SIMAGE: **S**ecure and Link-Quality Cognizant ***Image*** Distribution for Wireless Sensor Networks

Ramalingam K. C., Venkatachalam Subramanian, A. Selcuk Uluagac and Raheem Beyah
Communications Assurance and Performance Group
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332, USA
{ramalingam.chandrasekar, venkat.subbu}@gatech.edu
{selcuk, rbeyah}@ece.gatech.edu

***Abstract*—Wireless sensor networks (WSNs) are used in a range of critical domains (e.g., health care, military, critical infrastructure) where it is necessary that the nodes be reprogrammed with a new or modified code image without removing them from the deployment area. Various protocols have been developed for the dissemination of code images between sensors in multi-hop WSNs, where these sensor nodes may have varying levels of link quality. However, the code dissemination process in these protocols is hindered by the nodes with poor link quality. This results in an increased number of retransmissions and code dissemination time. Also, in several of the techniques, the code dissemination process is not secure and can be eavesdropped or disrupted by a malicious wireless sensor node in the transmission range. In this paper, we propose a simple approach, *Secure and Link-Quality Cognizant Image Distribution (SIMAGE)*, to enhance the existing code dissemination protocol using the available resources in the sensors. Specifically, our approach adapts to the varying link conditions via dynamic packet sizing to reduce the number of retransmissions and overall code dissemination time. Our approach also provides confidentiality and integrity to the code dissemination process by utilizing energy-efficient encryption and authentication mechanisms with RC4 and the CBC-MAC. We have evaluated SIMAGE in a network of real sensors and the results show that adjusting the packet size as a function of link quality reduces the retransmitted data by 93% and the image transmission time by 35% when compared to the existing code dissemination protocols. The trade-offs between reliability, security overhead, and overall transmission time for SIMAGE are also discussed.**

Index Terms - Wireless Sensor Networks, Secure Code Dissemination, Link Quality Indicator (LQI)

I. Introduction

Sensors are deployed in various locations such as underwater, volcano regions and underground, making them hard to recover or replace. Due to this diverse deployment of wireless sensor networks (WSNs) and many functionalities provided by them, reprogramming the deployed sensor nodes through wireless links becomes a necessary and desirable task. For instance, if software running on the node requires an update as a result of a security patch or additional functionality, it would be necessary to replace the existing code on the node with the new updated code. This process of disseminating code over wireless links is called *code dissemination*. Moreover, under hostile conditions, an attacker might try to modify or get access to disseminated code. The attacker could also attack the dissemination process itself by injecting malicious code dissemination packets to exhaust the limited energy of the sensor nodes. Therefore, it is imperative that the dissemination process be secured.

In addition to security, code dissemination protocols should address the variation in link quality between nodes. The nodes with poor link quality would hinder other nodes in the network from proceeding forward in the code dissemination process. Moreover, such nodes with poor links would yield increased retransmissions which would further degrade the security of the system as an attacker would have multiple opportunities to intercept the disseminated packets. In this paper, we introduce a *Secure and Link-Quality Cognizant Image Distribution (SIMAGE)* technique for WSNs which aims at reducing the number of retransmitted code dissemination packets. SIMAGE also provides energy efficient security services, confidentiality using the RC4 [1], [2] encryption algorithm and integrity using the CBC-MAC provided by the CC2420 [3] transceiver module of the sensors (e.g., MICAz and TelosB). The results from our experiments using real sensors show that code dissemination using SIMAGE is more efficient than other existing code dissemination protocols, in terms of code dissemination time and total number of retransmissions. The experimental results show that SIMAGE reduces the retransmission bytes by 93% and the image transmission time by 35% when compared with the existing code dissemination protocols under poor link quality conditions.

The contributions of our work are three-fold: 1) We design and implement a secure code dissemination protocol - SIMAGE; 2) we consider link quality between nodes during the dissemination process; and 3) we dynamically adapt the packet size based on link quality. As a result, our scheme outperforms previously proposed techniques.

The paper proceeds as follows: existing code dissemination protocols for WSNs are discussed in Section II. To motivate our work, an analysis of the problems encountered by the existing code dissemination protocol is provided in Section III. Section IV discusses the proposed scheme, SIMAGE, and explains its implementation in detail. The performance

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

evaluation of SIMAGE and comparison of SIMAGE with the existing code dissemination protocols are shown in Section V. Section VI summarizes the benefits of SIMAGE over existing code dissemination protocols and concludes the paper.

II. Related Work

Deluge [4] is the most widely used code dissemination protocol and is included in recent TinyOS distributions [5]. Deluge is a *page-by-page* code dissemination protocol proposed for WSNs. The base node advertises the new version of code and receiver node in response sends a request message for the new version. Next, the base station broadcasts the first page of the code image to the receiver nodes, with fixed sized data packets. Once all receiver nodes have received all the packets corresponding to that page, the next page transmission begins. In case a receiver node has lost some packets in the first page, all the lost packets will be retransmitted prior to the transmission of the next page. The code dissemination is not secure and can be eavesdropped on by an attacker. Moreover, as Deluge does not consider the link quality between nodes, the retransmissions between nodes with poor link quality will severely delay the propagation of the code image.

The *Secure Network Programming Protocol* [6] uses public-private key encryption to sign the advertisement packets of the code image and SHA-1 for computing the hash of each packet. In this work, the computed hash of one future packet is embedded into the payload of the previous packet in sequence and the hash of the first packet is embedded into the advertisement packet which is signed. So, a receiver node, upon authenticating the advertisement packet, verifies the integrity of the next received packet immediately and saves or discards the packet based on the computed hash. However, in the case of out-of-order delivery, the receiver needs to cache the packet and wait for the previous packet in sequence to verify the cached packets. This can be used by the attacker to inject fake packets and deplete the cache memory of the receiver node resulting in a denial-of-service attack.

Seluge [7] is essentially a secure version of Deluge and therefore, it preserves the page-by-page propagation method of Deluge. Similar to [6], a hash of each packet in a page is computed using SHA-1 and this value is embedded in the correspondingly sequenced packet of the previous page. So, once all the packets in a page are received, the receiver has all the hash values for the packets in the next page. Seluge also uses the *Elliptic Curve Digital Signature Algorithm* (ECDSA) to sign the initial code image advertisement. However, the SHA-1 and the ECDSA algorithm used are time consuming processes [8]–[10]. Note [6] and [7], similar to [4], do not consider the link-quality between nodes in the network.

Another recent work on the code dissemination process in WSNs, *DiCode* [11], provides a secure and distributed code dissemination protocol. As opposed to the aforementioned centralized approaches [4], [6], [7], the key difference in this technique is the usage of distributed control for code dissemination. However, the security features used in this approach are similar to that of Seluge. The experimental results also indicate that DiCode has longer propagation delays when compared to Deluge and Seluge. Although [11] proposes a distributed code dissemination protocol, it does not address any of the inter-node link quality issues.

Nonetheless, as discussed in [12] with experimental results, link quality prediction is important for better system provisioning and resource management. Moreover, the experiments conducted on IEEE 802.15.4 Zigbee radios with the CC2420 chipset illustrate that the *Link Quality Indicator (LQI)* is linear with respect to the instantaneous SNR. Finally, it is shown that LQI can be used as an effective parameter to predict instantaneous link quality between nodes [12].

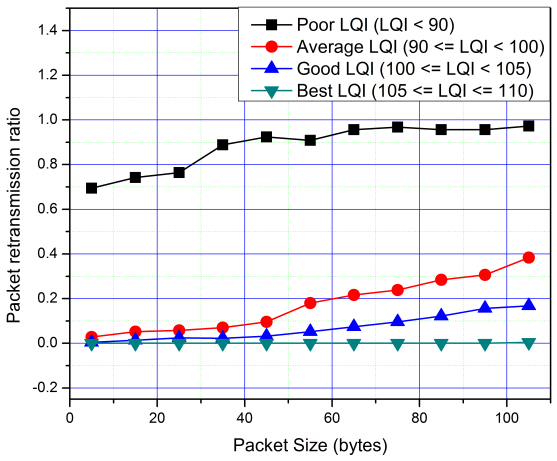
Therefore, in this paper, we propose SIMAGE, which provides *secure* and *link-quality aware* code image dissemination for WSNs.

III. Background and Motivation

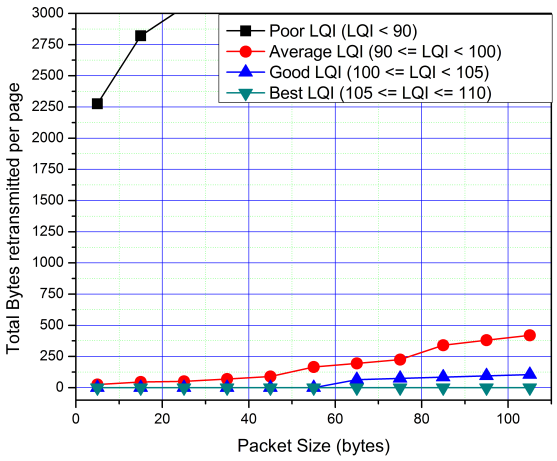
A common problem in many wireless networks is varying and poor link quality of the channel between the communicating nodes [12]. During communication, a reduction in the link quality increases the number of retransmissions between the nodes to transfer useful information. The increased retransmissions have a significant impact on WSNs which consists of resource-constrained sensor nodes. The code dissemination protocol in TinyOS, Deluge [4], uses a fixed packet size to transmit the code image between the wireless sensor nodes and is susceptible to increased packet loss under poor link quality conditions. For the same code image size, due to the increased packet loss rate, the code dissemination protocol in TinyOS [4] will need more retransmission bytes to disseminate the code image as analyzed in Section V. This, in turn will increase the total code dissemination time between the nodes.

The code dissemination protocol included in TinyOS distributions is a page-by-page transmission protocol, where a code image is split into pages of 1024 bytes. Each page is divided into an equal number of packets with a packet size value set by the user. If the base station has a newer version of the code image, it advertises this information to the network. Upon receiving the advertisement, nodes in the network send a request for a particular page. If there is packet loss during the transfer of the page from the base station, the receiver makes a request for the lost packets until all of the packets are received. For instance, consider that there are two receiver nodes which are one hop away from the base station and the first node has a channel with good link quality and second node has a channel with poor link quality. Then, during the dissemination process from the base station, the first node receives all the packets with less retransmission rounds when compared to the second node. The first node, before sending a request message for the next page, has to wait until the second node receives all the packets of the current page. Thus, the code dissemination time for the entire network will be affected by the increased retransmissions of the node with the poor link.

In order to analyze this, we conducted a simple preliminary experiment to calculate the packet retransmission rate for various packet sizes with different link quality conditions



(a) Packet Retransmission Ratio (ψ) vs. Packet Size



(b) Number of bytes retransmitted per page vs. Packet size

Fig. 1. Experimental results of Packet Retransmission Ratio (ψ) and number of bytes retransmitted per page

between a pair of MICAz motes. The link quality of the channel is estimated using the LQI value in the MICAz mote.

In the experimental setup, the first mote transmits 50 packets and the number of retransmissions is calculated. The experiment is repeated 10 times, with the same packet size and link quality condition. The average retransmission per 50 packets is calculated and the average Packet Retransmission Ratio (ψ) is calculated with the average retransmission value using Equation 1, where, φ is the number of retransmitted packets and τ is the total number of packets transmitted, including the retransmitted packets:

$$\psi = \frac{\varphi}{\tau} \tag{1}$$

The procedure is repeated to calculate the ψ for various packet sizes and various link quality values. Figure 1(a) shows the variation of ψ for an increase in packet size for four different LQI ranges. The graph shows that for a poor LQI range, value of ψ worsens as the packet size increases and becomes nearly equal to one. For the average LQI range, the value of ψ increases with an increase in packet size. For the good and best LQI ranges, the ψ value remains nearly equal to zero for a packet size increase up to 45 bytes. However, beyond 45 bytes the value of ψ for the good LQI range increases steadily while the value of ψ remains zero for the best LQI range.

Using the ψ value, computed from the experiment for various packet sizes and LQI values, the number of packets retransmitted per page can be calculated using Equation 2:

$$\phi = \sum_{i=o}^{\gamma-1} (\psi * n)(\psi)^i \tag{2}$$

where, ϕ is the cumulative number of retransmitted packets per page, γ is the total number of retransmission requests needed to transmit the entire page of size 1024 bytes, which can be computed using Equation 3:

$$\frac{d(n * \psi^{(\gamma+1)})}{d\gamma} = 0 \tag{3}$$

In Equation 3, γ is the minimum integer value, starting from zero, that satisfies the equation and n is the number of packets per page. Figure 1(b) shows the number of retransmitted bytes per page in the code dissemination protocol for various packet sizes in different LQI ranges. The figure shows that for an increase in packet size up to 55 bytes, the retransmitted bytes per page remains zero for the good and best LQI ranges. However, when the packet size exceeds 55 bytes, the retransmissions increase gradually for the good LQI range, while it remains zero for all packet sizes in the best LQI range.

Since transmitting a page with minimum retransmissions reduces the code dissemination time, to achieve better code dissemination under fluctuating link quality conditions, we propose a LQI based adaptive packet size code dissemination protocol. This protocol samples the LQI of all the receivers and determines the optimal packet size for disseminating a page. The details of the scheme is explained in Section IV.

IV. Proposed Scheme

Our technique, SIMAGE, which builds upon the code dissemination protocol of TinyOS, Deluge [4], was implemented and tested in a network of MICAz sensor motes. SIMAGE has two specific features. First, is the dynamic, adaptive packet size estimation, which determines an optimal packet size for transmitting a page using the LQI values of the request messages. Second, it provides security for code dissemination with energy efficient, stream cipher RC4 encryption [2] and integrity using the CBC-MAC. The details of the implementations are explained below.

A. Dynamic Adaptive Packet Size Algorithm

The experiments in Section III illustrate that different LQI ranges have different optimal packet sizes. So, using an adaptive packet size in the code dissemination process is instrumental to reduce the per page dissemination time. Hence, from the experiment in Section III, the optimal data packet size values for different LQI ranges have been determined and are shown in Table I. Note that the data packet size in

TABLE I
OPTIMAL PAYLOAD SIZE FOR DIFFERENT LQI RANGES

LQI	LQI Value Range	Optimal Payload Size (Bytes)
Poor LQI	< 90	5
Average LQI	90 - 99	20
Good LQI	100 - 104	40
Best LQI	105 - 110	80

Algorithm 1 Receiver Node

```
for each received data message do
  if DataPktSize != CurrPktSize then
    modify_parameters()
  end if
end for

function modify_parameters()
    δ ← DataPktSize/CurrPktSize
    CurrPktSize ← DataPktSize
    NewBitvectorSize ← CurrPktSize/δ
    NewBitvector ← Bitvector(δ)
end function
```

the table does not include the 7 byte data header overhead and the security header overhead.

The link quality between nodes in the network is measured using the LQI provided by the CC2420 transceiver module of the sensors. When the base station receives the request message, it stores the source ID and the LQI values of the requester node. While sending the first data message, the base station computes the average LQI of all the requester nodes and this gives a estimation of the link quality of the channel around the base station. Based on the measured average LQI, the size of the packets in the corresponding page is dynamically adapted to reduce the number of retransmissions involved. The algorithm for adaptive packet-sizing used in the receiver and transmitter nodes are shown in Algorithm 1 and Algorithm 2 respectively.

Algorithm 2 Transmitter Node

```
for each request message receieved do
  lqi ← LQI(RcvdPkt)
  map⟨src_id, lqi⟩ lqi_map ← (SrcId, lqi)
end for
for first data message transmission do
  if retransmission then
    compute ρ(lqi_map)
    DataPktSize ← DataPktSize(ρ)
    modify_parameters()
  else if not retransmission then
    compute ρ(lqi_map)
    MA ← ρ * sf + (1 − sf) * MA
    DataPktSize ← DataPktSize(MA)
    modify_parameters()
  end if
end for
```

In the transmitting node, if it is a first round of data transmission for a page, the *moving average (MA)* is calculated as a function of the *average LQI* (ρ) and *scaling factor (sf)* as the following:

$$MA = \rho * sf + (1 - sf) * MA \tag{4}$$

In Equation 4, the scaling factor (*sf*) is chosen such that the current LQI sample is given more weight than the previous LQI samples and any temporary fluctuation in the current LQI sample does not immediately affect the packet size estimation. Hence, the scaling factor (*sf*) is given a value of 0.4 based on our experiments. Note, for values less than 0.4 the moving average took longer to converge to an optimal LQI estimate, whereas, for values greater than 0.4 there are increased fluctuations in the LQI estimate. The data message packet size (*DataPktSize*) is determined by the estimated moving average, according to the data mapping in Table I. In the case of retransmission, only the nodes with poor LQI will request more packets. So to reduce packet losses, the *DataPktSize* during retransmission is determined according to the current average LQI (ρ) estimate instead of the moving average.

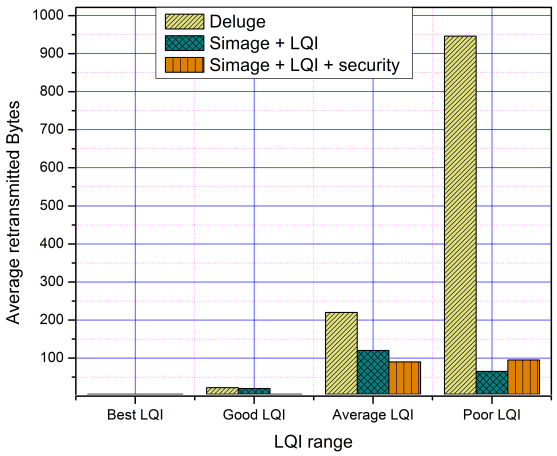
$$\delta = \frac{New\ packet\ size}{Current\ packet\ size} \tag{5}$$

After determining the new packet size, the *multiplication factor* (δ) is calculated using Equation 5. If the new packet size is greater than the current packet size, then δ number of current packets will be combined together into a single packet and transmitted. If the new packet size is less than the current packet size, then the current packet will be split into $\frac{1}{\delta}$ number of packets and transmitted. In SIMAGE, the current packet can be divided into smaller packets or, two or more current packets can be combined into a single packet. To facilitate this, the payload size value for different LQI ranges in Table I are chosen such that the bigger packet size values are a multiple of all other smaller packet size values. After determining the new packet size, the transmitter communicates it to the receiver using the data packet header. So, there is an one byte overhead in the data packet in our scheme when compared to the code dissemination protocol in TinyOS, Deluge [4].

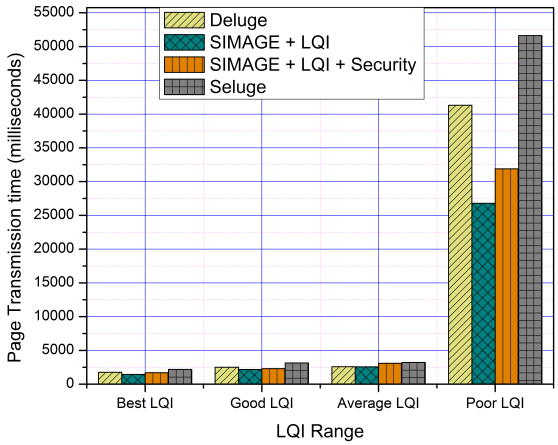
On receiving the first data packet of a page, the receiver will check for a change in packet size. If the new packet size value is greater or less than the current packet size, the receiver computes the δ value and populates the new bit-vector, which consists of the packet identifiers of packets to be transmitted, based upon the packets requested in the old bit-vector and δ value. After updating the new bit-vector, it checks whether the packet number of the received packet is requested by this node. If it was requested by this node, it stores the data, otherwise, it discards it.

B. Secure Dissemination

The three different types of messages involved in the code dissemination process are *advertisement messages*, *request*



(a) Average retransmitted bytes for various LQI ranges



(b) Page transmission time for various LQI ranges

Fig. 2. Experimental results of average retransmitted bytes and page transmission time

messages and *data messages*. Securing each of these message types protects against different types of attacks. For instance, securing only the data messages and not providing integrity for the request messages or advertisement messages will enable an attacker to perform a denial-of-service attack by flooding the nodes with request messages or advertisement messages, respectively. In SIMAGE, we ensure an increased level of security by encrypting all three types of messages using RC4 encryption and provide integrity using the CBC-MAC. The CBC-MAC integrity check is fast as it is provided by the CC2420 hardware module. We use a 64-bit key for CBC-MAC which adds only 8 bytes of overhead to the overall payload. Whereas, RC4 encryption uses a 128-bit key and being a stream cipher technique, it performs an in-place encryption and does not add any data header overhead.

V. Performance Analysis

In this section, we evaluate the performance of SIMAGE (with and without the security) under varying link quality conditions and compare it to the existing code dissemination protocols for TinyOS - Deluge [4] and Seluge [7]. Two different experiments were performed in order to analyze and expose the benefits of SIMAGE:

Experiment 1: An experimental setup consisting of two MICAz sensor motes is used. A binary image of 35,276 bytes is transferred between the nodes using the default code dissemination protocol (Deluge) [4], SIMAGE without security and SIMAGE with security. The three techniques mentioned above are compared with respect to the average number of retransmitted bytes per page and the average transmission time per page. This experiment is performed for different link quality conditions.

Figures 2(a) and 2(b) illustrate the results obtained from the first experiment. As seen in Figure 2(a), for the best LQI range, the default code dissemination protocol [4], SIMAGE with security and SIMAGE without security do not exhibit any retransmissions. Similarly, for the good LQI range, only a small amount of retransmissions are encountered by all three techniques. As the link quality decreases, Deluge experiences increased retransmissions as compared to SIMAGE with and without security. Furthermore, at very poor link quality conditions, Deluge exhibits a steep increase in the number of bytes retransmitted, whereas SIMAGE with and without security maintains an almost steady number of retransmitted bytes. Moreover, under such low link quality cases, the number of bytes retransmitted using SIMAGE without security is less than Deluge because the packet size is dynamically reduced to a minimal value. It can also be seen that SIMAGE with security has a higher number of retransmitted bytes than SIMAGE without security due to the fact that the former experiences additional security overhead for each packet transmitted.

Figure 2(b) depicts the same experiment illustrating the variations in the average time taken per page transfer. Under very good link quality conditions, SIMAGE without security proves to provide much faster page transmission than Deluge as larger packets are transmitted with relatively small packet loss rate. SIMAGE with security also proves to be faster than Deluge. However, SIMAGE with security becomes slower than SIMAGE without security due to the additional security overhead in each packet. As the link quality condition degrades, the time taken per page transfer by all three techniques tends to increase. This can be attributed to the fact that the number of retransmissions also increase. In the average link quality condition, Deluge and SIMAGE have almost equal page transfer times. This is due to the fact that under average link quality conditions, both protocols transmit packets of almost equal sizes (Deluge has 23 bytes and SIMAGE has 20 bytes). Again, due to the security overhead in each packet, SIMAGE with security takes slightly longer to transfer a page than the other two schemes. When the link quality further deteriorates and reaches the poor link quality condition, Deluge suffers a large number of retransmissions and takes a very long time to transfer a page. Whereas, it can be seen that SIMAGE with and without security transfers a page in a relatively short time. For performance reasons, Deluge keeps the radio switched on during code dissemination [4] and any

decrease in the code dissemination time will reduce the energy consumed by all the nodes in a network. So, reduction in the per page transmission time by SIMAGE, in turn reduces the energy consumption of the nodes and proves that SIMAGE is more energy efficient than Deluge.

Furthermore, Seluge [7], consumes more time to do the integrity check using the SHA-1 algorithm. SHA-1 takes around 15 milliseconds to compute the hash of a packet and the inter-packet transmission time is increased from 2 to 17 ms in order to accommodate the computation time [7], which will increase the per page transmission time. Using the performance results from Seluge [7], a comparison of per page transmission time is shown in Figure 2(b). SIMAGE with security outperforms both Deluge and Seluge under poor link quality conditions.

Experiment 2: The second experiment consisted of three levels of sensor nodes with two MICAz sensor motes at each level, where each level is seperated by a distance of one hop. While the previous experiments show the benefits of SIMAGE under various link quality conditions, the primary motive of this experiment was to prove the effectiveness of SIMAGE over the default code dissemination protocol [4] under very good link quality conditions as well. The time taken by the node in the last level to receive the entire binary image (35,276 bytes) is evaluated for both the default code dissemination protocol [4] and SIMAGE.

The results obtained from the second experiment are presented in Table II. The total time taken for the binary image transfer from source node to intermediate nodes and from source node to sink nodes are evaluated. All the nodes in the experiment are maintained under very good link quality conditions such that the retransmissions are largely reduced. This experiment reiterates that SIMAGE performs better than Deluge in terms of total transmission time for the entire binary image even under good link quality conditions. This also illustrates that SIMAGE makes better use of the channel when the link quality between nodes is good, whereas, Deluge does not make optimum use of the channel under good link quality conditions.

TABLE II
CODE DISSEMINATION TIME TAKEN BY DELUGE AND SIMAGE IN A MULTI-HOP NETWORK

Protocol	Node	Total Image Transfer Time (seconds)
Deluge	intermediate	147.492
	sink	296.379
SIMAGE	intermediate	96.572
	sink	198.216

VI. CONCLUSION

Wireless sensor nodes are always deployed in groups and the link quality between different pairs of nodes is not the same in the deployed area. Also, the code is disseminated hop-by-hop in a multi-hop WSN, where the delay in disseminating the code between the nodes with poor link quality hinders the overall code dissemination time. Our experimental results with real sensors show that using a fixed packet size in the current code dissemination protocols will increase the retransmissions between the nodes with poor link quality and hinder the overall code dissemination time. Our code dissemination protocol SIMAGE, samples the link quality of the channel using LQI as a metric and determines the optimal packet size before transmitting a page. The performance analysis of SIMAGE in the experiments show that by dynamically adapting the packet size, retransmission bytes are reduced by 93% and the per page transmission time is reduced by 35% for poor LQI values. Results also show that SIMAGE, during the good link quality condition, exploits the channel and disseminates the code faster when compared to the other code dissemination protocols.

SIMAGE with security uses the simple energy efficient stream cipher encryption algorithm RC4 [2] and a hardware based hashing function CBC-MAC for providing secure code dissemination. Since SIMAGE uses a dynamic adaptive packet size technique, the overhead of providing secure dissemination is reduced when compared to Seluge [7]. Our experimental results also show that SIMAGE with security outperforms the TinyOS code dissemination protocols Deluge [4] and Seluge [7] under the poor LQI ranges.

In our future work, we will improve the performance of SIMAGE in the average LQI ranges, where it performs similar to the default TinyOS code dissemination protocol. Also, calculating the average LQI value at both the transmitting and receiving ends will yield a better link quality estimate in a bidirectional channel.

VII. ACKNOWLEDGEMENTS

This work is partially supported by NSF Grant #CNS-1052769.

REFERENCES

[1] B. A. Forouzan, *Cryptography and Network Security (1st edition)*. McGraw-Hill, 2007.
[2] A. S. Uluagac, R. A. Beyah, Y. Li, and J. A. Copeland, "Vebek: Virtual energy-based encryption and keying for wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, 2010.
[3] "CC2420 datasheet," http://focus.ti.com/lit/ds/symlink/cc2420.pdf.
[4] 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*. ACM Press, 2004.
[5] "Tinyos documentation," http://docs.tinyos.net/.
[6] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *Proceedings of 5th IPSN*, 2006.
[7] S. Hyun, P. Ning, and A. Liu, "Seluge: Secure and dos-resistant code dissemination in wireless sensor networks," in *Proceedings of the 7th IPSN*, 2008.
[8] M. Passing and F. Dressler, "Experimental performance evaluation of cryptographic algorithms on sensor nodes," in *IEEE MASS*, oct. 2006.
[9] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. G. Dean, F. Mueller, and M. L. Sichitiu, "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis." in *CASES*, 2003.
[10] "Ecdsa performance evaluation," http://discovery.csc.ncsu.edu/.
[11] D. He, C. Chen, S. Chan, and J. Bu, "Dicode: Dos-resistant and distributed code dissemination in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. PP, no. 99, 2012.
[12] G. Zheng, D. Han, R. Zheng, C. Schmitz, and X. Yuan, "A link quality inference model for ieee 802.15.4 low-rate wpans," in *IEEE GLOBECOM*, dec. 2011.