A Demonstration of the RapidMesh Development Toolkit

Xiaozhou Li* Shivkumar C. Muthukumar* Changbin Liu* Joseph B. Kopena†
Mihai Oprea* Ricardo Correa* Boon Thau Loo* Prithwish Basu‡
*University of Pennsylvania †Drexel University ‡BBN Technologies
{mshivk,xiaozhou,changbl,mihaio,ricm,boonloo}@seas.upenn.edu,
tjkopena@cs.drexel.edu, pbasu@bbn.com

ABSTRACT

We propose the demonstration of the *RapidMesh* development toolkit for protocol simulation, implementation and experimentation of wireless mesh networks. *RapidMesh* utilizes *declarative networking*, a declarative, database-inspired extensible infrastructure that uses query languages to specify network behavior. *RapidMesh* integrates a declarative networking engine with the emerging ns-3 network simulator. Our demonstration presents the experimental evaluation of a variety of declarative MANET routing protocols under different dynamic settings on the ORBIT wireless testbed. The evaluation results are shown in the ns-3 visualizer to display the mobility pattern of network and to compare the protocol performance. We also demonstrate the development cycle for synthesizing and experimenting with new routing protocols.

1. INTRODUCTION

RapidMesh is a development toolkit for network protocol simulation, implementation and experimentation. We propose the demonstration of *RapidMesh* in conjunction with our paper presentation [5] in WiNTECH 2009. *RapidMesh* is a step towards building systematic tools for experimenting with routing protocols for mobile ad hoc networks (MANETs) under a variety of mobility settings. It also aims to provide a unified platform for evaluating the protocols in simulation and testbed-based emulation modes.

RapidMesh utilizes *declarative networking*, a declarative, database-inspired extensible infrastructure that uses query languages to specify network behavior. *RapidMesh* integrates a declarative networking engine with the emerging ns-3 [6] network simulator which is intended as an eventual replacement for the ns-2 simulator. Network protocols are specified using declarative specifications, which are then compiled into ns-3 code for simulation and analysis. The same declarative specifications can also be used as actual implementations under the ns-3 network emulator, thus providing a bridge between simulation and testbed-based experimentation.

A declarative approach enables modular reuse of resources and functions by allowing network programmers to say "what" they want, without worrying about the details of "how" to achieve it. Declarative networks are specified using *Network Datalog* (*NDlog*), which is a distributed recursive query language for querying networks. Declarative queries such as *NDlog* are a natural and compact way to implement a variety of routing protocols and overlay networks. For example, traditional routing protocols such as the path vector and distance-vector protocols can be expressed in a few lines of code [4], and the Chord distributed hash table in 47 lines of code [3]. When compiled and executed, these declarative networks perform efficiently relative to imperative implementations.

2. OVERVIEW

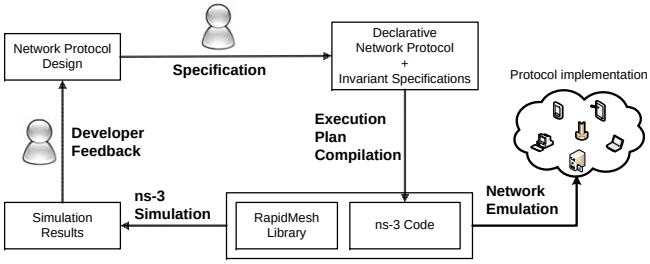

Figure 1: Overview of *RapidMesh*

Figure 1 provides an overview of *RapidMesh*. In the initial design phase, a network protocol design is used as the basis for specifying the network protocol using the *NDlog* declarative networking language. High-level invariant properties of the protocol can also be expressed in *NDlog* as *distributed triggers* which raise event alarms when invariants are violated.

In the *simulation mode*, the *RapidMesh* compilation process generates ns-3 code from the *NDlog* protocol specifications and invariants. The generated code either runs as an ns-3 application, or replaces routing protocol implementations at the network layer. The generated code implements dataflows (execution plans) with a similar execution model with the Click modular router [2].

In the *emulation mode*, declarative networking specifications are directly executed and deployed by using the the ns-3 network emulator. Each ns-3 simulation node

connects to the real physical network underneath using a raw socket. The main advantage of this is ensuring that all declarative networking protocols are evaluated in simulation and emulation within a common ns-3 code base.

3. DETAILS OF DEMONSTRATION

Our demonstration will showcase the implementation and evaluation of declarative MANET routing protocols on the ORBIT [9] wireless testbed. We select MANET protocol implementations, ranging from link-state routing (LS), hazy-sighted link-state routing(HSLS) [8], optimized link-state routing (OLSR) [1], dynamic source routing (DSR), and summary-vector based epidemic routing. We evaluate these protocols using different mobility patterns (e.g. random waypoint, Brownian motion, hierarchical mobility, etc.) supported by ns-3 and under different node speeds by running them in emulation mode. The network event traces from the emulation experiment will be replayed in the ns-3 visualizer [7].

The ns-3 visualizer is designed to playback network events by reading trace logs. It displays the actual movement of nodes in the arena to show the mobility pattern, node coordinates and speeds, etc. We have enhanced the visualizer to show protocol performance statistics side by side as the visualization progresses. We use this to show the per-node communication bandwidth and route quality metrics: *route validity* and *route stretch*.

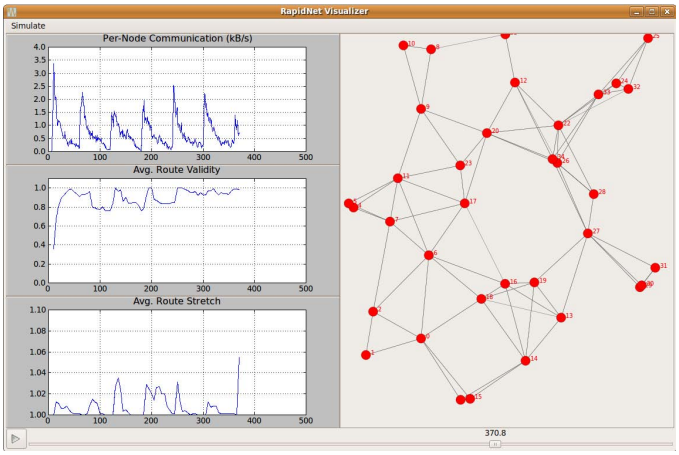

Figure 2: Screenshot of the visualizer

To illustrate, Figure 2 shows an example execution of the current version of our demonstration. The right panel shows the arena with the nodes and the links connecting them. On the left-panel we have one dynamically updating graph for each protocol statistic. This example shows the traces from a declarative link state protocol experiment running on the ORBIT testbed with 35 nodes which communicate with each other via IEEE 802.11a and move with Brownian motion model.

As another example, the visualization of the DSR protocol can display the network events like *RREQ* and *RREP* by coloring the nodes along the path. The network statistic plotted in this case is the communication bandwidth per requested route.

Further, we will demonstrate the development cycle of *RapidMesh* by walking through implementing an example protocol. We will show the protocol synthesis in *NDlog*, compilation to intermediate and ns-3 code using the *RapidMesh* compiler and executing it in simulation mode. Finally, we replay the trace logs in the visualizer.

4. ACKNOWLEDGMENTS

This work is based on work supported in part by NSF grants CNS-0721845, CNS-0831376, CCF-0820208, and CNS-0845552.

5. REFERENCES

[1] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). In *RFC 3626 (Experimental)*, 2003.

[2] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[3] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *ACM SOSP*, 2005.

[4] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *ACM SIGCOMM*, 2005.

[5] S. C. Muthukumar, X. Li, C. Liu, J. B. Kopena, M. Oprea, R. Correa, B. T. Loo, and P. Basu. RapidMesh: Declarative Toolkit for Rapid Experimentation of Wireless Mesh Network. In *The Fourth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, 2009.

[6] Network Simulator 3. http://www.nsnam.org/.

[7] ns 3 visualizer. http://code.nsnam.org/tjkopena/ns-3-decorator3/.

[8] C. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *ACM MobiHoc '01*, Long Beach, CA, 2001.

[9] O. W. N. Testbed. `http://www.winlab.rutgers.edu/docs/focus/ORBIT.html`.

6. DEMONSTRATION LOGISTICS

We will use two laptops to run the ns-3 visualizer and to provide an overview of *RapidMesh* development cycle. We will need access to two power sockets, one to power each laptop. No Internet access is necessary or assumed for the demonstration to work. We expect the demonstration setup time to be within 15 minutes.