CentMesh : Modular and Extensible Wireless Mesh Network Testbed

Jun Bum Lim, P. H. Pathak, M. Pandian, U. Patel, G. Deuskar, A. Danivasa, R. Dutta, M. Sichitiu
N.C.State University, Raleigh

Motivation

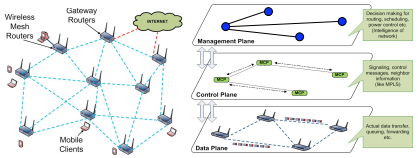
- ✓ Evaluation of protocols using wireless testbed in realistic environments is essential to understand their real-world applicability.
- ✓ Most of the early efforts of testbed designs have focused on simplifying the steps involved in day-to-day testbed operation and facilitating a certain amount of experimental repeatability.
- ✓ Developing software, controlling testbed resources, and running experiment in a wireless mesh testbed is a challenging task because of limited abstraction of the testbed environment, distributed operations in a multihop topology, and dependencies between software.

Requirements

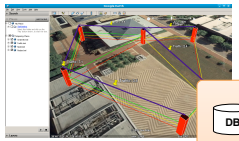
- ✓ **Loosely coupled system**: modification or failure of a process should not affect other processes, and an additional process should be easily inter-operable with the existing system.
- ✓ **Modular programming library**: software platform should provide a consistent set of programming interfaces for application/protocol developers and reduce the complexity of inter-process communication by hiding the underlying message processing.
- ✓ **Separation of common operations from core modules**: Common operations of protocols and network management should be separated from algorithm and protocol implementations to allow researchers to implement only the part of the stack they want to experiment with.
- ✓ **No dedicated backhaul network**: reliable remote access should be supported via the testbed network itself regardless of the operational status of the network.

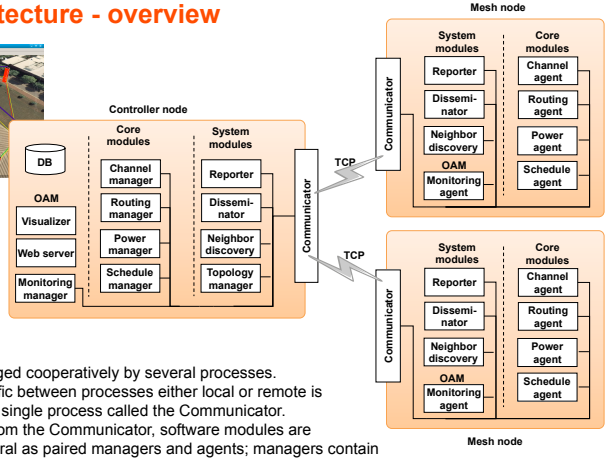
Introduction

- ✓ 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, control, and management planes.
- ✓ The modular structure of CentMesh software allows users to plug-and-play various existing modules and add new modules.



Software Architecture - overview

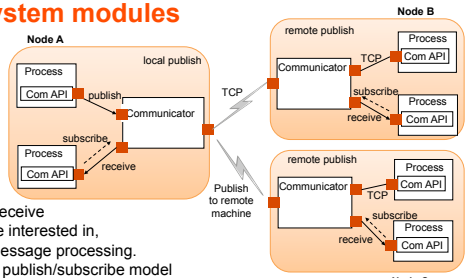




- ✓ The testbed is managed cooperatively by several processes.
- ✓ Control/signaling traffic between processes either local or remote is channeled through a single process called the Communicator.
- ✓ Based on the APIs from the Communicator, software modules are implemented in general as paired managers and agents; managers contain central intelligence/algorithm of the protocol, and agents perform actuating tasks based on manager's decision.

Software Architecture - system modules

- ✓ ***Communicator:***
  - A messaging system through which all the distributed processes can interact with each other in a loosely coupled manner.
  - Instead of client/server model, the Communicator uses publish/subscribe mechanism, which allows processes to send and receive messages based on the topic they are interested in, while hiding the detail of underlying message processing.
  - 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.
  - Common multihop operations such as unicast reporting and flooding are supported.

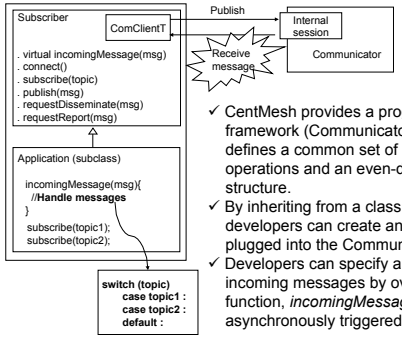


- ✓ ***Neighbor Discovery:***
  - Neighbor information is periodically detected.
  - All node report neighbor information to the controller, which then realize the entire network topology.
- ✓ ***System Crash and Recovery:***
  - Every mesh router has 3 kernel images and 3 respective root file systems – 1 for failsafe operations and 2 for experimentation.
  - Developers can modify any of 2 experimental images/file systems for their implementation.
  - Any software failures, hang-ups, or kernel crashes are handled using rebooting in failsafe mode.

Hardware Components

- ✓ CentMesh does not use any customized devices.
- ✓ A typical mesh node consists of off-the-shelf desktop computers.
- ✓ Each node contains upto 4 Atheros wireless cards, each connected on 4-to-1 miniPCI to PCI adapter.
- ✓ Each node runs Fedora distribution of Linux with Madwifi for wireless card.

Programming structure



- ✓ CentMesh provides a programming framework (Communicator API), which defines a common set of messaging operations and an even-driven programming structure.
- ✓ By inheriting from a class called *Subscriber*, developers can create an object that can be plugged into the Communicator.
- ✓ Developers can specify a handle for incoming messages by overriding a virtual function, *incomingMessage(msg)*, which is asynchronously triggered by *ComClientT*.

OAM (Operations, Administrations and Maintenance) modules

- ✓ **Network Monitoring and Visualization**:
  - The monitoring module is completely customizable; researchers can specify a set of commands to the monitoring manager.
  - Visualizer converts monitoring information into KML, which is used to visualize the network status on geographic maps using Google Earth.
- ✓ **Testing Module**:
  - Basic set of utilities like traffic generator, synchronization, logging etc. are provided to experimenters to assist them with the testing.
  - 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.

Software Architecture - core modules

- ✓ A module developed by the researcher to perform experiment.
- ✓ CentMesh provides a default channel assignment and routing protocol, which can be replaced by experimenters by their custom modules.
- ✓ **Routing:** controller node collects ETT (Expected Transmission Time) values of all the links in the network and computes routing paths.
- ✓ **Channel assignment:** channel manager uses greedy edge coloring algorithm on the network topology graph and determines the channel for each radio.

Research studies

- ✓ Based on the CentMesh, some of the ongoing research projects include:
  - Coarse-grain TDM scheduling
  - Back-pressure medium access control
  - Diverse routing
  - Joint channel assignment and routing