Jordan Davis  
Group 25  
Smart Home Intercom System

Assigned items: Mesh Network Protocol, User Interface technologies, Operating systems

Goal: Implementing a certain mesh network protocol on a single board computer. Determining which operating system best fits the scope of our project and finding out which user interface technology gives us the best design options possible.

Network Protocol (Need to prioritize topics and talk about Netsukuku)

The three best technologies for developing a network on are Better Approach to Mobile Ad-Hoc Networking (B.A.T.M.A.N.), Netsukuku and Qt’s built-in network programming module (QtNetwork). QtNetwork allows for creation of both broadcast and receiver systems along with having examples in their documentation. The primary potential issue for QtNetwork is that the broadcast and receiver examples are not fully documented; however, it appears to be fairly optimized for something we would want to do. Even in a mesh network, a unit will need the ability to broadcast and receive. There is limited potential for running QtNetwork on top of BATMAN or Netsukuku. QtNetwork relies heavily on built-in C++ TCP/IP packages while BATMAN and Netsukuku remove the need for IP addresses and create their own headers for packets, so it might require heavy changes. As stated in the documentation for BATMAN advanced (batman-adv), “Batman-adv operates entirely on ISO/OSI Layer 2 – not only the routing information is transported using raw ethernet frames but also the data traffic is handled by batman-adv.” Batman-adv has a quick start guide available online specifically for Debian systems, should we choose Raspbian as our operating system. The primary issue with mesh networks (both netsukuku and BATMAN) for this project in general is that they are optimized to be able to hop between nodes with minimal packet loss. Which is great as a whole; however, the reliability issues that could come from streaming over multiple hops will require more of a star topology. (The winner here is Qt with the potential for testing it over batman-adv.)

<https://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept>

<https://www.open-mesh.org/projects/batman-adv/wiki/Debian_batman-adv_AutoStartup>

<https://www.open-mesh.org/projects/batman-adv/wiki/Quick-start-guide>

<https://www.open-mesh.org/projects/batman-adv/wiki/Wiki>

<https://www.open-mesh.org/projects/batman-adv/wiki>

<https://www.open-mesh.org/projects/batmand/wiki/VisualizeMesh>

<https://downloads.open-mesh.org/batman/manpages/batmand.8.html>

<https://www.open-mesh.org/projects/batmand/wiki/AnnouncingNetworks>

<http://netsukuku.freaknet.org/>

<http://netsukuku.freaknet.org/main_doc/ntk_rfc/Ntk_p2p_over_ntk.pdf>

<http://netsukuku.freaknet.org/main_doc/topology.pdf>

<https://lab.dyne.org/Netsukuku>

<https://lab.dyne.org/NewNetsukukuOrg>

<https://lab.dyne.org/NewNetsukukuOrgWhatIsNetsukuku>

<https://wiki.gnome.org/Projects/Vala>

<https://blogs.gnome.org/despinosa/2016/09/30/should-we-drop-vala/>

<http://doc.qt.io/qt-4.8/network-programming.html>

<http://doc.qt.io/qt-4.8/qtmultimedia-module.html>

<http://doc.qt.io/qt-4.8/examples-network.html>

U.I. (Figure out end-goal for U.I. from group, one package might be best.)

Developing a user interface will require something that can run on Linux or Windows IoT as well as be somewhat cross platform. GTK+ has been around for a long time and has support for C++, Python and Vala. There are extensive examples for GTK+ online as well as it serving as the backbone for Qt. Qt is supported on Linux and Windows with guides and examples for a very expansive number of programs. Qt even has a Raspberry Pi guide. Kivy is developed in Python and has an expansive available API. There are enough Kivy examples to easily get started with Kivy. Qt could also serve as our networking backbone on Raspbian or Windows IoT. All off free licenses, with Kivy and GTK+ not having a proprietary version. Qt 4.8 is free, with the latest versions requiring a license. Qt has a built in multimedia platform that could potentially be used to stream video from unit to unit. While this could boil down to Python v C++, Qt has obvious advantages over both as it has expansive examples and can perform tasks that would require additional libraries in Kivy and GTK+.

<http://doc.qt.io/qt-4.8/opensourceedition.html>

<http://doc.qt.io/qt-4.8/supported-platforms.html>

<http://wiki.qt.io/Raspberry_Pi_Beginners_Guide>

<http://doc.qt.io/qt-4.8/linuxx11-support.html>

<http://doc.qt.io/qt-4.8/windows-support.html>

<http://doc.qt.io/qt-4.8/examples-multimedia.html>

<https://kivy.org/#home>

<https://www.qt.io/qt-in-iot/>

<https://kivy.org/docs/gettingstarted/intro.html>

<https://kivy.org/#home>

<https://kivy.org/docs/api-kivy.html>

<https://www.gtk.org/>

<https://www.gtk.org/features.php>

<https://developer.gnome.org/references>

Operating System

While Raspbian is the typical operating system for Raspberry Pi devices, it is potentially the most heavy of the OS options. Windows IoT offers limited support and it would require USB devices as it will not support the Raspberry Pi camera. Despite this, Windows IoT would offer a better development environment and allow for sensor data interpretation to be faster. Snappy Ubuntu is the next potential version to be used. It is lightweight and allows for downloading of snaps as a user needs. This would mean our software package would always be up to date; however, it would require extra start time. Raspbian can be customized and locked down to startup specifically into our software. Qt can work on any of these operating systems, while Batman-adv is limited to Linux systems.

<https://www.raspberrypistarterkits.com/products/operating-systems-raspberry-pi/>

<http://www.raspbian.org/>

Windows 10 IoT:

<https://www.microsoft.com/en-us/download/details.aspx?id=53360>

<https://developer.microsoft.com/en-us/windows/iot>

<https://wiki.archlinux.org/index.php/qt>

<https://archlinuxarm.org/>

<https://www.ubuntu.com/desktop/snappy>

<https://docs.snapcraft.io/snaps/intro>