Emulating Opportunistic Mobile Wireless Networks on ORBIT

Sankardas Roy, Jiang Li
Department of Systems and Computer Science, Howard University
sroy, lij@scs.howard.edu

1. Introduction

Our research project at Howard University deals with a special type of opportunistic mobile wireless networks (O-MWNet) known as **Pigeon Networks**. One of the main goals of the project is to emulate a pigeon network on **ORBIT testbed** which is physically located at Rutgers University. This testbed is accessible online and we are particularly interested in using its **GENI interface** for the emulation task. We measure the performance metrics (e.g. latency, throughput, and loss ratio) of a pigeon network in the emulation experiments. Simultaneously, we investigate the ease of using ORBIT testbed to do these experiments to uncover its strengths and limitations.

ORBIT''s GENI interface has two main elements: (a) the **OMF interface** allows an experimenter to prepare the machines of the testbed for the particular experiment and run the experiment., (b) the **OML library** provides the storage service for the experiment data**.** We extensively use both of these elements to emulate a pigeon network.

The rest of the poster has the following sections: A background of a pigeon network, a brief overview of the ORBIT testbed, how we emulate a pigeon network, our visualization module, a part of the emulation results, and finally, the critiques.

2. Pigeon Networks

A pigeon network is a special type of Disruption Tolerant Network (DTN). The unique feature of this specialized network is that each message carrier, also called a *pigeon*, is owned by a host node. There can be multiple host nodes present in the network, which are interested in communicating with each other. The owner host node of a pigeon is called the *home host* of the pigeon, and other host nodes are known as the *foreign hosts* with respect to that pigeon. A pigeon has dedication to its owner in the sense that it carries only those messages which are currently located at the owner, or which are destined to the owner. The fundamental unit of a pigeon network is a home host coupled with its pigeon, which is known as a *cell*. The whole network can be considered as the interaction among multiple cells.

In the emulation work, we focus on one *cell* (involving one home host and one pigeon) and its interaction with one or more foreign hosts. So, in a cell there are three kinds of nodes, namely (a) the pigeon P, (b) the home host H, and (c) multiple foreign host F_i. These nodes have the following characteristics. The pigeon node P is mobile and moves across the home host H and the foreign hosts whereas H and F_i are static. The pigeon node P acts as the message carrier. Figure 1 illustrates an example of such a network where the pigeon P starts from the home host H and visits the foreign hosts (F_i) in some sequence while it delivers/receives messages. The pigeon can return to the home host H before visiting all of the foreign hosts, and P can start again from H to make more visits to foreign hosts. The whole trajectory of the pigeon P is determined by the scheduling algorithm. We focus on the scenario where H has messages to be delivered to any foreign host F_i, and F_i can also have messages to be delivered to H. Of course, P is the message carrier for all of these messages.

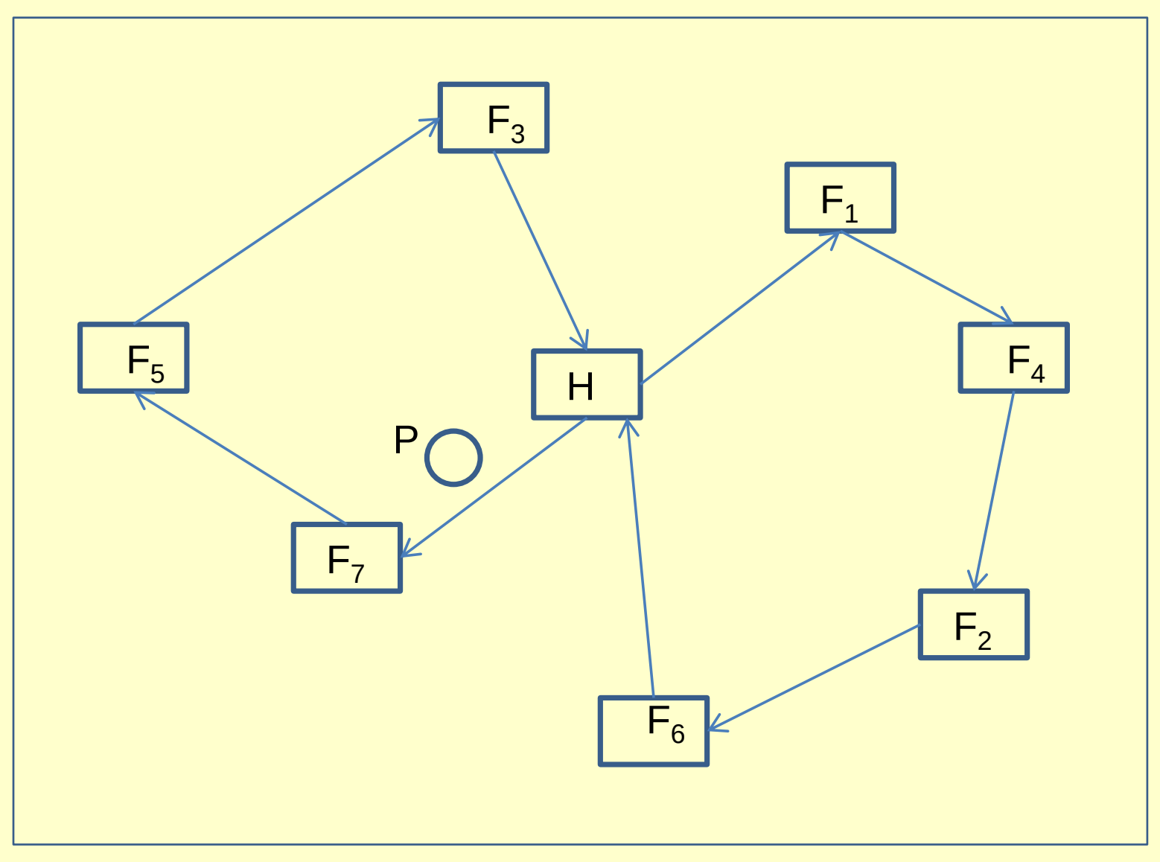


Figure 1: A pigeon network with multiple foreign hosts F_i – the pigeon P visits the seven foreign hosts F_i following a scheduling algorithm. P can visit the home host H with higher frequency, i.e. there can be multiple loops in P's trajectory, e.g. two loops in this example.

3. An Overview of the ORBIT Testbed

The ORBIT testbed (also called grid) has 400 computers which a user (i.e. an experimenter) can reserve for performing experiments. Each of these computers possesses a wireless antenna for communicating with others. If the user runs proper emulation programs in all or some of these computers, each such computer can take the role of one node of the emulated network. It is the responsibility of the experimenter to build the emulation executables and load them on the corresponding grid nodes before starting the emulation experiment.

The ORBIT testbed provides an interface known as the **OMF** interface through which the users can manage the resources. In particular, through the OMF interface a user can load image (with proper OS and precompiled executables) onto the grid nodes, and run the experiment on the grid nodes; the OMF interface can also save the current node image to permanent storage for future use.

A user is expected to write an **Experiment Description (ED)** script (which is in Ruby language) and run it via the OMF interface. ED is capable of acquiring the requested resources from the grid, configuring the nodes for the experiment, starting the emulation programs inside the nodes, and finally stopping the emulation programs after the requested time interval.

To help the user save the experiment data ORBIT testbed provides a measurement library called **OML**. The emulation program which runs on a grid node can use the OML APIs to save the experiment data during the experiment into a database server called the OML server. After the experiment is done, this data can be analyzed to compute the values of the intended metrics.

To further help the user explore the experiment data the ORBIT provides the **result service**. This service allows the user to retrieve the experiment data over the Internet in real time or after the experiment. The user can view the data on a web browser window on his/her local machine. Moreover, the user can add sql query to the browser URL to analyze the experiment data.

4. Emulating a Pigeon Network on ORBIT

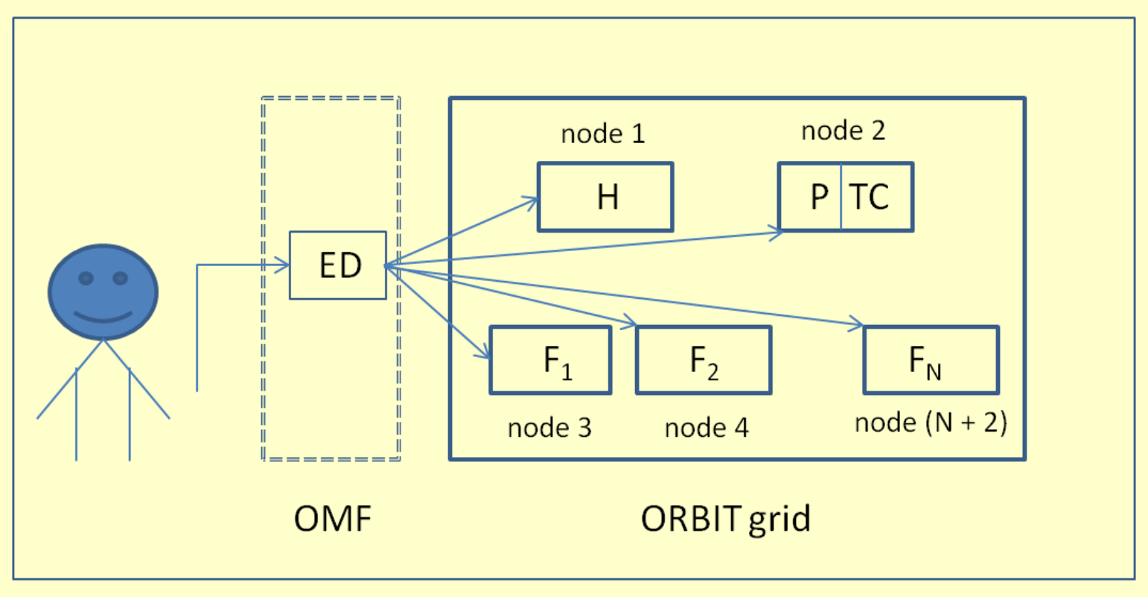
On ORBIT testbed we emulate a pigeon network which has one home host, one pigeon (i.e. message ferry), and multiple (say N) foreign hosts. The pigeon network is built using (N+2) real nodes of the ORBIT grid, among which one (say node 1) emulates the home host, one (say node 2) emulates the pigeon , and each of the rest (i.e. node 3, node 4, …, node N+2) emulates one foreign host. A packet filter e.g. iptables is used to emulate the mobility of the pigeon node and its transient connectivity with other nodes. To perform the experiment, the experimenter mainly needs two things: (a) an ED script which is run on the OMF interface, and (b) the emulation programs which are run on the grid nodes.

At the beginning of the experiment, the ED loads the driver module (e.g. ath5k) in each node, and configures the NIC as required. Then the ED starts the emulation executables inside the nodes, and finally terminates the executables after the requested time interval to stop the experiment.

We need to produce different emulation programs for building the pigeon network. Recall that the roles of the home host, the pigeon, and a foreign host are different. We have to a write a separate emulation program to realize each of the above roles. That means, we need one emulation program (called home program) to run in the home host node (i.e. node 1), one emulation program (called pigeon program) to run in the pigeon node (i.e. node 2), and one emulation program (called foreign program) to run in a foreign host node (i.e. node 3, node 4, …, node N+2).

On the pigeon node (i.e. node 2), in addition to the regular pigeon emulation program we also run one emulation program called topology controller (TC). The TC uses a packet filtering technique (e.g. iptables) to control the connectivity of the pigeon with the host nodes.

An ED script is run via **OMF (version 5.3)** interface. The ED starts the home program H, pigeon program P and foreign host programs F_i in the grid nodes. Our ED script is flexible to variable number of foreign hosts.



The ED also starts one emulation program called topology controller (TC) which runs in the pigeon node in addition to the regular pigeon program P. The TC uses a packet filter to control the connectivity of the pigeon.

Figure 2: The design of pigeon network emulation – the experimenter runs an Experiment Descriptor (ED) via the OMF interface, which manages (N+2) nodes on the grid. Node 1 runs host program H, node 2 runs the pigeon program P as well as the topology controller TC, and each of the rest of the nodes runs a foreign host program Fi.

We extensively use OML APIs (oml2 library) inside the emulation programs which allows them to keep on injecting observed data during the experiment into the OML server. The data are analyzed using a query language (e.g. sqlite3) after the experiment to compute the intended performance metrics.

We have explored how to extend the ED to run an experiment with multiple instances. The motivation is that in real life the experimenter often needs to run multiple iterations of an experiment with some varying parameters to collect statistically significant results. We have tried to design the ED in such a way that we move towards automating the whole experiment. However, we have not been fully successful.

5. Building a Visualization Module

We build a visualization module which shows the topology (location of the nodes) of the pigeon network on a browser window. It displays the real-time movement of the mobile nodes (e.g. the pigeon) in addition to the static nodes of the network as emulated on the ORBIT grid. We use d3.js module on the browser side which is a Javascript library for the visualization task.

We build a proxy server in the local machine which intermittently retrieves over the Internet the real-time experiment data from the OML server through the ORBIT **result service**. The data is stored in a local file and the browser keeps on receiving the data from the proxy server via asynchronous read operation. Figure 3 illustrates the whole system and shows the data flow path.

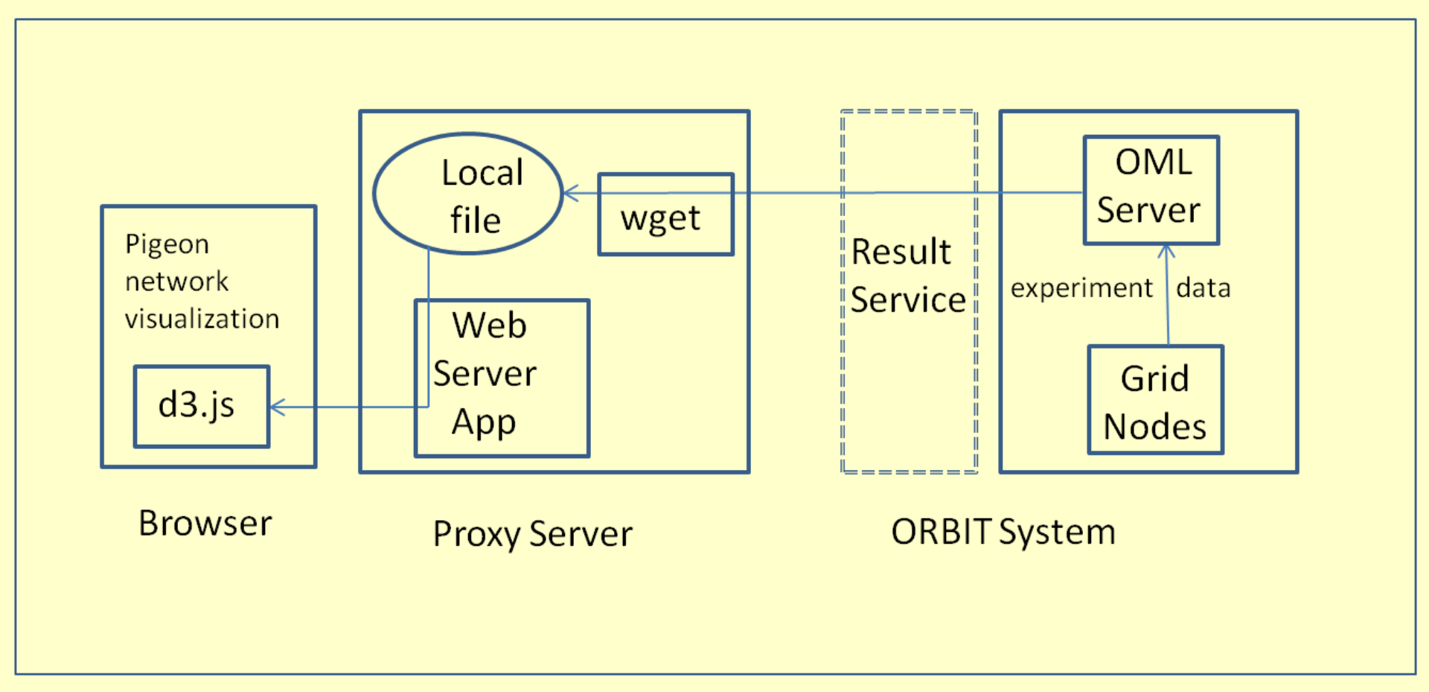


Figure 3: The real-time data flow in the visualization system– the experiment data are injected by the ORBIT grid nodes into the OML server, and then the wget program of the Proxy Server downloads the data in a local file. Finally, the Browser retrieves the data asynchronously and displays the emulated network.

6. Emulation Results

We analyze the OML data after the experiments to compute the intended matrices such as message latency. As an example, below we present one plot of the average message delay Vs. the pigeon's speed in emulation. It also compares the result with the numerical computation (NC). We notice that NC outputs less latency because (a) in emulation the connectivity of the pigeon with a host is discrete, (b) NC does not consider the message pickup and delivery time.

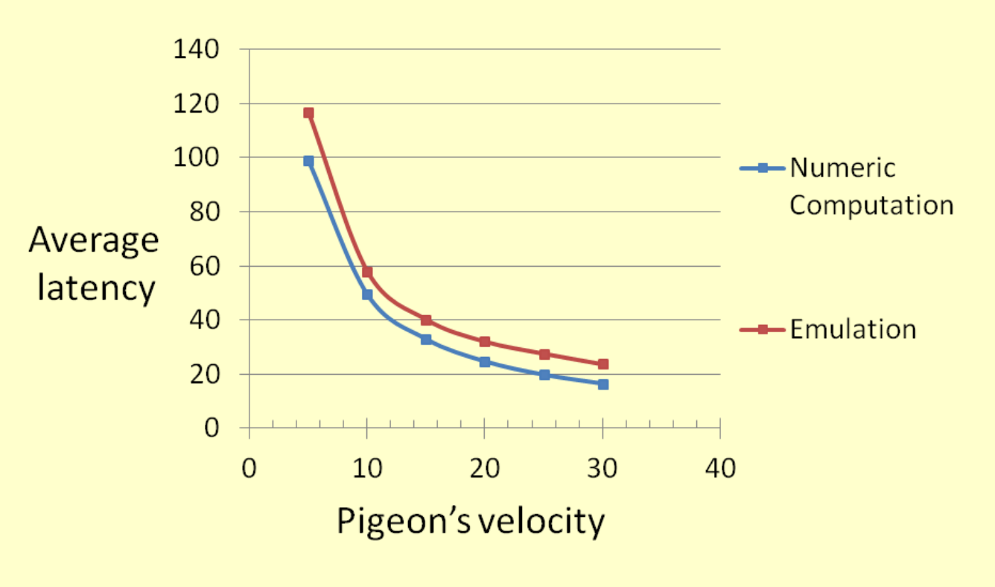


Figure 4: the average message delay Vs. pigeon's speed.

7. Critiques on ORBIT testbed

- Strengths
  (a) The administrators of ORBIT are always ready to provide with their kind help, detailed instructions, and suggestions through numerous emails. (b) the web service which an experimenter uses to reserve the resources works fine. It allows the experimenter to reserve the grid or a sandbox for a specific time interval over the next one week.

- Limitations
  (a) Lack of detailed/consistent information on ORBIT wiki, (b) unavailability of ED features (to the best of our knowledge) specialized for running a long duration experiment with multiple instances, (c) experiment involving tens of nodes on the ORBIT grid occasionally fail because some node may become unresponsive in the middle of the experiment.

- Our suggestions
  (a) A user can get information about ORBIT way faster had the wiki delivered updated and consistent information. It would be even better had the wiki possessed a tool to search for specific information via key words. (b) Unfortunately, we have found that with the current infrastructure, it is not possible to fully automate an experiment with multiple runs with varying parameters. A couple of new features need to be introduced with ED, which could help in achieving full automation in future.