CentMesh: Modular and Extensible Wireless Mesh Network Testbed △

JunBum Lim, P. H. Pathak, U. Patel, G. Deuskar, M. Pandian*, A. Danivasa, M. Sichitiu*, R. Dutta
Department of Computer Science, *Electrical and Computer Engineering Department,
North Carolina State University, Raleigh, NC, USA. Email: 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 using which users can implement their own modules (such as routing, scheduling etc.) and can plug them in and out of network stack. The basic functionality such as transport of control messages, broadcast etc. are provided to experimenters via a set of system modules. This allows the experimenters to implement only the part of network stack which they are interested in experimenting with, while reusing the other readily available CentMesh modules. Such an extensibility and reuse reduces the experimenters' overhead in development and configuration, and allows them to focus on the research problem of interest.**

I. Introduction and Motivation

Even though wireless multi-hop networks have been the focus of abundant research efforts in last few years, difficulty in conducting real-world experiments with wireless devices continues to be an issue. On the other hand, evaluation of protocols using wireless testbed in realistic environments is essential to understand their real-world applicability. An increasing amount of current research is directed towards *cross-layer* protocol design and issues addressing heterogeneity. This has lead to research and development of various wireless testbeds which provide experimentation platform as a service to research community. ORBIT [1] and EmuLab [2] are examples of testbed which allow researchers to perform experiments in controlled environment. Other wireless mesh testbed and experimental deployments include Roofnet [3], TFA network [4], UCSB MeshNet [5] etc.

Current wireless mesh network testbeds provide good amount of control over device configurations and overall experimentation cycle. Most of the early efforts of testbed design have focused on simplifying the process of experiments and facilitating a certain amount of repeatability. Though such factors are essential to consider in testbed design, there is a apparent need of clearly separating experimental concerns from the underlying testbed functionality. Operations such as information collection and dissemination are common in almost all protocol design, and such a distinction is necessary to allow experimenters to focus their efforts on core protocol design. As an example, an experimenter/researcher who is interested in evaluating his or her interference-aware channel assignment strategy on testbed should be allowed to implement it with great deal of flexibility without him or her worrying about how control messages gathering interference information will flow between mesh nodes.

In this paper, we present design and development of our wireless mesh network testbed *CentMesh*. CentMesh is a modular and extensible testbed that allows researchers to perform experiments in multi-radio mesh settings and evaluate their protocols. The fundamental design principle of CentMesh is clear separation between data transport, signaling and control, and management algorithms. The modular structure of CentMesh software allows users to *plug-and-play* various existing modules and add new modules. The advantages of such a design are two-fold. First, it allows the researchers to implement only the parts (modules) of the stack that they are interested in experimenting with. Second, newer modules developed during various experiments can grow rapidly which provides easy reuse and extensibility in other experiments. We discuss hardware components and software architecture in detail in next sections.

△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.

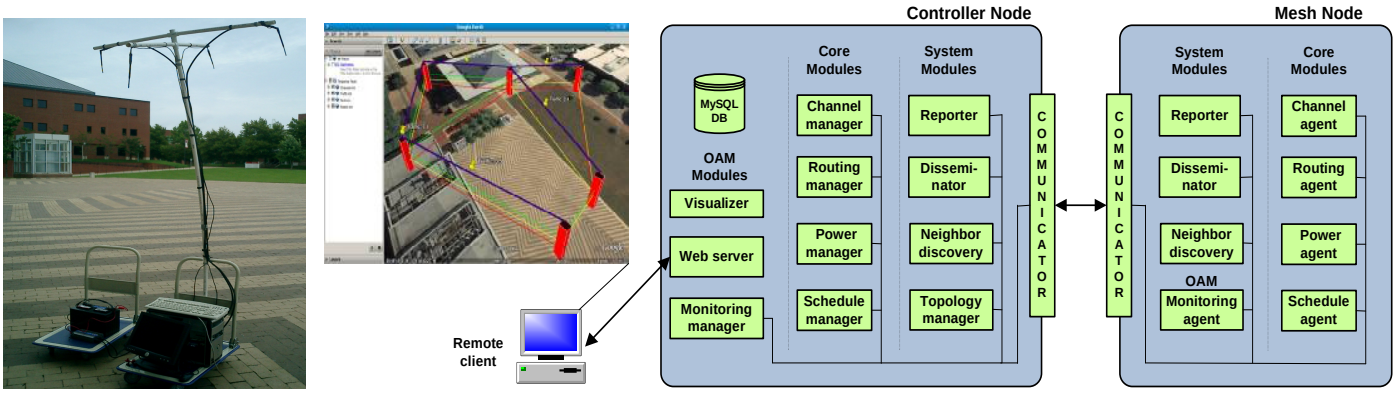
II. CentMesh: Hardware and Software

A. Hardware Components

CentMesh uses commodity hardware where each mesh node is a standard desktop computer. The CentMesh software is designed to run on any other hardware platform since we do not use any customized devices. A typical mesh node consists of a 1.5-GHz processor with 1 GB RAM and 20 GB hard disk. Also, we use EMP-8602 PLUS-S mini-PCI wireless cards which operate on Atheros AR5006 chipset providing 802.11 a/b/g networking. Every mesh node contains upto 4 radios, each connected on 4-to-1 miniPCI to PCI adapter. We use a PVC pipe assembly to create substantial separation between antennas (5dBi rubber duck omnidirectional) of radios. Each mesh node runs Fedora distribution of Linux and uses MadWiFi open-source driver for wireless cards. Currently, the testbed consists of 10-12 mesh nodes deployed inside the department building. While performing the outdoor experiments, we place the mesh nodes on pushcarts (Fig. 1a), which also helps in creating topologies of interest. The mesh nodes are powered with Power Express 400 Watt mains power inverter connected to Autocraft marine deep cycle batteries. Also, each mesh node is equipped with a GPS (Garmin 18x) which constantly provides location coordinates information. As described in the future work, we plan to complete the mesh nodes installation on top of equipment poles in Centennial campus in near future.

B. Software Architecture

The CentMesh testbed is a centralized control system where one of the mesh node is assigned the role of the controller. Though the underlying CentMesh software remains the same on every node, different services are started depending on whether a node is a controller node or a mesh node. 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 (fixed but not dedicated) operating on a fixed channel. Researchers can write their own modules, create



(a) CentMesh mesh node

(b) CentMesh software architecture

Fig. 1: CentMesh software and hardware

testing scripts and distribute them to mesh nodes. At the end of the experiments, similar automated services are provided to gather the experiment output and analyze the results.

CentMesh testbed design modularizes and clearly separates functionality in data, control and management planes. The modules are divided into system modules, core modules and extension modules. System modules provide basic functionality like transport of control messages, broadcast services, topology management etc. Also, researchers are provided with a modular programming library which they can use to write their own modules (core modules such as routing, channel assignment, power control etc.) and plug them in and out of the network stack.

1) System Modules: These modules are implemented as a part of the system software and are provided to the experimenters. They are an indispensable part of the CentMesh software that is always present and running on mesh nodes irrespective of type of experiments.

a) Communicator: All control and signaling traffic between mesh nodes is channeled through the communicator module. It provides reliable communication (using TCP connections) between various core modules and enables distributed design in a loosely coupled manner. Fig. 1b shows how the communicator works as an entry/exit point between various processes running on different mesh nodes.

b) Reporter and Disseminator: Build on top of communicator, reporter provides a unicast reporting functionality which can be used by mesh nodes to report various data to the controller. Similarly, disseminator acts as a flooding agent which can be used by a mesh node (typically controller) to broadcast information to other mesh nodes. Researchers can create their own messages with necessary information and can report/disseminate them easily using reporter and disseminator.

c) Neighbor Discovery: One of the most common requirements of any protocol design is correct detection of physical layer topology. The neighbor discovery process on a mesh node periodically probes its neighbors and gathers the local neighborhood information. All nodes report the neighbor information to the controller, which then realizes the entire network topology.

d) Bootstrap Routing: Based on the current network topology, controller node computes a simple and robust set of routing paths which are then distributed to the mesh nodes for actuation. Such a routing is necessary for example when experimenter's core module does not deal with routing at all and relies on system modules to establish and maintain the routes.

e) Data Repository: The controller node maintains a database which contains virtually every information regarding the network (link states, node status etc.). The information is periodically refreshed by various system modules. Information in the database is made available as input to the management algorithms of core modules. This way, the data repository acts as decoupling intermediary between various data collections and dissemination procedures.

f) System Crash and Recovery: It is possible that the modules developed by researchers can create software failures or traps which requires a systematic method of recovery. To address this, every CentMesh node contains three separate Linux installations in three different disk partitions. The first partition (referred as fail-safe) contains a minimal set of functionality which allows users to remotely access the node and recover it. The second and the third partitions are in general open to the researchers to perform their experiments. In case of system crash during the experiments, researcher can use the fail-safe mode to make changes to any other partition and resume their experiments. The access to the fail-safe mode by researchers is limited for the purpose of recovery and the fail-safe mode in general is maintained by the CentMesh administrator.

All the above mentioned system modules are made available to the researchers as part of the CentMesh software suite. Using these modules and the development API, researchers can easily write their own core modules.

2) Core Modules: We use publish/subscribe relationship to allow various core module processes on different mesh nodes to send and receive messages to each other based on the topic names. Current publish/subscribe mechanism is implemented on the top of communicator which maintains a directory of known topics and respective subscriptions. Such topic-based message filtering in 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. The management modules (or core modules) are implemented as paired managers and agents. The manager component of a core module resides typically on the controller

node while the agent component is functional on mesh nodes. The manager of a core module contains the central intelligence/algorithm of the protocol. The agents on the other hand are light-weight which gather necessary information for input to the manager algorithm and perform actuating tasks based on manager's decisions. As an example, a newly written core module (see Fig. 1b) for routing contains a routing manager (residing as a process in the controller) and a routing agent (in mesh nodes). The routing manager calculated the set of routing paths based on any specific strategy and distributes the routing paths to different nodes. The routing agents on different nodes which have subscribed to the particular topic of routing receive the route decisions and set the routes on the mesh nodes accordingly. Again, the actual transport of control messages such as routing messages from manager to agents is performed using preexisting system modules.

Hence, a core module is a module developed by the researcher to perform experiment. Following core modules are also developed as sample core modules for the purpose of reuse and demonstration. They may be replaced by experimenters by their own modules, but some routing module, some channel assignment, etc. must always be present. As an example, an experimenter who is not dealing with routing can reuse readily available routing module for the experiments.

a) Routing: The routing considers ETT (Expected Transmission Time [6]) of links as the metric for route calculations. It is well-known that ETT based routes achieve significantly better throughput than simple hop-count based routing paths. The routing agent on each node gathers ETT values of links and reports back to controller node. The controller node collects the data and puts it into the database. The routing manager operating on controller node retrieves the data and runs the routing algorithm to determine feasible set of paths. The routes are then distributed to all nodes and the entire process is repeated periodically.

b) Channel Assignment: Similar to routing module, the channel manager uses greedy edge coloring algorithm on the network topology graph and determines the channel for each radio. The current channel assignment mechanism is static and runs only once at the start of experiments.

3) Extension Modules: Extension modules are other modules which exactly or loosely follow the manager-agent model, or build on it, to create further functionality that may be useful to some specific experimenter. We provide a few such modules, primarily for OAM (described next), which may be used or built upon by other user-experimenters. However, extension modules serve no critical functions in the operation of the mesh, and can be eliminated by a particular experimenter if they so desire.

a) OAM Modules: Apart from the large set of system modules, CentMesh software suite also provides following OAM modules which are necessary for Operations, Administrations and Maintenance.

Network Monitoring and Visualization: A separate network monitoring module (design similar to core modules) run constantly in background which collects large set of network status information and reports it to the controller. The purpose of designing such a monitoring module is to facilitate exhaustive information about every possible network event/state to the researchers while they are performing the experiments. The monitoring module is completely customizable which allows researchers to control what information they want to be reported and at what intervals. In its simplest form, researchers can specify a set of commands (e.g. iwlist, iwconfig etc.) to the monitoring manager which then periodically runs them on different mesh nodes using monitoring agents.

Apart from user customizable information collection, monitoring module contains a fixed set of basic routines which are run periodically to gather network state information. These include node status information, link interference information, topology information and current routing paths. A separate visualization program (Fig. 1b) is provided which runs on the controller. The visualization program extracts the information provided by monitoring manager and converts it into KML format. The KML files are then used to visualize the network status information on geographic maps using Google Earth. Such visualization helps the researchers to quickly understand and interpret the monitoring information from a remote client.

Testing Module: Once the users have developed their own core module and configuration, a set of common testing utilities can be used to perform the evaluation. Basic set of utilities like traffic generator, synchronization, logging etc. are provided to experimenters to assist them with the testing. Also, the researchers can customize the testing module to run any tests between mesh nodes as per the interest and can control/monitor them from the controller node.

III. Research Studies and Ongoing Extensions

CentMesh is designed to support various different kinds of research projects. As described before, research like new routing protocols, MAC protocols, channel assignment strategies, power control mechanisms, end-to-end congestion control etc. can be developed and tested in current CentMesh framework. Apart from this, CentMesh can also support security research at various layers together with mobility management and modeling. Cross-layer protocol and radio resource management can also be evaluated in the current software architecture. Some of the ongoing research projects include coarse-grain TDM scheduling, back-pressure medium access control and diverse routing.

To increase the usefulness and usability of CentMesh, we are working on extending it in several ways. In the next phase of deployment, we plan to install the mesh nodes on light poles in university campus. This will allow the researchers to perform the experiments in realistic outdoor environment. We also plan to equip certain nodes with SDR (software defined radio) and sensors that will allow researchers to accommodate heterogeneity in their experiments. In terms of CentMesh software, we soon plan to release the software suite under open source license (GNU GPL 2.0) to make it available to other researchers for use.

References

[1] ORBIT Lab. [Online]. Available: http://www.orbit-lab.org/
[2] Emulab. [Online]. Available: http://www.emulab.net/
[3] MIT Roofnet. [Online]. Available: http://pdos.csail.mit.edu/roofnet
[4] TFA Rice Wireless Mesh. [Online]. Available: http://tfa.rice.edu/
[5] UCSB MeshNet. [Online]. Available: http://moment.cs.ucsb.edu/meshnet/
[6] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04*. NY, USA: ACM.