MobileDeluge: Mobile Code Dissemination for Wireless Sensor Networks

Xiaoyang Zhong*, Miguel Navarro*, German Villalba†, Xu Liang†, Yao Liang*
*Dept. of Computer and Information Science, Indiana University Purdue University, Indianapolis, USA
†Dept. of Civil and Environmental Engineering, Pittsburgh University, Pittsburgh, USA
{xiaozhon, mignavar, yliang}@cs.iupui.edu, {gev5, xuliang}@pitt.edu

***Abstract*— In this paper we present MobileDeluge, a general mobile network-reprogramming tool based on Deluge for wireless sensor networks (WSNs). MobileDeluge effectively addresses the weaknesses of Deluge and other traditional over-the-air reprogramming approaches for WSNs. It enables efficient code dissemination for *heterogeneous* WSN motes regularly operating over low-power links through a mobile base station. We evaluate the performance of MobileDeluge via both laboratory experiments and a real-world outdoor environmental WSN testbed. Results show that our proposed MobileDeluge leads to a significant improvement of the performance compared to the original Deluge for WSNs operating over low-power links.**

Keywords-network over-the-air reprogramming; heterogeneous WSNs; energy efficiency; long-term deployments; mobile tool

I. INTRODUCTION

Wireless sensor networks (WSNs) have gained increasing attention and interest from researchers for various environmental monitoring applications. As long-term deployments start being considered, network maintenance becomes a key issue. Among other tasks, such as replacing batteries and fixing broken nodes, network maintenance also involves reprogramming currently deployed nodes for updating and improving applications. Manually reprogramming sensor nodes is very cumbersome as it requires retrieving the nodes from their deployed locations. To address this problem, many approaches (e.g., [1] – [10]) have been proposed in the past years for supporting over-the-air programming (OAP) through wireless communications. Most of them have been thoroughly examined through simulations and laboratory experiments. However, real-world WSN deployments usually have some unique features which are very challenging to the existing OAP approaches.

First, heterogeneity becomes a common scenario in WSN deployments, where multiple node platforms (e.g., MICAz, IRIS, TelosB), sensors, and applications may coexist on the same WSN testbed. From this arises the need for point-to-point or subset reprogramming in WSNs. However, whereas very few studies (such as [2], [7]) support point-to-point/subset reprogramming, most existing approaches such as [1], [3] – [6], [8] – [10], disseminate the code image to all the nodes in the network. Such a dissemination approach simply fails for heterogeneous WSN deployments with multiple node platforms, where different code images are required for different subsets of nodes.

Secondly, real-world outdoor WSN deployments [15]–[17] usually work over low-power link layers for better energy efficiency, such as the typical low-power-listening (LPL) mode in TinyOS [13]. Sleep intervals in LPL mode largely extend per-packet delivery time. Since reprogramming usually involves bulk code image dissemination, the total delay significantly degrades the performance of the previous approaches ([1] – [9]). While the recent work of ROLP [10] addresses this problem by dynamically adjusting the sleep intervals during image dissemination, ROLP still disseminates the code image to the entire network, which fails to reprogram any heterogeneous WSN.

On the other hand, long-term outdoor WSN deployments would usually require periodic on-site maintenance visits (e.g., battery replacement and faulty node fixing) to keep the network operating in a healthy and sustainable manner [15]. In view of this, we take a new approach to simultaneously address both challenges described above. We introduce a novel concept of mobile code dissemination, and present MobileDeluge, a general mobile network-reprogramming tool based on Deluge [1]. Equipped with a gateway laptop and a base station, as shown in Fig. 1, MobileDeluge is a hand-held code dissemination tool for outdoor WSN deployments over low-power links. It enables wireless reprogramming of WSN nodes in harsh but accessible environments within a one-hop neighborhood with respect to the hand-held Deluge base station. MobileDeluge creates a control service to coordinate the mobile Deluge base station and the target sensor nodes within the neighborhood of the mobile Deluge base station for code dissemination. The key idea is to establish an instant connection between the mobile Deluge base station and its target sensor nodes within the neighborhood, where the target nodes are to be updated with the same new code image. Once the connection is established, the target nodes are asked to switch to a different channel and disable LPL so that they can be reprogrammed efficiently. Since MobileDeluge can be brought close to the target nodes when reprogramming is needed, the reprogramming is limited to a single-hop neighborhood. In this way, MobileDeluge enables a significant amount of energy savings at intermediate nodes compared to traditional multi-hop code dissemination approaches.

The main contributions of this work include the following:

- We introduce a novel concept of mobile code dissemination for WSNs, which is particularly suitable

978-1-4799-6036-1/14 $31.00 © 2014 IEEE
DOI 10.1109/MASS.2014.60

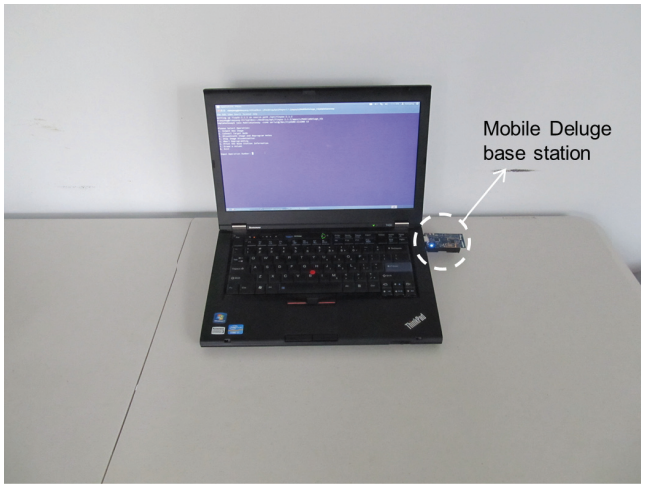


Figure 1. MobileDeluge, a hand-held mobile code dissemination tool.

for long-term outdoor WSN deployments in harsh but still accessible environments;

- We develop a mobile code dissemination tool system based on Deluge for efficient reprogramming heterogeneous WSNs that regularly operate over low-power links, simultaneously addressing both reprogramming challenges of heterogeneous WSNs and WSNs over low-power links;
- We evaluate and validate our proposed MobleDeluge through laboratory experiments and reprogramming a real-world outdoor WSN testbed.

The remainder of the paper is organized as follows. Section II describes the related work. Section III discusses our long-term outdoor WSN deployment. The design of MobileDeluge is presented in Section IV. Section V shows the evaluation results. Finally, Section VI concludes the paper.

II. RELATED WORK

Earlier network programming protocols usually distribute the entire new code image to the network [1 – 5]. Crossbow Network Programming (XNP) [2] was known to be the first network reprogramming protocol designed for WSNs. It operates with TinyOS [13], disseminating the whole code image to the nodes within a single hop network. Multi-hop Over-the-Air Programming (MOAP) [3] supports multi-hop network programming by employing a neighborhood-by-neighborhood transport mechanism called Ripple to distribute the new code to the whole network, and a simple sliding window method to track and manage retransmissions. Multi-hop Network Programming (MNP) [4] presents an efficient code dissemination mechanism by reducing the message collision and hidden terminal problem, attempting to guarantee that the node with the most impact in a neighborhood is selected to be the only source that transmits the new program. Deluge [1] is the *de facto* standard code dissemination protocol in TinyOS. It uses Drip protocol to control the code dissemination process. In Deluge, the code image is divided into a set of fixed-size pages, enabling spatial multiplexing so that large data objects are efficiently disseminated over a multi-hop network. CORD [5] delivers the code image to a subset of nodes called core nodes, and then the core nodes act as the source and disseminate the code image to their neighbors.

More recently, research efforts focus on incremental reprogramming approaches to reduce the transmitted data size, hence saving energy. Such approaches follow a common pattern of computing the difference between old and new code images, transferring this difference, and locally rebuilding the new program at the nodes.

Zephyr [6] performs a byte-level comparison between the old and new program binaries using an optimized version of the Rsync algorithm [11]. To reduce the size of the delta between the old and new programs, it applies application-level code modification, mainly in the function call indirection, for compensating the effects of function shifts caused by program modification. Different from Zephyr, the work of [7] uses block-level Rsync comparison algorithm to compute the differences between the old and new code images. As for dissemination, it uses BLIP IPv6 stack as the under layer routing protocol in supporting point-to-point multi-hop code distribution. At the node side, it uses a Deluge-like volume management to rebuild the new program. In [8], the longest common subsequence between old and new code images is computed using Hirschberg's algorithm [12] at a byte level. It builds an edit map specifying the edit sequence required by a node to transform the running program into a new program. It further applies a heuristic-based optimization strategy to encode the edit map to reduce the transmitted data size. EasiLIR [9] presents an energy efficient incremental wireless reprogramming scheme, which avoids read/write operations on nonvolatile memory (e.g., external flash memory) as much as possible. It applies *in situ* modification which directly modifies the binary code stored in memory to create the new program without entirely rebuilding the program. In case of large modifications that may break the program time constraint, it also applied a lightweight segmented rebuilding for directly creating a new image in memory.

To efficiently reprogram WSNs operating over low power links, ROLP [10] modifies the three-way handshaking scheme in Deluge. The idea is to synchronize the low power settings in a neighborhood through the exchanging of augmented Deluge control packets. When there are data to be sent, both sender and receiver nodes wake up to always-on states (i.e., set sleep intervals to 0). On the other hand, when the transmission is finished, the nodes go back to LPL mode again, tuning the sleep settings according to the neighbor nodes' information.

Figure 2. An illustration of deployed motes at ASWP WSN testbed.

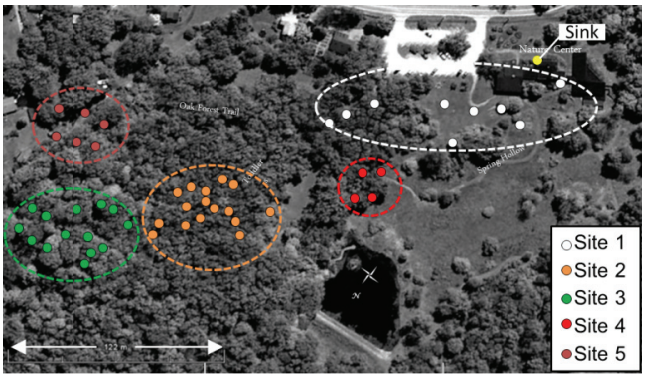


Figure 3. A map of the ASWP testbed indicating the relative locations of the base station and the nodes in 5 sites.

III. LONG-TERM WSN DEPLOYMENT

A. ASWP testbed

We have deployed and operated a real-world WSN testbed for over four years, containing over 50 sensor nodes, in a forested nature reserve at the Audubon Society of Western Pennsylvania (ASWP), USA. It collects ground-based measurement data for calibrating and validating scientific models in hydrology research [15]. The testbed uses two types of sensor nodes, MICAz [18] and IRIS [19], with an MDA300 [20] data acquisition board attached to each one. The base station, or the sink, is equipped with an IRIS mote with a permanent power supply. Each node is powered by rechargeable AA batteries and is protected by a waterproof enclosure to prevent possible damages from the harsh environmental conditions (i.e., weather, rain fall, snow and wild animals). Fig. 2 presents some examples of node configurations deployed at the ASWP testbed.

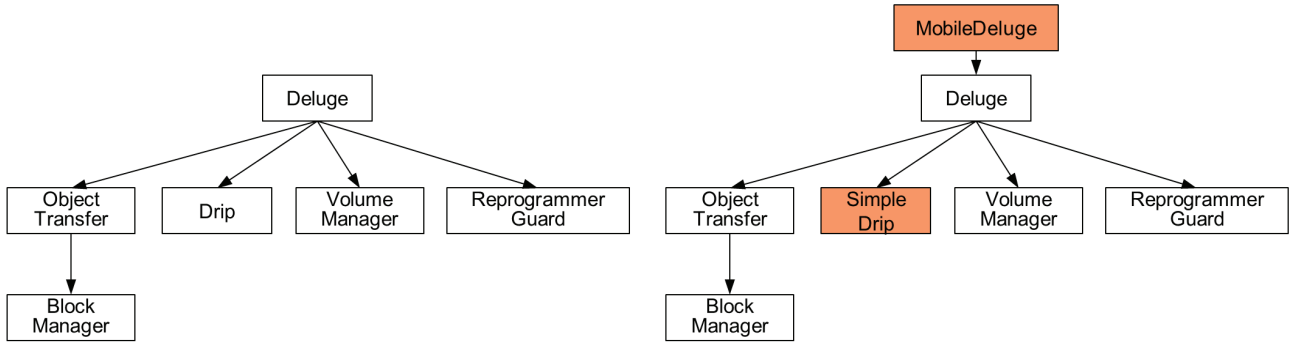
The data collection application is developed based on TinyOS 2.1.2, which incorporates the Collection Tree Protocol (CTP) [21] and LPL. All nodes are configured with a sleep interval set to *1* second in the LPL mode. Sensor data packets are sampled and transmitted every *15* minutes. The base station has a special configuration that allows it to stay awake and at the same time use the LPL preamble for all packet transmissions. The testbed is divided into five sites based on different requirements of sensor measurements for individual areas. Site 1 corresponds to the area next to the Nature Center, where the base station is located. The rest four sites are located in the forested hill-sloped region of the nature reserve. For each node platform, three application versions are developed: the version for relay nodes, the version for nodes with three external sensors, and the version for nodes with five external sensors. In total, there are six versions of programs deployed in the testbed. The five sites, including node positions, are presented in Fig. 3.

Operating a long-term outdoor WSN deployment in a harsh but accessible environment, such as the ASWP testbed, makes on-site network maintenance inevitable, which includes battery replacements, leaking enclosure fixes, and broken node replacements. In the network analysis of the ASWP testbed for the duration from August 2011 to August 2012 [15], it is found that nodes require attention about every *38* days on average. Consequently, such on-site maintenance required for long-term WSN operation are significantly more frequent than the reprogramming needed for algorithm/application improvement/update in our study. Although this observation is based on our experience with the ASWP testbed, we believe it indeed reveals a unique characteristic of long-term outdoor WSN deployments.

B. Reprogramming

Wireless reprogramming is clearly preferred in WSN deployments since manually reprogramming deployed nodes is very cumbersome. As we can see, the ASWP testbed has multiple node platforms (MICAz and IRIS) and regularly operates over low-power links, the typical features of real-world WSN deployment as described in Section I. To the best of our knowledge, none of the existing reprogramming approaches works in this scenario because none of them can address both network subset reprogramming and low-power link operation at the same time. This motivates our work. Since on-site visits of long-term outdoor WSNs are inevitable for network maintenance, mobile code dissemination is considered to effectively address the critical reprogramming challenges of heterogeneous WSNs operating over low-power links, which can be conveniently combined with on-site maintenance visits.

While the challenge of heterogeneous WSN reprogramming is clear, it would be helpful to give a sense about how inefficiently code dissemination approaches designed for always-on links would perform on a WSN over low-power links. We give an illustration example of the WSN LPL mode in which the sleep intervals are set to *1* second. Standard Deluge was used to disseminate a simple Blink program of *2592* bytes (*3* pages or *141* packets) for the evaluation of the performance with LPL mode and always-on mode, separately. The result is summarized in Table 1. With LPL mode, Deluge transmits about *200* times more packets than those over always-on links. It is about *50* times slower and consumes about *100* times more energy. This result clearly shows that LPL has dramatically degraded the efficiency of Deluge, as also indicated in [10].



(a) Deluge code structure

(b) MobileDeluge code structure (new components highlighted with respect to Deluge)

Figure 4. Code structures of Deluge and MobileDeluge.

TABLE I. PERFORMANCE OF DELUGE DISSEMINATION OF SMALL IMAGE WITH LPL ENABLED/DISABLED

	LPL	Always-on
ADV Pkts	25626	18
REQ Pkts	64	3
DATA Pkts	11792	141
Total Pkts	37487	162
Completion time (seconds)	297.97	4.90
Estimated energy consumption (mAs)	11564.21	111.72

IV. DESIGN

In this section, we present the design of MobileDeluge. MobileDeluge is a general network reprogramming tool built based on Deluge and effectively addresses both reprogramming challenges for long-term WSN deployments: heterogeneity and operating on LPL mode.

A. Deluge Overview

Deluge is a reprogramming system composed by several modules. First, Deluge uses Drip to disseminate command messages to the entire network for starting/stopping the image distribution process. Drip uses a Trickle [14] 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. In Deluge, a code image is divided into a set of fixed-size pages, whereas each page consists of a number of packets. Third, Deluge 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 image. The code structure of Deluge is shown in Fig. 4 (a).

Due to its dissemination nature to the entire network, Deluge fails in heterogeneous WSNs. Designed to operate over always-on links, Deluge has very poor performance with WSNs over low-power links, as illustrated in Section III.B. Besides, Drip messages introduce sustaining interference to the network in Deluge. Although the sending rate of Drip messages drops quickly when the network enters into a stable state following the Trickle timer, the minimal rate (e.g., *1024* seconds by default) may still be comparable to the data collection rate in low data rate sensor networks (e.g., in the ASWP testbed, the data is sampled every *15* minutes).

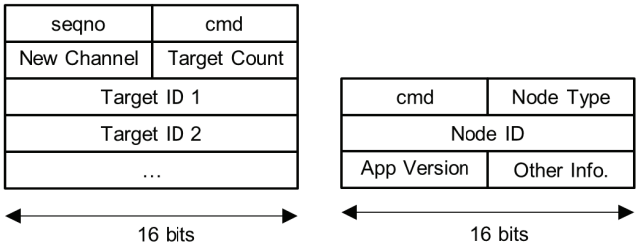
B. MobileDeluge Outline

To overcome the above weaknesses of Deluge, our design of MobileDeluge has the following key features: 1) one-hop network reprogramming, so that the reprogramming of a multi-hop network will be achieved by its mobility; 2) a novel control service enabling the retrieval of the platform information of the nodes in a one-hop neighborhood of the MobileDeluge base station (referred to as MobileBase), so that only the target nodes of the same platform type are reprogrammed at a time to address the heterogeneity; and 3) both MobileBase and the target nodes switched to a different channel with LPL disabled, allowing the fast and efficient transmission of the new code image without the interference to the rest of the network. In the following subsections, we present our design in detail.

C. Subset Reprogramming

MobileDeluge uses the basic broadcast scheme to establish the connection between the MobileBase and the target nodes, which limits its working range to a single hop. Its logic is split into two parts: the MobileBase side and the node side.

1) MobileBase: The MobileBase acts as a bridge between the target nodes and the mobile computer gateway connected to it. It receives commands from the gateway, and then broadcasts to the nodes. We divided the commands into two sets. The first set is the regular Deluge commands, which is directly processed by the standard Deluge logic. The other set, referred to as *Mobile commands*, is used for communication between the MobileBase and the target nodes. Basically two Mobile commands are defined: DISS and ABORT. DISS command starts a reprogramming cycle by notifying the target nodes to get ready, whereas ABORT is used to stop the



(a) command packet structure (b) reply packet structure

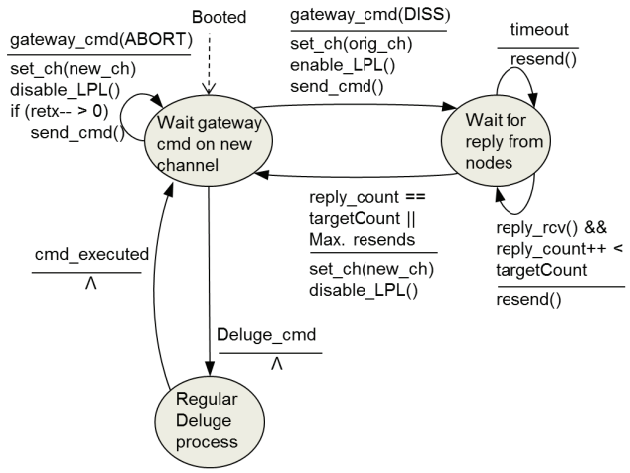
Figure 5. MobileDeluge packet structures.

reprogramming and reset the target nodes to their original state. In the Mobile command packet, the command occupies one byte, the other fields in the packet are the target nodes list, new channel, and other auxiliary information. The command packet structure is shown in Fig. 5 (a).

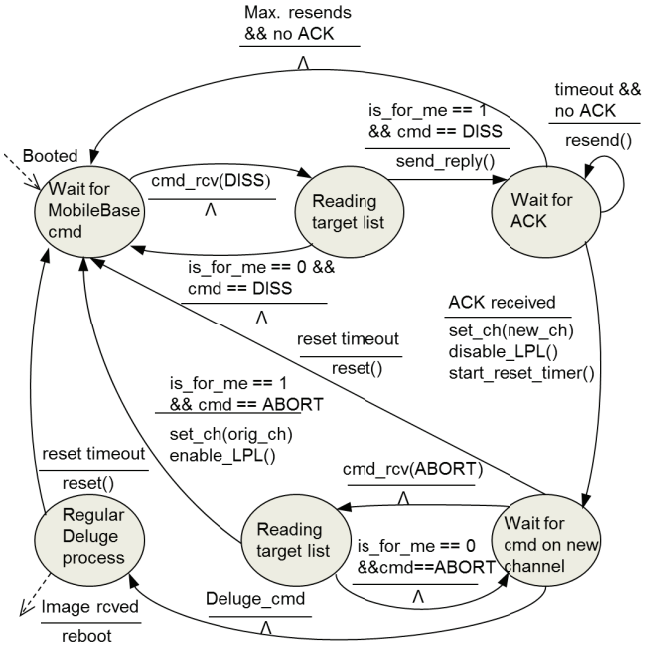
Before a reprogramming cycle is started, the target nodes are operating on the original channel with LPL enabled. In order to notify the target nodes, the MobileBase starts issuing a DISS command, which is broadcasted in the original channel of target nodes with LPL enabled. It then waits for the replies on the original channel. If all the nodes replied without delay, the MobileBase switches to a new channel and disables LPL. Otherwise rebroadcasting is needed until the maximal number of retransmissions is reached. When the MobileBase and the replied target nodes are on the new channel and over always-on links, regular Deluge commands will be issued to complete the reprogramming.

If the target nodes are mistakenly selected, or for some reason the reprogramming is no longer needed, an ABORT command can be issued to reset the nodes. The ABORT command does not require nodes' reply so that the nodes can reset as soon as the command is received. Instead, it is broadcasted for multiple times to ensure reliable delivery. Note that, when an ABORT command is needed, the target nodes are in the new channel. Thus, the MobileBase has to stay or switch to the new channel for broadcasting, depending on its current state. The MobileBase side's control is presented in Fig. 6 (a) as a finite state machine.

2) Node: Due to the broadcasting nature of wireless communications, a node's transmission can be received by any other nodes in its neighborhood. When a node operating in the regular application (i.e., the original state) receives a Mobile command packet, it checks the target list in the command packet. If it is not in the target list, the command is ignored. Otherwise, it responds according to the types of the command. If a DISS command is received, it sends a reply to the MobileBase and waits for an acknowledgment. If the reply packet is acknowledged, it switches to the new channel, and disables LPL, getting ready for reprogramming; otherwise, if no acknowledgment is recieved after several retransmissions,



(a) MobileBase side control logic



(b) Node side control logic

Figure 6. Finite state machines of MobileDeluge control logic.

it ignores the command. On the other hand, if an ABORT command is received, it resets to the original state (i.e., switches to the original channel and enables LPL) immediately.

When a node switches to the new channel, it waits for Deluge commands to finish the reprogramming. A reset timer is started to reset the node to the original state if the reprogramming is not finished in a certain time. The node side's control is presented in Fig. 6 (b).

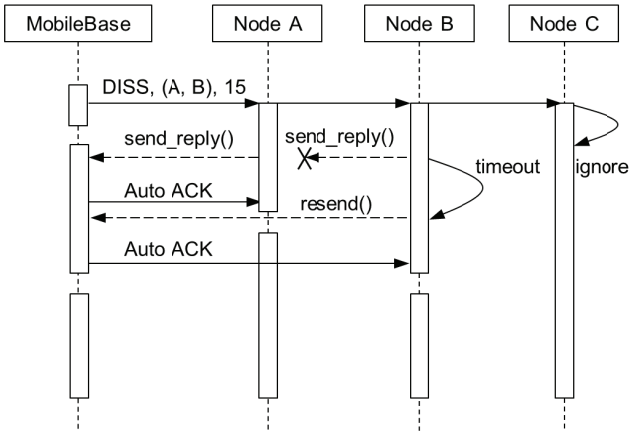


Figure 7. Start phase message exchange.

Fig. 7 shows an example of packet exchanges at the start phase of a reprogramming cycle. MobileBase broadcasts DISS command with target list (A, B) and new channel *15* in the original channel with LPL enabled. All nodes, A, B, and C, will receive the command. But only node A and B will send a reply after checking the target list. If the reply is lost, a retransmission is triggered (node B). When the acknowledgment is received, the nodes switch to the new channel and disable LPL. On the MobileBase side, when all replies are received, it switches to the new channel and disables LPL.

D. SimpleDrip

Since the reprogramming is limited to a single hop in our design, we replaced the multi-hop dissemination protocol Drip with a simplified version, referred to as SimpleDrip. SimpleDrip is a one-to-many single-hop dissemination protocol that maintains the same interfaces as Drip so that any protocols depending on Drip can seamlessly use it if single-hop dissemination is preferred. In the MobileDeluge, SimpleDrip replaces Drip to disseminate the Deluge commands to the target nodes in the neighborhood. Different from Drip, where every node periodically broadcasts according to the Trickle timer, in SimpleDrip, sender's (e.g., usually a base station) behavior and receiver's behavior are separate. The sender broadcasts the packet containing the new value that follows a linearly increasing timer, whereas receiver nodes do not transmit any packets. Thus, the rest of the network experiences no Drip traffic as all target nodes are receivers with the only sender being the MobileBase. The code structure of MobileDeluge is shown in Fig. 4 (b).

E. Mobile gateway

We develop the MobileDeluge gateway software, which runs on a laptop and controls the reprogramming cycle. It integrates the Mobile commands and the Deluge commands. MobileDeluge hence can form as a potential generic mobile command/query system for WSNs, with some extensions. Currently only two commands are implemented, as described above. When a node receives the DISS command, it can send very useful information along with the reply message, such as platform type, application version number, or neighbor table, to the gateway through the MobileBase. Target platform

```
* Please Select Operation:
* 1. Inject New Image
* 2. Connect Target Node
* 3. Disseminate Image and Reprogram motes
* 4. Stop Image Dissemination
* 5. Abort Reprogramming
* 6. Print the Base Station Information
* 7. Erase a Volume
* 8. Exit

* Input Operation Number: 2
Input Target Node ID List: 0xbb 204 0xdd
serial@/dev/ttyUSB1:57600: resynchronising

Sending command to: 187 204 221

Node 221 is READY       Type: MICAz     Version: 4
Node 187 is READY       Type: IRIS      Version: 5
Node 204 is READY       Type: TelosB    Version: 5
```

Figure 8. An illustration of MobileDeluge gateway interface, showing the platforms and application versions of target nodes.

information is significant in heterogeneous WSNs, since mistakenly reprogramming a node using the code image for a different platform can crash the node. In addition, version information tells whether reprogramming is necessary for the node. Fig. 5 (b) shows the reply packet structure. Fig. 8 shows an example of the MobileDeluge gateway interface.

V. Evaluation

We implemented MobileDeluge based on Deluge in TinyOS. In this section, we examine the performance of MobileDeluge in comparison to Deluge on a single-hop network over low-power links. However, to better understand the results, the performance of Deluge over always-on links is also provided as a reference. We shall compare the completion time and number of transmitted packets of the reprogramming process of both mechanisms. We used a sniffer [22] to record all the packets sent by the nodes. The sniffer attached a timestamp to each received packet, which is used to calculate the completion time. In this section, we present our evaluation not only based on laboratory experiments but also including actual reprogramming experience in our real-world ASWP testbed.

A. Lab experiments

Our ongoing WSN testbed study includes a combination of a basic data-collection service, CTP instrumentation for network management and analysis, routing inference, and network reprogramming with MobileDeluge. In order to accommodate all the information into one packet, we increased the TOS_DATA_LENGTH (i.e., the MAC layer payload size) to *75* bytes. Based on different sensor measurement requirements, there are six versions of programs differing in the configuration of parameters. Table 1 presents the image size of the testbed application with five external sensors when MobileDeluge or the standard Deluge is used, respectively. The other versions of programs differ in parameter configurations and have similar memory occupation.

TABLE II. CODE IMAGE SIZE

	With Deluge	With MobileDeluge
ROM (bytes)	43638	43568
RAM (bytes)	3362	3470
Image size (bytes)	44544	44544
No. Pkts	660	660

MobileDeluge occupies about *100* bytes more RAM than Deluge with all the functional augmentations discussed in the previous section, slightly less ROM, and the same image data size.

1) Reprogramming a single node: Table 3 shows the number of transmitted packets and the completion time for reprogramming a single node. Deluge with LPL was *21.8* times slower and sent *142* times more packets than MobileDeluge. On the other hand, despite the start phase, which took several seconds and sent 149 Mobile command packets over an LPL link, MobileDeluge has very similar behavior of the Standard Deluge, which is expected since once the start phase is finished, MobileDeluge actually works on the Deluge routine.

TABLE III. COMPARISON OF DELUGE AND MOBILEDELUGE FOR SINGLE NODE

	Deluge&LPL	MobileDeluge	Deluge
ADV Pkts	104852	188	187
REQ Pkts	302	46	44
DATA Pkts	44292	660	660
Start Pkts	0	149	0
Reply Pkts	0	1	0
Total Pkts	149446	1044	891
Completion time (seconds)	1365.95	59.95	54.96

2) Reprogramming a subset of nodes: To test MobileDeluge on heterogeneous networks, we setup a single hop network containing 5 nodes, in which 3 of them are MICAz nodes and others are IRIS motes. For Deluge, we only used 3 MICAz nodes, since it only works in homogeneous networks. MobileDeluge has successfully reprogramed all the target nodes in the hetergeneous network. Compared to Deluge over always-on links, it has transmitted about *200* more packets and taken *7* more seconds (which is in start phase) for completion. However, on low power links, Deluge took *24.7* times more completion time and transmits *138* times more packets.

TABLE IV. COMPARISON OF DELUGE (3 NODES) AND MOBILEDELUGE (3 OUT OF 5 NODES)

	Deluge&LPL	MobileDeluge	Deluge
ADV Pkts	111678	202	175
REQ Pkts	596	51	47
DATA Pkts	39916	663	663
Start Pkts	0	174	0
Reply Pkts	0	3	0
Total No. Pkts	152190	1093	885
Completion time (seconds)	1362.95	52.96	45.96

B. Testbed experience

MobileDeluge has been validated through reprogramming a subset of nodes in the outdoor ASWP testbed. We wirelessly reprogrammed the nodes in site 1, 2 and 4 (Fig. 3) with MobileDeluge, moved to a different reprogramming neighborhood at a time, and recorded the cost of disseminating a *50064* bytes' code image to the target nodes. The statistics are shown in Table 5. Due to the unreliable nature of wireless communications, the reprogramming statistics for each trial would vary from one to another. Reprogramming several nodes together needs more packets to be transmitted. However, the time consumption is very similar compared to reprogramming a single node. On the field, the size of the target subsets depends on the radio range of the MobileBase and relative locations of the target nodes. Since the code image is very large compared to regular data packets, the dissemination must be conducted in a very reliable manner. Thus, the effective radio range can be smaller than that in regular communication situations.

TABLE V. STATISTICS OF REPROGRAMMING IN THE FIELD

Target subset size	Type	Total Packets (DATA, ADV, REQ)	Completion Time (s)
1	IRIS	1196	63.95
1	IRIS	1172	74.94
1	MICAz	1219	87.93
1	MICAz	1214	65.95
1	MICAz	1195	72.94
3	IRIS	1257	68.94
3	IRIS	1283	72.94
3	MICAZ	1890	89.93
3	MICAZ	2115	55.96
4	MICAZ	2208	108.92
Total 21 Nodes	--	14749	762.42
Per Node Avg.	--	702.33	36.30

The manual reprogramming procedure starts from getting the enclosure from the tree. Then several screws that seal up the box and fix the node with the acquisition board have to be taken off, and then, the node needs to be attached to a laptop to be reprogrammed. The previous procedure of getting the node has to be reversed after the reprogramming to put the node back to its deployment location. Usually, it takes a whole day to finish reprogramming site 1, 2, and 4. Experience shows that MobileDeluge has significantly improved the efficiency of the field reprogramming work.

VI. CONCLUSION

Long-term outdoor WSN operations usually require thatnode deployments be located in accessible although sometimes harsh environments for the possibility of on-site maintenances. Those real-world WSN deployments are more likely to be heterogeneous and operating over low-power links, which makes existing code dissemination protocols unworkable. To address these challenges, we propose MobileDeluge, a novel mobile reprogramming tool which is able to reprogram heterogeneous WSNs regularly operating over low-power links. We have evaluated the performance of MobileDeluge through laboratory experiments and real-world outdoor WSN testbed reprogramming. Results show that MobileDeluge has efficiently addressed the reprogramming challenges of heterogeneous WSNs and WSNs over low power links at the same time, making it very suitable for long-term outdoor WSN deployments where on-site maintenance is usually needed. The design of MobileDeluge also illustrates a general approach for building a mobile code dissemination tool based on some existing code dissemination protocol, such as Deluge, which in principle can be applied to other existing code dissemination protocols as well. If outdoor WSN deployments are not accessible by the maintenance team, a new code dissemination protocol with the fixed control/sink node would need to be developed for heterogeneous WSNs over low-power links.

MobileDeluge is intended to be open-source software and will be released to the community.

ACKNOWLEDGMENTS

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] Crossbow Technology, "Mote in-network programming user reference," 2003. http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/Xnp.pdf

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

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

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] A. Tridgell, "Efficient algorithms for sorting and synchronization," Ph.D. Thesis, Australian National University, Canberra, 1999.

[12] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, 18(6):341-343, June, 1975.

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

[14] 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.

[15] M. Navarro, T. Davis, Y. Liang, and X. Liang, "A study of long-term WSN deployment for environmental monitoring," in *Proceedings of PIMRC*, September, 2013.

[16] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: sustainable sensing in the forest," *in Proceedings of SenSys*, 2009.

[17] X. Mao, X. Miao, Y. He, T. Zhu, J. Wang, W. Dong, X. yang Li, and Y. Liu, "Citysee: urban co2 monitoring with sensors," in *Proceedings of INFOCOM*, 2012.

[18] MICAz Wireless Measurement System, http://www.memsic.com.php5-12.dfw1-1.websitetestlink.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html

[19] IRIS Wireless Measurement System, http://www.memsic.com.php5-12.dfw1-1.websitetestlink.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html

[20] MDA300 Data Acquisition Board, http://www.memsic.com.php5-12.dfw1-1.websitetestlink.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html

[21] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: an efficient, robust, and reliable collection tree protocol for wireless sensor networks," in *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 3, 2013.

[22] A. Koubaa, S. Chaudhry, O. Gaddour, R. Chaari, N. Al-Elaiwi, H. Al-Soli, and H. Boujelben, "Z-Monitor: monitoring and analyzing ieee 802.15.4-based wireless sensor networks," in *Proceedings of Local Computer Networks (LCN),* 2011, pp. 939-947, October, 2011.