


CS486C – Senior Capstone Design in Computer Science

Project Description

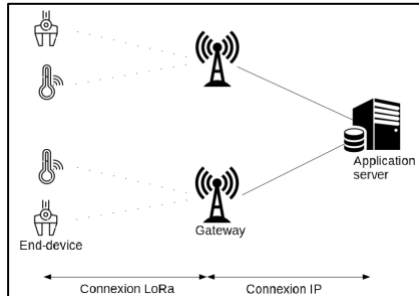
Project Title: Mobile Crowdsensing Framework over Low-power Wide Area Networks	
Sponsor Information:  CANIS LAB COMMUNITY-AWARE NETWORKS & INFORMATION SYSTEMS	Dr. Morgan Vigil-Hayes, Assistant Professor Community Aware Networks and Information Systems (CANIS) Lab School of Informatics, Computing, and Cyber Systems Northern Arizona University morgan.vigil-hayes@nau.edu cell: (209) 617-5593 office: (928) 523-4863

Project Overview:

81% of all U.S. adults own a smartphone. Smartphone penetration has given rise to a paradigm called mobile crowdsourcing (MCS), where a large group of individuals with mobile devices capable of sensing and computing collectively share data which is then used to measure, map, analyze, estimate or infer (predict) any processes of common interest. Just a few examples of the applications that leverage include:

- **Disaster Response.** After the 2010 Haitian Earthquake, the [Ushahidi crowdmapping](#) platform was used to map more than 3,500 events in close to real time, including breakout of fires and people trapped under buildings. This helped responders develop a more optimized approach, which resulted in more lives being saved.
- **Citizen Science.** The [iNaturalist app](#) allows amateur scientists to take pictures and record when and where they see various plants and animals, helping create a wide record of biodiversity and animal mobility. So far, the app has recorded observations of over 214,000 different species.
- **Data Activism.** The [Hollaback! app](#) was designed to allow women to report geolocated incidents of street harassment and sexual assault so that there would be a body of data to inform policymakers as well as records of their aggressors in case of escalation. [OpenCellID](#) was created so that users could collect signal strength and speedtest data in order to hold network providers and the Federal Communications Commission accountable to reported levels of cellular coverage and service level agreements.

While many MCS applications are clearly impactful for society at large, the infrastructure required for them to provide real-time sensory feedback is not available everywhere. Many rural and tribal communities still lack the ubiquitous LTE availability that is necessary to upload data in real-time from anywhere. Similarly, wilderness monitoring applications can only leverage opportunistic or delay tolerant MCS data collection due to lack of connectivity. Cellular networks can also be incapacitated during disaster events, rendering MCS approaches to rapid response unusable over a wide area. Moreover, even when wide area network connectivity is available, it is not always amenable to sensor data transmission. In urban centers, wide area networks are often congested and overloaded, and adding MCS data traffic to these networks can exacerbate these issues. Finally, for all MCS applications, power restrictions critically inhibit scale. This is largely due to the power required to transmit data over cellular networks. What is needed is a way to support MCS applications that relies on a new breed of networks as *an alternative to* existing cell networks.



The CANIS Lab seeks to (1) *empower communities in remote environments to collect data and generate content*; and (2) *measure network connectivity in areas that are underserved by current infrastructures*. Critical to these goals are tools that enable a more scalable approach to MCS. One of the ways we envision addressing the connectivity and power restrictions associated with MCS is by enabling smartphones to transmit MCS data over low-powered wide area networks (LPWANs). LPWANs are an emerging type of network technology that allow for low-bandwidth data transmission over a very wide area (10-15 km). Importantly, they only require a very small amount of power for data transmission across these wide areas. LPWANs have been

developed to support Internet of Things applications; thus, they are a promising connectivity option for the transmission of sensor data. In particular, **LoRaWAN** is a promising LPWAN technology given its utilization of unlicensed frequency bands and open standards. Despite these promising features, no software has been developed to allow apps to easily leverage LoRaWAN. ***The aim of this project is to address this shortcoming: we envision the development of an open source software platform that will allow apps to easily transmit various types of MCS data to existing MCS services and receive centralized control messages over LoRaWAN via Bluetooth.***

Some features will include:

- An accessible API that allows developers to easily leverage the library
- Ability to transmit data over LoRaWAN
- Ability to receiver control instructions over LoRaWAN
- Proof of concept app that is able to collect data using both participatory and opportunistic data collection models and contribute this data to existing MCS projects
- Ability to measure basic performance metrics of data flow, including end-to-end latency, throughput, and packet loss

It is anticipated that the development of this LoRaWAN interface library will be extremely impactful for computer scientists, domain scientists, and communities. By enabling any mobile application to use LoRaWAN instead of a cellular data connection, this project will serve as a foundational building block for an entire new generation of MCS applications that can operate where cellular networks are missing or unreliable. By being able to evaluate the performance of MCS data flows over LPWANs, computer scientists can better optimize LPWANs for a wider variety of MCS applications as well as design the next generation of smartphones for smarter and more connected communities.

Knowledge, skills, and expertise required for this project:

- Android development skills. The prototype will be done for Android, but keeping the design general enough to port to iOS later.
- Basic web application development skills. A simple web app “back end” will need to be built to serve as the “other end” of the data connection, i.e., a place for LoRa-connected devices to deposit their data.

Equipment Requirements:

- There should be no software required other than a development platform and software/tools freely available online
- LoRaWAN Gateway hardware (Available via CANIS Lab)
- Serial-to-LoRa hardware or Bluetooth-to-LoRa hardware or WiFi-to-LoRa hardware
- Android phones (Available via CANIS Lab)

Software and other Deliverables:

- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.

- Android library that facilitates simple low-volume data being transmitted over LoRaWAN. When data includes image or video files, only the textual metadata is transmitted over LoRaWAN and the image/video data is posted opportunistically (when cellular or WiFi Internet access is available) via an update API hook.
- Application proxy server that is able to receive data that is incoming from LoRaWAN and act as a proxy between the Android device and the MCS project servers.
- Demo application that demonstrates library features, including:
 - Posting data via LoRaWAN and existing MCS project APIs
 - Data may include: GPS coordinates, timestamp, textual descriptions from the user, tags from the user, cellular signal strength in dBm, cellular base station information
 - Integration with multiple existing MCS project APIs, including
 - [OpenCellID Single Measurement API](#) (posting measurements)
 - Does not require a user interface
 - iNaturalist API (posting observations)
 - Requires a simple user interface as a form template where users can enter data and send it off by pressing a button
 - Receiving push notifications that there is a need for the device to collect data for an MCS project
 - In the case of OpenCellID, this should be handled without user interface
 - In the case of iNaturalist, this should be handled by alerting the user
 - Logging for calculation of performance measurements:
 - Application proxy server
 - Packet arrival from Android app and MCS servers (packet size, packet payload, packet ID, timestamp)
 - Packet transmission to Android app and MCS server (packet size, packet payload, packet ID, timestamp)
 - Android app
 - Packet arrival from Application proxy server (packet size, packet payload, packet ID, timestamp)
 - Packet transmission to Application proxy server (packet size, packet payload, packet ID, timestamp)