IR-Seluge: Interference-Resilient Code Dissemination in Wireless Sensor Networks *

Sangwon Hyun, Peng Ning, An Liu
Department of Computer Science
North Carolina State University, Raleigh, NC 27695
{shyun2, pning, aliu3}@ncsu.edu

Abstract

Code dissemination, which refers to the process of propagating a new code image to deployed sensor nodes through wireless links, is an essential service for wireless sensor networks. It must guarantee that every node in a network correctly receives a complete code image. Several methods have been developed for reliable and efficient code dissemination in wireless sensor networks. However, none of them considers the negative impact of environmental and unintentional wireless interference. In particular, all existing approaches will fail if severe interference exists in a single channel (e.g., control channel). To address this problem, we present an interference-resilient code dissemination system named IR-Seluge for wireless sensor networks. IR-Seluge has been implemented in nesC on TinyOS and evaluated through a substantial set of experiments in a wireless sensor testbed, which consists of 73 MicaZ motes. The experimental results indicate that IR-Seluge effectively and efficiently mitigates the problem of wireless interference.

Keywords: *Wireless Sensor Networks, Code Dissemination, Wireless Interference, Resilience, Reliability*

1. Introduction

A wireless sensor network is expected to consist of a potentially large number of low-cost, low-power, and multifunctional sensor nodes that communicate over short distances through wireless links. Due to their potential to provide fine-grained sensing and actuation at a reasonable cost, wireless sensor networks are considered ideal candidates for a wide range of applications, such as industry monitoring, data acquisition in hazardous environments, and military operations.

It is desirable and sometimes necessary to reprogram sensor nodes through wireless links after they are deployed, due to, for example, the need of removing bugs and adding new functionalities. The process of propagating a new code image to the nodes in a network is commonly referred to as *code dissemination*. Code dissemination must ensure complete reliability, which means that every node in a network correctly receives a complete code image. Otherwise, the regions where the new code image does not reach cannot perform the intended tasks in the new code image. Moreover, unexpected inconsistency on the code images in the network may cause serious consequences due to unintended, wrong interactions between the different images.

Several code dissemination protocols (e.g., [5, 8, 20, 10, 18, 16]) have been developed to propagate code images to all sensor nodes using the ad-hoc network formed by the nodes themselves. In particular, Deluge [8], which uses an epidemic protocol [12] for meta data advertisement and spatial multiplexing for efficient propagation of code images, has been included in recent TinyOS distributions and widely used. Moreover, a few approaches have been proposed recently to extend Deluge for secure code dissemination in wireless sensor networks (e.g., [6, 7, 11, 9]).

However, none of these approaches considers the problems of environmental wireless interference. For example, if severe interference exists on the channel for code dissemination, none of these approaches would be able to complete code dissemination. It is natural to consider using multiple channels to mitigate the impact of wireless interference. Indeed, several protocols [22, 19, 21, 13] have been developed to improve code dissemination by exploiting multiple channels available on current sensor platforms (e.g., MicaZ [1], TelosB [2]). Unfortunately, all these approaches use a single, fixed control channel for communicating control messages; they will fail if there is serious wireless interference on the control channel.

In fact, all existing approaches can be easily disrupted by wireless interference. It is imperative to develop a new

*The material contained in this paper has been cleared through the authors' affiliation. Submission category:DCCS (regular paper), approximate word count:7,800, contact author:Sangwon Hyun

solution to enhance the reliability of code dissemination in the presence of wireless interference.

In this paper, we present the design, development, and evaluation of an interference-resilient code dissemination system named *Interference-Resilient-Seluge*, or simply *IR-Seluge*, for wireless sensor networks. IR-Seluge is an extension to Seluge [9], which is a secure code dissemination system based on Deluge [8] for wireless sensor networks. Seluge provides various security protections while preserving efficient dissemination mechanisms in Deluge. In addition to the security protection offered by Seluge, IR-Seluge provides additional resilient properties against wireless interference. Code dissemination in IR-Seluge may be concurrently performed in multiple dissemination flows using different channels, without relying on any specific channel (e.g., a control channel). The impact of interference on a single channel is limited to only the nodes staying in that channel. Moreover, if a channel becomes unavailable, nodes using the channel will migrate to other available channels. Thus, as long as not all channels are disabled in a region under wireless interference, IR-Seluge can still propagate code images in that region.

Our contribution in this paper is three-fold: Our first and most important contribution is the development of the IR-Seluge protocol, which provides efficient and strong interference resilience properties in code dissemination. (To the best of our knowledge, this paper is the first that considers wireless interference problem in code dissemination.) In particular, we make an original contribution in the design of multi-channel protocol mechanisms that allow nodes to notify their own status to their neighbor nodes on different channels and to switch channels to communicate with their neighbor nodes. Second, we extend the code base of Seluge to develop a readily available software package for interference-resilient code dissemination in wireless sensor networks. Finally, we provide a careful evaluation of IR-Seluge through both theoretical analysis and experimental evaluation, which demonstrate the nice properties and real-world performance of IR-Seluge.

The rest of this paper is organized as follows. Section 2 clarifies our assumptions and interference model. Section 3 gives a brief overview of Seluge. Section 4 presents the protocol design of IR-Seluge. Section 5 provides a qualitative analysis of IR-Seluge's interference resilience property. Section 6 and Section 7 describe the implementation and the experimental evaluation of IR-Seluge in a network of 73 MicaZ motes. Section 8 discusses related work, and Section 9 concludes this paper and points out the future research direction.

2. Assumptions and Interference Model

Assumptions: We assume that a sensor node can switch its radio frequency at run time. This is in fact true on most current sensor platforms (e.g., Mica2 [4], MicaZ [1], TelosB [2]). For instance, a MicaZ mote has 16 different channels with 5MHz spacing.

We assume Seluge as the underlying code dissemination system. Accordingly, we inherit the assumptions made by Seluge [9]. These include: (1) the source of the code images, i.e., the *base station*, is a powerful node with sufficient energy supply (e.g., a laptop PC); (2) sensor nodes are resource constrained in terms of computational power, communication capability, storage capacity, and energy; (3) each node has enough memory (i.e., external flash) to store the disseminated code image; (4) the base station has a private and public key pair, and each sensor node in the network is pre-configured with the base station's public key.

Interference Model: In this paper, we consider environmental and unintentional wireless interference. Wireless sensor networks frequently experience high loss rates and dynamic link quality changes [23]. Moreover, due to its use of open spectrum, a wireless sensor network will be highly likely to co-exist with different networks using the same frequency bandwidth, then the limited frequency bandwidth can be congested [24]. In addition, it is well known that electronic appliances like microwave oven cause strong interference to wireless sensor networks.

We assume that wireless interference may exist in communication channels potentially used for code dissemination. The affected area(s) may be the entire region or multiple subregions of the network. We assume that in different regions interference on different channels can exist. In a single region, multiple channels can be unavailable due to interference at the same time.

Note that if there is serious interference on all available channels in a region at the same time, IR-Seluge will fail in that region. Indeed, there is no solution with the radio transceivers on the current sensor platforms in such case. However, unlike other previous solutions, which will fail if severe interference on the channel for code dissemination or the control channel, IR-Seluge can disseminate code images as long as (any) one channel is available.

3. Background: Seluge Overview

Seluge is a secure code dissemination system for wireless sensor networks [9]. Seluge relies on Deluge [8] for efficient propagation of code images, and at the same time provides security protection to ensure the integrity of disseminated code images and defend against various DoS attacks at the dissemination protocol level.

Seluge adopts the page-by-page dissemination strategy from Deluge. A code image is divided into fixed-size pages, and each page is further split into same-size packets. Each sensor node can request a page only after completely receiv-

ing every packet in the previous page. Seluge uses a digital signature to bootstrap the authentication of a new code image, and uses cryptographic hash functions to authenticate the actual dissemination packets. For the second to the last page, the hash image of every packet is included in the corresponding packet of the previous page. As a result, the authentication of a page allows the authentication of every packet in the next page. Seluge uses Merkle hash tree [15] to authenticate the hash images of the packets in the first page, while the root of the Merkle tree is authenticated by the bootstrapping signature. Along with the page-by-page dissemination strategy, the packet structure of Seluge provides immediate authentication of every code dissemination packet.

Seluge prevents DoS attacks exploiting the epidemic propagation and suppression mechanisms of Deluge. The root cause of these vulnerabilities is the lack of authentication of advertisement and request packets. Seluge provides local broadcast authentication using cluster key, which allows each node to immediately authenticate an advertisement or data request packet upon receipt.

Finally, Seluge provides resistance to DoS attacks against bootstrapping signatures. By broadcasting a large number of packets with bogus signatures, the adversary can force all the receivers to perform expensive signature verifications and eventually exhaust their limited battery. To deal with such threats, Seluge uses *Message Specific Puzzle* [17] to effectively filter out forged signatures.

4. IR-Seluge Protocol

The basic idea of IR-Seluge is to fall back to alternative channels when the main channel for code dissemination is under severe wireless interference. Unlike previous approaches that also use multiple channels for code dissemination, IR-Seluge does not rely on any fixed control channel.

For the sake of presentation, we first clarify a few terms. The *primary channel* for a sensor node is a channel where the node uses to transmit and receive code dissemination packets. Each sensor node stays in the primary channel for at least the *channel switching period*. All the remaining channels except for the primary channel are called *secondary channels*. A node has an *active task* if it is transmitting/receiving, or scheduled to transmit/receive code dissemination packets. A node can have an active task only in the primary channel, and such a node cannot leave the primary channel before finishing the active task.

We now present the overall behavior of the IR-Seluge protocol. As in Seluge [9] and Deluge [8], IR-Seluge uses the epidemic protocol Trickle [12] for sensor nodes to exchange their state information related to code dissemination. To notify its current state related to a code image to its neighbors, each node periodically advertises the image version number, the number of received pages, its current primary channel number, and its remaining channel switching period. Since its neighbor nodes may be using any channel, the node chooses a channel randomly (based on certain distribution) to transmit the advertisement packet.

If a node discovers that a neighbor node in its primary channel is in a different state, the node follows the Seluge protocol to start transmitting or receiving code dissemination packets (i.e., start an active task) in the primary channel. Otherwise, the node analyzes advertisements from the neighbor nodes with different primary channels to figure out if there is any potential active task in one of its secondary channels. If yes, the node switches its primary channel to that secondary channel. However, if the answer is no, the node randomly selects a secondary channel and switch its primary channel there.

When a node finishes an active task (either sending or receiving), it resets its channel switching period. In other words, the node needs to stay in its current primary channel longer, since there is work to do.

Intuitively, when the primary channel of a node is under serious interference, the node will receive almost no message in the primary channel. Thus, after the channel switching period, the node will move to a different channel. If there is no severe interference in the new primary channel, the node will overhear advertisements from nodes either in the primary channel or other secondary channels. In either case, the node will find new active tasks in the primary channel or a secondary channel, and code dissemination can continue there.

In the following, we describe the IR-Seluge protocol in detail. Similar to the Seluge and Deluge protocols, there are three protocol states for each node: $Maintenance$, $Receiving$, and $Transmitting$. We describe the protocol behaviors for them separately.

4.1. Maintenance State

The main task of a node in $Maintenance$ (MT) state is to maintain a consistent view with its neighbor nodes (i.e., nodes within the signal range) on their program images. A node in MT state periodically broadcasts advertisement packets in both primary and secondary channels. Intuitively, such advertising allows the node to notify its neighbor nodes about its current status and detect any discrepancy from the neighbors.

When a node enters the MT state, it starts an advertisement period, which consists of a *monitoring* period and a *random backoff* period. The random backoff is to avoid message collisions due to simultaneous advertisements from neighbors. During the monitoring period, the node monitors the communication in its primary channel

over the neighborhood. At the end of each advertisement period, based on its observation, the node decides whether it transmits or suppresses the current advertisement, and if transmitting, the channel it will advertise in. After the current advertisement period, the node repeats the same procedure for the next advertisement period. We will present how to select a channel to advertise later in this section.

Our protocol follows Trickle [12] in adjusting the monitoring period (between MP_{min} and MP_{max}) to control its advertising frequency. If a node overhears an advertisement about a different version or a different number of pages during the current advertisement period, it sets the duration of the next monitoring period to MP_{min}. Otherwise, the node decides whether to increase it or not. Before increasing the monitoring period, the node checks if there is data it needs to download from its neighbors. If yes, it does not increase its monitoring period, since there is still on-going code dissemination. Otherwise, it doubles the next monitoring period unless it exceeds MP_{max}. This approach allows rapid propagation of a new program image, while consuming little resources in steady state.

4.1.1 How to Select Advertisement Channel

Since each node performs active tasks only in the primary channel, notifying each other among the neighbors using the same primary channel is very important. In addition, it is also important to have nodes using different primary channels communicate with each other. In IR-Seluge, each node advertises in the primary channel every n advertisement periods. For all the remaining periods, each node selects one of its secondary channels and advertises there. Moreover, if a node detects a potential active task in the primary channel but nothing different in the secondary channels, the node increases the frequency of advertisements in the primary channel. If a node has already overheard the same advertisement in the primary channel enough times, it advertises in a secondary channel instead of the primary one.

When selecting a secondary channel to advertise, a node gives higher priority to those where there are more nodes with smaller portion than itself. This is to maximize the possibility that more nodes will benefit from this advertisement (by, e.g., switching to this channel). But if these nodes already have a potential sender in their primary channel, we exclude the corresponding channel, because those nodes can receive the data from other potential sender there.

Now let us present the detailed procedure. If the current advertisement period is the turn to advertise in the primary channel and the node has not overheard at least k copies of the same advertisement from the primary channel during the current period, the node advertises in the primary channel. Otherwise, the node checks whether there is a secondary channel it needs to advertise in. For this checking procedure, we use the following two searches. In both searches, if there are multiple candidates meeting the condition, we randomly choose one of them.

- *Search 1* looks for the secondary channel where there are the nodes with the oldest version or the fewest number of pages in the current version. Intuitively, *Search 1* is to find a secondary channel where there are nodes this node can help.
- *Search 2* looks for the secondary channel where there are the nodes with the most number of pages. Intuitively, *Search 2* is to find a secondary channel where there are nodes from which this node can get help.

We need to consider three cases. In the first case, a node currently has something to receive in the primary channel. The node only performs *Search 1*. Since the node already has a potential sender in the primary channel, it does not need to perform *Search 2* to find a potential sender in a secondary channel for itself. In the second case, a node has potential receivers in the primary channel. The node first performs *Search 1*, and if it finds no channel, it further performs *Search 2*. In both cases, after the node finds a secondary channel, it switches its radio to that channel, advertises there, and switches back to the primary channel. Through advertising in that channel, the node tries to attract to its primary channel potential senders as well as receivers from its secondary channels. If the node cannot find any secondary channel through the two searches, it advertises in the primary channel or suppresses the current advertisement.

In the third case, the node has nothing to send or receive in the primary channel so far. The node then tries to select a secondary channel by *Search 1* and then *Search 2*. If no channel is found, the node randomly selects a secondary channel for advertisement.

4.1.2 How to Switch Primary Channel

The ability to switch the primary channel is critical for robustness against interference. It allows nodes to evade the channel under serious interference and continues code dissemination in other available channels.

Based on its observation of the current primary channel, a node in MT state considers switching its primary channel in the following three cases:

1. The node overhears an advertisement from a neighbor in a secondary channel with a different version number or different number of pages with the same version number for the disseminated code image.
2. The current channel switching period has expired.

3. The node completes a page.

Consider the first case. Intuitively, this case represents the situation where a node realizes that it can help others or get help from others if it switches its primary channel. Let X denote the node that considers switching its primary channel and Y the neighbor that triggered X's channel switch. Let C_X and C_Y be X's and Y's current primary channel, respectively. From Y's point of view, C_X is a secondary channel. Similarly, C_Y is X's secondary channel.

If Y has a later version than X, X changes its primary channel to C_Y right away, because updating to the latest version is more important than any active task in the current primary channel. Otherwise, X needs to examine its primary channel C_X further to determine whether it will switch to C_Y. First, if X currently has a neighbor in C_X with more pages in the current version, it does not switch to C_Y because X still needs to receive in C_X. In other words, the receiving active task in the primary channel has higher priority than potential active tasks in secondary channels. Next, suppose X only has neighbors with fewer pages in C_X. In this case, if X currently has at least one neighbor with the same number of pages as itself in C_X, it switches to C_Y with a certain probability p_1 ($0 < p_1 < 1$), since the node with the same number of pages will possibly serve those neighbors instead of X. Finally, if X has discovered no neighbors in C_X with different number of pages, it switches to C_Y right away.

Now consider the last two cases. Both of them are motivated by potential wireless interference. Consider the following two scenarios: (1) X is inside the region, where severe interference exists on C_X, and (2) X is on the boundary of the region (i.e., with its current primary channel X cannot communicate with neighbors on C_X inside the region, but can communicate with neighbors on C_X outside the region). In the former scenario, X has to escape C_X as soon as possible. In our protocol, if X has discovered no neighbors from any channel for the channel switching period, it randomly selects a secondary channel and switches its primary channel there. (Note that the node would have renewed the channel switching period or switched to another primary channel if it receives packets from other nodes.)

In the latter scenario, X cannot communicate with the neighbors on C_X inside the region with severe interference on C_X but can communicate with those on C_X outside the region. To communicate with the neighbors affected by the interference, X has to leave C_X. At the same time, if X has something to receive from the unaffected neighbors on C_X, it would be good for X to stay in C_X.

To deal with this problem, whenever X finishes receiving a page, X decides whether it will leave C_X or not based on the current circumstances in C_X. First, suppose X still has a neighbor with more pages in C_X. Then X can still receive data from the neighbor using C_X, but at the same time it may need to leave C_X for potential neighbors under serious interference with C_X. In our protocol, X moves to a random secondary channel with probability p_2 ($0 < p_2 < 1$). Second, suppose X does not have neighbors with more pages but neighbors with fewer pages in C_X. In this case, if X currently has at least one neighbor with the same number of pages as itself in C_X, it moves to a random secondary channel with probability p_1 ($0 < p_1 < 1$), since the node with the same number of pages will possibly serve those neighbors with fewer pages. Finally, if X has only discovered neighbors with the same number of pages in C_X, it moves to a random secondary channel right away.

4.2. Receiving State

IR-Seluge adapts the Deluge protocol [8] in the $Receiving$ (RX) state. If a node overhears an advertisement of more pages in the same version from a neighbor with the same primary channel, it transitions its state to RX to request those pages. While in RX state, a node never changes its primary channel and never broadcasts an advertisement.

Let R denote a node which transitions to RX state and S denote a neighbor that triggered R's state transition. In order to receive the newly discovered data, R sends a request to S (with a random backoff to avoid collisions due to simultaneous requests). Each request message includes the destination, image version, index of the requested page within the image, and requested packets in that page. For each request, only the destination node has the responsibility to transmit the requested packets. After sending the request, R waits for a fixed period of time. Whenever receiving a requested packet, R reinitiates this time period. If R receives no requested packet during the time period or R has completely received all the requested packets, R transitions back to MT state.

Our protocol uses the following suppression rules to reduce redundant transmissions. If R overhears an advertisement from a neighbor that uses the same primary channel and has fewer number of pages than itself, R suppresses its current request and transitions back to MT state, since that neighbor has higher priority to get served. However, this rule does not apply to advertisements from secondary channels, since the advertising node may find a potential sender in its own primary or a secondary channel different from R's primary channel. Similarly, if R overhears a request or data packet for a page whose index is less than that of the page to be requested, R suppresses its request for the same reason. If R overhears a request or data packet for the same page it is currently working on, R delays its request until the end of the data transmission for the overheard request or data transmission. If R still needs some packets in its

current working page after the data transmission, R sends a request for those packets.

4.3. Transmitting State

IR-Seluge also adapts the Deluge protocol [8] in the $Transmitting\,(TX)$ state. If a node receives a request destined to itself, it verifies whether it has the requested data, and if yes, it transitions to TX state to serve the request. Similar to RX state, a node in TX state never changes its primary channel and never broadcasts an advertisement.

Each node in TX state remembers the index of the requested page and the requested data packets in that page using a bit vector. The node starts broadcasting the requested packets as soon as it receives the request. When new requests for the same page are received during the transmission, the node aggregates the requested packets and transmits them in a round robin fashion, until all requested packets are transmitted. After broadcasting all the requested packets, the node transitions back to MT state.

Several suppression rules are also used in TX state to reduce redundant transmissions. Let S denote a node which just transitions to TX state. If S overhears an advertisement from a neighbor in the same primary channel with lower page index than the one S is about to broadcast, S suppresses its data transmission, since the neighbor is likely to request that page, which has higher priority. S then transitions back to MT state. Similarly, when overhearing a request or a data packet for a page with lower index than the page S is about to broadcast, S also suppresses the data transmission and transitions back to MT state.

5. Analysis

In this section, we provide a qualitative analysis of IR-Seluge's interference resilience property. An quantitative evaluation through experiments in a sensor testbed is given in Section 7.

In the following analysis, we assume that each noise source is randomly located in the network and generates strong interference in a single, random channel over its lifetime, which makes the channel unavailable. Sensor nodes are randomly distributed over the entire network region.

We use the following notation. Ch_i represents the ith channel in the system. N_{CHi_s} represents the sth noise source, which interferes with Ch_i. S_E and S_A denote the area of the entire network region and node A's signal range, respectively. x represents the total number of channels in the system. $t_i (1 \leq i \leq x)$ denotes the number of noise sources in the system that interfere with Ch_i.

Lemma 1. *The probability that any two nodes in the same neighborhood have at least one common available channel is greater than or equal to*

$$\sum_{i=1}^{x}(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2 \tag{1}$$

Proof. Let n_1 and n_2 denote two nodes in the same neighborhood. There are t_i noise sources that interfere with Ch_i in the network: $N_{CHi_1}, N_{CHi_2}, \ldots, N_{CHi_{t_i}}$. In order for Ch_i to be available to each node, it must be outside the signal ranges of all t_i noise sources. The probability that both n_1 and n_2 are not in the signal ranges of any of t_i noise sources is $(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2$ (a). Since there may be intersections between the signal ranges of t_i noise sources, the probability is actually greater than or equal to (a). Because there are x channels in total, the probability that n_1 and n_2 have at least one common available channel is greater than or equal to $\sum_{i=1}^{x}(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2$. □

Let us divide the time line into a series of unit time periods. For simplicity, we assume that each node switches to an independent channel across time periods.

Lemma 2. *The probability that any two nodes in the same neighborhood meet in Ch_i in any one of l time periods is*

$$1-(1-(\frac{1}{x})^2)^l \tag{2}$$

Proof. The probability that both n_1 and n_2 are in Ch_i in any single time period is $(\frac{1}{x})^2$. Then the probability that n_1 and n_2 do not meet each other in Ch_i for l time periods is $(1-(\frac{1}{x})^2)^l$. Therefore, the probability that n_1 and n_2 meet in Ch_i in any one of l time periods is $1-(1-(\frac{1}{x})^2)^l$. □

Lemma 3. *The probability that any two nodes in the same neighborhood meet in a channel available to both of them in any one of l time periods is greater than or equal to*

$$(1-(1-(\frac{1}{x})^2)^l)\sum_{i=1}^{x}(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2 \tag{3}$$

Proof. The probability that n_1 and n_2 meet in Ch_i in any one of l time periods and Ch_i is available to both of them is greater than or equal to $(1-(1-(\frac{1}{x})^2)^l)(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2$. Since there are x channels in the system, the probability that n_1 and n_2 meet in a channel available to both of them in any one of l time periods is greater than or equal to $(1-(1-(\frac{1}{x})^2)^l)\sum_{i=1}^{x}(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2$. □

As time goes on, $(1-(1-(\frac{1}{x})^2)^l)$ in (3) converges to 1 because $\lim_{l\to\infty}(1-(1-(\frac{1}{x})^2)^l)=1$. Then the probability that any two neighbor nodes meet in a channel available to both of them in any one of l time periods becomes greater than or equal to $\sum_{i=1}^{x}(\prod_{s=1}^{t_i}(1-\frac{S_{N_{CHi_s}}}{S_E}))^2$, and that is

exactly equal to the probability in Lemma 1. This implies if one node has a new code image that the other node in the same neighborhood needs, and they have at least one common available channel, then the new code image can be eventually propagated to the latter node in IR-Seluge.

6. Implementation

We have implemented IR-Seluge as an extension to Seluge [9], a secure remote programming system based on Deluge [8] for wireless sensor networks. On the base station side, IR-Seluge needs no additional functionality compared to Seluge. On the sensor side, we added two modules, `IRSelugeChTracker` and `IRSelugeChManager`, into the Seluge code base.

`IRSelugeChTracker` is responsible for maintaining the statistics about the primary and secondary channels based on received advertisement, request, and data packets. `IRSelugeChManager` includes all the functionalities related to managing the channels, such as managing the primary channel and the channel switching period, channel selection for advertisement packets, and channel switching for transmitting advertisement in a secondary channel.

Table 1. Code size (bytes) on MicaZ.

	ROM	RAM
Deluge	22,226	1,123
Seluge	45,258	2,278
IR-Seluge	49,236	2,481
TinyECC in Seluge/IR-Seluge	13,044	426

Table 1 shows the ROM and RAM usage of Seluge and IR-Seluge on MicaZ motes. The code sizes of Deluge and TinyECC are also included for reference purposes. It is easy to see that IR-Seluge just slightly increases the ROM and RAM consumption compared with Seluge. Seluge and IR-Seluge do significantly increase both the ROM and RAM consumption compared with Deluge, and a significant portion of the ROM increase is due to TinyECC.

7. Experimental Evaluation

In this section, we report the experimental evaluation of IR-Seluge in a wireless sensor testbed. We first compare the performance overhead of our scheme with Seluge without and with strong noise, and then further evaluate the interference resilience of IR-Seluge in different interference scenarios.

In order to emulate serious wireless interference, we use several nodes in the testbed as noise sources. For the noise sources, we modify the CC2420 Radio module in TinyOS so that a noise source keeps sending packets without clear channel assessment (CCA). With this program, each noise source continuously injects noise into a specific channel throughout each experiment.

Evaluation Metrics: We use two metrics in our evaluation: *average completion time* and *communication overhead*. The average completion time is the average time required for all nodes to receive a disseminated code image. The communication overhead is measured as the total number of packets transmitted by all the nodes during code dissemination. For communication overhead, we show the number of request and data packets and the number of advertisements separately.

Testbed: We perform the experiments using 73 MicaZ motes in the WiSeNet sensor testbed for our experiments. Figure 1 shows the layout of the testbed. The sensor nodes are deployed on the second floor of Engineering Building II at North Carolina State University. The testbed area includes offices, labs, server rooms, and corridors, covering an area of 152.5 feet $\times$ 97 feet. We equip each node with an Ethernet programming board, which provides remote access to the node. We only use the programming boards to gather evaluation results from the sensor nodes; they do not interfere with the radio communication between sensor nodes at all. We set the transmission power level of the MicaZ radio module (CC2420) as 0dBm. The square-shaped nodes with numbers are used as noise sources.

Figure 1. The WiSeNet testbed (73 MicaZ motes; 152.5 feet $\times$ 97 feet).

Experiment Setup: For our scheme, we run all the experiments with two different conditions on the initial primary channel: *random* and *fixed* initial primary channel. With the random initial primary channel, a node starts with a randomly selected channel as the initial primary channel, while with the fixed initial primary channel, each node starts with a designated initial primary channel. We examine the effect of these initial conditions on the performance of our scheme in each experiment scenario. We call our scheme

with random and fixed initial primary channel *IR-Seluge-RandomInitPCh* and *IR-Seluge-FixedInitPCh*, respectively.

We use channel 26 for Seluge, which does not overlap with 802.11 [3]. Likewise, we use channel 26 for the initial primary channel in IR-Seluge-FixedInitPCh. We set the channel switching period to 8 advertisement periods. A node tries to advertise in the primary channel for every 3 advertisement periods. We set $p_1 = 0.3$ and $p_2 = 0.1$. We follow the same settings in Seluge (and also Deluge) to set the lower and the upper bounds of the advertisement period to 512ms and 70 minutes, respectively. As in Seluge, we put 17ms interval between each data packet transmission to accommodate the time required by Seluge's verification operation. In each experiment, the dissemination starts from the star-shaped node located at the bottom-right corner in Figure 1. For each test case, we perform the same set of experiments 10 times and take the average over them. We also show 95% confidence intervals.

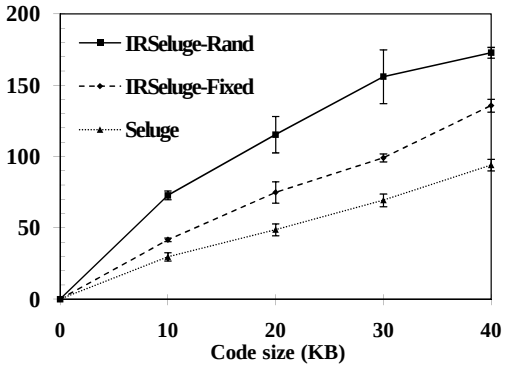
7.1. Scenario 1—No Interference

We first compare the performance of IR-Seluge and Seluge when there is no noise source. To see the impact of code image size on performance, we use four different sizes: 10K bytes, 20K bytes, 30K bytes, and 40K bytes. Figures 2(a) and 2(b) show the average completion time and communication overhead in the experiments, respectively.

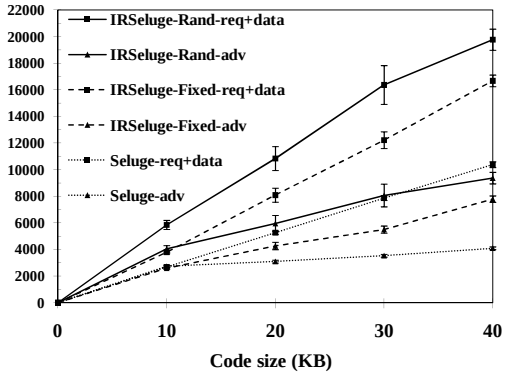
Let us first compare IR-Seluge-RandomInitPCh and Seluge. The average completion time of IR-Seluge-RandomInitPCh is about 80% to 145% longer than that of Seluge. Over all the experiments, the overhead for request and data packets of IR-Seluge-RandomInitPCh is about 100% larger than that of Seluge, and IR-Seluge-RandomInitPCh introduces about 99% more advertisement packets than Seluge.

In IR-Seluge-RandomInitPCh, since every node randomly selects its initial primary channel, the nodes stay in all 16 different channels when dissemination begins. Thus, it takes some time for each node to find neighbors in different channels that have data it needs. In addition, when a node broadcasts data in response to a request from a neighbor using the same primary channel, some neighbors that also need the same data may be in its secondary channels and thus cannot utilize overhearing as effectively as Seluge.

IR-Seluge-FixedInitPCh shows better performance than IR-Seluge-RandomInitPCh. In all experiments, IR-Seluge-FixedInitPCh introduces on average 45% longer average completion time, 53% more request and data packets, and 45% more advertisement packets than Seluge. In IR-Seluge-FixedInitPCh, every node stays in the same initial primary channel when dissemination begins. Therefore, the overhead is closer to Seluge than IR-Seluge-RandomInitPCh.



(a) Avg. completion time (sec)



(b) Number of message transmissions

Figure 2. Average completion time and communication overhead of IR-Seluge and Seluge in scenario 1.

These results indicate that IR-Seluge degrades the performance of Seluge when there is no noise source. However, the next evaluation will show that IR-Seluge can effectively survive severe interference that completely disables Seluge.

7.2. Scenario 2—Interference on a Single Channel

In this scenario, we compare Seluge and IR-Seluge when a single channel is under severe interference. We use four nodes (nodes 21, 24, 31, and 63) close to line 1 in Figure 1 as noise sources, which divides our testbed into two halves. Every noise source interferes with channel 26. We first run Seluge for one hour to see how far the dissemination flow can reach. As expected, Seluge's dissemination flow is completely blocked by the noise sources. Only the nodes to the right of line 2 in Figure 1 finish receiving the entire code image, but no nodes to the left of line 2 receive even a single page.

In contrast, both IR-Seluge-RandomInitPCh and IR-Seluge-FixedInitPCh successfully bypass the interference

and every node in the testbed completely receives the entire code image. Table 2 shows the performance overhead of our scheme in this scenario. IR-Seluge-FixedInitPCh introduces more overhead compared to the case of no noise sources. IR-Seluge-FixedInitPCh introduces about 1.4 times more delay, 0.8 times more request and data packets, and 1.2 times more advertisements than in the case of no noise sources. However, IR-Seluge-RandomInitPCh overcomes such interference without introducing extra overhead compared with the case of no noise sources.

Table 2. Average completion time and communication overhead of IR-Seluge in scenario 2 (Seluge is unable to disseminate beyond line 2 in Figure 1.).

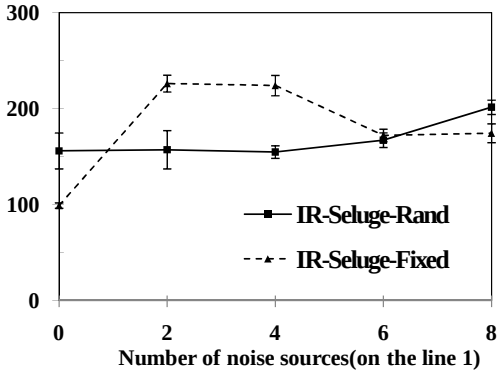
	IR-Seluge-RandomInitPCh	IR-Seluge-FixedInitPCh
Avg. completion time (sec)	147.62	235.51
Total number of request and data packet transmissions	15485.63	21512.45
Total number of advertisement packet transmissions	7462	12341.67

IR-Seluge-RandomInitPCh shows less performance overhead than IR-Seluge-FixedInitPCh in this scenario. This is because with IR-Seluge-FixedInitPCh the initial primary channels of all nodes are not available for the interference and it takes some efforts for these nodes to realize the unavailability of the primary channel and migrate to other channels. In contrast, given 16 channels, with IR-Seluge-RandomInitPCh only about 1/16 of the nodes have been affected by the interference on their initial primary channels.

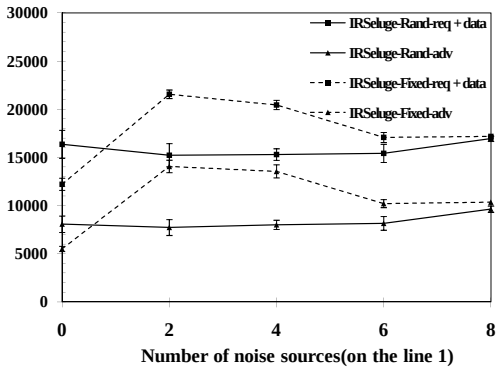
7.3. Scenario 3—Interference on Multiple Channels

In this scenario, we examine the performance of IR-Seluge when there is severe interference on multiple channels in a region of the network. We place 2 to 8 noise sources close to line 1 in Figure 1, and perform experiments in four cases: (1) nodes 21 and 24 as 2 noise sources; (2) nodes 21, 24, 31 and 63 as 4 noise sources; (3) nodes 21, 24, 31, 63, 20, and 25 as 6 noise sources; and (4) nodes 21, 24, 31, 63, 20, 25, 22, and 30 as 8 noise sources. Different from scenario 2, each noise source interferes with a different channel ranging from channel $(26 - \#noisesources + 1)$ to 26. We use 30K code image in the experiments.

Figures 3(a) and 3(b) show the average completion time, the number of advertisement packets, and the number of request and data packets sent in all the experiments.



(a) Avg. completion time (sec)



(b) Number of message transmissions

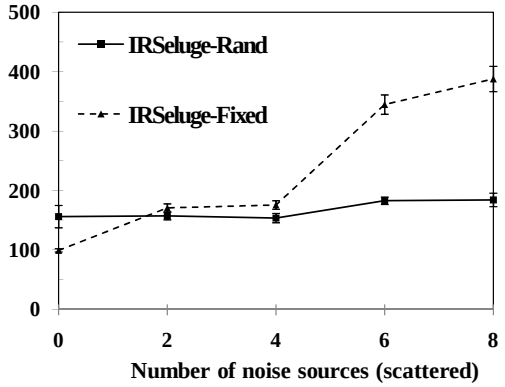
Figure 3. Average completion time and communication overhead of IR-Seluge in scenario 3.

In both figures, IR-Seluge-RandomInitPCh shows slightly increasing overhead as the number of noise sources increases. In the worst case of 8 noise sources, IR-Seluge-RandomInitPCh introduces about 29% longer average completion time, 3% more request and data packets, and 19% more advertisement packets than the case of no noise sources.

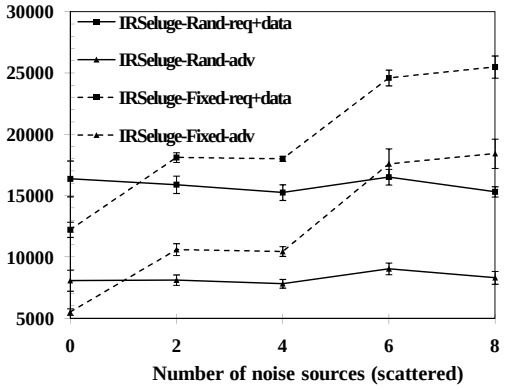
In both figures, IR-Seluge-FixedInitPCh shows higher overheads when there are 2 and 4 noise sources than the other cases. When the number of noise sources becomes 6 and 8, the overheads decrease. Our further investigation indicates that the overhead is sensitive to the location and the channel that each noise source interferes. Thus, there is no clear trend based on the number of noise sources. Compared to the case of no noise sources, IR-Seluge-FixedInitPCh shows about 100% longer average completion time, 119% more advertisement packets, and 56% more request and data packets in the worst case.

7.4. Scenario 4—Interference on Multiple Channels in Multiple Regions

In this scenario, we evaluate the performance of IR-Seluge when there is serious interference on multiple channels in multiple regions. We scatter 2 to 8 noise sources in various regions of the entire testbed, and perform the experiments in the following four cases: (1) nodes 13 and 50 as 2 noise sources; (2) nodes 13, 50, 3, and 48 as 4 noise sources; (3) nodes 13, 50, 3, 48, 24, and 61 as 6 noise sources; and (4) nodes 13, 50, 3, 48, 24, 61, 35, and 66 as 8 noise sources. Every noise source interferes with a different channel ranging from channel $(26 - \#noisesources + 1)$ to 26. As in scenario 3, we use the 30K code image.



(a) Avg. completion time (sec)



(b) Number of message transmissions

Figure 4. Average completion time and communication overhead of IR-Seluge in scenario 4.

Figures 4(a) and 4(b) show the average completion time and communication overhead in these experiments. In all cases, IR-Seluge-RandomInitPCh requires almost no extra overhead compared to the case of no noise sources. In the most severe case of 8 noise sources, the average completion time is only 18% longer than the case of no noise sources. The overhead for advertisement packets and that for request and data packets are almost same as in the case of no noise sources.

However, IR-Seluge-FixedInitPCh shows larger overhead than IR-Seluge-RandomInitPCh when there are noise sources. The average completion time of IR-Seluge-FixedInitPCh is about 56% longer than that of IR-Seluge-RandomInitPCh. In terms of communication overhead, IR-Seluge-FixedInitPCh introduces on average 37% more request and data packets and 70% more advertisement packets than IR-Seluge-RandomInitPCh.

Based on the experimental evaluation, we can conclude that IR-Seluge can effectively mitigate the effects of radio interference. In contrast, Seluge fails when its dissemination channel is under serious interference. Among the two variations, having a random initial primary channel performs better than having a fixed initial primary channel, particularly when serious interference exists in the initial primary channel. However, when there is no severe interference, IR-Seluge does introduce higher overhead than Seluge.

8. Related Work

Code dissemination is a critical issue to enable efficient tasking of wireless sensor networks. Several code dissemination protocols (e.g., [5, 8, 20, 10, 18, 16]) have been developed to propagate new code images using the ad-hoc wireless network formed by sensor nodes. Among these options, Deluge [8] has been widely used after being included in recent TinyOS distributions.

The integrity and availability of code dissemination are crucial in ensuring the success of code dissemination. Several groups of researchers have developed a number of schemes and systems to address these issues. Examples include Sluice [11], Secure Deluge [7], the approach in [6], and Seluge [9], all of which are secure extensions to Deluge. In particular, Seluge provides not only integrity protection of disseminated code images, but also resilience against DoS attacks that exploit the code dissemination protocol. In addition, Seluge does not harm any efficient dissemination mechanisms in Deluge. For such reasons, Seluge has been used as the foundation of the work presented in this paper.

Several protocols (e.g., [22, 19, 21, 13]) have been developed to improve the performance of code dissemination by exploiting the channel diversity available on the current sensor platforms. The objective of these protocols is to improve the efficiency of code dissemination, through a dedicated control channel and multiple concurrent dissemination channels.

However, all existing approaches can be disabled by strong interference in only a single channel (e.g., the control channel). The research in this paper complements these approaches by significantly improving the resilience against

wireless interference.

9. Conclusion

In this paper, we presented the design, development, and evaluation of an interference-resilient code dissemination system named IR-Seluge for wireless sensor networks. IR-Seluge is developed as an interference-resilient extension to Seluge [9]. Besides the security protections provided by Seluge, IR-Seluge offers additional resilience properties against wireless interference: As long as one or more channels are available, IR-Seluge can still propagate code images to the nodes in a region where severe wireless interference exists.

Our future research will be focused on optimizing the channel switching strategy so that IR-Seluge can offer the same interference resilience with lower overhead.

References

[1] MICAz: Wireless measurement system. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`.

[2] TelosB mote platform. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`.

[3] Crossbow. Tinyos getting started guide. `http://www.xbow.com/support/support_pdf_files/getting_started_guide.pdf`.

[4] Crossbow Technology Inc. MICA2 data sheet.

[5] Crossbow Technology Inc. Mote in-network programming user reference, 2003.

[6] J. Deng, R. Han, and S. Mishra. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06)*, April 2006.

[7] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler. Securing the deluge network programming system. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06)*, April 2006.

[8] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, November 2004.

[9] S. Hyun, P. Ning, A. Liu, and W. Du. Seluge: Secure and dos-resistant code dissemination in wireless sensor networks. In *Proceedings of the Seventh International Conference on Information Processing in Sensor Networks (IPSN '08)*, pages 445–456, April 2008.

[10] S. Kulkarni and L. Wang. MNP: multihop network reprogramming service for sensor networks. In *Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS '05)*, pages 7–16, June 2005.

[11] P. Lanigan, R. Gandhi, and P. Narasimhan. Sluice: Secure dissemination of code updates in sensor networks. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS '06)*, July 2006.

[12] 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 the 1st Symposium on Network System Design and Implementation (NSDI '04)*, March 2004.

[13] C. M. Liang, R. Musaloiu-E, and A. Terzis. Typhoon: A reliable data dissemination protocol for wireless sensor networks. In *Proceedings of the Fifth European conference on Distributed Computing in Sensor Systems (EWSN '08)*, January 2008.

[14] A. Liu, P. Ning, and C. Wang. Lightweight remote image management for secure code dissemination in wireless sensor networks. In *Proceedings of 2009 IEEE INFOCOM*, April 2009.

[15] R. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Apr 1980.

[16] V. Naik, A. Arora, P. Sinha, and H. Zhang. Sprinkler: A reliable and scalable data dissemination service for wireless embedded devices. In *Proceedings IEEE International Real-Time Systems Symposium*, pages 277–286, December 2005.

[17] P. Ning, A. Liu, and W. Du. Mitigating DoS attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks*, 4(1):1–35, February 2008.

[18] N. Reijers and K. Langendoen. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA '03)*, pages 60–67, September 2003.

[19] R. Simon, L. Huang, E. Farrugia, and S. Setia. Using multiple communication channels for efficient data dissemination in wireless sensor networks. In *Proceedings of the Second International Conference on Mobile Ad-hoc and Sensor Networks (MASS '05)*, November 2005.

[20] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report CENS-TR-30, UCLA, Center for Embedded Networked Computing, November 2003.

[21] L. Wang and S. S. Kulkarni. Gappa: Gossip based multichannel reprogramming for sensor networks. In *Proceedings of the Second International Conference on Distributed Computing in Sensor Systems (DCOSS '06)*, June 2006.

[22] W. Xiao and D. Starobinski. Poster abstract: Exploiting multi-channel diversity to speed up over-the-air programming of wireless sensor. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys '05)*, November 2005.

[23] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*, pages 1–13, Nov 2003.

[24] G. Zhou, J. Stankovic, and S. Son. The crowded spectrum in wireless sensor networks. In *Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets '06)*, May 2006.