# Smart Phone Based Mobile Code Dissemination for Heterogeneous Wireless Sensor Networks

Omor Faruk
Dept. of Computer and Information Science
Indiana University Purdue University
Indianapolis, USA
ofaruk@iupui.edu

Xiaoyang Zhong
Dept. of Computer and Information Science
Indiana University Purdue University
Indianapolis, USA
xiaozhon@iupui.edu

Yao Liang
Dept. of Computer and Information Science
Indiana University Purdue University
Indianapolis, USA
yaoliang@iupui.edu

Xu Liang
Dept. of Civil and Environmental Engineering
University of Pittsburgh
Pittsburgh, USA
xuliang@pitt.edu

***Abstract*—Low-power Wireless Sensor Networks (WSNs) are being widely used in various outdoor applications including environmental monitoring, precision agriculture, and smart cities. WSN is a distributed network of sensor devices where the software running on the sensor devices defines how the devices should operate. In real-world WSN deployments, sensor node's software update is required to fix bugs and maintain optimal operation. In this paper, we present a novel mobile code dissemination tool based on smart phone running on Android Operating System for heterogeneous WSN reprogramming. Our implementation builds upon Mobile Deluge with new enhancements and more convenient mobile code dissemination tool in practice. We have evaluated our application performance on Android platform, and validated our mobile tool with a real-world outdoor low-power heterogeneous WSN deployment, demonstrating its practical merit.**

***Keywords—wireless sensor networks, Internet of Things, over-the-air reprogramming, heterogeneous networks, energy efficiency, mobile tool.***

## I. INTRODUCTION

Low-power wireless sensor networks (WSNs) and Internet of Things (IoT) are being used extensively in various outdoor applications including environmental monitoring, precision agriculture, industrial monitoring/control, and smart cities. Like any other network infrastructure one important aspect of outdoor WSNs/IoT is the regular maintenance that includes replacing depleting batteries, fixing bugs, and updating and reprogramming the applications running on sensor nodes. For example, it is very common for monitoring WSN/IoT deployments where new physical variables are needed to be added in the monitoring process or an updated function is needed for the application functionality. Manual reprogramming motes one by one to change their code is not only very time consuming, but also, in some cases, impossible since motes are sometimes deployed in harsh environments which are not readily accessible. Hence the idea of remote reprogramming WSNs/IoT comes into picture where motes can be reprogrammed over the air without physically accessing them. Many approaches have been proposed in this regard (e.g., [1] – [9]).

Most existing approaches of over-the-air reprogramming disseminate the new application code image in the entire deployed WSN/IoT. However, this approach is not applicable for any heterogeneous WSN/IoT composed of different mote hardware platforms (e.g., MICAz, IRIS, TelosB). Furthermore, real-world outdoor WSN deployments (e.g., [10][11]), usually powered by battery, regularly run over low power links to conserve the battery energy, such as the typical low-power-listening (LPL) mode in TinyOS [12]. As sleep intervals in LPL mode largely extend per-packet delivery time, LPL mode is not suitable for over-the-air reprogramming. This is because that the bulk code image dissemination involved in mote reprogramming results in dramatic performance degrades of the existing approaches. ROLP [9] modifies the three-way handshaking scheme in Deluge [1] to overcome the issue of reprograming WSNs operating over low power links. By exchanging of augmented Deluge control packets, it tries to synchronize the low power settings in a neighborhood. If there are bulk data to be sent, both sender and receiver nodes wake up to always-on states by setting sleep intervals to zero; the nodes then go back to LPL mode when the transmission is finished. However, ROLP still fails for heterogeneous WSN reprogramming.

To address the challenges, MobileDeluge (MD), a novel tool of mobile code dissemination for heterogeneous WSNs/IoT is presented in [13][14]. MD is designed in such a way that can work effectively with heterogeneous motes, and also overcome the LPL mode problem by creating a separate channel from the data communication for over-the-air reprogramming. MD works with a mobile base station node (referred to as MobileBase) which can store multiple code images of different platform types. MD then can connect to multiple nodes with selected platform type as selected by the user, switch their channels to a pre-defined channel, and disable LPL for efficient reprogramming. However, one drawback of the current MD is the adoption of a laptop computer as the platform of its mobile gateway for all over-the-air reprogramming operations. In light of this, a more convenient way would be to use a same powerful device but smaller in size and lighter in weight. Handheld mobile devices such as smartphones, would be ideal in such a situation. Along with the advent of technology smartphones possess significant computation power to do complex tasks, which are being used in variety of interesting mobile applications including health care equipment monitoring, smart home solutions, portable hotspot and others. To this end, we design and implement a smartphone-based MobileDeluge (sMD) system, where the new mobile gateway is developed based on smartphone platform with an Android operating

This is the author's manuscript of the article published in final edited form as:
Faruk, O., Zhong, X., Liang, Y., & Liang, X. (2019). Smart Phone Based Mobile Code Dissemination for Heterogeneous Wireless Sensor Networks. 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), 1096–1102. https://doi.org/10.1109/UEMCON47517.2019.8993104

Fig. 1. Smartphone-based MobileDeluge (sMD), a mobile code dissemination tool for heterogeneous WSNs/IoT.

system. Aimed to extend the previous work of MB [13][14], the newly developed sMD is not only more convenient for conducting outdoor WSN/IoT over-the-air reprogramming, but also adds more functions to further ease the code dissemination operation and thus increase the reprogramming efficacy. Our sMD tool, as shown in Figure 1, has been tested and validated in both Lab setup and an outdoor heterogeneous and low-power WSN testbed for environmental monitoring. The remainder of the paper is organized as follows. Section II provides the background of Deluge and MD, and Android operating system (OS). Section III presents our sMB design and implementation. Section IV shows the performance evaluation of sMD. Section V provides the validation of sMD tool via a real-world outdoor heterogeneous and low-power WSN deployment. Finally, Section VI gives the conclusions and future work.

## II. Background

### A. Deluge and MobileDeluge

MobileDeluge is based on Deluge [1] which is the de facto standard code dissemination protocol in TinyOS for a multi-hop WSN. In Deluge, a code image is divided into a set of fixed-size pages, whereas each page consists of a number of packets. Deluge employs Drip protocol to disseminate packets over the entire network. Drip uses a Trickle [15] timer to rapidly disseminate small packets to the network. It broadcasts at a short timeout interval at the beginning. If no new data is detected in the last round, the interval is doubled, until the upper bound is reached; otherwise, the interval is reset to the shortest one. Second, a data object (i.e., code image) is distributed through the ADV-REQ-DATA three way handshaking mechanism to ensure the complete delivery. Deluge also uses a volume and block manager for handling erase and read/write operations of the data object in sensor nodes' external flash memory. Finally, a reprogram guard is used to verify whether the node is capable of rebuilding and loading the received new code image.

Due to its dissemination nature to the entire network, Deluge fails to work in heterogeneous WSNs. Moreover, even in homogeneous WSNs, the performance of Deluge is very poor with WSN deployments over low-power links, as Deluge was designed to operate over always-on links. Drip messages can also introduce sustaining interference to the WSN/IoT. Although the sending rate of Drip messages decreases quickly when the network enters into a stable state, the minimal rate (e.g., *1024* seconds by default) may still be comparable to the data collection rate in low data rate WSNs/IoT.

To address the node heterogeneity and the effectiveness for WSN reprogramming over low-power links as well as control the complexity, MobileDeluge attempts to overcome the above weaknesses of Deluge by pursuing the following design ideas [13]: 1) the protocol of Mobile Deluge only focuses on one-hop network reprogramming, where the reprogramming of a multi-hop network will be achieved by its mobility from one site to another site in the WSN deployment area; 2) it enables the retrieval of the platform self-description information of the nodes in a one-hop neighborhood of the MobileBase, to facilitate the selection of the target nodes of the same hardware platform type being reprogrammed at a time; and 3) it uses a different channel with LPL disabled for actual reprogramming, allowing the fast and efficient transmission of the new code image without the interference to the rest of the network.

### B. Android

Android is a popular OS platform for smartphones, which runs Linux kernel but many features and modifications are made in favor of devices that have lower memory and battery power. A distinct feature of Android is to force application developers to run lengthy commands on other threads than the User Interface (UI) thread. If UI thread is kept busy for certain amount of time, then ANR (Application Not Responding) dialog is shown to the user and the application terminates. Hence, Android provides various frameworks for managing the interaction between UI thread and user created threads. One such framework is called HaMeR, standing for Hander, Message and Runnable framework. The main UI thread receives a call back from the other thread once any required work is done in the worker thread. UI thread then can only update the UI components that the user interacts with.

## III. Design and Implementation

Mobile Deluge is aimed to address the two critical issues of over-the-air reprogramming in heterogeneous and low-power WSN/IoT [13]. To this end, Mobile Deluge protocol uses the basic broadcast scheme to establish the connection between the MobileBase and the target nodes, limiting its working range to a single hop. Its logic is split into two parts: the MobileBase side and the node side. The MobileBase acts as a bridge between the target nodes and the mobile gateway connected to it. It receives commands from the mobile gateway, and then broadcasts to the nodes. In addition to the regular Deluge commands which are directly processed by the standard Deluge logic, new commands, referred to as Mobile commands, are created in the Mobile Deluge protocol for handshaking communication between the MobileBase and the target nodes. Among Mobile commands, Connection and Abort are essential. To start a reprogramming cycle, MobileBase starts broadcasting a Connection command in the original channel to instruct target nodes to switch to a new channel and disable their default LPL behavior. Upon replies from the nodes without delay, the

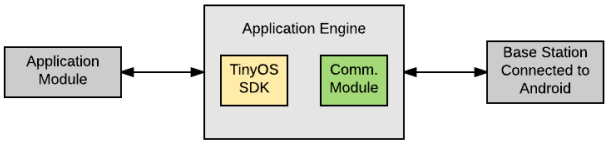

Fig. 2. An outline of the overall architecture of sMD system.

MobileBase itself switches to the new channel and disables its LPL. Once the MobileBase and the replied target nodes are all on the new channel and over always-on links, regular Deluge commands will be issued from MobileBase to conduct the reprogramming. On the other hand, from the perspective of sensor nodes, when a node receives a Mobile command packet, it first checks the target list in the command packet. If the node is not in the target list, the command is ignored. Otherwise, it responds according to the types of the command. If a Connection command is received, it sends a reply to the MobileBase and waits for an acknowledgment. If the reply packet is acknowledged by the MobileBase, it switches to the new channel and disables LPL, waiting for Deluge commands to perform the reprogramming; otherwise, if no acknowledgment is received after several retransmissions, it ignores the command. If a target node receives an Abort command, it switches to the original channel and enables LPL immediately.

MD uses a modified mechanism of the original Drip protocol called SimpleDrip, to disseminate the message to the target nodes. In this modified version, original Drips' many-to-many transmission pattern is simplified to one-to-many. Unlike the original Drip implementation where every node sends data periodically accordingly Trickle timer, it is only the mobile base station MobileBase to send the data packets. Hence, the packet only travels single hop. A linearly increasing timer is followed in the MobileBase that will broadcast the new packets to the network, while receiver nodes, on the other hand, do not reply.

The core of sMD is a new mobile gateway software development which runs on an Android based smartphone system. Figure 2 outlines the overall architecture of sMD system, which includes three major components: MD application engine, MD application module, and mobile base station MobileBase. Since Android smartphone itself does not support IEEE 802.15.4 wireless network protocol, a base station MobileBase (e.g., TelosB mote) is connected to the smartphone mobile gateway, similar to the original MD design. The sMD mobile gateway software is composed of gateway application engine and application module. The application engine is responsible for bi-directional communication between the MobileBase and the mobile gateway application module. We consider adopting TinyOS, a popular open-source operating system of tiny motes for heterogeneous low-power WSN/IoT deployments. Thus, TinyOS messages need be handled by our mobile gateway, which is achieved by using TinyOS SDK (Software Development Kit) inside the application engine.

### *A. Applicton Engine*

The application engine of our mobile gateway is developed by leveraging Android service framework. Application engine manages TinyOS SDK and communication module. It runs and terminates automatically without user intervention. TinyOS SDK module is responsible for parsing bytes to construct a meaningful TinyOS message. On the other hand, the communication module of the application engine is responsible for communicating with Android OS USB interface connected to MobileBase. Figure 3 illustrates the process of receiving data from the mobile base station MobileBase, whereas Figure 4 illustrates the process of sending data to the MobileBase.

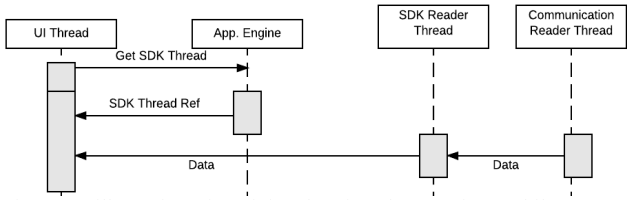

Fig. 3. An illustration of receiving data from base station MobileBase.

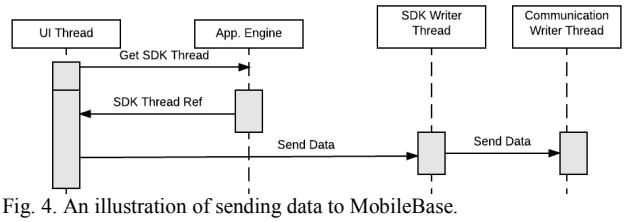

Fig. 4. An illustration of sending data to MobileBase.

### *B. TinyOS Serial Stack in Android*

As TinyOS SDK is not available for Android OS, in this work the original implementation of TinyOS SDK is ported with necessary modification for Android platform. TinyOS serial communication stack consist of five layers, which are from the highest layer to the lowest layer: application, message, packet, data link, and device abstraction. Device abstraction layer reads and writes bytes over USB serial connection, where hardware specific encapsulation bytes are stripped off, and the rest of raw bytes are then passed to data link layer. At data link layer, a meaningful frame is decoded using a HDLC (High-level Data Link Control) protocol. Packet layer performs decoding when it receives byte stream from data link layer and encoding when it receives packet from the layer above it. Decoding extracts payload from the serial packet encapsulation and sends the payload to the upper layer for further processing. Message layer is responsible for multiplexing different messages to appropriate receivers based on AM (Active Message) type, where AM type helps distinguish between different TinyOS messages. Applications register themselves with the message layer by an AM ID. Message layer maintains accounting of AM ID when senders register themselves. After receiving a packet, message layer checks whether a receiver exists; if yes, the packet is then forwarded to the receiver, and otherwise dropped.

### *C. Mobile Gateway Application Module*

We develop the sMD gateway application module, which is responsible for the subset reprogramming process in a single hop neighborhood. Currently our mobile gateway supports the following user operation commands, among which Detect, Detect Subset, Connect, and Abort correspond to new Mobile commands in Mobile Deluge protocol, whereas Disseminate and Stop correspond to the original Deluge commands.

- Detect: This operation discovers the nearby nodes in single-hop neighborhood and retrieves their battery voltage. This operation also retrieves the version number

and the platform information and node id of the nearby motes. Node(s) after receiving this command replies to the MobileBase with the requested information.

- Detect Subset: This operation is to detects any specific node(s) in a one-hop range.
- Connect: This operation connects to node(s) that the user selects using the mobile gateway by inputting node id(s). For single click this command checks the input node voltage and issues them the dissemination command if voltage is within the range. On the other hand, long pressing the button sends command unconditionally to the given nodes. Once connected, the target node(s) and MobileBase will be switched to a dissemination channel ready for code dissemination.
- Disseminate: This operation is to start the re-programming process. Once nodes receive this operation command they start to first reprogram themselves and then restart. Upon restarting, each individual node starts to run the new code image that was disseminated.
- Stop: Stop the dissemination of code image. The node(s) and MobileBase are still in the dissemination channel.
- Abort: This operation command is used to stop the process of code dissemination. The involved node(s) then returns to the regular operating channel from the dissemination channel and enables LPL mode.

An Android application when launched is governed by a single thread called the main thread or UI thread. When our mobile gateway application is first launched, it requests Android framework to create a service called UsbService. UsbService is used to communicate with the hardware interface connecting to the MobileBase via micro USB port. Once the mobile gateway software is started, UsbService probes for hardware device with specific PID (Product ID) and VID (Vendor ID), and then connects to the device by USBSerial library [16]. Once the device MobileBase is successfully connected to the smartphone host, multiple threads are started to communicate with the MobileBase.

### *D. Android User Interface*

sMD operation is backed by a simple UI with buttons to execute the required operation commands. Reply packets from remote motes are shown in the user interface screen. Error conditions and information are displayed as a toast message. Figure 5 shows a screenshot of the sMD mobile gateway running on Samsung Galaxy S3 phone, illustrating basic UI layout of the sMD. As we can see, the sMD app dashboard describes the program version, reprogramming channel and target device platform on the app bar. Program version is the version of the code image that will be disseminated by the sMD. Reprogramming channel can be changed from the application menu. Target device selects an appropriate code file for dissemination depending on the target node's hardware platform. Currently three platforms are supported: TelosB, MicaZ and Iris. Operation commands buttons are used to send the appropriate operations to the node(s). Finally, the bottom area is used to display the status or result of each operation.

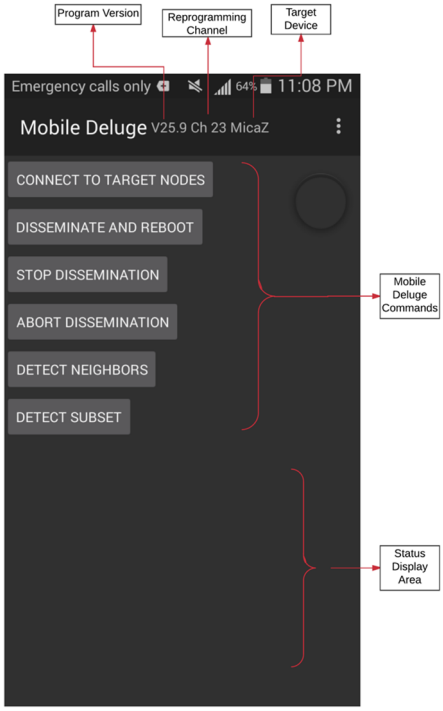


Fig. 5. UI layout of sMD mobile gateway.

## IV. Performance Evsluation

We discuss sMD performance evaluation in this section. The performance of our sMD can be considered in three aspects: smartphone UI performance, MD application performance, and Battery power performance.

### *A. IU Performance*

Rendering is the process of generating or drawing an image in the phone screen. Android OS updates the screen or activity in every 16ms which means that developers need to complete all necessary computation to update the screen after each user interaction in that 16ms time window. Any frame that is not ready within 16ms will not be received by the display hardware. These missed frames are called dropped frames. Updating


Fig. 6. sMD overdraw analysis by using UI overdraw indicators.

display requires both CPU and GPU hardware. One major performance problem of an application, due to improper UI display updating, would be a potential GPU overdraw. Figure 6 shows the screenshot taken to visualize any potential overdraw in Mobile Deluge application using Android GPU overdraw debug feature. If a portion of the display becomes red it means a significant amount of overdraw has occurred, wasting GPU and CPU processing power and time. As Figure 6 indicates, there is no red spots in the UI screen, which means no overdraw.

### *B. MD Application Performance*

The most important consideration in Android application development is not blocking the UI thread. If that condition is not satisfied, then Application Not Responding (ANR) error message is displayed on the phone screen, which is not a good experience to the user. If user hits the Ok button of ANR message, then the application process is killed by Android or if the user presses wait then the application keeps running but remains unresponsive to the user if significant amount of work is being done in the UI thread. Hence, Android provides easy to use API for developers to run long running computation or blocking IO calls in a separate thread. Also there are various communication mechanisms present to communicate back to the UI thread the computation result or data received from an IO device. Mobile Deluge app uses Android Handler and Message [17] to communicate back to the UI thread. The primary reason behind this communication is to update Activity resources such as handing click events, taking input from user and showing results in the screen which can only be done using the UI thread. While Android does provide any handy tool to analyze and debug ANR as well as misbehaving function, in our testing, we have used Android monitor to profile our application in runtime while reprogramming two TelosB motes. The testing result revealed that our UI thread is not blocked for any significant amount of time due to computation or blocking IO to the USB hardware connected.

### *C. Battery Power Performance*

Battery consumption is critical in mobile devices. In our case, it is even more important since the MobileBase is also powered by the Android phone. In order to measure the power consumed by our MD application we have used an application Power Tutor published in Android play store and measured the CPU usage in Milli Watts(mw). We have also compared our app power usage with popular Android apps like Google Chrome and YouTube. In the test, sMD conducted some over-the-air reprogramming and interacted with three motes. We have run several commands for about 10 minutes and recorded the amount of power consumed by the application. Test was done for YouTube video playback and Chrome web browsing for the same amount of time of 10 minutes respectively. Table I illustrates the comparison result. As we can see, our sMD application consumes less power than the other two popular applications. We note that the measured power consumption of MD did not include the power consumption by the MobileBase.

## V. REAL-WORLD WSN VALIDATION

An outdoor WSN testbed has been deployed and operated for years in a forested nature reserve at the Audubon Society of Western Pennsylvania (ASWP), USA. The ASWP WSN is a heterogeneous and low-power WSN testbed composed of three

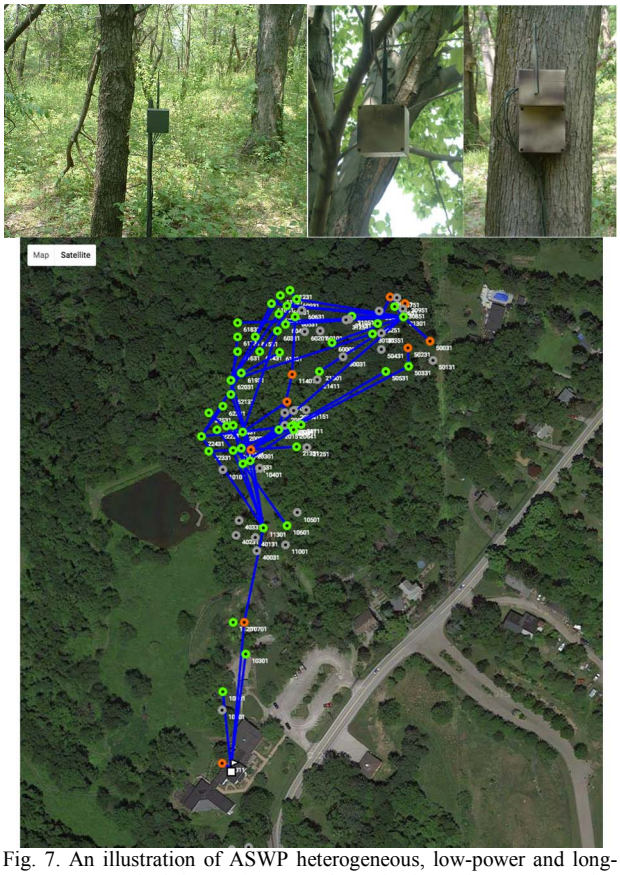


Fig. 7. An illustration of ASWP heterogeneous, low-power and long-term WSN testbed for environmental monitoring. The WSM routing topology shown in the lower diagram was monitored by the integrated network and data management system. Green dots indicated those nodes with good battery power, whereas orange dots indated the nodes with low battery power for operator's attention.

TABLE I. POWER CONSUMPTION COMPARISON

| Applications | Power Consumption (mW) |
|---|---|
| sMD | 130 |
| YouTube | 160 |
| Chrome | 157 |

different sensor node platforms MICAz, IRIS, and TelosB. ASWP testbed consists of about 100 sensor nodes and periodically collects ground-based measurement data for calibrating and validating scientific models in hydrology research [18]. Figure 7 presents some examples of node configurations deployed at the ASWP testbed. The ASWP WSN data collection application is developed based on TinyOS 2.1.2, which adopts the energy efficient and balanced routing protocol CTP+EER [19]. All sensor nodes run in the LPL mode, with an average sampling interval being 30 minutes. The WSN base station (i.e., sink node) is an IRIS mote with a permanent power supply connected to a WSN gateway computer [20], where the WSN gateway forwards the sensed data stream to the WSN data and network management system [21] via the Internet. The

WSN management system also enables to monitor the network operation and status in real time.

Operating a long-term outdoor WSN deployment in a harsh but accessible environment, such as the ASWP testbed requires frequent on-site network maintenance, including battery replacements, leaking enclosure fixes, and broken node replacements. Such on-site maintenance for long-term WSN operation makes our sMD a very convenient tool for the WSN over-the-air reprogramming which is needed for any WSN application updates and/or bug fixes. Our developed sMD is first tested in Lab setup and then validated in ASWP WSN testbed. sMD has added new features to mobile gateway operations compared to the previous MD tool. For example, it was possible before that a mote after receiving the reprogramming operation command sometimes does not reprogram itself and become unresponsive. The primary cause of this unresponsiveness is due to the battery voltage of the target mote is lower than the minimum requirement for the mote to be reprogrammed. Although this safety mechanism in Deluge could save the mote from corrupting its code image in the case of power failure, the mote became unresponsive and did not even respond to the Abort operation command. In such situation, the mote had to be restarted again to become accessible, which was very inconvenient and sometimes impossible without human interaction. In our sMD design and implementation, before issuing reprogram operation command, each target node's battery voltage is checked and filtered based on the specification of Deluge T2 [22]. Thus, the command is only sent to those devices that has met the minimum voltage criterion therefore solving the previous node's unresponsiveness problem. In addition, the sMD mobile gateway also provides a handy feature for advanced users to send the reprogram operation command even though the target node's battery voltage is lower than the standard threshold defined in the Deluge T2 documentation for testing and debugging purposes. Our ASWP WSN testing and validation show that sMD works well and more reliably. Furthermore, sMD mobile tool is significantly easier to carry around for WSN reprogramming.

## VI. Conclusions and Future Work

Long-term outdoor WSN/IoT deployments require application software updates and bug fixes beyond other regular maintenance tasks. Wireless sensor nodes in real-world outdoor applications are often deployed in harsh environments which make it unfeasible to perform reprogramming each sensor node manually. Most existing WSN reprogramming methods are not applicable for heterogeneous WSNs/IoT. Moreover, most existing reprogramming approaches are also particularly difficult for low-power WSN/IoT deployments. Therefore, it is critical and urgent to address the challenges faced for heterogeneous and low-power WSN reprogramming. Our developed smart phone based Mobile Deluge, an Android application referred to as sMD, provides a new mobile code dissemination tool which enables an easy, efficient and practical over-the-air reprogramming system for heterogeneous and low-power WSN/IoT nodes in real-world outdoor deployments. Our implementation also further improves the previous MobileDeluge system by adding more features to the reprogramming application based on our experiences.

The current sMD tool can be further improved by integrating an on-site database on the smartphone to keep track of node reprogramming status in the WSN/IoT deployment. Also, the on-site reprogramming database integrated with sMD can be synchronized with the online network management database by using the mobile data networks. Our future work also includes to add secure communication features for our sMD for code dissemination. Currently all the packets are transmitted unencrypted and therefore vulnerable to potential network attacks.

## Acknowledgment

This work is supported in part by the U.S. National Science Foundation under grants CNS-1320132 and CNS-1319331 to IUPUI and the University of Pittsburgh, respectively.

## References

[1] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in Proceedings of SenSys, 2004.

[2] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," Technical report, UCLA, Los Angeles, CA, USA, 2003.

[3] S. S. Kulkarni and L. Wang, "MNP: multihop network reprogramming service for sensor networks," in Proceedings of ICDCS, 2005.

[4] L. Huang and S. Setia, "CORD: energy-efficient reliable bulk data dissemination in sensor networks," in Proceedings of INFOCOM, 2008.

[5] R. K. Panta, S. Bagchi, and S. P. Midkiff, "Zephyr: efficient incremental reprogramming of sensor nodes using function call indirections and difference computation," in Proceedings of the USENIX Annual Technical Conference, 2009.

[6] J. Jeong, D. Culler, "Scalable incremental network programming for multihop wireless sensors," in International Journal of Communications, Network & System Sciences, Vol. 6, Issue 1, pp. 37-51, January, 2013.

[7] B. Mazumder and J. O. Hallstrom, "An efficient code update solution for wireless sensor network reprogramming," in Proceedings of EMSOFT, 2013.

[8] J. Qiu, D. Li, H. Shi and L. Cui, "EasiLIR: lightweight incremental reprogramming for sensor networks," in International Journal of Distributed Sensor Networks, 2014.

[9] Y. Gao, C. Chen, X. Liu, J. Bu, W. Dong, and X. Xu, "Reprogramming over low power link layer in wireless sensor networks," in Proceedings of MASS, October, 2013.

[10] X. Mao, X. Miao, Y. He, X.Y. Li, and Y. Liu. "CitySee: Urban CO 2 monitoring with sensors." In INFOCOM, 2012 Proceedings IEEE, pp. 1611-1619. IEEE, 2012.

[11] M. Navarro, T. W. Davis, G. Villalba, Y. Li, X. Zhong, N. Erratt, X. Liang, and Y. Liang. "Towards long-term multi-hop WSN deployments for environmental monitoring: An experimental network evaluation." Journal of Sensor and Actuator Networks 3, no. 4 (2014): 297-330.

[12] TinyOS. [Online]. Available: http://www.tinyos.net

[13] X. Zhong, M. Navarro, G. Villalba, X. Liang, and Y. Liang, "MobileDeluge: Mobile Code Dissemination for Wireless Sensor Networks," Proc. 11th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS), pp. 363-370, Philadelphia, Pennsylvania, October 28-30, 2014.

[14] X. Zhong, M. Navarro, G. Villalba, X. Liang, and Y. Liang, "MobileDeluge: A Novel Mobile Code Dissemination Tool for WSNs," Proc. 11th IEEE MASS, pp. 537-538 (demo paper), Philadelphia, Pennsylvania, October 28-30, 2014.

[15] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in Proceedings of NSDI, 2004.

[16] F. Herranz. Usb serial controller for android, nov 2016.

[17] Google Inc. Handler, aug 2017.

[18] G. Villalba, F. Plaza, X. Zhong, T. W. Davis, M. Navarro, Y. Li, T. A. Slater, Y. Liang, and X. Liang. "A Networked Sensor System for the Analysis of Plot-Scale Hydrology." Sensors 17, no. 3 (2017): 636.

[19] M. Navarro, and Y. Liang. "Efficient and Balanced Routing in Energy-Constrained Wireless Sensor Networks for Data Collection." In EWSN, pp. 101-113. 2016.

[20] N. Erratt, and Y. Liang. "The design and implementation of a general WSN gateway for data collection." Wireless Communications and Networking Conference (WCNC), 2013 IEEE. IEEE, 2013.

[21] M. Navarro, D. Bhatnagar, and Y. Liang. "An integrated network and data management system for heterogeneous WSNs." In Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, pp. 819-824. IEEE, 2011.

[22] TinyOS Wiki. Deluge T2, nov 2010.