

Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols

D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh
WINLAB, Rutgers University, 73 Brett Rd, Piscataway, NJ 08854, USA

Email: {ray, seskar, max, sachin, kishore, harisk, rjs, hliu, singh}@winlab.rutgers.edu

Abstract—This paper presents an overview of the ORBIT (*Open Access Research Testbed for Next-Generation Wireless Networks*) radio grid testbed¹, that is currently being developed for scalable and reproducible evaluation of next-generation wireless network protocols. The ORBIT testbed consists of an indoor radio grid emulator for controlled experimentation and an outdoor field trial network for end-user evaluations in real-world settings. The radio grid system architecture is described in further detail including an identification of key hardware and software components. Software design considerations are discussed for the open-access radio node, and for the system-level controller that handles management and control. The process of specifying and running experiments on the ORBIT testbed is explained using simple examples. Experimental scripts and sample results are also provided.

Keywords: *Wireless network testbed, radio grid, network emulation*

I. INTRODUCTION

It is recognized that powerful technology and market trends towards portable computing and communication devices imply an increasingly important role for wireless access in the next-generation Internet. At the same time, new sensor and pervasive computing applications may be expected to drive large-scale deployments of embedded computing devices interconnected via new types of short-range wireless networks. Although there is a great deal of research activity on future wireless/sensor networks and applications, it is observed that much of this work relies on a formal separation between the radio and networking layers due to the absence of easily available tools for modeling, emulation or rapid prototyping of a complete wireless network. As a result, research on wireless network protocols and applications tends to be conducted predominantly using simulations with simplified radio system models that do not capture real physical layer effects.

A recent technical report [1] states “Since it is difficult to conduct experiments with real mobile computers and wireless networks, nearly all published MANET articles are buttressed with simulation results, and the simulations are based on common simplifying assumptions”. Most of the simplifying assumptions made in the simulations, compounded by limited real-world physical layer modeling capabilities of existing simulators often affect the quality of the results [2] and also their reproducibility. Thus, there is an increasing need in the research community to be able to perform controlled experimental investigations of protocols and evaluations of system design using real-world wireless devices.

In the recent NSF-sponsored Network Testbeds Workshop Report [3], it was concluded that “open wireless multi-user experimental facility (MXF) testbed” for wireless networking would be increasingly important to the research community in view of the limitations of available simulation methodologies and the growing importance of “cross-layer” protocol research. The speed of innovation and productivity of researchers in the wireless networking field can be significantly improved with the development of a flexible, open-access wireless network testbed that is available to experimental researchers across the networking community. Such testbeds can also help accelerate consensus on adoption of standards via reproducible experimentation and access to open-source protocols. These considerations motivated the ORBIT testbed project which aims to provide a flexible, open-access multi-user experimental facility to support research on next-generation wireless networks.

The remaining sections are organized as follows: Section II explains the design requirements for a wireless testbed and the system architecture of ORBIT based on these requirements. We describe the various hardware and software components in Section III. Finally, we explain the life-cycle of an experiment on ORBIT in Section IV and present some initial experimental results using the ORBIT infrastructure in Section V.

II. ORBIT TESTBED DESIGN METHODOLOGY AND SYSTEM ARCHITECTURE

The development of a general-purpose open-access wireless multi-user experimental facility poses significant technical challenges that do not arise in wired network testbeds such as Emulab [4] or ABone [5]. In particular, it is far more difficult to set up a reproducible wireless networking experiment due to random time variations in mobile user location and associated wireless channel models. In addition, wireless systems tend to exhibit complex interactions between the physical, medium access control and network layers, so that strict layering approaches often used to simplify wired network prototypes cannot be applied here. Some of the basic characteristics of radio channels that need to be incorporated into a viable wireless network testbed are as follows

- Radio channel properties depend on specific wireless node locations and surroundings.
- Physical layer bit-rates and error-rates are time-varying.
- Shared medium layer-2 protocols on the radio link have a strong impact on network performance.

¹ Research supported by NSF ORBIT Testbed Project, NSF NRT Grant #ANI0335244

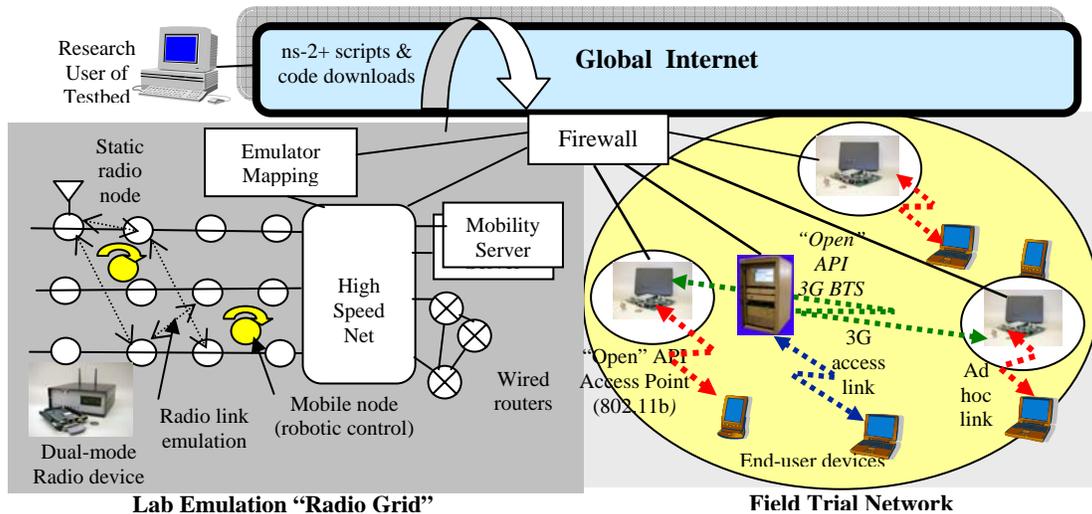


Figure 1 High-level View of Proposed 2-Tier System Architecture for ORBIT

- There are complex interactions between different layers of the wireless protocol stack and currently their mutual interaction cannot be studied easily.
- User's exhibit random mobility and location also plays a role.

A flexible wireless network testbed must be able to support experimental research on a broad range of wireless networking issues and application concepts with various network topologies and network layer protocol options. For the testbed to be useful, it should be scalable and cover a sufficiently broad range of wireless network research problems that might be anticipated over the next 5-10 years.

Some examples of systems or protocol designs that help to understand the overall design space under consideration are:

- Large-scale wireless networks based on 802.11a/b/g radio access along with new protocols for discovery, routing, security etc.
- Mobile ad hoc networks (MANET), typically based on 802.11x WLAN radios, extended to support multi-hop ad hoc routing protocols such as AODV [6] and DSR [7].
- Wireless sensor networks and pervasive computing applications involving embedded radio devices to create a "smart" environment.
- Mobile applications such as location-based services, VoIP over MANET etc.

The system architecture of the ORBIT testbed is based on the general requirements discussed above. The key design goals adopted for this testbed are summarized as follows:

- **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.

As shown in Fig. 1, the ORBIT testbed uses a two-tier architecture with a lab emulator/field trial network architecture to deal with the important issue of reproducibility in experimentation, while at the same time supporting the ability to evaluate protocol and application performance in real-world settings. In particular, the laboratory-based wireless network emulator is constructed using an innovative approach involving a large 2-dimensional array of static 802.11x radio nodes, which can be dynamically interconnected into specified topologies for wireless network experiments with reproducibility for quantitative evaluation of various new protocols, or application and system concepts. Once the basic protocol or application concepts have been validated on the lab emulator platform, users can migrate their software to a wireless field trial network that will provide a reconfigurable mix of cellular (3G) and 802.11x wireless access in a real-world setting (spanning a region about 5 km wide and 2 km long, including university campuses, suburbs and downtown areas). The first phase of this project involves setting up the indoor radio grid emulator, which will be the focus of the remainder of this paper.

The radio grid emulator (as shown in Fig. 2) currently consists of 64 wireless nodes having 802.11a/b/g wireless cards laid out in a 8x8 grid with ~1m spacing between nodes. Each node is connected via multiple high-speed Ethernet links for transfer of applications, control and management information. The system will be extended to 20x20 radio nodes in the next phase of work. Users can have full access to the radio nodes used in their experiments, download and run their own OS image and software packages, control and reboot the nodes, as well as access node console and console logs.

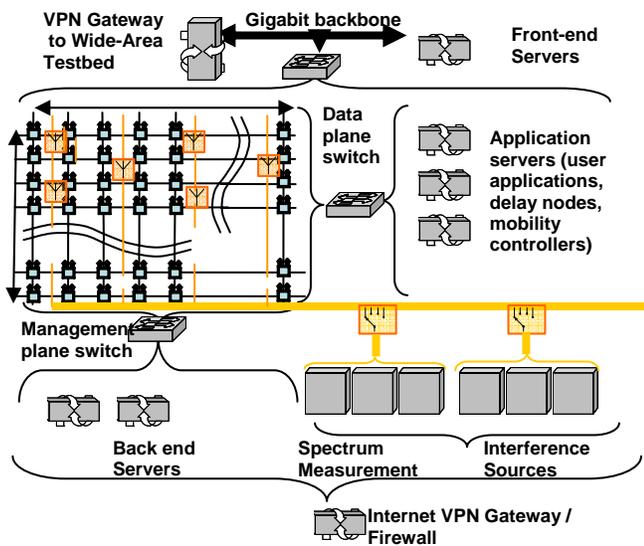


Figure 2 Orbit System Architecture

For example, experimenters can install their own network layer protocols or new application software to construct a specific networking or application scenario for study.

Power and interference levels corresponding to the selected radio system scenario are emulated through a “mapping” algorithm. Experimental data collection tools are also provided to support research evaluation, including network traffic and performance as well as radio link quality and spectrum usage aspects.

III. ORBIT TESTBED: HARDWARE AND SOFTWARE COMPONENTS

The ORBIT testbed includes the following major hardware and software components.

A. Hardware Components

1) **ORBIT radio nodes:** The radio nodes, as shown in Fig. 3, constitute the grid and serve as the primary platform for user experiments.



Figure 3 ORBIT: Radio Node

The radio node is a custom wireless node which consists of:

- 1-GHz VIA C3 processor with 512 MB of RAM and a 20 GB local hard disk
- two wireless mini-PCI 802.11a/b/g interfaces
- two 100BaseT Ethernet ports for experimental data and control respectively

- **integrated chassis manager**, that is used to remotely monitor the status of each radio node’s hardware. The nodes can be reset, powered on/off remotely by the CM through a third Ethernet interface

2) **Instrumentation subsystem** intended to provide capabilities for measurement of radio signal levels and to create various types of artificial RF interference (white noise, colored noise, microwave oven like noise etc.) inside the grid. The interference generator is based on RF Vector Signal Generator while the spectrum measurements are done using Vector Signal Analyzers.

3) **Independent WLAN monitor system** which provides a MAC/network layer view of the radio grid’s components using a number of WLAN “observers” spread across the system.

4) **Support servers** which includes the front-end servers for web services and backend servers for experimentation and data storage. The database servers support multi-terabyte storage capacity.

B. Software Components

Software packages and libraries have been developed to support both application/protocol evaluations. These include common libraries for traffic generation, measurement collection etc. and also provide easy hooks to enable “expert” users to develop their own applications, protocol stacks, MAC layer modifications and/or other experiments on the testbed. To give an idea of the flexibility that the software needs to provide, consider the following sample experiment scenarios,

- Simple users may only want to define a network topology using standard MAC, network and transport layer protocols as well as a standard traffic generator. The user may be interested in measuring standard supported statistics such as throughput, average delay, packet loss etc. These tools are provided as default libraries.
- More advanced users may want to run cross-layer experiments, which will need support from the kernel so as to allow access to the data and control plane of all layers of the standard protocol stack. For such users, full node access can be provided along with a framework for measurement collection so that they can easily define new statistics, choose measurement points and collect measurements based on samples or time intervals.

A layered approach and modular design, with open APIs, hides the unnecessary details of experiment operation and complexity from users. In addition to the testbed software packages and libraries, it is expected that re-usable components and packages will also be developed by the user community. In order to support user experiments, the ORBIT testbed has a software framework as shown in Fig.4 consisting of management/control software as well as user level application software for the radio nodes.

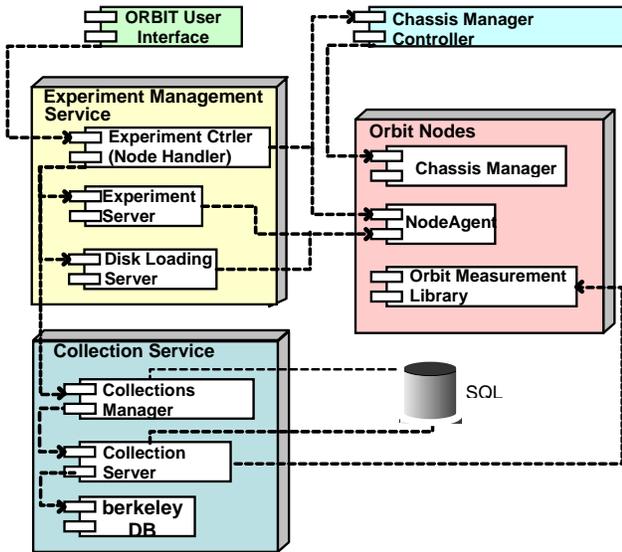


Figure 4 Software Architecture of ORBIT Testbed

1) *Management/Control Software*

The following testbed and experiment management software components have been developed.

a) *Node Handler*: The purpose of the Node Handler is to disseminate experiment scripts using multicast to the Node Agents residing on the individual nodes, in order to orchestrate the experiment. The Node Handler is *Tcl*-based and processes the experiment script, keeps track of the experiment steps and events, and sends them to the involved Node Agents at the appropriate time. The Node Agent reports back the state of experiment command execution to the Node Handler

b) *Collection Server (CS)*: The purpose of the collection server is to collect the reported measurements during the experiment. The nodes collect the statistics and send them to the collection server over a multicast channel after encoding them into XDR [8] format. This multicast channel is unique per experiment. The collection server provides a type-safe mechanism to collect experimental results reliably and store them for post-processing.

c) *Disk-Loading Server*: The purpose of the disk-loading server is to enable quick re-imaging of hard disks on the nodes as per the requirements of the user. This service works over a reliable multicast session using Frisbee [9] and is highly scalable. It allows for different groups of nodes to have different OS images between experiments.

2) *Software for Radio Nodes*

The following software components and libraries have been developed based on Linux kernel 2.6.4 as target platform to support the experiment and to provide libraries and interfaces for the user application development.

a) *Node Agent*: This is the component equivalent to NodeHandler that resides on the ORBIT nodes and listens to commands from the ORBIT Node Handler. It can run and stop the applications, dynamically pass the parameters to the applications, and report the experiment state to the controller.

b) *ORBIT Measurement Library (OML)*: OML defines the data structures and functions for sending/receiving and encoding/decoding measurement data that is exchanged in XDR format. Testbed users have the option to choose the filters to be applied to each measured metric. OML is used at the radio nodes (clients) and collection server. This software has been developed to reduce the burden of statistic collection on application developers.

c) *Libmac*

Libmac is a custom user-space C library that allows the applications to inject and capture MAC layer frames. It also allows manipulation of wireless parameters such as TxPower, channel settings and recording RSSI, noise on an aggregate and a per-packet basis. The primary purpose of libmac is to provide a bridge between device drivers and the applications such that application developers can easily use a standard interface to communicate with wireless device drivers instead of worrying about their underlying mechanism.

IV. LIFECYCLE OF AN EXPERIMENT

A typical ORBIT experiment involves experiment definition, node assignment, node configuration, loading of software packages, configuration of dynamic parameters and data collection. As shown in Fig. 5, the following steps are typically involved in an experiment.

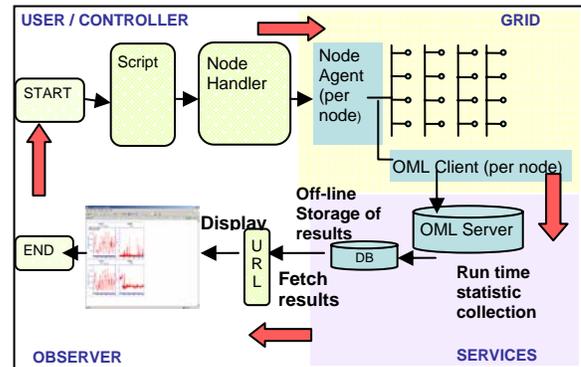


Figure 5 Life Cycle of an Experiment

- The experiment details are translated into a script that identifies the nodes to be assigned for the experiment, configures the wired and wireless interfaces according to the requirements of the experiment, fetches the appropriate application, libraries required to run the experiment and specifies (optional) statistic collection points and intervals
- This information is disseminated by the *NodeHandler* software to the corresponding *NodeAgent* residing on each node.
- The *NodeAgent* executes the script, performs the experiment which may involve statistics collection done by the OML library.
- A separate run-time and post-experiment database allows users to quickly view results during experiment run-time as well archive them for future retrievals and offline analysis.

V. SAMPLE EXPERIMENTAL RESULTS

To illustrate the life-cycle of an experiment, we explain a few sample experiments that follow the flowchart described in Fig. 5, with a script to define the nodes involved, configure interfaces, download necessary traffic generator and libraries needed to run the experiment, configure the statistics collection parameters and the database, and then handle the dynamic aspects such as changes in offered load, channels etc.

A. Experiment 1: To study the effect of 802.11b interference on the performance of a link under test (LUT)

1) Experiment details

The experiment consists of 8 nodes, with a sender sending UDP packets to a receiver (that forms the LUT) and six other interfering nodes that simply broadcast 802.11 packets on the same channel as the sender-receiver pair. Both the sender and all interferers transmit UDP packets of different packet sizes at 1 mW. All the nodes are configured to be on Channel 1 initially and then LUT is then moved away one channel at a time, until it operates on an orthogonal channel (Channel 6) w.r.t the interferers. The goal is to observe the effect on the packet loss of the obstructed link as it is moved to an orthogonal channel.

2) Experiment script

The above experiment contains a static configuration involving selection of nodes, initial configuration of interfaces such as channel settings, transmit power levels, IP addresses etc as well as fetching the appropriate application and libraries from the server to run the experiment. Fig. 6 demonstrates how this static configuration is translated into a *Tcl* script.

```
#Identify the nodes involved in the experiment using IP
# addresses, group them and reboot them.
expectNode node1-1 192.168.161.11 "sender" "exp1" "reboot"
expectNode node1-4 192.168.161.14 "receiver" "exp1" "reboot"
expectNode node3-1 192.168.161.31 "interferer" "exp1" "reboot"
...
#Set wireless interface in ad-hoc mode on all nodes (indicated
# by /*/* wildcards) w0 : wireless interface (eth2/wlan0).
configure /*/* /net/w0/mode ad-hoc
...
#Set the transmit power to 1 mW on all nodes of sender group .
configure /sender/* /net/w0/xmitPower 1

#When all nodes are configured, fetch application from server.
whenAll /*/*/system/status are "UP" do {
  install /*/* http://external1.orbit-lab.org/repository/exp1.tar
}
```

Figure 6 Sample script: Static configuration

Once the static part completes, the experiment is started and the dynamic parameter changes are handled as shown in Figure 7. Note that by using statements beginning with *whenAll*, it can be ensured that the pre-conditions necessary to execute the current instruction have been met. At the end of the experiment, the temporary database is cleaned; the results are time-stamped and stored on a separate database for easy access and future retrievals.

```
#Start collection server to enable statistics collection.
on /*/*/proc/status:RUNNING do {
  set url http://external1.orbit-lab.org/repository/oml_exp1.xml
  set s [::http::geturl "http://idb1.orbit-
  lab.org:5000/startCollectionServer?config_file=$url&app_nam
  e=exp1"]
}
#Start sender application on all the nodes of sender group.
run /sender/* sender_app /opt/orbit/bin/sender_app -c
  oml_client_pnp.xml -n node_name -t sender -i devw0 -p
  readparams
configure /sender/* /proc/genny/transport use_sock ...

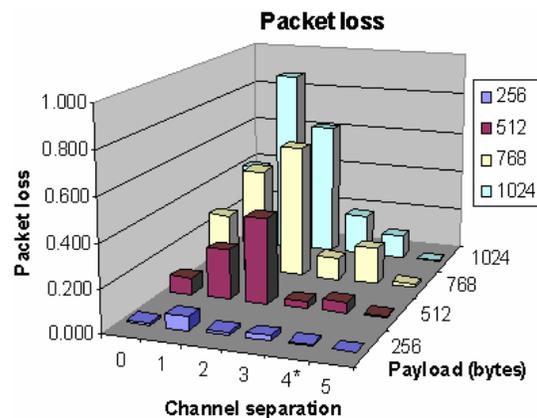
runExperiment $channel $packetSize $sleep $duration

# Dynamically changes channels from 1 to 6 and packet sizes
# from 256 to 1280 bytes during the experiment.
proc runExperiment {channel packetSize rate duration} {
  if { $packetSize > 1280 && $channel > 6 } {
    setStatus /experiment/state "DONE.OK"
    return
  } else {
    if { $channel > 6 } {
      set channel 1
      incr packetSize 256
      configure /sender/* /proc/sender_ap/payload_length
        $packetSize
    }
    after $duration [ runExperiment [incr channel] $packetSize $rate
      duration]
  }
}
```

Figure 7 Sample script: Dynamic configuration

3) Experimental Results

As shown in Fig. 8, as the channel separation between communicating pair and other interferers increases, the packet loss of the communicating pair reduces.



* Note that an infrastructure AP was operating on the adjacent channel

Figure 8 Packet loss for different channel separations and payload lengths

When the channel separation is 1 or 2, packet loss is higher due to lack of proper carrier sensing between the sender and the interferers. It is interesting to note that the packet loss for all packet sizes increases slightly when the channel separation is 4 (i.e the sender-receiver link is on Channel 5). This is attributed

to the fact that Channel 5 happens to be adjacent to an infrastructure AP (on channel 6).

B. Experiment 2: Effect of varying transmit power of sender on the performance in the presence of interferers

As a follow up to the previous experiment, we demonstrate the effect of changing the transmit power of the sender-receiver link while keeping the interferers' at 1mW on the packet loss for 1024-byte UDP packets at an offered load of 4 Mbps. As before, we have one sender-receiver pair (LUT) and 6 interferers. The channel separation between the LUT and interferers is progressively increased.

We observe that as the channel separation between the LUT and the interferers increases, the packet loss drops for all transmit power levels. Interestingly, we note that for channel separations of 1-4, the performance of the LUT is progressively better as the transmit power of the sender increases from 1mW to 100 mW. This indicates that interference on adjacent channels may be combated by adjusting transmit power levels of desired transmission.

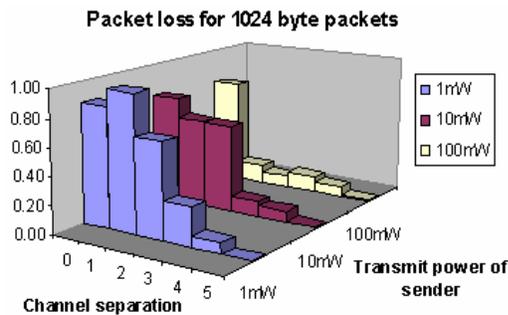


Figure 9 Effect of increasing Tx power of obstructed link in the presence of interferers

C. Multi-hop experiment with dual interface Forwarding node (FN)

The goal of this experiment is to measure the improvement in network performance in terms of throughput and packet loss for a multi-hop network with and without using a dual interface forwarding node. In Scenario 1 as shown in Fig. 9, we set up a chain topology of three nodes, with node 1 as source, node 3 as sink and node 2 as forwarding node. Node 2 is configured to forward the packets 1024 byte UDP packets received from node 1 to node 3 using a single interface. All the nodes operate at 11 Mbps, on the same channel and in ad-hoc mode.

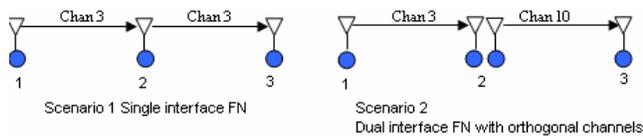


Figure 10 Multi-hop experiment with forwarding node

In Scenario 2, we configure node 2 to use two interfaces on orthogonal channels. The system throughput and packet loss for increasing offered loads are recorded. As seen in Fig. 10, the performance with a FN operating on orthogonal channels is

much better than a single interface FN in terms of throughput and packet loss.

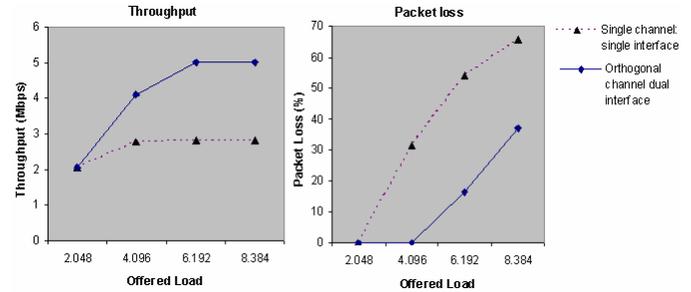


Figure 11 Performance of a multihop topology with a forwarding node (FN)

With the above different flavors of experiments as examples, we hope to validate and demonstrate the usability, flexibility, and user-friendliness that the ORBIT testbed provides for experimentation, data collection and analysis. Such a testbed can truly promote fair comparisons between results, and provide a useful platform to perform controlled experimental investigations of protocols.

VI. CONCLUSIONS

In this paper, we presented the design of a novel radio grid emulator testbed that is intended to facilitate a broad range of experimental research on next-generation protocols and applications. We have also explained a typical experimental lifecycle and provided sample experiments as proof-of-concept validation of the testbed design. Early end-user experiments on the ORBIT radio grid are expected to begin in the near future, and should lead to further validation and refinement of the testbed's design.

REFERENCES

- [1] David Kotz, Calvin Newport, Robert S. Gray, Jason Liu, Yongyu Yuan and Chip Elliott, "Experimental Evaluation of Wireless Simulation Assumptions, *Dartmouth Technical Report, TR2004-507*, <http://ftp.cs.dartmouth.edu/TR/TR2004-507.pdf>.
- [2] K. Pawlikowski, H.-D.J Jeong, and J.-S.R. Lee., "On credibility of simulation studies of telecommunication networks", *IEEE Communications Magazine*, 40(1):132-139, January 2002.
- [3] NSF Workshop on Network Research Testbeds., Chicago, IL, Oct 2002, http://www.net.cs.umass.edu/testbed_workshop/
- [4] Emulab Homepage, <http://www.emulab.net>.
- [5] Active Networks Backbone (ABone), <http://www.isi.edu/abone>.
- [6] C.E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing", *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.
- [7] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networks", *Mobile Computing* (edited by T. Imielinski and H. Korth), Kluwer, ch. 5, pp. 153-181.
- [8] XDR: External Data Representation Standard, RFC 1832, www.faqs.org/rfcs/rfc1832.html
- [9] Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci and Chad Barb, "Fast Scalable Disk Imaging with Frisbee", In *Proc. of the 2003 USENIX Annual Technical Conference*, June 2003.