Keio University Master's Thesis Academic Year 2011

Efficient Data Collection Protocol for Wireless Sensor Networks with Mobile Sink Nodes

Keio University Graduate School of Media and Governance

Kenji Yonekawa

Abstract

Efficient Data Collection Protocol for Wireless Sensor Networks with Mobile Sink Nodes

Summary

In recent years, Wireless Sensor Networks (WSNs) are becoming increasingly popular, and as a result, various kinds of WSN applications have been proposed. In the meantime, due to the spread of mobile devices, such as, PDAs and smartphones, users are now able to collect sensor data directly from WSNs. Since sensor nodes have severe constraint on resources, it is important to collect data efficiently. Agility in detecting link qualities is also essential, because qualities of links among mobile nodes change rapidly.

This thesis proposes CTDCP (Clustered-Tree Data Collection Protocol), a data collection protocol, and ALQDP (Agile Link Quality Detection Protocol), a link quality detection protocol. CTDCP leverages clustered-tree structure to improve both scalability and data collection efficiency. It further provides two key mechanisms: bypass tree, which minimizes the overhead of route construction and data collection, and broadcast-based routing, which reduces the number of data and acknowledgement packets. ALQDP employs linear regression to quickly detect link qualities, which enables mobile sink nodes to select most suitable paths upon the discovery of a new node.

We evaluate CTDCP and ALQDP in the real world environment and in simulation. The real world evaluation proves that, in comparison to Collection Tree Protocol and Four Bit Link Estimator, CTDCP and ALQDP 1) reduce energy consumption by approximately 35%, 2) decrease the path length and the number of retransmissions, and improve packet delivery ratio by about 2.5%, and 3) require roughly 2,200 bytes of RAM. Through simulation evaluation, in comparison to Sidewinder, the combination of CTDCP and ALQDP 1) reduces the number of transmissions per each source by approximately 43%, and 2) improves packet delivery ratio by about 3.6%. In spite of its large storage usage, the magnitude of improvement in packet delivery performance and overhead reduction is significant. Accordingly, it is proven that proposed protocols outperform existing protocols in terms of data collection efficiency. Our future work is maximizing the data collection performance by dynamically forming clusters, and improving the accuracy and flexibility of link quality estimation.

Keywords:

1 Wireless Sensor Networks 2 Data Collection Protocol 3 Link Quality Detection 4 Clustered Tree

Keio University Graduate School of Media and Governance
Kenji Yonekawa

修士論文要旨　　2011 年度 (平成 23 年度)

無線センサネットワークにおけるモバイルシンクノードのためのデータ収集プロトコル

論文要旨

近年，複数の無線センサノードを利用し，構築したネットワーク (WSN: Wireless Sensor Network) を用いるアプリケーションが提案，実装されている．また，PDA やスマートフォンなどのモバイルデバイスの普及により，ユーザが WSN から直接データを収集することが可能となっている．一方，センサノードの資源は厳しく制限されているため，データを効率的に収集することが重要な課題とされている．また，モビリティを持つノード間のリンクの品質は激しく変化するため，それを迅速に検知する必要がある．

本論文では，データ収集プロトコル，CTDCP (Clustered-Tree Data Collection Protocol)，及びリンク品質検知プロトコル，ALQDP(Agile Link Quality Detection Protocol) を提案する．CTDCP では，クラスタ化されたツリー構造を用いることで，高いデータ収集効率とスケーラビリティを達成している．さらに，CTDCP では 2 つのメカニズム，bypass tree とブロードキャストベースのルーティングを活用することで，ツリー構造のオーバーヘッド，及び送受信されるパケット数を削減している．ALQDP では機械学習を用いることで，迅速かつ高い精度でリンクの品質を検知し，モビリティを持つノードに最適なパスを提供している．

本論文では，実環境，及びシミュレーションでの評価を行った．実環境での評価を通じ，Collection Tree Protocol と Four Bit Link Estimator と比較し，CTDCP と ALQDP が，1) 電力消費量を約 35%削減，2) ホップ数，再送数の削減，並びにパケット配送率を約 2.5%向上，及び 3) 約 2,200 バイトの RAM を要することが実証された．また，シミュレーションの評価を通じ，Sidewidner と比較し，CTDCP と ALQDP が，1) 送信される総パケット数を約 43%削減，及び 2) パケット配送率を約 3.6%向上したことが証明された．評価結果から，提案手法がストレージを必要とする半面，オーバーヘッドの大幅な削減，及び高いパケット配送率を実現することが証明された．今後の展望として，最適なクラスタを動的に構成する手法の提案，並びにリンク品質検知の精度，柔軟性の向上を行う．

キーワード：
1 無線センサネットワーク　2 データ収集プロトコル　3 リンク品質検知プロトコル
4 クラスタツリー

慶應義塾大学大学院　政策・メディア研究科
米川 賢治

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, wireless sensor networks is becoming increasingly popular, and various kinds of applications have been proposed. In this chapter, we present the background, motivation and objective of the research, followed by the organization of the thesis.

1.1 Background

Recent advances in micro electro-mechanical system (MEMS) have lead to the miniaturization of sensor nodes. In addition, growth of wireless technologies provides low-energy, low-power wireless communication ability to the sensor nodes. As a result, many researchers have been proposing and implementing applications using wireless sensor networks (WSNs), which consist of hundreds of wireless sensor nodes.

Spread of wireless sensor nodes, PDAs, and smartphones allowed users to utilize them to collect sensor data directly from WSNs. Meanwhile, since wireless sensor nodes are designed to be small and cheap, they have severe constraint on resources, such as, CPU, memory, storage and energy source. Hence, various kinds of protocols have been proposed to efficiently collect sensor data from WSNs to sink nodes.

1.2 Motivation and Objective

This thesis proposes a data collection protocol which allows sink nodes to efficiently collect sensor data from WSNs. The proposed protocol aims to collect sensor data with high delivery ratio and small overhead. It is important to achieve high efficiency when collecting data, especially in WSNs, because sensor nodes have severe constraint on resources.

When collaborating mobile sink nodes with WSNs, efficiency and agility becomes additionally important. Although sensor nodes utilize low-energy, low-power radio chips, wireless communication consumes large amount of energy compared to other operations, such as calculation, sensing and storing data. Therefore, it is important to minimize the number of packets required to aggregate sensor data to sink nodes. The proposed protocol is designed to minimize not only the number of control packets, but also data packets by utilizing clustered-tree structure. In order to achieve high agility, we propose a link quality detection protocol which quickly detects the qualities of links among nodes.

This thesis describes the design and implementation of two protocols. One of them is a data collection protocol named Clustered-Tree Data Collection Protocol (CTDCP), which efficiently collects sensor data to multiple mobile sink nodes leveraging clustered-tree structure. Another is a link quality detection protocol named Agile Link Quality Detection Protocol (ALQDP), which quickly detects the qualities of links by utilizing estimation based on the history of data. The thesis evaluates those protocols in the real world environment and in simulation.

1.3 Organization

This thesis is organized as follows. In Chapter 2, we introduce the background of WSNs and mobility in sink nodes. We then present the importance of data collection in wireless sensor networks, and describe related work in Chapter 3. In Chapter 4, we present our approach of utilizing CTDCP, a data collection protocol, and ALQDP, a link quality detection protocol, and explain their design. Chapter 5 describes our implementation of CTDCP and ALQDP. We present the methodology and results of evaluation in Chapter 6. Finally, in Chapter 7, we conclude this thesis, and discuss our future work.

Chapter 2

Wireless Sensor Networks and Mobile Sink Nodes

In this Chapter, we discuss wireless sensor networks and the spread of mobile sink nodes. First, we discuss traditional WSNs consisting of static sensor nodes. We then explain the spread of mobile sink nodes, and applications utilizing them. Finally, we describe our target environment, issues, and requirements.

2.1 Wireless Sensor Networks

Due to the spread of wireless sensor nodes, researchers have been proposing various kinds of WSN applications [1] [2] [3] [4] [5] [6] [7] [8]. The majority of WSN applications assume static sensor/sink nodes: all sensing nodes and data collecting nodes are immobile. In recent years, collaboration of mobile sensor/sink nodes is drawing researchers' attentions. In this thesis, we target WSNs with static sensor nodes and mobile sink nodes.

2.2 Mobile Sink Nodes

In recent years, cooperation of mobile devices and WSNs has been attracting researchers' attentions. In this thesis, we focus on mobile sink nodes, and not on sensor nodes. Since sensor nodes are designed specifically for the purpose of sensing, we believe reduction in cost and size enables them to be deployed in various environments. In the meantime, use of mobile sink nodes has both advantages and disadvantages.

2.2.1 Spread of Mobile Sink Nodes

In the field of WSNs, there are two kinds of mobile sink nodes: passive, and active mobile sink nodes. We thoroughly discuss their characteristics as follows:

- Passive Mobile Sink Nodes
  Sensor nodes located close to the sink nodes have to forward a larger number of packets compared to those located far from the sink nodes. Therefore, sensor nodes consume energy unevenly, resulting in short network lifetime. Network lifetime is time until the *first node* exhausts its batteries; the idea of network lifetime is that, WSN is useless when at least one sensor node (especially the one that is forwarding the large number of packets) is no longer operational. Researchers have been arguing that static sink nodes are inappropriate in perspective of extending network lifetime. Passive mobile sink nodes move around in the area of WSNs, and collect sensor data. By doing so, the overhead for forwarding packets are thoroughly distributed among WSNs, hence, network lifetime is extended.

- Active Mobile Sink Nodes
  In this type of WSNs, sink nodes actively move in and out the coverage of WSNs. Wireless sensor nodes, PDAs and smartphones are usually used as sink nodes, which are equipped by humans, animals, robots, and vehicles. They do not travel for the sake of data muling, instead, they travel because it is their nature. Unlike traditional WSNs, mobile sink nodes do not acquire sensor data from the static sink node, instead, they collect sensor data directly from the WSNs. Researchers have been proposing to use smartphones as sink nodes in WSNs [9] [10]. Zhang et al. proposed uSD, an SD card which have capability to communicate with wireless sensor nodes over ZigBee [9]. This, in turn, shows that any kind of device with SD card slots can be mobile sensing/sink nodes. Park et al. proposed to collect sensor data directly from WSNs using smartphones [10]. Accordingly, the use of mobile devices as sink nodes is becoming increasingly popular.

We believe that in the near future, due to the spread of mobile devices, there will be a lot of WSN applications which assume the existence of mobile sink nodes. Therefore, in this thesis, we target **active** mobile sink nodes.

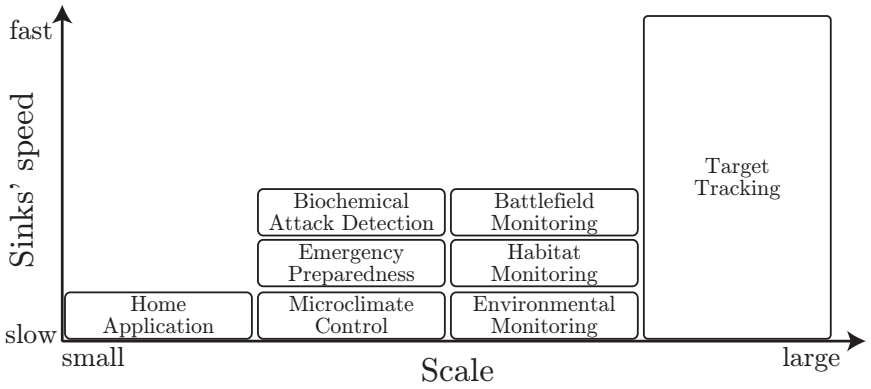


Figure 2.1: Scale vs. Sink Nodes' Speed in Various WSN Applications

2.2.2 Application Utilizing Mobile Sink Nodes

In this section, we introduce WSN applications which utilize mobile sink nodes. We discuss home application [1], biological/chemical attack detection [2], emergency preparedness [3], microclimate control [4], battlefield monitoring [5], habitat monitoring [6], environmental monitoring [7], and target tracking [8]. Fig. 2.1 illustrates the relationships between scales of WSNs and sink nodes' speeds in applications utilizing WSNs with mobile sink nodes.

Examples of home applications are smart room/home/building. In these applications, users collect sensor data from sensor nodes placed in the environment, and use context-aware or location-aware services. The majority of these applications are supplied in small-sized places, and they assume slow targets, such as, humans and robots, as sink nodes.

Biochemical attack detection, emergency preparedness and microclimate control suppose WSNs with building-size scale. Biochemical attack detection is utilized for detecting and alerting the biochemical attacks by terrorists, robber, etc. Emergency preparedness is used to suggest evacuation route in case of emergency. These applications are essential in saving human lives. Microclimate control is used to control dimming/air condition in rooms, buildings, etc. It is meaningful in providing users the optimal environment, while minimizing the waste of resources. They assume targets with various speed; biochemical attack detection and emergency preparedness usually suppose human moving fast, while microclimate control supposes humans and robots moving slowly.

Battlefield monitoring, habitat monitoring, environmental monitoring

and target tracking assume WSNs with field-size scale. Battlefield, habitat, and environmental monitoring are used to notify the users of movement of enemies, behaviors of animals, and environmental conditions, respectively. They have different targets of interest, however, their behavior is similar. Target tracking is used to locate humans, animals, robots, and vehicles. It assumes targets with various speeds, and it is one of the few applications which suppose targets moving fast.

Accordingly, most of current WSN applications with mobile sink nodes assume various scales, from home to field sized environment, and variety of targets, such as, humans, animals, robots, and vehicles. There are variety of scales of networks, although, only few of these applications suppose fast targets, such as, vehicles as mobile sink nodes. Consequently, the speed of targets is limited in the majority of applications. For these reasons, in this thesis, we assume various scales of WSNs, and the speed of humans/robots/animals as the speed of mobile sink nodes.

2.3 Target Wireless Sensor Networks

In this section, we first describe our target environment, followed by problems in that domain. We then discuss solutions and requirements for these problems.

2.3.1 Target Environment

There are multiple parameters to concern when leveraging WSNs. We thoroughly discuss 1) network type, 2) sensor nodes' density, 3) scalability, 4) sensor data and sink nodes, 5) sensor nodes' mobility, and 6) sink nodes' speed as follows:

1. Network Type
   There are three kinds of network types in WSNs: periodic, event-driven, and query-based. In periodic network, sensor nodes periodically sense data and transmit them toward sink nodes. In event-driven network, instead of reporting every data, each sensor node transmits packets only when it detects an event of interest within its range. In query-based network, sensor nodes transmit packets when they are requested. We argue that in the near future, current "one WSN for one application" style would be discarded, instead, the idea of "multi purpose WSNs" would be achieved. In multi purpose WSNs, multiple applications concurrently utilize a single WSN. It is considered that event-driven network is not suitable in multi purpose WSN, because

detection of a type of event cannot fulfill requirements of multiple applications. Therefore, we argue that multi purpose WSNs should be periodic or query-based networks. Query-based network is achieved by applications querying the network of multiple parameters (*e.g.* type of data, frequency of sensing, etc.). When numerous applications concurrently leverage a single network, it would be costly to manage queries, hence, we believe that periodic network would be dominant. For these reasons, this thesis targets periodic WSNs.

2. Sensor Nodes' Density
   The density of a WSN expresses the number of available links for each sensor node. This is greatly affected by the placement of sensor nodes; for example, sensor nodes placed along a road might have only one or two nodes to communicate with, meanwhile, those placed in a building might have multiple nodes to communicate with. In this thesis, we target WSNs with at least a couple of links for each sensor node. In order to achieve high data collection efficiency, MAC protocol, and data collection protocol should be utilized when the density of nodes is low, and high, respectively. Accordingly, we aim to achieve high data collection efficiency by leveraging data collection protocol.

3. Scale
   As described in Sec. 2.2.2, this thesis supposes various scales of WSNs: small as a room to large as a field. Note that scales of networks we are targeting are equivalent to those of our related work.

4. Sensor Data and Sink Nodes
   There are two kinds of WSNs regarding to sink nodes; a type of WSNs requires sensor data to be delivered to at least one sink node, and another requires all sensor data to be delivered to every sink node. The former type of WSNs can only be used if there is a central base station which collects and processes sensor data. Meanwhile, the latter type of WSNs can be utilized by any kind of applications, including networks which have multiple sink nodes operating different applications. The former type of WSNs has less flexibility, and does not suit in multi purpose WSNs. Therefore, this thesis targets WSNs which deliver all sensor data to every sink node.

5. Sensor Nodes' Mobility
   In traditional WSNs, the majority of applications only involve static sensor nodes. In recent years, due to the price reduction and the spread

of PDAs and smartphones, a lot of researchers have been proposing applications and middlewares for mobile sensor/sink/actuator devices. In this thesis, we assume WSNs with static sensor nodes and static/mobile sink nodes. The reasons we target this environment are threefold. First, the majority of WSN applications suppose static sensor nodes, and few applications, such as, participatory sensing, assume mobile sensor nodes. Second, the spread of mobile devices allows users to collect sensor data directly from WSNs, hence, we believe mobile sink nodes will become more and more popular. Finally, wireless sensor nodes are designed specifically for the purpose of sensing, hence, we argue that leveraging wireless sensor nodes as sensing device is more preferable.

6. Speed of Nodes
   In this thesis, we only assume mobility in sink nodes and not in sensor nodes, hence, we only discuss speed of sink nodes. Sinks nodes' speed differs depending on the targets they are attached to. We suppose sink nodes to be attached to humans, animals, and robots, therefore, we assume the maximum speed of targets as $10m/s$. This is reasonable, since they barely move that fast, and most of related work assume the same maximum speed for sink nodes.

In conclusion, in this thesis, we assume WSNs with 1) periodic type, 2) fair density, 3) various scales, 4) every sink node collect all sensor data, 5) static sensor/mobile sink nodes, and 6) maximum sink nodes' speed of $10m/s$.

2.3.2 Issues in Target Environment

In this section, we discuss issues arise in our target environment. Due to the recent advances in technologies, a lot of researchers target similar environment as ours. Although the existence of mobile sink nodes seems to be a small difference, movement of sink nodes induces high overhead in collision avoidance, location determination, and data collection.

- Collision Avoidance
  Sensor nodes use wireless communication to exchange information with others. In order to improve the delivery ratio of packets, it is important to avoid collision of packets, which is usually done in MAC protocols. Researchers have been proposing various kinds of MAC protocols for WSNs [11] [12] [13], which often are CSMA or TDMA-based protocols (in spite of its high efficiency, FDMA is usually difficult to

achieve in wireless sensor nodes due to the constraint of their radio chips). They have different strengths, such as anti-interference [11], channel utilization [12], energy efficiency [13], etc. When sensor nodes have mobility, they move in and out of other nodes' communication range, which often decreases performance of MAC protocols. In the meantime, numerous MAC protocols have been proposed in the area of WSNs, and minor modification of their parameters enables them to adapt to various kinds of environments. Consequently, we argue that rather than proposing another MAC protocol, the performance of collision avoidance should be improved by altering the parameters of existing protocols.

- Location Determination
  For some applications and protocols, locations of sensor nodes are an absolute requirement. In order to provide positions of sensor nodes, researchers have been proposing location protocols. Mobile sensor nodes freely move among the coverage of a WSN, therefore, the accuracy of location determination often decreases. In order to maintain high accuracy in determining sensor nodes' positions, either frequent location determination or future location prediction is essential. When a location protocol for a WSN with mobile nodes is achieved, the magnitude of improvement is assumed to be large, however, the number of applications depending on its information is limited.

- Data Collection
  Data collection is one of the most fundamental pieces in WSNs. Since almost every application utilizes data collected in WSNs, the efficiency of data collection is extremely important. Efficiency of data collection depends on the accuracy of choosing better path between sensor nodes and sink nodes. Frequent location updates of mobile nodes cause frequent path change, which increases the route management cost, and decreases data collection efficiency. Although researchers have been proposing data collection protocols for WSNs with mobile sink nodes, their data collection efficiency is still low.

Accordingly, the existence of mobile nodes in WSNs incurs problems in collision avoidance, location determination, and data collection. In this thesis, we choose to leverage data collection protocols to support WSNs with mobile sink nodes, because the magnitude of improvement is supposed to be large, and a lot of applications can appreciate it.

When considering data collection protocols for WSNs with mobile sink nodes, there are two problems: 1) low efficiency in collecting data, and 2) low agility in detecting link qualities. When mobile sink nodes exist in a WSN, route construction/management cost, as well as data delivery overhead increases, due to frequent route update and utilization of worse path. Link quality detection is important, because its information is utilized when choosing paths between the source and the sink node. Traditionally, link quality detection only assumed static nodes and it necessitates certain amount of time to detect the link qualities. In the meantime, link qualities between a mobile node and others change rapidly, hence, existing approaches have low accuracy and agility in detecting link qualities. In conclusion, there remains two problems in data collection protocols for WSNs with mobile sink nodes; low efficiency in collecting data, and low agility in detecting link qualities.

2.3.3 Requirements in Target Environment

In this section, we discuss the requirements for data collection protocols in WSNs with mobile sink nodes. Sec. 2.3.2 addressed two issues in data collection for WSNs with mobile sink nodes. This section describes the requirements for solving those problems: 1) improving data collection efficiency, and 2) increasing agility in detecting link qualities.

1. Improving Data Collection Efficiency
   Researchers have been measuring energy consumption of available operations in motes [14] [15]. Their experimental results prove that wireless communication requires considerably large amount of energy in comparison to other operations, such as activating CPU, writing/reading to flash memory, reading sensor values, and toggling LEDs. Consequently, in WSNs, it is important to improve data collection efficiently, since the amount of energy required for wireless communication is significant. Although existing data collection protocols aim to increase the data collection efficiency, a lot of them only assumes the existence of static nodes or a single mobile sink node. Therefore, there is a need to improve the data collection efficiency even when multiple mobile sink nodes exist in a WSN.

2. Increasing Link Quality Detection Agility
   Link quality detection is often required by data collection protocols; data collection protocols choose the most suitable path between the source and the sink node, referring to the information provided by link

quality detection protocols. Choosing the best path among available paths is meaningful, because it directly affects the efficiency of data collection. Meanwhile, link qualities between a mobile node and other nodes drastically change in a short period of time, therefore, agility in detecting link qualities has a significant role in selecting the best path. For instance, when link quality detection requires a long period of time, data collection protocols may leverage non optimal path for not knowing of better paths. In consequence, when considering the existence of mobile nodes in a WSN, link quality detection is desired to have high agility.

Accordingly, data collection protocols for WSNs with mobile sink nodes have two requirements: improving efficiency in data collection, and increasing agility in detecting link qualities.

2.4 Summary

In this Chapter, we discussed the relationships between WSNs and mobile sink nodes. Traditionally, most WSN applications only assumed static (immobile) sensor/sink nodes, however, due to the advances in technologies, a lot of recent WSN applications assume the existence of mobile sensor/sink nodes. This thesis assumes WSNs with 1) periodic type, 2) fair density, 3) various scales, 4) every sink node collect all sensor data, 5) static sensor/mobile sink nodes, and 6) maximum sink nodes' speed of $10m/s$. Our target environment has two problems: low efficiency in data collection, and low agility in detecting link qualities. In this thesis, we aim to solve these problems by improving data collection efficiency and link quality detection agility.

Chapter 3

Data Collection in Wireless Sensor Networks

In this Chapter, we discuss fundamental building blocks of WSNs: data collection and link quality detection protocols. We then introduce related work, and discuss their problems.

3.1 Data Collection Efficiency

Efficiency of data collection is extremely important, because it is related to resource usage at each sensor node, and qualities of applications. Due to the nature of wireless communication, packet collision is unavoidable. Therefore, there is a need to either avoid collisions, or complement lost of packets, which are usually solved in either MAC, data control or application layer. Fig. 3.1 illustrates the common layering of sensor node's operating system. In general, greater number of applications can be appreciated when improvements are made in lower layer. In this thesis, we aim to achieve high data collection efficiency utilizing data collection protocols for the reasons explained in Sec. 2.3.2. The performance of data collection protocols are depending on the accuracy of link quality detection protocols. We discuss their relationships in the following section.

3.2 Data Collection Protocols and Link Quality Detection Protocols

In WSNs, data collection protocol is one of the most fundamental protocols: it is a piece of middleware which enables applications to collect data effi-

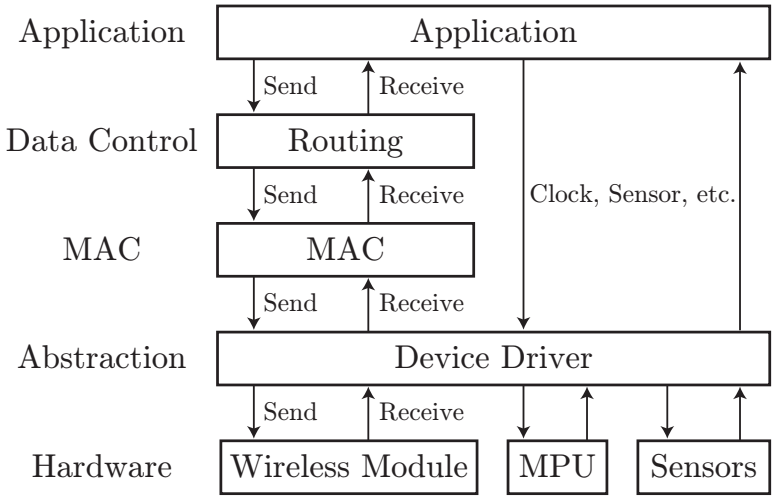


Figure 3.1: Brief Architecture of Sensor Nodes' Operating Systems

ciently. Data collection protocols are in charge of managing routes among sensor and sink nodes. Different data collection protocols have different policies in managing routes, which are usually based on link quality, distance, number of hops, and remaining energy. Data collection based on distance, number of hops, or remaining energy seems to be useful in conserving energy, however, we argue that the route decision should be based on link quality. Wireless sensor nodes usually have severe constraint on energy source, therefore, energy efficiency is extremely important. Although route decision based on distance, number of hops, and remaining energy seems to be meaningful, leveraging a link with bad quality induces high packet loss rate, which results in a large number of retransmissions. Consequently, the efficiency of data collection decreases while the energy consumption increases. For these reasons, we utilize link quality-based data collection.

The performance of a data collection protocol based on link quality is depending on the accuracy of link quality detection. Therefore, data collection protocols necessitate link quality detection protocols. Data collection protocols have three roles: 1) managing routes, 2) sending/forwarding packets, and 3) managing retransmissions. Meanwhile, link quality detection protocols are in charge of quantifying the qualities of links among sensor nodes. As described in Sec. 2.3.3, there are two requirements in our target environment: 1) improving data collection efficiency, and 2) increasing link quality detection agility. We achieve these requirements in data collection, and link quality detection protocols, respectively.

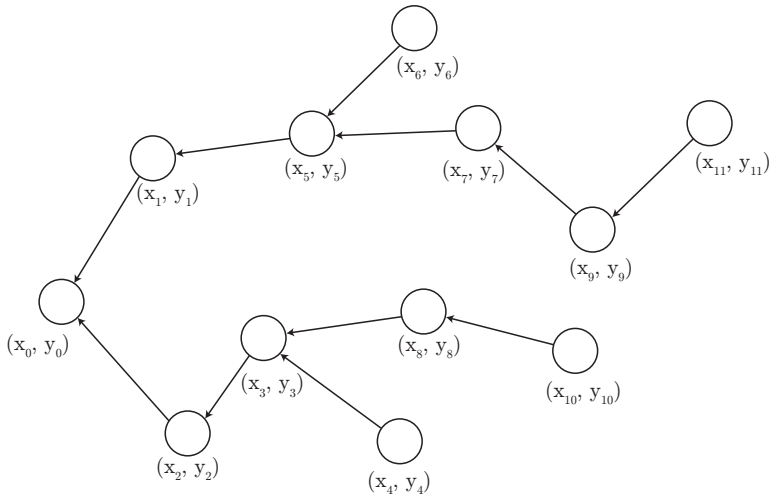


Figure 3.2: Geographic Data Collection Protocols

3.3 Data Collection Protocols

There are two kinds of data collection protocols for WSNs with mobile sink nodes; location-based, and non-location-based protocols. In this section, we introduce state of the art data collection protocols, while analyzing and discussing their problems.

3.3.1 Location-Based Data Collection Protocols

Location-based data collection protocols utilize location of each sensor node in the WSN. They can be divided into two types: geographic and grid-based.

- Geographic Data Collection Protocols
  Fig. 3.2 illustrates how geographic data collection protocols work. In geographic data collection protocols, each sensor node calculates the direction and/or the distance toward neighbor nodes, and forwards packets based on it [16] [17].

  Lee et al. proposed Data Stashing, which keeps data at sensor nodes near the sink nodes [16]. They argue that, relying on a single sensor node to deliver packets to a mobile sink node is inappropriate in perspective of data collection efficiency. In Data Stashing, packets are aggregated not only to a single forwarding node, but also to its neighbor nodes. Neighbor nodes do not forward packets to the sink node immediately, instead, they keep those packets until they have connec-

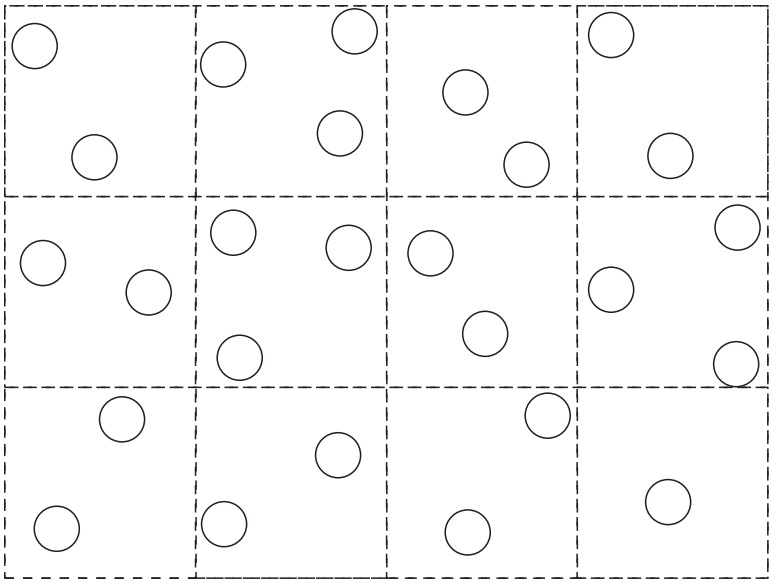


Figure 3.3: Grid-Based Data Collection Protocols

tivity to the sink node. Data Stashing achieves high packet delivery ratio and low latency, however, it requires large amount of energy and storage.

Keally et al. proposed Sidewinder, which utilizes aggressive estimation of sink nodes' location [17]. It leverages 60° rule introduced in BLR [18] to suppress the redundant forwarding. They argue that the use of forwarding table and unicast-based routing has high overhead when sink nodes have mobility. In sidewinder, sensor nodes which received a packet compete for the next hop forwarding. Although it achieves high delivery ratio and low latency, its calculation and retransmission overhead is significant.

- Grid-Based Data Collection Protocols
  Fig. 3.3 shows how grid-based data collection protocols work. In grid-based data collection protocols, sensor nodes are divided into grid, and packets are forwarded utilizing adjacent grid points [19] [20].

  Luo et al. proposed Two-Tier Data Dissemination (TTDD) [19]. They argue that traditional grid-based data collection protocols have low data collection efficiency because of reactive grid construction. In TTDD, sensor nodes proactively create grid structure to improve the data collection efficiency. After forming the grid structure, packets are

forwarded leveraging two-tiers; lower tier is within a grid, and higher tier is among grids. By utilizing two-tier structure, TTDD achieves high delivery ratio and low route decision overhead, however, its periodic grid construction induces high energy consumption and latency.

Wang et al. proposed Dual-Tree-based Data Aggregation (DTDA), which utilizes bidirectional paths between sender and sink nodes [20]. They argue that it is not preferable to rely on a single path among the source and the sink node. In DTDA, each sensor node establishes two paths toward sink nodes, utilizing grid structures. By employing bidirectional paths, it achieves high delivery ratio, however, its energy consumption is significant.

A lot of location-based data collection protocols accomplish high delivery ratio, however, researchers favor non-location-based protocols for two reasons as follows. First, location-based protocols require each sensor node's location information. In WSNs with mobile sink nodes, localization protocols need to determine positions of sensor nodes frequently, which incurs high overhead. Second, the distance between two nodes does not necessarily indicate the quality of the link. In consequence, location-based protocols have less chance of selecting the optimal path.

3.3.2 Non-Location-Based Data Collection Protocols

Non-location-based data collection protocols do not utilize location information, instead, they use qualities of links among sensor nodes. They can be classified into two types: cluster and tree-based.

- Cluster-Based Data Collection Protocols
  Fig. 3.4 indicates data collection protocols leveraging a cluster structure. Sensor nodes in cluster-based data collection protocols create clusters with nearby nodes, and select a cluster head among them. Each cluster head collects packets from sensor nodes in its own cluster, and communicate with other cluster heads. Researchers have been proving that cluster-based data collection protocols accomplish high scalability compared to other types of data collection protocols [21] [22].

  Younis et al. proposed Hybrid Energy-Efficient Distributed clustering (HEED) [21], which periodically changes cluster heads for load distribution. Due to the nature of cluster-based data collection protocols, cluster heads have to forward larger number of packets compared to

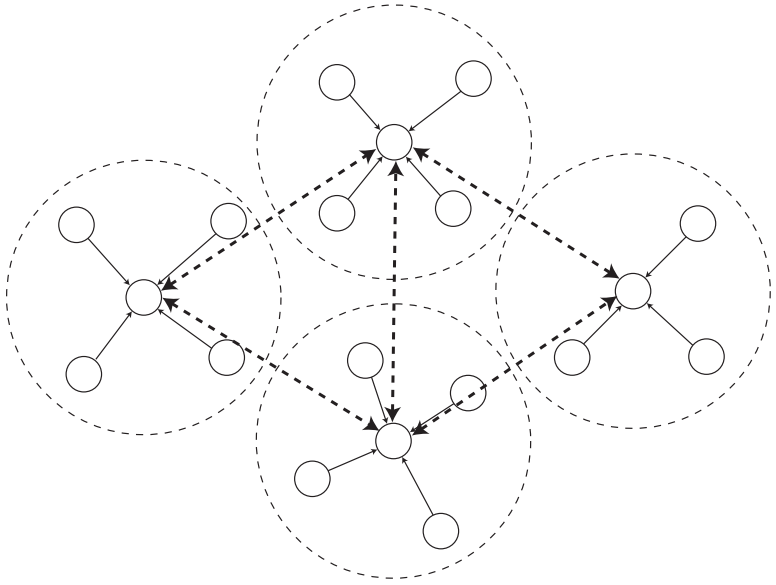


Figure 3.4: Cluster Data Collection Protocols

cluster members. DEED employs quick cluster head selection algorithm, which completes in $O(1)$. Although it successfully distributes overhead among all nodes, its data collection efficiency is low.

Cugola et al. proposed Context and Content-Based Routing (CCBR) [22], an extension of CBR [23]. One of its key features is content-based routing, in which, each sink node specifies the data of interest, and packets are forwarded according to it. In addition, CCBR adopts context-based routing, which filters packets according to the context. In spite of its high flexibility, CCBR has high management overhead.

- Tree-Based Data Collection Protocols
  Fig. 3.5 shows how tree-based data collection protocols work. In these protocols, sensor nodes form a tree structure, and aggregate packets toward the root of the structure. It has been proven that tree structure has decent scalability and high data collection efficiency [24] [25].

  Gnawali et al. proposed Collection Tree Protocol (CTP) [24], which adopts adaptive beaconing to minimize both overhead and latency. CTP is not designed for WSNs with mobile sink nodes, however, it demonstrates the data collection efficiency of tree structure. Adaptive beaconing is achieved by employing the idea of Trickle [26], which

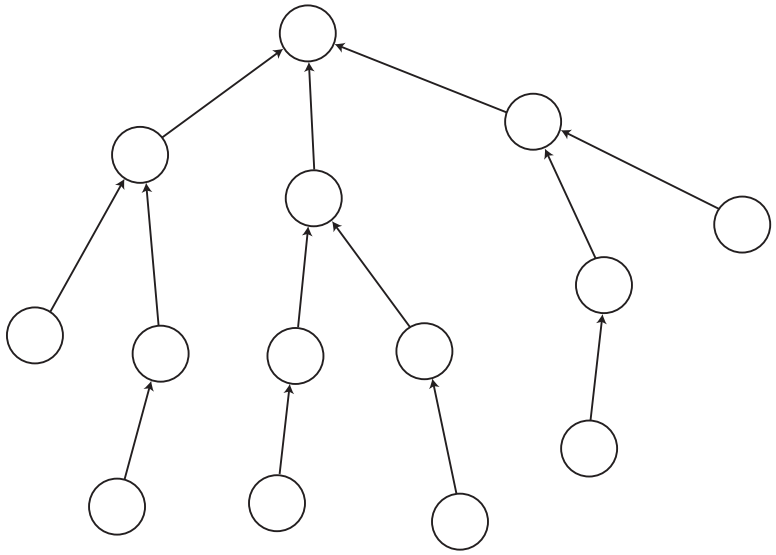


Figure 3.5: Tree Routing Protocols

doubles the waiting time until an event of interest occurs, and sets the waiting time to minimal once it happens. CTP is evaluated in a real world environment, and proved to have high delivery ratio, as well as data collection efficiency.

Förster et al. proposed Feedback ROuting to Multiple Sinks (FROMS) [25], which leverages multicast and reinforcement learning to improve data collection efficiency. FROMS utilizes multicast to minimize the number of packet transmissions when delivering packets to multiple destinations. In addition, FROMS employs reinforcement learning to improve data collection efficiency and latency. Through the evaluation, FROMS is proved to have high data collection efficiency. However, high overhead is incurred for utilizing multicast and heavy-weight learning algorithm. Moreover, tree structure is constructed from each sink node, and it is updated whenever the sink node moves, which induces large management cost.

Non-location-based data collection protocols are favorable in comparison to location-based protocols in perspective 1) not requiring location information, 2) achieving high data collection efficiency, and 3) involving small calculation overhead. Cluster and tree-based data collection protocols have different strengths and weaknesses: cluster-based protocols have high scalability and low data collection efficiency, while tree-based protocols have

low scalability and high data collection efficiency. Tree-based data collection protocols satisfy the requirements of our target environment, however, they have low scalability, and high overhead for constructing/managing tree structures.

3.4 Link Quality Detection Protocols

Accuracy of link quality detection protocols have significant impacts on data collection efficiency, because the information is utilized in managing routes. For instance, De Couto et al. deployed a network in the real world environment, and proved that poor accuracy of link quality detection can induce 200% or greater degradation of network throughput [27].

3.4.1 Analysis of Link Quality Detection Protocols

Traditional link quality detection protocols mainly have two problems: 1) utilizing only a few parameters, and 2) depending on hardware/layer specific information.

- Link Quality Parameters
  Link quality parameters can be divided into two types: hardware and software-based parameters.
  - Hardware-based parameters
  Three kinds of parameters can be acquired through hardwares: Link Quality Indicator (LQI), Received Signal Strength Indicator (RSSI), and Signal to Noise Ratio (SNR). These parameters are acquired from radio transceivers. LQI represents the quality of an incoming packet, which is defined by IEEE 802.15.4 standard. RSSI shows the signal strength of an incoming packet. SNR indicates the ratio of signal strength to noise of a received packet. Hardware-based parameters can be acquired quickly and inexpensively, however, they have four problems as follows. First, some parameters may be unavailable in certain platforms and/or hardwares. Second, these parameters are updated only when a packet is successfully received, therefore, they cannot be leveraged when packets are not received. Third, these metrics are derived using only a part of a packet, hence, their accuracies are low. Finally, utilization of only hardware-based parameters have been proven to have low accuracy in detecting link qualities [27] [28].
  - Software-based parameters
  Three kinds of link quality parameters can be acquired in software approach: Packet Reception Ratio (PRR), Required Number of Packet

retransmissions (RNP), and Expected Transmission count (ETX). These metrics are computed according to the number of transmitted/received packets. PRR represents the ratio of the number of successfully received packets to that of the transmitted packets. RNP indicates the number of retransmissions required for one successful delivery [29]. ETX represents required number of packet transmissions for one successful delivery [27]. These parameters leverage link layer information, therefore, any kind of platforms can utilize them, and they have high accuracies in detecting link qualities. However, calculation and certain number of packets are required to derive them, which results in high overhead and low agility.

- Dependency on Other Hardwares/Layers/Protocols
  In general, protocols are desired to be independent from other hardwares, layers, and protocols. This is because, cross-layer design has issues in semantics, dependencies, and design/implementation simplicity. First, semantics of the entire system is fairly important, because stability and dependability of a system is crucial. Second, when a protocol have high dependency on hardware, layers and/or protocols, there is a great chance that it works only on specific platform. Finally, simple design and implementation is essential in understanding and implementing on different platforms. In spite of its importance, a lot of existing protocols are tightly coupled with certain layers/protocols [30] [31]. MultihopLQI [30] and MintRoute [31] are some of the most well known data collection protocols in TinyOS [32], an open source, component-based operating system specialized for embedded networked sensor devices. They are depending on specific pieces of hardwares and/or layers, therefore, they cannot be utilized in the other platforms. Accordingly, when detecting link qualities among among sensor nodes, it is important for a protocol to be independent from other hardwares, layers, and/or protocols.

When detecting the link qualities among sensor nodes, it is important to choose reliable and accurate parameters, while eliminating the dependency on certain hardwares/layers/protocols.

3.4.2 Related Work

In recent years, researchers have been proposing link quality detection protocols following the idea described in Sec. 3.4.1 [27] [28] [33] [34]. We describe their characteristics, and discuss their strengths and weaknesses as follows.

Fonseca et al. proposed Four Bit Link Estimator (4B) [28], which utilizes four pieces of information; LQI, ETX, pin and compare bit. They analyzed the validity of LQI as an indicator of link quality, and concluded that it is inexpensive, and has low accuracy and reliability. ETX is utilized with Exponentially Weighted Moving Average (EWMA) to take the history of data into consideration. It has been proven that data collection protocols failed due to the inconsistencies in the neighbor tables of data collection and link quality detection layers [35]. In order to solve this problem, 4B provides pin and compare bit: pin is used to keep the specified neighbor in the table, and compare is used to query the network layer of link quality. Although 4B provides high accuracy in detecting link qualities, it only leverages ETX as a fine grained link quality indicator, hence, it lacks agility.

Baccour et al. proposed Fuzzy-link Quality Estimator (F-LQE) [33], which utilizes fuzzy logic for both input and output parameters. In order to improve the accuracy of link quality detection, they use four properties: packet delivery, asymmetry, stability, and channel quality. Packet delivery is expressed in the form of PRR applied with WMEWMA [36], which smoothens the curve and provides the estimation of the PRR. Asymmetry indicates the difference between link quality of uplink and downlink, because when considering acknowledgements of packets, not only the quality of uplink, but also that of downlink is important. The authors argue that high variability in link quality is not desirable, and provide stability parameter, which represents the variance of link qualities. Finally, F-LQE uses SNR as channel quality. F-LQE provides better granularity of link qualities compared to 4B, however, it lacks agility and mobile nodes support, because stability parameter has low affinity with mobility.

Liu et al. proposed Foresee (4C) [34], which leverages link quality prediction. The authors collected data in the real world environment using Tmote Sky motes, and use them to clarify appropriate machine learning algorithm in predicting link qualities. They evaluated Bayes classifier, logistic regression, and artificial neural networks, and concluded that logistic regression achieves highest accuracy. 4C leverages PRR and LQI as input parameters, and outputs the predicted link qualities. According to their evaluation results, 4C predicts link quality with mean square error of approximately 0.2. Despite 4C provides high accuracy in predicting link qualities, it has large overhead for leveraging costly machine learning algorithm, and requires a large number of data before starting prediction.

In data collection protocols, the accuracy of link quality detection is fairly important, because their performance highly depends on it. Researchers have been proposing various kinds of link quality detection protocols. How-

ever, they would not perform well in WSNs with mobile sink nodes, because they lack agility in detecting link qualities. In consequence, agile link quality detection protocol is desired.

3.5 Summary

In this Chapter, we discuss data collection protocols and link quality detection protocols, which are leveraged to fulfill the requirements of WSNs with mobile sink nodes. Researchers have been proposing various kinds of data collection protocols, and data collection protocols based on link quality are said to be superior to those based on location information. Non-location-based data collection protocols can be classified into cluster and tree-based protocols, which have issues in data collection efficiency, and scalability, respectively. Existing link quality detection protocols achieve high accuracy, however, they significantly lack agility. Therefore, agile link quality detection protocol is required.

Chapter 4

Approach and Design

In this Chapter, we first propose our approach of combining cluster and tree structures to supplement their disadvantages. Second, we propose Clustered-Tree Data Collection Protocol (CTDCP), a data collection protocol which provides high data collection efficiency. We then propose Agile Link Quality Detection Protocol (ALQDP), a link quality detection protocol which provides high agility and accuracy in detecting link qualities.

4.1 Approach

This thesis proposes Clustered-Tree Data Collection Protocol (CTDCP), a data collection protocol, and Agile Link Quality Detection Protocol (ALQDP), a link quality detection protocol. Each of them achieves high efficiency in collecting data, and high agility in detecting link qualities, respectively.

CTDCP utilizes clustered-tree structure to improve the data collection efficiency and scalability. It constructs tree structures among the static nodes, and let mobile sink nodes use the structures to minimize the route construction cost, as well as data delivery cost. CTDCP includes two mechanisms, **bypass tree** and **broadcast-based routing** to further improve its data collection efficiency. Bypass tree enables sensor nodes to bypass existing tree structures, and broadcast-based routing enables sensor nodes deliver packets to multiple destinations with a single transmission.

ALQDP utilizes **link quality estimation** along with link quality detection. It leverages both software and hardware parameters to achieve high accuracy and agility in detecting link qualities. Link quality estimation is the key to instantly estimate the link qualities among sensor nodes.

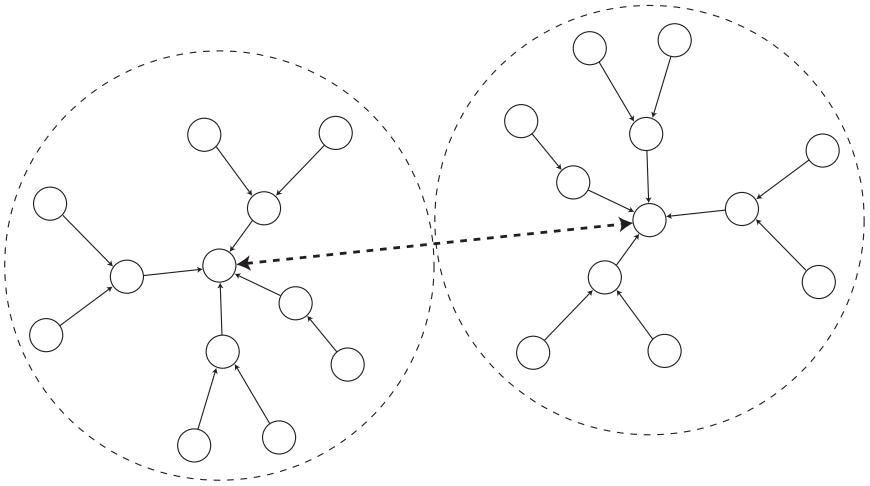


Figure 4.1: Clustered-Tree Structure

4.2 Design of CTDCP

In this section, we discuss two fundamental components of CTDCP: basic routing, and mobile sink management. We propose **clustered-tree structure**, **bypass tree**, and **broadcast-based routing**, which are leveraged to minimize route construction cost and high data collection efficiency.

4.2.1 Clustered-Tree Structure

Clustered-tree structure is leveraged to achieve high data collection efficiency and scalability simultaneously. Fig. 4.1 briefly shows the clustered-tree structure. Circles indicate sensor nodes, and thin dashed-circle indicates clusters. When sink nodes have mobility, they cannot fully appreciate high scalability that cluster structures provide, because inter-cluster communication increases as sinks nodes enter different clusters' coverage. In order to solve this problem, we extend the coverage of each cluster, and construct tree structure in it. The overhead for modifying path in a single tree structure is considerably low compared to switching among different clusters. Therefore, we argue that by utilizing clustered-tree structure, mobile sink nodes can appreciate both data collection efficiency and scalability.

Existing tree-based data collection protocols construct tree structure from each sink node. Fig. 4.2 illustrates how sink nodes create tree structures in existing protocols. Circles indicate statically placed sensor nodes, and numbers in them represent their sensor node ID. Whenever a sink

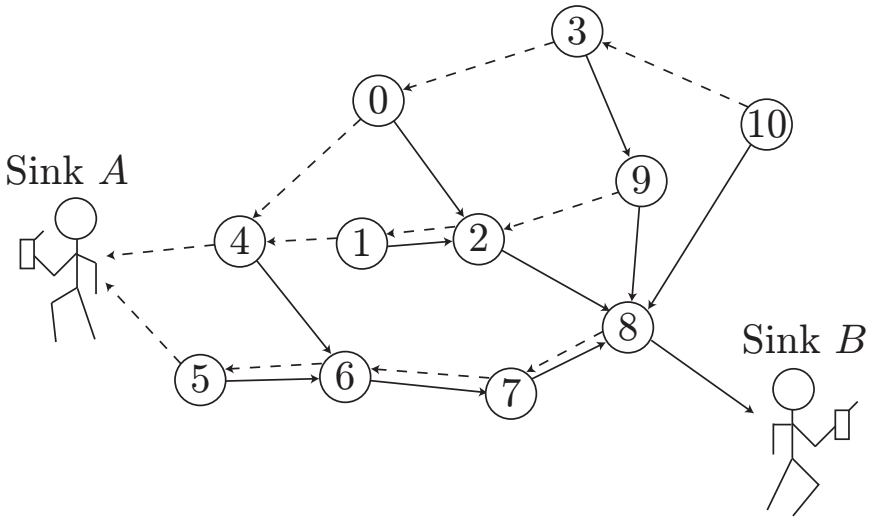
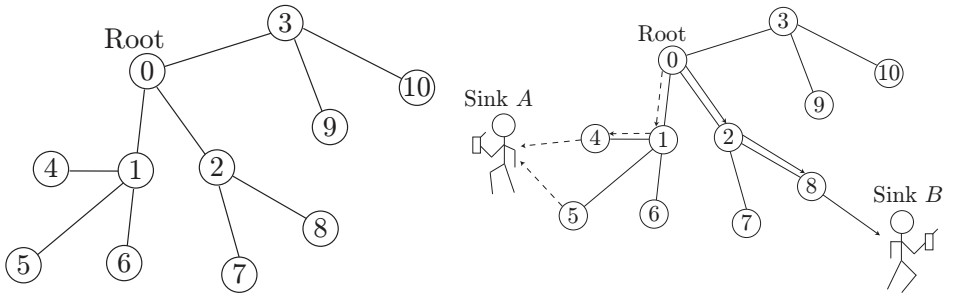


Figure 4.2: Existing Tree-Based Data Collection Protocols

node moves, it reconstructs the tree structure leveraging the whole network, hence, when a sink node has mobility, high overhead is induced not only at sink nodes, but also at static nodes. Accordingly, current tree-based data collection protocols have high delivery ratio for leveraging tree structures, however their overhead for constructing and managing the tree structure is significant.

Unlike existing tree-based data collection protocols, in CTDCP, sink nodes utilize tree structures formed by static sensor nodes. This approach suits our target environment, where sensor nodes are static and sink nodes are mobile. Fig. 4.3 illustrates the two processes of CTDCP. At first, CTDCP creates a tree structure with static sensor nodes (Fig. 4.3(a)), then, mobile sink nodes collect data utilizing that tree structure (Fig. 4.3(b)).

When leveraging clustered-tree structure, we have to consider the number of nodes involved in each cluster. For instance, small cluster causes high overhead when switching among different clusters, while large cluster results in low scalability because upper layered nodes have to forward a large number of packets. Researchers have been addressing the limitations of using tree structure for data collection [24] [37] [38]. The authors of CTP applied a tree-based data collection protocol to multiple testbeds, and confirmed that it achieves delivery ratio of at least 90% [24]. They evaluated CTP on various platforms with the number of nodes varying from 20 to 310, and confirmed that the combination of the large number of nodes and short inter packet interval results in poor data collection efficiency. Lin et al. discussed



(a) Constructing tree structure using static nodes

(b) Sink nodes collect data utilizing existing tree structure

Figure 4.3: Data Collection Process of CTDCP

the relationships between the combination of the number of sensor nodes and cluster size, and overhead of data collection and network lifetime [37]. They used simulation to prove the appropriate size and depth of tree structure, and confirmed that tree structures with large depth do not perform well. Koubaa et al. analyzed tree-based data collection protocols in terms of the number of children nodes and the depth of each node [38]. They examined the limitation on the number of packet transmission per given time interval by referring to beacon frames in IEEE 802.15.4. These work discussed the limitations of utilizing tree structure for data collection. CTP uses testbeds, however, the authors have not examined the relationships between the inter packet interval and the number of sink nodes. Therefore, appropriate cluster size remains unclear.

We argue that cluster size should be decided based on multiple parameters, such as, the number of source/sink nodes, inter packet intervals, noise level, etc. Meanwhile, these parameters are not deterministic until WSNs and applications are deployed. Therefore, we provide a tunable parameter, the maximum number of nodes per cluster.

4.2.2 Mobile Sink Management

The approach described in Sec. 4.2.1 reduces the overhead of constructing and managing tree structure from each sink node. Meanwhile, it produces another problem; optimal path between the source and the sink node may not be leveraged. Table 4.1 shows the sum and the average path length for each sink node in existing protocols (Fig.4.2) and CTDCP (Fig.4.3(b)). Path length is defined as the number of hops between the source and the destination node. As the number of hops increases, the chance of packet

Table 4.1: Path Length in Existing Data Collection Protocols and CTDCP

Protocols	Sink ID	Sum of Path Length	Average Path Length
Existing Protocols	Sink *A*	29	2.636
	Sink *B*	29	2.636
CTDCP	Sink *A*	38	3.455
	Sink *B*	42	3.818

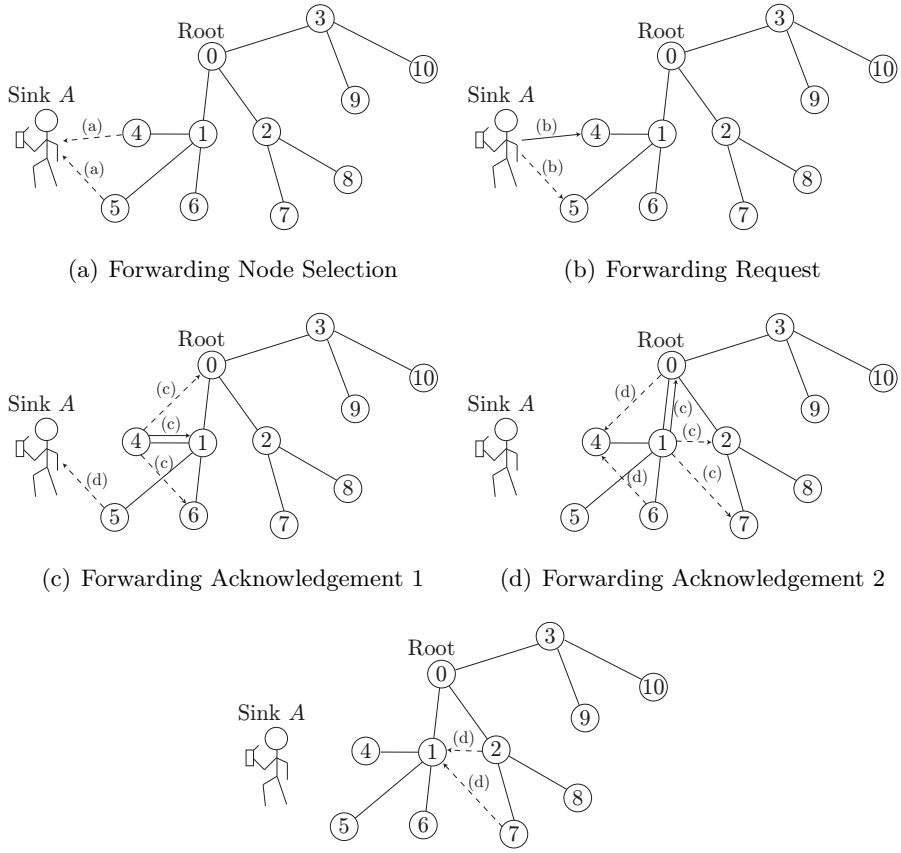
loss also increases, therefore, the path length is desired to be as short as possible. As shown in the table, delivery cost of CTDCP is considerably high in comparison to the existing tree structure. This, in turn, indicates that high data collection efficiency cannot be appreciated by simply utilizing existing tree structure. In order to solve this problem, we propose **bypass tree**.

Bypass tree enables packets to be delivered without passing through unnecessary paths. For example, in Fig. 4.3(b), node 7 has a path length of 5 toward the sink node *A* when simply using existing tree structure. Meanwhile, when node 7 bypasses the existing tree structure and sends packets toward node 1, the path length reduces from 5 to 3. Consequently, when bypass tree is fully applied, the path length would decrease. When the availability of bypassing is examined over the whole network, route construction cost becomes large. Sensor nodes located closer to the sink node usually have greater chance of decreasing the path length by bypassing other nodes. Therefore, we apply bypass tree only from the sink node up to the root node of the network.

Bypass tree is formed using four procedures, a) sink node selects a node to forward packets, b) sink node requests the selected node to forward packets, c) child node notifies its parent about forwarding packets, and d) neighbor nodes check the effectiveness of bypassing and notify others if required. We define them as "Forwarding Node Selection", "Forwarding Request", "Forwarding Acknowledgement", and "Bypass Request", respectively. Fig. 4.4 illustrates these procedures, where each alphabet represents each procedure.

(a) Forwarding Node Selection

In this procedure, sink nodes select the forwarding node among neighbor nodes. The Forwarding Node Selection is achieved by leveraging the information provided by link quality detection protocols. Fig. 4.4(a)



(a) Forwarding Node Selection

(b) Forwarding Request

(c) Forwarding Acknowledgement 1

(d) Forwarding Acknowledgement 2

(e) Bypass Request

Figure 4.4: Behavior of Bypass Tree

depicts how sink node A receives packets from neighbor nodes, and selects node 4 as the forwarding node. After selecting the forwarding node, the sink node leverages Forwarding Request to notify the selected node of it.

(b) Forwarding Request
In this procedure, the sink node sends a request to the selected forwarding node. Fig. 4.4(b) illustrates how sink node A requests the neighbor nodes to forward packets. Request message is broadcasted, and the packet includes the ID of the specified forwarding node (in case of the figure, the sink node includes the ID of node 4 in the request packet). Neighbor nodes, node 4 and 5, process the received packets in accordance to their role: node 4 executes Forwarding Acknowledgement, and

node 5 determines the effectiveness of bypassing. We leverage broadcast for two reasons; first, neighbor nodes which received the packet should check the effectiveness of bypassing, and second, when the sink node decides to change the forwarding node, it may include a request to stop forwarding packets in a single request.

(c) Forwarding Acknowledgement
Forwarding Acknowledgement is transmitted from the forwarding node up to the root node through the designated path. Fig. 4.4(c) and Fig. 4.4(d) illustrates first and second Forwarding Acknowledgments, respectively. Note that Forwarding Acknowledgments are broadcasted like Forwarding Request, so that neighbor nodes can check the effectiveness of bypassing. Neighbor nodes process the Forwarding Acknowledgement in the same manner as the Forwarding Request. At first Forwarding Acknowledgment, node 4 acknowledges its parent (node 1) of forwarding packets, and node 0 and 6 checks the effectiveness of bypassing. Meanwhile, after the second Forwarding Acknowledgment, node 0 does not acknowledge its neighbor nodes about forwarding packets, because it is the root of the network.

(d) Bypass Request
Sensor nodes which received Forwarding Request or Forwarding Acknowledgement check their neighbor table, and determine if they should bypass their parents or not. Fig. 4.4(c), 4.4(d), and 4.4(e) show how sensor nodes utilize Bypass Request. Effectiveness of bypassing is evaluated by comparing the quality of two paths: *itself* $\rightarrow$ *receiver* $\rightarrow$ *sender*, and *itself* $\rightarrow$ *sender*. When a node finds out bypassing is beneficial, it transmits a request packet to notify both sender and receiver nodes.

We define the process of selecting and requesting to start forwarding packets as **Attach**. On the other hand, the process of notifying to stop forwarding packets is defined as **Detach**. When the first forwarding node failed to receive detach packet transmitted by the sink node, useless routing information remains in each node up to the root. Therefore, the first forwarding node should check the reachability of the sink node, and transmits detach packet if necessary.

Fig. 4.5 illustrates how packets are forwarded after bypass tree is applied to CTDCP. Table 4.2 shows the sum and the average path length of existing tree-based protocols (Fig. 4.2) and CTDCP with bypass tree (Fig. 4.5). In comparison to existing tree-based approach, CTDCP leveraging bypass tree has slightly higher overhead in data collection, however, it drastically

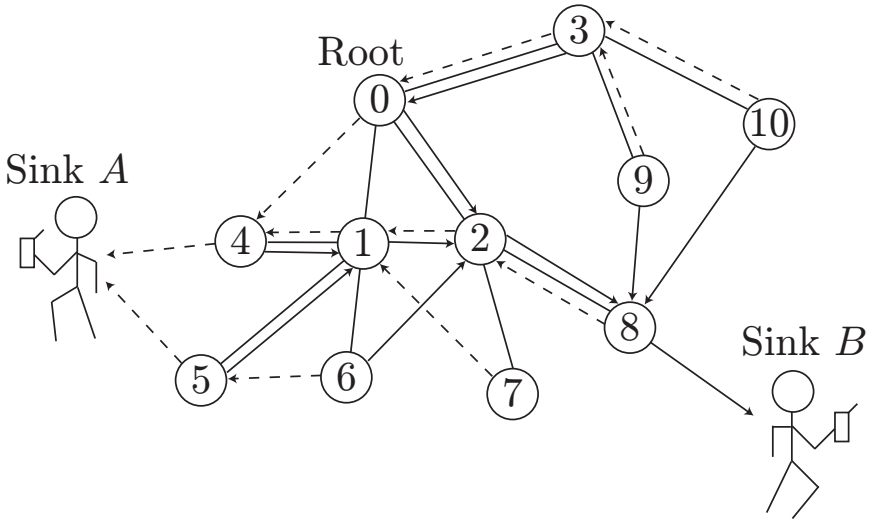


Figure 4.5: CTDCP Leveraging Bypass Tree

Table 4.2: Path Length in Existing Data Collection Protocols and CTDCP with Bypass Tree

Protocols	Sink ID	Sum of Path Length	Average Path Length
Existing Protocols	Sink *A*	29	2.636
	Sink *B*	29	2.636
CTDCP with Bypass Tree	Sink *A*	29	2.636
	Sink *B*	30	2.727

Table 4.3: Comparison of the Number of Transmissions in Unicast/Multicast/Broadcast

Type of Transmission	Total Number of Transmissions	Average Number of Transmissions
Unicast	55	5.000
Multicast	45	4.091
Broadcast	45	4.091

reduces the overhead of route construction/management. In existing tree protocols, each sink node requires the *whole* network to construct the best routes. On the other hand, in CTDCP with bypass tree, paths among sensor nodes are determined only up to the root of the tree structure, therefore, it reduces the overhead, as well as route construction time. Note that tree construction cost can become lower in CTDCP when sink nodes choose upper layered nodes (root or inner nodes in the tree structure) as forwarding nodes. For example, when a sink node selects node 0 as forwarding node, Forwarding Acknowledgement and Bypass Request is not required, and it can fully leverage the high efficiency paths.

CTDCP utilizes broadcast when transmitting data packets. Existing tree-based data collection protocols use either unicast [24] or multicast [25]. We argue that unicast is not an option when all data have to be delivered to multiple sink nodes, because it requires a large number of transmissions. Table 4.3 indicates the total and the average number of transmissions required in the WSN depicted in Fig. 4.5 when utilizing unicast, multicast, and broadcast. As shown in the table, unicast incurs a large number of transmissions in comparison to multicast and broadcast. When a large number of packets are being exchanged, possibility of packet losses increases, which causes unicast-based protocols to perform additionally worse. In addition, utilization of ACK or NAK incurs extra packets to be transmitted, which further worsens unicast-based protocols. Multicast and broadcast-based protocols require the same number of packet transmissions, however, multicast incurs higher overhead in each packet exchange. When leveraging multicast, each packet has to contain multiple destination addresses and additional length field, while broadcast can simply set broadcast address to the destination field. It has been proven that transmitting/receiving packets with larger payload require more energy [14] [15], hence, broadcast is superior to multicast in terms of energy efficiency.

When leveraging unicast or multicast, routing management is usually established at sender nodes, because the destinations of transmitting packets are specified by them. On the other hand, when utilizing broadcast, routing should be managed at the receiver nodes, because sender nodes specify broadcast address as the destination of transmitting packets. In consequence, CTDCP utilizes receiver-based routing management. Due to the nature of wireless communication, transmitted packets are received not only by the destination node, but also by the neighbor nodes. In the majority of wireless sensor nodes, destination of a packet is verified in the abstraction or the MAC layer; if the local node is the destination of the packet, it acknowledges the overlying layer, and if not, it discards the packet. In the meantime, when a packet is broadcasted, it is always acknowledged to the overlying layer. Generally, higher overhead is induced when packets are processed at upper layers. However, the overhead induced for processing packets in upper layers is considerably small compared to transmitting multiple packets, or transmitting packets with larger payload. Consequently, we argue that receiver-based routing have lower overhead in comparison to sender-based routing.

When leveraging broadcast-based routing, it is difficult to manage acknowledgements of reception of packets. One possible way to enable acknowledgements is to insert destination nodes' addresses in packets (which is the basic idea of multicast), however, it induces high overhead in each transmitting packet. Therefore, we propose an alternative approach to examine the delivery status of packets: we utilize packets forwarded by the parent node as acknowledgements. We show its brief behavior by taking the WSN in Fig. 4.5 as an example. Whenever node 10 transmits a packet, CTDCP enqueues that packet to its sending queue. When node 3 and 8 (parent of node 10) receive that packet, they forward it toward their parents utilizing broadcast. Upon the reception of packets forwarded by node 3 and 8, node 10 removes the corresponding packet from its sending queue. If node 3 or 8 did not receive the packet transmitted by node 10, or if node 10 did not receive packets forwarded by node 3 and 8, the packet remains in the sending queue, hence, it will be retransmitted.

In general, acknowledgements of reception of packets are achieved with ACK or NAK. When ACK is used, the receiver node replies an acknowledgement when it successfully receives a packet, meanwhile, when NAK is utilized, the receiver node sends an acknowledgement when it failed to receive a packet. The proposed approach is similar to ACK, however, it has three differences. First, our approach does not require additional packet transmissions. This is a significant advantage, because minimizing the number

of transmissions is meaningful in conserving energy. Second, our approach may induce high latency, because CTDCP moves the transmitted packet to the end of the sending queue after each transmission. Additional latency is usually in millisecond order, which can be tolerated in the majority of WSN applications. Third, our approach requires larger amount of storage at the sending node to keep track of successful delivery of packets. It has been proven that reading/writing data requires orders of magnitude smaller energy compared to transmitting a packet [14] [15], therefore, we argue that this overhead is negligible.

In conclusion, we provide two mechanisms to support multiple mobile sink nodes: **bypass tree** and **broadcast-based routing**. By utilizing bypass tree, sink nodes can leverage tree structure with low route construction and data delivery cost. In order to deliver packets to multiple sink nodes simultaneously, CTDCP employs broadcast-based routing, which decreases the total number of packet transmissions.

4.3 Design of ALQDP

This section describes two basic components of ALQDP, link quality detection and link quality estimation. The two components are leveraged to provide high accuracy and agility in detecting link qualities.

4.3.1 Link Quality Detection

ALQDP follows the idea of state of the art link quality detection protocols [28] [33] [34], and leverages **EWMA enabled ETX**, **RSSI**, and **LQI** as input parameters, and provides the ability to pin the specific neighbor node's information to the neighbor table. We discuss the output parameter of the ALQDP and present the methodology to manage the link quality information as follows.

Researchers have been thoroughly analyzing and discussing link quality indication parameters [29] [27] [30] [28] [33] [34]. In traditional link quality detection protocols, researchers have been using PRR as the indicator, which has the value from 0 to 1. Meanwhile, ETX is a metric which represents the number of transmissions required to successfully deliver one packet. In consequence, ETX has a range of 1 to infinite. A lot of researchers have been proving that ETX is an excellent indicator of link qualities [27] [28] [34]. Therefore, in this thesis, we leverage ETX as an output of ALQDP.

Recent link quality detection protocols have proven the effectiveness of utilizing multiple parameters [28] [33] [34]. According to their argument,

hybrid approach of utilizing both software and hardware parameters successfully improves the accuracy of link quality detection. Therefore, we also employ hybrid approach. We argue that utilization of multiple hardware parameters can provide high accuracy in detecting link qualities. LQI is a parameter specified in IEEE 802.15.4 standard, hence, it may be unavailable in some platforms. However, most of current wireless sensor nodes use IEEE 802.15.4 compatible radio transceivers, hence, it is reasonable to assume LQI as an acquirable parameter. We utilize inexpensive hardware parameters, RSSI and LQI, to filter links with good quality, and leverage ETX to the precise link quality. ALQDP leverages EWMA for ETX, which smoothen the ETX value by taking the history of data into consideration.

In the existing link quality detection protocols, the window size for calculating the software parameters and the weight of most recent data in EWMA (or WMEWMA) is static. We argue that, when the link quality between two nodes drastically changes in a short period of time, calculation of ETX value should take place frequently, so that the output reflects the link dynamics. Meanwhile, when a node has mobility, the link qualities among neighbor nodes almost constantly change, hence, the weight of the most recent data should be large. Consequently, we argue that the window size for deriving the software parameters, and the weight of the most recent data in EWMA should be altered according to link stability and nodes' mobility, respectively.

In this thesis, we limit the number of neighbor information stored on each sensor node. This is meaningful, because the number of neighbor information is proportional to the storage usage, and sensor nodes have severe constraint on storage. It has been proven that the inconsistencies between neighbor tables of the link quality detection and data collection protocols cause network failure [35]. In order to solve this problem, we leverage the idea of *white bit* proposed in 4B: we provide an interface to pin specific neighbor information in the neighbor table of ALQDP.

4.3.2 Link Quality Estimation

Agile link quality detection is fairly important, especially when collaborating mobile nodes. This section describes our link quality estimation protocol, which is utilized to provide high agility in detecting link qualities. There are three possible approaches to detect link qualities with high agility: *A*) leverage hardware parameters, *B*) force neighbor nodes to transmit packets rapidly, and *C*) estimate the link quality leveraging the history of data. Table 4.4 shows the comparison of the three approaches in perspective of

Table 4.4: Comparison of Approaches for Agile Link Quality Detection

Approach	Agility	Accuracy	Sender Overhead	Receiver Overhead
Use of Hardware Parameters	High	Low	Low	Low
Rapid Transmission by Neighbor Nodes	Moderate	Decent	High	High
Estimation Leveraging History of Data	High	-	Low	Moderate

agility, accuracy, sender overhead, and receiver overhead.

A. Leveraging Hardware Parameters
Link quality detection based on hardware parameters (RSSI, LQI, SNR, etc.) is inexpensive and quick: quality of a link can be obtained upon the reception of a packet. However, accuracies of hardware parameters are low, therefore, this approach is not desirable. The overhead for leveraging hardware parameters is low at sender and receiver nodes, because they can easily be obtained from the radio transceivers.

B. Forcing Neighbor Nodes to Transmit Packets Rapidly
Some researchers argue that the use of software parameters (PRR, RNP, ETX, etc.) is preferable in terms of accuracy in detecting link qualities [24] [28]. In spite of their high accuracy, a certain number of data are required to derive them. Therefore, in this approach, receiver nodes collect a sufficient number of data by requesting neighbor nodes to transmit packets rapidly. It has been discussed that this approach provides high accuracy in detecting link qualities in a short period of time, however, there remain two problems: 1) link quality detection at unusual environment, and 2) high overhead at sender nodes. First, rapid transmissions by neighbor nodes cause wireless environment to be unusual. Therefore, we argue that link qualities determined in this approach are not trustworthy. Second, this approach induces high energy consumption at sender and receiver nodes, because transmission and reception of packets require large amount of energy. This is not desirable, because sensor nodes have severe constraint on energy source.

C. Estimating the Link Quality Leveraging the History of Data
The fundamental idea of this approach is to determine the relationships

Table 4.5: Data Collection Environment

Parameter	Values
Hardware	Iris mote
Operating System	TinyOS 2.x
Number of Nodes	33
Distance	$1.5m$ apart
Inter Packet Interval	2 second
Duration	9 hours
Place	5F of ι-o corridor at Shonan Fujisawa Campus, Keio University

between hardware and software parameters. By discovering their relationships, sensor nodes can utilize inexpensive hardware parameters to estimate fine grained link qualities. Consequently, this approach is as quick as acquiring hardware parameters. Since this approach does not induce any overhead at sender nodes, it is preferable than forcing neighbor nodes to rapidly transmit packets. However, receiver nodes require certain amount of calculation to determine the relationships between hardware and software parameters. We argue that incurring high overhead at receiver nodes is acceptable for mobile sink nodes, because it is usually effortless to charge or replace their batteries.

Accordingly, we argue that estimating the link quality leveraging the history of data best accommodates our requirements. However, the accuracy of the approach remains unclear, therefore, there is a need to clarify it. In order to clarify the accuracy of the approach, we conduct an experiment. The experiment has four steps: *A*. collect data in the real world environment, *B*. analyze the relationships between each parameter, *C*. choose a suitable algorithm, and *D*. define tunable parameters.

A. Collecting Data

One of the most important factors to concern is the adaptability to the real world environment. Therefore, data collection should be done in an environment where actual WSNs would be deployed.

Table 4.5 shows our data collection environment. We used Iris mote [39] as the hardware, which is an embedded sensor device widely used in the field of WSNs, and implement a data collection application leveraging TinyOS2.x [32], an open source, component-based operating system de-

Figure 4.6: Data Collection Environment

signed for small embedded devices. We deployed 33 nodes on a line with inter distance of $1.5m$, in other words, the furthest node is placed $49.5m$ away from the sink node. Inter packet interval is set to 2 seconds, and we let the application operate for 9 hours. We placed the sensor nodes in the fifth floor of *ι-o* corridor in Shonan Fujisawa Campus, Keio University. Fig. 4.6 shows our data collection environment. The red solid circle indicates the sink node, and the red dashed oval shows sensor nodes placed in a line.

We leverage polling-based approach to collect packets: the sink node transmits a packet to a specific node, and the receiver node replies a packet to the sink node. The sink node modifies the destination ID after each transmission, hence, the sink node collects data from sensor nodes starting from node ID 1 to 33, repeatedly. The sink node includes a sequence number in its transmitting packets. The receiver node retrieves RSSI and LQI from the hardware, and calculates ETX. Then, it replies the packet with link quality parameters and generated sequence number. When the sink node receives a packet, it reports the content of the packet along with calculated ETX, and RSSI and LQI obtained from the hardware. Through the data collection, we collected over 740,000 data.

B. Analyzing the Relationships Between Each Parameter

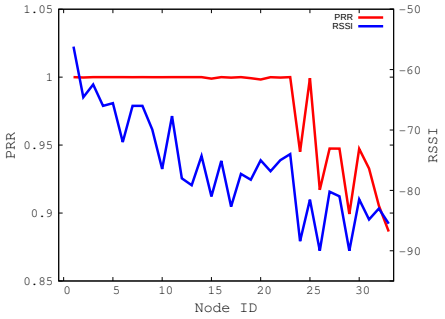
Fig. 4.7 illustrates the relationships between PRR, RSSI, LQI, and ETX, where x-axis represents the node ID, left y-axis shows the PRR, and right y-axis indicates either RSSI, LQI, and ETX. The data shown in the figure are averaged over all collected data in the experiment. Fig. 4.7(a)-4.7(c) and Fig. 4.7(d)-4.7(f) represents the data received at sender nodes, and the sink node, respectively. Since RSSI indicates the signal strength of received packets, it is understandable that sensor nodes with greater ID (in other words, nodes which are placed further from the sink node), has lower RSSI value. The accuracy of RSSI in detecting PRR is decent: RSSI decreases as PRR decreases, however, when clients receive packets (Fig. 4.7(a)), the RSSI have small relation to PRR. The LQI value is not proportional to PRR, therefore, simply using LQI as a link quality predicator would not work well. Nevertheless, when the LQI significantly decreases (node 26 in Fig. 4.7(b) and Fig. 4.7(e)), we observed degradation in PRR. This, in turn, shows that LQI can be leveraged to filter links with bad qualities. It is considered that significant decrease in LQI value at node 26 is caused by self interference: radio packet is reflected by the walls, and/or ceilings, and collides with each other, causing packets to be damaged. In the meantime, ETX leverages link layer information, therefore, it has high accuracy in detecting the link qualities.

Fig. 4.8 depicts the relationships between ETX, RSSI, and LQI. We process the whole data with the window size of 100 packets, and calculate the average of each parameter. As the figures indicate, the RSSI and LQI decreases linearly as the ETX increases. This results seem to be conflicting with the relationships illustrated in Fig. 4.7, however, the data used in Fig. 4.8 are processed data, which are averaged over a certain number of data. Consequently, this result indicates that the use of processed RSSI and LQI values can be leveraged to estimate the software parameters.

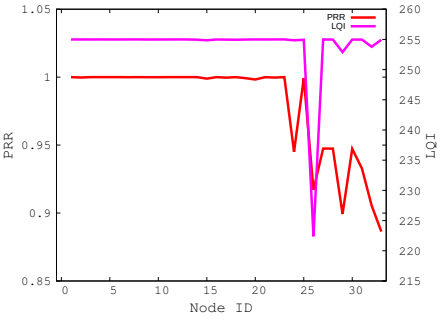
C. Choosing a Suitable Algorithm

There are various machine learning algorithms which estimate a value from given data sets. We are assuming small embedded devices as sink nodes, therefore, it is essential to consider the computational cost. Typically, the performance of a machine learning algorithm is proportional to the computational cost. Therefore, it is important to determine the algorithm which has high accuracy along with small amount of calculation.

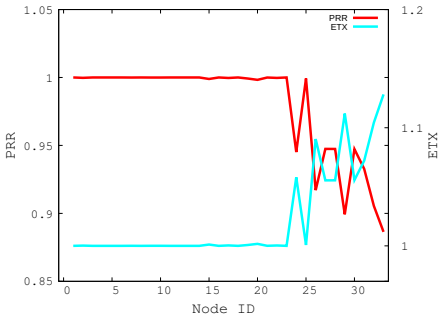
We implemented linear regression and softmax regression using nesC. However, due to the constraint of programmable flash size, softmax regres-



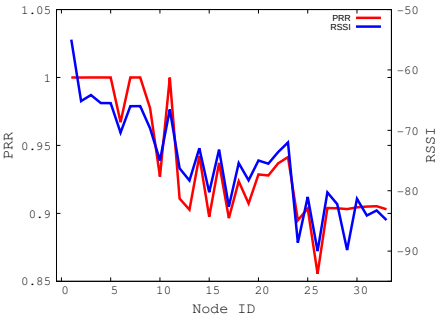
(a) PRR vs. RSSI (Sink to Client)



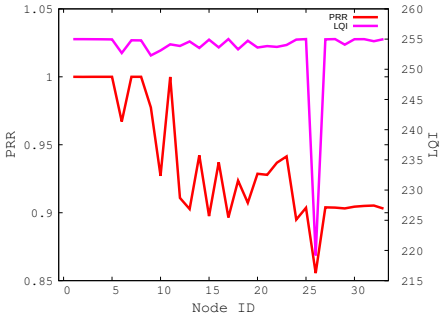
(b) PRR vs. LQI (Sink to Client)



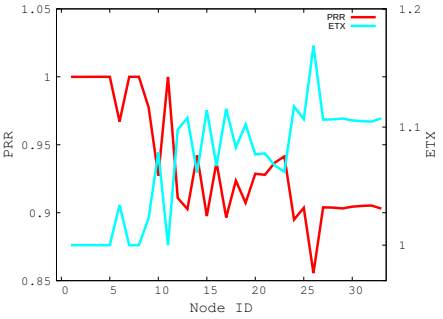
(c) PRR vs. ETX (Sink to Client)



(d) PRR vs. RSSI (Client to Sink)

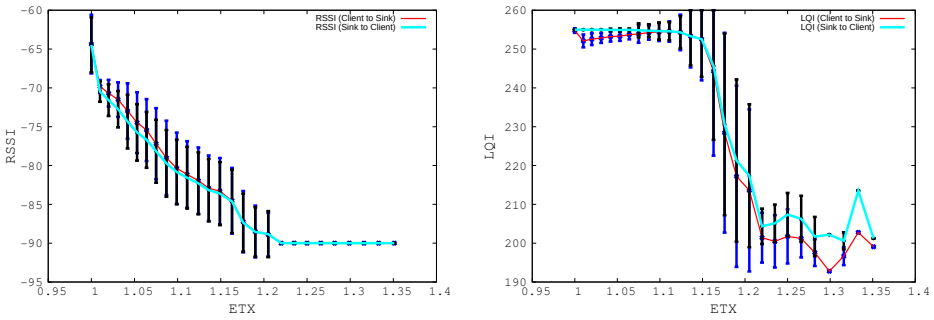


(e) PRR vs. LQI (Client to Sink)



(f) PRR vs. ETX (Client to Sink)

Figure 4.7: Relationships Between PRR and {RSSI, LQI, ETX}



(a) ETX vs. RSSI

(b) ETX vs. LQI

Figure 4.8: Relationships Between ETX, RSSI and LQI

sion could not fit on Iris mote, which is one of the latest sensor devices. Liu et al. employed logistic regression to estimate the link qualities, and proved that it has a reasonable accuracy [34]. In spite of their high accuracy, the overhead for deriving the equation is significant: calculation cost of natural logarithm is extremely high especially in small embedded devices with low performance processors. Moreover, we argue that the use of logistic regression is overkill; according to our experimental result (Fig. 4.8(a) and Fig. 4.8(b)), linear model is sufficient to express the relationships between hardware and software parameters. In order to improve the performance of linear regression, we adopt two novel ideas: 1) minimizing the calculation overhead, and 2) omitting data setes with the best metric.

1. Minimizing the Calculation Overhead
   We leverage two approaches to minimize the calculation overhead: I) combining RSSI and LQI, and II) employing EWMA to compress multiple information.

   I) Combining RSSI and LQI:
   Linear regression determines least squares with a linear equation, and estimates a parameter from given variables.

$$y = ax + b \tag{4.1}$$

   Equation 4.1 expresses the simplest equation of linear regression, where x and y stands for variables, and a and b expresses constants. The first step is to determine the constants (a and b) of the equation: their values are defined by minimizing the sum of square root of the difference between given sets of variables (x and y) and the values

calculated using the equation. Then, by substituting either variable, another variable can be determined.

$$a = \frac{n \sum_{k=1}^{n} x_k y_k - \sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k}{n \sum_{k=1}^{n} x_k^2 - \left(\sum_{k=1}^{n} x_k \right)^2} \tag{4.2}$$

$$b = \frac{\sum_{k=1}^{n} x_k^2 \sum_{k=1}^{n} y_k - \sum_{k=1}^{n} x_k y_k \sum_{k=1}^{n} x_k}{n \sum_{k=1}^{n} x_k^2 - \left(\sum_{k=1}^{n} x_k \right)^2} \tag{4.3}$$

The Equations 4.2 and 4.3 represent the equations to solve two constants, a and b, where n expresses the number of data sets. These equations only involve simple iteration, however, when there are more than two variables, the calculation becomes extremely complicated: conversion calculation is required, which involves huge amount of calculation. In order to minimize the calculation overhead, we combine RSSI and LQI. By doing so, the equation becomes binomial, which involves much simpler calculation. Our challenge is to determine the method of combining these parameters.

II) Employing EWMA To Compress Multiple Information:
When utilizing linear regression, we use the combined parameter of RSSI and LQI as x, and ETX as y. In the Equations 4.2 and 4.3, the value of n represents the total number of training data sets. The number of data sets is proportional to the performance of the algorithm. In the meantime, when the number of training data sets is large, calculation overhead becomes large. We leverage ETX as an output parameter of ALQDP, therefore, the possible output values are limited. For instance, when the ETX calculation window is set to 10 packets, the minimum and the maximum possible ETX values are 1 and 11, respectively. Therefore, we employ EWMA to compress data sets which have the same ETX value. By doing so, the value of n in the Equations 4.2 and 4.3 is limited to the range of ETX values rather than the number of data sets, which is significantly small.

2. Omitting Data Sets with the Best Metric
We argue that, accuracy of link quality detection can be improved by omitting the data sets with best metric. For example, when considering the relationships between ETX and RSSI (Fig. 4.8(a)), RSSI values

Table 4.6: Tunable Parameters of Link Quality Estimation

Parameter	Values
Window Size (W)	10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 200, 500
Weight of RSSI (W_R)	0 - 19 (step = 1)
Weight of LQI (W_L)	0 - 19 (step = 1)
Weight of Most Recent Data (λ)	0.05 - 0.95 (step = 0.05)

have large variance when ETX equals to 1. This implies that when RSSI is higher than a certain value, quality of the link is most likely the best. This, in turn, shows that a data set with the best ETX value worsens the accuracy of the estimation. Therefore, we omit these data sets from the training data.

Defining Tunable Parameters

There are several parameters which need to be defined when utilizing the algorithm we described above. We first discuss the parameters to define, and then, explain the procedure and the result of parameter definition process.

Parameters To Determine:

First, we need to define the frequency of equation determination. This is proportional to the overhead: frequent equation update induces large calculation overhead. We define this frequency as window size (W).

In order to minimize the calculation overhead, ALQDP combines RSSI and LQI. Since the combining process takes place whenever a node receives a packet, it is desired to be lightweight. Therefore, we choose to simply apply weight on RSSI and LQI. We define the weight of RSSI and LQI as W_R and W_L, respectively.

$$EWMA_t = \lambda Y_t + (1 - \lambda) EWMA_{t-1} \tag{4.4}$$

Equation 4.4 shows the general equation of EWMA calculation, where λ represents the weight of the most recent data, which has significant impact on the accuracy of the equation. We define the weight of most recent data in EWMA as λ.

Parameter Definition Procedure:

Table 4.7: Defined Parameters of Link Quality Estimation

Parameter	Values
Window Size (W)	35
Weight of RSSI (W_R)	2
Weight of LQI (W_L)	1
Weight of Most Recent Data (λ)	0.10
Mean Square Error	0.209

In order to define the parameters of the protocol, we utilized the data collected in the experiment. The procedure consists of two steps: determining the equation using W training data sets, and calculating the mean square error between the actual and estimated ETX value of the following W data. We repeat these procedures with the whole data sets. Table 4.6 indicates the parameters used to determine the parameters of the link quality estimation protocol.

We repeated the determination process for every combination, and confirmed that the combination of parameters indicated in Table 4.6 achieved the highest accuracy. We affirmed that proposed estimation protocol achieves the mean square error of 0.209, which is equivalent to 4C [34]. Although the estimation accuracy is equivalent, we argue that ALQDP is superior to 4C due to two reasons. First, the calculation overhead of ALQDP is considerably small compared to 4C. 4C leverages logistic regression, on the other hand, ALQDP utilizes linear regression with some improvement features, which further minimize the calculation overhead. Second, ALQDP requires smaller number of training data. 4C requires at least 2,000 data sets as training data. In contrast, ALQDP can start estimation process after collecting 35 data sets, which is substantially small. For these reasons, we argue that ALQDP is superior to 4C in terms of overall performance.

Fig. 4.9 shows the Estimation errors in perspective of each window, where x-axis indicates windows, and y-axis represents square error. The green horizontal line represents the average of all data. We observed larger errors in the first 200 data sets, which implies that ALQDP performs poor when it has the small number of training data sets. However,ALQDP still requires much smaller number of training data sets to estimate link qualities with high accuracy compared to 4C.

Fig. 4.10 indicates the estimation errors when parameters are modified from the defined values, where x-axis indicates each parameter, and y-axis

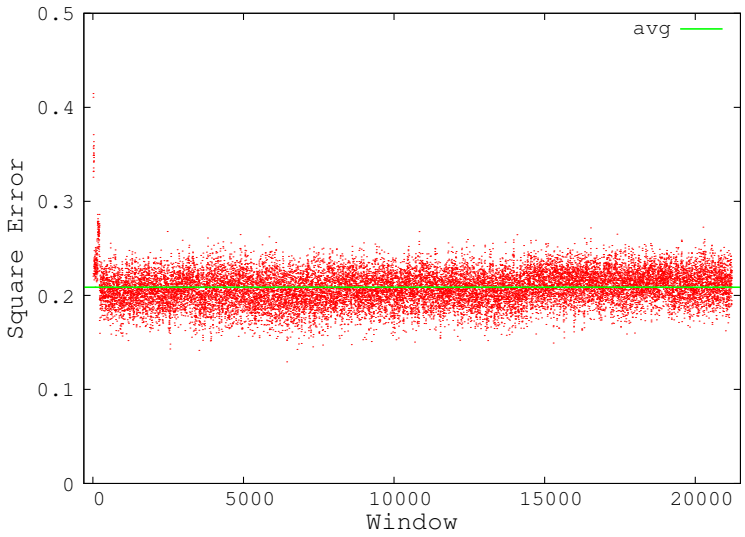
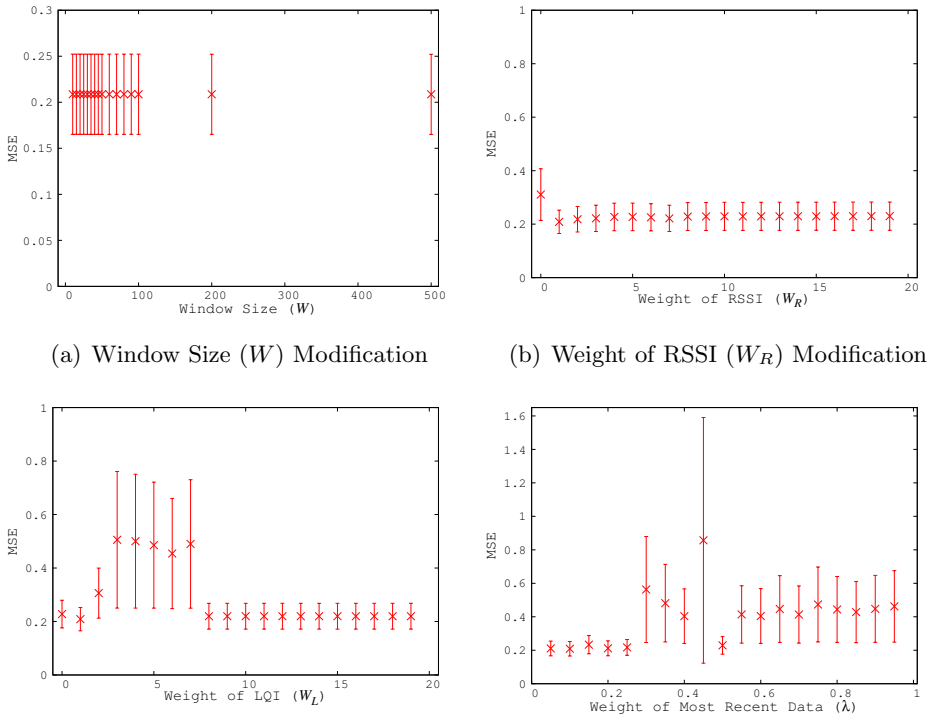


Figure 4.9: Estimation Errors with Defined Parameters

represents the mean square error. In each figure, x indicates the average value, and upper and lower bounds represent the standard deviation. As shown in Fig. 4.10(a), we could not observe much differences in errors when the window size is modified. This indicates that the frequency of parameter determination can be low. However, when the window size is large, sensor nodes have to collect a large number of data before starting estimation. Therefore, we choose to use the window size which achieved highest accuracy. Fig. 4.10(b) and Fig. 4.10(c) illustrate the estimation errors when the weight of RSSI and LQI is modified, respectively. The errors remained small for the most values. The weight of RSSI and LQI have small influence on the amount of calculation, therefore, we simply used the values with the smallest errors. As shown in Fig. 4.10(d), we confirmed wide variety of errors by modifying the weight of the most recent data in EWMA. Smaller errors were observed when λ is small, which signifies that the use of the history of data is meaningful.

4.4 Summary

In this Chapter, we propose our approach of combining cluster and tree structure to achieve high data collection efficiency and scalability, simultaneously. In order to achieve high efficiency in collecting data, and high agility in detecting link qualities, we propose CTDCP, a data collection protocol,



(a) Window Size (W) Modification

(b) Weight of RSSI (W_R) Modification

(c) Weight of LQI (W_L) Modification

(d) Weight of Most Recent Data (λ) Modification

Figure 4.10: Parameters and Estimation Errors

and ALQDP, a link quality detection protocol, respectively. We propose bypass tree, which reduces the route construction cost, as well as data delivery cost. CTDCP also utilizes broadcast-based routing to minimize the overhead of packet transmissions. ALQDP leverages the combination of software and hardware parameters to improve the accuracy of link quality detection while minimizing the calculation overhead. ALQDP improves the agility in detecting link qualities by utilizing link quality estimation. We conduct a real world experiment, and used the collected data to define the parameters of link quality estimation. As a result, we determined the equation which has the mean square error of approximately 0.209, and considerably low calculation overhead.

Chapter 5

Implementation

In this Chapter, we describe the implementation of CTDCP and ALQDP. First, we discuss the platform we used to implement proposed protocols on. Second, we explain the system overview. Finally, we describe the implementation of CTDCP followed by ALQDP.

5.1 Implementation Platform

Researchers have been proposing wide variety of data collection protocols [16] [17] [19] [20] [21] [22] [22] [25], and implementing them on various kinds of platforms, such as, Mica2, and Micaz motes. We choose to implement proposed protocols on Iris mote [39], a successor of Micaz mote, which is widely used in the area of WSNs. Table 5.1 shows the specification of Micaz and Iris mote. There are not much differences between Micaz and Iris mote, except for RAM size and the amount of energy required in sleep mode. Since these differences do not affect the performance of data collection protocols nor link quality detection protocols, it is reasonable to implement proposed protocols on Iris mote.

Motes can be programmed through TinyOS [32], Xmesh, Contiki, etc. TinyOS is an open source, component-based operating system designed for small wireless devices. We choose to implement CTDCP and ALQDP on TinyOS2.x, because the majority of related work use it. Accordingly, we implement CTDCP and ALQDP using network embedded systems C (nesC), an event-driven programming language used in TinyOS.

Table 5.1: Comparison Between Micaz and Iris Mote

	Micaz	Iris
CPU	Atmel Atmega128L	Atmel Atmega1281
CPU Type	8bit	8bit
Clock Speed	7.37MHz	7.37MHz
Flash Memory	128kB	128kB
Serial Flash	512kB	512kB
RAM	4kB	8kB
Radio Chip	TI CC2420	Atmel RF230
Frequency Band	2,400-2,483.5MHz	2,400-2,480MHz
TX Data Rate	250kbps	250kbps

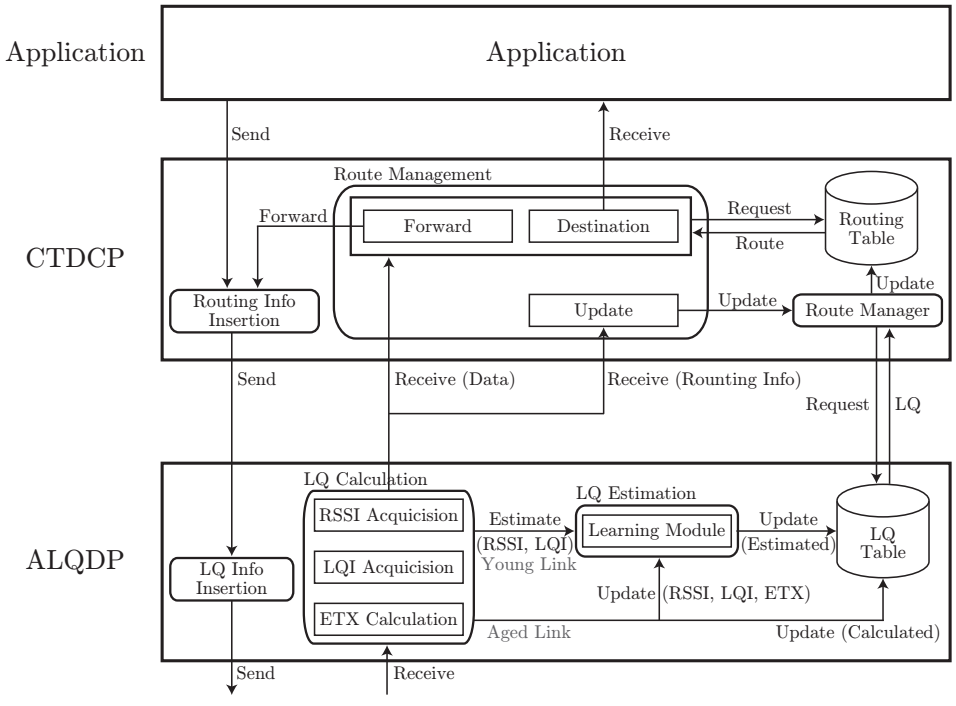


Figure 5.1: System Architecture of CTDCP and ALQDP

5.2 System Overview

Fig. 5.1 illustrates the overall system architecture of proposed protocols. The upper most layer in the figure is the application layer, and we assume the existence of MAC or abstraction layer beneath the link quality detection layer. We describe the behaviors of CTDCP and ALQDP as follows:

Whenever the overlying layer transmits packets, CTDCP inserts the routing information, which includes, ID of the sender node, hop count, and path link quality. The ID of the sender node is leveraged to identify the source node of a packet. The hop count is utilized to find out the route inconsistency: when a node receives a packet which has the same content with different hop count, there is a great chance that there is a route inconsistency. In the meantime, path link quality is leveraged by neighbor nodes to check if their current paths are most preferable. When the node receives a packet, CTDCP processes it based on its type; if it is a routing packet, CTDCP updates the routing table according to its information, and if it is a data packet, CTDCP checks the routing table and decides to either discard, forward, or notify the application of the reception of the packet.

ALQDP inserts link quality information whenever the overlying component is transmitting a packet. This information is leveraged to measure link qualities among neighbor nodes. Whenever a node receives a packet, ALQDP retrieves RSSI and LQI from the radio chip, and calculates the ETX utilizing the sender node's link quality information. We define two types of links in accordance with the number of received packets, *Aged Link* and *Young Link*. Aged Link indicates that the node received enough number of packets to calculate ETX, on the other hand, Young Link suggests that the link quality should be estimated using the link quality estimation protocol. Whenever the node receives a packet through Aged Link, ALQDP updates the parameter of the equation utilizing the RSSI, LQI and ETX, and updates corresponding link quality entry. Whenever the node receives a packet through Young Link, ALQDP estimates the ETX by passing the RSSI and LQI to the link estimation protocol, and updates the link quality table according to it. In this manner, ALQDP provides a link quality table of both Aged Link and Young Link.

5.3 Implementation of CTDCP

This section describes our implementation of CTDCP. CTDCP has two basic components: Routing Manager and Packet Processor. The Routing Manager component manages routes within a sensor node and among neighbor

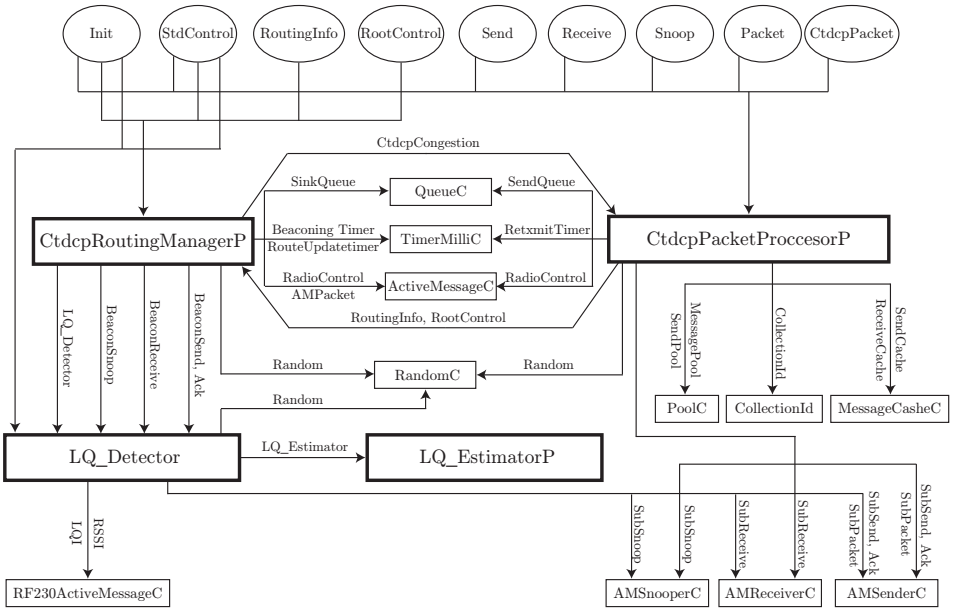


Figure 5.2: Wiring of the Entire System

nodes. The Packet Processor component processes received packets.

Fig. 5.2 depicts the wiring of the entire system. The thick squares indicate the main components we implemented: CtdcpRoutingManagerP, CtdcpPacketProcessorP, LQ_Detector, and LQ_Estimator. Thin squares indicate the generic components used by our system, and names beside the arrows represent the interfaces used to connect components. Circles depicted in upper part of the figure indicates the interfaces that application need to wire. They are used for sending/receiving packets and acquiring CTDCP's information.

5.3.1 Routing Manager Component

Fig. 5.3 illustrates the main component of Routing Manager (CtdcpRoutingManagerP). CtdcpRoutingManagerP leverages radio transmission/reception interfaces provided by LQ_DetectorP, the main component of ALQDP. Consequently, information is added and retrieved in link quality detection layer whenever CtdcpRoutingManagerP sends or receives a packet. Routing Manager has three roles: cluster formation, basic routing management, and sink nodes management.

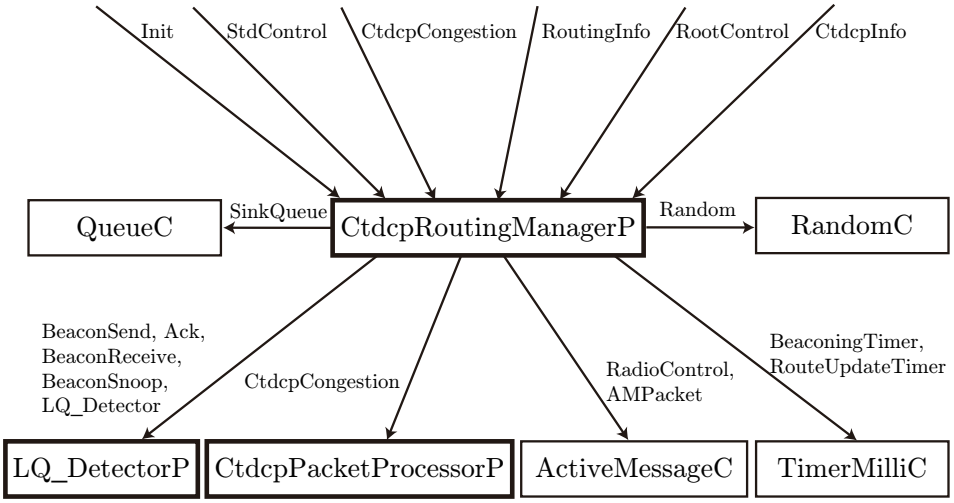


Figure 5.3: Wiring of Routing Manager

```
typedef struct {
    am_addr_t neighbor_addr;
    am_addr_t parent;
    uint16_t etx_to;
    uint16_t etx_from;
    uint16_t subtree_size;
    bool congestion;
} ROUTING_ENTRY;
```

Figure 5.4: Routing Entry Structure

Basic Routing Management

Fig. 5.4 illustrates the structure of routing entry leveraged in CTDCP. ROUTING_ENTRY is utilized for routing among statically placed sensor nodes, and its members are described as follows:

neighbor_addr: Link layer address of the neighbor node.
parent: Parent of the neighbor node. The parent is in charge of forwarding packets transmitted by the neighbor node.
etx_to: ETX value from the local node toward the neighbor node.
etx_from: ETX value from the neighbor node toward the local node.
subtree_size: The number of nodes in its subtree. This indicates the number of packets to forward.

```
typedef nx_struct {
    nx_am_addr_t parent;
    nx_uint16_t etx;
    nx_uint16_t subtree_size;
    nx_uint8_t flag;
} CTDCP_ROUTING_PACKET;
```

Figure 5.5: Routing Packet Structure

congestion: Indicates if the neighbor node is congested. Congestion is set to TRUE when the neighbor node is forwarding the excessive number of packets.

Overlying application layer is in charge of defining the numbers of entries. Normally, the size of them is proportional to the performance of the Routing Manager, because it can choose the next hop from the larger number of neighbor nodes. In the meantime, they are also proportional to the storage usage. Since storage size is severely constrained in wireless sensor nodes, table size should be small in perspective of storage usage.

The Routing Manager leverages *Routing Packet* to exchanging routing information among neighbor nodes. Fig. 5.5 shows the structure of the Routing Packet, and "nx_" prefix signifies that it is a network type. Each member of the packet is described as follows:

parent: Link layer address of parent of the node which transmitted the packet. If the packet is transmitted using unicast, the parent replies an acknowledgement, otherwise, neighbor nodes simply update their routing entry according to the information.

etx: The etx field expresses the ETX value from the transmitting node toward the root node.

subtree_size: This indicates the number of nodes in its subtree.

flag: There are three flags, FREQUENT_ROUTE_UPDATE, NODE_CONGESTED, and ROOT_DETACH. FREQUENT_ROUTE_UPDATE is leveraged to request neighbor nodes to aggressively transmit routing information, NODE_CONGESTED is utilized to notify neighbor nodes to ask for path change, and ROOT_DETACH is used to notify a node to act as a cluster head.

We leverage two types of timers to manage routing table; RouteUpdateTimer and BeaconingTimer. The frequency of RouteUpdateTimer is

defined by the overlying applications. Whenever the RouteUpdateTimer is fired, the Routing Manager compares the link quality between its current destination and neighbor nodes in the routing table. If the Routing Manager finds a link with better quality, it notifies the new parent node of the destination change using unicast. We choose to leverage unicast, because the receiver node can simply acknowledge the sender node of the reception of packet, while neighbor nodes can receive the same packet using snooping (In TinyOS, overheard packet is notified through Snoop interface). BeaconingTimer is equivalent to RouteUpdateTimer, but has two differences: first, BeaconingTimer send notification packets even when the route is not updated, and second, the timer interval is infrequent. Notification process is always invoked when BeaconingTimer fires, so that the routing information is shared among neighbor nodes. BeaconingTimer employs the idea of Trickle [26]: the timer interval is exponentially increased, and when the event of interest occurs, the timer is set to the minimal interval. Trickle is originally designed for code propagation, however, the idea of adaptive interval is proved to work well in routing protocols [24]. The interval of BeaconingTimer is set to the minimal when the node 1) has no destination, and 2) receives a packet with FREQUENT_ROUTE_UPDATE flag. A node may have no destination when it has just booted, or the parent node got lost. In these cases, the Routing Manager aggressively searches for a new route by setting the timer interval to minimal. When the node has just booted, the Routing Manager requests neighbor nodes to frequently transmit routing information, by setting FREQUENT_ROUTE_UPDATE flag in the transmitting beacon packet. When neighbor nodes receive a packet with FREQUENT_ROUTE_UPDATE flag, Routing Manager resets the interval of BeaconingTimer to minimal. Newly booted node utilizes these routing information to find a path.

Whenever the node transmits a Routing Packet, the Routing Manager inserts local node's routing information. The Routing Manager sets the link layer address of its parent node as parent. Note that when a node has no route, it sets broadcast address as parent. In this way, receiver nodes can understand that the transmitting node does not have valid route. The Routing Manager inserts the ETX value from the local node toward the root node as etx. When the local node is a root node, the Routing Manager fill the etx field with 0 instead of 1, otherwise, it is impossible to identify root nodes, because a node which is one hop away from a root node with perfect link quality also has ETX of 1. Two types of flags are utilized in Routing Packet: FREQUENT_ROUTE_UPDATE flag is set when requesting neighbor nodes to transmit routing information rapidly, and NODE_CONGESTED is set

```
1  if (packet->parent != INVALID)
2      if (packet->parent == LOCAL_ADDR || packet->etx == 0)
3          routing_table.insert(packet->origin);
4          routing_table.pin(packet->origin);
5      end if
6      routing_table.update(packet->origin, packet->info);
7  end if
8  if (packet->flag == FREQUENT_ROUTE_UPDATE)
9      BeaconingTimer.setInterval(minimal);
10 end if
```

Figure 5.6: Pseudo Code for Processing Routing Packet

when the local node is forwarding the excessive number of packets.

Whenever a node receives a Routing Packet, Routing Manager updates its routing table according to its information. Fig. 5.6 shows the pseudo code for processing a Routing Packet. The Routing Manager only updates the routing information if the parent section of the packet is not invalid, because neighbor node with no valid path should not be leveraged. When parent field of the packet is equivalent to the local node's address, or etx section of packet equals to 0, Routing Manager forcefully inserts the information to the routing table, and pin it. This is because, this information indicate that the local node has to forward packets transmitted by the sender node, and the transmitting node is a root node, respectively. When an entry is pinned, the Routing Manager cannot remove or overwrite it until it is unpinned. Updating process (line 6) updates the sender node's information in the routing table. The Routing Manager updates the etx_to and etx_from in the routing table leveraging underlying link quality detection layer, and sets the congestion to TRUE if it is indicated. When the entry of the sender node exists in the routing table, Routing Manager simply updates the information, and when there is no entry, Routing Manager tries to insert the entry. If there is a room in the routing table, Routing Manager simply creates an entry, and if not, Routing Manager overwrites an entry with the worst link quality. When the packet has FREQUENT_ROUTE_UPDATE flag, Routing Manager sets the interval of BeaconingTimer to minimal, so that the routing information is notified to the neighbor nodes aggressively.

Cluster Formation

In order to form clusters among the WSN, the Routing Manager provides a tunable parameter, CTDCP_MAX_CLUSTER_SIZE, to overlying applications. This parameter defines the maximum number of nodes involved in each cluster. We suppose overlying application to select and define root nodes of the WSNs. There are two reasons we employ this idea: first, the application may want to collect data at statically placed root node while delivering data to multiple mobile sink nodes, and second, we believe that costly root node selection should not be done in routing protocols, instead, position of root nodes better be selected when deploying the WSN. We provide an interface, RootControl for overlying layer to set/unset the root status of a node.

When the RouteUpdateTimer fires in the root node, Routing Manager searches its routing table for entries which have itself as parent, and calculates the sum of their subtree_size. This value indicates the number of nodes involved in its cluster. Whenever it exceeds CTDCP_MAX_CLUSTER_SIZE, the root node splits its cluster into two. The root node searches its routing table for a node which has the subtree_size closest to half of the current cluster size, and transmits a Routing Packet with ROOT_DETACH flag to that node. When a node receives a Routing Packet with ROOT_DETACH flag, it declares itself as a root node, and starts acting as a cluster head. Although there are many ways to split a cluster, we believe that splitting a cluster into halves is the most simple and effective way. When a cluster is divided to approximately the same size, their performance are supposed to be equal. Meanwhile, Routing Manager simply detaches a whole subtree from a cluster, therefore, there is no need to reconstruct routes in each cluster.

In the meantime, Routing Manager considers to combine clusters if the sum of sensor nodes involved in two clusters is smaller than CTDCP_MAX_CLUSTER_SIZE. Whenever the RouteUpdateTimer fires in the root node, Routing Manager calculates the sum of subtree_size of neighbor root node in the routing table and itself. We argue that it is better for a node with smaller cluster size to be merged to a node with larger cluster size, because, it is usually better to have the smaller number of nodes involved in each subtree. For these reasons, the following procedure is only leveraged by the root node with smaller cluster size. The root node stops acting as a root node, instead, it operates as an inner node of a tree structure. It transmits a Routing Packet to the other root to notify that the node became a subtree of that node. This procedure allows Routing Manager to combine two clusters

```
typedef struct {
    am_addr_t sink;
    am_addr_t source;
    uint16_t etx_to;
    uint16_t etx_from;
    uint8_t bypass_size;
    am_addr_t *bypass_nodes;
    uint8_t bypassed_size;
    am_addr_t *bypassed_nodes;
    bool congestion;
} SINK_ROUTING_ENTRY;

typedef struct {
    am_addr_t sink;
    am_addr_t bypass_from;
    am_addr_t bypass_to;
    uint16_t etx_from_to;
    uint8_t flag;
} SINK_BYPASS_LIST;
```

Figure 5.7: Sink Routing Entry Structure

without reconstructing tree structures, because, from the perspective of the root node, this process simply attaches a subtree.

Sink Nodes Management

Fig. 5.7 illustrates the structure of sink routing entry and bypass list leveraged in CTDCP. The SINK_ROUTING_ENTRY is used for routing toward sink nodes, and SINK_BYPASS_LIST keeps track of bypassing and bypassed nodes. Their parameters are described as follows:

– SINK_ROUTING_ENTRY:

sink: The link layer address of the sink node in the WSN. This indicates that the packets should be delivered to the specified sink node.

source: The link layer address of the source node. When the local node receives packets from the source node, it forwards them toward the sink node.

etx_to: ETX value from the local node toward the sink node.

etx_from: ETX value from the sink node toward the local node.

bypass_size: The number of bypass node in the entry.

*bypass_nodes: The list of bypass nodes. When the local node receives packets from a node in the list, it forwards them toward the its own parent node.

```
typedef nx_struct {
    nx_am_addr_t sink;
    nx_am_addr_t parent;
    nx_am_addr_t detach;
    nx_uint16_t etx;
    nx_uint8_t flag;
} CTDCP_SINK_ROUTING_PACKET_T;
```

Figure 5.8: Sink Routing Packet Structure

bypassed_size: The number of bypassed node in the entry.

*bypassed_nodes: The list of bypassed node. The nodes in the list is bypassed, hence, the local node does not forward their packets even if they are transmitted by the source node.

congestion: Indicates if the neighbor node is congested. Congestion is set to TRUE if the node is forwarding the excessive number of packets.

– SINK_BYPASS_LIST:

sink: The link layer address of the associated sink node.

bypass_from: The link layer address of the node which is being bypassed.

bypass_to: The link layer address of the destination node.

etx_from_to: The ETX value from the bypassed node to the bypassing node.

flag: This indicates if the entry is being used or not.

The number of entries in SINK_ROUTING_ENTRY, bypassing_size, and bypassed_size is defined by the overlying application. SINK_BYPASS_LIST keeps track of link qualities among related neighbor nodes, and continually check the validity of bypassing.

The Routing Manager leverages *Sink Routing Packet* to manage sink node's routes. Fig. 5.8 shows the structure of the Sink Routing Packet, and each member of the packet is described as follows:

sink: Link layer address of the sink node. This information is leveraged to distinguish multiple sink nodes.

parent: The parent and detach information is leveraged to manage routing for sink nodes. Their roles change in accordance to the procedure of bypass tree construction.

detach: Explained above.

etx: The field shows the ETX value from the local node toward the sink node.

flag: The flag section indicates the role of the packet.

Sink Routing Packets can be classified into three types: *a*) attach, *b*) detach, and *c*) bypass. We describe each of the them as follows.

a) attach

Attach message is leveraged in one of the two events: 1) when a sink node requests a node to forward packets, or 2) when the forwarding node notifies its parent of it. For the simplicity of explanation, we define the transmitting node as "requesting node" and receiver node as "requested node". When a requesting node notifies the requested node of forwarding packets, it defines each field (sink, parent, detach, etx, flag) of a packet with, address of the attaching sink node, address of the requested node, broadcast address, ETX value from the requested node to the requesting node, and CTDCP_SINK_FLAG_ATTACH, respectively. Note that sink, detach and flag field of packets are fixed throughout an attaching event. When attach packet is successfully transmitted, the requesting node moves that packet to the end of the sending queue, and inserts the information to the SINK_ROUTING_ENTRY. CTDCP sets the sink, source, etx_to, etx_from information in accordance to the transmitting information, and initialize bypass_size and bypassed_size with 0. Upon reception of a Sink Routing Packet with CTDCP_SINK_FLAG_ATTACH flag, the requested node notifies its own parent of forwarding packets. The requesting node leverages the packet transmitted by the requested node as an acknowledgement, and removes the packet from its sending queue. Consequently, when requesting node did not receive that packet, it retransmits the packet.

b) detach

The detaching process is similar to the attaching process. When detaching from a forwarding node, the requesting node sets the address of requested node to the detach field, and set CTDCP_SINK_FLAG_DETACH to the flag field. Note that parent and etx field of the packet is unused in detaching process. Upon reception of a detach packet, CTDCP removes an entry which has the same sink node as the received packet, and transmits a detach packet to its parent node. When a sink node decides to change the forwarding node, it can request attaching and detaching with one packet, by specifying both parent and detach field, and setting CTDCP_SINK_FLAG_ATTACH and CT-

DCP_SINK_FLAG_DETACH flag. This allows CTDCP to reduce the latency when switching between nodes, while minimizing the packet transmission overhead by achieving two queries with one packet.

c) bypass
The bypassing process takes place along with attaching process. Whenever a node (which is not the parent of the requesting node) receives an attach packet, it determines the effectiveness of bypassing. For the simplicity of explanation, we define this node as "bypass node". If the node decides to bypass the requested node, it sets the parent, detach, and flag field of a packet with, link layer address of the requesting node, that of the requested node, and combination of CTDCP_SINK_FLAG_BYPASS_PARENT and CTDCP_SINK_FLAG_BYPASS_DETACH. Similarly to the process of an attach packet, the bypass node moves the transmitted packet to the end of the sending queue after each transmission. When the requesting node receives the bypass packet, it inserts the bypass node's address to the bypass_nodes, thereafter, it forwards packets transmitted by the bypass node. Meanwhile, when the requested node receives the bypass packet, it adds the link layer address of the bypass node to bypassed_nodes, and stops forwarding packets transmitted by the bypass node. They reply an acknowledgement by adding either CTDCP_SINK_FLAG_ACK if the insertion is succeeded, and CTDCP_SINK_FLAG_BYPASS_FAIL if the insertion failed. When the bypass node receives an acknowledge with CTDCP_SINK_FLAG_BYPASS_FAIL, it removes the bypass packet from the queue, and notifies another node to remove itself from its bypass table. On the other hand, when the bypass node receives an acknowledgement with CTDCP_SINK_FLAG_ACK from either requesting or requested node, it removes the corresponding flag and set broadcast address to the parent or detach field. This procedure allows the bypass node to receive acknowledgement only once from each node, hence minimizes communication overhead. After receiving successful acknowledgements from both nodes, bypass node inserts an entry to its SINK_BYPASS_LIST. The bypass node sets sink, bypass_from, bypass_to, etx_from_to field with, link layer address of the sink node, that of requested node, that of requesting node, and ETX value from the requested node to the requesting node, respectively. Whenever the link quality information among the bypass node, requesting node, and requested node is updated, the bypass node redetermines the validity of bypassing. If bypassing is no longer effective, bypass node transmits

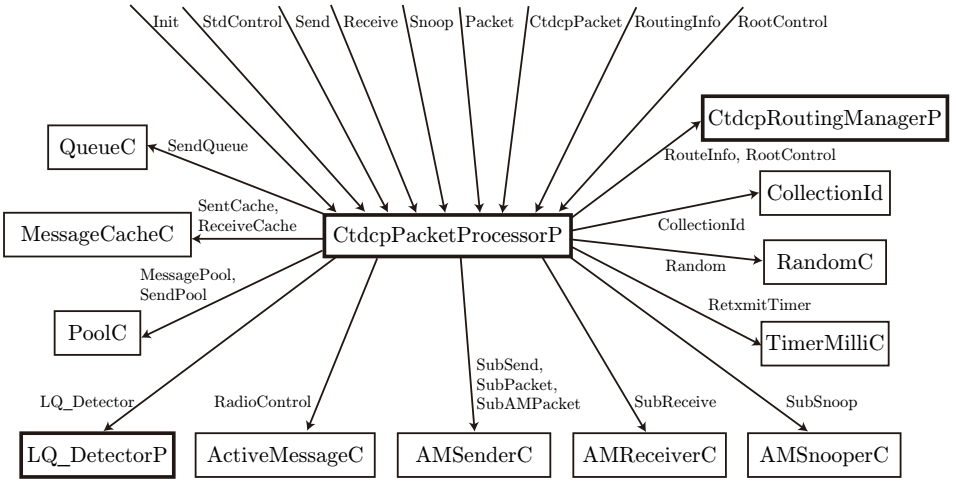


Figure 5.9: Wiring of Packet Processor

a bypass packet with CTDCP_SINK_FLAG_RM_BYPASS, and removes the bypass entry from the requesting and requested node.

Transmitting a packet with larger payload is proven to require greater amount of energy [40], hence, our approach of having both parent and detach field in a packet requires more energy. However, both of them are leveraged except when a sink node enters or leaves the WSN, which seldom occurs. Therefore, we argue that the advantage of combining multiple queries into one packet surpasses the disadvantage of having possibly unnecessary field.

Routing Manager aggressively transmits Sink Routing Packet, by checking its send queue for unsend packets after every successful transmission of a packet. This enables to establish path from each sensor node toward the sink node quickly.

5.3.2 Packet Processing Component

Fig. 5.9 indicates the components of Packet Processor (CtdcpPacketProcessorP). CtdcpPacketProcessorP leverages routing information and root status provided by CtdcpRoutingManagerP, and supplies link layer information to LQ_DetectorP. It leverages generic radio transmission/reception interfaces, which implies that link quality information is not inserted or retrieved when it exchange packets. Meanwhile, it provides packet exchange interfaces, which are leveraged by the overlying applications. This component is in charge of sending and forwarding data packets. Its basic role is to transmit packets queried by the local application, and to forward packets transmitted

```
typedef nx_struct {
    nx_uint8_t collection_id;
    nx_am_addr_t source_addr;
    nx_uint8_t seq;
    nx_uint8_t hop_count;
    nx_uint16_t etx;
    nx_uint8_t flag;
} CTDCP_DATA_HEADER_T;
```

Figure 5.10: Header of Data Packet in CTDCP

by child nodes.

Fig. 5.10 illustrates the structure of header leveraged in CTDCP data packets. Each parameter is described as follows:

collection_id: Indicates the collection ID of the packet. Collection ID is a generic parameter utilized in routing protocols of TinyOS.

source_addr: Link layer address of the source node. TinyOS only provides the link layer address of a transmitting node, therefore, when packets are forwarded by multiple nodes, applications cannot determine the actual source node. Hence, CTDCP remembers the source node utilizing its data header.

seq: The sequence number of the packet. This information is leveraged to detect duplicate packets in the sending queue.

hop_count: This indicates the hop count of the packet. This information is utilized to find route inconsistency.

etx: The ETX value from the source node toward the root node. Neighbor nodes use this information to find better paths.

flag: This field is equivalent to flag field in CTDCP_ROUTING_PACKET (Fig. 5.5). The transmitting node may use FREQUENT_ROUTE_UPDATE and NODE_CONGESTED flag. This information is leveraged to request rapid link quality detection, and to notify neighbor nodes of congestion, respectively.

Packet Processor leverages a sending queue (SendQueue) and a sent cache (SentCache) to manage sending and forwarding packets. A packet is enqueued to the SendQueue when the application requests to transmit the packet, or when the local node needs to forward the received packet. Packet Processor determines the necessity of forwarding packets by verifying it with Routing Manager. Routing Manager informs Packet Processor

to forward packets when 1) the local node is a parent of the transmitting node in the ROUTING_ENTRY, 2) the transmitting node is source node in the SINK_ROUTING_ENTRY, and 3) the transmitting node is in the bypass_nodes of SINK_ROUTING_ENTRY.

After successful transmission of packets, Packet Processor moves the packet to the end of the SendQueue, and start the retransmit timer. This timer is utilized to wait certain amount of time for the acknowledgement of the transmitted packet. When the timer is invoked, Packet Processor dequeues the first packet in the SendQueue and transmits it. Packet Processor repeats this procedure until the SendQueue becomes empty. Upon reception of packets, Packet Processor checks if a duplicate packet is in the SendQueue by comparing the collection_id, source_addr, and seq field. When the received packet exists in the SentCache, it signifies that the child node did not receive the packet forwarded by the local node, therefore, Packet Processor unicasts the same packet to the child node as acknowledgement.

Another job of the Packet Processor is notifying the link quality detection layer of link layer information. Since each layer of operating systems should be independent from each other, link quality detection should not be directly accessing the information of data packets. Therefore, Packet Processor only provides the information of successful and unsuccessful delivery of packets, utilizing acknowledgements and retransmissions.

5.4 Implementation of ALQDP

This section describes our implementation of ALQDP. Fig. 5.11 illustrates the structure of ALQDP. LQ_Detector provides link quality information to LQ_Estimator, and also uses the estimated link quality provided by it. LQ_Detector uses packet exchange interfaces, and provides the same interfaces to the upper layers. It simply leverages outgoing and incoming packets to insert and retrieve link quality information.

ALQDP has two components: Link Quality Detector and Link Quality Estimator. The Link Quality Detector detects the link qualities based on both hardware and software information. In the meantime, Link Quality Estimator estimates the high accuracy software parameter from inexpensive hardware parameters.

5.4.1 Link Quality Detector Component

When Link Quality Detector determines the quality of links among nodes, it modifies the ETX calculation window size and EWMA's weight of most

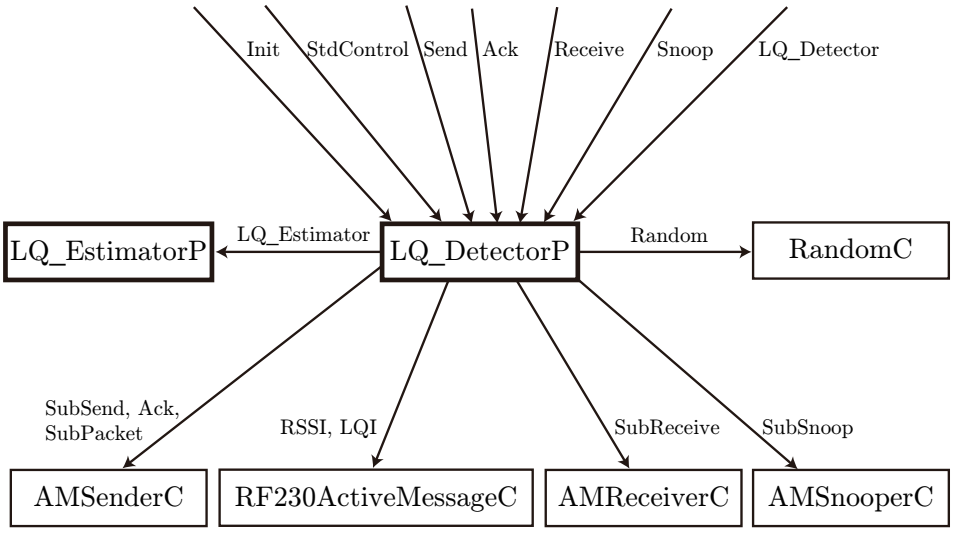


Figure 5.11: Wiring of ALQDP

recent data to meet the status of sensor nodes. We employ the idea of Trickle in ETX calculation window size to adapt to high dynamic links, while suppressing the calculation overhead. We define the minimum and the maximum window size as 3 and 10, respectively, based on the findings in 4B [28], and our experiment described in Sec. 4.3.1. Meanwhile, we define the weight of the most recent data in EWMA according to the mobility of the node: if either of the connected node has mobility, the weight is defined as 0.9 and if not, the weight is defined as 0.5. The weight of the most recent data is meaningful, however, we have not figured the confidential value for it, because at this point, we do not have much data with mobile nodes. Therefore, our future work is to discover the most suitable weight which works in variety of environment.

Fig. 5.12 illustrates the link quality entry. The parameters of LINK_QUALITY_ENTRY are described as follows:

addr: This indicates the link layer address of the corresponding node.

flag: The flag section indicates multiple state. This signifies the validity, pin, mobility, and updating information of the entry.

inquality: The inquality shows the ETX value from the target node toward the local node.

etx: This indicates the ETX value from the local node toward the target node.

last_seq: This indicates the sequence number of the previously received

```
typedef struct {
    am_addr_t addr;
    uint8_t flag;
    uint8_t inquality;
    uint8_t etx;
    uint8_t last_seq;
    uint8_t recv_packets;
    uint8_t fail_packets;
    uint8_t window_size;
    uint16_t rssi;
    uint16_t lqi;
} LINK_QUALITY_ENTRY;
```

Figure 5.12: Link Quality Entry Structure

packet.

recv_packets: The number of successfully received packets.

fail_packets: The number of packets which was not received.

window_size: This field remembers the window size of the entry.

rssi: RSSI value of previously received packet.

lqi: LQI value of previously received packet.

The link quality table is in charge of storing the number of successful and unsuccessful reception of packets. These values are leveraged to compute the ETX values. The window_size is leveraged to provide different ETX calculation window size for each target node. The rssi and lqi fields are not directly leveraged by Link Quality Detector; they are used as input parameters of Link Quality Estimator.

Fig. 5.13 shows the structure of link quality packet. We describe each member of it as follows.

– LQD_HEADER:

flag: This indicates the number of footer entries in the packet, and the mobility status.

seq: This indicates the sequence number, which is leveraged by neighbor nodes to calculate the ETX value.

– LQD_FOOTER_ENTRY:

link_addr: The link layer address of the associated node.

inquality: The inquality information, which is utilized to determine ETX value from the specified node toward the transmitting node.

– LQD_FOOTER:

```
typedef nx_struct {
    nx_uint8_t flag;
    nx_uint8_t seq;
} LQD_HEADER;

typedef nx_struct {
    nx_am_addr_t link_addr;
    nx_uint8_t inquality;
} LQD_FOOTER_ENTRY;

typedef nx_struct {
    LQD_FOOTER_ENTRY *entry_list;
} LQD_FOOTER;
```

Figure 5.13: Link Quality Packet Structure

*entry_list: List of link quality entries.

Link Quality Detector does not transmit packets on its own, instead, it adds header and footer to the transmitting packet. It leverages footer with variable length, which lengthens packets only when required. It has two roles: first, it computes the inquality and notifies the target node of it, and second, it updates link quality table according to the received packets.

Whenever packets are transmitted through Link Quality Detector, it inserts sequence number, and increments it after every transmission. The neighbor nodes which received the packet updates recv_packets and fail_packets by subtracting the last_seq from the received packet's sequence number, and updates last_seq. Data packets do not leverage Link Quality Detector, however, Packet Processor notifies it of successful and unsuccessful transmission of packets, which is also used to update recv_packets and fail_packets. In consequence, link quality detection leverages not only control packets, but also data packets to improve the accuracy of detecting link qualities. When the sum of recv_packets and fail_packets exceed window_size, Link Quality Detector calculates the inquality, and set UPDATE flag to the entry. The UPDATE flag requests Link Quality Detector to transmit ETX information to the target node. Link Quality Detector increments the number of footer entries in the flag, and adds corresponding footer entry to footer of the packet. Upon reception of packets, Link Quality Detector obtains the number of footer entries in the packet from the flag field. When Link Quality Detector finds an entry which has the local node's address as link_addr, it updates the etx field in the link quality table. Accordingly,

Link Quality Detector provides bidirectional ETX values of neighbor nodes. Whenever the node receives a packet with link quality information, Link Quality Detector informs Routing Manager of it, which is used to verify the effectiveness of bypassing.

We express the link quality in the form of ETX multiplied by 10, and rounding off the decimal. In this way, the output of ALQDP can be leveraged without involving costly decimal numbers.

5.4.2 Link Quality Estimator Component

This component provides estimated link qualities based on RSSI and LQI values. When the Link Quality Detector updates the ETX values, Link Quality Estimator inserts RSSI, LQI, and ETX into its estimation equation. After inserting W combination of information, Link Quality Estimator recomputes the equation, so that the other components can use the most updated data. We simply implemented the link quality estimation described in Sec. 4.3.2 as a standalone component with nesC.

5.5 Summary

In this Chapter, we describe our implementation of CTDCP and ALQDP. First, we clarified the platform used to implement our system on; we leverage Iris motes as hardware, and TinyOS2.x as a platform. We then describe overall structure of proposed two protocols, CTDCP and ALQDP. Finally, Implementation of CTDCP and ALQDP are described in detail.

Chapter 6

Evaluation

In this Chapter, we present the evaluation of proposed protocols. We conduct evaluations both in the real world environment and in simulation. For each of them, we provide the evaluation methodology, including evaluation environment, metrics and comparison targets, and then present its results and discussions.

6.1 Real World Evaluation

We conduct two types of real world evaluation to clarify the performance of 1) the basic data collection, and 2) the data collection with mobile sink nodes. Unless otherwise noted, values expressed in this Section are rounded to four significant figures.

6.1.1 Evaluation of the Basic Data Collection

Through this experiment, we examine the fundamental data collection performance achieved by the combination of CTDCP and ALQDP. Even though this thesis targets WSNs with mobile sink nodes, it is essential for data collection protocols to have high data collection efficiency in a WSN consisting only of static nodes. The reason is twofold. First, CTDCP leverages tree structures constructed with static nodes for data collection, hence, its data collection efficiency highly depends on the performance of the tree structure. Second, mobile sink nodes may be immobile for a long period of time, or CTDCP may be used only with statically placed sensor nodes. In these cases, data collection efficiency with static nodes is essential.

Evaluation Methodology

CTDCP leverages tree structures to deliver packets toward mobile sink nodes, therefore, we compare it against existing tree-based data collection protocols. We choose CTP [24] as the comparison target for two reasons: first, it is proven to achieve high data collection efficiency, and second, it is implemented as a TinyOS library in the main branch. For the link quality detection protocol of CTP, we use its default protocol, 4B [28].

We evaluate CTDCP and CTP in perspective of *A*) packet delivery ratio, *B*) the number of retransmissions, *C*) path length, *D*) energy consumption, and *E*) storage usage. Unless otherwise noted, we evaluate each item by deriving the average of all data.

A. Packet Delivery Ratio
Packet delivery ratio indicates the end-to-end packet delivery ratio between the sink node and each sensor node. The packet delivery ratio is calculated by dividing the number of received packets by that of transmitted packets. The delivery ratio is desired to be as high as possible.

B. The Number of Retransmissions
CTDCP and CTP utilize retransmissions to complement packet losses. We modified them to include a retransmission field in each packet, and increment its value after every successful transmission. Despite packet retransmission is essential in increasing the delivery ratio, it incurs high overhead not only at the sender nodes, but also at its neighbor nodes. In consequence, it is preferable to have high delivery ratio and the small number of retransmissions at the same time.

C. Path Length
Path length represents the number of hops between the source and the sink node. It is determined by incrementing the value of the hop count field whenever a node receives a packet. It is usually better to have shorter path length, because it is meaningful in 1) reducing the possibility of packet collisions, and 2) minimizing energy consumption at every forwarding node. In some cases, higher delivery ratio is achieved by leveraging a path with larger path length, however, it incurs additional overhead for forwarding packets. Therefore, it is desired to have high delivery ratio and shorter path length simultaneously.

D. Energy Consumption
One of the most notable characteristics of sensor nodes is having severe constraint on energy source, hence, it is important to minimize energy

Table 6.1: Real World Evaluation Environment for the Basic Data Collection

Parameter	Values
Sensor Nodes	21 Iris motes
Place	*o*208 at Shonan Fujisawa Campus, Keio University
Duration	Over 55 hours
Inter Packet Interval	5 seconds
MAC	B-MAC
Maximum Retransmissions	30
Routing Table Size	10
Routing Queue Size	13
Routing Cache Size	4
Link Quality Table Size	10

consumption. Energy consumption is calculated by subtracting the voltage of batteries at the end of the experiment from that at the beginning of it. Any kind of operation in sensor nodes consumes energy, therefore, energy consumption represents the overhead of the protocols. Energy consumption is affected by the characteristics of both sensor nodes and batteries, therefore, we use them as pairs throughout the evaluation.

E. Storage Usage
In general, wireless sensor nodes are equipped with small storage. In order to evaluate storage usage, we leveraged the data size of ROM and RAM required by the application used in the evaluation. The performance of data collection and link quality detection protocols are usually proportional to their table size. Therefore, it is preferable for protocols to require small storage, because the remaining storage may be utilized to increase the table size.

Table 6.1 shows the parameters used in the evaluation. One iris mote is leveraged as a sink node, which is connected to a personal computer through MIB520, a USB gateway. We deployed 21 Iris motes in the room *o*208 at Shonan Fujisawa Campus, Keio University. The inter packet interval is set to 5 seconds to verify the data collection performance, rather than their collision avoidance abilities. We used the default values of CTP and 4B for MAC protocol, maximum retransmissions, table size, queue size, and cache size. We let the evaluation run for at least 55 hours.

Table 6.2: Evaluation Results of the Basic Data Collection in the Real World Environment

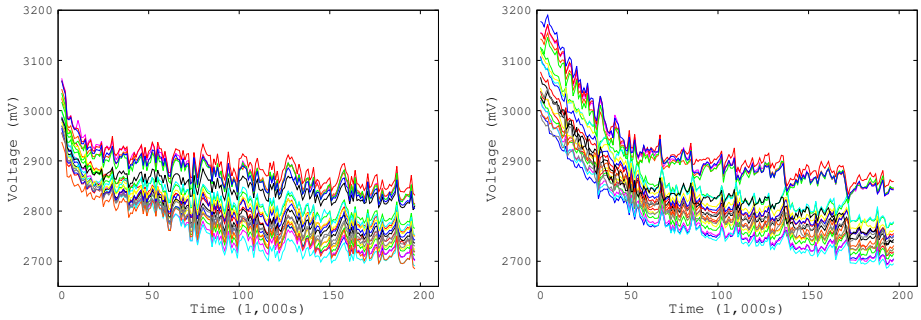
Parameter	Protocol	Result
Average Packet Delivery Ratio	CTDCP + ALQDP	1.000
	CTP + 4B	0.9999
Average Number of Retransmissions	CTDCP + ALQDP	0.03177 Retxs/packet
	CTP + 4B	0.03689 Retxs/packet
Average Path Length	CTDCP + ALQDP	1.210
	CTP + 4B	1.905
Average Energy Consumption	CTDCP + ALQDP	1.147μV/s
	CTP + 4B	1.805μV/s
Storage Usage	CTDCP + ALQDP	ROM: 38,502 bytes RAM: 6,789 bytes
	CTP + 4B	ROM: 34,832 bytes RAM: 4,585 bytes

Evaluation Results and Discussions

Table 6.2 shows the evaluation results of the basic data collection in the real world environment.

- Packet Delivery Ratio
  The delivery ratio is equivalent in CTDCP and CTP. There are two reasons they both achieved high delivery ratio. First, packets are barely damaged or lost in Iris mote, because it is equipped with high performance radio transceiver. Second, tree structure has high delivery ratio in data collection. It has been proven that tree structures perform well in data collection [24] [25], hence, it is understandable that they both achieved high delivery ratio.

- The Number of Retransmissions
  We confirmed that CTDCP has the slightly smaller number of retransmissions compared to CTP. The maximum number of retransmissions in CTDCP and CTP is 3 and 4, respectively. Since we defined the maximum number of retransmissions as 30 in each protocol, it is difficult to believe that some packets are lost in CTP due to excessive retransmissions. Consequently, we argue that packet losses in CTP are caused by the overflow of sending/forwarding queue, which implies that congestion control is poor in CTP.



(a) Energy Consumption in CTDCP + ALQDP

(b) Energy Consumption in CTP + 4B

Figure 6.1: Energy Consumption in the Basic Data Collection Evaluation

- Path Length
  We observed large differences in the average path length in CTDCP and CTP. This signifies the superiority of proposed protocols in terms of minimizing the overhead. We argue that there are two reasons that proposed protocols achieve shorter path length. First, the larger number of packets is exchanged in CTP, because it leverages ACK to acknowledge the reception of packets, meanwhile, CTDCP eliminates it by utilizing data packets as acknowledgements. As a result, sensor nodes in CTP have greater chance of overflowing their sending/-forwarding queue, which causes their children nodes to change their parent to a node with possibly worse link quality. Second, we believe that the use of Iris mote favors ALQDP over 4B, because the LQI value of RF230 (radio transceiver used in Iris mote) barely changes, and 4B only leverages LQI as a filter when inserting information of a node into its neighbor table. ALQDP is less affected by it, because it utilizes the combination of RSSI and LQI as a filter.

- Energy Consumption
  The evaluation results prove that the proposed protocols require substantially small amount of energy in comparison to the existing protocols. Fig. 6.1 illustrates voltage measured at each sensor node, where x-axis represents time in 1,000 seconds scale, and y-axis expresses voltage in millivolt. Each line in the figure represents energy consumption of each sensor node. As expressed in the figure, voltage of batteries have large jitter when measured at sensor nodes, hence, we use the average of all sensor nodes to minimize the errors. We observed

large jitter in voltage especially at the beginning of the experiment, therefore, we omit them in the energy consumption calculation.

The combination of CTDCP and ALQDP reduced energy consumption by approximately 35% in comparison to that of CTP and 4B. This is a significant improvement, because energy source is usually severely limited in sensor nodes. We believe there are two reasons that proposed protocols consume lower energy compared to the existing protocols. First, CTDCP has shorter path length and the smaller number of retransmissions, which signifies that it involves smaller number of packet exchanges. Transmission and reception of packets require large amount of energy compared to other operations, therefore, we argue that CTDCP reduces energy consumption by minimizing the number of packet exchanges. Second, we argue that ACK packets leveraged in CTP induces additional overhead. CTP utilizes unicast-based routing, and requires receiver nodes to reply acknowledgements upon the reception of packets. On the other hand, CTDCP utilizes broadcast-based routing, and leverages forwarded data packets as acknowledgements. Consequently, we argue that CTP consumes more energy for leveraging additional acknowledgement packet.

- Storage Usage
  The storage usage indicated in the table is the data size of the application used in the evaluation. Consequently, protocols themselves require much smaller storage, because the application utilizes large queue for transferring received packets to the personal computer. The combination of CTDCP and ALQDP requires additional approximately 2,200 bytes of RAM compared to that of CTP and 4B. We used the same parameters for the neighbor tables, queues, and caches in proposed and existing protocols, therefore, the results purely indicates their differences in storage usage. We believe there are two reasons that proposed protocols necessitate larger amount of storage. First, CTDCP is designed to support multiple mobile sink nodes, which requires additional neighbor table and bypass table. Second, CTDCP and ALQDP require large amount of storage for leveraging bypass tree, broadcast-based routing, and link quality estimation. Although differences in their data size is not negligible, we argue that the efficiency of data collection is more important. In addition, we believe that sensor nodes will be equipped with larger memory and storage in the near future, therefore, it is considered that large storage usage in proposed protocols is acceptable.

Table 6.3: Real World Evaluation Environment for Data Collection with Mobile Sink Nodes

Parameter	Values
Sensor Nodes	33 Iris motes
Number of Mobile Sink Nodes	1 - 3 nodes
Place	2F of Iota and Omicron at Shonan Fujisawa Campus, Keio University
Duration	12 - 32 hours

Through the evaluation, we confirmed that in comparison to existing protocols, the proposed protocols 1) accomplish the lower possibility of retransmissions, 2) achieve shorter path length, 3) require less energy, and 4) demand extra storage. Despite their large storage usage, we believe that strengths of proposed protocols surpass the disadvantage, because the magnitude of improvement is significant.

6.1.2 Evaluation of Data Collection with Mobile Sink Nodes

This section describes the real world evaluation conducted to clarify the performance of data collection with mobile sink nodes.

Evaluation Methodology

Through this experiment, we investigate the data collection performance of proposed protocols when multiple mobile sink nodes exist in the WSN. In this experiment, we choose to compare proposed protocols against CTP [24] and 4B [28]. Since CTP is designed for statically placed sensor/sink nodes, we compare its data collection efficiency at the static sink node against that of CTDCP at the static sink node as well as multiple mobile sink nodes.

We evaluate CTDCP and CTP in terms of A) packet delivery ratio, B) the number of retransmissions, C) path length, and D) energy consumption. We evaluate these parameters for the reasons explained in Sec. 6.1.1.

Table 6.3 shows the parameters modified from the evaluation explained in Sec. 6.1.1 We placed 33 Iris motes in the second floor of Iota and Omicron buildings and inside of the room ι208 and o208 at Shonan Fujisawa Campus, Keio University. Fig. 6.2 illustrates the map and placement of sensor nodes, where the square and circles indicate a sink node and sensor nodes, respectively. The numbers in the circles represent the ID of each sensor node. We kept the doors of ι208 and o208 opened when mobile sink nodes existed, and

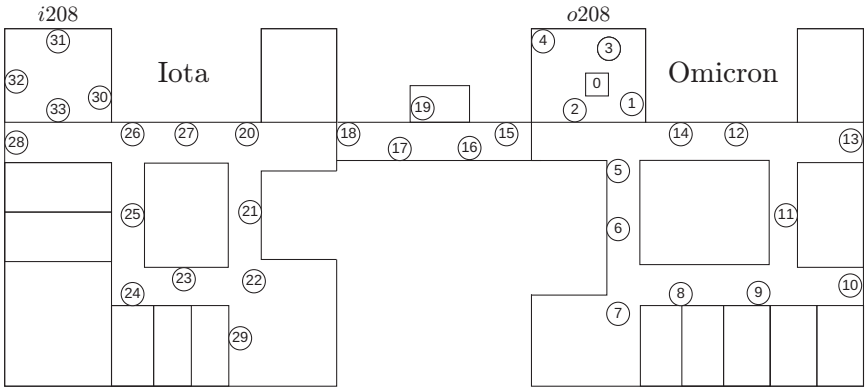


Figure 6.2: Deployment of Sensor Nodes

in the remaining time, we left them as is. In this experiment, we asked 18 participants to hold an Iris mote and a personal computer as a sink node, and act freely in the coverage the network. Participants include 5 females and 13 males, who are aged from 18 to 24. We asked each participant to collect data for at least 10 minutes.

Evaluation Results and Discussions

We conduct the experiment in three consecutive days: we evaluated proposed protocols on the first and second days, and evaluated existing protocols on the last day. Each experiment is conducted for approximately 12.5, 19.5, and 32.5 hours. We had 13 and 5 participants on the first and second day, respectively. Mobile sink nodes collected data over four hours in total.

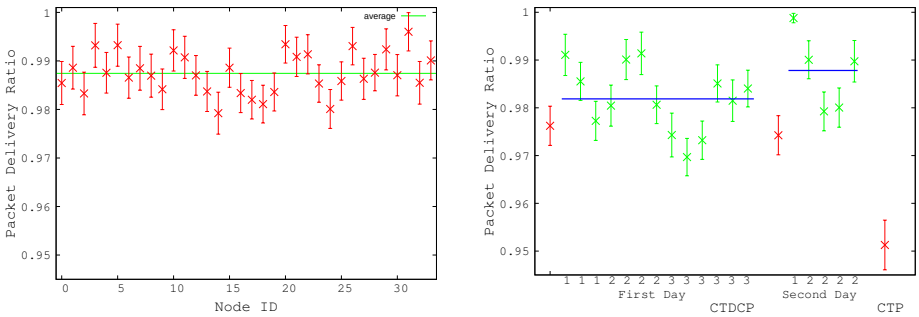
Table 6.4 shows the evaluation results of data collection with mobile sink nodes in the real world environment. The evaluation results expressed in the table are calculated from the data acquired at statically placed sink node. Note that the results of CTDCP and ALQDP are the average of two days.

- Packet Delivery Ratio

  We confirmed that CTDCP outperforms CTP in perspective of delivery ratio. Multiple environmental factors could be affecting the delivery ratio, however, we argue that proposed protocols are superior to existing protocols for the following reasons. First, the doors of the room $\iota 208$ and $o208$ may become obstacles and cause packet losses. Nevertheless, the delivery ratio of every sensor node remained constant throughout the evaluation, hence, we argue that the state of the doors did not affect the delivery ratio. Second, existence of hu-

Table 6.4: Evaluation Results of Data Collection with Mobile Sink Nodes in the Real World Environment

Parameter	Protocol	Result
Average Packet Delivery Ratio	CTDCP + ALQDP	0.9755
	CTP + 4B	0.9513
Average Number of Retransmissions	CTDCP + ALQDP	0.2259 Retxs/packet
	CTP + 4B	0.6585 Retxs/packet
Average Path Length	CTDCP + ALQDP	1.721
	CTP + 4B	2.666
Average Energy Consumption	CTDCP + ALQDP	2.602μV/s
	CTP + 4B	4.021μV/s



(a) Packet Delivery Ratio at Mobile Sink Nodes

(b) Packet Delivery Ratio at Different Sink Nodes

Figure 6.3: Packet Delivery Ratio at Mobile Sink Nodes

mans might have become obstacles. However, the first two days were weekdays and the last day was a weekend, therefore, this factor would favor existing protocols over proposed protocols. In the meantime, we observed significant differences in the number of retransmissions and the path length in proposed and existing protocols. These factors have significant influences on the delivery ratio, therefore, we argue that differences in the delivery ratio are caused by the differences of their data collection performance.

Fig. 6.3 illustrates the packet delivery ratio derived at mobile sink nodes. In each figure, x indicates the average delivery ratio, and upper and lower bounds represent its standard deviation. Fig. 6.3(a)

shows the packet delivery ratio in perspective of each sensor node, where x-axis indicates the sensor nodes' ID, and y-axis represents the delivery ratio. The green horizontal line indicates the average of all nodes. We did not observe any significant difference in the delivery ratio among sensor nodes, therefore, it is considered that CTDCP successfully delivered packets from all sensor nodes toward multiple mobile sink nodes. Fig. 6.3(b) illustrates the delivery ratio at static and mobile sink nodes, where the red and green bars indicates their delivery ratio, respectively. Note that the numbers of the x-axis indicate the number of mobile sink nodes simultaneously existed in the network, and the blue horizontal line represents the average delivery ratio at mobile sink nodes. As shown in the figure, we confirmed that mobile sink nodes have higher packet delivery ratio in comparison to static sink nodes. For example, average delivery ratios at mobile sink nodes are 0.9819, and 0.9878 on the first and second day, respectively.

There are three reasons that mobile sink nodes outperform the static sink node in perspective of delivery ratio; two from environmental factors, and one from performance superiority. First, we kept the doors of the room $\iota 208$ and $o208$ opened when mobile sink nodes were collecting packets. Therefore, mobile sink nodes might performed well for collecting packets from sensor nodes placed inside of the rooms without having doors as obstacles. Second, the static sink node was placed inside of room $o208$, meanwhile, mobile sink nodes move freely in the area of the network. Therefore, mobile sink nodes might had better link qualities between other sensor nodes. Finally, higher delivery ratio may be achieved by leveraging CTDCP and ALQDP. In CTDCP, bypass tree enables sensor nodes to forward packets through the most efficient paths, which is greatly appreciated by mobile sink nodes. Meanwhile, by leveraging ALQDP, mobile sink nodes can instantly estimate the link qualities between neighbor nodes, which enables them to collect packets through links with high quality aggressively.

- The Number of Retransmissions
  We confirmed large differences in the number of retransmissions between proposed and existing protocols. Since the number of retransmissions signifies the overhead of data collection, we conclude that the proposed protocols have lower data collection overhead compared to existing protocols. Note that not all packets are delivered to the sink node, hence, the actual number of retransmissions is larger than those indicated in the table. Fig. 6.4 depicts the CDF of the number

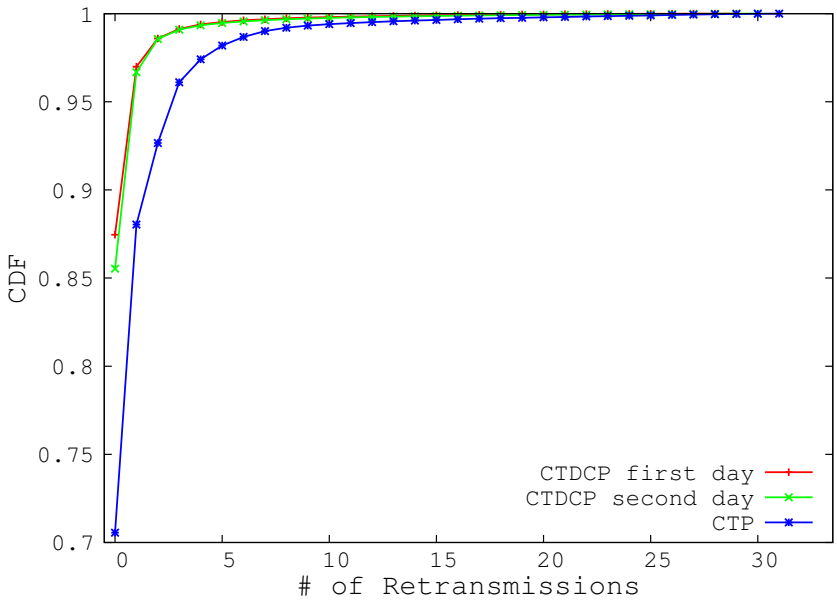
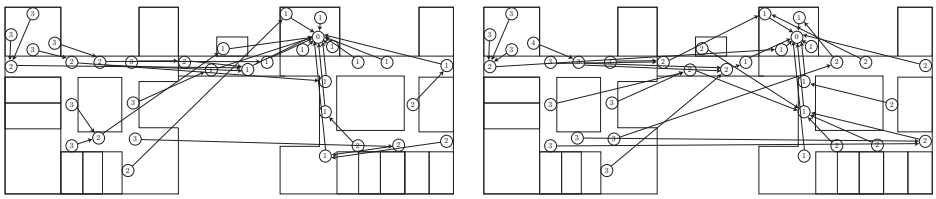


Figure 6.4: CDF of the Number of Retransmissions

of retransmissions, where x-axis indicates the number of retransmissions, and y-axis represents the CDF. Approximately 85% and 70% of packets are delivered to the sink node without involving any retransmissions in CTDCP and CTP, respectively. This signifies the superiority of CTDCP in perspective of the retransmission overhead.

CTP may have larger number of retransmissions due to link failures. However, throughout the experiment, we did not observe any significant degradation in the delivery ratio, hence, we believe that no link failure occurred. We argue that CTP requires a large number of retransmissions due to its poor congestion control. When a node has a lot of children nodes, it has to forward a large number of packets, which increases the possibility of packet collisions and incurs a large number of retransmissions. Therefore, we believe that the larger number of packet retransmissions in CTP is caused by its poor load distribution.

- Path Length
  The evaluation results prove that proposed protocols outperform existing protocols in perspective of path length. Fig. 6.5 illustrates the routing topology in the middle of the experiment, where numbers in the circles indicate the path length. We argue that proposed protocols achieved shorter path length for minimizing the number of packet



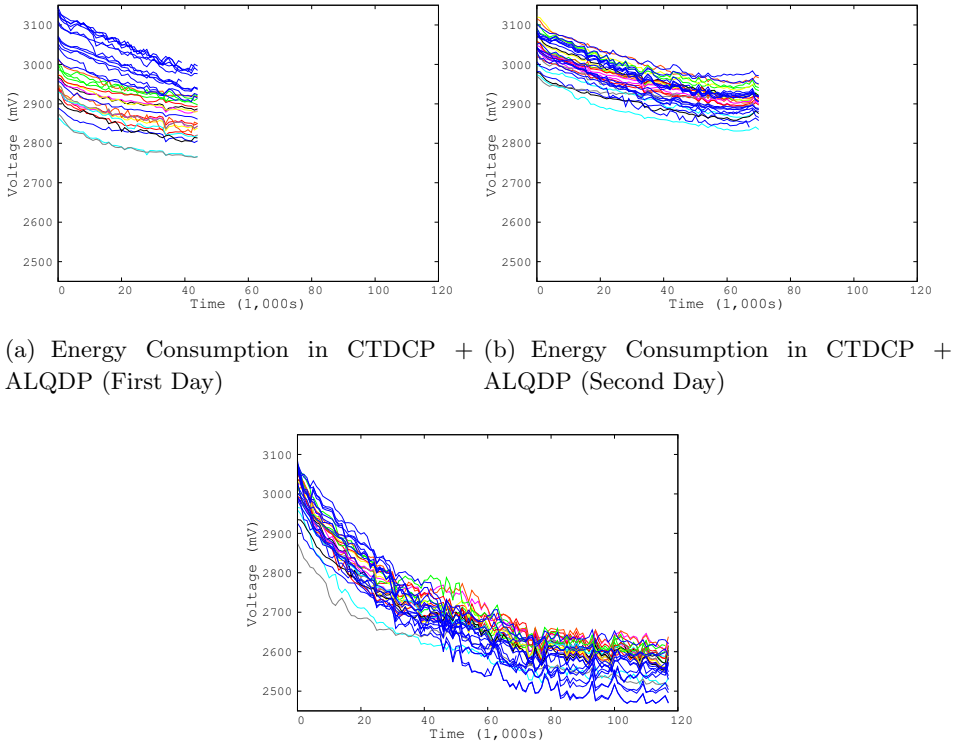
(a) Routing in CTDCP + ALQDP (b) Routing in CTP + 4B

Figure 6.5: Routing in the Real World Evaluation

transmissions. The overhead for replying acknowledgements is large in existing unicast-based protocols. On the contrary, CTDCP leverages forwarded data packets as acknowledgements, which reduces the number of transmissions at each forwarding node. Consequently, in CTDCP, forwarding nodes have less chance of becoming congested, therefore, sensor nodes can utilize nodes with better link qualities as their parents.

- Energy Consumption
  We confirmed significant difference in energy consumption in proposed and existing protocols. Energy consumption indicates the overhead, therefore, we conclude that the proposed protocols are superior to existing protocols in terms of data collection overhead. Fig. 6.6 illustrates energy consumption of each sensor node during the experiment, where x-axis represents time in 1,000 seconds scale, and y-axis indicates voltage in millivolt. Each line in the figure represents energy consumption of each sensor node. We argue that sensor nodes in CTP consume large amount of energy for exchanging greater number of packets. Through the evaluation, we confirmed that sensor nodes in CTP have longer path length and larger number of retransmissions in comparison to CTDCP. Moreover, CTP leverages ACK, which increases the number of exchanged packets. Since packet exchange requires large amount of energy compared to other operations, it is understandable that CTP consumes greater amount of energy compared to CTDCP.

  We measured energy consumption at mobile sink nodes, however, we could not derive accurate values due to instability in measured voltage. Voltage measured at sensor nodes shape sawtooth (See Fig. 6.6), hence, it is difficult to determine their energy consumption accurately. In fact, we calculated energy consumption at mobile sink nodes, and observed increase in voltage for some participants. This implies that, when the



(a) Energy Consumption in CTDCP + ALQDP (First Day)

(b) Energy Consumption in CTDCP + ALQDP (Second Day)

(c) Energy Consumption in CTP + 4B

Figure 6.6: Energy Consumption in Data Collection with Mobile Sink Nodes

experiment is conducted for short period of time, energy consumption is not a valid metric for clarifying the overhead.

The evaluation results prove that in comparison to existing protocols, the proposed protocols 1) increased the delivery ratio by approximately 2.5%, 2) reduced the number of retransmissions by roughly 66%, 3) minimized path length by about 35%, and 4) reduced energy consumption by approximately 35%. These results signify that the combination of CTDCP and ALQDP achieves high data collection efficiency. In addition, we confirmed that mobile sink nodes collected packets with higher delivery ratio compared to the static sink node, which indicates the superiority of CTDCP in managing mobile sink nodes.

6.2 Simulation Evaluation

This section describes the simulation evaluation. We conduct two types of experiments in order to clarify the performance of CTDCP against existing protocols, and the effectiveness of clustered-tree structure. For each experiment, we explain the evaluation methodology, results, and discussions.

There are a lot of simulators for simulating WSNs, such as TOSSIM [41], OMNET++, ns2, ns3, QualNet, SENSE, etc. There are two factors we should consider when choosing a simulator. First, in order to compare proposed protocols against others fairly, it is important to use widely used simulator. In the field of WSNs, a lot of researchers have been leveraging TOSSIM, OMNET++, and ns2 as a simulation tool. Second, it is essential to have compatibility with the actual implementation, because behaviors of protocols may change depending on the platform, programming language, etc. In terms of compatibility, only TOSSIM and OMNET++ supports nesC as programming language and have compatibility with TinyOS. However, OMNET++ cannot simulate TinyOS2.x, which is the latest version of TinyOS. For these reasons, we leverage TOSSIM throughout the simulation evaluation.

6.2.1 Evaluation Against Existing Protocols

This section describes the simulation evaluation conducted to examine the performance of the proposed protocols against existing protocols.

Evaluation Methodology

We choose Sidewinder [17] as the comparison target for two reasons as follows. First, although Sidewinder is a location-based data collection protocol, it outperforms existing protocols in perspective of the delivery ratio, time delay, and energy efficiency. Second, it utilizes TOSSIM in its evaluation, and it is compared against state of the art, high performance data collection protocols. FROMS [25] is another candidate, because it is similar to CTDCP in terms of utilizing multicast and implementing with nesC. However, we choose not to use it as a comparison target, because its authors use OMNET++ in the evaluation, and their presentation of evaluation is not clear. In Sidewinder, each sensor node leverages the idea of Sequential Monte Carlo theory to continuously estimate sink node's location, and forwards packets according to it. Authors of Sidewinder argue that relying on a single sensor node to forward packets lacks reliability, and employ the 60° rule proposed by Heissenbüttel et al [18]. In Sidewinder, every sensor node

which received a packet determines the 60° zone of the transmitted node, and compete for next hop forwarding.

We evaluate proposed protocols against Sidewinder in perspective of A) packet delivery ratio, B) number of transmissions per each source, and C) time delay. Unless otherwise noted, we evaluate each item by deriving the average of all data.

A. Packet Delivery Ratio
This indicates end-to-end packet delivery ratio between the source and the sink node. The packet delivery ratio is derived by dividing the number of received packets by that of transmitted packets. The packet delivery ratio is one of the most fundamental factors for data collection, therefore, this value is desired to be as high as possible.

B. Number of Transmissions per Each Source
One of the most major methodologies to evaluate the overhead is to measure the energy consumption, however, it is not preferable when utilizing simulators. For instance, Antonopoulos et al. measured energy consumption of various operations in wireless sensor node, and concluded that the characteristics of energy consumption on actual testbed differs from that in the data sheet and/or in simulators [15]. Consequently, we argue that energy consumption measured in TOSSIM is not a good metric for evaluating the overhead.

In this experiment, we choose to use the number of transmissions per each source as a metric of the overhead. This is the summation of the path length, the number of retransmissions, and the routing overhead. Since data collection protocols are designed to improve the efficiency of data collection, it is reasonable to evaluate protocols based on the number of transmissions per each source.

C. Time Delay
Time delay indicates the time required for a packet to be delivered to the sink node since it leaves the source node. It is derived by subtracting the reception time of a packet from the transmission time of it. Time delay is essential especially in real time applications, because they require latest data as quickly as possible.

Table 6.5 shows the parameters used in the experiment. We mimic the simulation parameters used in Sidewinder. 500 nodes are randomly placed in the area of $215m$ x $215m$. We used the radio model of Micaz mote [42], and define the radio range as $25m$. We used Random Waypoint without pause

Table 6.5: Simulation Parameters Used in the Evaluation Against Existing Protocols

Parameter	Values
Area Size	$215m$ x $215m$
Radio Range	$25m$
Mobility Model	Random Waypoint without pause time
MAC	B-MAC
Duration	1 hour
Inter Packet Interval	1 second
Number of Sensor Nodes	500 nodes
Number of Source Nodes	3 nodes
Number of Sink Nodes	1-3 nodes
Sink Node's Speed	$\{0,2,4,6,8,10\}m/s$

time [43] as the mobility model, and modified the speed of sink nodes from 0 to $10m/s$ with step of $2m/s$. The number of sink nodes is modified between one and three, and the number of source node is fixed to three. Sink and source nodes are randomly selected among the whole network before each simulation. We used B-MAC [44] as the MAC protocol, which is the default CSMA-based protocol used in TinyOS. The duration of simulation is set to one hour, and the inter packet interval is fixed to one second. We repeat the simulation 30 times, and evaluate each item by deriving the average of all simulations. Note that for each simulation cycle, we used newly generated sensor nodes' placement and mobility model. We did not set any limit to the maximum number of clusters in CTDCP, therefore, a single tree structure is constructed among the whole network. Before we start collecting packets with mobile sink nodes, we let the simulation run for an hour in CTDCP, so that it can construct structures with static nodes. We could not obtain the source code of Sidewinder, therefore, we use its evaluation results in our evaluation.

Evaluation Results and Discussions

We present the evaluation results of simulation evaluation against Sidewinder as follows:

- Packet Delivery Ratio

  Fig. 6.7 illustrates the evaluation results of the packet delivery ratio,

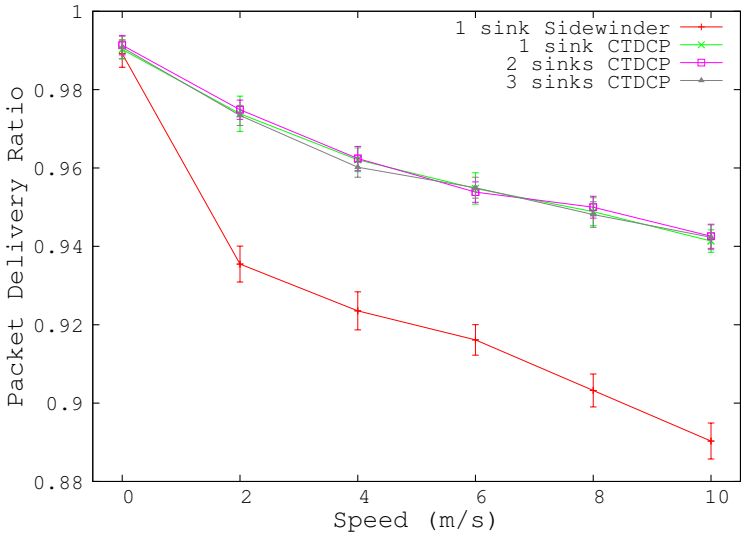


Figure 6.7: Packet Delivery Ratio in Simulation Evaluation Against Existing Protocols

where x-axis indicates the speed of sink nodes, and y-axis represents the end-to-end delivery ratio. Throughout the experiment, we observe higher delivery ratio in CTDCP compared to Sidewinder. Moreover, the evaluation results prove that the differences of the delivery ratios increase as the speed of mobile sink nodes increases. In the meantime, the evaluation results indicate that the delivery ratio decreases as the speed of mobile sink nodes increases. In CTDCP, when sink nodes have higher speed, the frequency of changing the forwarding node increases, which induces higher chance of packet collisions. Meanwhile, in Sidewinder, as the speed of mobile sink nodes increases, the number of hops increases, which incurs higher chance of packet collisions. Accordingly, the relationships between the speed of mobile sink nodes and packet delivery ratio is clear.

We did not observe any degradation in the delivery ratio in accordance to the number of sink nodes. This result seems controversial, since aggregating packets to multiple sinks nodes most likely cause additional packet collisions. However, in this experiment, the number of source nodes is limited to 3 out of 500 nodes, therefore, we believe that the chance of packet collision is small.

There are two reasons CTDCP outperforms Sidewinder in terms of delivery ratio. First, it is considered that the use of link quality-based

routing is superior to distance-based routing. Since the distance or the number of hops between two nodes does not necessarily indicate the link quality between them, link quality-based routing performs better for leveraging paths with higher quality. Second, it is considered that the tree-based data collection leveraged in CTDCP enables it to collect packets more efficiently compared to geographical data collection employed in Sidewinder. This is understandable, because data collection protocols based on tree structures are proved to have high delivery ratio [24] [25].

CTDCP achieved higher delivery ratio compared to Sidewinder, however, the magnitude of improvement seems to be small when considering the difference in their overhead. CTDCP successfully reduced the number of transmissions per each source by approximately 43% in comparison to Sidewinder. The number of transmissions per each source represents the summation of the path length, the number of retransmissions, and routing overhead, therefore, high delivery ratio achieved by Sidewinder seems controversial. However, a large portion of the overhead in Sidewinder is caused by its approach of leveraging multiple nodes competing for the next hop forwarding. Its approach has been proven to have high delivery ratio [45] [18], therefore, it is understandable that Sidewinder achieves high delivery ratio along with high overhead.

- Number of Transmissions per Each Source

  Fig. 6.8 illustrates the evaluation results of the number of transmissions per each source, where x-axis indicates the speed of mobile sink nodes, and y-axis represents the number of transmissions per each source. Note that the results may change depending on the beaconing frequency in Sidewinder. Since we did not evaluate Sidewinder with multiple mobile sink nodes, we simply multiplied the evaluation results of one sink node by the number of sink nodes. This is reasonable, because Sidewinder does not suppose multiple mobile sink nodes, and it requires independent paths between source nodes and each sink node. On the other hand, CTDCP enables sensor nodes to deliver a packet to multiple destination with a single transmission by leveraging broadcast-based routing. Therefore, the superiority of CTDCP increases as the number of sink nodes increases.

  CTDCP decreases the number of transmissions per each source by 30 - 50% (approximately 43% in average) in comparison to Sidewinder. The number of transmissions per each source indicates the overhead of

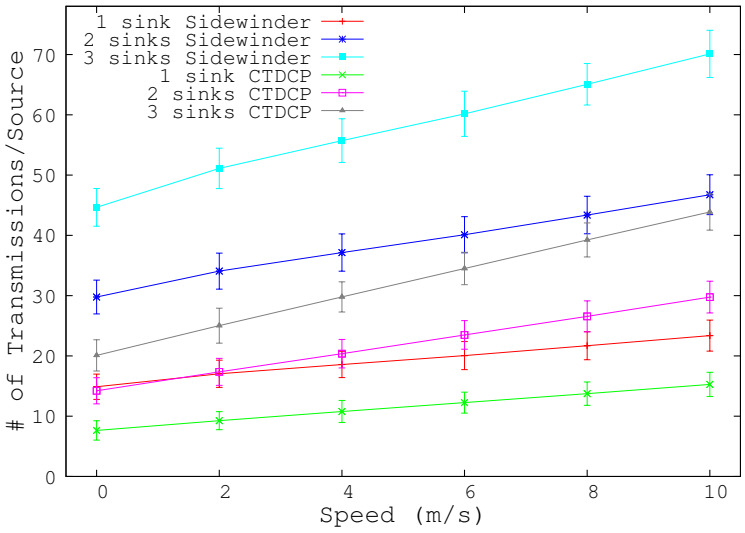


Figure 6.8: The Number of Transmissions per Each Source in Simulation Evaluation Against Existing Protocols

data collection. Accordingly, we conclude that CTDCP outperforms Sidewinder in terms of data collection overhead. Since transmission and reception of packets consume considerable amount of energy, the evaluation result implies that CTDCP is superior to Sidewinder in terms of energy consumption as well.

There are three reasons that CTDCP achieved lower data collection overhead compared to Sidewinder. First, CTDCP utilizes more efficient paths by leveraging link quality-based routing. Since Sidewinder utilizes geographic-based routing, it has slightly shorter path length. However, the distance or the number of hops between two nodes does not necessarily indicate the link quality between them. Therefore, Sidewinder has higher overhead for leveraging paths with worse link qualities. Second, CTDCP eliminates the need of replying acknowledgements by leveraging data packets as acknowledgements. This is meaningful, because the overhead of independent acknowledgements is large. Finally, Sidewinder induces high overhead for utilizing multiple nodes competing for the next hop forwarding. Sidewinder leverages this approach, because it is proved to have high delivery ratio in the field of geographic routing [45] [18], however, its overhead is significant.

- Time Delay

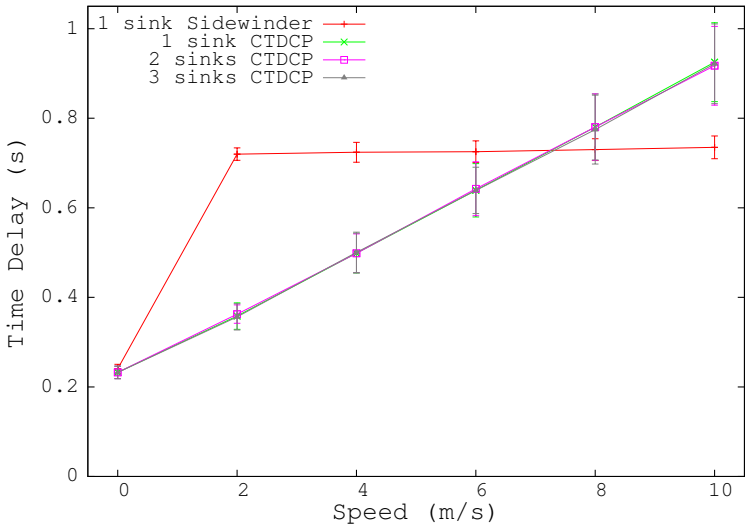


Figure 6.9: Time Delay in Simulation Evaluation Against Existing Protocols

Fig. 6.9 depicts the evaluation results of the time delay, where x-axis represents the speed of mobile sink nodes, and y-axis indicates the time delay in second precision. In Sidewinder, the time delay drastically increases when sink nodes has mobility, however, it remains constant irrespective of the speed of mobile sink nodes. On the contrary, in CTDCP, the time delay linearly increases as the speed of mobile sink nodes increases. As shown in the figure, in comparison to Sidewinder, CTDCP has lower time delay when the speed of a mobile sink node is slower than approximately $7m/s$.

Throughout the evaluation, we did not observe any relationships between the time delay and the number of mobile sink nodes. The time delay is affected by three factors as follows. First, the path length directly affects the time delay, because, in general, as the number of hops increases, the time delay also increases. However, path length remained constant when the number and/or the speed of mobile sink nodes are modified, therefore, we believe that the path length has small effect on the time delay. Second, retransmissions may induce additional time delay. Although, the time delay of Sidewinder remained constant when its delivery ratio decresed. Therefore, we argue that retransmissions has small influence on the time delay. Finally, mobility of sink nodes may incur extra time delay. In CTDCP, the frequency of changing forwarding nodes is proportional to the speed of mobile sink

nodes. Therefore, it is understandable that the time delay increases as the speed of mobile sink nodes increases. In contrast, Sidewinder maintains constant time delay by aggressively estimating sink nodes locations. This, in turn, implies that CTDCP can also suppress the time delay by employing sink node's location estimation.

We confirmed different characteristics in the time delay in accordance with the speed of mobile sink nodes in CTDCP and Sidewinder: in CTDCP, the time delay linearly increases as the speed of mobile sink nodes increases, while in Sidewinder, it remains constant. This thesis assumes humans, animals, and robots as mobile sink nodes, which barely move fast. Therefore, we conclude that CTDCP performs better in most cases.

The evaluation results prove that in comparison to Sidewinder, the combination of CTDCP and ALQDP achieves 1) higher delivery ratio, and 2) lower overhead. This, in turn, shows that proposed protocols outperform existing protocols in perspective of data collection efficiency. We also confirmed that, as the speed of mobile sink nodes increases, time delay linearly increases and remains constant in CTDCP and Sidewinder, respectively. Our thesis targets humans, animals, and robots as mobile sink nodes, hence, we conclude that CTDCP has lower time delay in the majority of cases.

6.2.2 Evaluation of Clustered-Tree Structure

This section describes the simulation evaluation conducted to examine the effectiveness of clustered-tree structure.

Evaluation Methodology

Through this experiment, we examine the performance of data collection protocols based on cluster, tree, and clustered-tree structure. All of them are based on CTDCP; we simply modified the depth and size of each cluster. Cluster structure is formed by limiting the depth of each tree to 1. By doing so, a WSN is split into multiple clusters, and each cluster is formed as a single hop network. When creating tree structures, we unset the limit of the number of nodes involved in each cluster. In this way, CTDCP creates a single tree structure with the whole WSN. Finally, we limit the number of nodes in each cluster by defining CTDCP_MAX_CLUSTER_SIZE. By doing so, CTDCP form a clustered-tree structure among the WSN.

We evaluate these structures in perspective of A) path length, B) packet delivery ratio, and C) number of transmissions per each source. Unless

Table 6.6: Simulation Parameters Used in the Evaluation of Clustered-Tree Structure

Parameter	Values
Area Size	2,000m x 2,000m
Radio Range	300m
Mobility Model	Random Waypoint without pause time
MAC	B-MAC
Duration	5 hours
Inter Packet Interval	300 seconds
Number of Sensor Nodes	500 nodes
Number of Source Nodes	500 nodes
Number of Sink Nodes	1-3 nodes
Sink Node's Speed	{0,2,4,6,8,10}m/s

otherwise noted, we evaluate each item by deriving the average of all data.

A. Path Length
The path length indicates the number of hops. We acquire the path length by modifying CTDCP to include a path length field in each packet, and increment its value whenever a node receives a packet. We evaluate the path length, because it indicates the overhead of each structure.

B. Packet Delivery Ratio
Packet delivery ratio indicates the end-to-end delivery ratio between source nodes and the sink node. It is calculated by dividing the number of received packets by that of transmitted packets.

C. Number of Transmissions per Each Source
We leverage the number of transmissions per each source for the same reasons explained in Sec. 6.2.1. In this experiment, all comparison targets are based on CTDCP, hence, the computation overhead should be equivalent. Consequently, the difference in the number of transmissions per each source accurately indicates the overhead difference of each structure.

Table 6.6 shows the parameters leveraged in this experiment. 500 nodes are randomly placed in the area of 2,000m x 2,000m. We used the radio model of Iris mote [39], and defined the radio range as 300m. For the mobility model, we used Random Waypoint without pause time [43], and

altered the speed of sink nodes from 0 to $10m/s$ with step of $2m/s$. The number of sink nodes is changed between one and three, and we utilized all sensor nodes as source nodes. We used B-MAC [44], the default CSMA-based protocol, as the MAC protocol. The duration of simulation is set to 5 hours, and the inter packet interval is fixed to 300 seconds. We repeat the simulation 10 times, and evaluate each item by deriving the average of all simulations. We generate new sensor nodes' placement and mobility model for each simulation, and every comparison target leverages the same generated set in a single simulation cycle. Before we start collecting data, we let the simulation run for an hour, so that CTDCP can construct each structure.

In this experiment, we divide the network into 5 clusters in clustered-tree structure, in other words, 5 cluster structures are formed, and there are 100 nodes in each of them. At this point, we are not confident about the suitable cluster size for clustered-tree structure. However, Gnawali et al. deployed CTP on multiple testbeds [24], and according to their results, 100 nodes per each cluster is considered to be appropriate.

Evaluation Results and Discussions

This section presents the evaluation results and discussions of simulation evaluation of different structures.

- Path Length
  Fig. 6.10 illustrates the average path length of each structure, where x-axis indicates the speed of mobile sink nodes, and y-axis represents the path length. We observed equivalent path length in tree and clustered-tree structure. On the other hand, the path length of cluster structure is considerably long compared to the others. In cluster structure, each cluster head has to be able to communicate with at least another cluster head. In addition, the path length between a cluster head and its member is limited to one, hence, the path length among cluster heads is considerably large. Accordingly, it is understandable that the path length is long in cluster structure.

- Packet Delivery Ratio
  Fig. 6.11 illustrates the evaluation results of the packet delivery ratio, where x-axis represents the speed of mobile sink nodes, and y-axis indicates the delivery ratio. Throughout the evaluation, we confirmed that the delivery ratio decreases as the number of sink nodes increases, and the speed of mobile sink nodes increases. In this experiment, every

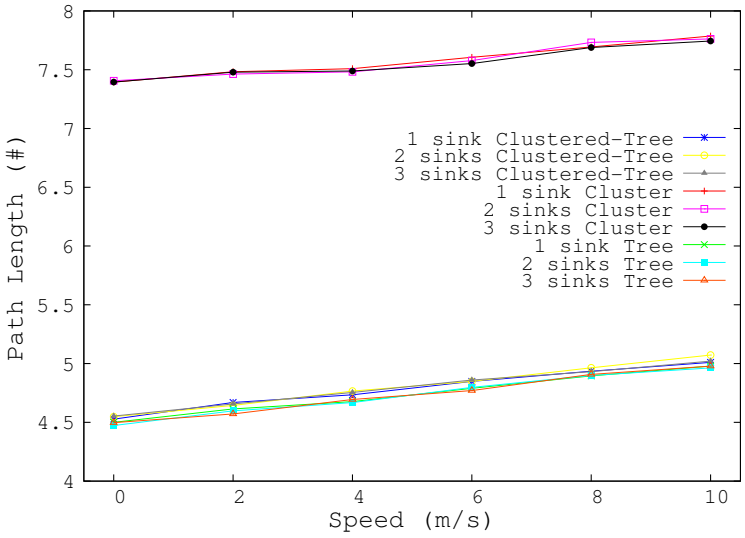


Figure 6.10: Path Length in Simulation Evaluation of Clustered-Tree Structures

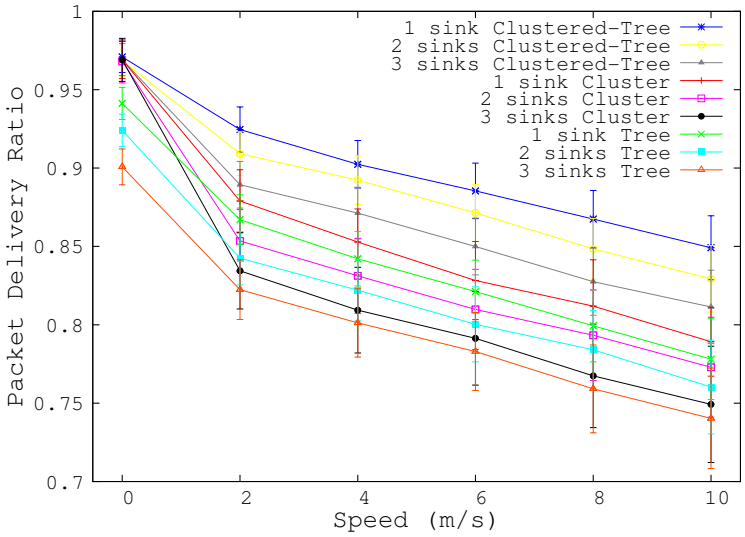


Figure 6.11: Packet Delivery Ratio in Simulation Evaluation of Clustered-Tree Structures

sensor node reports data periodically toward sink nodes, hence, the possibility of packet collisions increases proportionally as the number of sink nodes increases. Consequently, the delivery ratio decreases as the number of sink nodes increases. When the speed of mobile sink nodes increases, frequency of changing forwarding node also increases. Path modification requires large number of routing packet exchange, therefore, when the speed of mobile sink nodes is high, the delivery ratio decreases due to higher chance of packet collisions.

Through the evaluation, we confirmed that tree structure has low delivery ratio compared to the others. In tree structure, larger number of packets are being discarded due to the overflow of sending queue at sensor nodes near the root node of the network. In cluster and clustered-tree structures, overhead of forwarding packets is thoroughly distributed among multiple sensor nodes. On the contrary, in tree structure, a small number of nodes has to forward an excessive number of packets for utilizing a single tree structure. The delivery ratio when sink nodes have no mobility best illustrates this; the delivery ratio in tree structure is low compared to the others, and it decreases as the number of sink nodes increases. In clustered-tree structure, the network is divided into multiple clusters, hence, sensor nodes near the root node of each cluster have small number of packets to forward, which reduces the possibility of queue overflow.

When sink nodes have no mobility, the delivery ratio of cluster and clustered-tree structure is equivalent, however, when sink nodes have mobility, cluster structure performs worse. In cluster structure, each cluster is small, hence, mobile sink nodes have greater chance of selecting a new forwarding node from different clusters. When changing forwarding nodes, packets aggregated to the old forwarding node may have to be delivered to the new forwarding node, which induces large number of packet collisions. Consequently, it is considered that lower delivery ratio in cluster structure is caused by the overhead of changing forwarding nodes.

- Number of Transmissions per Each Source
  Fig. 6.12 indicates the evaluation results of the number of transmissions per each source, where x-axis represents the speed of mobile sink nodes, and y-axis shows the number of transmissions per each source, respectively. Through the evaluation, we confirmed that the number of transmissions per each source is proportional to the speed of mobile sink nodes. The number of transmissions is affected by two factors.

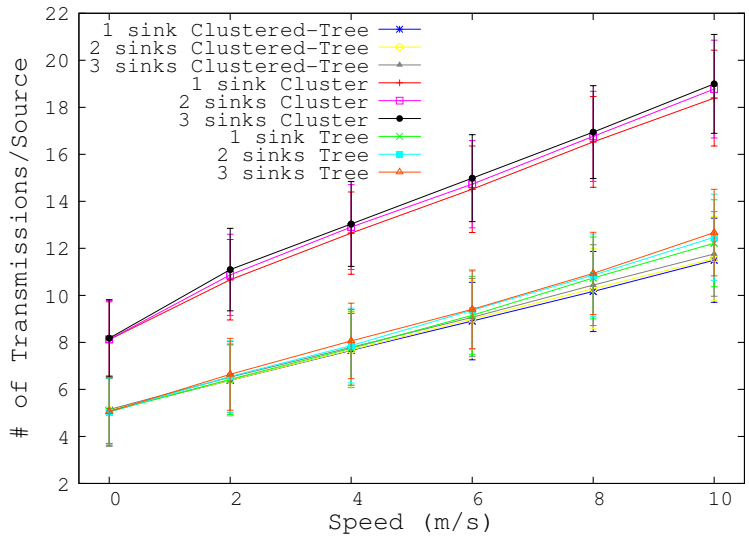


Figure 6.12: Number of Transmissions per Each Source in Simulation Evaluation of Clustered-Tree Structures

First, the control overhead becomes large when the speed of mobile sink node is high. When a mobile sink node decides to change the forwarding node, it transmits a control packet toward the head of the cluster. The frequency of changing forwarding node is proportional to the speed of mobile sink nodes, hence, the number of transmissions increases proportionally to the speed of mobile sink nodes. Second, when the delivery ratio is low, the number of retransmissions increases. Accordingly, we argue that the number of transmissions increases as the delivery ratio decreases.

Cluster structure has considerably large number of packet transmissions compared to others. The number of transmissions per each source is the summation of the path length, the number of retransmissions, and the routing overhead. Consequently, we argue that cluster structure has large number of transmissions per each source due to its longer path length, and the overhead for managing mobile sink nodes.

The evaluation results prove that 1) cluster structure has decent delivery ratio and high overhead, 2) tree structure has low delivery ratio and low overhead, and 3) clustered-tree structure has high delivery ratio and low overhead. Consequently, we conclude that clustered-tree structure is superior to others in perspective of data collection efficiency.

In this experiment, we divide the network into five clusters in clustered-tree structure. Although the decision is made based on the findings of existing work, there remains a room for improvement. When dividing a network into clusters, we should consider multiple factors, such as, the number of source/sink nodes, inter packet intervals, noise level, etc. Our future work is to determine their relationships, and propose a method to maximize the data collection performance by dynamically forming clusters.

6.3 Summary

In this Chapter, we explain four experiments we conducted; two in the real world environment and two in simulation. Real world evaluation is conducted to evaluate the basic data collection performance, and data collection performance with mobile sink nodes. According to their results, in comparison to the combination of CTP and 4B, that of CTDCP and ALQDP 1) reduces energy consumption by about 35%, 2) decreases the path length and the number of retransmissions, and improves the delivery ratio by approximately 2.5%, and 3) requires roughly 2,200 bytes of RAM. In addition, we confirmed that mobile sink nodes collected packets with higher delivery ratio compared to static sink nodes. Simulation evaluation is conducted to clarify the performance of proposed protocols against existing protocols, and the effectiveness of clustered-tree structure. Through the simulation evaluation against existing protocols, we confirmed that proposed protocols 1) reduce the number of transmissions per each source by about 43%, and 2) improve the delivery ratio by approximately 3.6%. Finally, we proved that clustered-tree structure is superior to cluster and tree structures in terms of data collection efficiency. Although the combination of CTDCP and ALQDP requires large storage, its superiority in data collection efficiency is significant, hence, we conclude that it outperforms existing protocols.

Chapter 7

Conclusion and Future Work

In this Chapter, we summarize this thesis, and discuss future direction of our research.

7.1 Conclusion

In this thesis, we proposed CTDCP, a data collection protocol, and ALQDP, a link quality detection protocol. CTDCP constructs a clustered-tree structure with static sensor nodes, and leverages it to collect data at multiple mobile sink nodes. ALQDP utilizes linear regression to quickly and accurately estimate the link qualities among sensor nodes. These approaches and designs enable them to efficiently collect sensor data while providing agility and accuracy in detecting link qualities.

We implemented CTDCP and ALQDP on TinyOS, and evaluated them in the real world environment and in simulation. The real world evaluation proves that, in comparison to CTP and 4B, CTDCP and ALQDP 1) reduce energy consumption by approximately 35%, 2) decrease the path length and the number of retransmissions, and improve packet delivery ratio by about 2.5%, and 3) require roughly 2,200 bytes of RAM. Through simulation evaluation, we confirmed that in comparison to Sidewinder, the combination of CTDCP and ALQDP 1) reduces the number of transmissions per each source by approximately 43%, and 2) improves packet delivery ratio by about 3.6%. Accordingly, we conclude that proposed protocols outperform existing protocols in perspective of data collection efficiency.

Due to the recent advances in technologies, the spread of mobile devices, such as, wireless sensor nodes, PDAs, smartphones, allows users to collect data directly from WSNs. This thesis proposes two novel protocols, CTDCP and ALQDP, which are designed to improve the efficiency of

data collection, and agility of link quality detection, respectively. Through the simulation evaluation, we confirmed that proposed method of combining cluster and tree structure successfully complements their disadvantages. There still remains a need to clarify the relationships between the cluster size and other factors, however, this finding is novel, and it provides insights for future data collection protocols in large scale WSNs. Moreover, through the process of defining parameters of the link quality estimation protocol, we proved that the use of lightweight machine learning algorithm along with some improvement features provides a sufficient accuracy in estimating the link qualities. This, in turn, demonstrates that the use of heavyweight algorithms leveraged in existing work is an overkill. Accordingly, the idea of providing improvement features and modifying existing lightweight algorithms greatly impacts not only link quality estimation protocols, but also wide variety of other protocols, such as, data collection, dissemination, error correction, time synchronization, and location determination. In conclusion, our two findings in this thesis substantially contribute to the filed of WSNs research, and we believe that they have large influence on future design and implementation of various kinds of protocols.

7.2 Future Work

In this section, we describe our future work: dynamic cluster formation, and redefinition of parameters in link quality estimation.

At this point, CTDCP requires overlying applications to define the maximum cluster size, which is leveraged to form clusters among a WSN. Through the simulation evaluation, we confirmed that when a large number of nodes involves in a WSN, clustered-tree structure is superior to cluster and tree structure in terms of data collection efficiency. However, in order to maximize the performance of data collection, it is preferable for CTDCP to autonomously modify clusters for performance maximization. Therefore, our future work is to clarify the relationships between data collection efficiency and multiple factors, such as, the number of source/sink nodes, inter packet intervals, and propose a method to autonomously form appropriate clusters.

In this thesis, we defined parameters of the link quality estimation based on data collected in the real world environment. However, we only used a type of sensor node during the data collection. Since the characteristics of platforms have large effect on link quality, current link quality estimation may not perform well on different platforms. Accordingly, our future work is to collect data with various testbeds, and redefine parameters of link quality estimation to improve its estimation accuracy and flexibility.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Hideyuki Tokuda for his technical and professional advice, encouragement, and guidance throughout my years in bachelor's and master's degrees. I am deeply grateful to Professor Osamu Nakamura, and Lecturer Jin Nakazawa for their valuable and technical comments on this thesis. I would also like to express my appreciation to Research Associate Takuro Yonezawa for his consecutive support of my research.

I would like to thank the members of the lab, especially, Associate Professor Kazunori Takashio, Research Assistant Tomotaka Ito, Dr. Michio Honda, Dr. Hiroshi Sakakibara, and Dr. Naoya Naotame for their comments, advices and encouragement. I would like to thank fellows of LINK and move! research group for their valuable discussions, supports, and friendships. I wish to thank numerous members of Tokuda and Murai lab for their great support. I am grateful to Mr. Takahiro Nozawa for treating me as if I were his brother. I wish to express my warm and sincere thanks to Professor Naomi Sugimoto for her consecutive support and guidance.

Last, but not least, I would like to appreciate my beloved parents Hiroyuki and Masae, adorable sisters Konomi and Chihiro, and my brother Grape for their consecutive support of my daily life. I would not have been able to write this thesis without their encouragement and understandings.

Kenji Yonekawa
February 14, 2012

Bibliography

[1] A. Tabar, A. Keshavarz, and H. Aghajan. Smart Home Care Network Using Sensor Fusion and Distributed Vision-Based Reasoning. In *Proc. of the ACM VSSN Conf.*, pages 145–154, 2006.

[2] P. Tzanos and M. Zefran. Stability Analysis of Information Based Control for Biochemical Source Localization. In *Proc. of the IEEE IRCA Conf.*, pages 3116–3121, 2006.

[3] A. Boukerche, R. Araujo, F. Silva, and L. Villas. Wireless Sensor and Actor Networks Context Interpretation for the Emergency Preparedness Class of Applications. *Elsivier Computer Communications*, 30(13):2593–2602, 2007.

[4] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *Proc. of the ACM Sensys Conf.*, pages 51–63, 2005.

[5] T. Bokareva, W. Hu, S. Kanhere, B. Ristic, and N. Wales. Wireless Sensor Networks for Battlefield Surveillance. In *Proc. of the ADM LWC Conf.*, pages 1–5, 2006.

[6] V. Dyo, S. Ellwood, D. Macdonald, A. Markham, C. Mascolo, B. Pásztor, S. Scellato, N. Trigoni, R. Wohlers, and K. Yousef. Evolution and Sustainability of a Wildlife Monitoring Sensor Network. In *Proc. of the ACM Sensys Conf.*, pages 127–140, 2010.

[7] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. SensorScope: Application-Specific Sensor Network for Environmental Monitoring. *ACM TOSN*, 6(17):1–32, 2010.

[8] R. Olfati-Saber and P. Jalalkamali. Collaborative Target Tracking Using Distributed Kalman Filtering on Mobile Sensor Networks. In *Proc. of the IEEE ACC Conf.*, pages 1100–1105, 2011.

[9] J. Zhang, C. Chen, J. Ma, N. He, and Y. Ren. uSink: Smartphone-Based Moible Sink for Wireless Sensor Networks. In *Proc. of the IEEE CCNC Conf.*, pages 90–95, 2011.

[10] U. Park and J. Heidemann. Data Muling with Mobile Phones for Sensornets. In *Proc. of the ACM Sensys Conf.*, pages 162–175, 2011.

[11] M. Liang, B. Priyantha, J. Liu, and A. Terzis. Surviving Wi-Fi Interference in Low Power ZigBee Networks. In *Proc. of the ACM Sensys Conf.*, pages 309–322, 2010.

[12] P. Dutta, S. Dawson-Haggerty, Y. Chen, C. Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. In *Proc. of the ACM Sensys Conf.*, pages 1–14, 2010.

[13] T. Lei, S. Yanjun, O. Gurewitz, and D. Johnson. PW-MAC: An Energy-Efficient Predictive-Wakeup MAC Protocol for Wireless Sensor Networks. In *Proc. of the IEEE INFOCOM Conf.*, pages 1305–1313, 2011.

[14] V. Shnayder, M. Hempstead, B. Chen, W. Allen, and M. Welsh. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Proc. of the ACM Sensys Conf.*, pages 188–200, 2004.

[15] C. Antonopoulos, A. Prayati, T. Stoyanova, C. Koulamas, and G. Papadopoulos. Experimental Evaluation of a WSN Platform Power Consumption. In *Proc. of the IEEE IPDPS Conf.*, pages 1–8, 2009.

[16] H. Lee, M. Wicke, B. Kusy, O. Gnawali, and L. Guibas. Data Stashing: Energy-Efficient Information Delivery to Mobile Sinks Through Trajectory Prediction. In *Proc. of the ACM Sensys Conf.*, pages 291–302, 2010.

[17] M. Keally, Z. Gang, and X. Guoliang. Sidewinder: A Predictive Data Forwarding Protocol for Mobile Wireless Sensor Networks. In *Proc. of the IEEE SECON Conf.*, pages 1–9, 2009.

[18] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli. BLR: Beacon-Less Routing Algorithm for Mobile Ad Hoc Networks. *Elsivier Computer Communications*, 27(11):1076–1086, 2004.

[19] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang. TTDD: Two-Tier Data Dissemination in Large-Scale Wireless Sensor Networks. *Springer Wireless Networks*, 11(1-2):161–175, 2005.

[20] N. Wang, S. Chang, Y. Huang, C. Chen, and Y. Chen. An Efficient Data Aggregation Scheme for Grid-Based Wireless Sensor Networks. In *Proc. of the ACM IWCMC Conf.*, pages 1213–1217, 2010.

[21] O. Younis and S. Fahmy. HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks. *IEEE TMC*, 3:366–379, 2004.

[22] G. Cugola and M. Migliavacca. A Context and Content-Based Routing Protocol for Mobile Sensor Networks. In *Proc. of the Springer EWSN Conf.*, pages 69–85, 2009.

[23] A. Carzaniga and A. Wolf. Content-Based Networking: A New Communication Infrastructure. In *Proc. of the Springer IMWS Conf.*, pages 59–68, 2002.

[24] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proc. of the ACM Sensys Conf.*, pages 1–14, 2009.

[25] A. Förster and A. Murphy. FROMS: A Failure Tolerant and Mobility Enabled Multicast Routing Paradigm with Reinforcement Learning for WSNs. *Elsivier Ad Hoc Networks*, 9(5):940–965, 2011.

[26] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proc. of the USENIX NSDI Conf.*, pages 2–2, 2004.

[27] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Springer Wireless Networks*, 11:419–434, 2005.

[28] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four-Bit Wireless Link Estimation. In *Proc. of the ACM HotNets Conf.*, 2007.

[29] A. Cerpa, J. Wong, M. Potkonjak, and D. Estrin. Temporal Properties of Low Power Wireless Links: Modeling and Implications on Multi-Hop Routing. In *Proc. of the ACM MobiHoc Conf.*, pages 414–425, 2005.

[30] MultihopLQI. Index of /tinyos-2.x/tos/lib/net/lqi. `http://www.tinyos.net/tinyos-2.x/tos/lib/net/lqi/`, Dec. 2011.

[31] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *Proc. of the ACM Sensys Conf.*, pages 14–27, 2003.

[32] TinyOS. TinyOS Community Forum. `http://www.tinyos.net/`, Nov 2011.

[33] N. Baccour, A. Koubaa, H. Youssef, M. Jamâa, D. Rosario, M. Alves, and L. Becker. F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks. In *Proc. of the Springer EWSN Conf.*, volume 5970, pages 240–255, 2010.

[34] T. Liu and A. Cerpa. Foresee (4C): Wireless Link Prediction Using Link Features. In *Proc. of the ACM IPSN Conf.*, pages 294–305, 2011.

[35] K. Langendoen, A. Baggio, and O. Visser. Murphy Loves Potatoes: Experiences From a Pilot Sensor Network Deployment in Precision Agriculture. In *Proc. of the IEEE IPDPS Conf.*, pages 174–174, 2006.

[36] A. Woo and D. Culler. Evaluation of Efficient Link Reliability Estimators for Low-Power Wireless Networks. Technical Report UCB/CSD-03-1270, University of California, Berkeley, 2003.

[37] C. Lin, P. Chou, and C. Chou. HCDD: Hierarchical Cluster-Based Data Dissemination in Wireless Sensor Networks with Mobile Sink. In *In. Proc. of the ACM IWCMC Conf.*, pages 1189–1194, 2006.

[38] A. Koubaa, M. Alves, and E. Tovar. Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks. In *Proc. of the IEEE RTSS Conf.*, pages 412–421, 2006.

[39] MEMSIC. Iris mote. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135%3Airis`, Dec 2011.

[40] Y. Panthanchai and P. Keeratiwintakorn. An Energy Model for Transmission in Telos-Based Wireless Sensor Networks. In *Proc. of the JCSSE Conf.*, 2007.

[41] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of the ACM Sensys Conf.*, pages 126–137, 2003.

[42] MEMSIC. Micaz mote. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148%3Amicaz`, Dec 2011.

[43] J. Yoon, M. Liu, and B. Noble. Sound Mobility Models. In *Proc. of the ACM MobiCom Conf.*, pages 205–216, 2003.

[44] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the ACM Sensys Conf.*, pages 95–107, 2004.

[45] S. Biswas and R. Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *Proc. of the ACM SIGCOMM Conf.*, pages 133–144, 2005.