國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

實作 XD: 致力於高效率之無線感測網路資料傳輸機制

Implementation of XD: A Data Collection Protocol Targeting at Mission-Critical WSN Applications

王宇廷

Yu-Ting Wang

指導教授：黃寶儀 博士

Advisor: Polly Huang, Ph.D.

中華民國 102 年 7 月

July, 2013

國立臺灣大學（碩）博士學位論文
口試委員會審定書

實作 XD: 致力於高效率之無線感測網路資料傳輸機制

Implementation of XD: A Data Collection Protocol Targeting at Mission-Critical WSN Applications

本論文係王宇廷君（R00942093）在國立臺灣大學電信工程學研究所完成之碩（博）士學位論文，於民國 102 年 7 月 2 日承下列考試委員審查通過及口試及格，特此證明。

口試委員：[illegible]（簽名）

[illegible]（指導教授）　　[illegible]

[illegible]

所　　長　吳宗霖（簽名）

誌謝

這篇論文的完成，不僅是 XD 這個題目的結束，它所代表的，更是我在研究和寫程式能力上的成熟。感謝黃寶儀老師在這兩年中的苦心教導，讓我們獲得的不只是研究成果，還有獨立研究和批判思考的能力，以及面對事情應有的態度。除此之外，我特別感謝老師支持我在研究最繁重的碩二上，將一門很重的程式課修得很好，大大的補足了我的背景。感謝陳伶志老師在我升碩二暑假時指導 XD 的研究，也讓我在簡報表達上更加進步。感謝 SY 學長，從我加入實驗室以來，不知蒙受他多少在嵌入式系統上的教導及經驗傳承，總是在我遇到瓶頸的時候，傾聽我的問題並予以適當的協助。感謝我的合作伙伴孟林，在研究上的互相討論以及他架設 server 的能力，著實讓我獲益良多，使得 XD 的研究順利不少。也感謝在研究 XD 的過程中，一起討論過的 Yetta、Junction、依仙、揚鈞、淙勻。特別感謝淙勻在我碩一程式能力不好時，耐心親切的給了我兩次關鍵的幫助，讓我相當感動。

在這兩年的日子裡，感謝有同居的孟林、淙勻、揚鈞作伴一起奮鬥，讓我研究的日子不孤單。還有我的家人，在我徬徨的時候永遠支持並相信我，是我繼續前進的最大力量，謝謝你們！

摘要

這篇論文著眼於實作出無線感測網路中的一種資料傳輸機制—Cross-layer Diffusion (XD)，其設計理念為及時且可靠的傳送資料，以用於重要任務如老人護理中心的定位追蹤系統。XD 結合了 MD 以及混和 CSMA 和 TDMA 的機制。MD 是一種隨機重覆傳送資料的路由協定,其建立的拓撲資訊亦同時被 XD 中的 TDMA 所使用。以前的模擬結果宣稱 XD 比 MD 效能更好。而就我們所知，這篇論文是第一個將 XD 實做出來的研究。但結果顯示，不管我們使用和以前模擬一樣的設定或實際應用的環境像是定位系統,在絕大部份的情況下 MD 的效能都比 XD 好。這是由於在 XD 的模擬中使用了不切實際的小 TDMA 時槽寬度，導致預期的 XD 網路傳輸容量比實際來得大，高估了 XD 的效能。

關鍵字：無線感測網路、資料收集、資料配送、跨層設計、時間同步、實作

Abstract

This paper focuses on implementing a data dissemination mechanism of wireless sensor network (WSN) - Cross-layer Diffusion (XD), which aims at providing timely and reliable data transmission for mission critical applications such as location tracking system in elderly care centers. XD is a combination of MD, an opportunistic redundant data transmission routing protocol, and hybrid CSMA and TDMA which utilizes topology information established by MD. Prior work claims that XD is better than MD based on the results of simulation, and this paper is the first that XD is successfully implemented. Based on the evaluation of XD and MD on BL-live testbed with settings in simulation and in real applications such as location system, we conclude that MD is better than XD in general due to impracticably small TDMA slot width used in simulation which determines the network capacity of XD.

Keywords - Wireless Sensor Network, Data Collection, Data Dissemination, Cross-layer Design, Time Synchronization, Experimentation

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Timely and reliable data transmission at high traffic load has always been the goal of mission-critical data collection protocol. For example, location system in elderly care centers which collects location data and vital signals of the elderly must transmit these data to the nursing staff with high delivery rate and low end-to-end delay even in the circumstance where numerous data needs to be forwarded within a short period of time.

To achieve such requirement, a work [1] proposes magnetic diffusion (MD), which is good in real-time data availability and very simple in implementation. With these advantages, MD is currently adopted as the data collection mechanism of location system in the elderly care center of National Taiwan University Hospital Beihu Branch

[41][42].

MD, inspired by the physics in magnetism, is a simple diffusion-based data dissemination mechanism which mimics the behavior that magnetic materials are attracted to magnets by low-to-high magnetic fields. Making the analogy of data sink to magnets and data to nails, the data will be attracted towards the sink through multiple shortest paths by the three basic principle in MD: (1) each node is assigned a charge number in a way that the closer the hop distance to the sink, the larger the charge is; (2) every node transmits packets with its own charge; (3) relay node forwards data only if the charge of data is smaller than itself. The end-to-end delay of MD is low because the paths selected in this way are the shortest in hop count. However, due to several paths discovered and the fact that only simple CSMA is adopted as the MAC mechanism of XD, packets may collide severely when the network density is high.

To solve this problem, another work [3] proposes Cross-layer Diffusion (XD) which hybridizes CSMA with radio range based TDMA as the MAC mechanism for MD. The special TDMA aims at achieving a good balance of data delivery rate and end-to-end delay by utilizing the magnetic charges established by MD. The simulation results provided in that work shows that XD is promising. Nevertheless, as far as we know, XD has not been implemented successfully until now. Therefore, the goal of this work is to complete the implementation of XD and evaluate whether XD is better than MD.

The following contributions are made in this paper:

- This is the first work succeeding in implementing XD. Furthermore, we design all the component by ourselves instead of using TinyOS which has all the components ready-to-use but problematic revealed in prior work [4], making the usage of bandwidth in XD at least twice more efficient than prior works [3][4].
- We evaluated XD versus MD with the same settings as in simulation to show a serious problem that is not considered in simulation - the performance of XD declines drastically with increasing TDMA slot width, which makes XD inapplicable in general.
- Also, we evaluated XD versus MD with settings in real applications such as location system [41][42] to show that XD is not as appropriate as MD considering that data collection protocol should provide enough bandwidth to support many users.

The rest of this paper is organized as follows. Chapter 2 presents the related work. The mechanism of XD is described in Chapter 3. Chapter 4 provides the simulation results and the reason of unsatisfactory performance of XD in prior implementations. The key elements in implementing XD are described in Chapter 5, followed by the evaluation of XD versus MD in Chapter 6. We discuss the results in Chapter 7 and lesson learned in Chapter 8. Finally we conclude in Chapter 9.

Chapter 2
Related Work

Data collection requires routing protocol for path discovery as network layer and MAC protocol for data forwarding as link layer. Up to now, plenty of data collection mechanisms for wireless sensor network have been developed, in network layer, link layer, or both, to cope with different task purposes. According to the number of source node and sink node, the scenario can be classified as: multi-sources to single-sink (many-to-one), single-source to multi-sink (one-to-many), and multi-sources to multi-sink (many-to-many). Moreover, the nodes (source, sink, or relay) may be mobile. In general, the most important metrics include: data delivery ratio (reliability), end-to-end delay (latency), and radio duty cycle (energy consumption), scalability. XD is designed in network layer and link layer to have high data delivery ratio and low end-to-end delay for multi-sources to single-sink in both static and mobile scenarios, while being able to conserve energy by turning off the radio in unused TDMA slots or slots without incoming packets.

For routing protocols, there are some well-known mechanisms showing good performance for specific scenario [20]. CTP [12] is a state-of-the-art mechanism for static nodes in many-to-one scenario. It maintains link quality to find good path to sink node by using datapath validation and adaptive beaconing. Previous work [13] shows that CTP is much better than MD [1] in general. Trickle [25], originally designed for propagating code updates, can serve as an algorithm for one-to-many scenario. Its requirement of data consistency and reliability trade off latency and energy consumption. On the contrary, Dozer [24], the state of the art which achieves ultra-low energy consumption for low traffic data collection to one sink, is more susceptible to dynamic environments. Being a mission critical protocol, XD has a better balance in those metrics. MUSTER [26] is one of the few routing protocol particularly designed for many-to-many communication. But for services which require only one sink, MUSTER still suffers from the control overhead caused by many sources, which is not a problem in XD.

When it comes to mobile cases, MobiSense [27], using two-way end-to-end IPv6 UDP connections, is the one expressly designed for mobile sources as far as we know; as for mobile sink scenario, Backpressure Collection Protocol (BCP) [28] outperforms other protocols [29][30] which also aim at dealing with such case. Compared with tree routing protocols, backpressure algorithms suffer from large end-to-end delay at the order of seconds for only one hop from sink. Even BCP utilize LIFO queue rather than traditional FIFO queue to decrease end-to-end delay to a large extent, it is still at the order of several tens of milliseconds for one hop. On the basis of MD, XD is able to adapt to dynamic environments, while only having several millisecond one-hop delay.

Medium access control protocols in link layer are usually classified as TDMA-based and contention-based, which can be further divided into sender-initiated [31][32][33][34] and receiver-initiated [35][36][37][38]. It is well known that TDMA leads to larger end-to-end delay and pure contention-based protocols perform worse than others under congested network. As a result, XD combines radio range based TDMA and CSMA, the simplest method in contention-based approach, as its MAC protocol to balance data delivery rate and end-to-end delay. Besides [1], XD not only outperforms ZMAC [11], another mechanism with hybrid TDMA and CSMA, but also has lower complexity in TDMA schedule.

FTSP [8] is a well-known time synchronization protocol available in TinyOS [5]. PulseSync [9] further increases the synchronization accuracy especially for larger network size. However, both of them rely on a high speed 7.37MHz quartz oscillator which is not provided for every common used WSN platform. Compared with 32 KHz oscillator on all platform, high resolution clock source not only consumes more energy but also requires more bytes in the clock counter. To improve these drawbacks, VHT [10] adopts high speed clock only to recording packet timestamp, thus offers 1 microsecond synchronization accuracy without those trade-off. Since the platform we use does not have a high speed crystal oscillator, XD adopts a naive but reasonable time synchronization method with synchronization error smaller than 300 microseconds for 3-hop network size, only decreasing the transmission capacity of XD a little.

Most mechanisms avoid using too much flooding and broadcasting because they are costly in terms of energy, latency, and bandwidth. Nevertheless, Flash Flooding [17] controls the concurrency of transmission between neighboring nodes and takes advantage of capture effect [15][16], which is the ability of FM radio to receive one of several concurrent transmissions, to improve the inefficiency in traditional flooding.

Unfortunately, scalability problem exists in Flash because the probability of packet reception decreases as the number of concurrent transmissions increases. XD does not rely on such opportunistic method.

Glossy [18] provides a much more powerful flooding than Flash by synchronizing all concurrent transmissions of the same packet to make them form constructive interference which no longer has scalability problem. The requirement is that the temporal displacement should not exceed 0.5μs for 802.15.4 radio chip. To achieve this, it demands a very delicate development and parameter settings. In [19], there are more detailed explanation about constructive interference and derivation of the max temporal displacement.

As an extended work of Glossy, LWB [20] becomes a complete WSN communication protocol with single layer. It supports multi-sources and multi-sinks for both static and mobile cases. As a result of using Glossy, LWB is topology independent. Furthermore, its high data delivery ratio and low energy consumption beat some state-of-the-art protocols [12][24][26][28] in different scenarios which they are designed for. In spite of small latency Glossy has, the schedule overhead of LWB leads to end-to-end delay at the order of seconds.

Glossy-based communication protocols indeed perform very well especially in terms of data delivery rate, energy consumption, and scalability. However, after a closer look the source code of Glossy, we found that the condition to assure constructive interference requires not only fine-tuned time parameters but also assembly level codes which have to be modified with related operation. Although constructive interference seems to be platform-independent, the approach to achieve that is not easy to port on different platforms. Therefore, simple methods such as XD are still necessary.

Chapter 3 Mechanism

XD combines Magnetic Diffusion (MD) [1], an opportunistic redundant data transmission routing protocol, and hybrid CSMA and TDMA which utilizes the magnetic charges established by MD. We describe routing mechanism and MAC mechanism separately in the following subsections.

3.1 Routing Mechanism

The idea of MD comes from the phenomenon in magnetism that magnetic materials are attracted to magnets by low-to-high magnetic fields since it is similar to the process of data collection. As a result, the three basic principle in MD are: (1) each node is assigned a charge number in a way that the closer the hop distance to the sink, the larger the charge is; (2) every node transmits packets with its own charge; (3) relay

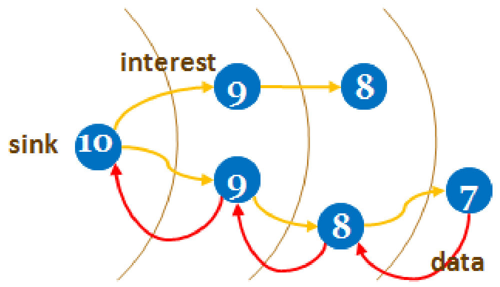


Figure 3.1: Flow of Interest and Data Packets in MD [3].

Number in the circle indicates charge of each node.

node forwards data only if the charge of data is smaller than itself. Therefore, data travel towards sink, as showed in Figure 3.1, by potentially multiple paths without unnecessary broadcasting. Besides, duplicates should be discarded to avoid network congestion.

Another type of packet in MD is called interest which is used to establish the charge of each node. In the beginning, sink broadcasts interest with its charge, which is the highest in the network. Upon receiving interest with new sequence number, the node decreases the charge number by one, records the modified number as its charge, and forwards that packet with new charge. Such process will set charges which could guide data packet flow to sink and will be periodically executed to adapt to potential topology change. Unlike data packet, the forwarding policy is to check the sequence number of the packet instead of the charge number.

Consider TDMA is adopted by XD, the two types of packet can be operated in data propagation phase and interest broadcast phase separately to reduce collision of important interest packet. Moreover, interest packet in XD also serves as the synchronization packet.

3.2 MAC Mechanism

To have higher data delivery rate, XD uses hybrid CSMA-TDMA to reduce more collisions than traditional MD with CSMA. Typically, MD only adopts CSMA to avoid collisions. In Figure 3.2, we classify collisions into six canonical types based on radio

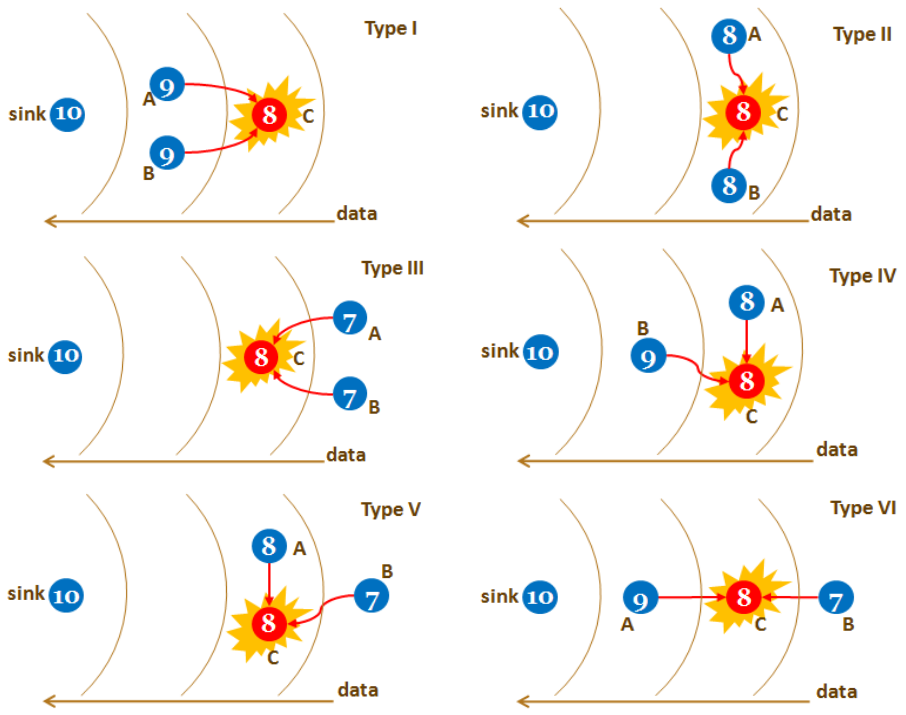


Figure 3.2: Types of Collision in MD [3].
Packets from node A and B collide at node C.

range information in MD. Type I, II, and IV can be ignored since transmission in those types does not forward data towards the sink. Type III and V can be partially reduced if the sending nodes can hear each other, otherwise those nodes will form hidden terminals as type VI. To reduce collisions further, XD hybridizes CSMA and radio range based TDMA, preventing neighboring nodes with different charge sending simultaneously. Assuming transmissions cannot be hear from nodes farther than adjacent levels, only three time slots are used in TDMA schedule to lower the tradeoff in end-to-end delay. Therefore, the schedule is that nodes send when the slot number equals to its number of charge mod 3, as showed in Figure 3.3. Besides, nodes which are one charge level closer to sink than those sending nodes should wait to receive, while nodes which are one charge level farther can sleep to save energy. XD is designed as such to get high data delivery rate and low end-to-end delay.

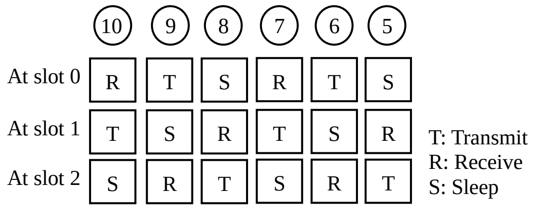


Figure 3.3: Three-Level TDMA Schedule [3].

Number in the circle indicates charge of each node.

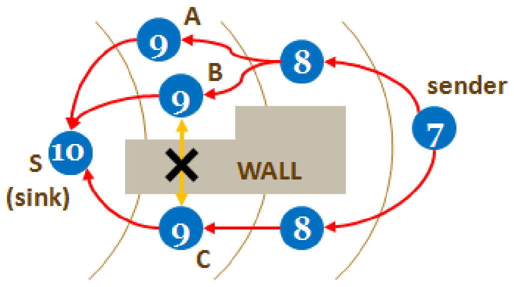
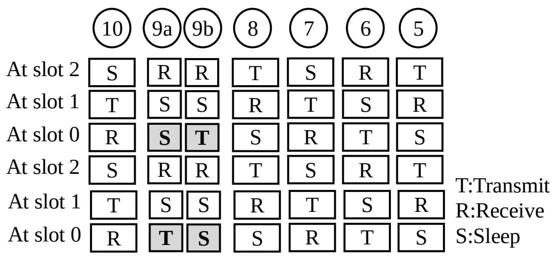


Figure 3.4: Hidden Terminal Effect around Sink [3].

Though node A, B and C are neighbors of the sink, node C is separated from node A and B by a corner in indoor environment.

Consider that the workload of forwarding is higher when a node is closer to the sink in terms of charge level, collisions from hidden terminals between nodes at the end of different paths toward sink, e.g., node B and C in Figure 3.4, would decrease packet delivery rate greatly. To prevent this, XD further refines the TDMA schedule at the sink by dividing the time slot of its neighbors into multiple sub-slots such that each sub-slot is used by one group in these neighbors which share the same carrier sense region. This



	10	9a	9b	8	7	6	5
At slot 2	S	R	R	T	S	R	T
At slot 1	T	S	S	R	T	S	R
At slot 0	R	**S**	**T**	S	R	T	S
At slot 2	S	R	R	T	S	R	T
At slot 1	T	S	S	R	T	S	R
At slot 0	R	**T**	**S**	S	R	T	S

T:Transmit
R:Receive
S:Sleep

Figure 3.5: Extended TDMA Schedule [3].

The nodes with charge 9 are neighbors of the sink. That slot is further divided into two for two groups of neighbors, i.e., 9a and 9b.

can be achieved by collecting neighbors' information at the sink. An example of the refined schedule is shown in Figure 3.5, where 9a goes to sleep when 9b transmits packets and vice versa.

If not specified, MD in this paper means MD with CSMA.

Chapter 4
Prior Work

XD was proposed with a set of trace-driven simulations using ns-2 in the beginning [3]. That work compares XD to MD using traces collected from a WSN with 12 nodes deployed on the 6th floor of Berry Lam Hall at National Taiwan University as showed in Figure 4.1.

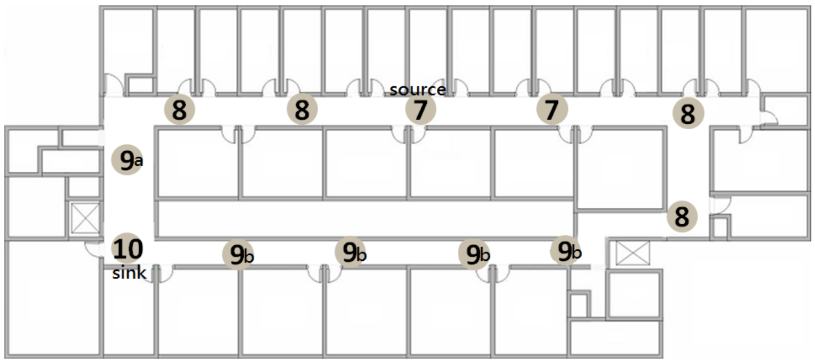


Figure 4.1: Testbed Topology in Simulation [3].

Number in the circle indicates charge of each node after interest broadcast phase, while 9a and 9b indicate different groups of neighbors around sink.

Those traces in simulation are the packet reception rate (PRR) of each node by programming 11 nodes as receiver and 1 node as sender. Each of the 12 nodes will take turn to be the sender. The PRR is obtained by calculating the ratio of the number of packets received to the number of packets sent. In the simulation, when a node sends a packet, other nodes determines based on the measured PRR whether the packet can be heard at the intended receiver. No capture effect is considered in simulation, which is the ability of some frequency modulated (FM) radios to correctly demodulate one of several concurrently transmitted packets because the modulation of the weaker signal no longer exists at the demodulator output or at least is attenuated to a very high degree [15].

Figure 4.2 and Figure 4.3 are the simulation results of 1000 128-byte packets

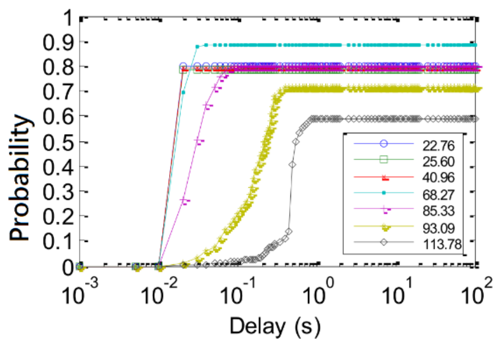


Figure 4.2: CDF of End-to-End Delay in MD with CSMA [3].

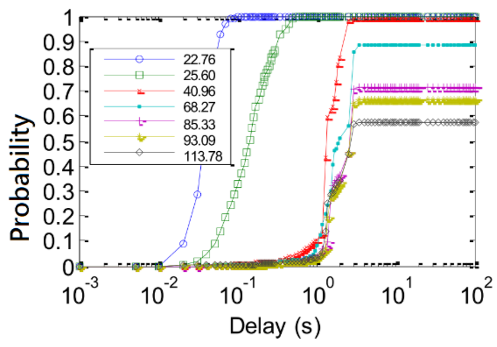


Figure 4.3: CDF of End-to-End Delay in XD [3].

transmitted by the source node in topology showed in Figure 4.1. The results indicate that XD performs better than MD in terms of packet delivery rate in the low-load cases, i.e., the data rate below 68.27kbps. But this, however, trades off the end-to-end delay, a problem general to TDMA-based solutions.

To understand the difference between MD and XD in detail, packets are classified

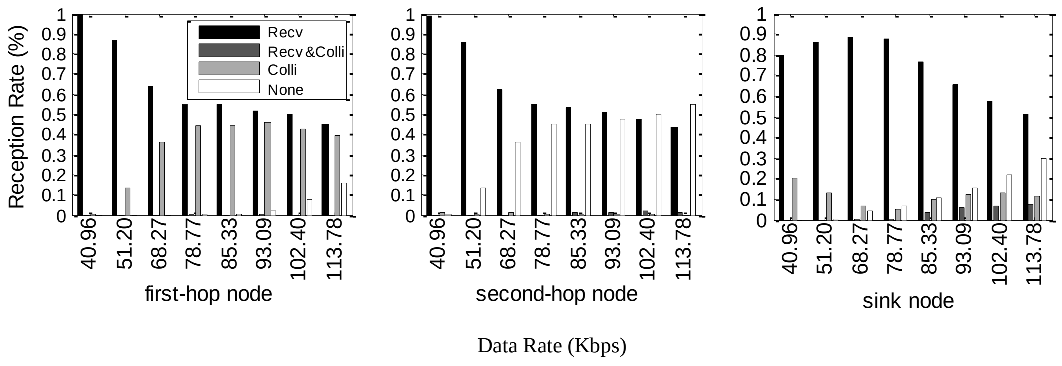


Figure 4.4: Distribution of Packets Observed at Each Node in MD with CSMA [3].

First-hop and second-hop nodes are nodes with charge 8 and 9 in Figure 4.1, separately.

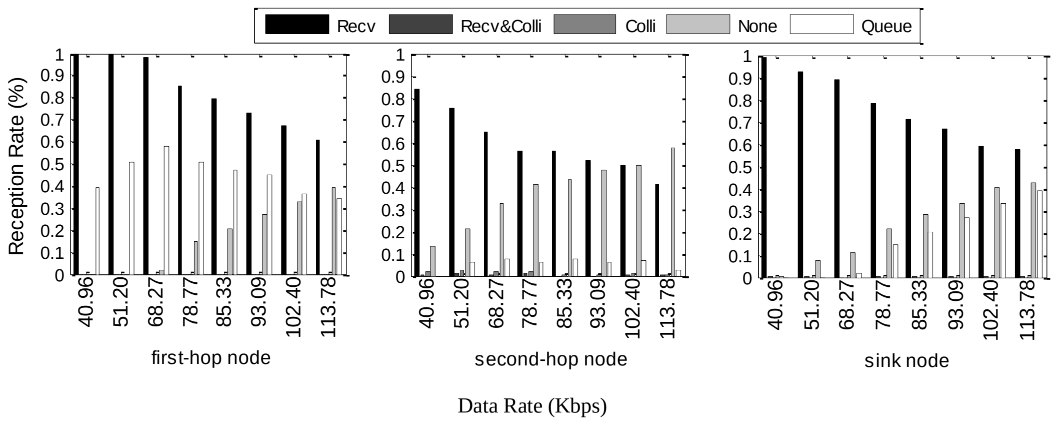


Figure 4.5: Distribution of Packets Observed at Each Node in XD [3].

First-hop and second-hop nodes are nodes with charge 8 and 9 in Figure 4.1, separately.

into four types by how they are received at each node along the transmission path and the distribution are showed in Figure 4.4 and Figure 4.5. "If all duplicates of a packet are successfully received at the node, the packet is categorized as the first type (Recv). If a certain duplicate of a packet arrives while at least one of the duplicates is collided, the packet is categorized into the second type (Recv & Colli). The third type is for packets whose duplicates are all collided (Colli). The last type is for packets that are never sent by the upstream nodes (None). The sum of the four types is 100% which is the total number of packets sent by the data source. [3]" Besides, there is a new bar (Queue) in Figure 4.5 for XD to indicate packets drops of sending queue at that node due to overflow, while the bar (Queue) shown at the sink node presents as the queue status of the source node. As for MD, there are no sending queue drops. We can see that there are hardly any collisions in XD and sending queue drop at the source node is the main reason of loss of packets in XD.

Figure 4.4 also explains why the packet delivery rate of MD is highest at middle traffic load rather than at low traffic load [3]. This is because at low traffic load, the few collisions at first-hop node and second-hop node lead to more collisions at the sink node than at middle traffic load. Moreover, we can see that the proportion of (Colli) and

(None) at sink in MD is lowest at middle traffic load, which is the main reason of loss of packets in MD.

To the best of our knowledge, only two works [3][4] made attempt to implement XD. Both of them use TinyOS 2.x [5], which is a widely-used operating system designed for WSNs. However, the component-based architecture which encapsulates low-level codes make it hard to modify codes at low-level without causing compatibility problems, leading to inefficient usage of bandwidth and worse performance of XD than MD in those works. To make things worse, that problem also makes TDMA in their implement inaccurate, which means XD had not been implemented successfully.

Chapter 5 Implementation

Based on lessons from prior works, we choose to implement all component of XD by ourselves with IAR Embedded Workbench [6], which is a set of development tools for building and debugging embedded application. The platform we use is Taroko, which is a clone of standard TelosB, featuring a MSP430F1611 microcontroller [39] and CC2420 radio [40] compliant with the IEEE 802.15.4 standard. The following sections introduce the key design elements.

5.1 Packet Format

To lower the control overhead, we discard the default 802.15.4 packet format and define our own, showed in Figure 5.1, for both data and interest. After the sending

preamble	SFD	length	type	charge	last hop ID	packet source ID	sequence number	Timestamp	payload	Frame Check Sequence
4	1	1	0.5	0.5	2	2	2	4	0~114	2

Figure 5.1: The Packet Format of XD.

The numbers indicate the size of each field in bytes.

command is executed, it takes 4-byte byte (or 6-byte by default) periods for radio to calibrate frequency before sending preamble. Preamble and SFD (Start of Frame Delimiter) are generated by hardware and not recorded in radio buffer.

The length byte is a must to indicate the followed frame length. The next byte combines packet type and charge, the keys of mechanism of XD. Last hop ID is reserved for potential future control. Packet source ID and sequence number are set by data source and used to differentiate packets with different content in payload. Timestamp is a must of interest packet for sink to synchronize all the other nodes. For data packet, we use timestamp to analyze end-to-end delay and consider it as part of payload. The maximum payload size is 114 since the maximum packet size is limited to the size of radio buffer, which is 128 bytes on Taroko. Frame check sequence (FCS) is the check sum of the whole packet except the first length byte. If enabled, verification of FCS can be done automatically by hardware. Excluding preamble and SFD, the overhead of data packet in XD is 10 bytes, smaller than the typical 12-byte overhead in 802.15.4 and 24-byte overhead in prior work [4].

5.2 Network Layer

Figure 5.2 shows the flow of routing mechanism as described in section 3.1 . This mechanism is implemented in low-level codes with radio driver to save the time of pushing a packet into different queue, i.e., the memory space on MCU (micro-controller) to receive the incoming packet from radio buffer is the location at the back of the queue which corresponds to the type of incoming packet. The data queue size is set to be 50 while the interest queue size is set to be 5 because sink only broadcast a few interest

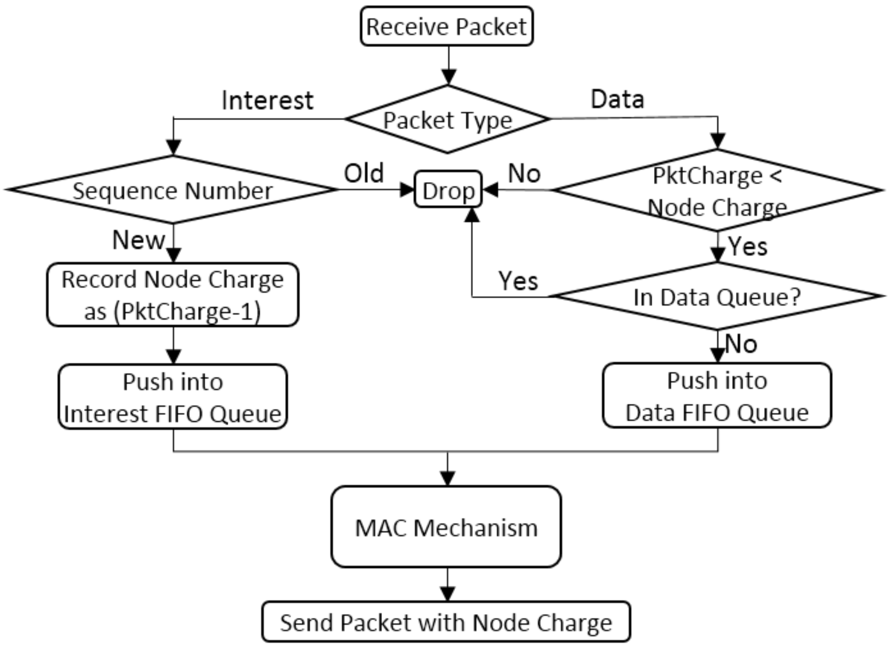


Figure 5.2: Flow Diagram of Routing Mechanism.

packets in each interest phase as described in section 5.3.2. The queue size should not be set larger due to limited 10KB RAM size on our platform. Duplicates is checked by a cache of size 251 with sequence number and data source ID as its cache key. The cache function is:

$$((\text{sequence number}) \times (\text{data source ID}))\ \%\ (\text{cache size})$$

5.3 MAC Layer

This section describes the implementation detail of TDMA and CSMA in XD. The former is separated into time synchronization and TDMA schedule in different subsections.

5.3.1 Time Synchronization

One of the basics in time synchronization is the exchange of timestamp between nodes. This can be done by extracting information from hardware. On the radio chip, there is a pin indicating the end of transmitting or receiving SFD with timing error smaller than 1 μs, which is much smaller than the granularity of a common used 32768 Hz clock source. That pin is connected to the timer capture pin, which records the timestamp automatically whenever the SFD events happen. As a result, the sender can insert the captured timestamp in a packet and the receiver can compare the timestamp difference between the sender and itself, achieving time synchronization of them.

However, one-time synchronization is not enough because the difference of those timestamps still grows with time due to clock skew, which is difference of frequency between each clock caused by manufacture errors, and clock drift, which is change in clock frequency as temperature or supply voltage varies. To keep all clocks time-synchronized within a certain error threshold, synchronization process should be performed periodically. To have a longer period of synchronization while all the synchronization errors are below the threshold, clocks can automatically calibrated against the reference clock before next synchronization process based on information gathered previously.

In our implementation, a 32768 Hz crystal oscillator is used as the clock source of each node. The procedure of synchronization is as follows. First, we utilize interest packet as synchronization packet and broadcast it multiple times in each interest broadcast phase, which is at the beginning of every minute as described in section 5.3.2. Second, we choose the local time of sink as the global time, i.e., reference clock. In this way, timestamp of sink is broadcasted with interest packet. With the spread of interest packet, all nodes in the network are synchronized. Third, we calibrate clocks of those nodes by the prediction of difference in time between reference clock and them according to the average of the difference in the past 5 minutes. In this way, we compensate not only clock skew between different clocks but also clock drift due to variant temperature and supply voltage, having an acceptable synchronization accuracy required in XD as showed in section 6.1 .

5.3.2 TDMA Schedule

We reserved a small duration, shorter than 350ms, at the beginning of every minute for interest broadcast phase, which is long enough for 30 nodes to broadcast five interest packets. If there are more nodes in the network such that the period of interest phase is too long, we can increase the interval between interest broadcast phases to balance the control overhead.

As for data propagation phase, TDMA slots are scheduled as described in section 3.2 . The only thing we need to decide is the width of each slot. Consider that the performance of XD is dominated by queue drops while MD is not affected as described in Chapter 4, our goal is to make the width as short as possible. In simulation, the slot width is only 5ms, where 4.1ms is used for transmitting a packet with size up to 128-byte and the rest 0.9ms is all for CSMA back-off without considering any computation time. However, limited to the low clock frequency of a microcontroller and some operations not considered in simulation, the computation time in reality usually takes a long duration. Compared with the 15ms slot width of all prior implementation [3][4], the 6.946ms slot width of our implementation is really a large improvement. Other than the 5ms for packet transmission and back-off, the rest 1946μs is composed of 170μs for CCA sampling described in section 5.3.3, 128μs for radio frequency calibration, 160μs for preamble and SFD transmission described in section 5.1 , 488μs for synchronization error threshold, 1000μs for all the other computation. Although it may be possible to make the slot width shorter by adopting a better synchronization and further optimize the current computation, it is impossible to make the width shorter than 6ms. Besides, if there are some applications added, the width may need to be increased. Therefore, 6.946ms slot width is used in our evaluation.

5.3.3 CSMA

The CSMA part is specifically designed to match the operation of XD. First, nodes perform random back-off before executing the sending command. Unless specified, the default back-off parameters in our implementation are chose to be the same as in simulation, which uses 904μs for back-off periods and 41 back-off choices.

After the back-off, nodes perform CCA (Clear Channel Assessment) [33] for about 170μs continuously. If no packets are detected and the RSSI (Receive Signal Strength Indicator) values are continuously weaker than the CCA threshold, the sending command will be executed. The CCA threshold is set to be -85dBm, higher than the maximum RSSI of environment noise, which is -89dBm.

For general radio driver provided by the manufacturer, the default -77dBm CCA threshold, much higher than ourselves, increases the probability of channel false clear and packet collision. Besides, the default CCA mechanism only takes one RSSI sample, which increases the probability of channel false busy and decreases bandwidth utilization.

All of the above operations are executed in the time slot where the nodes can send. This is accomplished by inserting several additional timing checks between those operations. Unlike component-based TinyOS, our CSMA can be hybridized with TDMA in low-level codes to optimize the performance.

Chapter 6
Evaluation

This chapter evaluates time synchronization error first in section 6.1 . Then we compare XD with MD under simulation settings after verifying the correctness of our implementation in section 6.2 . At last, we compare XD to MD under scenario of real applications such as location system [41][42] in section 6.3 .

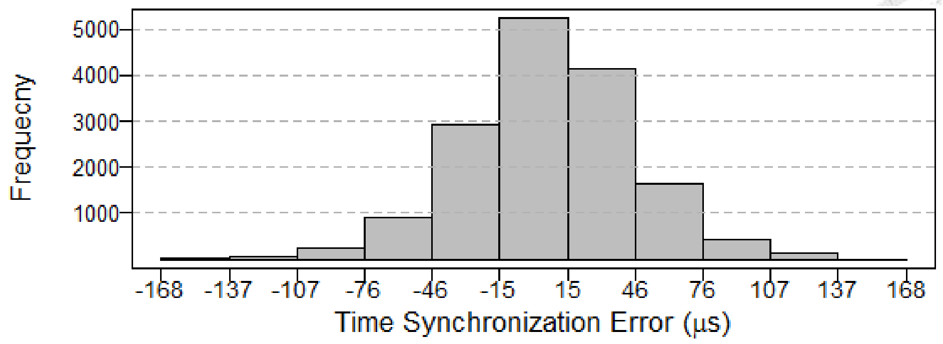


Figure 6.1: Distribution of Time Synchronization Error between Each Pair of Nodes under Constatnt Temperature.

6.1 Time Synchronization Error

In the beginning, we hardcode magnetic charges of 4 nodes to evaluate time synchronization error of a 3-hop network under constant temperature. Using information of difference between timestamps of a receiver and a sender as described in section 5.3.1, we make the statistics of time synchronization error between each pair of nodes for several hours, as showed in Figure 6.1. We can see that the maximum error is about only 1/3 of the synchronization error threshold set in section 5.3.2, which is 488us.

To further evaluate the performance under dynamic temperature environment, we

Temperature (℃)	25	25→40	40	40→10	10	10→40	40	40→10	10	10→25	25
Duration (min)	25	7	25	30	25	15	25	30	25	7	25

Table 6.1: Temperature Settings of the Temperature Chamber.

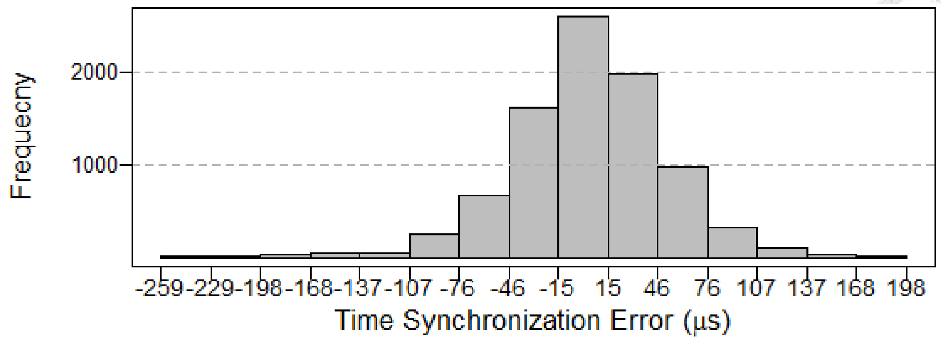


Figure 6.2: Distribution of Time Synchronization Error between Each Pair of Nodes under Dynamic Temperature.

deployed those nodes in a temperature chamber with settings showed in Table 6.1, where temperature ranges from 15℃ to 40℃ with max rate of ±25℃/15min. Even the environment temperature changes so drastically, the maximum error showed in Figure 6.2 is still about only 1/2 of the synchronization error threshold.

For all the other experiments of XD in real testbed such as BL-live testbed [7], we make the same statistics as showed in Figure 6.3 to make sure we evaluate XD under accurate TDMA. Even if we do not plot such figure, there are checking codes on nodes to warn if the maximum synchronization error exceeds the error threshold.

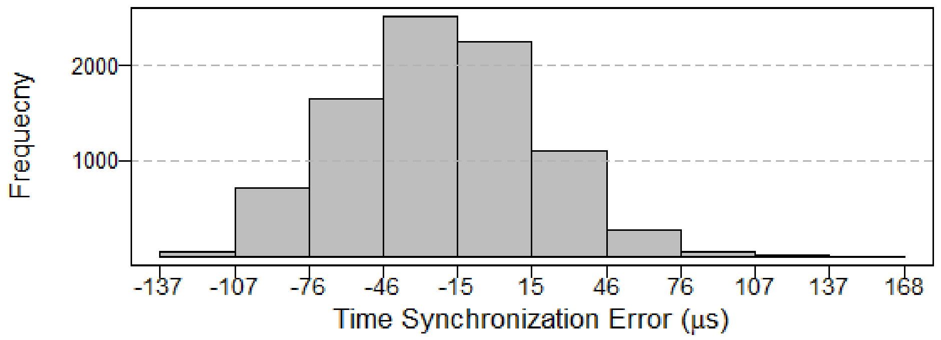


Figure 6.3: Distribution of Time Synchronization Error between Nodes Deployed in Real Testbed.

6.2 Performance of XD and MD under Simulation Settings

This section aims at verifying the correctness of our implementation and comparing XD to MD under simulation settings. We first introduce the consequences caused by the differences between implementation and simulation in section 6.2.1. Afterwards, we compare the result of implementation to that of simulation for both XD and MD to verify the correctness of our implementation in section 6.2.2. Finally, we compare XD to MD under simulation settings in section 6.2.3.

6.2.1 Preliminary

All experiments of our implementation in section 6.2 are evaluated with almost the same settings and topology used in simulation as described in Chapter 4. There are only three differences. First, the number of packets evaluated in implementation is much more than in simulation to make the statistics of our experiment robust. We repeat each experiment 10 times, where at least 2500 packets are sent by the source node each time. Abnormal experiments caused by crash of USB connection and unexpected interference are eliminated before we get the average of them.

Second, the TDMA slot width of XD in simulation is too small to be realistic as described in section 5.3.2. Because of the characteristic of TDMA, the bigger the slot width is, the less the network capacity of XD is. Moreover, the relationship between slot width and network capacity of XD is inverse proportional. When the network capacity is saturated, packets in the sending queue start to be dropped due to queue overflow, which dominates the packet delivery rate in XD as explained in Chapter 4. Therefore, to achieve comparable performance as in simulation in terms of packet delivery rate, the traffic load of XD should be scaled down by the slot width ratio of implementation to simulation, which is 1.3892 (6.946/5.000). However, the end-to-end delay at high traffic load in implementation will definitely be larger than in simulation due to larger TDMA slot width.

Finally, the channel we use in evaluating the result of implementation is different from the trace of simulation. There are 16 channels ranging from 11 to 26 available in 802.15.4, where channel 25 and 26 are not overlapped with Wi-Fi and channel 26 are used in simulation. It is well-known that packet delivery rate in channel 25 or 26 is better than in other channels in a Wi-Fi environment such as BL-live testbed. However, there are constantly 802.15.4 interference in channel 25 and 26 on BL-live testbed from nearby building since last year. Therefore, we have to use other channels with more interference than the channel used in simulation. Interfered by Wi-Fi signals, the number of packets delivered decreases due to the increase of bit error rate. In addition, the network capacity also decreases because Wi-Fi interference will make the probability of CCA (Clear Channel Assessment) false busy higher. As a result, we expect a lower packet delivery rate of implementation in both MD and XD than the rate of simulation, where the difference of delivery rate in XD should be larger than in MD because of more sending queue drops happen only in XD, explained in Chapter 4.

6.2.2 Comparison between Implementation and Simulation

Figure 6.4, Figure 6.5, and Figure 6.6 show the implementation result of MD, XD, and XD with traffic scaled down, separately. To make it easier to compare the performance, we organize the packet delivery rate of simulation and implementation in Table 6.2 and Table 6.3.

(kbps)	22.76	25.60	40.96	68.27	85.33	93.09	113.78
MD with CSMA	79.9	78.8	79.5	88.5	79.9	71.8	58.5
XD	99.9	99.9	99.2	88.6	71.5	66.5	57.7

Table 6.2: Packet Delivery Rate in simulation.

(kbps)	22.76	25.60	40.96	68.27	85.33	93.09	113.78
MD with CSMA	93.8	93.7	93.0	83.5	73.0	68.0	57.4
XD	98.0	97.5	89.8	58.4	47.1	42.9	35.1
XD with Traffic Scaled Down	98.6	98.4	96.7	82.8	65.3	60.3	48.8

Table 6.3: Packet Delivery Rate in Implementation.

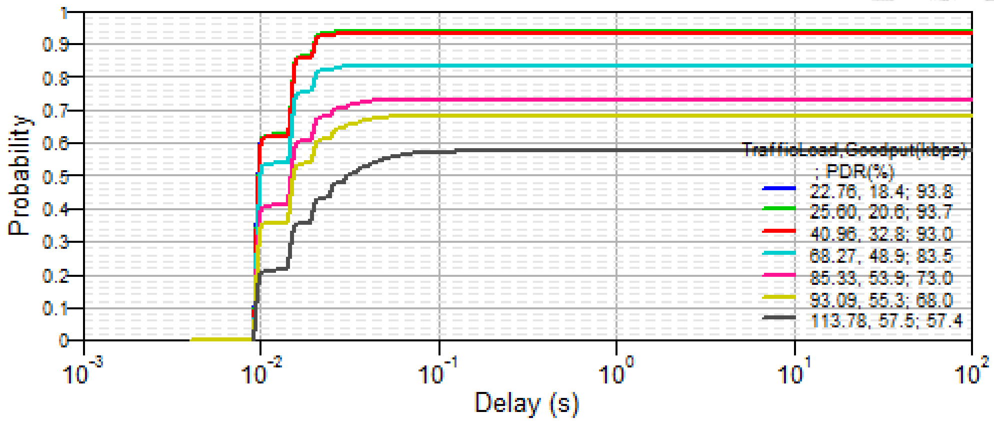


Figure 6.4: CDF of End-to-End Delay in MD with CSMA under Simulation Settings.

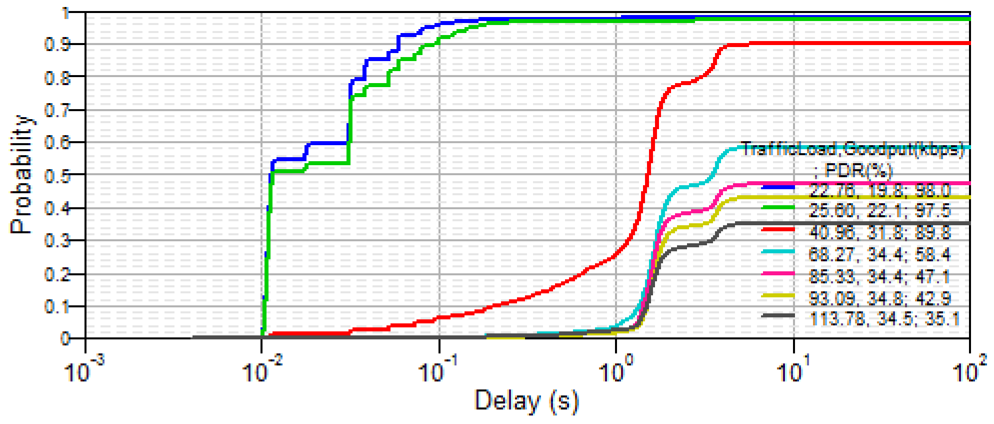


Figure 6.5: CDF of End-to-End Delay in XD under Simulation Settings.

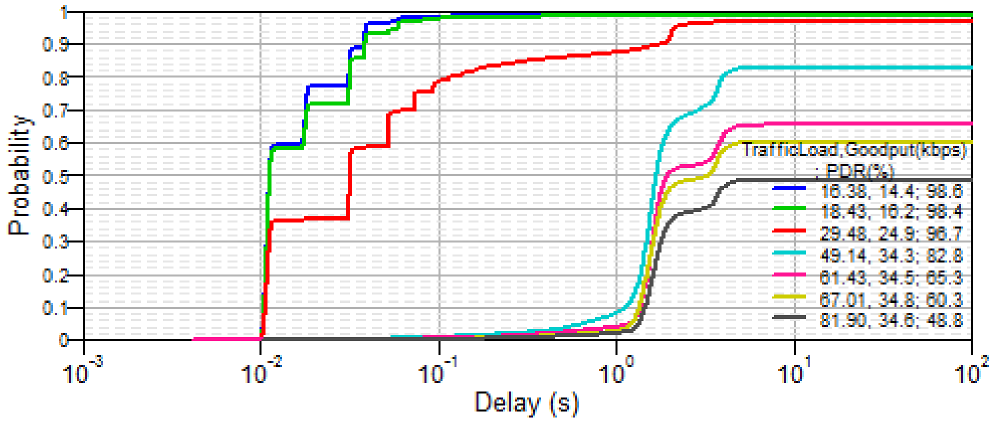


Figure 6.6: CDF of End-to-End Delay in XD with Traffic Scaled Down by Slot Width Ratio of Implementation to Simulation.

We first compare the result of XD in simulation to XD with traffic scaled down in implementation. Comparing Figure 4.3 to Figure 6.6 and Table 6.2 to Table 6.3, we can see that the delivery rate in implementation is lower than in simulation. This is because channel with more interference leads to higher bit error rate and more sending queue drops as explained in section 6.2.1. Besides, the difference of packet delivery rate grows with the increase of traffic load because the increase of sending queue drops caused by CCA false busy due to interference is larger when the traffic load is higher. As for end-to-end delay, the delay for traffic load at 40.96 kbps in implementation is much lower than in simulation because that traffic load happen to be at the transition region of the end-to-end delay. To sum up, XD with traffic scaled down in implementation is comparable to XD in simulation in same channel environment.

Then we compare the result of MD in simulation to MD in implementation. Comparing Figure 4.2 to Figure 6.4 and Table 6.2 to Table 6.3, we can see that the delivery rate in implementation is quite different from our expectation. For traffic less than or equal to 40.96 kbps, delivery rate in implementation is much higher than in simulation. As verified in [3], this is because of not considering any capture effect in simulation. For traffic larger than or equal to 68.27 kbps, delivery rate in implementation is lower than in simulation as expected in section 6.2.1. In addition, the trend is that the difference of delivery rate between implementation and simulation at traffic load in this range decreases with growing traffic load. Referring to Figure 4.4, we can see that the proportion of packet collisions in simulation rises with growing traffic load from 68.27 kbps. As a result, the proportion of packets collided in simulation but recovered by capture effect showed in the result of our implementation also rises with growing traffic load from 68.27 kbps, leading to decreasing difference of delivery rate with growing traffic load. As for end-to-end delay, the delay at high traffic in implementation is smaller than in simulation because more duplicate packets are received at the sink by the help of capture effect. To sum up, the result of MD in implementation is better than the result in simulation at low traffic rate due to capture effect and worse than the result in simulation at high traffic rate due to channel with more interference. If there is no capture effect, MD in implementation is comparable to MD in simulation in same channel environment.

6.2.3 Comparison of XD to MD

This subsection compare XD to MD under simulation settings in implementation by Figure 6.4, Figure 6.5, and Table 6.3. We also make the corresponding comparison of

(kbps)	22.76	25.60	40.96	68.27	85.33	93.09	113.78
Name	Very low		Low	Medium	High		

Table 6.4: Types of Traffic Rate in Section 6.2.3.

them in simulation by looking at Figure 4.2, Figure 4.3, and Table 6.2. To make the comparison clearer, we divide the range of traffic rates into four region as showed in Table 6.4. In the following discussion, we make the comparison by the order from very low traffic load to high traffic load. If not specified in this subsection, the performance of XD/MD is referred to the performance of XD/MD in implementation.

For very low traffic load, XD is better than MD in both implementation and simulation. The difference of delivery rate is 4.2% and will be much larger when there is no capture effect. For low traffic load, XD is worse than MD by 3.2% but much better than MD in simulation. This is also due to the existence of capture effect in the result of implementation. If there is no capture effect, XD will definitely be better than MD. For medium traffic load, XD is much worse than MD due to a great number of sending packet drops, while XD with traffic scaled down is comparable to MD as expected in simulation. For high traffic load, XD is worse than MD in simulation. However, due to lots of sending queue drops in XD, the performance of it is much worse than expected in simulation.

It is worth to note that the trend of comparison in implementation between XD with traffic scaled down and MD is almost the same as the trend of comparison in simulation between XD and MD except that MD in implementation at low and very load traffic load is not as bad as expected due to capture effect. To conclude, XD is better than MD at low and very load traffic load whether there is capture effect or not. However, due to impractically small slot width used in simulation as explained in section 5.3.2, XD is much worse than expected at higher traffic load. Moreover, if there are some inevitable computation time that needs to be added in slot by applications, the performance of XD will decrease greatly due to sending queue drops as showed in Table 6.5 while MD will not be affected. The proportion P_{QD} is calculated by

Traffic load (kbps)	22.76	25.6	40.96	68.27	85.33	93.09	113.78
Inter Packet Interval (ms)	45	40	25	15	12	11	9
TDMA slot width (ms)							
5	0%	0%	0%	0%	20%	27%	40%
6	0%	0%	0%	17%	33%	39%	50%
7	0%	0%	0%	29%	43%	48%	57%
8	0%	0%	0%	38%	50%	54%	63%
9	0%	0%	7%	44%	56%	59%	67%
10	0%	0%	17%	50%	60%	63%	70%

Table 6.5: Proportion P_{QD} of Sending Queue Drops in XD.

$$
\begin{aligned}
P_{QD} &= 1 - \frac{\min(\text{network capacity}, \text{traffic load})}{\text{traffic load}} \\
&= 1 - \min\left(\frac{\text{network capacity}}{\text{traffic load}}, 1\right) \\
&= 1 - \min\left(\frac{\text{inter packet interval}}{\text{TDMA slot width } \times 3}, 1\right)
\end{aligned}
$$

6.3 Compare XD to MD in Real Application

In section 6.2 , only one source node is used to compare the performance of XD to MD. To verify the conclusion of comparison between XD and MD in section 6.2 is independent of the number of source nodes, we evaluate XD and MD under scenario of real applications such as location system [41][42] in this section with topology showed in Figure 6.7, where different number of source nodes send 128-byte packets at inter packet interval (IPI) of 200ms. The sources used for different experiment are listed in

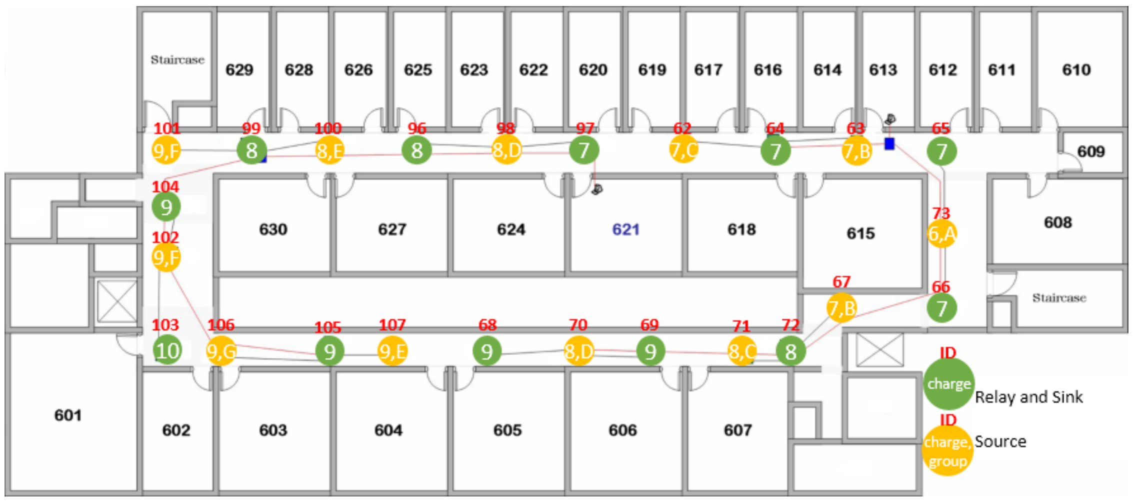


Figure 6.7: Topology Evaluated in Section 6.3

Group name on source nodes is used to distinguish different experiments.

Number of source nodes	1	3	5	7	9	11
Working source nodes	A	A,B	A,B,C	A,B,C,D	A,B,C,D,E	A,B,C,D,E,F
Traffic Load (kbps)	5.12	15.36	25.60	35.84	46.08	56.32
PDR of MD (%)	95.7	87.8	83.8	78.5	77.8	76.8
PDR of XD (%)	97.4	96.0	86.0	74.7	70.8	68.9

Table 6.6: Packet Delivery Rate (PDF) of MD and XD When Different Number of Source Nodes Send Packets at IPI of 200ms.

Table 6.6 by the group name labeled in Figure 6.7. We repeat each experiment 5 times, where about 1000 packets are sent by each source node each time. Abnormal experiments caused by crash of USB connection and unexpected interference are eliminated before we get the average of them.

From Table 6.6, we can see that the trend of comparison between XD and MD is that XD is better at low traffic load and worse at higher traffic load, which is the same as the trend in section 6.2 , i.e., under simulation settings where only one source node is used. Nevertheless, at same traffic load, the delivery rate is much lower when there are more source nodes because of more collisions caused by more intense contention. Besides, we can see that the delivery rate of MD drops faster than XD with increasing number of source nodes at low traffic load, i.e., experiment of 3 source nodes in Table 6.6. This fact shows that the TDMA mechanism in XD indeed helps lower collisions while packets in MD are prone to collide especially when the network density is high.

However, there is tradeoff between collisions and sending queue drops which make XD much worse than MD at high traffic load.

Chapter 7
Discussion

According to the results of evaluation, it is possible to combine the advantages of XD and MD without drawbacks by switching the operation mode between XD and MD under different traffic load. If the traffic load is low, i.e., there would be no sending queue drops in XD, data should be collected by XD. Otherwise, the operation mode should be switched to MD. In this way, data delivery rate at low traffic load would be slightly higher than MD. However, proposed in the same year as XD, CTP [12] has already become the state-of-the-art provided in TinyOS and is proved to be much better than MD in general [13]. Therefore, even if we improve MD by switching to XD mode under low traffic load, the performance will still be worse than CTP.

Chapter 8
Lessons Learned

The most important lesson learned from this work is that we should not simply trust the results of simulation, especially when you are not the one who does the simulation. Throughout the evaluation of MD and XD in this work, we see that there are many discrepancies between the results of implementation and simulation. The key factor making the performance of XD worse than MD is the larger TDMA slot width, which is not a complex parameter, in implementation than in simulation. However, we did not realize it until we finished the implementation and the evaluation. Therefore, we would like to share our experiences in this chapter and hopefully it will save time for future endeavors.

For people who want to build a protocol based on simulation results or conduct experiments to compare with results in simulation, the first step should definitely be reading the paper of that work to make sure you understand how the simulation is done. Second, you should be critical about everything of the simulation such as choices of parameters, equations chosen to approximate other equations, methods used for simulations, etc. There is a good chance that people keep the biased results just to make their work seem plausible enough to be published. The best way to examine this is to check whether the authors explain clearly how the method is chosen and whether they have tried several different parameters which may happen in real world. Even if the simulation is the work you did before, you still need to examine those things again because you may have a better insight at this time. Third, we would suggest people to check and run the simulation program by yourself to make sure there is no bugs especially when the implementation part is very time-consuming. Finally, keep in mind of the parameters and the models in simulation when you implement. Redo simulation as soon as you find out there is something not considered in the simulation which differs from the real situation such that you can adjust anything that does not make sense as early as possible.

Chapter 9
Conclusion

To the best of our knowledge, this is the first paper that XD is successfully implemented. Furthermore, network capacity of XD in our implementation is made closed to the limit. The comparison between XD and MD is evaluated with settings in simulation and in real application. All of the results show that XD is indeed better than MD at traffic load less than 40.96 kbps, especially when there is no capture effect or the network density is high. However, due to unattainably small slot width used in simulation, XD is much worse than MD at traffic load higher than 40.96 kbps. Moreover, the performance of XD declines with increasing width of TDMA slot caused by potentially extra applications. As a result, we conclude that MD is more appropriate than XD in general.

Reference

[1] Hsing-Jung Huang, Ting-Hao Chang, Shu-Yu Hu and Polly Huang, "Magnetic diffusion: Disseminating mission-critical data for dynamic sensor networks," MSWiM, 2005.

[2] Ting-Hao Chang, "Reliable data dissemination in practical sensor networks", M.S. thesis, National Taiwan University, 2006.

[3] Chieh-Ting Huang, "XD: A cross-layer designed data collection mechanism for mission-critical WSNs in urban buildings", M.S. thesis, National Taiwan University, 2009.

[4] Ling-Yen Wu, "Experimental study of a cross-layer designed data collection mechanism for WSN in NTU hospital", M.S. thesis, National Taiwan University,

2011.

[5] Open-source operating system designed for wireless embedded sensor networks – TinyOS. http://www.tinyos.net/.

[6] IAR Embedded Workbench. http://www.iar.com/.

[7] BL-Live Testbed. http://nslab.ee.ntu.edu.tw/wiki/doku.php?id=projects:bl-live_testbed.

[8] Miklos Maroti, Branislav Kusy, Gyula Simon and Akos Ledeczi, "The flooding time synchronization protocol," In Proceedings of ACM SenSys, 2004.

[9] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal clock synchronization in networks," In Proceedings of ACM SenSys, 2009.

[10] T. Schmid, P. Dutta, and M. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," In Proceedings of ACM/IEEE IPSN, 2010.

[11] Injong Rhee, Ajit Warrier, Mahesh Aia and Jeongki Min, "ZMAC: A hybrid MAC for wireless sensor networks," IEEE/ACM Transactions on Networking, June 2008.

[12] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. "Collection tree

protocol," In Proceedings of ACM SenSys, 2009.

[13] Yuan-Feng Tsai, "Experimental study of two data dissemination mechanisms for WSN in NTU hospital", M.S. thesis, National Taiwan University, 2011.

[14] O. Landsiedel, E. Ghadimi, S. Duquennoy, M. Johansson, "Low power, low delay: Opportunistic routing meets duty cycling," In Proceedings of ACM/IEEE IPSN, 2012.

[15] K. Leentvaar and J. Flint, "The capture effect in FM receivers", IEEE Trans. Commun., 24(5), 1976.

[16] D. Davis and S. Gronemeyer, "Performance of slotted ALOHA random access with delay capture and randomized time of arrival," IEEE Trans. Commun., 28(5), 1980.

[17] J. Lu and K. Whitehouse, "Flash fooding: Exploiting the capture effect for rapid fooding in wireless sensor networks," In Proceedings of IEEE INFOCOM, 2009.

[18] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. "Efficient network flooding and time synchronization with Glossy," In Proceedings of ACM/IEEE IPSN, 2011.

[19] Y. Wang, Y. He, Xufei Mao, Y. Liu, Z. Huang, and X.Y. Li, "Exploiting

constructive interference for scalable flooding in wireless networks", In IEEE INFOCOM, 2012.

[20] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," In Proceedings of ACM SenSys, 2012.

[21] O. Landsiedel, F. Ferrari, and M. Zimmerling, "Poster abstract: Capture effect based communication primitives," In Proceedings of ACM SenSys, 2012.

[22] K. K. Chintalapudi and L. Venkatraman, "On the design of MAC protocols for low-latency hard real-time discrete control applications over 802.15.4 hardware", In Proceedings of ACM/IEEE IPSN, 2008.

[23] F. Ö sterlind, L. Mottola, T. Voigt, N. Tsiftes, and A. Dunkels, "Strawman: Resolving collisions in bursty low-power wireless networks," In Proceedings of ACM/IEEE IPSN, 2012.

[24] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," In Proceedings of IPSN, 2007.

[25] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," In Proceedings of NSDI, 2004.

[26] L. Mottola and G. P. Picco. MUSTER: Adaptive energy-aware multi-sink routing in wireless sensor networks. IEEE Transactions on Mobile Computing, 2011.

[27] A. Gonga, O. Landsiedel, and M. Johansson, "MobiSense: Power-efficient micro-mobility in wireless sensor networks," In Proceedings of DCOSS, 2011.

[28] S. Moeller et al, "Routing without routes: The backpressure collection protocol," In Proceedings of IPSN, 2010.

[29] J. W. Lee, B. Kusy, T. Azim, B. Shihada, and P. Levis, "Whirlpool routing for mobility," In Proceedings of MobiHoc 2010.

[30] T. Schoellhammer, B. Greenstein, and D. Estrin, "Hyper: A routing protocol to support mobile users of sensor networks," Technical report, UCLA, 2006.

[31] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," In Proceedings of INFOCOM 2002.

[32] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," In Proceedings of SenSys 2003.

[33] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," In Proceedings of Sensys 2004.

[34] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC

protocol for duty-cycled wireless sensor networks," In Proceedings of SenSys, 2006.

[35] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In Proceedings of SenSys, 2008.

[36] R. Musăloiu -E., C.-J. Liang, and A. Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In Proceedings of IPSN, 2008.

[37] P. Dutta, R. Musăloiu-E., I. Stoica, and A. Terzis, "Wireless ACK collisions not considered harmful," In Proceedings of HotNets, 2008.

[38] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. M. Liang, and A. Terzis, "Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," In Proceedings of ACM SenSys, 2010.

[39] Texas Instruments. MSP430F1611 datasheet, 2009.

[40] Texas Instruments. CC2420 datasheet, 2007.

[41] Tsung-Han Lin, I-Hei Ng, Seng-Yong Lau, Kuang-Ming Chen, and Polly Huang, " A microscopic examination of an RSSI-signature-based indoor localization system," in the workshop on HotEmNets, 2008.

[42] Chun-Chieh Hsiao, Yi-Jing Sung, Seng Yong Lau, Chia-Hui Chen, Fei-Hsiu Hsiao, Hao-hua Chu, and Polly Huang, "Towards long-term mobility tracking in NTU hospitals elder care center," in the workshop on SmartE, 2011, in conjection with PerCom, 2011.