Computer Networks 56 (2012) 2664–2676

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Bulk data dissemination in wireless sensor networks: Modeling and analysis

Wei Dong [a,*], Chun Chen [a], Xue Liu [b], Guodong Teng [a,c], Jiajun Bu [a], Yunhao Liu [d]

[a] Zhejiang Provincial Key Laboratory of Service Robot, College of Computer Science, Zhejiang University, Hangzhou 310027, PR China
[b] School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 0E9
[c] Institute of Service Engineering, Hangzhou Normal University, Hangzhou, 310036, China
[d] School of Software and TNLIST, Tsinghua University, PR China

A R T I C L E I N F O

Article history:
Received 23 April 2011
Received in revised form 8 October 2011
Accepted 14 April 2012
Available online 21 April 2012

Keywords:
Dissemination
Pipelining
Wireless sensor network

A B S T R A C T

Wireless sensor networks (WSNs) have recently gained a great deal of attention as a topic of research, with a wide range of applications being explored. Bulk data dissemination is a basic building block for sensor network applications. The problem of designing efficient bulk data dissemination protocols has been addressed in a number of recent studies. The problem of accurately analyzing the performance of these protocols, however, has not been addressed sufficiently in the literature. In this work, we show a way of accurately analyzing the performance of bulk data dissemination protocols in WSNs. Our model can be applied to practical network topologies by use of the shortest propagation path. Our model is accurate by considering topological information, impact of contention, and impact of pipelining. We validate the analytical results through testbeds and detailed simulations. Results show that the analytical results fit well with the testbed results and simulation results. Further, we demonstrate that the analytical results can be used to aid protocol design for performance optimizations, e.g., page size tuning for shortening the completion time.

1. Introduction

Wireless sensor networks (WSNs) have recently gained a great deal of attention as a topic of research, with a wide range of applications being explored. Bulk data dissemination is a basic building block for sensor network applications. Example protocols such as Deluge [1] and MNP [2], which distribute new binaries into a network, enabling complete system reprogramming.

In many industrial environments, we usually experience a great number of hazards that can range from strong mechanical vibrations, high temperatures, fragile surfaces, and even explosive gases [3]. It is difficult or infeasible to physically collect previously deployed nodes. On the other hand, WSN software often needs to be changed after deployment for a variety of reasons—upgrading node software, correcting software bugs, and patching security holes. Enabling sensor nodes to be reprogrammable over the air is crucial for the maintenance of WSNs.

The problem of designing efficient bulk data dissemination protocols has been addressed in a number of recent studies [1,2,4–6]. Although they demonstrated the efficiency of their approaches through testbeds or simulations, they failed to investigate rigorously the performance of these protocols because of evaluation limitations. For example, testbeds suffer from scalability issues while simulations can either be inaccurate or time-consuming. Both of these limitations call for the need of efficient and accurate performance modeling and analysis, which could be useful to protocol design, performance evaluations and optimizations.

Compared to simulations, analytical models have the following benefits. First, analytical models have simpler interface for use. Most analytical models expose a few

* Corresponding author.
E-mail addresses: dongw@zju.edu.cn (W. Dong), chenc@zju.edu.cn (C. Chen), xueliu@cs.mcgill.ca (X. Liu), teng@zju.edu.cn (G. Teng), bjj@zju.edu.cn (J. Bu), yunhao@greenorbs.com (Y. Liu).

1389-1286/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.04.007

key system parameters for modeling while simulation needs specific implementations. Second, analytical models are more extensible than simulators. For example, current sensor network simulators only support a limited number of platforms while good analytical models can be easily extended by modifying a few system parameters. Third, analytical models have a lower run-time overhead than simulators, and thus are more scalable than simulator by supporting larger network scales.

The problem of accurately analyzing the performance of these protocols, however, has not been addressed sufficiently in the literature. The analysis of [1] only applies to linear structures. The analysis of [5] is far from accurate because it does not consider protocols details. The analysis of [7] also suffers from inaccuracy issues because, for example, it does not model page pipelining (i.e., the data image is segmented into fixed-sized pages to allow spatial multiplexing), which is a common technique for bulk data dissemination. Compared to these modeling approaches, our model is much more accurate. This is important as it can give practical guidelines to network operators. For example, with accurate estimated dissemination time, we can configure the network to enter low power model after dissemination to save energy.

In this work, we show a way of accurately analyzing the performance of bulk data dissemination protocols in WSNs. In particular, we analyze the Deluge protocol, the de facto bulk data dissemination protocol based on TinyOS [8]. Our model can be applied to practical network topologies by use of the shortest propagation path. Our model can also be applied to other dissemination protocols (e.g., MNP [2], Rateless Deluge [6], and CORD [9]) by substituting the protocol-specific factors. Finally, our model is accurate by considering topological information, impact of contention, and impact of pipelining.

We validate the analytical results through testbeds and detailed simulations. We find that the analytical results fit well with the testbed results and simulation results. Further, we demonstrate that the analytical results can be used to aid protocol design for performance optimizations, e.g., page size tuning for shortening the completion time.

Compared to our preliminary work appeared in [10], this paper includes the following extensions: (i) we give the detailed algorithm for the multihop models; (ii) we extend our single hop model to capture impacts of both upstream nodes and downstream nodes; (iii) we discuss extensions to coding-based protocols, two-phase protocols (for heterogeneous networks); (iv) we include testbed evaluations as well as detailed examination of time dynamics of the dissemination process.

The contributions of our work are highlighted as follows:

– To the best of our knowledge, we are the first to propose an accurate model for bulk data dissemination that can be applied to practical network topologies.
– We validate the analytical results through testbeds and extensive simulations, and our analytical results fit well with the testbed results and simulation results.
– We demonstrate how our approach can be used to aid protocol design for performance optimizations.

The rest of this paper is structured as follows: Section 2 describes backgrounds in bulk data dissemination. Section 3 gives an overview of our approach. Section 4 presents the details of our analytical approach. Section 5 shows the experiments and evaluation results. Section 6 discusses the related work. Finally, Section 7 gives a conclusion of this paper.

2. Backgrounds

In this section, we give a brief description of bulk data dissemination in WSNs. In particular, we describe the Deluge protocol, the standard reprogramming protocol in the TinyOS [8] distribution. Deluge [1] uses a three-way handshake and NACK-based protocol for reliability, and employs segmentation (into pages) and pipelining for spatial multiplexing. Here we describe it in three different phases.

(i) Advertisement, Request, and Page Transmission. In this phase, each node advertises about its local images. Note that an image may not be complete during the process of transmission. Deluge enforces strict ordering of page transmission [1]. An image is said to be larger if it has more available pages. When a node (receiver) learns that another node (sender) has a larger image, it will send a request and prepare to receive the data packets. When the sender receives a request, it will transit from the `MAINTAIN` state to the `TX` state. Then it starts transmitting the requested data packets in the current page.

(ii) Page Retransmission. If a receiver loses some packets in a given page, it will remain in the `RX` state, sending requests to the most recently heard neighbor for the missing packets. On the other hand, the sender will enter into the `MAINTAIN` state whenever it completes transmitting a requested page. When it receives a request for retransmission purpose, it sends the packets immediately. When all missing packets in a given page are received, the receiver enters into the `MAINTAIN` state and prepares to receive the next page.

(iii) New Page Transmission. The sender and receivers will enter into the `MAINTAIN` state whenever a page completes. At this time, the receiver will prepare to receive the next page and increase the advertising frequency by resetting its advertisement timer. When the sender hears an inconsistent advertisement from any of the receivers, it will also reset its advertisement timer to let the receivers learn about the availability of a larger image in time. When a receiver finds a larger image, it will request to the sender for the next page. After the sender receives a new page request, it will start transmitting immediately.

Many other dissemination protocols are based on Deluge. For example, MNP [2] additionally provides a detailed sender selection algorithm to choose a local source of the code which can satisfy the maximum number of node;

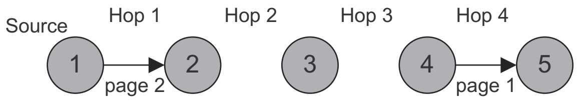

Fig. 1. Pipelining. Example four-hop network. A three-hop spacing is required for effective spatial multiplexing. In this example, a simultaneous broadcast from 1 and 3 would collide at 2.

Rateless Deluge [6] uses random linear codes to encode a data packet to reduce the retransmission cost.

Below, we identify two behaviors that these protocols share in common which should be captured by our analytical approach.

(i) Segmentation and Pipelining. Most bulk data dissemination protocols [1,2,5,6] take advantage of pipelining to allow parallel transfers of data in networks. Pipelining is done through segmentation: a program image is divided into several pages, each of which contains a fixed number of packets. Instead of completely receiving a whole program image before forwarding it, a node becomes a source node after it receives only one complete page. For transmission of a large program image, pipelining can increase the overall throughput significantly [4]. In order to realize the full benefit of spatial multiplexing, the protocol need to ensure that transfers of different pages do not interfere with each other. Fig. 1 shows that the concurrent transmission of page 2 and page 1 must be at least three hops away to avoid collisions.

(ii) Negotiation. Most bulk data dissemination protocols [1,2,5,6] use negotiation-based approach [4] to avoid the broadcast storm problem. A simple negotiation protocol contains three types of messages [4]: (1) ADV: the source node advertises its received object (metadata); (2) REQ: its neighbors send back requests after receiving the ADV to notify the source node about which objects are needed; (3) DATA: the source node only sends out the requested objects. Fig. 2 shows the state transition diagram of a typical negotiation-based dissemination protocol, Deluge. Negotiation helps Deluge minimize the number of senders to advertise in a time interval. It further reduces the number of senders by giving higher priority to a node that transmits a page with a smaller page number [4].

3. Overview

In this section, we formally present the problem we are addressing, and present our approach towards solving it.

3.1. Problem formulation

We are interested in determining the completion time of a given page for any sensor node in the network. More formally, given a data image of P pages and a network of N sensor nodes, we are interested in estimating the completion time of a given page under a specific segmentation scheme. Our model in this paper does consider different page sizes that may be utilized by different protocols (one page consists of N_{pkt} packets where N_{pkt} is a model parameter that can be varied). We will use the denotation C_k^p to designate the completion time of the pth page at node k. Thus, we are interested in determining the completion time, C_k^p, for all $1 \leqslant k \leqslant N$ and $1 \leqslant p \leqslant P$.

The completion time of a given page for a sensor node depends on the following factors. (i) The completion time of the page of the previous-hop node. This requires finding the propagation path and determining the completion times for all the nodes along the path. (ii) The single-hop propagation time of the page. This requires capturing protocol details and effects of contention to avoid unrealistic assumptions. Our goal is to develop an efficient and accurate model that captures the propagation behavior of bulk data dissemination protocols in terms of completion time at the per page basis.

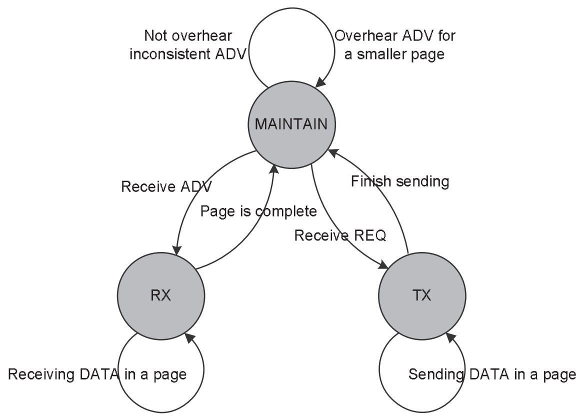

Fig. 2. State transition diagram of the Deluge protocol.

3.2. Approach overview

A high level block diagram of our approach is shown in Fig. 3 with pointers to sections where different parts are described in this paper. The centerpiece is a protocol model. The protocol model models two behaviors that bulk data dissemination depends on. (i) The multihop

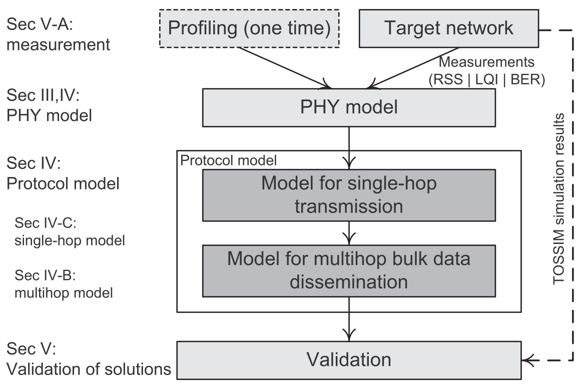
Sec V-A: measurement
Profiling (one time)
Target network
Measurements (RSS | LQI | BER)
Sec III,IV: PHY model
PHY model
Sec IV: Protocol model
Protocol model
Model for single-hop transmission
Sec IV-C: single-hop model
Sec IV-B: multihop model
Model for multihop bulk data dissemination
TOSSIM simulation results
Sec V: Validation of solutions
Validation

Fig. 3. Approach overview.

propagation behavior for all the pages. (ii) The single-hop propagation behavior for one page. The multihop propagation behavior depends on single-hop propagation behaviors amongst other factors such as propagation path and impact of pipelining. The single-hop propagation behavior depends on the physical-level model such as packet reception ratio (*PRR*).

The physical-level model captures *PRR* of a single link. We use the same approach as in [11] to model this dependency, i.e., via measurements in a one-time profiling experiment. The profiling is done for each sensor node platform, and can be reused. The physical-level model is seeded by link-wise measurements in the target WSN, such as the Received Signal Strength Indicator (*RSSI*) and the Link Quality Indicator (*LQI*) provided by the radio hardware [12]. This seeding now makes the upper layer models amenable to numeric solutions.

We validate the entire approach by comparing the completion times estimated by the modeling approach with both testbed results and TOSSIM simulation results.

4. Analytical approach

In this section, we present the details of our analytical approach. First, we give baseline notations in Section 4.1. Then, we describe the multihop model and single-hop model in Section 4.2, 4.3, respectively, which constitute the protocol model for the Deluge protocol [1]. We show the extensibility of our approach by extending our model to other dissemination protocols, i.e., MNP [2], Rateless Deluge [6], and CORD [9] in Sections 4.4–4.6, respectively.

4.1. Baseline notations

Before we present our analytical approach, we first define the following notations:

- P is total number of pages for a bulk image. One page consists of N_{pkt} packets.
- $G = (V,E)$ is a direct graph, where each vertex $n_i \in V (1 \leqslant i \leqslant |V| = N)$ represents a sensor node, and each edge e_{ij} is assigned with a weight $w_{ij} = T_{ij}^{pg} \in E$ $(1 \leqslant i,j \leqslant N)$ which represents the singe-hop transmission time for one page.
- PT_k is the Dijkstra shortest path from a source node, s, to node k. $|PT_k|$ is the number of vertices in the path. $len(PT_k)$ is the weighted path length, i.e.,

$$len(PT_k) = \sum_{e_{ij} \in PT_k} w_{ij} \quad (1)$$

  By its definition, among all the paths from s to k, PT_k is the shortest path with the smallest weighted path length.
- C_k^p is the completion time of page p at node k.
- $T_{ij}^{adv}, T_{ij}^{req}, T_{ij}^{data} (1 \leqslant i,j \leqslant N)$ are the single-hop transmission time spent in advertisement, request, and data transmission. Their sum equals to the overall single-hop transmission time, i.e.,

$$T_{ij}^{pg} = T_{ij}^{adv} + T_{ij}^{req} + T_{ij}^{data} \quad (2)$$

- ρ_i is the effective number of neighboring nodes of i. i.e.,

$$\rho_i = \sum_{\forall k: PRR_{ki} > 0.1} PRR_{ki} \quad (3)$$

  where PRR_{ki} is the packet reception ratio from k to i. Note that we use a *PRR* threshold of 0.1 [13] to exclude neighbors with very poor link qualities. α_i is the fraction of upstream neighboring nodes of i, and β_i is fraction of the downstream neighboring nodes of i, i.e.,

$$\alpha_i = \frac{\sum_{len(PT_k) < len(PT_i)} PRR_{ki}}{\rho_i} \quad (4)$$

$$\beta_i = \frac{\sum_{len(PT_k) > len(PT_i)} PRR_{ki}}{\rho_i} \quad (5)$$

- τ_l, τ_r, and λ are Deluge's design and implementation parameters [1]. τ_l is the approximate time interval between two advertisements; τ_r is the maximum time interval between two requests; λ is the request threshold such that when a node exceeds λ number of requests, it will transit into the `MAINTAIN` state and wait for another advertisement before making additional requests.

4.2. Modeling multihop propagations

To determine the completion time of a given page for a sensor node, C_k^p, we need to find the propagation path originated from the source node, s, and determine the completion times of all the nodes along the path, considering the impact of pipelining.

We will use T_{ij}^{pg} to denote the propagation time for one page from node i to node j. The computation of T_{ij}^{pg} will be described in the next subsection. Because a page transmission starts when any of the upstream nodes completes the page, we compute the propagation path as the shortest path from the source node, where T_{ij}^{pg} denotes the cost metric between a pair of neighboring nodes. Thus, we employ the Dijkstra algorithm [14] to determine the propagation path to any node, say node k, in the network. We denote this path as PT_k.

Next, we will compute the completion times for all the nodes along the path by considering the impact of pipelining. We will use C_q^p to denote the completion time of the pth page at the qth node along the path PT_k to simplify the notations $(1 \leqslant q \leqslant |PT_k|)$. The completion time, C_q^p, depends on two constraints.

- *C1*: it is no smaller than $C_{q-1}^p + T_{(q-1)q}^{pg}$, i.e., the reception time of page p at the previous node plus the single-hop propagation time.
- *C2*: it is no smaller than C_{q+3}^{p-1} when $q + 3 \leqslant |PT_k|$, or $C_{|PT_k|}^{p-1}$ when $q + 3 > |PT_k|$. It is because page $p - 1$ should go beyond at least three hops away before page p can be transmitted to avoid concurrent transmissions and collisions (see Fig. 1).

Based on the above two constraints, we have,

$$C_q^p = \max\left(C_{q-1}^p + T_{(q-1)q}^{pg}, C_{q'}^{p-1}\right) \quad (6)$$

where $q' = \min(q+3, |PT_k|)$. We start with $C_1^1 = C_{q'}^0 = C_0^p = T_{01}^{pg} = 0$, and recursively compute the completion times along the path, PT_k, according to Eq. (6).

Finally, we get the value of C_k^p along the path PT_k. We will denote it as $C_k^p[k]$ (i.e., the C_k^p calculated using path PT_k) in order to differentiate the completion times of node k that may be involved in the computation of completion times for other nodes, where node k acts as an intermediate node in the propagation path. We use $C_k^p[k']$ to denote the completion time of node k using path $PT_{k'}$ containing k. We always select the maximum completion time because any transmissions at the downstream nodes of node k will postpone the transmission to node k, i.e.,

$$C_k^p = \max_{1 \leqslant k' \leqslant N} (C_k^p[k']) \tag{7}$$

Algorithm 1. Completion time computation

```
  Input: $G = (V, E)$ where $|V| = N$, and
   $w_{ij} = T_{ij}^{pg} (1 \leqslant i, j \leqslant N)$
 Output: $C_k^p$ where $1 \leqslant k \leqslant N$, and $1 \leqslant p \leqslant P$
 1: procedure ComputAll
 2:   for $k \leftarrow 1, \ldots, N$ do
 3:     ComputC($k$)
 4:   return
 5: procedure ComputC($k$)      ▷ Comp. time of k using
    one path
 6:   Find the Dijkstra path from $s$ to $k$ as $PT_k$
 7:   for $p \leftarrow 1, \ldots, P$ do
 8:     for $q \leftarrow 1, \ldots, |PT_k|$ do
 9:       if $q = 1$ then
10:         $C_1 \leftarrow 0$
11:       else
12:         $C_1 \leftarrow C_{q-1}^p + T_{(q-1)q}$
13:       if $p = 1$ then
14:         $C_2 \leftarrow 0$
15:       else if $q + 3 \leqslant |PT_k|$ then
16:         $C_2 \leftarrow C_{q+3}^{p-1}$
17:       else
18:         $C_2 \leftarrow C_{|PT_k|}^{p-1}$
19:       $C_q^p[k] \leftarrow \max(C_1, C_2)$
20:   for $q \leftarrow 1, \ldots, |PT_k|$ do
21:     for $p \leftarrow 1, \ldots, P$ do
22:       if $C_q^p[k] > C_q^p$ then
23:         $C_q^p \leftarrow C_q^p[k]$
24:   return
```

Algorithm 1 outlines the process to compute C_k^p. The input is a directed graph $G = (V, E)$ where V is the set of sensor nodes, and E is the set of directed edges between two nodes whose weights correspond to the single-hop propagation times, i.e., T_{ij} (the computation of T_{ij} will be described in the next subsection).

There are several issues that worth emphasizing in our multihop modeling approach. First, we choose the shortest path to calculate the completion time. In the data dissemination process, a dissemination tree is implicitly created by the dissemination protocol starting from the source node. Each node uses broadcast to disseminate the data. A page transmission starts when any of the upstream nodes completes the page. Hence, the page completion time can be calculated as if the page is propagated from the shortest propagation path. Indeed, there may be some hotspot nodes which experience more packet transmissions. Many practical protocols (e.g., Deluge [1], MNP [2], Rateless Deluge [6]) already use techniques such as overhearing, request suppression to reduce collisions and contentions. Second, our model captures the effect that nodes near the center of the network may experience longer transmission delays than edge nodes, which is shown by early experimental studies [1]. This is because, in our model, a center node is more likely to be involved in the propagation path for other nodes. Hence the transmission of the current page is more likely to be postponed by the transmission of the previous page in two hops range. This is captured by Eq. (6). Third, our current model only considers a single source node. Also we consider a relatively simple interference model. In practice, the combination of parallel transmissions, even if happening three hops away, may causes interference in the new transmission. If such a realistic model is considered, our PHY model should be adapted to employ the *SINR-PRR* model in recent studies [15–18]. We will consider multiple source nodes and more accurate interference models as directions of our future work.

4.3. Modeling single-hop propagations

In the previous subsection, we have used the single-hop page transmission time to seed the multihop propagation model. In this subsection, we will determine the single-hop transmission time, T_{ij}^{pg}, based on the pair-wise *PRR* value given by the physical-level model.

The computation of T_{ij}^{pg} was preliminarily analyzed in [1] for linear structures. Compared to prior work, our approach has two important extensions. First, our approach considers pair-wise link characteristics (e.g., *PRR*). This is important for complex network topologies where a poor communication link may significantly impact the propagation time. Second, our approach captures contentions much more accurately. Effects of upstream nodes and downstream nodes are captured. A node cannot transmit a new page when any of its upstream nodes transmit or when any of its downstream nodes request for an older page. This is important to make our model extensible to recently proposed coding-based dissemination protocols like Rateless Deluge [6]. These extensions are necessary to make our approach applicable to practical network topologies and general bulk data dissemination protocols.

The page transmission time from node i to node j, T_{ij}^{pg} is the sum of time spent in advertisement, request, and data transmission, as is given by Eq. (2).

The expected time for node j to receive advertisement from node i is given by,

$$T_{ij}^{adv} = \frac{1}{PRR_{ij}} \tau_l (1 + N_{supp}) \tag{8}$$

where τ_l is the approximate time interval between two advertisements; PRR_{ij} is the packet reception ratio from node i to node j; and N_{supp} is the expected number of advertisement suppressions. To determine PRR_{ij}, we need to create an empirical relationship to a specific measurement between two nodes. We express this relationship as a function $f(\cdot)$, such that $PRR_{ij} = f(M_{ij})$, where M_{ij} denotes a certain measurable metric between two nodes, e.g., *RSSI* [12]. We determine $f(\cdot)$ from a prior profiling study. Note that this function models hardware properties rather than wireless propagation in an actual deployment. Thus, such prior profiling study is possible. The profiling methodology to determine $f(\cdot)$ will be discussed in Section 5.1. We set $N_{supp} = 1$ as reported in [1].

The expected time for node j to send requests is given by,

$$T_{ij}^{req} = \frac{\tau_r}{2} N_{req} + \left(\left\lceil \frac{N_{req}}{\lambda} \right\rceil - 1 \right) T_{ij}^{adv} \tag{9}$$

where $\frac{\tau_r}{2}$ is expected time between two requests; N_{req} is the expected number of requests. We note that in Deluge, when a node exceeds its limit of λ requests, it must transit to the MAINTAIN state and wait for another advertisement before making additional requests. Thus, $(\lceil N_{req}/\lambda - 1 \rceil) T_{ij}^{adv}$ is the expected amount of time spent waiting for additional advertisements when the request limit of λ is exceeded. In the analysis of [1], the value of N_{req} is experimentally fixed as 5.4. We find that the value of N_{req} is related to the weighted average *PRR* of a given network. The details will be described in Section 5.2.

The expected time for node i to transmit a page to node j is given by,

$$T_{ij}^{data} = N_{pkts} \frac{1}{PRR_{ij}} T_{pkt} (\alpha_i + \beta_i) \rho_i \tag{10}$$

where N_{pkts} is the number of packets in a page; PRR_{ij} is the packet reception ratio from node i to node j as mentioned earlier; T_{pkt} is the transmission time for a single packet; ρ_i is the effective number of neighboring nodes of i; α_i is the fraction of upstream neighboring nodes of i; β_i is the fraction of downstream neighboring nodes of i. They are formally defined in Section 4.1.

We assume the data transmission time, T_{ij}^{data}, has a linear relationship to $(\alpha_i + \beta_i)\rho_i$ due to two reasons. First, node i delays when any of its upstream nodes $(\alpha_i \rho_i)$ transmits. Second, node i remains transmitting (the current page) when any of its downstream nodes $(\beta_i \rho_i)$ loses some packet in the current page and sends a request to node i. $\alpha_i = \beta_i = 0.5$ implies that node i has the same number of upstream nodes and downstream nodes.

4.4. Extension to MNP

It should be noted that our analytical model can be extended to bulk data dissemination protocols other than Deluge [1], by substituting protocol-specific factors. In this subsection, we extend our analytical method to a different dissemination protocol, MNP [2], by substituting the single-hop propagation time T_{ij}^{pg}.

For MNP, the single-hop propagation time T_{ij}^{pg} is computed as follows [7],

$$T_{ij}^{pg} = T_{ij}^{adv} + T_{ij}^{data} \tag{11}$$

The time for requests is not included in. The reason is as follows. In MNP, the advertising node sends a maximum number of κ advertisement messages to allow sender selection. Thus, by the end of the waiting interval T_{ij}^{adv}, the advertising node has received at least one request messages and is ready to serve a receiving node.

The expected time for κ advertisement messages is given by,

$$T_{ij}^{adv} = \kappa \cdot \Delta\tau \tag{12}$$

where $\Delta\tau$ is the average duration between two advertisements.

After T_{ij}^{adv}, the source sends N_{pkts} packets in a page. It also sends a `Start Download` and `End Download` message signifying the start and end of each page. Thus, the expected time for data transmission of a page is given by,

$$T_{ij}^{data} = (N_{pkts} + 2) \frac{1}{PRR_{ij}} T_{pkt} (\alpha_i + \beta_i) \rho_i \tag{13}$$

4.5. Extension to coding-based protocols

Recently, several coding-based dissemination protocols are proposed to improve the performance of Deluge in lossy networks like Rateless Deluge [6]. Rateless Deluge uses rateless code to encode a packet before transmission. After receiving an expected number of encoded packets, the receiving node uses Gaussian elimination to decode the packets.

For Rateless Deluge [6], the data transmission time is replaced by,

$$T_{ij}^{data} = N_{pkts} \frac{1}{PRR_{ij'}} T_{pkt} \alpha_i \rho_i + T_{coding} \tag{14}$$

where T_{coding} is the data decoding time for a page. In Rateless Deluge, it is reported as 1.96 s for a page consisting of 20 packets [6]. Note that β_i in Eq. (10) is not included in because an encoded packet can be decoded at multiple receivers. However, the transmitted packets of node i is dominated by the worst link from node i, which is denoted as $PRR_{ij'}$. For example, we consider a grid topology with internode spacing d. The maximum transmission range of a sensor node is denoted by $d_{\max}$. Then, $PRR_{ij'}$ can be calculated as,

$$PRR_{ij'} = g\left(\left\lfloor \frac{d_{\max}}{d} \right\rfloor \cdot d \right) \tag{15}$$

where $g(\cdot)$ is a function that maps a distance to the *PRR*. The above equation shows a way of calculating the "worst" *PRR* in grid topologies. We obtain the *PRR* by mapping the maximum communication distance. Such a model can well capture the packet transmission performance without interference. It is appropriate in our case because practical dissemination protocols employ many techniques such as overhearing, suppression, to greatly reduce interference in a neighborhood. For example, when one node is transmitting in a neighborhood, other nodes will suppress their own traffic and overhear the current transmissions.

4.6. Extension to CORD

In this subsection, we discuss the extensions to heterogeneous networks with respect to a typical two-phase dissemination protocol, CORD [9].

CORD employs a two-phase approach to disseminate the data. In the first phase, super nodes are selected to form a connected dominating set (CDS) and data is disseminated to all super nodes. In the second phase, super nodes transfer data to non-super nodes.

An important property of super node selection in CORD is that all super nodes are connected and a non-super node has at least one neighboring super node. Our model can be extended to CORD as follows:

– Compute the collected dominating set (CDS) of the network according to the algorithm employed by the protocol, e.g. Cheng's algorithm [19] for CORD.
– Compute the data dissemination time of the first phase. We use the shortest path algorithm given in Algorithm 1 to model the multihop dissemination process among the CDS nodes.
– Compute the data dissemination time of the second phase. We use our single hop model to compute the single hop dissemination time from super nodes to non-super nodes. Since the transmissions from super nodes to non-super nodes are in parallel, the dissemination time of the second phase is dominated by the longest single hop transmission time.
– The overall dissemination time can be computed by adding up the dissemination times of the first phase and the second phase.

4.7. Discussions

In this subsection, we discuss the messaging complexity and the energy overhead of the protocols.

Messaging complexity. Most dissemination protocols employ negotiation-based approach to achieve reliability. For example, Deluge and MNP use three types of messages, i.e. advertisement message, request message, and data message. Typically, the number of data messages dominates the total transmission cost. The transmission cost of each message types can be different for different protocols. For example, MNP has a lower number of data messages than Deluge because sender selection in MNP mitigates packet interference. However, MNP has a higher number of advertisement messages because it needs a sufficient number of advertisements to allow sender selection.

Energy cost. Energy consumption is an important metric for sensor network protocols. According to [5], the energy consumption in an nxn grid network can be approximated as

$$E = \sum_{h=1}^{h_{max}} N_{Nh} E_h = \sum_{h=1}^{h_{max}} \left[N_{Nh} \frac{h N_p N_{pkt} C}{P_s} \right] \tag{16}$$

where N_p is the number of pages consisting of the image; N_{pkt} is the number of packets consisting of a page; C is the energy cost of transmitting and receiving a packet once and P_s is the probability of successful transmission of a packet over a single hop; h_{max} is the network diameter and N_{Nh} be the number of nodes at hop h. For nxn grid, $N_{Nh} = h + 1$ when $1 \leqslant h \leqslant n - 1$ or $N_{Nh} = 2n - h - 1$ when $n \leqslant h \leqslant 2n$.

We can see several facts from this equation. First, the data image size greatly impacts the total energy consumption. Second, the link qualities also impact the energy consumption. MNP achieves a lower energy cost compared to Deluge. One of the reasons is that P_s will be higher since sender selection in MNP effectively mitigates interference.

5. Experiments and evaluations

We use both a testbed of 16 TelosB motes and a bit-level simulator, TOSSIM [20], to conduct experiments and evaluations. In Section 5.1, we describe the profiling experiment to determine the *PRR* value per link. In Section 5.2, we present our parameter settings. Specifically, we describe how we determine the number of request messages as mentioned in Section 4. In Section 5.3, we validate analytical results by both testbed results and TOSSIM simulation results. Section 5.4 compares the performance of non-coding and coding-based protocols using our analytical model. In Section 5.5, we show the computational time of our approach as compared to simulation time and actual dissemination time. Finally, in Section 5.6, we demonstrate how our analytical model can be used to aid protocol design for performance optimizations.

In [5], authors analyze the dissemination time and energy cost of three dissemination protocols. The approach does not consider protocol details, such as advertisement-request-data message exchanges. The analytical results are not compared against simulation or testbed results. In [7], authors employ epidemic theory to investigate the propagation behavior of protocols such as Trickle, Deluge. The approach is used to study the asymptotic behavior for vulnerability analysis. The approach considers protocols details. However, it fails to capture pipelining effects and the impact of link qualities to the multihop propagation path. The analytical results are compared against a simulator called JProwler, but are not validated against high fidelity simulators like TOSSIM and testbed experiments.

We do not compare with these approaches since these approaches can derive inaccurate results for our settings. For example, according to the analysis of [5], the completion time for the Blink application is nearly 4000 s even in a dense 10×10 square (i.e., PRR = 0.98). This is inaccurate compared to less than 500 s in TOSSIM.

In the following, we will use "d1" x "d2" x "spacing" to denote a network topology where d1 is the number of nodes in a row, d2 is the number of nodes in a column, and spacing is the internode spacing in feet. For example, the denotation "$10 \times 10 - 15$" means a 10×10 grid with internode spacing equal to 15 ft (there are 100 nodes); "$15 \times 15 - 5$" means a 15×15 grid with internode spacing equal to 5 ft (there are 225 nodes). The topologies for simulations are constructed by the LossyBuilder utility in the TinyOS distribution. This utility generates an nss file, describing the *BER*s between a pair of nodes. The linear topology can be considered a special case of the grid by representing just the edge [1]. For example, in this paper, we have considered $N \times 2$ linear networks where N ranges from 10 to 48. In all the experiments, we use a single source node, locating at one corner of the topology.

5.1. Profiling experiments

In order to determine the profile: $f(\cdot)$: measurements $\mapsto$ *PRR*, we need to do a set of measurement experiments. For a given radio, we can choose a specific measurable metric, e.g., *PRR*, *RSSI*, *LQI*, or *BER*. The *PRR* is a direct measurement of link quality. However, it may be costly to measure because a node needs to transmit a number of packets to get an accurate estimate.

There are two approaches to obtain the PRR statistics. (i) The *active* approach requires each node transmit/receive a number of measurement packets. For an N node network, the active approach requires O (N) trials where each node transmits in turn while receivers measure the channel condition. This approach is doable for small scale networks and is employed in [11,15,21]. Some works in WSNs [12,17] correlates *RSSI* to *PRR*. This approach effectively reduces the packets need to be transmitted because nodes only need to get the value of *RSSI* from one packet. The *RSSI* values can be measured by having each node take in turn and sending a set of broadcast packets. For a given broadcasting sender, rest of the nodes record the measurements. For an N node network, the measurement requires $O(N)$ measurement steps and provides the metrics for all the $N(N-1)$ links. (ii) To mitigate the profiling overhead, we can instead use a *passive* approach employed in [18]. In the passive approach, each node passively listens to packet receptions at run-time. The collected PRR statistics are then collected to the sink node via the collection protocol (e.g. CTP [22]) typically exists in a sensor network.

In our testbed experiments, we profile pair-wise *PRR*s directly in order to simplify the validation process. In the profiling experiment, we let each node broadcast 200 packets (with 36 bytes packet length) in turn; other nodes count the received packets in order to determine the *PRR* over the link. We gather the pair-wise *PRR*s of the network by the CTP data collection protocol [22]. In our simulations,

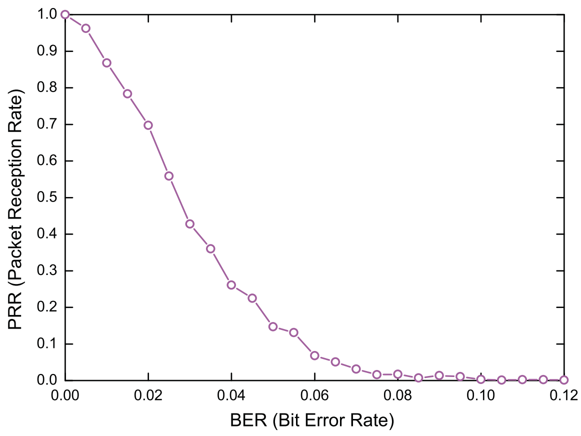

Fig. 4. The correlation between *BER* and *PRR* in TOSSIM, $f_3(\cdot)$: *BER* $\mapsto$ *PRR*.

we use TOSSIM [20] for validation. As TOSSIM 1.x uses the bit error rate (*BER*) to describe a link, we determine the profile: $f_3(\cdot)$:*BER* $\mapsto$ *PRR*, via the `CountRadio` benchmark in the TinyOS distribution [8]. Fig. 4 plots the correlation between *BER* and *PRR* in TOSSIM (the packet length is 36 bytes).

5.2. Parameter settings

According to the implementation of Deluge [1], we set $\tau_l = 2$ s, $\tau_r = 0.5$ s, $\lambda = 2$. We determine T_{pkt}, i.e., the transmission time of a packet as follows. For TelosB motes used in the testbed,

$$T_{pkt}[\text{TelosB}] = \frac{36\ \text{bytes/pkt} \times 8\ \text{bits/byte}}{250\ \text{Kbps}} + \frac{10\ \text{ms}}{2} \approx 5\ \text{ms} \tag{17}$$

where 10 ms is the maximum initial backoff delay for the CC2420 driver in TinyOS. For Mica2 motes used in the simulation of TOSSIM,

$$T_{pkt}[\text{Mica2}] = \frac{36\ \text{bytes/pkt} \times 8\ \text{bits/byte}}{19.2\ \text{Kbps}} + \frac{10\ \text{ms}}{2} = 20\ \text{ms} \tag{18}$$

where 10 ms is the maximum initial backoff delay for the CC1000 driver in TinyOS.

We have also mentioned that the value of N_{req} is related to the weighted average *PRR* of a given network. In the following, we will experimentally determine the value of N_{req}.

Intuitively, the value of N_{req} is related to the average *PRR* of a network as in a lossy network the value of N_{req} will be large while vice versa. In order to calculate the average *PRR* of a network, we first setup a cutoff value to exclude poor links [13]. However, we find that the average *PRR* cannot directly reflect the "connectivity" of the network. For example, we find that the average *PRR* for a 10×2 linear structure with 10 ft spacing is even lower than a 10×2 linear structure with 15 ft spacing. The reason is that the network 10×2–10 also has a large number of intermediate links which makes the average *PRR* small.

In order to address the above issue, we use the weighted average *PRR* of a network. We divide *PRR*s into several discrete regions, e.g., $(0.1, 0.2], \ldots, (0.9, 1]$. Each region is associated with a weight that is defined as follows.

$$w_1(R) = \#\ \text{of links whose } PRR \in R \tag{19}$$

We associate each *PRR* with a weight function $w_2(\cdot)$:*PRR* $\mapsto$ weight. The weight of a *PRR* is the weight of the region the *PRR* falls within.

$$w_2(PRR) = w_1(R), PRR \in R \tag{20}$$

The normalized weight of a *PRR* is hence

$$w_3(PRR) = \frac{w_2(PRR)}{\sum_{\forall PRR'} w_2(PRR')} \tag{21}$$

Thus, the weighted average *PRR* of a network is calculated as follows.

$$PRR_{nw} = \sum_{1 \leqslant i,j \leqslant N} PRR_{ij} \cdot w_3(PRR_{ij}) \tag{22}$$

Table 1
The weighted average *PRR* of a network (PRR_{nw}).

Topology	Weighted avg. *PRR*	Topology	Weighted avg. *PRR*
$10 \times 10 - 5$	0.918	$10 \times 2 - 5$	0.98
$10 \times 10 - 10$	0.825	$10 \times 2 - 10$	0.916
$10 \times 10 - 15$	0.784	$10 \times 2 - 15$	0.86
$10 \times 10 - 20$	0.428	$10 \times 2 - 20$	0.61

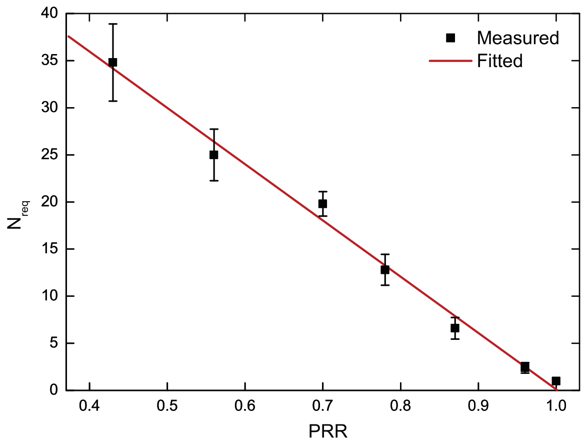

Fig. 5. Correlation of PRR_{nw} and N_{req}.

Table 1 shows the weighted average *PRR* of some typical network topologies. Finally we experimentally determine the relationship of PRR_{nw} and N_{req} as follows. We setup two nodes running the Deluge protocol under different *PRR* settings. For each experiment, we run at least five times, and obtain both the mean value and standard deviation. Finally, we fit the data to a linear function. Fig. 5 shows our results. Note that $N_{req} \approx 6$ when $PRR_{nw} = 0.9$, this result is close to that $N_{req} = 5.4$ when $PRR_{nw} = 0.9$ reported in [1].

5.3. Model evaluation and validation experiments

In this section, we validate our analytical results by both testbed results and detailed simulation results. All experiments are conducted at least five times. For the TOSSIM simulation results, we omitted the standard derivations because the variance is small.

5.3.1. Testbed results

Our testbed consists of 16 TelosB motes with CC2420 radio. The radio has a maximum transmission rate as high as 250 Kbps. According to Section 5.2, we set $T_{pkt} = 0.005$ s. We place the motes in a 4×4 grid with transmission power level set to 2 with internode spacing about 0.4 ft. The TelosB motes support 31 power levels. We have used power level 2 to simulate the multihop behaviors of data dissemination. With such a power level, nodes can only communicate within a short distance. Hence, we have used a short internode spacing of 0.4 ft. The Deluge base station is placed in one corner of the grid. In addition, we introduce another node (sync node) with the maximum transmission power in order to periodically synchronize all other nodes, such that the actual dissemination time can be obtained. We use the packet-level time synchronization interface provided by TinyOS, which can achieve synchronization accuracy in less than 1 ms [23].

Table 2
4×4 testbed results and analysis results (s).

Benchmark	Actual time (mean/sd)	Analysis time	Error (%)
Blink	32.57/4.3	34.69	6.5
Blink + Deluge	179.94/17.2	185.48	3

Table 3
TOSSIM results and analysis results (s).

Topology	Dissem. time	Analysis time
$10 \times 10 - 5$	260.02	264.46
$10 \times 10 - 10$	494.87	484.97
$10 \times 10 - 15$	541.12	564.31
$10 \times 10 - 20$	1498.17	1472.10
$10 \times 2 - 5$	37.57	48.16
$10 \times 2 - 10$	111.14	136.15
$10 \times 2 - 15$	220.93	236.56
$10 \times 2 - 20$	849.6	828.37

We would like to validate the accuracy of our approach using relatively small data images and relatively large images. The `Blink` application represents a simple application which compiles to 7 pages. The `Blink` application with Deluge support (i.e., `Blink + Deluge`) represents a relatively complex application which compiles to 35 pages. The N_{req} used in the experiments is determined as follows: we first measure pair-wise *PRR*s of the network; then we calculate the network *PRR*, PRR_{nw}, following the approach described in Section 5.2; finally, we obtain N_{req} using the mapping function depicted in Fig. 5. Table 2 shows the actual dissemination time and analysis time for two benchmarks. Note that `Blink + Deluge` represents the `Blink` application with Deluge support, i.e., reprogrammable. Hence, it is much larger than `Blink`. We can see that our approach can accurately predict the dissemination time with approximately 5.5 s maximum absolute error.

5.3.2. Simulation results

In order to investigate the generality and scalability of our approach, we use the bit error model of TOSSIM. We validate our approach with different internode spacings (from 5 ft to 20 ft), different network topologies (i.e., square and linear structures) in a larger scale. Because TOSSIM 1.x simulates the Mica2 motes and the CC1000 radio, we set $T_{pkt} = 0.020$ s according to Section 5.2.

Table 3 shows the dissemination time (for `Blink`) in TOSSIM and the analysis time for 8 network topologies. We can see that in all the cases, our analytical results fit fairly well with the simulation results, with approximately 25 s maximum errors.

5.3.3. A detailed look

To get a closer look at how accurately our approach models the dissemination process, we also plot time dynamics for the first and last page for two typical network topologies, i.e., 10×10–10, and 10×10–15, respectively.

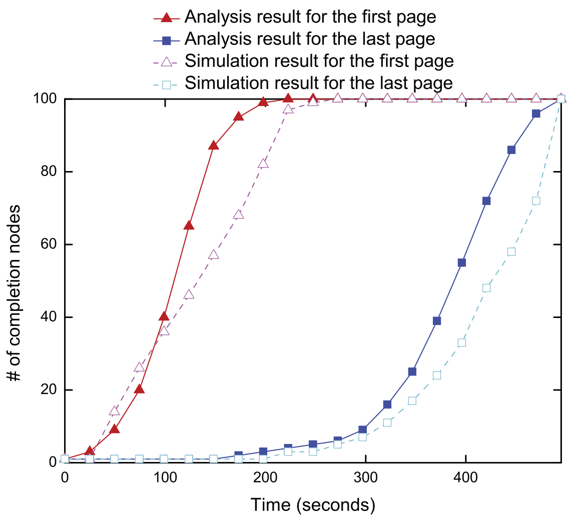

Fig. 6. Time dynamics in $10 \times 10 - 10$.

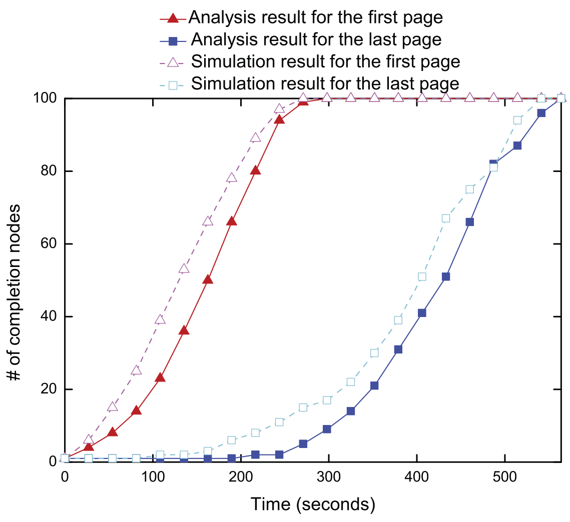

Fig. 7. Time dynamics in $10 \times 10 - 15$.

Figs. 6 and 7 show the results. We can see that the start time of the last page is very late in time. This is because all the transmissions of the previous pages in the downstream links will postpone the current transmission. Our approach can accurately capture this effect as explained in Section 4.2.

5.3.4. Comparisons

For linear structures, we compare our approach with Deluge's approach as described in [1]. We disseminate the `Blink` application in 10×2 linear structures with 5, 10, 15, and 20 ft spacing respectively. Fig. 8 plots the completion times for our approach, Deluge's approach, and simulation results. We can see that our approach fits better than Deluge's approach, especially in the sparse case. This is because of two reasons. First, the number of request messages (N_{req}) is fixed in Deluge's model. This is not true when packet loss rate increases. Second, Deluge's approach does not consider pair-wise link qualities. This is inaccurate because a poor link will probably dominate the propagation delay along the path.

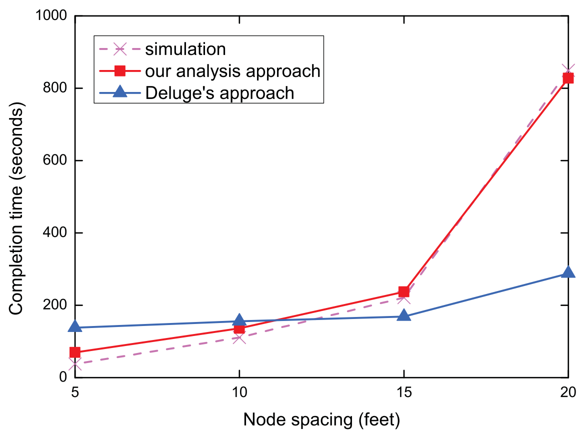

Fig. 8. Overall completion time comparisons in a 10×2 linear structure with different node spacings.

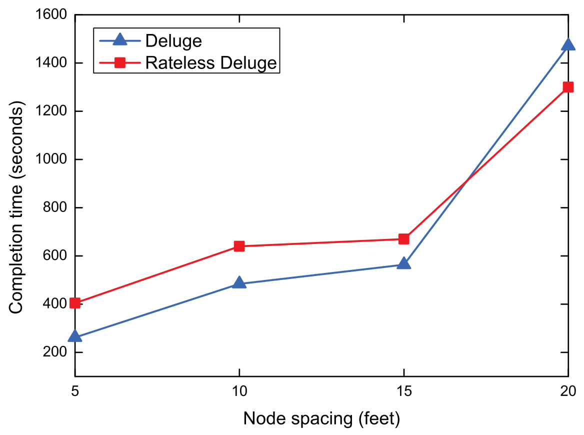

Fig. 9. Deluge vs. Rateless Deluge.

5.4. Coding vs. non-coding

In this subsection, we use our analytical model to get some insights into the performance of Deluge [1] (non-coding protocols) and Rateless Deluge [6] (coding-based protocols) in multihop sensor networks. Rateless Deluge has not been tested under sufficiently large multihop networks due to testbed limitations [6].

Without loss of generality, we set the maximum transmission range ($d_{\max}$) as 35 ft as a distance larger than 35 ft will lead to very poor link qualities. Fig. 9 shows the overall completion time for Deluge and Rateless Deluge. Our analytical results show that the Rateless Deluge performs better than Deluge only in sparse networks (e.g., with 20 ft spacing). The reason why it performs even worse than Deluge in dense networks can be explained by Eq. (14). That is, the data transmission time is dominated by the worst link in the neighborhood in order to serve nodes that are far away. Besides, the coding overhead is an additional cost

Table 4
Computation time comparisons (s).

Topology	Dissem. time	Sim. time	Analysis time
$2 \times 2 - 5$	24	5	0.001
$5 \times 5 - 5$	46	79	0.016
$8 \times 8 - 5$	102	569	0.14
$10 \times 10 - 5$	266	3188	0.30
$15 \times 15 - 5$	2098	21612	1.05

for coding-based protocols. Our result is compliant with earlier observations pointed in [1] that FEC improved performance in sparse networks while harming performance in dense networks. The decreased performance in dense networks is due to the existence of highly variable link qualities where nearby neighbors had high link qualities and nodes further away had poor link qualities. Coding-based protocols work best in environments where loss rates are predictable and have low variance, allowing the amount of redundant data transmitted to be tuned to match the link qualities of the network [1].

5.5. Computation time

In order to calculate the completion times of N nodes, we need to run the Dijkstra algorithm to determine the propagation path from the source to each of the other nodes ($O(N^2)$); then we compute the completion times along the path for all the pages ($O(NP)$); we record the maximum completion times for a given node during this process. Thus, the overall complexity of our approach is,

$$N \cdot O(N^2) \cdot O(NP) = O(N^4P) \tag{23}$$

We implement our approach on a typical PC with Intel 2.8 GHz CPU, 1 GB DDRII RAM. The program is about 500 lines of code in Perl. We compare the computation time of our approach with that of TOSSIM simulation time and actual completion time for dissemination. Table 4 shows the results. We can see that our analytical approach is much faster than simulation, especially when the network scales. For example, for the 15×15–5 case, the simulation time is nearly 6 h while our approach only costs a few more than one second. This indicates that our approach can be applied to large-scale networks with very low time cost.

5.6. Performance optimizations

There are several factors affecting the overall completion time of the Deluge protocol, e.g., the page size. Deluge requires all nodes keep their radio on during dissemination in order to allow pipelining. Therefore, optimizing the overall completion time will greatly reduce energy consumption as radio component consumes the largest fraction of energy on sensor nodes [24]. We notice that there is a tradeoff in determining the page size for dissemination. A large page size (which means small number of pages for a fixed-sized image) will limit pipelining for spatial multiplexing while a small page size (which means large number of pages for a fixed-size image) will add the per page negotiation overhead. With our analytical approach, we can easily evaluate the completion times with different page size. We illustrate the benefits of our approach through a 48×2–10 linear structure, disseminating the `Blink` application with 12 pkts/page, 24 pkts/page, and 48 pkts/page respectively. Fig. 10 shows the completion times under different page sizes. Our analytical approach finds that the segmentation scheme with 24 pkts/page is optimal compared to the other two. The simulation results validate our analytical results, indicating that 24 pkts/page will reduce the completion time by 16.6% compared to the 48 pkts/page, the default page size in Deluge.

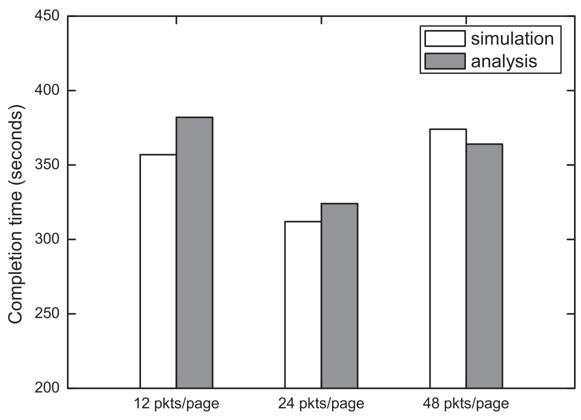

Fig. 10. Shortening the completion time by page size tuning.

6. Related work

WSNs have recently gained a great deal of attention as a topic of research, with a wide range of systems [8,25,26] and applications [27,28] being explored. Bulk data dissemination is a basic building block for sensor network applications [4,29].

Deluge [1] is perhaps the most popular reprogramming protocol used for reliable code updates in WSNs. It uses a three-way handshake and NACK-based protocol for reliability, and employs segmentation (into pages) and pipelining for spatial multiplexing. It is highly optimized and can achieve one ninth the maximum transmission rate of the radio supported under TinyOS. It was also mentioned by the author of Deluge that the use of FEC improves performance in sparse networks while harming performance in dense networks due to the existence of highly variable link qualities [1].

Stream [5] reduces the size of data actually disseminated by pre-installing the reprogramming protocol before deployments. Stream is insufficient for complex applications which usually include a large number of kernel components.

Elon [30] addresses this issue by introducing the concept of replaceable component. In the network reprogramming process, only the replaceable component needs to be disseminated, greatly reducing the dissemination cost.

MNP [2] provides a detailed sender selection algorithm to choose a local source of the code which can satisfy the maximum number of nodes.

Recently, several coding-based reprogramming protocols specifically designed for WSNs are proposed to address the deficiency of Deluge in sparse and lossy networks, including Rateless Deluge [6], SYNAPSE [31], and AdapCode [32]. They all use network coding to encode a packet before transmission. After receiving an expected number of encoded packets, the receiving node uses Gaussian elimination to decode the packets. The difference is that Rateless Deluge [6] uses Random Linear Codes; SYNAPSE [31] uses Fountain Codes; AdapCode [32] also uses linear codes, but the coding scheme is adaptively changed according to the link quality.

Generally, designers carry out simulations or testbed experiments to evaluate the performance these protocols. Various tools have been proposed in the literature to aid system design and evaluations [33,34,20]. However, testbed experiments requires at least a prototype system to be deployed and compared to prototyping, conducting offline analysis is inexpensive in terms of time and resources. Generally, simulators allow the designers to tune many different parameters and provide a fairly good resemblance of the real environment and compared to detailed simulation, theoretical analysis provides an alternate method of designing systems with lower cost.

Although theoretical analysis of WSNs is a relatively new area, there is a growing interest and new types of analysis are continuously developed [35–37]. With respect to bulk data dissemination, there are a few preliminarily work on performance modeling and analysis. In [1], the authors only analyze the performance of Deluge in linear structures. In [5], the authors analyze the performance of code dissemination without consideration for protocol details, such as advertisement and request overheads. In [7], the authors propose a framework based on epidemic theory for vulnerability analysis of broadcast protocols in WSNs. They did not consider topological information and the impact of pipelining, thus the result is not accurate.

Compared to [1], our work is more general and can be applied to practical network topologies. Compared to [5,7], our work is much more accurate by considering topological information, impact of contention, and impact of pipelining. We validate the analytical results with testbed and extensive simulations via TOSSIM. We believe that the attractive benefits of theoretical analysis towards system design will foster the development of a much needed theoretical foundation in the field of WSNs.

7. Conclusions

In this work, we show a way of accurately analyzing the performance of bulk data dissemination protocols in WSNs. Our model can be applied to practical networks by use of the shortest propagation path. Our model is accurate by considering topological information, impact of contention, and impact of pipelining. We validate the analytical results through testbeds and detailed simulations. Results show that the analytical results fit fairly well with the testbed results and simulation results. Further, we demonstrate that the analytical results can be used to aid protocol design for performance optimizations, e.g., page size tuning for shortening the completion time.

Acknowledgement

This work is supported by the Fundamental Research Funds for the Central Universities (2012QNA5007), the National Science Foundation of China (Grant No. 61070155), the Program for New Century Excellent Talents in University (NCET-09-0685), and in part by NSERC Discovery Grant 341823-07, FQRNT grant 2010-NC-131844, and CFI Leaders Opportunity Fund 23090.

References

[1] J.W. Hui, D. Culler, The dynamic behavior of a data dissemination protocol for network programming at scale, in: Proceedings of ACM SenSys, 2004.
[2] S.S. Kulkarni, L. Wang, MNP: Multihop Network reprogramming service for sensor networks, in: Proceedings of IEEE ICDCS, 2005.
[3] S. Shin, T. Kwon, G.-Y. Jo, Y. Park, H. Rhy, An experimental study of hierarchical intrusion detection for wireless industrial sensor networks, IEEE Transactions on Industrial Informatics.
[4] Q. Wang, Y. Zhu, L. Cheng, Reprogramming wireless sensor networks: challenges and approaches, IEEE Network Magazine 20 (3) (2006) 48–55.
[5] R.K. Panta, I. Khalil, S. Bagchi, Stream: low overhead wireless reprogramming for sensor networks, in: Proceedings of IEEE INFOCOM, 2007.
[6] A. Hagedorn, D. Starobinski, A. Trachtenberg, Rateless deluge: over-the-air programming of wireless sensor networks using random linear codes, in: Proceedings of ACM/IEEE IPSN, 2008.
[7] P. De, Y. Liu, S.K. Das, An epidemic theoretic framework for evaluating broadcast protocols in wireless sensor networks, in: Proceedings of IEEE MASS, 2007.
[8] TinyOS, <http://www.tinyos.net>.
[9] L. Huang, S. Setia, CORD: energy–efficient reliable bulk data dissemination in sensor networks, in: Proceedings of IEEE INFOCOM, 2008.
[10] W. Dong, C. Chen, X. Liu, J. Bu, Y. Liu, Performance of bulk data dissemination in wireless sensor networks, in: Proceedings of IEEE/ACM DCOSS, 2009.
[11] A. Kashyap, S. Ganguly, S.R. Das, A measurement-based approach to modeling link capacity in 802.11-based wireless networks, in: Proceedings of ACM MobiCom, 2007.
[12] K. Srinivasan, P. Levis, RSSI is under appreciated, in: Proceedings of EmNets, 2006.
[13] K. Srinivasan, M. Kazandijeva, S. Agarwal, P. Levis, The β-factor: measuring wireless link burstiness, in: Proceedings of ACM SenSys, 2008.
[14] T.T. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms (1990).
[15] L. Qiu, Y. Zhang, F. Wang, M.K. Han, R. Mahajan, A general model of wireless interference, in: Proceedings of ACM MobiCom, 2007.
[16] A. Kashyap, S. Das, S. Ganguly, A measurement-based approach to modeling link capacity in 802.11-based wireless networks, in: Proceedings of ACM MobiCom, 2007.
[17] R. Maheshwari, S. Jain, S.R. Das, A measurement study of interference modeling and scheduling in low-power wireless networks, in: Proceedings of ACM SenSys, 2008.
[18] S. Liu, G. Xing, H. Zhang, J. Wang, J. Huang, M. Sha, Passive interference measurement in wireless sensor networks, in: Proceedings of IEEE ICNP, 2010.
[19] X. Cheng, M. Ding, D. Du, X. Jia, Virtual backbone construction in multihop ad hoc wireless networks, Wireless Communications and Mobile Computing 6 (2006) 183–190.
[20] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, in: Proceedings of ACM SenSys, 2003.
[21] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, J. Zahorjan, Measurement-based models of delivery and interference in static wireless networks, in: Proceedings of ACM SIGCOMM, 2006.
[22] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proceedings of ACM SenSys, 2009.
[23] TinyOS TEP133, <http://www.tinyos.net>.
[24] J. Polastre, R. Szewczyk, D. Culler, Telos: enabling ultra-low power wireless research, in: Proceedings of ACM/IEEE IPSN/SPOTS, 2005.

[25] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R. Han, MANTIS OS: an embedded multithreaded operating system for wireless micro sensor platforms, ACM/Kluwer Mobile Networks and Applications Journal (MONET), Special Issue on Wireless Sensor Networks 10 (2005) 563–579.
[26] W. Dong, C. Chen, X. Liu, K. Zheng, R. Chu, J. Bu, FIT: a flexible, light-weight, and real-time scheduling system for wireless sensor platforms, IEEE Transactions on Parallel and Distributed Systems 21 (1) (2010) 126–138.
[27] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, W. Hong, A macroscope in the redwoods, in: Proceedings of ACM SenSys, 2005.
[28] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J.A. Stankovic, T.F. Abdelzaher, J. Hui, B. Krogh, VigilNet: an integrated sensor network system for energy–efficient surveillance, ACM Transactions on Sensor Networks (TOSN) 2 (1) (2006) 1–38.
[29] S. Yi, H. Min, Y. Cho, J. Hong, Adaptive multilevel code update protocol for real-time sensor operating systems, IEEE Transactions on Industrial Informatics 4 (4) (2008) 250–260.
[30] W. Dong, Y. Liu, X. Wu, L. Gu, C. Chen, Elon: enabling efficient and long-term reprogramming for wireless sensor networks, in: Proceedings of ACM SIGMETRICS, 2010.
[31] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A.F.H. III, M. Zorzi, SYNAPSE: a network reprogramming protocol for wireless sensor networks using fountain codes, in: Proceedings of IEEE SECON, 2008.
[32] I.-H. Hou, Y.-E. Tsai, T.F. Abdelzaher, I. Gupta, AdapCode: adaptive network coding for code updates in wireless sensor networks, in: Proceedings of IEEE INFOCOM, 2008.
[33] G. Werner-Allen, P. Swieskowski, M. Welsh, MoteLab: a wireless sensor network testbed, in: Proceedings of ACM/IEEE IPSN/SPOTS, 2005.
[34] B.L. Titzer, D.K. Lee, J. Palsberg, Avrora: scalable sensor network simulation with precise timing, in: Proceedings of ACM/IEEE IPSN, 2005.
[35] V. Prasad, T. Yan, P. Jayachandran, Z. Li, S.H. Son, J.A. Stankovic, J. Hansson, T. Abdelzaher, ANDES: an ANalysis-based DEsign tool for wireless Sensor networks, in: Proceedings of IEEE RTSS, 2007.
[36] Q. Cao, T. Yan, J.A. Stankovic, T.F. Abdelzaher, Analysis of target detection performance for wireless sensor networks, in: Proceedings of IEEE/ACM DCOSS, 2005.
[37] J. Ahn, B. Krishnamachari, Performance of a propagation delay tolerant ALOHA protocol for underwater wireless networks, in: Proceedings of IEEE/ACM DCOSS, 2008.

Wei Dong received the BS and Ph.D. degrees in Computer Science from Zhejiang University in 2005 and 2010, respectively. He was a Postdoc Fellow at the Department of Computer Science and Engineering in Hong Kong University of Science and Technology from Dec. 2010 to Dec. 2011. He is currently an assistant professor at the College of Computer Science in Zhejiang University. His research interests include networked embedded systems and wireless sensor networks.

Chun Chen received his Bachelor of Mathematics degree from Xiamen University, China, in 1981, and his Masters and Ph.D. degrees in Computer Science from Zhejiang University, China, in 1984 and 1990 respectively. He is a professor in College of Computer Science, the Dean of College of Software, and the Director of Institute of Computer Software at Zhejiang University. His research activity is in image processing, computer vision, CAD/CAM, CSCW and embedded system.

Xue Liu is an associate professor in the School of Computer Science at McGill University. He received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in 2006. He received his B.S. degree in Mathematics and M.S. degree in Automatic Control both from Tsinghua University, China. He has also worked as the Samuel R. Thompson Associate Professor in the University of Nebraska-Lincoln and HP Labs in Palo Alto, California His research interests are in computer networks and communications, smart grid, real-time and embedded systems, cyber-physical systems, data centers, and software reliability. Dr. Liu has been granted 1 US patent and filed 4 other US patents, and published more than 150 research papers in major peer-reviewed international journals and conference proceedings, including the Year 2008 Best Paper Award from IEEE Transactions on Industrial Informatics, and the First Place Best Paper Award of the ACM Conference on Wireless Network Security (WiSec 2011).

Guodong Teng received his Ph.D. degree in Computer Science from Zhejiang University in 2011. He received his B.S. degree from Hangzhou Normal University in 2000 and his M.S. degree from Hangzhou DianZi University in 2005. He is a lecturer in the Institute of Service Engineering at Hangzhou Normal University. His research interests include networked embedded systems and wireless sensor networks.

Jiajun Bu received the B.S and Ph.D. degrees in Computer Science from Zhejiang University, China, in 1995 and 2000, respectively. He is a professor in College of Computer Science and the deputy dean of the Department of Digital Media and Network Technology at Zhejiang University. His research interests include embedded system, mobile multimedia, and data mining. He is a member of the IEEE and the ACM.

Yunhao Liu received his BS degree in Automation Department from Tsinghua University, China, in 1995, and an MA degree in Beijing Foreign Studies University, China, in 1997, and an MS and a Ph.D. degree in Computer Science and Engineering at Michigan State University in 2003 and 2004, respectively. He is a member of Tsinghua National Lab for Information Science and Technology, and the Director of Tsinghua National MOE Key Lab for Information Security. He is also a faculty at the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology. Being a senior member of IEEE, he is also the ACM Distinguished Speaker.