Architecture

Components 3

Network *3*

Heartbeat *3*

Command and Control *3*

Listener *3*

Mesh Model *4*

Monitor *4*

Diagrams 4

Legend *4*

Overview *4*

Monitor-Node *5*

MESH *6*

Design Decisions 6

Bson *6*

Qt Creator *6*

Components

The following are the major components of the system. These do not map directly onto the code base in the source directory. The last line for each component says precisely where the code for that component resides.

Network

Purpose	To route packets and transmit data between nodes.
Runs On	Each node of the MESH.
Description	Layer 1 and 2 of networking.
Code	*Code for the external MANET that is running (i.e., BATMAN or 80211s)*

Heartbeat

Purpose	To provide the monitor with the status and behavior of the network.
Runs On	Each node of the MESH.
Input	Reads network statistics and tables from the node on which it runs.
Output	Periodically sends packets to the listener containing the info it read.
Code	MeshKit/src/heartbeat-sender/heartbeat.py

Command and Control

Purpose	To respond to traceroute and other special requests from the monitor.
Runs On	Each node of the MESH.
Input	Listens for packets from the model and from other nodes.
Output	Sends packets to the listener and to other nodes.
Code	MeshKit/src/heartbeat-sender/CCResponse.py

Listener

Purpose	To listen for packets from the mesh.
Runs On	The Monitor-Node node.
Input	To listen for heartbeat packets and command and control packets.

Output	Calls function to pass received information to the monitor.
Code	MeshKit/src/mesh-model/HeartBeatListener.* MeshKit/src/mesh-model/MKControl.*

Mesh Model

Purpose	To consolidate the information received by the listener.
Runs On	The Monitor-Node node.
Input	A function call by listener with a packet from the MESH.
Output	Updates the MESH model.
Code	MeshKit/src/mesh-model/*(all files except those used by the Listener)*

Monitor

Purpose	To show the behavior of the MESH to the user.
Runs On	The Monitor-Node node.
Input	Receives data by calling the functions of the model.
Output	Behavior of the MESH to the user.
Code	MeshKit/src/gui/Monitor/

Diagrams

Legend

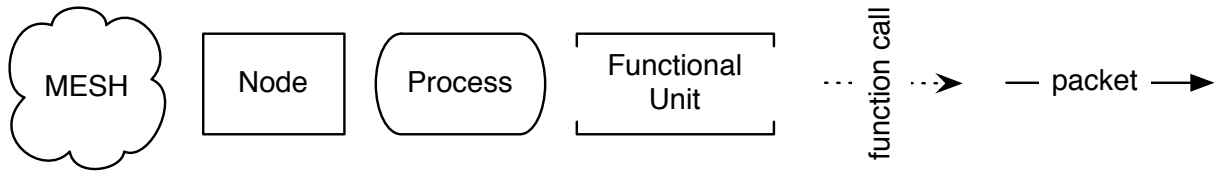

Figure 1: Legend

Overview

Figure 2 distinguishes the Monitor-Node from the rest of the nodes in the MESH. C&C (Command and Control) request packets are sent from the Monitor-Node to the MESH to request special features to be performed by the C&C program running on the nodes of the MESH. Upon receiving a C&C request packet, the C&C program performs the requested task and sends a the C&C response packet back to the Monitor

The Monitor-Node is the physical piece of hardware that is running the Monitor software. There may be more than one Monitor-Node in a network, but for simplicity we only show one in the Figure 2. Furthermore, the Monitor-Node may be a participating node in the MESH in that it is sending heartbeat packets, or it may be only observing network activity without sending heartbeat packets. Figure 2 shows the Monitor-Node as an non-participating network observer.

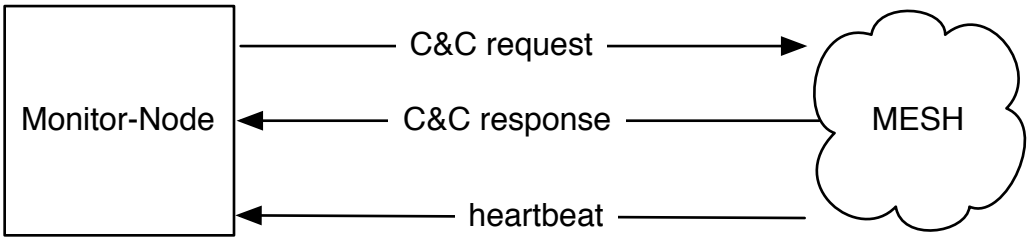

Figure 2: Overview

Monitor-Node

The MESH is omitted from this Figure 3. The Monitor-Node runs the MESH Observer Process which contains three functional units: Monitor, Model, and Listener. These units communicate with the MESH in the manner described in the *Overview* section. The Monitor calls the functions in the interface provided by the Model to get the behavior of the MESH. The Listener calls functions in the interface provided by the Model to give it new raw data (heartbeat and C&C packets) with which to update itself.

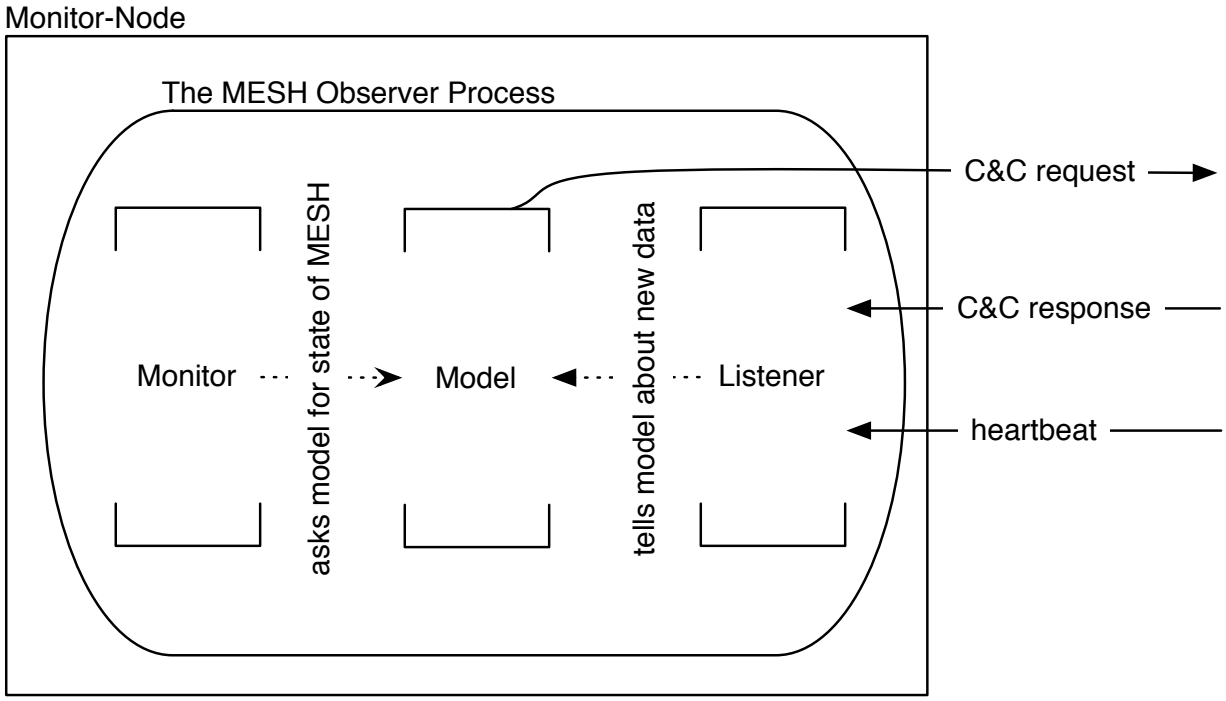

Figure 3: Monitor-Node

MESH

The Monitor-Node is omitted from Figure 4. There may be more nodes in the MESH than the two that are shown.

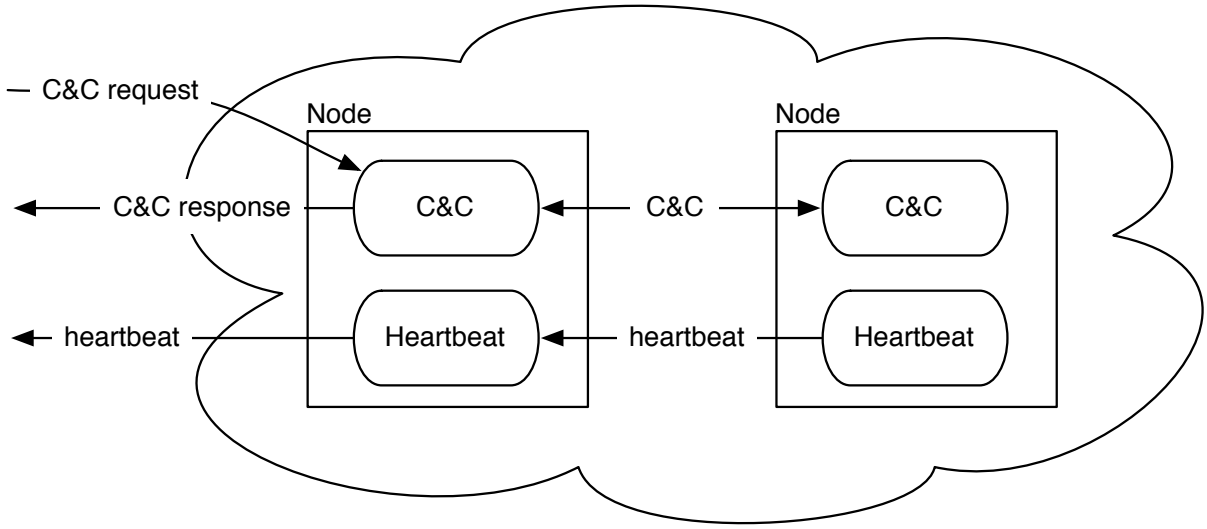

Figure 4: MESH

Design Decisions

Bson

Bson was chosen in order to make packets compatible between C++ and Python encoders and decoders. However, it turned out that the C++ and Python encoders and decoders didn't read the Bson packets the same way, so Bson didn't solve the compatibility problem. Since it was too expensive to remove the Bson from the project, the work around was to call Python scripts from C++ code.

Bson was chosen over Json because Bson allows access to any data element whereas Json requires that each element be iterated over in order. However, this advantage was not needed because we ended up iterating over each element in order anyway. Additionally, Bson was a poor choice because it has worse performance than Json.

Qt Creator

Qt Creator was chosen because it provides

- a GUI framework,
- IDE which includes a visual UI builder,
- and an integrated debugger.

Although Qt Creator's framework had both C++ and Python versions, C++ was chosen because produces faster executables, was more stable, and was better documented than the Python version of the framework.