CentMesh: Modular and Extensible Wireless Mesh Network Testbed*

J. Lim[1], P. Pathak[1], M. Pandian[2], U. Patel[1], G. Deuskar[1], A. Danivasa[1], M.L. Sichitiu[2], and R. Dutta[1]

[1] Dept. of Computer Science,
North Carolina State University, Raleigh, NC, USA
[2] Dept. of Electrical and Computer Engineering
North Carolina State University, Raleigh, NC, USA
dutta@csc.ncsu.edu

Abstract. In this paper we present the design of our wireless mesh network testbed (CentMesh), which facilitates experimentation as a service. CentMesh differs from other testbeds in terms of its modular, flexible and extensible design. The CentMesh software suite provides a modular programming library that can be modified and/or extended by the users of the testbed, allowing them to implement their own modules (e.g., routing, scheduling etc.). The basic services such as transport of control messages, broadcast, etc., are provided to experimenters by a set of system modules. Modularity allows the experimenters to implement only the part of network stack that they are interested in experimenting with, while reusing the other readily available CentMesh modules.

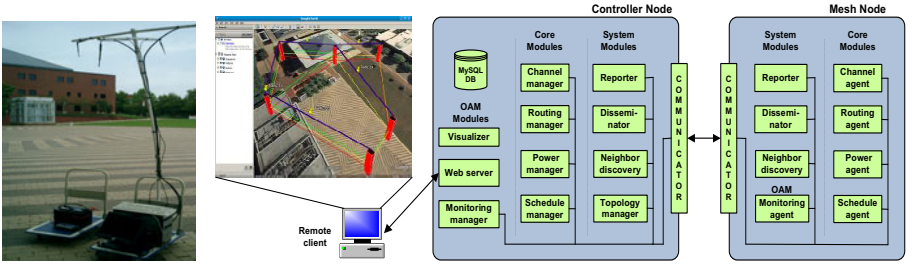
1 Introduction and Motivation

Even though wireless multi-hop networks have been the focus of considerable research efforts in last few years, conducting real-world experiments with wireless devices continues to be difficult. Most of the early efforts [1, 2, 3, 4, 5] in testbed design have focused on simplifying the process of experimentation and facilitating a certain amount of repeatability.

In this paper, we present the design and development of our wireless mesh network testbed *CentMesh*. The fundamental design principle of CentMesh is a clear separation between data transport, signaling and control and management algorithms. The modular structure of CentMesh software allows users to *plug-and-play* existing modules and add new modules. The advantage of such a design is two-fold. First, it allows the researchers to implement only the modules of the stack that they are interested in experimenting with. Second, newer modules needed for various experiments can be developed rapidly and integrated in the testbed independently of all the other testbed modules.

* This work is supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI). The contents of this paper do not necessarily reflect the position or the policies of the U.S. Government.

T. Magedanz et al. (Eds.): TridentCom 2010, LNICST 46, pp. 619–621, 2011.



(a) CentMesh node (b) CentMesh software architecture

Fig. 1. CentMesh software and hardware

2 CentMesh: Hardware and Software

2.1 Hardware Components

CentMesh uses commodity hardware, each mesh node being a standard desktop computer. The CentMesh software is designed to run on any other hardware platform since we do not use any customized devices. Currently, the testbed consists of 10 mesh nodes deployed inside our department building. Each mesh node contains four Atheros cards, each connected on 4-to-1 miniPCI-to-PCI adapter. While performing the outdoor experiments, we place the mesh nodes on pushcarts (Fig. 1a).

2.2 Software Architecture

The CentMesh testbed uses a centralized control system where one of the mesh node is assigned the role of the controller. Different from other current state-of-art testbeds, we do not deploy any wired backhaul using Ethernet to connect and control the nodes. Instead control messages flow between nodes using one of the radios operating on a fixed channel. The design of CentMesh is modular and clearly separates functionality in control, management, and data planes. The modules are divided into system modules, core modules, and extension modules.

System Modules. System modules are implemented an indispensable part of the CentMesh software that is always present and running on mesh nodes irrespective of the experiment.

The testbed is managed cooperatively by several processes. Control and signaling traffic between processes either local or remote is channeled through a single process called the Communicator. Instead of the client/server model, the Communicator uses a publish/subscribe mechanism, which allows processes to send and receive messages based on the topic they are interested in, while hiding the details of the underlying message processing. Topic-based message filtering in the publish/subscribe model creates logical channels in the network-wide control path, which provides greater flexibility and scalability compared to a typical client-server model. Fig. 1b shows the Communicator as an entry/exit point between various processes running on different mesh nodes. The Communicator supports common multihop operations such as unicast reporting and flooding.

The neighbor discovery process on a mesh node periodically probes its neighbors and collects local neighborhood information. All nodes report the neighbor information to the controller.

For ease of recovery, in case of system crash, every CentMesh node contains three separate Linux installations in three separate disk partitions. The first partition (referred as fail-safe) contains a minimal set of functionality that allows users to remotely access the node and manually recover the node. The second and the third partitions are open to the researchers to perform their experiments.

Core Modules. Core modules are developed by the researchers as part of the experiment. Using the APIs from the CentMesh software, software modules are implemented in general as paired managers and agents; managers contain central intelligence/algorithm of the protocol, and agents perform actuating tasks based on the manager's decision.

We developed two sample core modules: a channel assignment module that uses a greedy edge coloring algorithm and a routing protocol with ETT (Expected Transmission Time) as the routing metric. Both modules can be replaced by experimenters with their own channel assignment and routing modules; however, a channel assignment and a routing module must always be present. An experimenter who is not focused on routing research can reuse the available routing module for the experiments. Similarly, experimenters can swap any other core modules.

Extension Modules. A separate network monitoring module (designed similar to core modules) collects network status information (e.g., installed routes, interface information, data rates, etc.) and reports it to the controller. The visualization program extracts the information provided by monitoring manager and graphically displays the network status using Google Earth.

3 Research Studies and Ongoing Extensions

CentMesh is designed to support many different types of of research projects. Some of the ongoing research projects include coarse-grain TDM scheduling and joint channel assignment-routing protocols. Furthermore, CentMesh also supports security research at various layers as well as mobility management and modeling. In the next phase of deployment, we plan to install the mesh nodes outdoors in our university campus. We soon plan to release the software suite under open source license to make it available to other researchers for their use.

References

1. ORBIT Lab, `http://www.orbit-lab.org`
2. Emulab, `http://www.emulab.net`
3. MIT Roofnet, http://pdos.csail.mit.edu/roofnet
4. TFA Rice Wireless Mesh, `http://tfa.rice.edu`
5. UCSB MeshNet, `http://moment.cs.ucsb.edu/meshnet`