe_slice-2014-10-19t11.24.35.125-04.00-topo-success'.
INFO Topology: Loaded topology 'system:topo:imaged'.
INFO exp: node19-2.grid.orbit-lab.org assigned role of router with GUID: 1
INFO exp: node19-2.grid.orbit-lab.org will also host the GNRS server
INFO exp: node20-1.grid.orbit-lab.org assigned role of router with GUID: 2
INFO exp: node19-1.grid.orbit-lab.org assigned role of client with GUID: 101
INFO exp: node20-2.grid.orbit-lab.org assigned role of client with GUID: 102
INFO exp: Definition of resources completed
```

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

- In the client's terminal:
  - mfping -c -m 101 -o 102 -n 10



```
root@node1-1:~# mfping -c -m 101 -o 102 -n 10
64 bytes received: seq_n=0, time=25.1470 msec
64 bytes received: seq_n=1, time=23.7070 msec
64 bytes received: seq_n=2, time=20.0559 msec
64 bytes received: seq_n=3, time=24.0371 msec
64 bytes received: seq_n=4, time=23.1831 msec
64 bytes received: seq_n=5, time=20.3069 msec
64 bytes received: seq_n=6, time=24.1379 msec
64 bytes received: seq_n=7, time=19.6230 msec
64 bytes received: seq_n=8, time=20.3931 msec
64 bytes received: seq_n=9, time=20.2239 msec
```

Exercise 1: Finish

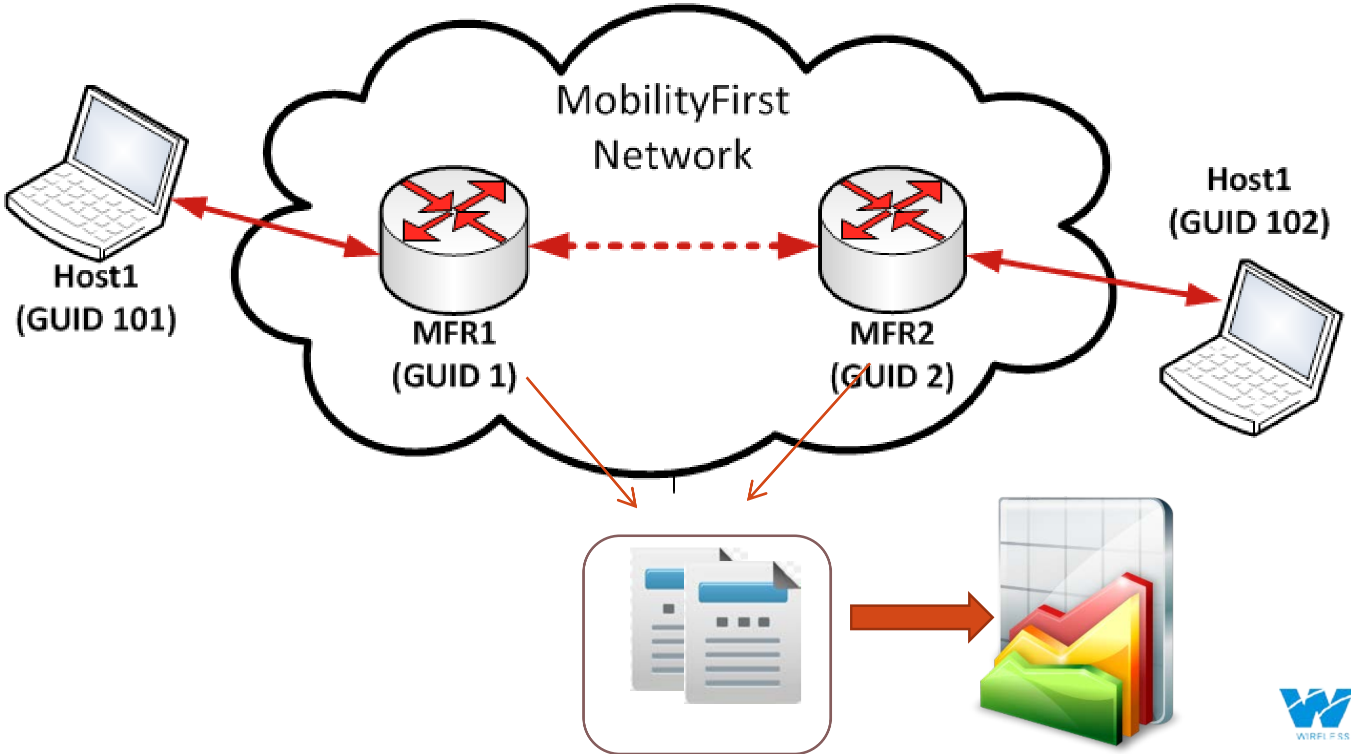
- Kill the *mfping* server using Ctrl-C on the corresponding node.
- On the grid's console running the experiment script, interrupt the experiment using the Ctrl-C key combination.

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



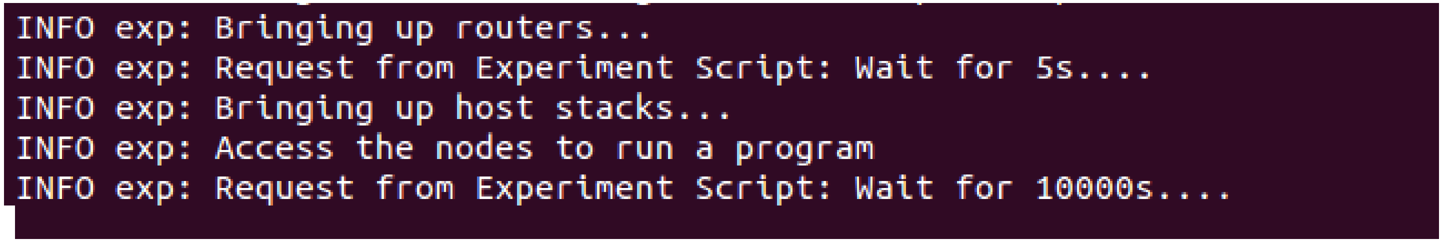
Exercise 2: Design/Setup

- 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

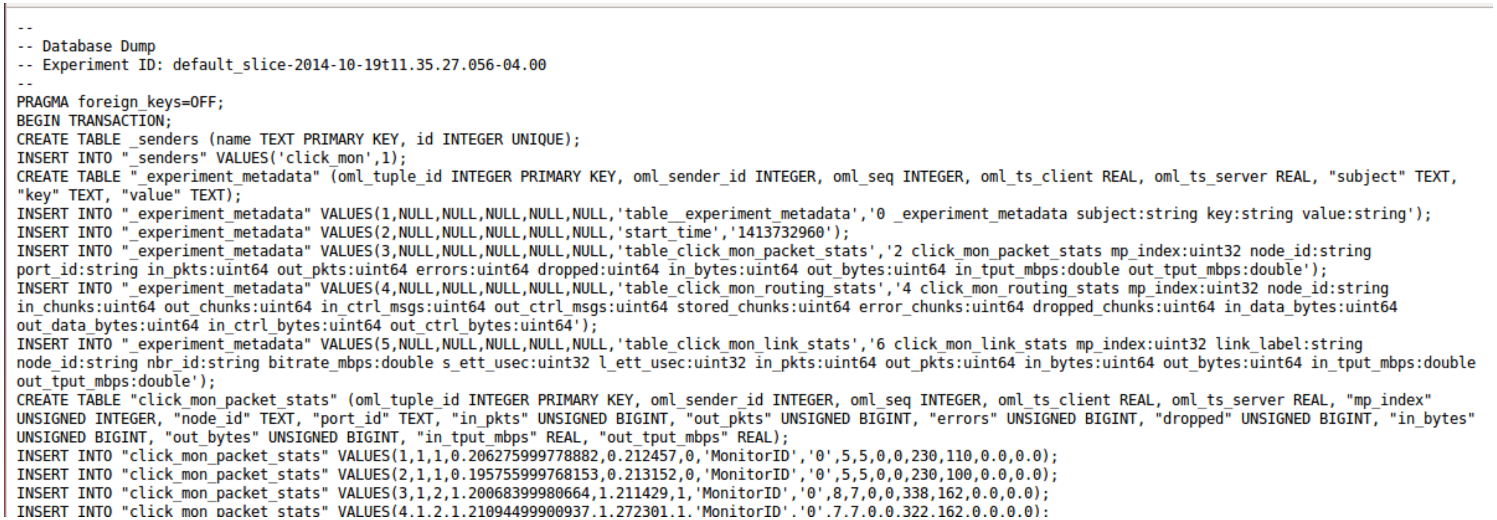
- Execute the exercise:
  - omf exec exercise2.rb
- If you see this line you can test the network as follows (like exercise 1):



```
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 -0 101
- 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>



```
--
-- Database Dump
-- Experiment ID: default_slice-2014-10-19t11.35.27.056-04.00
--
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE _senders (name TEXT PRIMARY KEY, id INTEGER UNIQUE);
INSERT INTO "_senders" VALUES('click_mon',1);
CREATE TABLE "_experiment_metadata" (oml_tuple_id INTEGER PRIMARY KEY, oml_sender_id INTEGER, oml_seq INTEGER, oml_ts_client REAL, oml_ts_server REAL, "subject" TEXT,
"key" TEXT, "value" TEXT);
INSERT INTO "_experiment_metadata" VALUES(1,NULL,NULL,NULL,NULL,NULL,'table__experiment_metadata','0 _experiment_metadata subject:string key:string value:string');
INSERT INTO "_experiment_metadata" VALUES(2,NULL,NULL,NULL,NULL,NULL,'start_time','1413732960');
INSERT INTO "_experiment_metadata" VALUES(3,NULL,NULL,NULL,NULL,NULL,'table_click_mon_packet_stats','2 click_mon_packet_stats mp_index:uint32 node_id:string
port_id:string in_pkts:uint64 out_pkts:uint64 errors:uint64 dropped:uint64 in_bytes:uint64 out_bytes:uint64 in_tput_mbps:double out_tput_mbps:double');
INSERT INTO "_experiment_metadata" VALUES(4,NULL,NULL,NULL,NULL,NULL,'table_click_mon_routing_stats','4 click_mon_routing_stats mp_index:uint32 node_id:string
in_chunks:uint64 out_chunks:uint64 in_ctrl_msgs:uint64 out_ctrl_msgs:uint64 stored_chunks:uint64 error_chunks:uint64 dropped_chunks:uint64 in_data_bytes:uint64
out_data_bytes:uint64 in_ctrl_bytes:uint64 out_ctrl_bytes:uint64');
INSERT INTO "_experiment_metadata" VALUES(5,NULL,NULL,NULL,NULL,NULL,'table_click_mon_link_stats','6 click_mon_link_stats mp_index:uint32 link_label:string
node_id:string nbr_id:string bitrate_mbps:double s_ett_usec:uint32 l_ett_usec:uint32 in_pkts:uint64 out_pkts:uint64 in_bytes:uint64 out_bytes:uint64 in_tput_mbps:double
out_tput_mbps:double');
CREATE TABLE "click_mon_packet_stats" (oml_tuple_id INTEGER PRIMARY KEY, oml_sender_id INTEGER, oml_seq INTEGER, oml_ts_client REAL, oml_ts_server REAL, "mp_index"
UNSIGNED INTEGER, "node_id" TEXT, "port_id" TEXT, "in_pkts" UNSIGNED BIGINT, "out_pkts" UNSIGNED BIGINT, "errors" UNSIGNED BIGINT, "dropped" UNSIGNED BIGINT, "in_bytes"
UNSIGNED BIGINT, "out_bytes" UNSIGNED BIGINT, "in_tput_mbps" REAL, "out_tput_mbps" REAL);
INSERT INTO "click_mon_packet_stats" VALUES(1,1,1,0.206275999778882,0.212457,0,'MonitorID','0',5,5,0,0,230,110,0.0,0.0);
INSERT INTO "click_mon_packet_stats" VALUES(2,1,1,0.195755999768153,0.213152,0,'MonitorID','0',5,5,0,0,230,100,0.0,0.0);
INSERT INTO "click_mon_packet_stats" VALUES(3,1,2,1.20068399980664,1.211429,1,'MonitorID','0',8,7,0,0,338,162,0.0,0.0);
INSERT INTO "click_mon_packet_stats" VALUES(4,1,2,1.21094499900937,1.272301,1,'MonitorID','0',7,7,0,0,322,162,0.0,0.0);
```

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