An Analytical Model for Coding-Based Reprogramming Protocols in Lossy Wireless Sensor Networks

Jun-Wei Li, Shi-Ning Li, *Member, IEEE*, Yu Zhang, *Member, IEEE*, Tao Gu, *Senior Member, IEEE*, Yee Wei Law, Zhe Yang, *Member, IEEE*, Xingshe Zhou, *Member, IEEE*, and Marimuthu Palaniswami, *Fellow, IEEE*

Abstract—Multi-hop over-the-air reprogramming is essential for remote installation of software patches and upgrades in wireless sensor networks (WSNs). Several recent coding-based reprogramming protocols have been proposed to enable efficient code dissemination in high packet loss environments. An accurate and formal analysis of the performance of these protocols, however, has not been studied sufficiently in the literature. In this paper, we present a novel high-fidelity analytical model based on the shortest path algorithm to measure the completion time by incorporating overhearing and packet coding. This model can be applied to any coding-based reprogramming protocol by substituting the coding part with protocol specific operations. We conduct extensive testbed experiments to evaluate the performance of our proposed model. Based on the analytical and numerical experiments, we find that 1) overhearing causes significant reduction of the completion time in dense wireless sensor networks, particularly, it reduces 50-70 percent of the total completion time when the packet reception rate is 0.896; 2) coding delay plays a key role in the total completion time compared to the communication delay when the packet coding parameters are selected appropriately, for example, the communication delay is about 65 percent of the coding delay when the number of packets per page is 16 for the finite field size 2^8; 3) the total completion time can be minimized when the number of packets per page is close to 24 and the finite field size is close to 2^4.

Index Terms—Reprogramming, code dissemination, network coding, analytical model, lossy wireless sensor networks

1 Introduction

Wireless sensor networks (WSNs) have been widely used to perceive and interact with the physical world for different purposes such as weather monitoring [1], forest surveillance [2], and healthcare assistance [3]. As application requirements evolve, the software on sensor nodes often needs to be updated for new versions or patched for bugs. Physical access to these nodes is usually restricted, and post-deployment or manually updating sensor nodes is prohibitively very costly. In this case, multi-hop over-the-air reprogramming presents an ideal solution.

Traditional reprogramming protocols–with Deluge [4] being the benchmark–suffer from steep performance degradation when the packet loss rate is high, as a result of interference, jamming, or natural environmental factors. Recently, the network coding-based protocols have been introduced to improve the *resilience* of reprogramming protocols to packet loss, i.e., when packet loss is substantial, the coding-based reprogramming protocols require significantly less time and energy to complete compared to the traditional reprogramming protocols. For example, when 20 percent of packets are lost, the coding-based protocol such as Rateless Deluge [5] requires about 76.4 percent of the time to complete reprogramming in a multi-hop linear network which consists of 16 sensor nodes [6]. The time cost of Rateless Deluge will be reduced further when the packet loss rate is higher than 20 percent.

Existing coding-based reprogramming protocols have demonstrated the efficiency of code dissemination through simulation or testbed, however simulations are neither accurate enough nor efficient while testbeds suffer from scalability issues. Compared to simulation and testbed, an analytical model is more rigorous and it has many benefits [7]. In addition, an analytical model is important because 1) it provides an economical alternative to labor-intensive simulations and field experiments for performance quantification; 2) it sheds valuable insights into the "mechanics" of these protocols, providing essential clues for improving their performance.

Such an analytical model aiming to accurately analyzing the performance of existing coding-based reprogramming protocols, however, has not been explored sufficiently in the literature. Previous models still have the following limitations: 1) their model ignores overhearing and thus provides no insight into the influence of overhearing on the total completion time; 2) their model ignores encoding delay which is neither realistic nor accurate as data packets are

- *J.-W. Li, S.-N. Li, Y. Zhang, Z. Yang, and X. Zhou are with the School of Computer Science, Northwestern Polytechnical University, Shaanxi, China. E-mail: {ljw, lishining, zhangyu, zyang, zhouxs}@mail.nwpu.edu.cn.*
- *T. Gu is with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3000, Australia. E-mail: tao.gu@rmit.edu.au.*
- *Y.W. Law is with the School of Engineering, the University of South Australia, Adelaide, SA 5001, Australia. E-mail: YeeWei.Law@unisa.edu.au.*
- *M. Palaniswami is with the Department of Electrical and Electronic Engineering, the University of Melbourne, Parkville, VIC 3010, Australia. E-mail: palani@unimelb.edu.au.*

Manuscript received 1 Jan. 2016; revised 24 Mar. 2016; accepted 17 Apr. 2016. Date of publication 28 Apr. 2016; date of current version 19 Dec. 2016. Recommended for acceptance by W. Wang.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2560805

usually encoded before transmission; 3) their model sets a fixed value for decoding delay. As a result, their model does not allow adjustment of packet coding parameters to optimize the encoding/decoding process, or reaching a trade-off between coding delay and communication delay to minimize total completion time. In comparison, existing studies [4], [8] on overhearing show that a node receives 3.35 times the number of data packets on average due to overhearing in a linear network. Our early experimental studies on Rateless Deluge (i.e., a coding-based protocol) also reveal similar findings in dense grid networks.

In this paper, we present a novel high-fidelity analytical model based on the shortest path algorithm to measure the completion time by incorporating overhearing and packet coding. Our model is able to quantify the effect of overhearing on the completion time in dense networks and optimize the completion time through analyzing packet coding parameters. The proposed model can be applied to general networks by using the shortest path algorithm, and can also be applied to any coding-based reprogramming protocol as we build it in the way that the coding part can be substituted. For example, we can replace the coding part with protocol specific operations, such as encoding and decoding in Rateless Deluge, SYNAPSE++ [8], and ReXOR [9], which makes our model flexible and extensible. In addition, the model is also compatible with the traditional reprogramming protocols when the coding part in our model is removed. We conduct comprehensive testbed experiments and the results show that the performance predicted by our model fits well with the testbed results. Compared to existing analytical approaches, our model is much more accurate because we find that overhearing causes significant reduction of the completion time in dense networks. In particular, it reduces 50-70 percent of the total completion time when the packet reception rate (PRR) is 0.896, validating our experimental studies. Hence, we characterize overhearing in our model to accurately reflect the actual process. Based on the analytical and numerical experiments, we also find that coding delay plays a key role in the total completion time compared to the communication delay when packet coding parameters (e.g., the finite field size and encoding coefficient) are appropriately selected. For example, the communication delay is about 65 percent of the coding delay when the number of packets per page is 16 for the finite field size 2^8. Moreover, the total completion time could be minimized when the number of packets per page is close to 24 and the finite field size is close to 2^4. These advantages will be lost if encoding delay is ignored or the decoding delay is set as a fixed value, especially in SYNAPSE++ where the *degree distribution* in packet coding must be carefully optimized [10]. Our model scales well as compared to existing analysis models, and the results of our analytical model match that of simulations with a very low prediction error (i.e., less than 8 percent).

This paper extends our preliminary work which appears in [11] in the following aspects: 1) we improve the existing algorithm for the multi-hop model; 2) we make our model extensible to any coding-based reprogramming protocol; 3) we discuss and characterize the overhearing between upstream and downstream nodes; 4) we conduct extensive testbed experiments to evaluate our model and analyze the coding delay to select appropriate packet coding parameters based on Rateless Deluge, a representative coding-based reprogramming protocol.

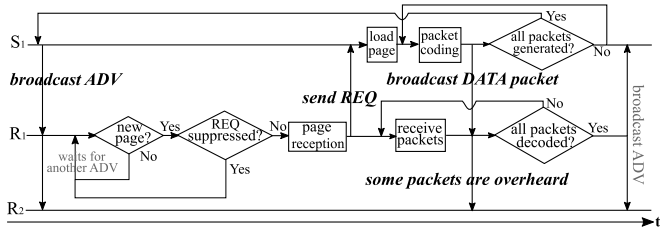


Fig. 1. Overview of coding-based reprogramming, main steps are marked bold italic.

In summary, the paper makes the following contributions:

- To the best of our knowledge, this appears to be the first comprehensive analytical model for coding-based reprogramming protocols in lossy wireless sensor networks. It is flexible and extensible in the way that it can be used as a theoretical model to analyze and quantify the performance of any existing coding-based reprogramming protocol. The results from our extensive testbeds are in good agreement with the performance predicted by our model.
- We characterize overhearing in our model to reflect the real behaviors of coding-based reprogramming protocols in dense networks, which is more realistic and accurate. The observation from our experiments shows that overhearing in dense networks reduces 50-70 percent of the total completion time.
- We also incorporate and analyze packet coding in our model which is typically ignored in existing protocols. We optimize the coding process by selecting appropriate packet coding parameters, and analyze its consequence by comparing it to the communication delay. The analytical results show that the communication delay is about 65 percent of the coding delay when the number of packets per page is 16 for the finite field size 2^8, and the minimal total completion time can be achieved when the number of packets per page is close to 24 and the finite field size is close to 2^4.

The remainder of this paper is organized as follows. Section 2 describes the background in code dissemination. Section 3 gives the problem statement, notations and an overview of our model. Section 4 presents the details of our analytical model. Section 5 reports the results from our evaluation. Section 6 discusses the related work. Section 7 concludes the paper.

2 Background

In code dissemination, a code image (a.k.a. firmware) is divided into equal-sized pages, these pages are disseminated page by page from the base station to all other sensor nodes. The three-way handshaking process (ADV-REQ-DATA) is used for transmitting each page, as shown in Fig. 1 (main steps are marked).

In Fig. 1, when possessing a code image, the sender S_1 first broadcasts an advertisement message (ADV) that includes the current version number of the code image, the

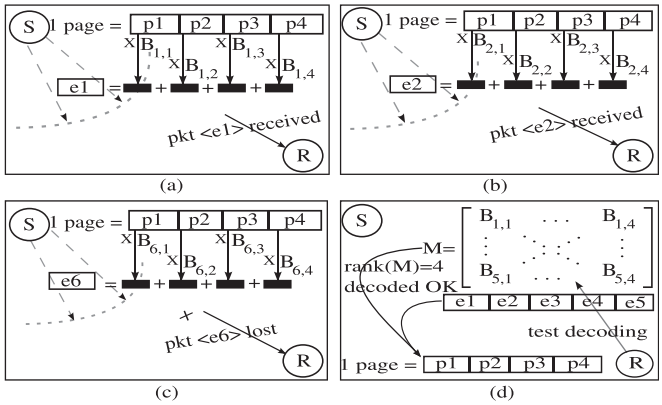


Fig. 2. Packet coding. (a) S encodes the first packet e_1 from a page and broadcasts it, R receives it; (b) S encodes the second packet e_2 and broadcasts it, R receives it; (c) S encodes the last packet e_6 and broadcasts it, R loses it; (d) R receives five packets and tests if decoding of the original four packets is OK.

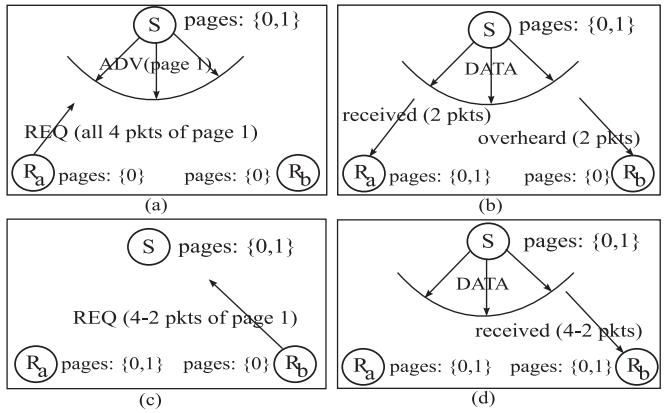


Fig. 3. Overhearing in packet transmission. (a) S broadcasts ADV for page 1, R_a sends REQ for all the four packets; (b) R_a receives two of all packets in page 1, R_b overhears two packets; (c) R_b sends REQ for the rest two packets from S; (d) R_b receives the rest two packets.

number of pages contained in the image, and the number of new pages. Its neighbor nodes R_1 and R_2 receive the message and compare the advertised page number with their existing page number, and R_1 sends a request packet (REQ) to S_1 and starts page reception if a new page is available, otherwise R_1 has to wait for another ADV. Any ADV containing the old code image from R_1 and R_2 will be suppressed. If R_1 has not exceeded the limit of λ request packets, it sends the REQ, or it must wait for another ADV before making additional requests. In the page reception phase, R_1 prepares to receive all the packets in this page, in traditional reprogramming protocols, S_1 starts broadcasting data packets (DATA) to R_1 upon request. However, in coding-based reprogramming protocols, S_1 loads the requesting page from the flash at first upon request and generates each encoded data packet from the page (this is known as packet coding), then S_1 starts broadcasting each encoded data packet (DATA) to R_1 until all the packets are generated. R_2 may overhear some encoded data packets although it has not sent any request before. S_1 may broadcast another ADV for the next page if all the encoded packets of the current page are generated. R_1 tries to decode the received encoded packets. If the original packets cannot be successfully decoded, it has to receive more encoded packets, otherwise it will broadcast an ADV containing information for this decoded page. The packet coding and overhearing are important to coding-based reprogramming, we therefore describe them in details in the following sections.

2.1 Packet Coding

Packet coding, which consists of encoding and decoding, is used during the packet transmission phase to overcome packet loss. Fig. 2 demonstrates the coding details when six encoded packets are generated from all the four original packets in one page.

Encoded packets are generated from all the original packets in a page before transmission. The number of encoded packets is usually greater than the number of original packets for the purpose of resisting packet loss in highly lossy wireless environments. In Rateless Deluge, each encoded packet is linearly computed from all the original packets in a page and the coefficients are generated randomly from the finite field $GF(2^8)$. Each encoded packet is then transmitted to receivers.

When a sufficient amount of encoded packets are received at any receiver, Gaussian elimination is applied to resolve the matrix of received coefficients, if the rank of the matrix is greater than or equal to the number of original packets, the original packets can be decoded.

2.2 Overhearing

After the sender receives any request message in a page from its downstream nodes, it broadcasts all the encoded packets in that page. All its neighbors can then receive and store these packets although it is not the node sent the request before.

As illustrated in Fig. 3, when one of the overhearing nodes needs to send a request for page 1 afterwards, it will send the request for the rest two packets of page 1 as two packets have been overheard already. If one neighbor has overheard all the four packets in page 1, its request message for page 1 will be suppressed and the total completion time can be reduced accordingly.

3 Overview

In this section, we give an overview of our proposed analytical model. We formulate the problem of determining the completion time of code dissemination as follows. Among N sensor nodes $\{S_1, S_2, \ldots, S_N\}$ in a network, S_1 is designated as the base station that disseminates a code image to all other sensor nodes. A code image C is divided into P equal-sized pages $\{C_1, C_2, \ldots, C_P\}$. Let us denote the completion time for sending page C_p from node S_i to node S_j by $T(p, i, j)$, where $p \in [1, P]$ and $i, j \in [1, N]$. Then, the total completion time of code dissemination is $\max_j T(P, 1, j)$. When the number of overheard pages $P_{oh} > 0$, the total completion time becomes $\max_j T(P - P_{oh}, 1, j)$ and it will be reduced. Table 1 defines all the symbols used.

In general, $T(p, i, j)$ depends on:

1) the propagation path l from S_i to S_j;
2) the duration of single-hop operations for all the sensor nodes along propagation path l.

Therefore, the completion time of C_p on each sensor node along the propagation path has to be determined.

TABLE 1
List of Frequently Used Symbols

Symbol	Definition
N	total number of nodes
P	number of pages divided from code image C
ψ	page size (in bytes)
ϕ	number of packets per page
ϵ	encoding coefficient
q	exponent of finite field size in $GF(2^q)$
ϕ_{oh}	the number of overheard packets per page
P_{oh}	the number of overheard pages
L	number of nodes along propagation path l
T_{model}	the completion time calculated by model
$T_{testbed}$	the completion time calculated by testbed

Furthermore, the time cost of all protocol behaviors such as packet coding and page pipelining has to be determined.

Our analytical model consists of two main components: *multi-hop propagations* and *single-hop operations*. The "multi-hop propagations" component captures the dynamics of a code image being disseminated across a network, and takes an input from the "single-hop operations" component. The "single-hop operations" component captures the complexity of meta-data negotiation and packet coding, and takes input from the physical layer model, which encapsulates physical parameters such as packet reception rate, receiver signal strength indicator (RSSI), or link quality indicator (LQI). These physical parameters can be obtained from experiments or simulations, as shown in Section 5.2.

4 Analytical Approach

In this section, we present the *multi-hop propagations* component, followed by the *single-hop operations* component.

4.1 Modeling Multi-Hop Propagations

The completion time $T(p,1,j)$ of any page C_p from S_1 to S_j depends on the propagation path l from S_1 to S_j. We apply Dijkstra's shortest path algorithm to determine the propagation path as noted in [12], [13]. Compared to the conference version [11], our model is extensible to capture different sender selections in protocols similar to MNP [14] and ECD [15] when generating the propagation path, as such path is composed of sender nodes selected across the network. This extension depends on definition of $weight$ in the shortest path algorithm. For example, we could define $weight$ as maximum number of requests from receivers in MNP [14], while in ECD [15], we define $weight$ as the largest sum of outbound link qualities. In implementation, maximum number of requests could be replaced with largest link qualities of feedback channels because link quality of feedback channel influences the number of requests successfully sent from each receiver. The default $weight$ is set as link quality between any two nodes to capture sender selections for protocols similar to Deluge, including Rateless Deluge, SYNAPSE++, ReXOR, AdapCode and Splash.

Suppose the path l from S_1 to S_j consists of L nodes, and we label the nodes as $S_{l(1)}, S_{l(2)}, \ldots, S_{l(k)}, \ldots, S_{l(L)}, l(k) \in [1, N], k \in [1, L]$. To determine $T(p,1,l(k))$, we first observe that

$$T(p,1,l(k)) \geq T(p,1,l(k-1)) + T(p,l(k-1),l(k)), \quad (1)$$

where $T(p,l(k-1),l(k))$ represents the duration of single-hop operations, including sending C_p from $S_{l(k-1)}$ to $S_{l(k)}$, $p \geq 1, k \geq 2$. The equality in (1) is due to the fact that page C_p is propagated from node $S_{l(k-1)}$ to node $S_{l(k)}$, whereas the inequality in (1) is due to the fact that not all the packets in page C_p arrive through the shortest path. Next, we consider page pipelining (a.k.a. spatial multiplexing [4]), which is the concurrent transmission of different pages by different nodes. The nodes must be at least *three hops* apart to avoid collisions. In an ideal scenario of page pipelining, when node $S_{l(k+3)}$ finishes receiving page C_{p-1}, node $S_{l(k)}$ also finishes receiving page C_p. In other words, for $p \geq 2$, we have

$$T(p,1,l(k)) \geq T(p-1,1,l(\min(k+3,L))). \quad (2)$$

To approximate $T(p,1,l(k))$ by its lower bound, we combine (1) and (2), i.e., for $k \geq 2$, we have

$$T(p,1,l(k)) \geq \max\{T(p,1,l(k-1)) + T(p,l(k-1),l(k)), T(p-1,1,l(\min(k+3,L)))\}. \quad (3)$$

The maximum completion time $\max_j T(P,1,j)$ from all the shortest paths is calculated recursively according to (3), with the initial condition $T(p,1,l(1)) = 0$. Fig. 4 shows the algorithm to calculate $T(p,1,j)$ according to (1), (2), and (3). In Fig. 4, the matrix $weight[N][N]$ contains values representing link quality of physical channel calculated between any two nodes for N nodes, the link gain of the link layer model in TOSSIM is used in simulations and PRR is used in testbeds; the array $weight[N][N]$ appears as the input of Dijkstra's shortest path algorithm, the array $SP[j]$ represents the propagation path l, and it is generated by this algorithm, $SP[j]$ contains nodes from the base station to node S_j; $|SP[j]|$ represents L, is the number of nodes in path l.

4.2 Modeling Single-Hop Operations

We split the computation of expected duration for single-hop operations into two phases: communication and coding, i.e., $E(T(p,l(k-1),l(k))) = E(T_{comm}) + E(T_{coding})$. The communication phase is carried out for meta-data negotiation. The meta-data negotiation mechanism proposed by SPIN [16] and used in Deluge and its derivatives (e.g., Rateless Deluge), enables the epidemic-like propagation of a code image from the base station to all other sensor nodes. Therefore, from a receiver's perspective, $E(T_{comm})$ includes the delay for the reception of advertisement packets, the transmission of request packets, and the reception of data packets. In another word,

$$E(T_{comm}) = E(T_{adv}) + E(T_{req}) + E(T_{data}). \quad (4)$$

The expected time of advertisement reception can be estimated as

$$E(T_{adv}) = E(N_{pkt}) \cdot E(r_i) \cdot (1 + E(N_{supp})), \quad (5)$$

where $E(N_{pkt}) = \frac{1}{PRR}$ is the expected number of transmissions for one packet, at a packet reception rate of PRR. The measurements of PRR and its correlation properties will be discussed in Section 5.2. $E(r_i)$ specifies the expected wait time between advertisement messages. $E(N_{supp})$ specifies

Input: $weight[N][N]$, PLAT, PROTO
Output: $T(p,1,j)$
$SP[N][N] \leftarrow$ FINDPATH(weight, PLAT, PROTO)
for $j \leftarrow 1, N$ **do**
  for $p \leftarrow 1, P$ **do**
    $l \leftarrow SP[j]$, $L \leftarrow |SP[j]|$
    if $p = 1$ **then**
      for $k \leftarrow 1, L$ **do**
        $x \leftarrow l(k-1)$
        $T(1,1,l(k)) \leftarrow T(1,1,x) + T(1,x,l(k))$
      end for
    else
      for $k \leftarrow 1, L$ **do**
        $x \leftarrow l(k-1)$, $z \leftarrow l(\max(k+3, L-1))$
        $T(p,1,l(k)) \leftarrow T(p,1,x) + T(p,x,l(k))$
        if $T(p,1,l(k)) \leq T(p-1,1,z)$ **then**
          $T(p,1,l(k)) \leftarrow T(p-1,1,z)$
        end if
      end for
    end if
  end for
end for
procedure FINDPATH(weight, PLAT, PROTO)
  if PLAT = TOSSIM **then** ▷ Simulation
    $weight[i][j] \leftarrow$ |link gain between S_i and S_j|
  else ▷ Testbed
    $weight[i][j] \leftarrow$ 1-(PRR between S_i and S_j)
  end if
  ▷ l.q. is abbreviation of link qualities
  if PROTO = MNP **then**
    $weight[i][j] \leftarrow$ 1/(largest l.q. of fdbk channels)
  else if PROTO = ECD **then**
    $weight[i][j] \leftarrow$ 1/(largest sum of outbound l.q.)
  end if
  run Dijkstra's shortest path algorithm on $weight$
  record shortest path for N nodes in SP
  return SP
end procedure

Fig. 4. Algorithm to calculate $T(p,1,j)$.

the number of times that an advertisement is suppressed by the previous node on the propagation path before transmission. We set $E(N_{supp}) = 1$ as in Deluge.

The expected time of a request transmission can be estimated as

$$E(T_{req}) = E(N_{pkt}) \cdot E(N_{reps}) \cdot \frac{\tau}{2} + E(T_{back}), \quad (6)$$

where $E(N_{pkt})$ is the same as in (5). $E(N_{reps})$ specifies the expected number of request packets needed to receive one page. $\frac{\tau}{2}$ specifies the expected duration of random back-off before a request transmission. $E(T_{back})$ specifies the expected time for additional advertisements and is defined as

$$E(T_{back}) = E(N_{pkt}) \cdot \left(\frac{E(N_{reps})}{\lambda} - 1\right) \cdot E(T_{adv}), \quad (7)$$

where λ is the number of requests, more than which the node has to wait for another advertisement before making additional requests.

The expected time of data reception is estimated as

$$E(T_{data}) = E(N_{pkt}) \cdot (\phi - E(\phi_{oh})) \cdot E(T_{pkt}) \cdot E(N_{mac}), \quad (8)$$

where $E(N_{pkt})$ is the same as in (5). $E(\phi_{oh})$ represents the expected number of overheard packets per page. When ϕ_{oh} equals the total number of packets per page ϕ, the current page transmission is totally reduced and the number of overheard pages P_{oh} in multi-hop propagations increases by 1. $E(\phi_{oh})$ and $E(P_{oh})$ are examined in Section 5.2. $E(T_{pkt})$ specifies the transmission time for one packet. $E(N_{mac})$ captures the expected time of the MAC-layer delay due to contention, and we set $E(N_{mac}) = 1$.

For the coding phase, we consider only the most time-consuming operations: encoding and decoding, i.e., $E(T_{coding}) = E(T_{enc}) + E(T_{dec})$. As different coding-based reprogramming protocols use different packet coding schemes, we quantify $E(T_{enc})$ and $E(T_{dec})$ in the following sections.

4.3 Estimating Coding Delay

Instead of calculating the coding delay and set a constant value in the existing model, we incorporate flexible packet coding parameters in the analytical model. In Rateless Deluge, random linear code is used for encoding and decoding data packets. To send a page consisting of ϕ packets $X_1, \ldots, X_\phi$, the packets are encoded as $\epsilon\phi$ packets $Y_1, \ldots, Y_{\epsilon\phi}, \epsilon = 1.5$ according to (9):

$$\begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,\phi} \\ \beta_{2,1} & \cdots & \beta_{2,\phi} \\ \vdots & \ddots & \vdots \\ \beta_{\epsilon\phi,1} & \cdots & \beta_{\epsilon\phi,\phi} \end{bmatrix} \begin{bmatrix} X_{1,j} \\ X_{2,j} \\ \vdots \\ X_{\phi,j} \end{bmatrix} = \begin{bmatrix} Y_{1,j} \\ Y_{2,j} \\ \vdots \\ Y_{\epsilon\phi,j} \end{bmatrix}, \quad (9)$$

where $X_{i,j}$ denotes the jth byte of X_i, and β's are pseudo random numbers. By populating the matrix on the left hand side with pseudo random numbers, the matrix is rendered almost surely non-singular. Decoding $Y_1, \ldots, Y_{\epsilon\phi}$ to $X_1, \ldots, X_\phi$ is by means of Gaussian elimination.

A page can be pre-coded in anticipation of future requests to reduce the dissemination delay. The expected time of encoding, taking pre-coding into account, is thus

$$E(T_{enc}) = \epsilon\phi \cdot E(\rho) \cdot E(T_{epkt}), \quad (10)$$

where $E(\rho)$ specifies the ratio which the time overhead can be reduced due to pre-coding, which is constrained by RAM size, page size $\psi = 21\phi$ (i.e., the payload of one packet in Rateless Deluge has 21 bytes), and delay before pre-coding. $E(\rho)$ is derived from an experiment in Rateless Deluge. $\epsilon\phi$ specifies the expected number of packets to be encoded in Rateless Deluge. A packet is encoded byte by byte, and each byte needs $(5q\,AND + 2q\,XOR + 4q\,SHIFT + 4q\,CMP)$ operations estimated. The expected time to encode a packet is thus

$$E(T_{epkt}) = \frac{q\psi}{\lfloor \frac{q+7}{8} \rfloor} \cdot (5 \cdot \tau_A + 2 \cdot \tau_X + 4 \cdot \tau_{SH} + 4 \cdot \tau_C), \quad (11)$$

where the finite field size is by default $2^q = 2^8$ in Rateless Deluge. τ_A, τ_X, τ_{SH}, and τ_C are the time for byte-wise AND, XOR, SHIFT, and CMP operations, respectively.

The decoding procedure in Rateless Deluge consists of Gaussian elimination. The expected time of decoding is

$$E(T_{dec}) = E(N_{dec}) \cdot E(T_{ge}), \tag{12}$$

where $E(N_{dec}) = \frac{1}{1-Pr(dec.fail)}$ specifies the expected number of times of decoding, under a decoding failing probability of $Pr(dec.fail)$, depending on number of packets per page ϕ and the finite field size 2^q. $E(T_{ge})$ specifies the expected duration of Gaussian elimination, and it is estimated as

$$E(T_{ge}) = \frac{\phi^2 + 3\phi - 1}{3} \cdot T_{epkt} + \frac{4\psi(2\phi^2 + 3\phi - 5)}{3} \cdot \tau_S, \tag{13}$$

where T_{epkt} is as defined in (11), and τ_S is the time for byte-wise SUB operation.

4.4 Extension to SYNAPSE++

By substituting the coding phase and protocol-specific operations, we can extend our analytical model to other coding-based reprogramming protocols.

In SYNAPSE++, the code image is subdivided into P pages (or packets) with ψ bytes each. The encoding process depends on the *degree distribution* $\rho(d)$ and a degree d_n is set for the encoding process. Each encoded packet is generated by randomly and uniformly picking d_n packets from the original P packets and is the bitwise, modulo 2 sum of these d_n packets. The encoding time can be estimated as

$$E(T_{enc}) = \psi \cdot E(N_{enc}) \cdot (E(d) - 1) \cdot \tau_X, \tag{14}$$

where $E(N_{enc})$ specifies the expected number of encoded packets generated and is constrained by the overhead set for the *degree distribution*. $E(d)$ is the expected number of packets to pick for all encoded packets. τ_X here is the time for byte-wise XOR operation and can be replaced by a 16-bit word-wise operation for any target platform used microcontroller TI MSP430.

In the decoding procedure, the decoding time can be estimated similar to 12.

4.5 Extension to ReXOR

Unlike Rateless Deluge, SYNAPSE++ and other coding-based reprogramming protocols, ReXOR encodes only lost packets in the retransmission phase. As in Deluge, ReXOR transmits original packets in the first phase. If some packets are lost at the receivers, the request messages will be sent back and the sender will broadcast extra encoded packets in the following retransmission phases. In each retransmission phase, the sender needs to wait for *request vectors* from the receivers at first and then execute the sequential coloring algorithm to determine which original packets should be XOR-ed together to generate each encoded packet. When all XOR encodings finish, the encoded packets will be broadcast. Extra retransmission phases will be required if some packets are still lost after previous retransmission phase(s). Thus, the encoding time of ReXOR is estimated as

$$E(T_{enc}) = \mu \cdot (E(T_{wait}) + E(T_{clr}) + (E(\phi_l) - 1) \cdot \tau_X), \tag{15}$$

where μ is the expected number of retransmission phases. $E(T_{wait})$ denotes the expected interpage waiting time for collecting *request vectors* and its relationship with average neighbors *avgnb* determined in ReXOR. $E(T_{clr})$ specifies the expected time of executing the coloring algorithm and its calculation depends on the number of *request vectors* and PRR. The results of T_{clr} under some workloads are shown in ReXOR. $E(\phi_l)$ is the expected number of lost packets within one page and can be defined as $\phi \cdot (1 - PRR)$. τ_X is defined in (11).

When any receiver receives the encoded packets, it retrieves the lost packets by XOR-ing each encoded packet with those original packets which are used in the sender to generate this encoded packet. Then the decoding time can be estimated as

$$E(T_{dec}) = (\phi - E(\phi_l)) \cdot \tau_X. \tag{16}$$

4.6 Extension to Splash

In Splash, the whole dissemination is composed of tree pipelining and local recovery phase. For the expected time of tree pipelining $E(T_{tp})$, as no coding and message negotiation is needed, we model time of single-hop operations as $E(T(p, l(k-1), l(k))) = E(T_{comm})$ and have

$$E(T_{comm}) = E(T_{sync}) + E(T_{chCyc}) + E(T_{data}), \tag{17}$$

where T_{sync} and T_{chCyc} represent the synchronization overhead and channel diversity overhead, respectively. $E(T_{tp})$ can be calculated as in Fig. 4.

To calculate the expected time of local recovery phase, we notice that CSMA is used in local recovery phase. We have to find out the packet loss distribution from the empirical traces to calculate packet reception rate. With the packet reception rate, the expected recovery time for each node can be estimated and the expected time of local recovery phase can then be modeled as the maximum expected recovery time. The packet loss distribution can be easily derived from the model in [17]. Therefore, the expected recovery time for node i $(1 \le i \le N)$ can be estimated by packet transmissions as

$$E(T_{lr}^i) = \frac{\phi_{lost}^i}{PRR_{lr}^i} \cdot (E(T_{pkt}) + E(T_{tv})) - E(T_{tv}), \tag{18}$$

where ϕ_{lost}^i is the number of lost packets after tree pipelining. PRR_{lr}^i is the packet reception rate calculated from packet loss distribution. T_{pkt} and T_{tv} represent time of packet transmission and time interval between packet transmission, respectively. As a result, the expected time of local recovery phase is $E(T_{lr}) = \max_{1 \le i \le N} E(T_{lr}^i)$.

4.7 Extension to Adaptive-Coding-Based Protocols

AdapCode [18], which exploits adaptive codes, is also supported by our model. Each sent packet is encoded from ϕ_{adap} original packets, where $\phi_{adap} = f(N_{nb})$ is a mapping function of the number of neighbors, N_{nb}, of the sender (the mapping is shown in Table 3 of AdapCode). To transmit one page, $\frac{\phi}{\phi_{adap}}$ encoded packets are generated. Therefore, we have the encoding delay $E(T_{enc}) = \frac{\phi}{\phi_{adap}} \cdot E(T_{epkt})$, where T_{epkt} is time to generate one encoded packet. For decoding,

TABLE 2
All Setups of Two Testbeds Used in Experiments

Setup	RSSI (dBm)	Platform	Topology	Density
(M1)	$-82\sim-87$	MICAz	4×4	Dense
(M2)	$-84\sim-90$	MICAz	4×4	Sparse
(T1)	$-79\sim-91$	TelosB	4×4	Sparse
Tea	$-65\sim-67, -84\sim-93$	TelosB	9×2	Sparse

Gaussian elimination is used, so we have the expected decoding delay $E(T_{dec}) = E(T_{ge})$, where T_{ge} is the time of Gaussian elimination, and is a function of ϕ_{adap}.

5 Evaluations

We now move to evaluate our analytical model by computing the completion time of coding-based reprogramming process. We use existing coding-based reprogramming protocol such as Rateless Deluge (an extension of Deluge T2) as a benchmark. Testbed experiments are conducted for comparison in different network topologies under different node spacing and image size. We introduce the prediction error $err_{testbed,\ model}$ to describe the consistency between analytical results of our model and experimental results, it is calculated as

$$err_{testbed,\ model} = \frac{|T_{testbed} - T_{model}|}{T_{testbed}}, \tag{19}$$

where $T_{testbed}$ and T_{model} denote the completion time calculated by testbed experiments and the analytical model, respectively. Besides, we examine the computation complexity and scalability of our model.

Furthermore, we will evaluate the analysis of packet coding (consisting of encoding and decoding) through our model. Packet coding parameters can be analyzed to compare coding delay and communication delay, and should be carefully chosen to reduce the total completion time. Overhearing, and its impact on the completion time in dense networks will be evaluated in our testbed experiments.

5.1 Experiment Setup

Before presenting our experimental results, we describe testbeds we built to setup parameters and evaluate our model. We deploy two testbeds (i.e., 4×4 grid and 9×2 linear) using MICAz and TelosB. Table 2 lists all the four setups used in our experiments. (M1), (M2), and (T1) are three setups of the grid testbed, *Tea* is the setup of the linear testbed. (M1) is used for dense networks, (M2), (T1), and *Tea* are for sparse networks. (M1) and (M2) are composed of MICAz nodes, (T1) and *Tea* are composed of TelosB nodes. Fig. 5 presents *Tea* which is mounted on the ceiling of the main hallway in our laboratory at our school and one $4\times$ grid setup (T1). We deploy *Tea* along the hallway with two lines where nodes on each line are 8.0 m apart and nodes at the same position of each line are 1.5 m apart, similar to the setup in Indriya [19]. To collect sensor data, every four nodes in the grid setups are connected to a 4-port USB hub and all the USB hubs are plugged into a laptop computer for further data analysis. In *Tea*, the top-level hub is deployed in the center of the line and connected to the laptop, all the hubs and

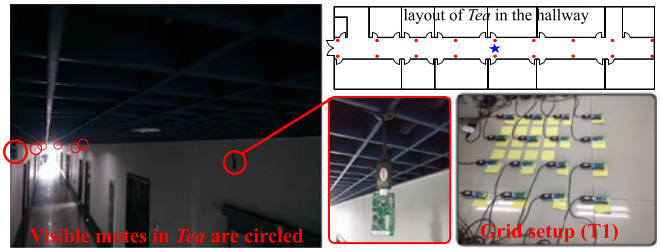


Fig. 5. The linear testbed *Tea* and grid setup (T1).

nodes form a 5-tier topology. In all the experiments, the base station node is located at one corner. *Tea* shares the same ceiling with many 802.11 access points, and the same hallway with heavy student/mobile phone traffic.

As the RF output power of CC2420 (radio chip used in both MICAz and TelosB) is controlled by a register $TXCTRL.PA\_LEVEL$, we conduct experiments to select an appropriate output power (in dBm) to set up the sparse and dense networks. We set $PA\_LEVEL = 2$ for (M1), (M2) and (T1) and set $PA\_LEVEL = 3$ for *Tea*. Channel 13 is selected for all the testbeds. RSSI is measured by running *RssiDemo* (appeared in TinyOS 2.x tutorials directory). For (M1), (M2) and (T1), *RssiDemo* runs on the nearest pairwise nodes. For *Tea*, *RssiDemo* runs on pairwise nodes with an inter-node spacing of $1.5\,\mathrm{m}$ and pairwise nodes with $8.0\,\mathrm{m}$. Each RSSI value in Table 2 is the sum of *RssiDemo* reading and RSSI offset (-45 dBm for CC2420).

The usage of our analytical model necessitates the empirical determination of several key parameters. The purpose of testbed experiments is hence twofold: 1) determine or "set up" these parameters; 2) evaluate the analytical model. In Section 5.2, we describe model parameters, specifically PRR and its relation to number of requests messages. In the remaining of this section, we report the results from a series of experiments.

5.2 Parameter Settings

Experiments are conducted to setup parameter settings for our model, same settings could be concluded in simulations. We set the expected random back-off time $E(\tau) = 0.144\,\mathrm{s}$ and set $\lambda = 2$ such that a node makes a maximum of two requests after an advertisement (see (7)). Through experimental results, we are able to determine the following parameter values.

- The expected wait time between advertisement messages, $E(r_i) = 0.8\,\mathrm{s}$;
- The expected transmission time for one packet, $E(T_{pkt}) = 0.00862\,\mathrm{s}$.

For MICAz and TelosB nodes used in the testbed experiments, we need to determine additional T_{pkt} as follows.

$$T_{pkt} = \frac{50\,\mathrm{bytes/pkt}}{250\,\mathrm{kbps}} + 5.12\,\mathrm{ms} = 0.00672\,\mathrm{s} \tag{20}$$

where each active message has 50 bytes (in *CC2420.h*) and $5.12\,\mathrm{ms}$ is the maximum initial backoff time from CSMA implementation of the CC2420 radio driver in TinyOS 2.x (in *CC2420CsmaP.nc*).

The values of $E(N_{reps})$, $E(\phi_{oh})$, and $E(P_{oh})$ depend on PRR. We determine PRR in both testbeds and simulations

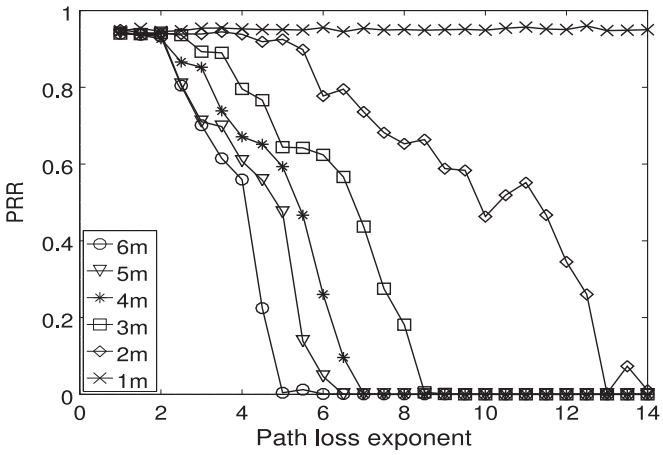


Fig. 6. PRR versus path loss exponent.

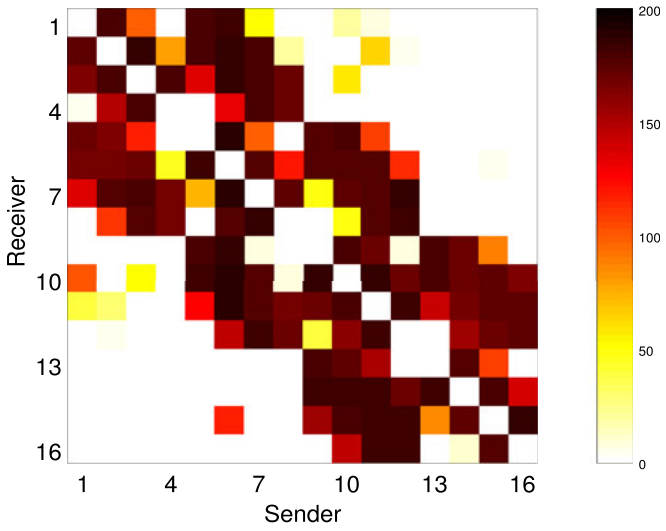


Fig. 7. Pairwise packet receptions in (M1).

because our model can be extended and evaluated in both situations.

For simulations, we use the discrete-event simulator TOSSIM to measure PRR. In TOSSIM, the log-normal shadowing channel model [17] is used to model radio connectivity between nodes. Given a reference distance D_0 (in meters), and the corresponding path loss PL_{D_0} (in dB), the link quality between nodes can be calculated, given the path loss exponent, shadowing deviation, and noise floor (in dBm). Sensor node MICAz is simulated in TOSSIM. Simulations are conducted by running the *CountRadio* application from TinyOS 2.x on two nodes that are separated from each other with variable node spacing and path loss exponent. We program each node to send 1,000 packets to and receive from the other node, and count the number of received packets. The PRR is estimated as follows:

$$PRR = \frac{\sum_{i=1}^{\Delta} \Omega_i}{\Delta\sigma}, \tag{21}$$

where Δ specifies the number of experiments conducted, $\sigma = 1{,}000$ describes the number of packets sent, and Ω_i specifies the number of packets received successfully in the ith experiment.

Fig. 6 plots the results of PRR determined in TOSSIM with variable path loss exponent and node spacing, PRR drops when the path loss exponent increases. When the node spacing approaches reference distance D_0 (= 1.0 m), the results are unusable due to the limitation of the radio model. Table 3 presents PRR measurements in simulations when a path loss exponent is 4.0.

In testbeds, PRR cannot be calculated like in TOSSIM as any local PRR value does not represent PRR for the entire network due to dynamic behaviors in real-world environments. Three steps are needed to determine PRR in the testbeds. First, we need to identify one-physical-hop (*1ph*) neighbors for each node. Second, we calculate accumulated average pairwise PRR between each node and its every neighbor as accumulated *1ph* PRR for each node. Third, PRR is the average value of accumulated *1ph* PRR of all nodes.

Fig. 7 plots the results of pairwise packet receptions measured from (M1) when 200 packets are sent from each node. In (M1), every node has good links with its *1ph* neighbors, the code image will spread fast in the network and the overall PRR is high. In *Tea*, many bad links exist for its neighbors, dissemination in the whole network is slow due to unbalanced links and the overall PRR is low. Table 4 shows the results of PRR and its median values (for reference) calculated from testbeds. This method avoids inaccuracy to some extent when link hole exists in the networks where most of the links have high pairwise PRR when using the weighted average PRR reported in [7].

Average number of request packets $E(N_{reps})$ can be determined after PRR. Fig. 8 shows the relation between $E(N_{reps})$ and PRR. In Fig. 8, $E(N_{reps})$ increases as PRR decreases, when PRR is high (over 80 percent), most of the nodes send no requests as data packets are overheard in advance. $E(N_{reps})$ drops slowly when PRR is small (less than 80 percent), but drops sharply when PRR is over 80 percent.

The number of overheard packets per page ϕ_{oh} depends on estimated PRR in any neighborhood area. Fig. 9 plots the relation between ϕ_{oh} and PRR. The increment of number of overheard packets when PRR is high is slower than that when PRR is low. This is true because good link quality contributes to such increment while further increment is constrained by density for high link quality. At last, the expected number of overheard pages $E(P_{oh})$ is calculated as $E(P_{oh}) = \frac{\phi_{oh}}{\phi} P$.

TABLE 3
PRR Measurements in Simulations

spacing (m)	PRR	spacing (m)	PRR
1.5	0.9405	2.0	0.9380
2.5	0.9050	3.0	0.7959
3.5	0.7796	4.0	0.6711
4.5	0.6402	5.0	0.6075
5.5	0.5858	6.0	0.5597

TABLE 4
PRR Measurements in Testbeds

Setup	PRR	PRR (med)	Setup	PRR	PRR (med)
(M1)	0.8960	0.8995	(T1)	0.5593	0.6043
(M2)	0.5246	0.5510	*Tea*	0.6495	0.6433

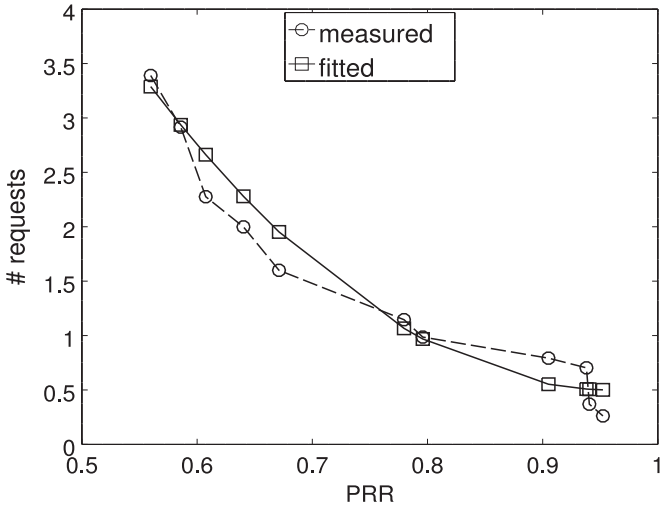


Fig. 8. Number of requests versus PRR.

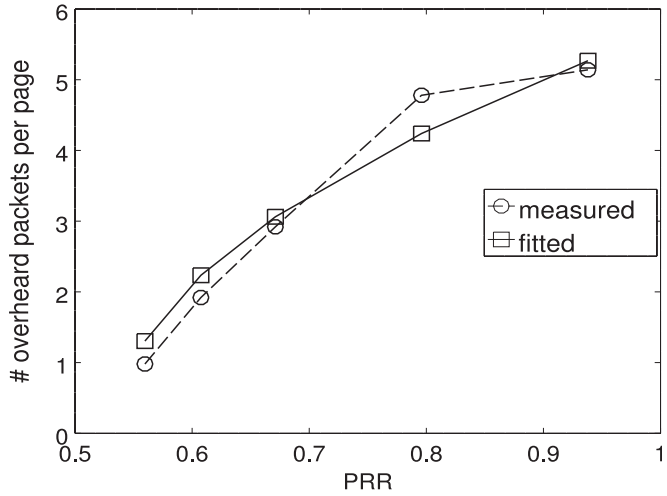


Fig. 9. Number of overheard packets per page versus PRR.

5.3 Results on Node Spacing and Image Size

An analytical model should be general and accurate in many scenarios. First, we test our model in both linear and grid networks. For example, (M1) and (T1) are grid networks while *Tea* is a linear network. Second, different node spacing is set. Third, we prepare three code images *Blink*, *Oscilloscope*, and *GoldenImage*. These code images are generated from TinyOS 2.x, each has 16, 82, and 193 pages, respectively. The packet coding parameters in our model are set as in Rateless Deluge. We conduct testbed experiments in above aspects to demonstrate that the analytical model is accurate to predict the completion time in real-world deployments. The comparison results are shown in Table 5.

As shown in Table 5, the prediction errors in these testbed setups are 3.116, 5.161, 5.106, 0.433, and 5.402 percent, respectively. The results show that the completion time increases more slowly compared with the increase of image size in the same setup (M1). The prediction errors $err_{testbed,\ model}$ between experimental results and analytical results are small and below 6 percent in all testbeds, thus validating accuracy of our model. As (M1), (T1), and *Tea* have different node spacing, the completion time calculated by analytical model matches that by each testbed experiment. Although *Tea* is deployed in the hallway and may interfere with 802.11 and when people walk around, the analytical results match experimental results in the completion time with low prediction error. Furthermore, we also compare our model with a secure version of Rateless Deluge called Sreluge [20] in the absence of attacks, and the results are similar to Rateless Deluge.

5.4 Results on Scalability

Before presenting evaluation results, we present the computational complexity of the analytical model. The computational complexity has two parts, as shown in Fig. 4. One part is the calculation of shortest paths, $O(|E| + N \log N)$. The other part is the calculation of dissemination delay in these paths, $O(PNL)$, which can be reduced to $O(PN)$ by optimization in Fig. 4. As a result, the computational complexity of the analytical model is

$$O(PN) + O(|E| + N \log N), \tag{22}$$

where $|E|$ represents number of connected links in the network. P, N are defined in Table 1.

We implement our model in C on a HP EliteBook 8440p laptop to examine the computational complexity for different scales. The laptop has a quad-core 2.27 GHz CPU with 8 GB RAM and runs Linux. We first use TOSSIM to generate different grid network topologies, then we compile and run our model on them to disseminate *Blink*, the analytical results are shown in Table 6. Table 6 shows that, for a 30×30 grid network, the analytical model finishes in less than 2 seconds. The runtime in a 30×30 grid network is about 14 times of that in a 5×5 grid network.

We also record the runtime of our analytical model in testbed setups (M1) and (M2) with different binary image size, all time recordings are less than 0.1 s and can be ignored compared with the results presented in Table 5 which is actually the average of several runs. Furthermore, we use TOSSIM on above topologies to examine whether the analytical model can scale well. As TOSSIM cannot capture computation time on real motes since it does not simulate the execution of mote instructions, it is not a fair

TABLE 5
Completion Time versus Image Size: Testbed Results

Setup	Application	image size (bytes)	number of pages	$T_{testbed}$ (s)	T_{model} (s)	$err_{testbed,\ model}$ (%)
(M1)	*Blink*	2,102	16	41.067	39.826	3.116
(M1)	*Oscilloscope*	12,340	82	209.870	199.570	5.161
(M1)	*GoldenImage*	31,346	193	492.150	468.240	5.106
(T1)	*Blink*	2,102	22	1,182.000	1,176.900	0.433
Tea	*Blink*	2,102	22	172.710	182.041	5.402

TABLE 6
Analytical Results of Runtime to Calculate the Completion Time

Topology	Our model (s)	Topology	Our model (s)
5×5	0.122	20×20	0.452
10×10	0.158	30×30	1.647

comparison for coding-based protocols. However, testbeds suffer from scalability or availability issues. Our model has been validated in testbed experiments and is able to calculate computation time separately, we could use this part of computation time from our model to compensate simulations. The comparison results are shown in Fig. 10, prediction errors (similar to Equ. (19)) between our model and simulations are presented.

In Fig. 10, we conclude that the prediction error is below 8.0 percent when the network size increases from 25 to 100 and increases slower than the network size, thus proving that our model scales well in these topologies.

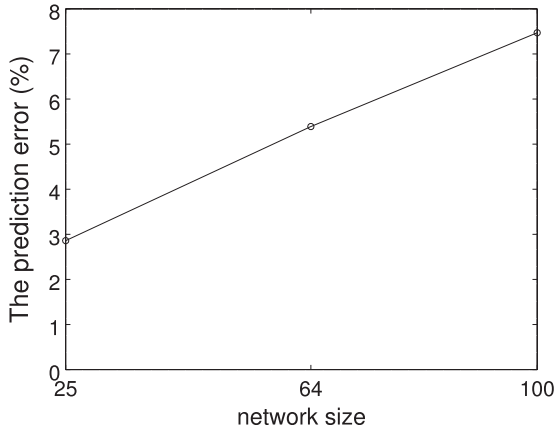


Fig. 10. The prediction error of our model in different scales.

5.5 Results on Packet Coding Parameters

Different from the existing model with constant coding delay [7], the packet coding parameters in our model can be changed. We evaluate how packet coding parameters (e.g., the finite field size 2^q of $GF(2^q)$ and the number of packets per page ϕ) affect the completion time of coding-based reprogramming protocols. In our model, packet coding is analyzed and we are able to compare coding delay and communication delay to select appropriate packet coding parameters to reduce the total completion time. We use the *Blink* application, and choose $q \in [2, 16]$, and $\phi \in \{8, 16, 24, 32, 40, 48\}$. We note that when q equals 16, decoding needs 65 KB memory space to store inverses, this requirement exceeds the RAM size of most sensor node platforms (e.g., TelosB has 10 KB RAM) [5]. We calculate the completion time for both cases when coding is accounted for and when it is not, the results are shown in Fig. 11.

Case when coding is accounted for. When the number of packets per page ϕ is fixed, the completion time increases with q, because the encoding complexity is positively correlated with q. When the finite field size 2^q is fixed, the completion time increases with ϕ but experiences a temporary decline of $8 \leq \phi \leq 16$. This observation is contrary to the incorrect intuition that the completion time increases continuously with ϕ (more packets to encode/decode) for a fixed q.

Case when coding is not accounted for. For a fixed finite field size 2^q, the completion time declines as ϕ increases. This is because when the number of requests is reduced, the time wasted in waiting for and receiving advertisements is also reduced.

Furthermore, when $\phi = 24$ and $q \geq 4$, the coding delay exceeds that of communication. Additionally, when $\phi = 16$, either a decrease or an increase in ϕ will cause a significant increase in the completion time. When $\phi \geq 32$, a decrease in q is accompanied with a significant decline in completion time. Based on the above observation, we find that the communication delay is about 65 percent of the coding delay when the number of packets per page is 16 for the finite field size 2^8, the minimal total completion time can be achieved when the number of packets per page is close to 24 and the finite field size is close to 2^4. Moreover, the combination $\phi = 16$ and $q = 8$ provides close-to-minimum completion time and ease of implementation for the coding-based reprogramming protocol Rateless Deluge.

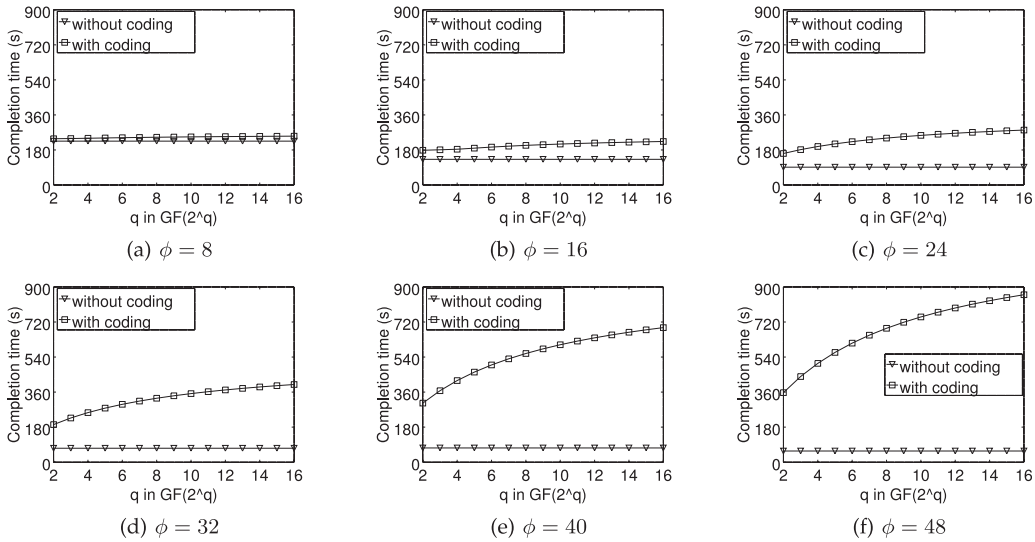
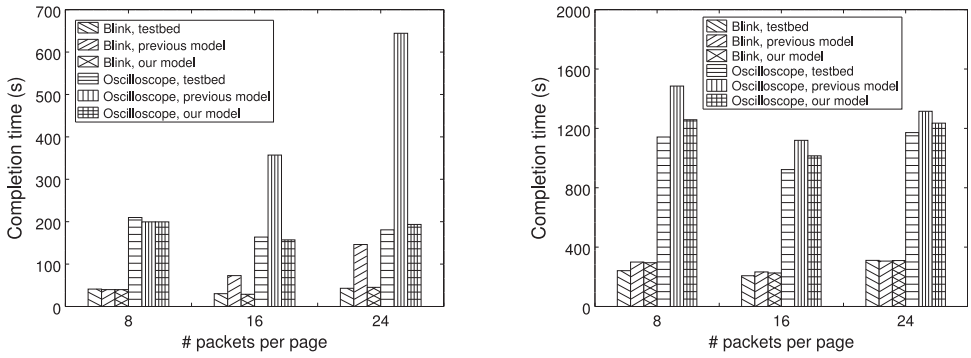


Fig. 11. Analytical results showing the impact of the finite field size 2^q and number of packets per page ϕ.



(a) Dense setup (M1). (b) Sparse setup (M2).

Fig. 12. Testbed results showing the impact of number of packets per page ϕ.

5.6 Results on Overhearing

We conduct testbed experiments to examine the analytical model in sparse and dense networks when the number of packets per page changes. Two different applications are used. For *Blink*, the image size is 2,102 bytes. For *Oscilloscope*, the image size is 12,340 bytes. The analytical results of previous model (the conference version [11] of our model) are shown for comparison. The comparison results are shown in Fig. 12.

From the results, we see that the analytical results of our model match that of testbed experiments in all conditions. However, the analytical results for the previous model match that of testbed experiments only when the number of packets per page ϕ is 8. The dynamic behaviors, especially overhearing, in dense networks have been studied in previous work [4], [8], but overhearing introduced in Section 2.2 will cause such mismatch in coding-based reprogramming protocols as 1) any one encoded packet is identical in coding-based reprogramming protocols, such as Rateless Deluge, which is different from Deluge that each packet is indexed in order, any overheard packet will contribute for decoding; 2) request messages can be sharply reduced when all the packets in one page may have been overheard from neighboring nodes. By importing variables P_{oh} and ϕ_{oh} to consider the effects of overhearing, the analytical results of our model match that of testbed experiments, compared to previous model. As a result, we conclude that overhearing reduces the total completion time by 50-70 percent in the dense networks as depicted in Fig. 12a.

For *Blink*, when the number of packets per page ϕ is 8, 16 and 24, it has 16, 8 and 6 pages each. For *Oscilloscope*, when ϕ is 8, 16 and 24, it has 82, 41 and 28 pages each, respectively. From Fig. 12a, we conclude that, first, when ϕ increases, the total number of pages decreases and the loss on total completion time of the binary image having less pages is worse when the same amount of pages are overheard (P_{oh} is the same). Second, when ϕ increases, the total number of generated encoded packets $\epsilon\phi$ also increases, the packets for the same page will be exposed with longer time period and it will increase the probability of overhearing for neighboring nodes (ϕ_{oh} is increased).

6 Related Work

Deluge [4] is the current benchmark of reprogramming protocols. It establishes the page pipelining mechanism to speed up the three-way handshaking page transmission. MNP [14] employs a sender slection algorithm that makes sure there is at most one source transmitting the code image at a time to reduce collision in one neighbourhood. Freshet [21], Stream [22], Zephyr [23], and ECD [15] are other traditional non-coding-based reprogramming protocols. R3 [24] is a recent work on incremental reprogramming, which is an orthogonal research field against ours, to optimize relocatable code. CoCo [25] exploits link correlation [26], [27] to construct core structure based reprogramming.

Compared with traditional non-coding-based reprogramming protocols, Rateless Deluge [5] is a coding-based reprogramming protocols that is based on Deluge. It utilizes random linear coding in page transmission. SYNAPSE++ [8] is another coding-based reprogramming protocol that LT coding is introduced in data packets transmission. AdapCode [18], RTOC [28], ReXOR [9], LR-Seluge [29] and MT-Deluge [6] are all coding-based. CodeDrip [30] is a coding-based protocol for dissemination of small values. Recent new progresses on data dissemination exploit link correlation [31], [32] and constructive interference [33], [34]. The work in [35] conducts experiments and simulations to show potential of link correlation and network coding for data dissemination. CD [36] exploits link correlation to build a tree structure called "correlated tree" before reprogramming. SYREN [37] is a multi-packet flooding protocol and exploits the synergy among link correlation and network coding to eliminate the overhead of explicit control packets in networks with high correlation and to pipeline transmission of multiple packets. Splash [38] exploits constructive interference to achieve fast and efficient dissemination. However, unlike most coding-based dissemination, Splash has not been designed for and tested in a high-loss environment. Due to its timing requirement during both packet transmission and reception, its performance may not be robust to timing error caused by packet loss.

We focus on constructing an analytical model to evaluate the performance of coding-based reprogramming protocols. Mathematical analysis of the performance of protocols in WSNs is challenging, but research interest is growing in this area [39], [40]. In the area of reprogramming protocols, some preliminary work on performance modeling has been done. Hui and Culler [4] analyze the performance of Deluge, however, their model is only applicable to linear topologies. In contrast, our model is applicable to any network topology.

The analysis in [22], [41] ignores either page pipelining or other protocol details. Dong et al.'s work [7] is the closest to

ours. Their analytical model is applicable to any topology, and they have established a relationship between packet reception rate and number of request packets. We propose refinement on the relationship between PRR and number of request packets. The most significant difference between existing work and ours is that we present a thorough analysis of packet coding operations in three concrete coding-based reprogramming protocols and establish an analytical model. We evaluate one of these coding-based protocols through the analytical model in terms of completion time.

Starobinski and Xiao [13] propose a *theoretical framework* to evaluate the asymptotic performance limits of data dissemination in multi-channel, single-radio WSNs. In comparison, the target of our model is concrete coding-based reprogramming protocols for the more common single-channel, single-radio WSNs.

The analysis in our previous work [11] studies the performance of coding-based reprogramming protocols in terms of completion time and examines influence of both the number of packets per page and finite field size. In this paper, we further evaluate the performance of coding-based reprogramming protocols in extensive testbed experiments. Moreover, we examine the completion time of a representative coding-based reprogramming protocol Rateless Deluge under different image size and inter-node spacing. Overhearing and its influence on the performance of coding-based reprogramming is observed and discussed in dense networks.

We evaluate coding-based reprogramming protocols and examine the impact of several parameters by the analytical model. Rulefit [42], [43] can be used to identify the important impacting ones from these parameters. The κ factor [44] is a new metric as a predictor of performance of network coding protocols and can be used to examine the accuracy of an analytical model for Deluge and our model for coding-based Rateless Deluge in predicting the total completion time.

7 Conclusion

An analytical model serves as an economical alternative to computationally expensive simulations and labor-intensive field experimentations for the performance evaluation of coding-based reprogramming protocols. Moreover, it provides essential clues for ongoing improvements of network protocols. To measure the completion time of existing coding-based reprogramming protocols, we propose a novel high-fidelity analytical model by incorporating overhearing and packet coding. The model is applicable to any coding-based reprogramming protocol by substituting the coding part with protocol specific operations. We examine the analytical model using extensive testbed experiments, the results of a representative coding-based protocol (i.e., Rateless Deluge) confirms the validity of our model. Based on the analytical and numerical experiments, we find that overhearing causes significant reduction of the completion time in dense networks. We also find that coding delay plays a key role in the total completion time compared to the communication delay when packet coding parameters (e.g., the finite field size and encoding coefficient) are appropriately selected. Our model is capable of analyzing and optimizing packet coding parameters of other coding-based reprogramming protocols, such as SYNAPSE++ and ReXOR.

Acknowledgments

This work is supported by National Science and Technology Major Project of China under grant No. 2012ZX03005007. Yu Zhang is supported by NPU Foundation for Fundamental Research under grant No. JC20110268. Yee Wei Law and Marimuthu Palaniswami are partly supported by the EC under contract CNECT-ICT-609112 (SOCIOTAL). Zhe Yang is supported by NSFC under grant No. N2014KA0031 and NPU Foundation for Fundamental Research under grant No. GEKY1003.

References

[1] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Trans. Sens. Netw.*, vol. 6, no. 2, pp. 17:1–17:32, 2010.

[2] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and X. Li, "Does wireless sensor network scale? A measurement study on greenorbs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 10, pp. 1983–1993, Oct. 2013.

[3] J. Zhang, R. Wang, S. Lu, J. Wang, J. Gong, Z. Zhao, H. Chen, L. Cui, N. Wang, and Y. Yu, "Easicprs: Design and implementation of a portable Chinese pulse-wave retrieval system," in *Proc. 9th ACM Conf. Embedded Netw. Sens. Syst.*, 2011, pp. 149–161.

[4] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. 2nd ACM Conf. Embedded Netw. Sens. Syst.*, 2004, pp. 81–94.

[5] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes," in *Proc. 7th Int. Conf. Inform. Process. Sens. Netw.*, 2008, pp. 457–466.

[6] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu, "Exploiting concurrency for efficient dissemination in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 691–700, Apr. 2013.

[7] W. Dong, C. Chen, X. Liu, G. Teng, J. Bu, and Y. Liu, "Bulk data dissemination in wireless sensor networks: Modeling and analysis," *Elsevier Comput. Netw.*, vol. 56, no. 11, pp. 2664–2676, 2012.

[8] M. Rossi, N. Bui, G. Zanca, L. Stabellini, R. Crepaldi, and M. Zorzi, "Synapse++: Code dissemination in wireless sensor networks using fountain codes," *IEEE Trans. Mobile Comput.*, vol. 9, no. 12, pp. 1749–1765, Dec. 2010.

[9] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, "A lightweight and density-aware reprogramming protocol for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 10, pp. 1403–1415, Oct. 2011.

[10] W. Du, Z. Li, J. Liando, and M. Li, "From rateless to distanceless: Enabling sparse sensor network deployment in large areas," *IEEE/ACM Trans. Netw.*, vol. pp, no. 99, pp. 1–14, 2015.

[11] J.-W. Li, S.-N. Li, Y. Zhang, Y. W. Law, X. Zhou, and M. Palaniswami, "Analytical model of coding-based reprogramming protocols in lossy wireless sensor networks," in *Proc. IEEE Int. Conf. Commun.*, 2013, pp. 1867–1871.

[12] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Liu, "Performance of bulk data dissemination in wireless sensor networks," in *Proc. 5th IEEE Int. Conf. Distrib. Comput. Sens. Syst.*, 2009, pp. 356–369.

[13] D. Starobinski and W. Xiao, "Asymptotically optimal data dissemination in multichannel wireless sensor networks: Single radios suffice," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 695–707, Jun. 2010.

[14] S. Kulkarni and L. Wang, "Energy-efficient multihop reprogramming for sensor networks," *ACM Trans. Sens. Netw.*, vol. 5, no. 2, pp. 16:1–16:40, 2009.

[15] W. Dong, Y. Liu, Z. Zhao, X. Liu, C. Chen, and J. Bu, "Link quality aware code dissemination in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1776–1786, Jul. 2014.

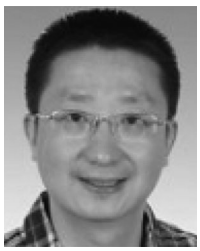
[16] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *Proc. ACM/IEEE 5th Annu. Int. Conf. Mobile Comput. Netw.*, 1999, pp. 174–185.

[17] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," in *Proc. 1st Annu. IEEE Commun. Soc. Conf. Sens., Mesh Ad Hoc Commun. Netw.*, 2004, pp. 517–526.

[18] I.-H. Hou, Y.-E. Tsai, T. F. Abdelzaher, and I. Gupta, "Adapcode: Adaptive network coding for code updates in wireless sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 2189–2197.
[19] M. Doddavenkatappa, M. Chan, and A. Ananda, "Indriya: A low-cost, 3d wireless sensor network testbed," in *Proc. 7th Int. ICST Conf. Testbeds Res. Infrastructures Develop. Netw. Communities*, 2011, pp. 302–316.
[20] Y. W. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, "Secure rateless deluge: Pollution-resistant reprogramming and data dissemination for wireless sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, pp. 5:1–5:22, 2011.
[21] M. D. Krasniewski, R. K. Panta, S. Bagchi, C.-L. Yang, and W. J. Chappell, "Energy-efficient on-demand reprogramming of large-scale sensor networks," *ACM Trans. Sens. Netw.*, vol. 4, no. 1, pp. 2:1–2:38, 2008.
[22] R. Panta, I. Khalil, and S. Bagchi, "Stream: Low overhead wireless reprogramming for sensor networks," in *Proc. IEEE Conf. Comput. Commun.*, 2007, pp. 928–936.
[23] R. K. Panta, S. Bagchi, and S. P. Midkiff, "Efficient incremental code update for sensor networks," *ACM Trans. Sens. Netw.*, vol. 7, no. 4, pp. 30:1–30:32, 2011.
[24] W. Dong, C. Chen, J. Bu, and W. Liu, "Optimizing relocatable code for efficient software update in networked embedded systems," *ACM Trans. Sens. Netw.*, vol. 11, no. 2, pp. 22:1–22:34, 2014.
[25] Z. Zhao, W. Dong, J. Bu, T. Gu, C. Chen, X. Xu, and S. Pu, "Exploiting link correlation for core-based dissemination in wireless sensor networks," in *Proc. 11th Annu. IEEE Commun. Soc. Conf. Sens., Mesh Ad Hoc Commun. Netw.*, 2014, pp. 372–380.
[26] S. M. Kim, S. Wang, and T. He, "Exploiting causes and effects of wireless link correlation for better performance," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 379–387.
[27] Z. Zhao, W. Dong, G. Guan, J. Bu, T. Gu, and C. Chen, "Modeling link correlation in low-power wireless networks," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 990–998.
[28] A. D. Wood and J. A. Stankovic, "Online coding for reliable data transfer in lossy wireless sensor networks," in *Proc. 5th IEEE Int. Conf. Distrib. Comput. Sens. Syst.*, 2009, pp. 159–172.
[29] Y. Zhang and Y. Zhang, "Lr-seluge: Loss-resilient and secure code dissemination in wireless sensor networks," in *Proc. 35th IEEE Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 497–506.
[30] N. d. S. R. Júnior, M. A. Vieira, L. F. Vieira, and O. Gnawali, "Codedrip: Data dissemination protocol with network coding for wireless sensor networks," in *Proc. 11th Eur. Conf. Wireless Sens. Netw.*, 2014, pp. 34–49.
[31] S. Wang, S. M. Kim, Y. Liu, G. Tan, and T. He, "Corlayer: A transparent link correlation layer for energy efficient broadcast," in *Proc. ACM 19th Annu. Int. Conf. Mobile Comput. Netw.*, 2013, pp. 51–62.
[32] T. Zhu, Z. Zhong, T. He, and Z.-L. Zhang, "Achieving efficient flooding by utilizing link correlation in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 121–134, Feb. 2013.
[33] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with glossy," in *Proc. 10th Int. Conf. Inform. Process. Sens. Netw.*, 2011, pp. 73–84.
[34] W. Du, J. C. Liando, H. Zhang, and M. Li, "When pipelines meet fountain: Fast data dissemination in wireless sensor networks," in *Proc. 13th ACM Conf. Embedded Netw. Sens. Syst.*, 2015, pp. 365–378.
[35] S. I. Alam, S. Sultana, Y. C. Hu, and S. Fahmy, "Link correlation and network coding in broadcast protocols for wireless sensor networks," in *Proc. 9th Annu. IEEE Commun. Soc. Conf. Sens., Mesh Ad Hoc Commun. Netw.*, 2012, pp. 59–61.
[36] Z. Zhao, W. Dong, J. Bu, Y. Gu, and C. Chen, "Link-correlation-aware data dissemination in wireless sensor networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 9, pp. 5747–5757, Sep. 2015.
[37] S. Alam, S. Sultana, Y. C. Hu, and S. Fahmy, "Syren: Synergistic link correlation-aware and network coding-based dissemination in wireless sensor networks," in *Proc. IEEE 21st Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2013, pp. 485–494.
[38] M. Doddavenkatappa, M. C. Chan, and B. Leong, "Splash: Fast data dissemination with constructive interference in wireless sensor networks," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 269–282.
[39] R. R. Rout and S. K. Ghosh, "Adaptive data aggregation and energy efficiency using network coding in a clustered wireless sensor network: An analytical approach," *Elsevier Comput. Commun.*, vol. 40, pp. 65–75, 2014.
[40] Y. Wang, M. C. Vuran, and S. Goddard, "Cross-layer analysis of the end-to-end delay distribution in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 305–318, Feb. 2012.
[41] P. De, Y. Liu, and S. Das, "An epidemic theoretic framework for vulnerability analysis of broadcast protocols in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 8, no. 3, pp. 413–425, Mar. 2009.
[42] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," *Ann. Appl. Statist.*, vol. 2, no. 3, pp. 916–954, 2008.
[43] J. Wang, W. Dong, Z. Cao, and Y. Liu, "On the delay performance analysis in a large-scale wireless sensor network," in *Proc. IEEE Real-Time Syst. Symp.*, 2012, pp. 305–314.
[44] K. Srinivasan, M. Jain, J. I. Choi, T. Azim, E. S. Kim, P. Levis, and B. Krishnamachari, "The κ factor: Inferring protocol performance using inter-link reception correlation," in *Proc. ACM 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 317–328.

Jun-Wei Li received the BS degree in computer science from Northwestern Polytechnical University, Xi'an, China, in 2010. He is currently working toward the PhD degree with the School of Computer Science at Northwestern Polytechnical University. His research interests include optimization, code dissemination, and wireless sensor networks.

Shi-Ning Li received the BS and MS degrees in computer science from Northwestern Polytechnical University, Xi'an, China, in 1989 and 1992, respectively. He received the PhD degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2005. He is currently a professor at the School of Computer Science, Northwestern Polytechnical University. His research interests include mobile computing and wireless sensor networks. He is a member of the IEEE.



Yu Zhang received the MS degree in computer science in 2002 from the Northwestern Polytechnical University, Xi'an, China. He was a scholarship researcher at the University of Melbourne, before becoming an associate professor at the School of Computer Science, Northwestern Polytechnical University. His research areas include protocol design, optimization of wireless sensor networks, and power management of mobile computing. He is a member of the IEEE.

Tao Gu received the PhD degree in computer science from the National University of Singapore. He is an associate professor in the School of Computer Science and Information Technology, RMIT University. His current research interests include mobile and pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, cyber physical system, Internet of Things, and online social networks. He is a senior member of the IEEE.

Yee Wei Law received the PhD degree from the University of Twente, and was a research fellow at the University of Melbourne, before becoming a lecturer at the University of South Australia. His main research interests include the security and privacy aspects of sensor networks, smart grids, and more generally the Internet of Things. He received the Best Paper Award at ICTC 2012.

Zhe Yang received the PhD degree in electrical and computer engineering from the University of Victoria, Victoria, British Columbia, Canada, in 2013. He then joined the School of Computer Science, Northwestern Polytechnical University, with the exceptional promotion to associate professor. His research areas include protocol design, optimization, and resource management of wireless communication networks. He is a member of the IEEE.

Xingshe Zhou received the MS degree from the Northwestern Polytechnical University, Xi'an, China, in 1984. He is a professor with the School of Computer Science, Northwestern Polytechnical University. He is the director with the Shaanxi Key Laboratory of Embedded System Technology, Xi'an, China. His research interests include embedded computing and pervasive computing. He is a member of the IEEE.

Marimuthu Palaniswami received the PhD degree from the University of Newcastle. He is now a professor at the University of Melbourne. He was the convener of the ARC Research Network on ISSNIP. His research interests include control, machine learning, signal processing and their applications to smart grids, biomedical engineering, wireless sensor networks, and the Internet of Things. He became a fellow of the IEEE and IEEE distinguished lecturer in machine learning in 2012.