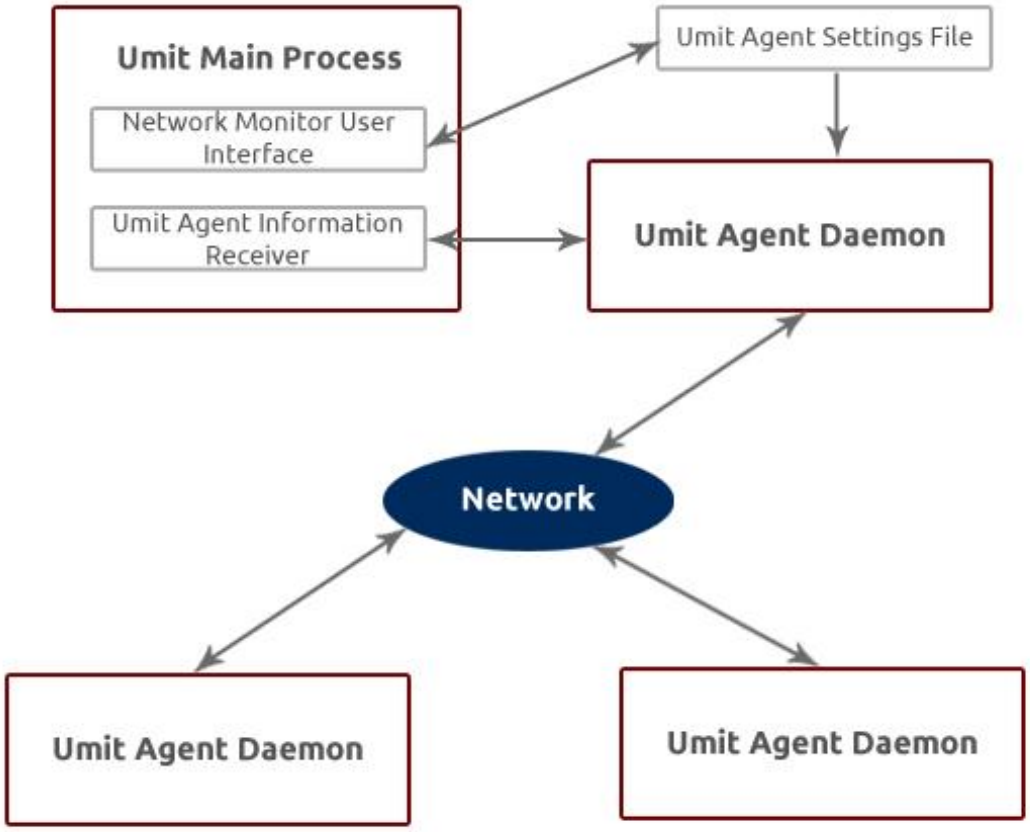
Umit Google Summer of Code 2011 Proposal
Network Inventory: Next Generation

I. General view of the implementation.



The basic idea of the project is to enhance the Network Inventory trough the possibility of receiving asynchronous messages from hosts in the network, besides the already implemented scheduled scans. For this, I propose the following components:

- The **Network Monitor User Interface**: This will basically be the UI which will show the results. There are multiple versions of doing this, but most likely it will consist of a gtk.TreeView showing the recent events. Also, statistics should be shown (this will be dependent on the user configuration). A configuration dialog will be available to the user who will be able to set things like: what statistics to track, what statistics to show, logging, adding remote agents, general configuration needed for the daemon (some will be added as needed when implementing). These configurations will be saved in the **Umit Agent Settings File** (something like

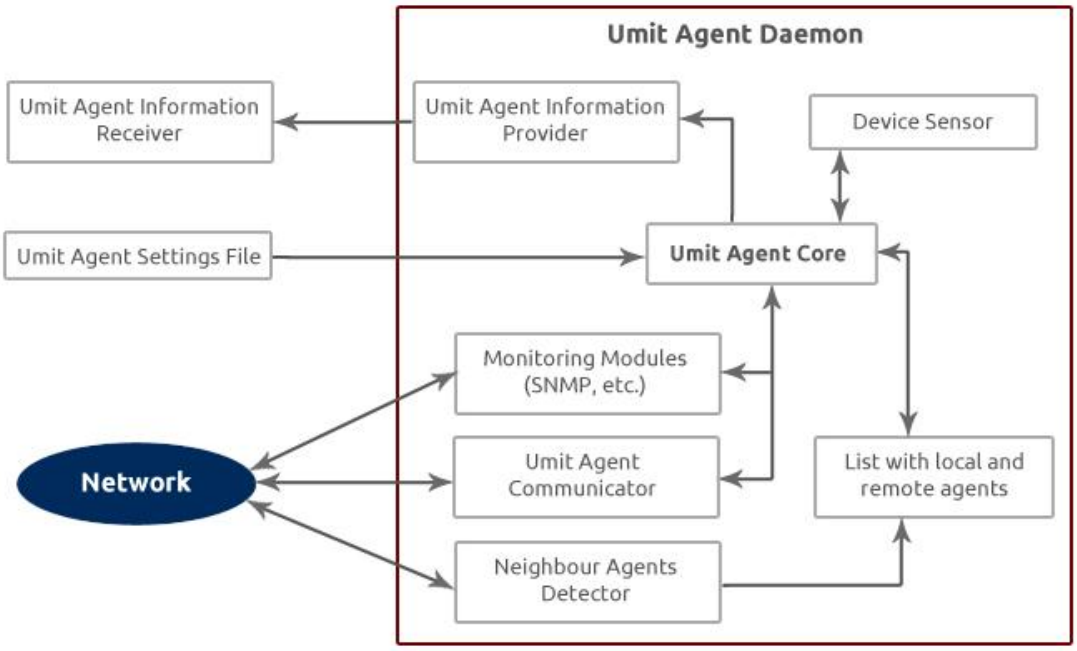
/etc/umit-agent/settings.conf on Linux) and will be read up by the Umit Agent Daemon.

- The **Umit Agent Information Receiver** will receive information from the Agent Daemon. Depending on the Agent location, some sort of IPC mechanisms will be used. Though in the above diagram it isn't obvious, but the Agent can be installed on another host. In that case, the user needs to configure the needed addressing from the configuration window. If the user configured a local daemon and it isn't installed, then it will be installed at this step. If there is one running he should inform him that Umit has launched so the daemon can send him messages.

- The **Umit Agent Daemon** is the component which will do most of the work, but it's most important task will be to listen for network events and communicate it to the Umit Agent Information Receiver. Also, it will send local events to other agents. It will be able to communicate with remote agents if configured so by the user. More details about the Umit Agent architecture in the next section.

II. The Umit Agent Daemon

The architecture of the Agent can be seen bellow. Its components will be discussed on the next page.



- The **Device Sensor** will listen to events on the local machine (like how much bandwidth each application uses, processor load or anything that will be decided to be useful information). It will send the information to the Core which will send it trough the Communicator to the other agents which subscribed for this information or trough the Information Provider to the Information Receiver (a component in the main Umit process – can be seen in the first section). To what events the device sensor listens should be configurable by the user in the UI (mentioned in the first step) and loaded by the agent trough the settings file.
- The **list with local and remote agents** while it isn't a functional component, it's an important part of the agent as it keeps track of the discovered agents (received from the Neighbor Agents Detector) or the remote ones configured trough the settings file (which will be stored here by the Core). It must save which of these agents also want information from the current one.
- The **Neighbor Agents Detector** is responsible, as its name says, to detect neighbor agents and save that information in the list mentioned above.
- The **Monitoring Modules** offer additional functionality to the agent as it allows him to be extended with other listeners (like listening to SNMP traps). The information received from these modules will be sent to the Core which will further send them to the Information Provider and the Communicator as configured. These modules will run in their own thread and signal the core when information is received. They must implement a given UmitMonitoringModule interface and format the received message in a local one which can be further passed.
- The **Umit Agent Communicator** will listen to messages from other agents and also send them messages as needed. There will be 2 threads running here: one for sending and one for receiving.
- The **Umit Agent Information Provider** has the only task to inform the Umit main process (more exactly the Umit Agent Information Receiver component) of the gathered information as configured.
- The **Umit Agent Core** loads the settings from the file and manages all the above components according to those settings. It should also implement security measures to prevent flooding with async messages.