Architecture and Framework for Supporting Open-Access Multi-user Wireless Experimentation

S. Ganu, I. Seskar, Max Ott, D. Raychaudhuri, S. Paul

http://www.orbit-lab.org
Project Rationale

• **Current wireless research**
  - Primarily simulation based or small in-house experimental setups
  - Difficult to repeat experiments
  - Excessive setup and data collection times may hinder rapid prototyping and experimentation

• **Key design goals**
  - Support multi-user wireless experimentation
  - Remotely accessible, lights-out operation
  - Facilitate choreographing of experiments
  - Automate measurement collection
  - Capture experiment description so as to repeat as often as necessary
ORBIT Testbed: Background

- Seeded by NSF grant under the Networking Research Testbeds (NRT) program

- Collaborative effort: Rutgers, Columbia, and Princeton, along with industrial partners Lucent Bell Labs, IBM Research and Thomson

- Developed and operated by WINLAB, Rutgers University
ORBIT: Indoor Grid

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

Gigabit backbone

Application Servers

RF/Spectrum Measurements

Interference Sources

Internet VPN Gateway / Firewall
Key Requirements

- **scalability**, in terms of the total number of wireless nodes (~100’s).
- **reproducibility** of experiments which can be repeated with similar environments to get similar results.
- **open-access flexibility** giving the experimenter a high-level of control over protocols and software used on the radio nodes
- **extensive measurements capability** at radio PHY, MAC and network levels, with the ability to correlate data across layers in both time and space
- **remote access** testbed capable of unmanned operation and the ability to robustly deal with software and hardware failures
Key Software Considerations

- Unlike wired testbeds, difficult to isolate experiments – mainly serial mode of operation
- Need to quickly offload users at the end of the slot
- Reduce start up and clean up times
Software components

**Experiment Controller**
- Choreograph experiments
- Capture experiment details to facilitate repetition

**Measurement Framework**
- Efficient measurement collection at run-time
- Avoids delays at end of experiment to collate measurements

**Libmac**
- Provides driver independent hooks to the application developers to collect measurements from at radio PHY, MAC layers
Experiment Controller (NodeHandler)

Support Services

NodeHandler

Experiment Script

Repository

Applications

Multicast channel

NodeAgent(s)

Interface initializations and configuration e.g Intel, Atheros, Cisco
Experiment Controller

• A central NodeHandler process communicates with NodeAgents (present on each active node in the experiment)

• Instructs nodes to configure interfaces, launch applications etc.

• Communication
  – Over multicast – scalable
  – Using experiment scripts
OML: Orbit Measurement Library

• Experiments are about collecting measurements
• How to collect them efficiently in a distributed environment like ORBIT?
OML: Orbit Measurement Library

- **Client**
  - Simple API for application writers
  - Filters reduce the amount of reportable data
  - XDR encoded data over multicast channel

- **Collection Server**
  - Berkeley DB used for scalability
  - SQL database for persistent data archiving
  - One multicast channel per experiment for logical segregation of data, and scalability
OML: Pluggable Filters

- Not all measurements may be needed
- Allow dynamic preprocessing before reporting to database
- Experimenter can choose the granularity (per packet or every N packets, per second or every N seconds)
Real time Statistics

```sql
mysql> select * from metrics;
+-------------------------------+-----------------+------------+---------+
| id  | metric | units | type    |
+-------------------------------+-----------------+------------+---------+
| 1   | rssi   | db    | float   |
| 2   | noise  | db    | float   |
| 3   | throughput | bps | float   |
| 4   | offered_load | bps | float   |
+-------------------------------+-----------------+------------+---------+
4 rows in set (0.00 sec)
```

```sql
mysql> select * from metrics_values where id=3 and node_id="orbit4";
+-------------+-------+--------------+--------+------------------------+------------------------+
| id | value | packet_size | node_id | sequence_no | timestamp              |
+-------------+-------+--------------+--------+------------------------+------------------------+
| 3 | 1086464 | 256 | orbit4 | 4 | 1084223553 |
| 3 | 1515520 | 256 | orbit4 | 10 | 1084223554 |
| 3 | 1503232 | 256 | orbit4 | 16 | 1084223555 |
+-------------+-------+--------------+--------+------------------------+------------------------+
```

MATLAB, Excel for Mysql allows easy post processing
Libmac

User-space C library
- To inject and capture MAC layer frames.
- To manipulate wireless interface parameters at both aggregate and per-frame levels.
- To communicate wireless interface parameters over the air, on a per-frame level.
- Allows application developers to interface with driver measurements through simple function call
Mobility Emulation: Our Approach

- Uses software spatial switching
- Emulates trajectory by switching to different radio and antenna positions as time progresses
Chassis Manager Controller

- Web/Program interface for remote control of nodes
- Provides facilities for power on, reboot, hard/soft power off
- Console access to node
- Logging of node state (on/off), temperature, and voltage

Main Grid Node Status

(CM IP: 10.1.x.y)
Frisbee*

- Fast and automated way to image any number of nodes

- Frisbee – Client/Server application that facilitates fast transfers of entire disk images.
  
  - Baseline Node Image (300 MB) currently takes ~5 minutes to install on all 64 grid nodes

* From Emulab Testbed, University of Utah
Putting it all together

**USER / CONTROLLER**

Experiment Script

- **Node configuration**
  - Select nodes
  - Configure interfaces
- **Application configuration**
  - Download application and libraries
  - Configure application parameters
- **OML configuration**
  - Configure measurement collection parameters

(Change channel, power, sleep on/off etc during experiment)

**GRID**

- **NODE**
  - Experiment details
  - Node configuration
  - Application configuration
  - OML configuration

- **OML Client (per node)**
  - Static
  - Dynamic

- **OML Server**
  - Off-line Storage of results
  - Run time statistic collection

- **DB**
  - Display
  - Fetch results

**SERVICES**

**OBSERVER**
Sample Experiments using ORBIT

- Sender-Receiver Experiment
- Node 4-3 sends to Node 5-4
  - 11 Mbps
  - ‘b’
  - Ad-hoc (or Master-Managed)
  - 3 Mbps offered load
  - Measure RSSI, Throughput at receiver and offered load at sender
Define Sender

```ruby
# Define nodes used in experiment
#
nodes([4,3], 'sender') { |node|
    node.image = nil  # Use default disk image

    # experiment property space
    node.prototype("test:proto:sender", {   # use prototype "sender"
        'destinationHost' => '192.168.5.4',       # Set it's property "destinationHost"
        'packetSize' => Experiment.property("packetSize"),
        'rate' => Experiment.property("rate")       # bind the remaining properties to defaults
    })                                                # Can be overridden later

    node.net.w0.mode = "master"
    node.net.w0.type = 'b'
    node.net.w0.essid = "helloworld"               # Set wireless parameters
    node.net.w0.ip = "%192.168.%x.%y"
    node.net.w0.rate = "11m"
}
```

w0, w1 are interpreted by nodeAgent according to the card being used

- e.g Intel  w0= eth2, w1= eth3
- Atheros w0=ath0, w1= ath1
Define Receiver

# Define nodes used in experiment
#
nodes([5,4], 'receiver') {|node|
  node.image = nil                    # assume the right image to be on disk
  node.prototype("test:proto:receiver", {
    'hostname' => '192.168.5.4',
    'protocol' => 'udp_libmac'        # Use Libmac to report RSSI
  })
  node.net.w0.mode = "managed"
  node.net.w0.type = 'b'
  node.net.w0.essid = "helloworld"
  node.net.w0.ip = "%192.168.%x.%y"
}
# Now, start the application
whenAllInstalled() {
  allNodes.startApplications

  # Set packet size to 1024 bytes
  # and packet rate to 3000 Kbps
  NodeSet['sender'].send(:STDIN, 'proc/otg', 'size 1024')
  NodeSet['sender'].send(:STDIN, 'proc/otg', 'rate 3000')

  # Run the experiment for 60 seconds
  wait 60

  # Stop the applications
  allNodes.stopApplications
  Experiment.done
}

View Results (during exp.)

Cross layer effects! Throughput, RSSI vs time
Welcome

ORBIT is a two-tier laboratory emulator/field trial network testbed designed to achieve reproducibility of experimentation, while also supporting evaluation of protocols and applications in real-world settings.

The laboratory-based wireless network emulator uses a novel approach involving a large two-dimensional grid of 802.11x radio nodes which can be dynamically interconnected into specified topologies with reproducible wireless channel models.

Once the basic protocol or application concepts have been validated on the lab emulator platform, users can migrate their experiments to the field trial network which provides a configurable mix of both high-speed cellular (3G) and 802.11x wireless access in a real-world setting.

Orbit is seeded by a $5.45M/yr grant from the NSF under the Networking Research Testbeds (NRT) program. The project is a collaborative effort between several university research groups in the NY/NJ region: Rutgers, Columbia, and Princeton, along with industrial partners Lucent Bell Labs, IBM Research.