ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 2, February 2013

Fast distribution of Data in Wireless Sensor Network using Concurrency Operation

Geetha N, Janani V

***Abstract*— Wireless Sensor Network can be applied in verity of applications in real time. Efficient data dissemination enables parameter reconfiguration, network reprogramming, security holes patching, software bug fixing and etc. The current dissemination protocols use network coding techniques to face the packet losses. The coding overhead is the bottleneck in dissemination delay. We propose MT-Deluge, a multi-threaded design of a coding-based data dissemination protocol which separates the coding and radio operations in to two threads and carefully scheduling their executions, MT-Deluge shortens the dissemination delay effectively. To improve the performance of this protocol an incremental decoding algorithm is employed. MT-Deluge shortens the dissemination delay by 25.5-48.6% when compared to a typical data dissemination protocol.**

***Index Terms*—Dissemination, Network protocol, Reprogramming, wireless sensor Network.**

I. INTRODUCTION

Wireless sensor network have been widely used to perceive and interact with the physical world for different purposes. After the network is deployed, the network operators often need to disseminate data to a network of sensors. Therefore, data dissemination is an essential building block of WSN applications. There are two approaches in the existing protocols namely non coding–based approaches and coding–based approaches. The performance of non-coding-based [1], [2] degrades in lossy networks with unreliable wireless links. When the link qualities become then the number of retransmissions grows which further makes the data packets to be collided. Coding based approaches [3]-[5] employ network coding technique to encode the native packets in to encoded packets at the sender side. The sender does not retransmit all lost packets of its receivers, but encodes some new encoded packets and retransmits them.

When a sensor node is sending or receiving packets, the radio is busy but the CPU utilization is low and when the sensor is encoding or decoding packets the CPU utilization is high but the radio is idle. This inspired to exploit the concurrency potential of sensor nodes.

MT-Deluge, a novel multi-threaded design exploits the concurrency potential of sensor nodes by separating the coding operation and radio operation into different threads. Multithreaded design are able to amortize the overheads of encoding and decoding into the idle-wait durations during the radio operations, so as to reduce the dissemination delay of coding-based approaches.

Integrating multi-threading with current coding-based approaches has several practical challenges. First, when a node receives a sufficient number of packets, how to recover the native packets as soon as possible? Second, when multi-threading is utilized, how to synchronize multiple threads correctly and efficiently? To address the first challenge, an incremental decoding algorithm is proposed. Instead of starting the decoding procedure after receiving all packets, the receiver starts to decode after obtaining a small number of packets. The incremental decoding algorithm can make use of the idle-durations in receiving, mitigating the impact of decoding to dissemination delay. To address the second challenge, adaptive packet-level synchronization is introduced to synchronize multiple threads correctly. Besides, when the receiver is receiving data packets from the sender, the start time of the decoding procedure is optimized to get the best performance.

II. RELATED WORK

Existing approaches can be divided into two categories, i.e., non-coding-based and coding based approaches. Deluge [1] is the de facto standard non coding-based dissemination protocol. In Deluge, every sensor node advertises the version number of its current program, then nodes with older version programs request program updates. Then these receivers receive the packets from the sender and use an NACK-based mechanism to improve the reliability. The program image is divided into fixed size pages which are further divided into native packets to send. When a sensor node has received an entire page, it forwards this page to other nodes before receiving the next page. This mechanism enables spatial multiplexing which reduces the total delay significantly. Rateless Deluge [4] is a

coding-based data dissemination protocol. It uses random linear coding reduce the retransmission overhead and two additional optimizations (i.e., precoding and ACKless transmission) further improve the performance. It is reported in [4] that Rateless Deluge can get 15-30% savings in the data plane and 50-80% in the control plane than original Deluge. Different from these previous works, MT-Deluge exploits the concurrency potential of sensor nodes and is able to shorten the dissemination delay effectively.

III. IMPROVEMENT FOR SINGLE-HOP NETWORKS

In multi-hop networks, the sink node (i.e., the one starts dissemination) encodes all the packets, and this encoding is not time consuming compared to the long dissemination delay. That is the reason why we do not make the encoding operation and sending operation run concurrently at the sink node. In a single-hop network, however, the total dissemination delay is relatively short. Therefore we modify MT-Deluge for single-hop network that the sink node encodes and sends packets concurrently. The workflow of a single-hop network is shown in Figure 1. The sender starts sending the encoded packets earlier before the improvement. The time saving is not significant but it is still worth doing this since the total dissemination delay is relatively short in a single-hop network.

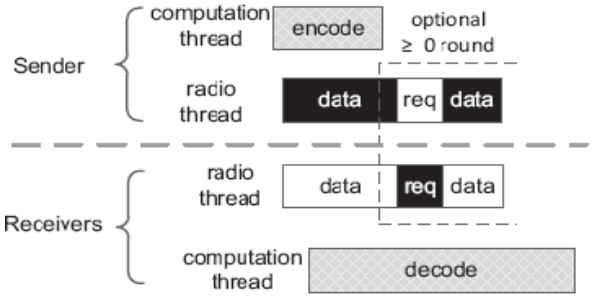


Fig. 1: Improvement for a single-hop network.

IV. MULTI-THREADED DESIGN OVER-VIEW

MT-Deluge is proposed to improve the efficiency of coding-based dissemination protocols. The aim of multi-threaded design is to allow the coding operations and the radio operations to work concurrently. In the multi-threaded design, a sensor node can decode and send packets simultaneously, so there is no separate TX state or DEC state. In Figure 2, the black box represents the sending operation, the white box represents the receiving operation and the grey box represents coding operations (i.e., encoding and decoding). There are three nodes. Node A sends one page to node B and node B sends the same page to node C. In the single-threaded design a node only starts to send the page to next node after decoding and encoding. In the multithreaded design as shown in Figure 2, a radio thread and a computation thread are executed concurrently. Node B can forward the page to node C once it has collected a small number of encoded packets sent by node A. The computation thread can start to decode even earlier than the sending thread. Once node B has obtained several packets, the computation thread can start to decode. Once node B has obtained enough encoded packets, the decoding procedure ends after a short period of time. Therefore, the page dissemination time can be shortened.

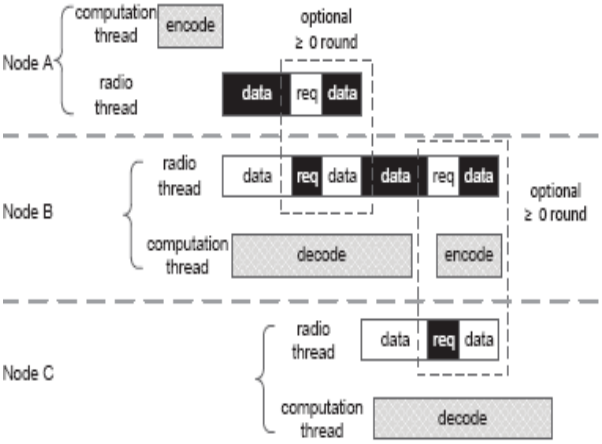


Fig. 2: Workflow of multi-threaded design.

V. MULTITHREADED DESIGN AND IMPLEMENTATION

The implementation of multi-threaded design has two main challenges. One is to shorten the total dissemination delay by minimizing the waiting time throughout the dissemination procedure. The other is to synchronize different threads running on each sensor node correctly and efficiently. To address the first challenge, we propose an incremental decoding algorithm to start the decoding procedure earlier. To synchronize multiple threads correctly, a packet-level synchronization mechanism is employed. In addition, an adaptive page size adjustment scheme is utilized to optimize the page size given the link quality. MT-Deluge changes the page transfer mechanism from single-threaded design to multi-threaded design. In this section, we describe the solutions and the implementation details to address the challenges.

I. Incremental Decoding Procedure

In Rateless Deluge, a page is decoded after the receiver has received all encoded packets. After decoding, the native packets are obtained. Then the receiver encodes the native packets and sends them to other nodes. In MT-Deluge, the multi-threaded design allows the computation thread and radio thread to run concurrently. Therefore, an obvious benefit is that the receiver can just forward the received encoded packet to other nodes and decode the encoded packets at the same time. In MT-Deluge, the whole incremental decoding is divided into four procedures. Procedure one is the outer most procedure which calls other procedures; the procedure two performs standard Gaussian elimination; the procedure three performs one step of incremental decoding; the procedure four performs back substitution. The algorithm starts when I initial encoded packets have been obtained at the receiver side. Then the seeds attached in these packets are used to generate the coefficients of the corresponding rows (the coefficients of each encoded packet forms one row). Standard Gaussian elimination further calculates the echelon matrix of these initial rows. When the computation thread is running, the radio thread can receive new encoded packets at the same time. The corresponding rows of coefficients are generated by the seeds attached in these data packets. Then the newly received packets are inserted into the matrix and the updated matrix is calculated to be an echelon matrix When enough encoded packets have been received, back substitution is used to obtain the native packets of the page. In incremental method, the I initial rows are calculated to the echelon form by standard Gaussian Elimination and the m new rows are attached and calculated to echelon form by one step incremental decoding. Compared with the original Gaussian Elimination, this incremental algorithm changes the order of elimination, but does not change the number of subtractions. Therefore, the complexity of incremental decoding algorithm is the same as original Gaussian elimination which is O(N3) given an N × N matrix.

II. Multi-thread Synchronization

We use an adaptive packet-level synchronization mechanism to provide correct and efficient synchronization between multiple threads. The mechanism has three key features. First, packet-level synchronization between the radio thread and the computation thread provides high precision. This feature ensures the correctness. Second, the actual workflow of each sensor node is adaptively chosen at the run-time. Third, different page sizes are chosen according to different link qualities in order to shorten the total delay. These two features improve the efficiency. Compared with packet-level synchronization, statelevel synchronization is another choice in the implementation. In fact, we use state-level synchronization to synchronize radio thread and computation thread outside the incremental decoding phase. If incremental decoding is not performed, state-level synchronization will be sufficient. For example, when the sender enters state TX from state ENC, it can use synchronization primitives (e.g., a semaphore). However, when incremental decoding is used, statelevel synchronization is not sufficient because it does not have any separate states anymore. For example, when the node is receiving packets, it is probably doing decoding at the same time. It is hard to say whether the node is in RX state or DEC state. The workflow and some related code of incremental decoding using packet-level synchronization are depicted in Figure 3.

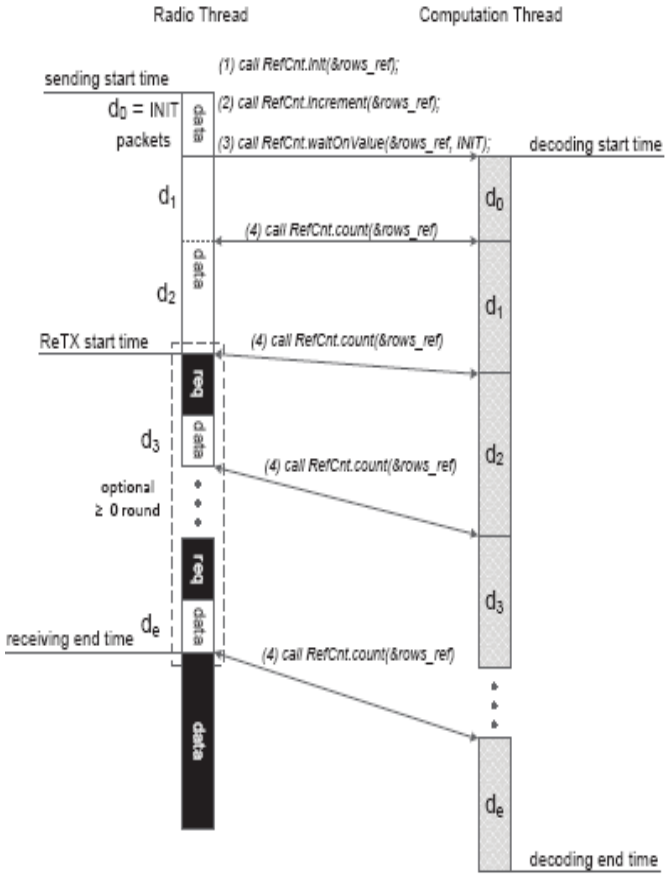


Fig. 3: Packet level synchronization

As same as previous figures, the black box represents the sending operation, the white box represents the receiving operation and the grey box represents coding

operations (i.e., encoding and decoding). In the current implementation, we use a synchronization primitive in named ReferenceCounter (RefCnt) to record the received number of packets. There are four operations regarding RefCnt. Operation (1) initializes the value of RefCnt to zero. Operation (2) increases the value of RefCnt when a new encoded packet is received. Operation (3) waits till INIT packets have been received. Operation (4) is used to get the current number of received packets. d1 to de represent all the encoded packets received and each of them represents a segment of the page. The INIT value indicates the number of received packets before the decoding procedure starts (INIT [1, k], k is the page size). Smaller INIT can make the start time of decoding earlier. However, setting the INIT too small will introduce too much synchronization overhead. As described in figure, the waitOnValue method will be called many times if the INIT is set to be too small. In this implementation, we find out that setting INIT to k/3 can get good performance d0 (equals to INIT) is set before the program runs, but d2 to de are chosen at run-time. Operation (4) is used to get the current number of received packets and these newly received packets are incrementally decoded subsequently. This adaptive mechanism to determine the number of the next packet segment can avoid wait time of both the radio and computation threads. In addition, link quality can affect the packet receiving progress. Retransmission is needed when link quality is poor, which makes the packet receiving progress much slower. Link quality can only be obtained after the program runs. Therefore, the adaptive mechanism can also work well under different link qualities.

III. Page Size Adjustment

The last feature is that we set different *k* (i.e., page size) under different link qualities. Large *k* means long decoding time but small number of pages. In original Deluge, *k* is 48 and in Rateless Deluge *k* is 20. The reason is that the decoding time of a page consisting of 48 packets is nearly 7 seconds which significantly impacts the dissemination delay. However, large page size (e.g., 48 in original Deluge) can be beneficial without considering the decoding cost. By multi-threading, the decoding cost is amortized into the packet sending and receiving times. Thus in MT-Deluge, a large page size *k* is beneficial. Figure 4 shows the precise one page transmission time line on one sensor node. Note that *Ttx* equals to *Trx*. *Tretx* consists of several *Treq* (i.e., requesting time) and *Trrx* (i.e., re-receiving time). *Tidec* is the incremental decoding time. In Figure 4, there are two arrows which represent two constraints. Constraint (a) means that the decoding procedure starts only after *INIT* packets have been received. Constraint (b) means that the data retransmission starts only after the decoding and encoding procedures completes. The reason is that the encoded packets have already been received by the receivers with high probability and only newly encoded packets can be used in retransmission.

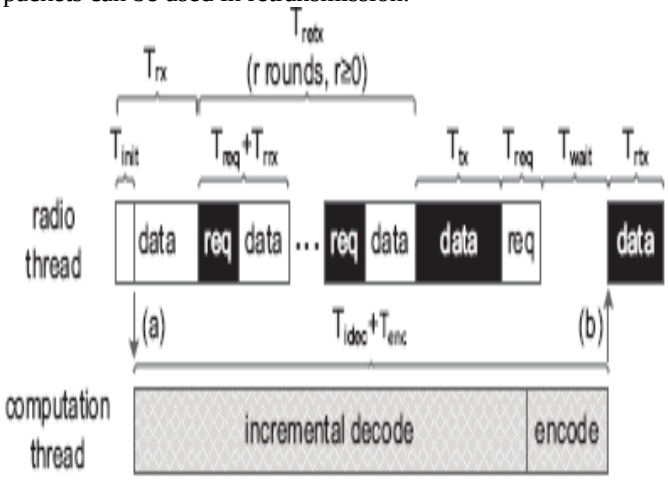
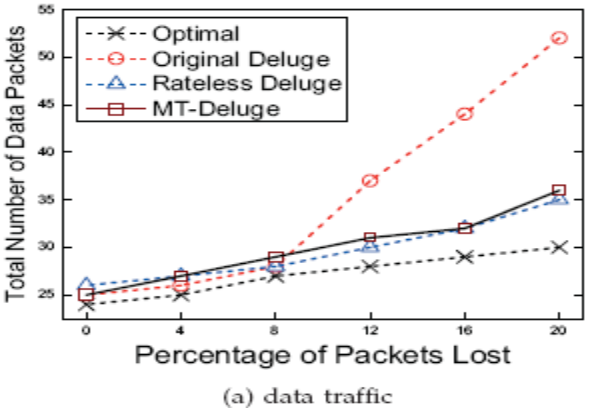
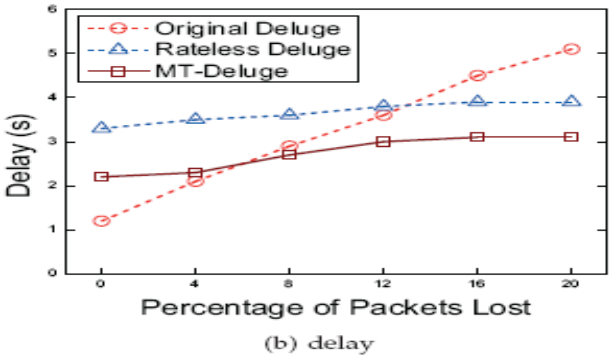


Fig. 4: One page transmission time line.

VI. COMPARISON OF DELAY PERCENTAGE



(a) data traffic



(b) delay

Fig. 5: Data traffic and delay

Figure 5(a) shows that MT-Deluge transmits as similar number of packets as Rateless Deluge. Figure 5(b) shows that MT-Deluge has a shorter delay than Rateless Deluge in the single-hop network. On average, MT-Deluge costs 25.5% less time than Rateless Deluge in a single-hop network when the page size is 24.

VI. CONCLUSION AND FUTURE WORK

In this paper, we provide a multi-threaded design for data dissemination in WSNs. Current coding-based data dissemination protocols introduce encoding or decoding overhead and this problem limits the scalability of these protocols. We propose MT-Deluge, a multithreaded design which allows the radio operations and the computations run concurrently. We implement it and the graph is shown for comparison of data traffic and delay of Deluge, Rateless Deluge and MT-Deluge. The graph shows that the MT-Deluge has much shorter dissemination delay. For future work, we will consider designing multi-threaded versions of other protocols and leveraging power management to further improve the network performance for WSNs.

ACKNOWLEDGMENT

Our thanks to the experts who have contributed for the development of concurrent exploitation for fast data distribution in wireless sensor network and its implemented solution.

VII. REFERENCES

[1] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *Proc. of ACM SenSys*, 2004.

[2] R. K. Panta, I. Khalil, and S. Bagchi, "Stream: Low Overhead Wireless Reprogramming for Sensor Networks," in *Proc. of IEEE INFOCOM*, 2007.

[3] M. Rossi, G. Zanca, L. Stabellini, R. Crepaldi, A. F. H. III, and M. Zorzi, "SYNAPSE: A Network Reprogramming Protocol for Wireless Sensor Networks using Fountain Codes," in *Proc. of IEEE SECON*, 2008.

[4] A. Hagedorn, D. Starobinski, and A. Trachtenberg, "Rateless Deluge: Over-the-Air Programming of Wireless Sensor Networks using Random Linear Codes," in *Proc. of ACM/IEEE IPSN*, 2008.

[5] W. Dong, C. Chen, X. Liu, J. Bu, and Y. Gao, "A Light-Weight and Density-Aware Reprogramming Protocol for Wireless Sensor Networks," IEEE Transactions on Mobile Computing, vol. 99, no. PrePrints, 2010.

[6] Y. Gao, J. Bu, W. Dong, C. Chen, L. Rao, and X. Liu, "Exploiting concurrency for efficient dissemination in wireless sensor networks," in Proc. of IEEE DCOSS, 2011.

[7] 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. of IEEE INFOCOM, 2008

N.Geetha received her B.E degree in Computer Science Engineering from Anna University Coimbatore in 2011 and currently doing her M.E degree in Adhiyamaan College of Engineering. She can be reached at tnidharshini@gmail.com

V.Janani received her B.E degree in Computer Science Engineering from Periyar University in 2003 and M.E degree in Computer Science Engineering from Anna University Chennai in 2008 and currently working as Assistant Professor in Adhiyamaan College of Engineering She can be reached at vajjiram.janani@gmail.com