Spring 2015

# INTERMITTENTLY CONNECTED DELAY-TOLERANT WIRELESS SENSOR NETWORKS

Ying Li
*University of New Hampshire, Durham*

# INTERMITTENTLY CONNECTED DELAY-TOLERANT WIRELESS SENSOR NETWORKS

BY

YING LI

B.S., Hubei University of Technology, China, 2004
M.S., Hubei University of Technology, China, 2007

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy
in
Computer Science

May 2015

This dissertation has been examined and approved in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science by:

Dissertation Director, Dr. Radim Bartos, Associate Professor of Computer Science

Dr. R. Daniel Bergeron, Professor of Computer Science

Dr. Michel Charpentier, Associate Professor of Computer Science

Dr. Nicholas J. Kirsch, Assistant Professor of Electrical and Computer Engineering

Dr. Zheng Peng, Assistant Research Professor of Computer Science and Engineering, University of Connecticut

Dr. Robert Russell, Associate Professor of Computer Science

Dr. Elizabeth Varki, Associate Professor of Computer Science

On April 24, 2015

Original approval signatures are on file with the University of New Hampshire Graduate School.

# DEDICATION

*To my parents, and dearest BJ*

# ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my advisor Dr. Radim Bartos, for his inspiring guidance and great wisdom, who did his utmost to help me through the entire doctoral research. Without his guidance and persistent help, the dissertation would not have been possible.

I would like to thank my committee members: Dr. Daniel Bergeron, Dr. Michel Charpentier, Dr. Nicholas J. Kirsch, Dr. Zheng Peng, Dr. Robert Russell, and Dr. Elizabeth Varki, who provided me with helpful suggestions and invaluable advice throughout my research at University of New Hampshire. I also thank Dr. Phil Hatcher, Dr. Jim Weiner, Dr. Swapnil Bhatia, Dr. Adam Villa, Qian Liu, and Can Xiong for their time and various forms of support.

In addition, a thank you to Dr. Yitang Zhang, who lets me know the spirt of a successful researcher, and whose enthusiasm for research has a lasting effect on me. I am also grateful for his uncountable help in my daily life over the past five years.

I would also like to express my gratitude to my colleagues in InterOprability Laboratory: Dr. Mikkel Hagen, Timothy Carlin, James Swan, Timothy Winters, Erica Johnson, Bob Noseworthy, and Jeff Laird for their help in both my work and research.

I have received generous financial and academic support from the InterOperability Laboratory and University of New Hampshire during my graduate study. I appreciate the research assistantship offered by the InterOperability Laboratory from 2010 to 2014 and the various assistantships offered by the Department of Computer Science in my first three years of study. I am also thankful for the dissertation year fellowship granted by the Graduate School in 2014-2015, which made my research and dissertation smoothly.

A special thank you is given to Dr. Honggeng Zhou from the Department of Economics at University of New Hampshire, for his kind financial support for three semesters (Fall 2008, Spring 2009, Fall 2009).

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT


INTERMITTENTLY CONNECTED DELAY-TOLERANT WIRELESS SENSOR NETWORKS

by

Ying Li

University of New Hampshire, May 2015

Intermittently Connected Delay-Tolerant Wireless Sensor Networks (ICDT-WSNs), a branch of Wireless Sensor Networks (WSNs), have features of WSNs and the intermittent connectivity of Opportunistic Networks. The applications of ICDT-WSNs are increasing in recent years; however, the communication protocols suitable for this category of networks often fall short. Most of the existing communication protocols are designed for either WSNs or Opportunistic Networks with sufficient resources and tend to be inadequate for direct use in ICDT-WSNs.

In this dissertation, we study ICDT-WSNs from the perspective of the characteristics, challenges and possible solutions. A high-level overview of ICDT-WSNs is given, followed by a study of existing work and our solutions to address the problems of routing, flow control, error control, and storage management. The proposed solutions utilize the utility level of nodes and the connectedness of a network. In addition to the protocols for information transmissions to specific destinations, we also propose efficient mechanisms for information dissemination to arbitrary destinations. The study shows that our proposed solutions can achieve better performance than other state of the art communication protocols without sacrificing energy efficiency.

# Chapter 1

# Introduction

Intermittently Connected Delay-Tolerant Wireless Sensor Networks (ICDT-WSNs) are a new branch of Wireless Sensor Networks (WSNs), which have characteristics of WSNs and Opportunistic Networks. These characteristics include the limited resources, such as energy supply, computation capability, storage space, bandwidth, communication range [2] and the intermittent connectivity. Intermittent connectivity means that the end-to-end paths do not always exist in networks [13]. These difficulties make the design of communication protocols for ICDT-WSNs a challenging task, although ICDT-WSNs have been commonly used in areas whose development environments are unsafe or even impossible for human to access. Examples of use include wildlife tracking [32], assisting submarine location estimation [90], solar-powered autonomous underwater vehicle (SAUV) platform for underwater networks [6], coal mine structure surveillance [39] and sandstorm forecast [78].

Most of the existing protocols cannot be directly employed in ICDT-WSNs, since they are either designed for WSNs or DTNs that do not take all limitations of ICDT-WSNs into consideration. Without reliable, robust and efficient communication protocols, the performance of ICDT-WSNs is degraded resulting in shortened network life time, decreased propagation speed and increased packet loss rate. As a consequence, the development of ICDT-WSN applications is constrained.

## 1.1 Background

### 1.1.1 Wireless Sensor Networks

*Wireless Sensor Networks* (WSNs) have been extensively studied and widely used in the recent decade. A WSN can consist of one to several types of sensor nodes such as visual, thermal, acoustic, infrared, radar, low sampling rate magnetic, and seismic [2]. WSNs are mission-oriented: all sensor nodes of a WSN cooperate together to accomplish the mission of the network, such as collecting environmental data from a designated area and tracking an object. According to the environment the WSNs are developed for, WSNs can be categorized into terrestrial, underwater or underground:

- *Terrestrial* WSNs are developed above ground, and are usually composed of hundreds to thousands of low-cost sensor nodes [87]. The terrestrial WSNs can be used for environment sensing and monitoring, industry monitoring [23] and surface exploration. Radio Frequency (RF) communication is widely used in terrestrial WSNs. Energy efficiency is very important for terrestrial WSNs, since the power of sensor nodes is very limited even with solar cells.

- *Underwater* WSNs consist of a variable number of sensors and vehicles that are sparsely deployed under water for oceanographic data collection, pollution monitoring, offshore exploration, disaster prevention, navigation assistance and tactical surveillance applications [3]. Instead of RF communication, acoustic communication is preferred in underwater WSNs, because of the high attenuation of RF in aquatic environments [25]. Compared to the terrestrial WSNs, underwater WSNs suffer more severe challenges: longer propagation delay, less bandwidth, more severely impaired channels and non-rechargeable, limited power.

- *Underground* WSNs comprise of a number of sensor nodes buried underground or placed in coal mines or caves, used to monitor a variety of underground conditions [39, 4]. RF communication can be used in underground WSNs [39], but the underground environment causes high attenuation of electromagnetic waves. Akyildiz [4] points out that Magnetic

Induction (MI) and seismic waves might be better for communication in underground WSNs. In addition to the challenges of underwater WSNs, signal fade is unavoidable in underground WSNs.

With the development of micro-electro-mechanical systems (MEMS) technology, sensor nodes have become smaller, lighter, smarter and cheaper. In addition to the main categories of WSNs mentioned above, WSNs are now being used in airplane surveillance [8] and body sensor networks [14, 57].

The network infrastructures, sensor nodes and communication protocols can be different from one WSN to another. Because WSNs are mission oriented, the topology design and device selection for a WSN depends on the application for each WSN.

Generally, WSNs have little or no infrastructure. According to the manner of node deployment, WSNs can be divided into two groups: *ad hoc* WSNs and *pre-planned* WSNs. Ad hoc WSNs have no infrastructure, the sensor nodes are deployed into a field randomly, possibly scattered from an airplane and left unattended. In order to maintain connectivity and detect failures, the protocols and algorithms for ad hoc WSNs should be able to self-organize. The ad hoc nature makes this category of WSNs suitable for disaster relief and operations in inaccessible areas. Pre-planned WSNs, on the contrary, are more structured networks, and can be grouped into wireless mesh networks. Sensor nodes in pre-planned WSNs are placed at particular positions in a pre-planned manner, such that topologies are well designed beforehand. For several examples of typical pre-planned WSNs see underwater WSNs [3, 25] and underground WSNs [4, 39].

According to the mobility of sensor nodes, WSNs can be categorized into *static* WSNs and *mobile* WSNs. WSNs that only consist of non-moving sensor nodes are static WSNs. WSNs containing self propelled sensor nodes are mobile WSNs. Depending on the design of a network, the movement of sensor nodes in a network can be controllable and predictable. This property not only distinguishes mobile WSNs from mobile ad hoc networks (MANETs), but also provides an advantage for communication protocol design.

### 1.1.2 Opportunistic Networks

Opportunistic networks make no assumption regarding the existence of a complete path between a source and destination. *Delay-Tolerant Networks (DTNs)* [10] can be seen as a well-known example of opportunistic networks. A DTN is an overlay on top of regional networks[1], and provides interoperability between these networks [17]. DTNs are challenging networks, where the architectures and communication protocols used in traditional networks may operate poorly. The challenges associated with DTNs are intermittent connectivity, long or variable delay, asymmetric data rates, and high error rates.

The Delay-Tolerant Networking Research Group (DTNRG) [15] discusses the *bundle layer* as the overlay DTN architecture, which not only provides a transparent communication among different regional networks, but also hides the disconnection and delay from the application layer.

The bundle layer sits between the application layer and the transport layer in the DTN protocol stack. In the bundle layer, the application data is encapsulated into bundles with bundle headers and passed to the transport layer. The bundle layer implements *store-and-forward* message switching to overcome the network interruption, and provides end-to-end reliability across a DTN through custody transfers. Nodes in DTNs have persistent storages to store bundles and support *custody transfers*.

Custody transfers achieve end-to-end reliability by employing node-to-node retransmission in the bundle layer to prevent data loss and corruption. If a node requires custody transfers, it starts a time-to-acknowledge retransmission timer after sending a bundle to the next node. If the next node accepts the custody, it returns an acknowledgement to the sender. If no acknowledgement is received before the sender's timer expires, the sender retransmits the bundle. If a node supports custody transfers, it must store a bundle until another node accepts custody or the bundle's time-to-live expires. Otherwise a node only needs to store a bundle until the outbound links are available.

The node name in a DTN consists of two parts, the *region ID* and the *entity ID*. The bundle layer provides transparent communication among different regional networks through the region

[1] A regional network is a network in which the communication characteristics are homogeneous.

| Attributes | ZebraNet [32] | CenWits [26] | Sandstorm forecast [78] | Coalmine structure surveillance [39] |
|---|---|---|---|---|
| **Goal** | Wildlife tracking | Search-and-rescue system | Weather forecast | Disaster monitoring |
| **Connectivity** | Zebras carry sensor nodes and move | People ware sensor nodes and move | Sand cover sensor nodes temporarily | Obstructions block transmission temporarily |
| **Nodes** | Limited energy & storage | Limited storage & processing capability | Limit energy | Limited energy & storage |
| **Data** | Position, temperature and speed | Position | Environment data | Position |
| **Requirements for reliability and time limit** | Y | Y | Y | Y |

Table 1.1: Instances of ICDT-WSNs

ID. The routing within a regional network is based on the entity IDs.

## 1.2 Intermittently Connected Delay-Tolerant WSNs

ICDT-WSNs are intended for networks where besides low cost, intermittently connectivity and the lack of network infrastructure are the key characteristics. At the same time, ICDT-WSNs are expected to deliver performance that meets the needs of the network users. As a result, ICDT-WSNs can be applied in a range of scenarios (Table 1.1 gives several examples).

### 1.2.1 ICDT-WSN Characteristics

ICDT-WSNs are *mission-oriented*: as it is the case with WSNs, all nodes in an ICDT-WSN cooperate together to accomplish the common mission of the network. This requires that they

should provide high rate of data delivery with acceptable latency.

ICDT-WSNs emphasize *local communication* rather than focusing solely on the source-to-sink traffic. A node collects information in the area it located at and typically communicates with the nodes in its neighborhood. Furthermore, routing decisions are made locally without requiring global network information.

ICDT-WSNs, as the name suggests, interconnect *sensor nodes*. Each node in an ICDT-WSN performs sensing and collects information from the local area. The collected information is distributed as needed using wireless communication to serve the needs of the mission.

The constraints on the cost of nodes lead to the limitations on energy supply, storage space, and the computational capacities. Sensor nodes are usually equipped with battery to supply energy. Typically the amount of memory in one sensor node is less than 512 KB [55]. The storage capacity and computation capability of a sensor node are far less than that of a computational node, such as nodes in DTNs.

The *intermittent connectivity* of ICDT-WSNs can be caused by environment-driven transient losses of connectivity between nodes or by the sparseness of a network combined with node mobility:

- Environmental factors, such as interference or obstructions, may cause a loss of connectivity on a temporal scale larger than the transmission time of a packet. Sandstorm forecasting is one example of scenario where sand may cover sensor nodes from time to time due to the wind and the connectivity is lost for a period of time. Another such example would be loss of connectivity between nodes in an acoustic underwater network due to steep gradient of salinity caused by shifting tidal currents [6, 28].

- Mobile nodes may come into contact only occasionally while operating at other times outside of the communication range of each other. Node mobility can be random, uncontrollable, controllable, or predictable. ZebraNet is an example of a network with random and uncontrollable node mobility — each zebra that carries a sensor node moves randomly. CenWits is an example for predictable node mobility — visitors wearing sensor nodes walk along

the predetermined trail. These sensor nodes collect the visitors' positions and corresponding time, and communicate locally in order to pass the information back to the control center.

### 1.2.2 Node Model

An ICDT-WSN node consists of a transceiver, a protocol processing unit, a transmission queue, packet storage, energy manager, and an application controller. Figure 1.1 shows the node model. The queue stores the packets waiting to be transmitted while the packet storage is used for in-network caching of generated, received, or overheard packets with the goal of reducing energy consumption and improving packet latency during loss recovery. The energy manager informs both the protocol processing unit and the application manager about the available energy and controls the energy distribution to each unit of a node. Application manager executes the mission of a node and interfaces with the communication and energy subsystems.

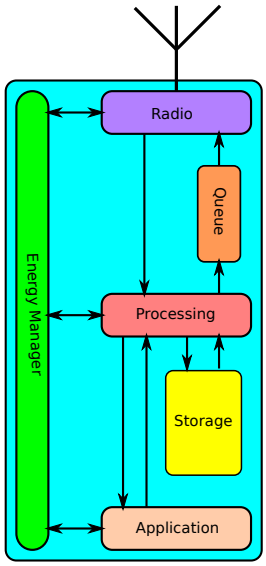



Figure 1.1: Node model

### 1.2.3 Challenges

Routing, flow control and error control are three main problems addressed in ICDT-WSNs communication protocols in layers above the routing layer. Due to the intermittent connectivity,

the traditional end-to-end flow control and error control are less appropriate and hop-by-hop approaches are more common. Since a complete path from a source to a destination may not exist, the routing algorithms must be capable to operate even in such cases. Limits on the resources in IDCT-WSNs make efficient in-network storage management and energy efficiency critical to a protocol design. The following general problems will be considered in the ICDT-WSN protocol design proposed in this project:

**Routing:** Deliver data packets to destinations through store-and-forward to overcome the intermittent connectivity.

**Flow control:** Avoid congestion and shorten the queue length.

**Storage management:** Maximize the utility of storage.

**Error control:** Loss recovery for packet loss happened during transmission and due to the overflow of storage or queue.

**Energy efficiency:** Reduce unnecessary transmissions.

To evaluate the performance of communication protocols for ICDT-WSNs, the following four metrics are commonly used: packet delivery rate, packet latency, number of hops per packet and energy consumption [77]. The definitions of these metrics are:

**Packet delivery rate:** The ratio between the number of distinct packets received by the sink and the number of packets sent by all sources to this sink.

**Packet latency:** The time interval between the moment a packet generated by a source and the time it is received by the sink.

**Number of hops per packet:** The number of times a packet was transmitted on its way from its source to the sink. In case multiple copies of the packet are delivered to the destination, the number of transmissions of the first delivered copy is considered.

**Energy consumption:** The total energy consumption during a packet delivery will be the sum of transmission and reception times of all nodes in the network participating in the delivery of this packet.

These metrics have direct connections to the problems mentioned above. Table 1.2 lists out the relationship between the problems and the four principal metrics.

| **Metric** | **Improved by addressing** |
|---|---|
| Packet delivery rate | Flow control, storage management, error control |
| Packet latency | Routing, flow control, storage management, error control |
| Number of hops per packet | Routing |
| Energy consumption | Energy efficiency |

Table 1.2: Measures

# Chapter 2

# Research Design and Methods

## 2.1 Importance of the Study

When a disaster strikes, the ability to communicate is of the utmost importance during the search and rescue operations. Disasters may occur in locations without any communication infrastructure, or may cause the destruction or serious damage to the existing infrastructure. The need for rapid response and limited resources may not permit a complete communication infrastructure buildup or full repair. As a result, *intermittent connectivity* and the *lack of network infrastructure* become the key characteristics that define the problem that the designers of communication solutions face. Furthermore, while achieving global connectivity is desirable and the ultimate goal, quick establishment of a limited local connectivity is of immediate value.

ICDT-WSNs are designed serve scenarios such as the ones outlined above. The characteristics of ICDT-WSNs make them similar to *Wireless Sensor Networks (WSNs)* [2] and *Opportunistic Networks* [53]. However, due to the differences in network node capabilities and the intermittent connectivity, most of the existing protocols from these two fields cannot be directly employed in ICDT-WSNs [41]. Without an appropriate protocol to provide reliable, robust, and efficient communication, ICDT-WSNs suffer from inefficiencies and degraded performance.

## 2.2 Research Approach

The research approach includes identifying the key problems, design of fundamental methods to address these problems and experiments to evaluate the performance. The key problems and measures in this area are discussed in Section 1.2. Figure 2.1 outlines the fundamental methods that are proposed to be used to address the problems in this project. The proposed research focusses on problems in the transport and network layers through a combination of end-to-end and hop-by-hop mechanisms. The rest of this section outlines the proposed methods to address the problems in the design of communication protocols for ICDT-WSNs.

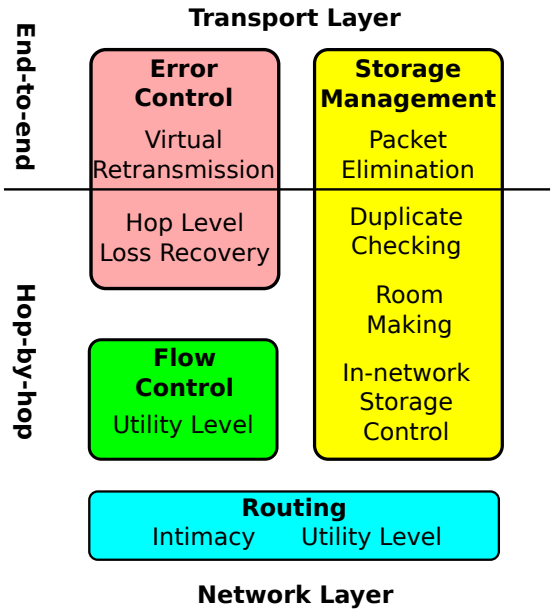



Figure 2.1: Components of the proposed research

### Error Control

*Virtual retransmission* and *hop level loss recovery* are loss recovery mechanisms for error control. Hop level loss recovery guarantees there is no hop level packet loss between a sender and a receiver. Virtual retransmission takes advantage of the packets saved in the intermediate nodes to provide end-to-end reliability. As a consequence, the number of end-to-end retransmissions can be reduced and the packet delivery rate and energy efficiency can be enhanced.

**Routing and Flow Control**

*Utility level of nodes* and *intimacy* are used to improve routing and flow control. *Utility level of nodes* is a quantitative measure of the health status of nodes, which includes the available energy, storage space and current queue length in its calculation. It is used to make better routing decision, forwarding data packets to senders' neighbors with more available resources, such as more free memory or energy. As a result, the congestion can be alleviated, and energy consumption and packet latency, caused by overflow-triggered retransmissions, can be reduced. *Intimacy* is a metric to measure the closeness of a node to a destination in order to help route packets to the nodes with better chances to deliver packets to the destination. A sender uses the intimacy to decide whether to send and where to send a data packet. This main objective is to deliver packets more quickly.

According to the mechanism to decide next hops, routing algorithms can be grouped into *single-copy* and *multi-copy*. Algorithm 1 and 2 show the framework for single-copy routing algorithms, and Algorithm 3 and 4 show the framework for multi-copy routing algorithms. The parts with underlines are the works that will be explored; except in-network storage control specified in the storage management section, the details for rest parts are listed as follows.

*Balance between utility level and intimacy* aims to find the best next hop according to the health state and intimacy of a node. Nodes that are usually in the communication range of the destination are less healthy and can deliver packet more successfully than other more healthy nodes, while they are more likely to suffer network congestion and node failures. An in-depth study of the relationship between these measures and metrics can lead to a better combination.

*Better next hop* is a mechanism to find a set of appropriate next hop nodes in the neighborhood of a sender. The sender will forwards each appropriate next hop node a copy of data packet in order to enhance the packet delivery rate without sacrificing packet latency and energy efficiency.

**Storage Management**

Storage management is addressed through three mechanisms: *acknowledgement-assisted storage management*, *room making*, and *in-network storage control*. *Acknowledgement-assisted*

**Algorithm 1** Single-copy Sender

$RQST$: request, $RRPLY$: request reply, $BH$: best next hop
**while** $node_i$ has packets to forward **do**
  $node_i$ broadcasts RQST
  **if** $node_i$ receives RRPLYs **then**
    $BH = \{\}$
    **for all** $neighbor_j$ in the received RRPLYs **do**
      BH = BALANCE BETWEEN UTILITY LEVEL AND INTIMACY
    **end for**
    **if** $BH \neq \{\}$ **then**
      forwards packets to $neighbor_j$
    **end if**
  **end if**
**end while**

**Algorithm 2** Single-copy Receiver

$DATA$: data packet
**if** $node_i$ receives DATA **then**
  DUPLICATE CHECKING & IN-NETWORK STORAGE
  caches DATA in the queue
**end if**
**if** $node_i$ overhears DATA **then**
  DUPLICATE CHECKING & IN-NETWORK STORAGE CONTROL
**end if**

**Algorithm 3** Multi-copy Sender

$RQST$: request, $RRPLY$: request reply, $IM$: intimacy, $UL$: utility level
**while** $node_i$ has packets to forward **do**
  $node_i$ broadcasts RQST
  **if** $node_i$ receives RRPLYs **then**
    **for all** $neighbor_j$ in the received RRPLYs **do**
      **if** $neighbor_j$ is a BETTER NEXT HOP **then**
        forwards packets to $neighbor_j$
      **end if**
    **end for**
  **end if**
**end while**

**Algorithm 4** Multi-copy Receiver

$DATA$: data packet
**if** $node_i$ receives DATA **then**
  DUPLICATE CHECKING & IN-NETWORK STORAGE CONTROL
  caches DATA in the queue
**end if**

*storage management* improves the utility of nodes' limited storage leading to quicker recovery from packet loss and reduced energy spent on retransmissions. It includes *packet elimination* and *duplicate checking*. Packet elimination is an end-to-end mechanism, which is to delete confirmed packets from a node's storage after receiving an end-to-end acknowledgement. Duplicate checking is a hop-by-hop mechanism, which is to check whether there is a copy of the received packet before saving the packet into a node's storage.

*Room making* is to eliminate less important data packets even if these data packets are unconfirmed. This mechanism will be triggered when congestion is caused by heavy traffic. Algorithm 5 provides a framework for applying room making. The assumption for room making is that the total amount of space of a node memory is fixed such that if more space is used for the storage, less space is used for the queue. Room making will decide where to remove less important data packets, from the storage or the queue, in order to let more important incoming data packets can be proceeded. The purpose of room making is to improve packet latency without sacrificing packet delivery rate and energy efficiency.

**Algorithm 5** RQST Receiver

1: $RQST$: request, $RRPLY$: request reply
2: **if** $node_i$ receives RQST **then**
3:  **if** the memory is full **then**
4:  ROOM MAKING
5:  **end if**
6:  transmits RRPLY back to the RQST sender
7: **end if**

*In-network storage control* is to save memory for more useful data packet while providing sufficient copies of data packets in the intermediate nodes to support reliability. Since more copies of data packets in the network can lead to an easier loss recovery when packet loss happens. However, redundancy can degrade the network performance especially if resources are limited. The goal of in-network storage control is to find the threshold to cut unnecessary redundancy while providing robust network without sacrificing energy efficiency.

## 2.3 Research Tools

Simulation study is the main method applied in this project. Contiki [11] system the tool used in most of the study. Contiki is an open source operating system for memory-constrained, low-power devices, with a particular focus on wireless Internet of Things devices. In addition to the uIP and uIPv6 mechanisms, Contiki provides the Rime stack for lightweight network protocol design. This study utilizes the Rime stack to design the new communication protocols for ICDT-WSNs. Contiki system includes a simulator, Cooja [50], to simulate a network of Contiki nodes. The nodes can be emulated nodes that emulate the entire hardware. Cooja provides an option to track the energy consumption of nodes. Cooja is also capable to simulate a dynamic network by taking advantage of the nodes' positions of a mobile scenario. In this study, Bonnmotion [1] is used to generate mobile scenarios with respect to the desired mobility models, to imitate the intermittently connected scenarios.

## 2.4 Contributions

The contributions of this dissertation consist of new energy-efficient solutions for the problems of routing, flow control, error control, and storage management in ICDT-WSNs.The proposed methods for routing and in-network storage control can provide efficient data transmission for intermittently connected scenarios without significant in-network storage capacity requirements. As a result, reliable local communication can be achieved in intermittently connected scenario without a network infrastructure and a high cost. The ideas for storage management and energy efficiency can also reduce latency and improve resource utilization in the scenarios without serious resource limitations.

# Chapter 3

# Related Work

## 3.1 Transport Protocols

Due to the characteristics of ICDT-WSNs, the traditional transport protocols that are widely used in Internet (TCP, UDP) cannot be applied in ICDT-WSNs directly [29]. User Datagram Protocol (UDP) would not be an appropriate option for ICDT-WSN applications that require reliable delivery, such as military surveillance. Transmission Control Protocol (TCP) is inefficient in ICDT-WSNs, since TCP is designed for traditional networks where packet loss is mainly due to traffic congestion [72]. But there are many reasons other than traffic congestion causing packet loss in ICDT-WSNs, such as sensor nodes out of memory, signal attenuation and energy exhaustion. Therefore, the congestion control mechanism in TCP fails to determine traffic problems correctly, leading the protocol to perform poorly. Moreover, the end-to-end reliability of TCP has to be managed by source nodes, which can make the limited energy of sensor nodes drain quickly.

Different applications may require different levels of reliability. As a result, transport protocols that can provide multiple levels of loss recovery are more flexible to meet the various requirements for the diversity of applications in ICDT-WSNs. Because of the multi-hop transmission in ICDT-WSNs, congestion is more likely to happen at sensor nodes that are geographically closer to the sink or have a higher probability of moving into the sink's communication range. Therefore,

effective congestion control is required to reduce packet loss, save energy, extend network lifetime and enhance throughput.

There are two approaches for loss recovery and congestion control in ICDT-WSNs: end-to-end and hop-by-hop. End-to-end approaches can be thought of as centralized methods. The sink node is usually the end that manages reliability and executes congestion control. The benefits of end-to-end approaches are: 1) the sink node usually has plenty of energy; 2) the sink node can have a more complete view of traffic in the whole network than a single sensor node. Hop-by-hop approaches can be thought of as decentralized methods. Every node in the network has the responsibility to provide hop level loss recovery and congestion control. Compared to the end-to-end approaches, hop-by-hop congestion control can react faster when a problem is detected and the retransmission distance of hop-by-hop loss recovery is shorter. But hop-by-hop approaches are less flexible than the end-to-end approaches in providing variable reliability levels. Due to the intermittent connectivity, hop-by-hop approaches are more efficient than end-to-end approaches in ICDT-WSNs.

Most of the existing transport protocols are designed for either WSNs or DTNs and cannot meet all the requirements of ICDT-WSNs. These transport protocols provide either end-to-end reliability or congestion control. To the best of our knowledge, no literature points out that combining protocols together can achieve better performance than applying a single protocol that provides both services. In the rest of this subsection, we focus our study on transport protocols that provide both end-to-end reliability and congestion control. The outlined transport protocols have been proposed in recent years and have some features than can be used to design transport protocols for ICDT-WSNs.

*Hop*: Hop [37] provides end-to-end reliability and congestion control in a hop-by-hop manner for wireless mesh network. It uses *block*s as the transmission unit and disables the link layer acknowledgement. A block consists of a number of packets. A node $N_i$ transmits a BSYN message to a node $N_{i+1}$ when finished forwarding a block of packets to $N_{i+1}$ to indicate the end of the block. After $N_{i+1}$ receives the BSYN message, a BACK message is sent by $N_{i+1}$ to $N_i$ to indicate

the lost packets in that block. When $N_i$ receives the BACK message, it retransmits any lost packets indicated by the BACK message to $N_{i+1}$. The procedure continues until the block has been completely received by $N_{i+1}$. Then $N_{i+1}$ forwards the block to its next hop $N_{i+2}$. In this manner, Hop not only guarantees hop level reliability, but also reduces the per-packet based acknowledgements by employing per-block based acknowledgements.

Hop applies virtual retransmission and in-network storage to decreases the number of end-to-end retransmissions when provides end-to-end reliability. Hop assumes that when transmitting a block along a path $P$, the nodes near $P$ might overhear parts of the block. Nodes in Hop cache the received and overheard packets. If a node $A$ on $P$ fails, another node $B$ in the neighborhood of $A$'s upstream node $C$ is selected as the new relay node. Instead of retransmitting the complete block to $C$, virtual retransmission allows $B$ to only send the missing packets of the block to $C$.

Hop uses backpressure to provide hop-by-hop congestion control. In Hop, each node keeps a threshold H, which is the difference between the number of blocks the node receives and the number of blocks it transmits to the next node. After completely receiving H blocks, a node does not respond to the arriving BSYN messages from upstream nodes until it transmits more blocks to its downstream nodes. The next hop to forward blocks is chosen through backpressure to avoid the heavy traffic directions.

*Rate-Controlled Reliable Transport (RCRT)*: RCRT [51, 52] is an end-to-end reliable transport protocol for WSNs, which provides centralized congestion control at the sink node and supports concurrent flows. In RCRT, the sink node maintains two lists for each flow. One is the list of missing packets, and the other is the list of out-of-order packets. RTRC takes advantage of a NACK-based end-to-end recovery scheme that entries in the missing packet lists are sent back to the sources through Negative Acknowledgements (NACKs).

There are three distinct logical components in the congestion control mechanism of RCRT: congestion detection, rate adaption and rate allocation. The intuition of the congestion detection is that if the lost packets are recovered quickly enough, the network is not congested. The way to measure the loss recovery time is the lengths of the out-of-order packet lists. Let $r_i$ be the

transmission rate of source $i$, $RRT_i$ be the round trip time of source $i$, and $L_i$ be the length of the out-of-order packet list for source $i$ at an observation time. The number of rounds for recovering the lost packets of source $i$ at the observation time is measured:

$$L_{norm,i} = \frac{L_i}{r_i RRT_i}. \tag{3.1}$$

If $L_{norm,i}$ of any list is higher than the higher threshold, the network is congested. If $L_{norm,i}$ of every list is lower than the lower threshold, the network is declared uncongested.

The rate adaptation of RCRT uses Additive increase/multiplicative decrease (AIMD) on the aggregate rate of all flows observed at the sink. Whenever RCRT detects the network is congested, it decreases the rate:

$$R(t+1) = R(t)M(t) \tag{3.2}$$

where $R(t)$ is the sum of currently assigned rates $r_i(t)$ for all flows $i$, $M(t)$ is the multiplicative decrease factor (Equation 3.3), and $p_i(t)$ is the current delivery ratio of the congested flow from source $i$.

$$M(t) = \frac{p_i(t)}{2 - p_i(t)} \tag{3.3}$$

If the network is uncongested, RCRT increases the rate:

$$R(t+1) = R(t) + A \tag{3.4}$$

where $A$ is the additive increase factor, a positive constant.

After adapting the overall rate, the rate allocation component of RCRT assigns rates $r_i(t)$ to each flow, with the sum of all $r_i(t)$ equal to $R(t)$. The prototype of RCRT provides three rate allocation rules: demand-proportional, demand-limited and fair, which makes RCRT flexible in rate allocation. Demand-proportional is set so that each flow $i$ has desired rate $d_i$, and the assigned rate is calculated as follows:

$$r_i(t) = d_i \rho_i(t) \tag{3.5}$$

$$\rho_i(t) = \frac{R(t)}{D} \tag{3.6}$$

where $\rho_i(t)$ is the allocation ratio, and $D$ is the sum of desired rates of all flows. Demand-limited is defined such that $R(t)$ is divided equally among all flows as long as the assigned rate $r_i$ is less than or equal to the desired rate $d_i$. Fair is defined such that $R(t)$ is divided equally among all flows regardless of desired rate.

Unlike RCRT, *Retiring Replicants* [71] utilizes AIMD to provide a hop-by-hop congestion control. Retiring Replicants dynamically controls the amount of replicants in a networks by letting each node maintains the drop and replication counters. According to these numbers, nodes decide to increase or decrease the replicants locally. Retiring Replicants enhances the network performance when the underlying routing protocols are flooding based protocols. However, since Retiring Replicants does not provide reliability, the delivery rate cannot be improved significantly.

*Sensor Transmission Control Protocol (STCP)*: STCP [29] supports end-to-end reliability for multiple flows and has the ability to provide multiple levels of reliability. Initially, source nodes send session initiation packets to inform the sink of what flows they will send, such as the number of flows, flow type, transmission rate and reliability level. After source nodes receive ACK packets for their initiation packets, they can start to transmit data.

If a flow is a continuous flow, the sink will know the expected arrival time of the next packet in the flow through the transmission rate sent in the initiation packet. If the sink does not receive a packet within the expected time frame, it transmits a NACK packet with the sequence number of the missing packet in the flow. When the sink receives the next packet, it transmits an ACK packet. The sink keeps a list of transmitted NACK packets to deal with NACK packet loss by periodically checking the list and resending NACK packets until all missing packets have been received. When a missing packet is received, its entry is removed from the NACK packet list. The source nodes retransmit missing packets after receiving NACK packets.

Each source node keeps a buffer timer to prevent buffer overflow. The buffer timer depends on the transmission rate of packets and the network conditions. When a source node's buffer timer

expires, the node's buffer is cleaned.

If a flow is an event-driven flow, the sink sends an ACK after receiving each packet. Source nodes cache packets until corresponding ACKs are received, and then remove confirmed packets from their buffers. Each source node keeps a retransmission timer. When the retransmission timer of a source node expires, packets in the node's buffer are retransmitted.

STCP treats multi-level reliability for continuous flows and event-driven flows differently. For continuous flows, the sink measures the reliability level by the fraction of the packets received successfully. The sink sends NACKs for retransmission only when the reliability level goes below the required level. But for event-driven flows, source nodes calculate the reliability before transmitting packets. Packets are cached at the source nodes if the reliability goes below the required level.

Congestion control in STCP is through the binary congestion notification bit in the packet header. Each sensor node maintains two thresholds for its buffer $t_{lower}$ and $t_{higher}$. When the buffer size reaches $t_{lower}$, the sensor node sets the congestion notification bit in the packets it forwards with a certain probability. When the buffer size reaches $t_{higher}$, the sensor node sets the congestion notification bit in every packets it forwards. When receiving a packet with congestion bit set, the sink sets the congestion bit in the ACK packet. After receiving an ACK packet with congestion bit set, the source node either adjusts the transmission rate or routes the successive packets along a different path.

*Flush*: Flush [33] provides end-to-end reliability and hop-by-hop congestion control for one flow in a network at a time. Data is divided into packets and sent in a pipelined scheme by Flush. The end-to-end reliability is guaranteed through selective negative acknowledgement (NACK). The sink initializes the transmission by sending a request to a target node. A source node sends packets after receiving a request. The sink keeps track of received packets, and after an estimated round-trip-time the sink sends a NACK packet with the sequence numbers of the first three missing packets. This procedure continues until the sink receives all required packets.

Flush uses a hop-by-hop rate control mechanism to avoid congestion. The algorithm follows

two policies: 1) a node transmits only when its successor is free from interference; 2) the sending rate of a node cannot exceed the sending rate of its successor. So the sending interval at node $i$ is the maximum value of either the minimum delay $d_i$ or the sending interval of its successor node $D_{i-1}$. The minimum delay is the sum of $\delta_i$ the time node $i$ takes to transmit a packet, $\delta_{i-1}$ the time node $i-1$ transmits the packet and $f_{i-1}$ the time the packet moves out of the interference range of node $i-1$. When a node's queue length exceeds a threshold, the node temporarily doubles its transmission time $\delta$ to increase the delay.

*Burst Forwarding*: Burst forwarding [16] alone does not provide end-to-end reliability and congestion control, but it provides a way to employ TCP in WSNs. Burst forwarding is a pipelined forwarding protocol over TCP that can provide high throughput and low energy consumption. Burst forwarding supports single flow transmission, and applies multi-channel operation to achieve high throughput.

Burst forwarding performs *clear channel assessments* (CCA) before starting transfer a burst. If a sender tests the channel is idle, it starts to transfer a burst. If a receiver tests the channel is not idle, it starts to listen. Packets are forwarded in burst. When a receiver receives a packet, it sends an acknowledgement back to the sender. After received an acknowledgement, a sender keeps forwarding the following packets. If no acknowledgement received by a sender after forwarded a packet, the sender retransmits the packet.

Burst forwarding provides two-level retransmissions: *link-layer retransmission* and *MAC-layer retransmission*, which can reduce the number of end-to-end retransmissions of TCP. Link-layer retransmissions reduce the number of data transfers between the link and transport layers since the messages can be stored in the radio transceiver. Instead of retransmitting a lost packet, the MAC-layer retransmission transmits the lost packet with the new packets in burst after a back-off time.

*Asymmetric and reliable transport (ART) mechanism*: ART [70] provides different levels of reliability for two different types of flows. Flows in a network are classified into upstream flows, which are from sensors to the sink, and downstream flows, which are from the sink to sensors. ART

assumes these two types of flows have different loads and require different levels of reliability. A downstream flow has lighter load but requires higher reliability that all packets of this flow should be received by the sensors. An upstream flow, on the contrary, has heavier load but requires lower reliability that the sink does not have to receive every packet as long as it acquires the event. ART also assumes that congestion seldom happens in downstream flows but frequently occurs in upstream flows.

ART proposes a centralized, energy-aware sensor classification algorithm to elect a minimal set of nodes that have higher energy levels and covers the whole target area. The nodes in this set are called *essential nodes* (E-nodes), and all the other nodes in the network are called *non-essential nodes* (N-nodes). This algorithm is executed periodically in order to prolong the network lifetime.

ART provides reliability by using asymmetric acknowledgement (ACK) and negative acknowledgement (NACK) between E-nodes and sink. In a downstream transfer, the sink sends a set of queries to an E-node. Each query has a sequence number. If the query is the last one in the set, the sink sets the Poll/Final (P/F) bit. When an E-node receives the queries, it checks sequence numbers to detect query loss. When a gap in sequence numbers is detected, E-node sends a NACK with the sequence numbers of the lost queries back to the sink. The sink retransmits the lost queries after receiving a NACK. When an E-node successfully receives all queries, it checks the P/F bit in received queries. If the P/F bit is set, the E-node sends a ACK to sink. The sink periodically sends message with P/F bit set until the ACK is received.

For upstream reliability, ART proposes a low overhead ACK mechanism. When a new sensing value is obtained, an E-node decides whether the value is an event-alarm. If it is an event-alarm, the E-node sets the *Event Notification* (EN) bit in a new message and saves it into the buffer until the message gets confirmed. Otherwise, the E-node sends a new message to the sink and removes it from the buffer. The sink only sends ACKs to messages with the EN bit set. If an E-node does not receive an ACK for an event-alarm after a timeout, it retransmits the event-alarm message.

Table 3.1: Attribute comparison of transport protocols providing both end-to-end reliability and congestion control

| **Protocols** | **Reliability** | | | | **Congestion Control** | | | | **Energy Control** | **Buffer Management** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Level** | **Direction** | **Type** | **Notification** | **Type** | **Detection** | **Notification** | **Alleviation** | | |
| **Hop** | packet | both | virtual retransmission | BSYN/ BACK | hop-by-hop | backpressure | N/A | hold blocks | using block and BSYN/ BACK to reduce overheads | N/A |
| **RCRT** | packet | upstream | end-to-end | NACK | end-to-end | loss recovery time | explicit | adjust rate by AIMD | N/A | N/A |
| **STCP** | event / packet (variable levels) | upstream | end-to-end | NACK/ACK | end-to-end | queue length | implicit | slow down the rate or change to another path | N/A | buffer timer to clean buffer for continuous flows |
| **Flush** | packet | upstream | end-to-end | NACK | hop-by-hop | queue length | N/A | slow down by increasing the delay | N/A | N/A |
| **ART** | event / query (different levels) | both | end-to-end | NACK/ACK | hop-by-hop | service time | explicit | N-nodes temporarily stop sending | energy-aware sensor classification | N/A |

ART provides congestion control only for upstream flows. The congestion control is handled by E-nodes in a distributed manner. When an event-alarm is sent, the E-node triggers a *congestion timeout* (CTO). CTO is dynamically determined through the round trip time (RTT). If an ACK is not received after the CTO, the E-node broadcasts a *congestion alarm* (CA) message to its neighboring N-nodes. After receiving a CA message, N-nodes will temporarily stop sending their sensing messages. If the E-node still does not receive the ACK after another CTO period, it will increase the broadcast hop-count. If the ACK is received, a congestion-safe message is announced by the E-nodes to resume normal operation on the network.

Table 3.1 gives the comparison of the previously discussed transport protocols (Retiring Replicants [71] and Burst Forwarding [33] are not included in the table, since Retiring Replicants does not provide reliability and Burst Forwarding needs to work with TCP to provide end-to-end reliability and congestion control). Except Hop, all the other transport protocols rely on end-to-end mechanisms to provide end-to-end reliability and congestion control. Hence they are less suitable to ICDT-WSNs. In addition, energy efficiency and buffer management are seldom taken into consideration. Without effective energy efficiency methods and buffer management, the limitation on the sensor node energy and storage can degrade the performance of these protocols in ICDT-WSNs. Therefore, transport protocols for ICDT-WSNs have improvement space in disconnection-oriented design, energy efficiency and buffer management in the future.

## 3.2 Routing Protocols

Routing is a challenging topic in ICDT-WSNs, since traditional routing protocols may not be applicable due to the large requirement of memory for complete link-state information. Existing network protocols can be divided into three categories: single-copy [69] (or forwarding), multiple-copy [69] (or flooding) and hybrid. *Single-copy protocols* transmit only one copy of a message along a carefully selected path to the destination. *Multiple-copy protocols* transmit several copies of a message to sensor nodes within a network, and expect that at least one copy will reach the

destination. [48, 44] point out that multiple-copy protocols, such as epidemic based protocols [22], can improve the delivery capacity. However, Xu and Wang [84, 80] have proved that in scenarios where the network connectivity is highly unstable, the buffer occupancy increases as the network size grows. The trade-offs between single-copy and multiple-copy are: 1) multiple-copy protocols can operate with minimal network information, while single-copy protocols need to know more network information to calculate a good path; 2) multiple-copy protocols may cause unnecessary redundancy, but single-copy protocols have to employ expensive packet recovery due to packet loss.

Network protocols can be grouped into proactive or reactive protocols, depending on when they calculate the routing path. In *proactive network protocols*, all routes are computed and maintained before they are needed. *Reactive network protocols* calculate routes only when needed. To find a good path from a source to a destination is comparatively easier for proactive network protocols, because sensor nodes have a complete view of the network connectivities. But proactive network protocols consume more sensor resources to compute and maintain the routing tables, especially when network topologies change frequently. In reactive protocols, sensor nodes can easily compute and maintain routing tables, since the sizes of routing tables are smaller. However extra delay will be introduced due to the path computation before sending a message. Hybrid network protocols combine both approaches.

In ICDT-WSNs, there are times when a complete path from a source to a destination does not exist. As a result, the network protocols that assume the existence of a complete path from a source to a destination can perform poorly. Because of the limitations on energy and storage of sensor nodes, the network protocols that are not energy efficient and storage friendly are less appropriate for ICDT-WSNs. Zhang gave a comprehensive survey for routing in intermittently connected networks in [89] from the perspective of mobile ad hoc networks (MANET) and DTNs. In the rest of this subsection, we outline several recently proposed network protocols for intermittently connected scenarios, and evaluate their network characteristics, node attributes and protocol properties from the perspective of ICDT-WSNs.

*H + 1 hop*: H + 1 hop [22] is a multiple-copy epidemic routing protocol for DTNs. It proposes an approach to reduce the network overhead without sacrificing the delivery speed and delivery rate. H + 1 hop introduces a threshold called *equilibrium point*, and divides the forwarding process into two sub-processes according to this threshold. The equilibrium point is the number of duplicate copies of a packet in a network. In epidemic routing it is critical that the amount of infective nodes grows quickly when the number of copies of a packet is low, conversely, the number of infective nodes should decrease as the number of copies of a packet grows. In the first sub-process, H + 1 hop lets nodes distribute the packet to non-infective nodes in the transmission range. After the number of duplicate copies of the packet reaches to the equilibrium point, H + 1 hop enters into the second sub-process, in which nodes that carry a copy of the packet do not distribute to nodes other than the destination.

*Convergent Hybrid-replication Approach to Routing in Opportunistic Networks (CHARON)*: CHARON [63, 64] aims to provide a simple but efficient solution to address the routing problem in highly mobile, sparse sensor networks where the future topologies of networks are unpredictable. It minimizes the number of messages exchanged and provides a way for urgent messages to be delivered quickly.

CHARON uses *estimated delivery delay (EDD)* as the main routing metric. It forwards messages from nodes with higher EDD to nodes with lower EDD. The EDD of a node $n_j$ is the smallest sum of the $EDD_{n_i}$ and $ICT_{n_i}$ among its neighbors (Equation 3.7), where $EDD_{n_i}$ is the EDD of its neighbor $n_i$, $ICT_{n_i}$ is the *inter-contact time* between $n_j$ and $n_i$ and $K$ is the set of neighbors of $n_j$.

$$EDD_{n_j} = min\{EDD_{n_i} + ICT_{n_i}\}, \forall n_i \in K \tag{3.7}$$

CHARON is an energy and buffer space aware protocol. Other than EDD, CHARON employs a customizable multi-variable utility function to calculate routing scores, which includes battery level, free buffer space and other application-specific ones combined in any number of ways. In order to save energy and reduce the opportunity of a routing loop, messages are forwarded to nodes

with lower EDDs and higher utility scores than the one holding the message.

CHARON is a hybrid network protocol. Nodes forward non-critical messages while keeping a local copy of the messages. The local copies are called *Zombie*s, which cost no extra energy, but do utilize node memory, and can only be directly delivered to the sink. Whenever a regular message is received by the sink, its corresponding zombie is removed. With urgent messages, a flooding mode similar to PROPHET [47] is used to make messages arrive at the sink quickly. With high-priority messages, EDD is the only metric used in path calculation in order to minimize the latency. With low-priority messages, both EDD and the utility score are used in order to extend the network lifetime.

*Replication-Based Efficient Data Delivery Scheme (RED) and Message Fault Tolerance-Based Adaptive Data Delivery Scheme (FAD)*: [81] provides two data delivery schemes for delay/fault-tolerant mobile sensor networks (DFT-MSN). Both schemes provide data transmission and buffer management mechanisms.

Data transmission in RED is based on *nodal delivery probability*, which indicates the likelihood that a node can deliver a data message to the sink. Let $\xi_i$ denote the delivery probability of node $i$, which is initialized to 0. Each sensor node maintains a timer $\Delta$ to calculate its $\xi$. If node $i$ could not transmit any data message during $\Delta$, $\xi_i$ is reduced. If node $i$ transmits a data message to another node $k$, $\xi_i$ is updated according to $\xi_k$. Data messages are stored in a FIFO queue at each node. When node $i$ has a message to send and moves into the communication range of a set of nodes, it earns the delivery probability and available buffer space of each node via simple handshake messages. Then node $i$ transmits the message to neighbor $j$ with available buffer space, the highest delivery probability in the set of neighbor which is also higher than the node's own delivery probability, $\xi_j > \xi_i$.

Message management in RED uses an erasure-coding approach to address the trade-off between delivery ratio/delay and overhead. The table of delivery probability $p$ and corresponding optimal number of blocks $b$ and minimum replication overhead $S$ is stored in each source node. Whenever a source node generates a message, it checks the table to find the optimal $b$ for its current

delivery probability $\xi$ ($\xi = p$), and encodes the data messages into $S \times b$ data blocks, which are then put into the queue for transmission.

FAD is an advanced version of RED that avoids the inaccuracy of the erasure-coding approach used in RED. It takes advantage of both the nodal delivery probability introduced in RED and the message fault tolerance that indicates the redundancy of a message in the network as well as message priority. Unlike RED, nodes that employ FAD keep a copy of the message after transmitting it to another node, which leads to multiple copies of the message and redundancy in the network. FAD assumes that each message carries a field to keep its fault tolerant value, which is defined as the probability that at least one copy of the message is delivered to the sink by other nodes in the network. Let $F_i^j$ denote the fault tolerant value of message $j$ in the queue of node $i$. $F_i^j$ is initialized to zero when a message is generated. After a node $i$ broadcasts a message $j$ to its neighbor nodes, there are $Z + 1$ copies of message $j$ in the network, where $Z$ is the set of neighbor nodes of node $i$. The fault tolerance value of the message transmitted to neighbor node $\psi_z$ is updated according to Equation 3.8.

$$F_{\psi_z}^j = 1 - (1 - [F_i^j])(1 - \xi_i) \prod_{m=1, m \neq z}^{Z} (1 - \xi_{\psi_m}). \tag{3.8}$$

The fault tolerance value of the message at node $i$ is updated according to Equation 3.9.

$$F_i^j = 1 - (1 - [F_i^j]) \prod_{m=1}^{Z} (1 - \xi_{\psi_m}). \tag{3.9}$$

In Equation 3.8 and Equation 3.9, $[F_i^j]$ is the fault tolerance of the message $j$ in the queue of node $i$ before being transmitted.

Fault tolerance value is used to manage storage. The higher the fault tolerance value is, the less important the message is, since more copies of the message are in the network. To decide whether to store an arriving message $M$ at a node follows two rules: 1) if the queue of a node is full and the fault tolerance of the message at the end of the queue is larger than the fault tolerance of $M$, $M$ is dropped. Otherwise, the message at the end of the queue is dropped, and $M$ is inserted

into an appropriate position in the queue according to its fault tolerance; 2) If the fault tolerance of $M$ exceeds a threshold, $M$ is dropped even if the queue is not full.

Data transmission in FAD is based on the delivery probability introduced in RED. When node $i$ has a message $j$ at the front of its queue ready to transmit, and is within the communication range of a set of nodes $Z$, node $i$ multicasts message $j$ to all nodes in $Z$ that have higher delivery probabilities than node $i$ and available buffer space. Meanwhile, node $i$ sets the fault tolerance of message $j$ to just below the threshold in order to reduce unnecessary transmission overhead.

*Shortest path routing protocol*: [31] proposes a shortest path network protocol for DTNs, which is basically a link-state protocol in which each node has a complete view of the network topology. It assumes that the packet loss is mainly due to buffer overflow, and the waiting time for connections is the main factor of end-to-end delay. Shortest path routing protocol is based on traditional network routing, such as OSPF, but modifies the decision making strategy for a delay-tolerant environment.

The shortest path routing protocol uses per-contact routing, which re-computes routing tables when a contact arrives. In order to improve performance, it reduces the time for waiting for a better connection by using *short circuiting*. Short circuiting temporarily assigns an available contact a cost of zero in nodes' local routing tables whenever a contact becomes available. This temporary value is only used to compute the shortest route, but not propagated to other nodes. The combination of per-contact and short circuiting guarantees routing decisions are made with the most recent information and employs serendipitous contact. When making a new connection, nodes exchange summary vectors that list link-state tables received by nodes. Each link-state table has a sequence number, so that nodes can identify the most recent table. After exchanging missing updates, nodes re-compute routing tables and forward messages to other nodes. The *Minimum Estimated Expected Delay (MEED)*, as shown in Equation 3.10, is the metric used in this protocol to assign a cost to

each contact.

$$\textit{MEED} = \frac{\sum_{i=1}^{n} d_i^2}{2t}, \tag{3.10}$$

where $n$ is the total number of disconnected periods, $d_i$ is the duration of a given disconnected period, and $t$ is the total time interval to observe these disconnections.

*Spray and Focus*: Spray and Focus [68] is a hybrid network protocol for intermittent connected networks. The protocol follows the basic idea of *Spray and Wait* [66] but improves the *Wait phase* by letting each relay forward its copy further. It assumes that nodes move slowly and periodically transmit beacons to recognize each other's presence. Each node maintains a summary vector, a set of timers and a set of utility values. A summary vector lists message IDs a node has stored and relayed. A timer records the time since two nodes last saw each other. A utility value indicates the possibility a node can deliver a message to another node.

In *Spray phase*, the source node creates $L$ *forwarding tokens* for a message it generates. When two nodes meet, they exchange their summary vectors and check for common messages. If a node $A$ carrying a message copy with forwarding tokens $n > 1$ encounters a node $B$ with no copy of this message, $A$ spawns and forwards a copy of the message with $\lfloor n/2 \rfloor$ tokens to $B$, and reduces the tokens of its own copy to $\lceil n/2 \rceil$. Initially, $n = L$. But if a node has a message copy with only one token, it switches to Focus phase.

In *Focus phase*, the forwarding decision is according to the node's utility. A node $A$ forwards to node $B$ a message destined for node $D$, if and only if Equation 3.11 is true.

$$U_B(D) > U_A(D) + U_{th}, \tag{3.11}$$

where $U_B(D)$ is the utility value of B for D, $U_A(D)$ is the utility value of A for D, and $U_{th}$ is a utility threshold. The timer is one possible utility metric. Let $\tau_i(j)$ denote the timer of node $i$ for

node $j$ in the network. Initially, set the timer according to Equation 3.12.

$$\forall i, j : \tau_i(i) = 0, \tau_i(j) = \infty \tag{3.12}$$

Whenever $i$ encounters $j$, update the timer according to Equation 3.13.

$$\tau_i(j) = \tau_j(i) = 0 \tag{3.13}$$

At every clock tick, the timer is increased by one (Equation 3.14).

$$\tau_i(j) = \tau_i(j) + 1, \tau_j(i) = \tau_j(i) + 1. \tag{3.14}$$

According to the forwarding rule in the focus phase, when node $A$ encounters node $B$ at distance $d_{AB}$, it can decide whether $B$ is the next node to forward a message copy according to Equation 3.15,

$$\forall j \neq B : \tau_B(j) < \tau_A(j) - t_m(d_{AB}), \tag{3.15}$$

where $t_m(d_{AB})$ denotes the expected time node $A$ takes to move to node $B$ under a given mobility model $m$.

Each message carries a time-to-live value. If the value expires, the message is removed from the node buffer and its record is also deleted from the node's summary vector.

*TTL-based routing (TBR)*: TBR [56] aims at maximizing the delivery ratio with minimum network overhead in mobile opportunistic networks through a hybrid scheme. The *time-to-live* (TTL) of a message is used to manage a node buffer and indicates the delivery priority of messages.

Table 3.2: Comparison of disconnection-oriented routing protocols

| | **H + 1 hop** | **CHARON** | **RED** | **FAD** | **Shortest Path Routing Protocol** | **Spray and Focus** | **TBR** |
|---|---|---|---|---|---|---|---|
| **Network Model** | DTN | Low density opportunistic WSNs | DFT-MSN | DFT-MSN | DTN | Intermittently connected mobile networks | Mobile opportunistic networks |
| **Node Mobility** | Normal | High mobility | Normal | Normal | - | Slow mobility | Normal |
| **Energy Aware** | × | ✓ | × | × | × | × | × |
| **Buffer Aware** | × | ✓ | ✓ | ✓ | × | × | ✓ |
| **Buffer Management** | × | × | ✓ | ✓ | × | × | ✓ |
| **Buffer Size** | Sufficient | Limited | Limited | Limited | Sufficient | Sufficient | Limited |
| **Computation Complexity** | Simple computation | Simple computation | Requires extra computation for decoding | Simple computation | Computation and exchange | Simple computation | Simple computation |
| **Single-copy case/ Multiple-copy case/ Hybrid** | Multiple-copy | Multiple-copy (urgent messages) + Hybrid (normal messages) | Single-copy | Multiple-copy | Single-copy | Hybrid | Hybrid |

In TBR, each message has a replication count field ($L_k$), a TTL field and a list of *previously visited nodes* (LVN). The replication count field indicates how many copies of this message a node can spray. The $L_k$ is set to a user-defined algorithm parameter $L$ initially, and updated after each successful message transmission. The TTL field specifies the valid time (in minutes) of the message in the network. The LVN is used to avoid routing loop. A message is only forwarded to neighbors that are not in the message's LVN.

Each node maintains a forward list, a delete list and an acknowledged message list. The *forward list* stores the forwarding priorities of messages. When a contact becomes available, the message on the top of the list will be delivered first. Let $P_{k_f}$ denotes the forwarding priority of the message $m_k$ (Equation 3.16),

$$P_{k_f} = \frac{1}{H_k \times TTL_k \times s_k}, \tag{3.16}$$

where $TTL_k$ is the TTL of $m_k$, $H_k$ is the hop count of $m_k$ and $s_k$ is the size of $m_k$. The *delete list* stores the a prioritized list of messages to be deleted. If a node buffer is full when it receives another message with higher priority, the message at the end of the list will be deleted from the buffer. Let $P_{k_d}$ denote the deleting priority of the message $m_k$ (Equation 3.17),

$$P_{k_d} = \frac{L_k}{s_k}, \tag{3.17}$$

where $L_k$ is the replication count field value of $m_k$, and $s_k$ is the size of $m_k$. The *acknowledged message list* records the messages that have been delivered successfully.

When two nodes encounter each other, they first exchange their acknowledged message lists. If nodes have messages in their buffer that are in the acknowledged message lists, the messages are removed. If a node has some messages destined to the other node, it transmits those messages. If a node has forwarding messages, it picks the messages from the top of the forward list as long as the messages meet the conditions $L_k > 1$, $s_k < S_{max}$ and the other node is not in the messages' LVNs. $S_{max}$ is the maximum transmittable message size of a contact. TBR calculates it every time when a new contact is available. The $L_k$ of each message is divided by 2 before forwarding a copy

to the other node. When $L_k = 1$, a message can only be delivered to the destination directly or dropped from the buffer when its TTL expires.

Considerations on energy efficiency, buffer management and computation complexity are necessary and important to network protocols in ICDT-WSNs. Table 3.2 gives a comparison of the protocols mentioned in this subsection according to these considerations. As can be seen, none of these network protocols meets all requirements of ICDT-WSNs. The reason is that these network protocols are designed for DTNs, opportunistic networks, MANETs and DDTMs [12], but not for ICDT-WSNs. Hence, energy efficient network protocols for ICDT-WSNs with buffer management and low computation complexity are a future research direction.

### 3.2.1 Information Dissemination

To disseminate information to nodes in the *region of interest (ROI)* has been extensively studied in mobile networks. Due to the uncertainty of the information destinations, the protocols used to disseminate information are usually operate without knowledge of the destinations. In vehicular ad hoc networks (VANETs), broadcast is a wildly applied technique to disseminate information [73, 9, 61, 88]. In order to provide efficient information dissemination, much of existing work in VANETs aims to mitigate the broadcast storm [74, 82].

Although broadcast storm mitigation can enhance the protocol efficiency, the amount of energy spent on duplicate information transmissions is significant. Especially in intermittently connected mobile wireless sensor networks where the nodes are with strict power limitations, those broadcast mechanisms can drain energy quickly. Epidemic routing [75] is a flooding based protocol for intermittently connected mobile wireless sensor networks. It utilizes summary vector exchange to eliminate the transmissions of duplicate information. However, Epidemic routing aims to deliver information to every node in a network, therefore, amount of energy is consumed by the transmissions to the nodes outside of the ROI that are not interested in the information.

Many flooding based protocols try to enhance the efficiency of Epidemic routing by reducing unnecessary transmissions. However, most of them reduce unnecessary transmissions by using the

knowledge of the destination information. Therefore, these mechanisms cannot provide efficiency in regional information dissemination scenarios, since the destination information is uncertain in such scenarios. Copy-control is a mechanism to reduce unnecessary transmissions [67, 68, 49, 56]. The number of allowed copies of a packet is calculated through the estimated delay to deliver a packet to the destination. When the destinations are uncertain, the estimated delay is hard to predict. Another approach is to calculate the probability to meet with nodes in a network, and rely on the calculated probabilities to route packets to nodes that are more likely to deliver the packets to the destinations [47, 21]. These methods can eliminate unnecessary transmissions to the nodes that are less likely to meet with the destinations. However, they are infeasible in the regional information dissemination scenarios where the destinations are unknown. H + 1 hop [22] takes advantage of the SIR model of Epidemic Theory to reduce unnecessary transmission. It is difficult to apply SIR model in the ROI of a target where the nodes in the ROI are highly unstable.

Other recent papers [79, 46] discuss information dissemination from different perspectives: a study of the dissemination properties under Levy Mobility is given in [79], a study of dissemination in static energy-harvesting wireless sensor network is given in [46].

## 3.3 Link Layer

The communication media of ICDT-WSNs is a wireless channel whose nature is broadcast. This feature makes the medium access and data transmission over the common medium complicated. Moreover, restrictions on nodes' energy, storage and computation capability, asymmetric links, dynamic topology, and the large number of nodes of a network make media access control (MAC) protocols for WSNs distinct from the protocols for traditional networks.

Due to the power limitation of sensor nodes, energy efficiency is a primary goal in MAC protocol design. The scalability of the network size, topology and node density is a further goal. Because all nodes in a WSN cooperate together to accomplish a mission, the requirements on fairness, latency, throughput and bandwidth utilization are less strict than in traditional networks.

There are four factors that can drain energy quickly without effective management: collisions, overhearing, overhead and idle listening. Much work on addressing these four issues has been done over the past decades, which can be divided into four groups: scheduled protocols, protocols with common active periods, preamble sampling protocols and hybrid protocols that combine the techniques used in the former three groups [5].

In *scheduled protocols*, all nodes in the network follow a well-designed schedule to communicate. Time division multiple access (TDMA) is a canonical module. A well-designed schedule can avoid collisions and overhearing, and can minimize idle listening. However, to design such a schedule for a large WSN is overly complex. Additionally, the scalability and flexibility of scheduled protocols are limited in scheduled protocols.

In *common active period protocols*, sensor nodes have common active periods to communicate. So synchronization is required in this category of protocols. These protocols reduce the energy spent in idle listening, but bring up the problem on the suitable length of active slots and the problem of sleep delay. Even though sleep delay is not a significant problem in ICDT-WSNs, the length of the active slot is tightly relevant to energy efficiency.

*Preamble sampling protocols* reduce the energy spent on synchronization by using long preambles, but the energy consumption is transferred from receivers to transmitters. Receivers only need to wake up for a short period of time to sense the channel, but transmitters need to spend more energy transmitting long preambles. This is acceptable when the traffic is light; otherwise, long preamble transmitting will consume too much energy. Moreover, if collisions happen, the energy spent on long preambles becomes increasingly problematic. The duty cycle is also limited in this type of MAC protocols, since the length of preamble is relevant to the channel check interval of nodes.

Most of the MAC protocols for WSNs are suitable for ICDT-WSNs when the propagation delay in networks is normal (e.g., RF communication in terrestrial environment). However in the scenario where the propagation delay is very long (e.g., acoustic communication in underwater environment), propagation-delay tolerant MAC protocols are required [45, 85]. In this subsection,

we focus our study on MAC protocols designed for the scenarios where propagation delay is not a significant problem. We outline several protocols that are well known or recently designed in each category, and study their advantages and disadvantages.

*Traffic-adaptive medium access protocol (TRAMA)*: TRAMA [58] assumes there is a single, time-slotted channel for both signaling and data transmission, and adequate synchronization is attained. Time is organized as signaling slots and transmission slots. Signaling slots are random-access, and transmission slots are scheduled-access. TRAMA consists of three components: the Neighbor Protocol (NP), the Schedule Exchange Protocol (SEP) and the Adaptive Election Algorithm (AEA). TRAMA starts in signaling slots in which nodes use NP to obtain two-hop neighborhood information. Then during transmission slots, nodes use SEP to build and maintain traffic-based schedules for the one-hop neighbors before deciding their state (transmit, receive or sleep) by AEA. AEA selects collision-free transmitters and receivers based on the information from NP and SEP. In this way, nodes sleep until they need to transmit or receive in the transmission slots.

*Multi-Channel Lightweight Medium Access Control (MC-LMAC)*: MC-LMAC [27] is a multi-channel access scheduled MAC protocol, which aims at improving the achievable throughput in WSNs rather than energy efficiency. In MC-LMAC, each node chooses one time slot from a channel in a distributed way, which guarantees that the same slot/channel pair is not used by more than one node which avoids conflicting transmissions.

Nodes in MC-LMAC transit between five states: initialization, synchronization, discovery, time-slotted channel selection, and medium access. When nodes join the network, they begin in the initialization state in which nodes sample the medium for incoming packets to synchronize with the network. If such packets are received, nodes move into the synchronization state and synchronize with the network by the current time slot information and frame number carried in the packets. Then, nodes follow the schedule to receive packets in the upcoming slots. If nodes have data to send, they pick up a random wake-up frame in channels. Before the wake-up frames, nodes enter the discovery state. In this state, each node gets a list of free time-slots and channels in neighborhood. In the list, each entry represents one channel, which is a string of 1s or 0s that

indicates whether the slot is used or free with the length of the string equaling the number of time-slots in that channel. After the discovery state, nodes come into the time-slotted channel selection state. In this state, every node executes bitwise OR operation against the list to pick a free time-slot channel pair that is not used by the node's one-hop neighbors. If a node successfully chooses an empty time-slot, the node proceeds into the media access state, otherwise, it goes back to the synchronization state. In the media access state, nodes can transmit data in their selected time slots of the selected channels. During other slots, nodes detect potential incoming packets. If collisions are detected, the nodes go back to the synchronization state. If a synchronization error happens, the nodes go back to the initialization state.

*S-MAC*: S-MAC [86] assumes that applications have long idle periods, and can tolerate some latency. S-MAC reduces idle listening by letting nodes sleep for a fixed period and then wake up for the same duration to listen for any node that wants to talk to it. In order to reduce control overhead, neighboring nodes need to synchronize to have the same listen/sleep schedule. In this way, each node maintains a schedule table, which stores the schedules of all its neighbors. In the beginning, nodes need to build the schedule tables. First, every node listens for a period of time. If no schedule is received during this time, it randomly chooses a time to go to sleep and broadcasts a SYNC message with its schedule to inform that it will go to sleep after $t$ seconds. If a schedule is detected before the node chooses its own schedule, the node sets its schedule to the received schedule, and rebroadcasts the schedule after a random delay $t_d$ to inform that it will go to sleep after $t - t_d$ seconds. If a node receives a different schedule after it has set and broadcasted its own schedule, it adopts both schedules. The listening duration is divided into two parts. The first part is for nodes to synchronize to have the same listen/sleep schedule through SYNC packets. Nodes that receive SYNC packets adjust their timers right away. The second part is for receiving RTS (request to send) packets. Both parts are divided into several time slots for senders to detect collisions (carrier sense).

In order to avoid collisions, S-MAC sets a duration field in each transmitted packet to indicate how long the remaining transmission will be. If a node receives a packet that is not destined for

it, the node records the duration field value called *network allocation vector* (NAV) and sets a timer. NAV indicates how long the receiving node should keep silent. Once the timer starts, the NAV value is decreased until it reaches to zero. When a node has data to send, it first checks the current NAV value. If the NAV value is zero, the node performs carrier sense. Broadcast packets are sent without using RTS/CTS (clear to send) packets. Unicast packets are sent after exchanging RTS/CTS. After a receiver receives a unicast packet, it sends ACK back to the sender. If a node fails to acquire access to the medium, it goes back to sleep until the receiver is free and listens again.

To avoid overhearing, S-MAC let nodes that hear the RTS or CTS packets sleep until the current transmission is over. When a node receives a packet destined for other nodes, it updates its NAV by the duration field in the packet.

S-MAC reduces the resources spent on retransmitting a long data packet by divided the packet into many small fragments and transmits them in a burst. A sender waits for ACK from its receivers for each fragment. If an ACK is not received, the sender extends the reserved transmission time for one more fragment, and retransmits the packet.

*Timeout-MAC (T-MAC)*: T-MAC [76] follows the basic idea of S-MAC with the active/sleep schedule and the way nodes decide their schedules. However, instead of using the fixed duty cycle in S-MAC, T-MAC introduces an adaptive duty cycle to improve energy efficiency. In T-MAC, nodes periodically wake up to communicate with their neighbors, and then go back to sleep until the next frame. During the sleep time, new generated messages are queued. The communication between nodes follows the sequence of RTS/CTS/DATA/ACK similarly to S-MAC. In the active period, nodes keep listening and potentially transmit. When the active period ends, nodes go back to sleep. If no active event occurs for a time $TA$, the active period ends, which is called the adaptive duty cycle. In order to optimize the sleep period, T-MAC moves all communication to the beginning of the active time in a burst. The minimum length of $TA$ should be larger than the maximum contention duration and the time to exchange RTS and CTS. The length of $TA$ is a key factor in energy consumption. The longer $TA$ is, the more energy is consumed. By reducing the

active duration, T-MAC can achieve better energy efficiency.

The adaptive duty cycle can cause an early sleep problem in which a node goes to sleep when its neighbor still has data to send to it. To solve this problem, T-MAC uses the *future-request-to-send* (FRTS) packet, which indicates how long a node needs to be awake. A FRTS packet is sent right after a node overhears a CTS message destined for another node. When a node receives a FRTS, it will stay active for the time indicated by the FRTS rather than going back to sleep. Then its neighbor can send data to it after the channel is clear. This feature increases the throughput of T-MAC by introducing additional overhead.

*B-MAC*: B-MAC [54] is a preamble sampling protocol that searches for outliers during channel arbitration. If there exists an outlier from the received signals during the sampling period, B-MAC declares the channel is clear. If five samples are taken and no outlier is detected, the channel is busy. So a good estimation of noise floor is important in this procedure. In B-MAC, each node takes signal strength samples at a time when the channel is assumed to be clear, such as immediately after transmitting a packet. The average value of the samples is used as a simple low pass filter for the noise floor estimate.

During the *low power listening* (LPL) state, a node wakes up and turns on its radio to check activity. If no activity is detected, the node is forced back to sleep after a timeout. If activity is detected, the node turns power on and stays awake for the time required to finish receiving the incoming packet. After reception, the node goes back to sleep. The length of preamble should be at least the interval that the channel is in LPL state.

*X-MAC*: X-MAC [7] ameliorates the overhearing problem by dividing one long preamble into a series of short preamble packets with a small gap between every two packets, during which the transmitter pauses transmitting. Each preamble packet contains the ID of the target node. When a node wakes up and receives a preamble packet, it looks up the target ID in the packet. If the node is not the target, it goes back to sleep immediately. If it is the target, it sends an acknowledgement to the transmitter during the gap between the preamble packets. After the transmitter receives an acknowledgement, it stops transmitting preamble and starts sending the data packet. Similar to

B-MAC, the length of the preamble sequence must be greater than the maximum sleep period of receivers.

When multiple transmitters want to send packets to the same node, there is no need to send preambles to the node that is already awake. X-MAC addresses this problem by letting the receiver keep awake for a short period of time after receiving a data packet. During this period, if a transmitter hears the acknowledgement from its target node while waiting for a clear channel, it waits for a random back-off time and then transmits without sending any preamble. The back-off time should be long enough to allow the on-going transmission to finish, and the active time of the receiver after receiving a data packet should be equal to the longest back-off time.

*Zebra MAC (Z-MAC)*: In Z-MAC [59], carrier sense multiple access (CSMA) is used inside a large time division multiple access (TDMA) slot, such that Z-MAC can utilize CSMA when the traffic is light and switch to TDMA if the traffic becomes heavy. Initially, each node runs the DRAND [60] algorithm to be assigned a time slot that is unique within the node's two-hop neighborhood. In *low contention level* (LCL), a node can compete for any time slot. However, in *high contention level* (HCL), a node can compete for a slot only when it is the owner of the slot or a one-hop neighbor of the owner of a slot. When a node has data to transmit, it always performs carrier sense first and transmits after the channel is clear. If the node is the owner of a slot, it has the highest priority to send. Otherwise, it needs to wait a random back-off time, after which the node can transmit in that slot if the slot is still unused. The back-off time needs to be long enough in order to allow the owner access to the slot.

A node is in the HCL when it receives an *explicit contention notification* (ECN) message from a two-hop neighbor within the last $t_{ECN}$ period. Otherwise, it is in LCL. ECN messages are sent by a transmitter when it detects the channel is in high contention by measure the noise level of the channel. When a node receives an ECN message, it sets a local HCL flag. The HCL flag is automatically reset unless the node receives another ECN message within $t_{ECN}$. The system sets the refresh time $t_{ECN}$.

Table 3.3: Comparison of MAC protocols for WSNs

| Protocol | Mode | Advantages | Disadvantages |
|---|---|---|---|
| **TRAMA** | Schedule based | • Automatically adapt scheduling to traffic load.<br>• Nodes wake up only when they have data to send or receive | • Computation complexity is high.<br>• Assume adequate synchronization among nodes. |
| **MC-LMAC** | Schedule based | • Provide high throughput<br>• Low computation complexity. | • Low energy efficiency<br>• Multiple channels required. |
| **S-MAC** | Common active period | • Set up a common active period to reduce overhead.<br>• Reduce overhearing in unicast.<br>• Reduce energy spent in retransmission by dividing a long data packet into small fragments.<br>• Comparatively less computation. | • Applications must tolerate some latency.<br>• Fixed duty cycle. |
| **T-MAC** | Common active period | • Adaptive duty cycle to reduce energy consuming. | • Additional overhead needed to support adaptive duty cycle. |
| **B-MAC** | Preamble based | • Low computation complexity.<br>• No synchronization needed. | • Long preamble causes unnecessary overhearing. |
| **X-MAC** | Preamble based | • Reduce unnecessary overhearing by dividing long preamble into a series of small preamble packets.<br>• Keep nodes awake after finishing receiving data to avoid sending preambles to awake nodes. | • Introduce new overheads. |
| **Z-MAC** | Hybrid (CSMA/TDMA) | • Increase channel utilization.<br>• No extra overhead for collision avoidance after initial set up phase. | • Not easy to introduce new nodes, since time slots are assigned to nodes when the network is developed.<br>• Comparatively high computation complexity during the initial set up phase. |

As we can see from table 3.3, each link-layer protocol has its own benefits and drawbacks. Most of the protocols are either suitable for light contention scenarios or heavy contention scenarios. While those protocols suitable for both scenarios have high computation complexity. For applications whose traffic is predictable, most existing protocols can be applied. However, for those unpredictable situations, protocols that are without high computational complexity and can handle different scenarios and are preferable.

## 3.4 Open Problems and Research Directions

In Table 3.4, we check the communication protocols mentioned in the previous section against the requirements of ICDT-WSNs. From the table, we can see that seldom protocols meet all requirements, since the existing communication protocols are not designed for ICDT-WSNs. As a result, there is a lot of space for further research. In this section, we outline some of the open problems in ICDT-WSNs and suggest the research directions in addressing these problems.

### 3.4.1 Open Problems and Research Directions in Transport Protocols

In Section 3.1, we discuss the existing transport protocols, and compare several protocols designed in recent years from reliability, congestion control, energy efficiency and buffer management in Table 3.1. Due to the intermittent connection of ICDT-WSNs, the hop-by-hop transport protocols that provide end-to-end reliability and congestion control through hop-by-hop manner are more appropriate. However, existing hop-by-hop transport protocols do not take energy efficiency and buffer management into account. As a result, energy efficiency and buffer management become the open problems in transport protocols.

In order to be energy efficient and storage friendly, an effective buffer management mechanism in a transport protocol can save space for more important messages, and can reduce energy consumption in unnecessary retransmissions. Better-designed transport protocols can take better advantage of the information from the control messages to improve reliability, congestion con-

trol and buffer management. As a result, the energy consumption can be reduced by reducing the volume of the control message traffic.

Recently proposed hop-by-hop transport protocol [42], *Acksis*, providing both reliability and congestion control, addressed the buffer management problem by taking advantage of the per-block based acknowledgement. Through this mechanism, Acksis can enhance the network delivery rate by employing in-network storage regardless of the traffic situations of ICDT-WSNs.

### 3.4.2 Open Problems and Research Directions in Network Protocols

Section 3.2 introduces the categories of existing network protocols. Table 3.2 gives the comparison for several network protocols designed in recent years from network characteristics, node attributes and protocol properties. Reactive protocols are more suitable for ICDT-WSNs due to the intermittent connectivity. Reactive protocols for ICDT-WSNs should also consider the energy efficiency, buffer management and computation complexity. There are several reactive/ protocols with simple computation complexity, however, only a few of them take energy efficiency and buffer management into consideration. Hence, energy efficiency coupled with buffer management is also an open problem in network layer protocols.

We argues that hop-by-hop transport protocols are more appropriate to ICDT-WSNs, hence, the cross-layer protocols that combine reactive network protocols with hop-by-hop transport protocols is an approach for the open problems of transport and network layers. Employing such a cross-layer protocol can introduce less overhead than employing a transport protocol and a network protocol. As a result, this type of cross-layer protocols can be more energy efficient. In addition, the storage management mechanism of this type of cross-layer protocol can address the insufficient storage problem in both transport layer and network layer.

Recently proposed scheme SMITE [24] addressed the storage limitation of nodes by applying *Bloom filter*. Bloom filter can improve the storage capability of nodes with low computation complexity. However, it introduces the complexity to higher layer when providing reliability and integrity. Diff-Max [62], another recent work, proposed a new framework to separate routing and

Table 3.4: Comparison of communication protocols with ICDT-WSN requirements

| Protocol | Disconnection-oriented | Energy Efficiency | Storage Friendly | Computation Simplicity |
|---|---|---|---|---|
| **Hop** | ✓ | ✓ | × | ✓ |
| **RCRT** | × | × | × | ✓ |
| **STCP** | × | × | ✓ | ✓ |
| **Flush** | × | × | × | ✓ |
| **ART** | × | ✓ | × | × |
| **H + 1 hop** | ✓ | × | × | ✓ |
| **CHARON** | ✓ | ✓ | × | ✓ |
| **RED** | ✓ | × | ✓ | × |
| **FAD** | ✓ | × | × | ✓ |
| **Shortest Path Routing Protocol** | ✓ | × | × | × |
| **Spray and Focus** | ✓ | × | × | ✓ |
| **TBR** | ✓ | × | ✓ | ✓ |
| **TRAMA** | N/A | ✓ | N/A | × |
| **MC-LMAC** | N/A | × | N/A | ✓ |
| **S-MAC** | N/A | ✓ | N/A | ✓ |
| **T-MAC** | N/A | ✓ | N/A | ✓ |
| **B-MAC** | N/A | ✓ | N/A | ✓ |
| **X-MAC** | N/A | ✓ | N/A | ✓ |
| **Z-MAC** | N/A | ✓ | N/A | × |

scheduling in backpressure algorithm for better throughput, easier implementation and flexibility. Meanwhile, Ji at al. [30] have explored the inefficiencies of backpressure. As a result, the backpressure algorithm can lead to energy inefficiency. These recently proposed works address the open problems in network protocols in ICDT-WSNs through either introducing new schemes or optimizing the existing mechanisms to be more energy efficient or storage friendly. However, these approaches do not fully address the open problems and leave space open for future research.

### 3.4.3 Open Problems and Research Directions in MAC Protocols

The MAC protocols are discussed in Section 3.3. Most of the existing MAC protocols are traffic-dependent efficient. The protocols that are efficient in both heavy and light traffic are with high computation complexity. Accordingly, the protocols with low computation complexity that are traffic-independent efficient is an open problem to further research.

Preamble based protocols can save energy spent on synchronization when traffic is light, while common active period protocols can save energy spent on preamble sending when traffic is heavy. Both categories have better scalability and less computation complexity than scheduled protocols. Thus, the combination of common active period protocols and preamble based protocols with low computational complexity is a way to handle the scenarios where the traffic is unpredictable.

# Chapter 4

# Acknowledgement-assisted Storage Management Transport Protocol

Hop-by-hop transport and in-network storage are mechanisms suitable for ICDT-WSNs. Hop-by-hop congestion control can react rapidly after congestion is detected. In-network storage can significantly reduce the number of end-to-end recoveries under the condition that the storage is plentiful. In WSNs, this assumption is not appropriate, since the storage capability of nodes is very limited. Typically the amount of the storage in one node is less than 512 KB. Table 4.1 lists the storage capability of several nodes available today [55]. From the table, we can see that storage management is necessary for hop-by-hop transfer and in-network storage in WSNs. Moreover, because of the requirement for low cost nodes, to add extra storages to nodes is not an optimal solution. Hence, storage limitations must be taken into consideration for in-network storage in WSNs. Otherwise, the network could be overloaded, and its performance will be degraded.

*Transport Protocol with Acknowledgement-assisted Storage Management (Acksis)* [42] is a communication protocol designed for ICDT-WSNs. Acksis implements end-to-end reliability and congestion control in a hop-by-hop manner. It employs overhearing and in-network storage to reduce end-to-end retransmissions when packet loss happens. It also takes advantage of backpressure and acknowledgement messages to achieve congestion control and to manage node storage in

Table 4.1: Node's extra nonvolatile storage size

| Node | Storage size(KB) |
|---|---|
| WeC | 32 |
| Renè | 32 |
| Renè2 | 32 |
| Dot | 32 |
| Mica | 512 |
| Mica2Dot | 512 |
| Mica2 | 512 |
| Telos | 128 |

order to minimize storage overflow and increase delivery rate without sacrificing delivery speed.

## 4.1 Protocol Overview

In Acksis, packets are grouped into blocks to send. The block size is pre-calculated through the number of nodes in a network and the minimum storage size of nodes in the network, which will be explained in Subsection 4.3. An example of Acksis communication is shown in Figure 4.1. Figure 4.2 is the state diagram of storage management of Acksis.

There are four acknowledgement messages in Acksis:

**FIN:** sent by a block $B$'s sender to $B$'s receiver after the last packet of $B$ is sent, to indicate the end of the block to the receiver.

**RPLY:** sent by $B$'s receiver to $B$'s sender after the receiver receives the FIN of $B$ to guarantee hop-by-hop reliability. If there is packet loss, indicate the lost packet number in the message.

**E2ERPLY:** sent by $B$'s destination to $B$'s source after $B$ has been received completely to notify that $B$ has been received by the destination successfully. Nodes execute storage manage-

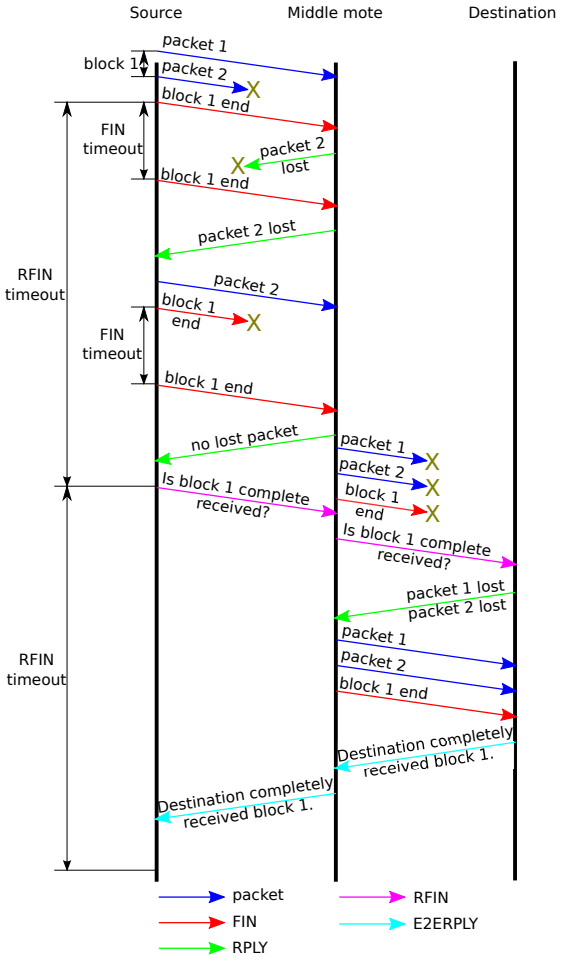



Figure 4.1: One example of Acksis communication

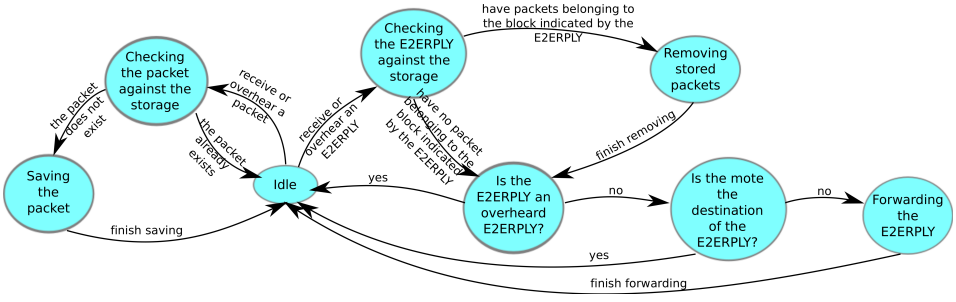



Figure 4.2: State diagram of storage management of Acksis

ment when they receive or overhear an E2ERPLY message, which will be interpreted in Subsection 4.4.

**RFIN:** sent by $B$'s source if it does not receive E2ERPLY of $B$ after a certain period of time to guarantee end-to-end reliability.

Figure 4.1 shows an example packet exchange between nodes using Acksis. During this communication process, nodes execute storage management when they receive or overhear a packet or E2ERPLY message according to the state machine in Figure 4.2. In the rest of this section, we outline how Acksis works.

Initially, a sender sends a block $B$ to a receiver followed by a FIN message. The receiver stores packets of $B$ that have not been cached in its storage while increasing its backlog value by one per packet. Any node that overhears the transmission will store overheard packets that are not already stored. After receiving the FIN message of $B$, the receiver checks for lost packets in $B$ before sending an RPLY message to the sender. When the sender receives the RPLY message, it retransmits any lost packets. This process continues until the receiver completely receives $B$. If there are no lost packets, the sender decreases its backlog value by the block size.

If several nodes send FIN messages to the same receiver simultaneously, then the receiver serves the first arriving FIN message. A node does not acknowledge more FIN messages until it forwards one more block to a downstream node. If there is no RPLY message received before timeout, the sender resends the FIN message. The sender cannot send more blocks if the current one has not been completely received by the receiver.

When $B$ reaches its destination, the destination sends out an E2ERPLY message to $B$'s source. When an intermediate node receives the E2ERPLY message, it deletes the cached $B$ from its storage, and forwards the E2ERPLY message to the next hop. If the E2ERPLY message reaches $B$'s source before timeout, the source deletes $B$ from its storage and is ready to send the next block. Otherwise, the source sends an RFIN message. Nodes that overhear the E2ERPLY message remove $B$ from their storage, but do not forwarded the E2ERPLY message further.

When a node receives the RFIN message for $B$, it searches the storage to see whether all

packets within $B$ are cached. If there are missing packets, the node sends an RPLY message to the sender of the RFIN message to ask for a retransmission. The process continues until the node receives the whole block $B$. It then forwards the RFIN message to the next hop. If there are no missing packets, the RFIN message is forwarded to the next node directly. If the node is the destination, an E2ERPLY message for $B$ will be sent out.

The main differences between Acksis and Hop [37]:

- In Acksis, the block size is pre-calculated through the number of nodes in a network and the minimum storage size of nodes in the network, which will be illustrated in Subsection 4.3.
- In Acksis, nodes take advantage of the received and overheard E2ERPLY message to manage the storage.

## 4.2 Congestion Control and Flow Control

Acksis employs three mechanisms to provide congestion control in a hop-by-hop manner.

1. **Backpressure**. Each node maintains a backlog table of its neighbors' queue lengths, and periodically sends its queue length to its neighbors to update their backlog tables. When a node receives one uncached packet, its queue length is increased by one. After completely forwarding a block to the next hop, its queue length is decreased by the size of the block. The next hop is the neighbor with the shortest queue length.
2. **RPLY block**. A node cannot send more blocks if the current one has not been completely received by the receiver.
3. **FIN constrain**. A node replies to one FIN message in a first come first serve manner if it receives more than one simultaneously.

**Backpressure** is a method to signal the congestion level. The longer a neighbor's queue length is, the more congested the neighbor is. Leveraging of backpressure can alleviate the congestion by directing traffic to the paths with lighter traffic load. Moreover, by picking routing paths

through backpressure, a node can react to the congestion rapidly, which is more appropriate to intermittently connected networks. The **RPLY block** and **FIN constrain** can alleviate the congestion by withholding sending blocks until the congestion is relieved.

In Acksis, the flow control is achieved through **E2ERPLY lock** ensuring that a source cannot inject more blocks into the network until it receives the E2ERPLY message for the block it just sent. There are two reasons leading to the source failing to receive the E2ERPLY before the timeout: network congestion and signal attenuation. Both are valid reasons not to inject new blocks into the network. Waiting for an E2ERPLY message before injecting new blocks can help to relieve the overall network congestion.

## 4.3 Block Size Calculation

In this study, we assume that the network is a pre-planned WSN, in which the number of nodes in the network is known, and node C is out of node A's communication range if both note A and note C are in note B's communication range. The block size in Acksis is determined on the basis of the number of nodes in the network and the minimum storage size of nodes. The size is pre-calculated and shipped with nodes when the network is deployed initially. The block size equals the floor of the minimum storage size of nodes divided by the number of nodes in the network:

$$BlockSize = \left\lfloor \frac{Min(StorageSize(i))}{count(S)} \right\rfloor, \forall i \in S \tag{4.1}$$

where $S$ is the set of nodes in the network, and $i$ is an individual node.

The idea of the block size calculation is based on a worst case consideration in this study for simplicity, but do not exclude other methods in future work. A node can only receive or overhear transmissions from one-hop neighbors, since only two nodes can be in the same communication range in this pre-planned network. Due to E2ERPLY lock, each source can inject one block into the network before it receives an E2ERPLY for the block it sent. The worst case is that a node needs

to cache all blocks from the rest of nodes in the network. If there is not enough space available, the storage of the node will overflow causing unnecessary retransmissions. Equation 1 reduces the probability of overflow by letting each node reserve space for the transmissions from other nodes in the network.

## 4.4 Storage Management

The goal of Acksis storage management is to save storage space for valuable packets without causing unnecessary retransmissions, and control storage size to minimize overflow. Simply saving every packet a node received or overheard may cause the storage overflow resulting in packet loss. Simply dropping the oldest packets can also cause unnecessary retransmissions. Without storage size control, blindly transmitting packets to nodes without sufficient available space causes overflow and wasted energy. All of these negatively affect the network performance.

Acksis solves the storage management problem through three mechanisms:

1. **Block size calculation**. This is explained in Subsection 4.3.

2. **Duplicate checking**. Checking the storage before caching a packet. If the packet has not been cached yet, it is cached. Otherwise, it is dropped.

3. **Buffer cleaning**. Checking the storage after receiving or overhearing an E2ERPLY message. If there are packets in the block indicated by the E2ERPLY message cached in the storage, they are deleted.

Mechanism 1 avoids transmitting packets to nodes without sufficient free storage in a worst case scenario, which is elaborated in detail in the Subsection 4.3. Mechanism 2 saves space by avoiding repeatedly caching packets, which greatly reduces the opportunity of storage overflow. Mechanism 3 deletes the packets that already arrive at the destination from storage in order to save spaces for valuable packets. Together these mechanisms significantly reduce the packet loss caused by storage overflow.

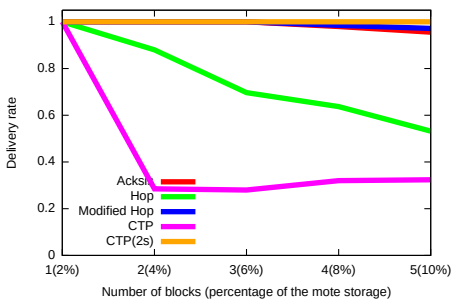
## 4.5 Simulation and Experiments

### 4.5.1 Simulator and Experiment Setup

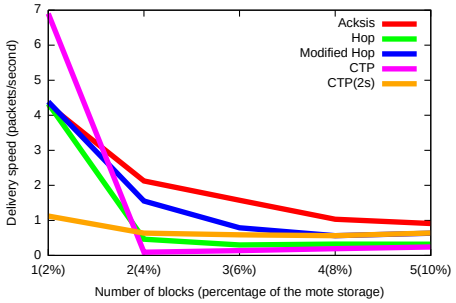
TinyOS [34] is a popular operation system designed for low-power wireless devices such as nodes used in WSNs. TOSSIM [35] is a discrete event simulator for TinyOS sensor networks. TOSSIM models the network in a weighted, directed graph, $G = \{V, A\}$. Weights of $A$ specify the receive signal power between two nodes.

In this study, we implement the Acksis protocol and Hop protocol on TinyOS. The Hop protocol is designed for a regular mesh wireless network where the storage is not extremely limited. So it stores every packet nodes receive or overhear. In order to investigate the performance of acknowledgement-assisted storage management of Acksis, we implement a modified Hop allowing nodes to check their storage before caching packets and store the packets that have not been cached yet. This eliminates the performance improvements by using duplicate checking and allows us to focus on the improvements due to the buffer cleaning of Acksis protocol. We also want to compare Acksis with protocols designed for WSN, so we chose Collection Tree Protocol (CTP) as an example, which is a common WSN communication protocol. All four protocols are tested in TOSSIM. In the following experiment analysis, *Hop* is the original Hop protocol that nodes store everything they receive or overhear without duplicate checking. *Modified Hop* has the duplicate checking, storing packets that are not already cached by the nodes. *CTP* sends sequential packets without interval. *CTP(2s)* has two seconds sending interval between two consequent packets. When comparing Acksis, Hop and Modified Hop, these protocols use the same block size and same beacon timer for nodes to exchange backlog tables with neighbors, in order to focus on the performance of storage management.

We ran the experiments on a $7 \times 7$ grid with the receive signal power over each edge of $-60.0$ dBm which is a typical value for wireless networks. Since node storage is limited and we want to investigate the performance of the acknowledgement-assisted storage management of Acksis, we set the data packet queue size of each node to 50.




*(a) Acksis and Modified Hop have much higher delivery rates than Hop. CTP has the lowest delivery rate. CTP(2s) has a perfect delivery rate.*




*(b) Acksis has higher speeds than Hop and Modified Hop. CTP has the highest delivery speed initially, but declines quickly. CTP(2s) has comparatively low delivery speed.*

Figure 4.3: Delivery rate and delivery speed comparison in a low contention and collision scenario

We use two metrics to evaluate the performance: *delivery rate* and *delivery speed*. Delivery rate is the number of received distinct packets at the destination divided by the total number of packets sent by the sources. Delivery speed is the number of packets sent by sources divided by the time to receive all of them at the destination. If there are packet losses, the time is the sum of the actual time to receive the last received packet and the estimation time to receive the rest of the packets. The higher the loss rate is, the longer the estimation time is. Unlike throughput, delivery speed is not the overall transmission capability of the network, but the speed to deliver a certain amount of data. Since TOSSIM does not consider energy consumption, we will inspect this metric in Chapter 5. Each data point used for plotting is the average value of 100 runs of the experiment, with a 95% confidence interval.

## 4.5.2 Experiment Results and Performance Evaluation

### 4.5.2.1 Single source

In the single source experiment, the protocols are tested in light contention and collisions scenario. In this experiment we set the source to be the top left node of the grid and the destination to be the right bottom node of the grid. The block size of Acksis is one, which is calculated through Equation 4.1. Hop and Modified Hop use the same block size. The source sends out

a flow of packets. The runtime of each run is 30 minutes simulator time. Figure 4.3 shows the delivery rates and delivery speeds of the protocols.

In Figure 4.3 (a), the delivery rate of Acksis and Modified Hop are very close and high, and decrease slowly as the flow length increases. However, the delivery rate of Hop is much lower than the other two, and decreases quickly with the flow length increasing. The reason for the quick decrease in the delivery rate of Hop is that the longer flows need more rounds to get delivered completely. Therefore the probability of retransmission also increases, which leads to the waste of node storage for the duplicate packets. The longer the queue is, the higher the probability of storage overflow, which results in the delivery rate of Hop decreasing quickly. Collection Tree Protocol is designed for light traffic scenarios, and has no end-to-end reliability guarantee. So it can reach 100 percent delivery rate when it has two seconds sending interval. But when it has no sending interval, its delivery rate decreases quickly with the flow length increase.

In Figure 4.3 (b), the delivery speed of Acksis, Hop and Modified Hop decrease with flow length increasing. However Acksis has higher delivery speed than the other two. This is because Acksis deletes useless packets to save space, which keeps the queue length comparatively shorter than Hop and Modified Hop. So the queue time of packets in Acksis is comparatively less than the other two. CTP has good delivery speed initially, but due to its high loss rate the delivery speed decreases quickly. CTP(2s) has a fairly low delivery speed because of the two seconds sending interval.

#### 4.5.2.2 Multiple sources

In the multiple sources experiment, the protocols are tested in a high contention and collision scenario. In this experiment, we increase the number of sources from left to right diagonally. Figure 4.4 shows how the sources increase in this experiment. Initially, only the top left green node transmits, which accounts for 1% of all nodes. Next, the three green nodes on the left of the 3(6%) diagonal start transmitting. The sources continue to increase until all the green nodes in the grid start transmitting, which accounts for 57% of all nodes. All sources send simultaneously. The

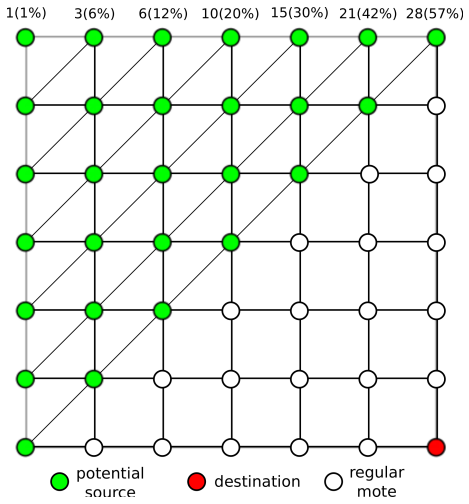
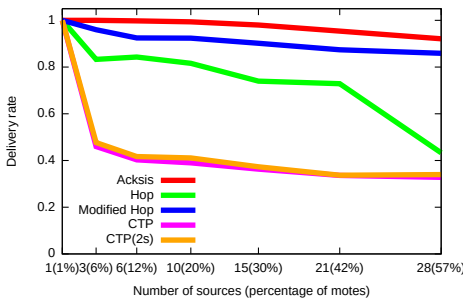
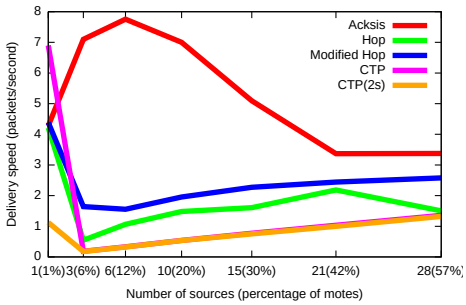



Figure 4.4: Network layout for the experiment with multiple sources




*(a) Acksis has better delivery rates than Hop and Modified Hop. CTP and CTP(2s) have the lowest and very close delivery rates.*

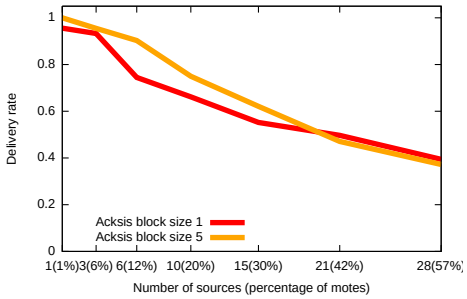



*(b) Acksis has higher delivery speeds than Hop and Modified Hop. CTP has the highest delivery speed initially, but declines quickly. CTP(2s) has the lowest delivery speed all the time.*

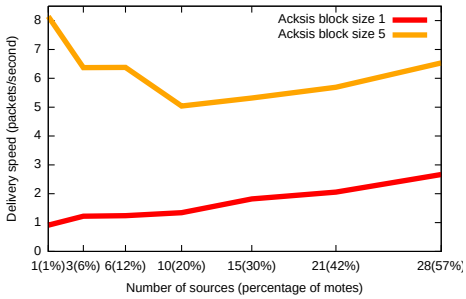
Figure 4.5: Delivery rate and delivery speed comparison in a high contention and collision scenario

runtime of each run is 30 minute simulator time. The block size for Acksis, Hop and Modified is one. Each source sends a single packet. So in this experiment, contention and collision increase as the percentage of sources increase. Figure 4.5 shows the delivery rates and delivery speeds of the protocols under the multiple sources scenario.

In Figure 4.5 (a), the delivery rate of Acksis is higher than Hop and Modified Hop. This is because Acksis has comparatively higher delivery speed than the other two, which is shown in Figure 4.5 (b). The higher the delivery speed, the more packets can be delivered during a certain period. As discussed in the previous experiment, the queue lengths of nodes implementing Acksis




*(a) Acksis with block size 1 has better delivery rates when more than 40% nodes are sources.*




*(b) Acksis with block size 1 has a lower delivery speeds because it takes more rounds to send one flow than Acksis with block size 5.*

Figure 4.6: Delivery rates and delivery speeds of different block sizes

are comparatively shorter than that of the nodes implementing Hop and Modified Hop. As a result, the queue time of packets in Acksis is shorter than Hop and Modified Hop, and the delivery speed of Acksis is higher than the other two protocols. When the number of sources increases, the delivery rates of Acksis and Modified Hop get close, since the delivery speed of the two protocols also gets close. The reason for this change is the flow control mechanism. When there are more nodes injecting packets into the network, the E2ERPLY buffer cleaning will not change the queue length much. Hence, the difference in node queue length in these two protocols is insignificant, so that the delivery speeds of the two protocols are getting closer as the number of sources increase. The delivery rate and delivery speed of Hop decreases quickly when more than 40% of the nodes are sources. In Hop, nodes store every packet received or overheard which causes storage overflows that degrades the performance. Due to a lack of end-to-end reliability and the capability to deal with heavy traffic, both CTP and CTP(2s) have the lowest delivery rates and delivery speeds.

#### 4.5.2.3 Different block sizes

The block size does affect the performance of Acksis. In the different block sizes experiment, Acksis with different block sizes is tested under a high contention and collision scenario. In this experiment, the sources are selected in the same fashion as in multiple sources experiment. But, instead of each source sending a single packet, a flow of five packets needs to be sent by each

source. Acksis with block size 5 sends one flow at one time, while Acksis with block size one needs five rounds to send one flow. The runtime of the experiment is 60 minute simulator time. Figure 4.6 shows the delivery rates and delivery speeds of each condition.

In Figure 4.6 (a), the delivery rates of Acksis with both sizes decrease as the number of sources increase. When more than 40% of nodes are sources, Acksis with block size one has a better delivery rate than Acksis with block size five, since storages overflow happens when the block size is five.

In Figure 4.6 (b), Acksis with block size five has higher delivery speeds than Acksis with block size one. Because when sending the same flow, Acksis with block size five needs fewer rounds than Acksis with block size one.

This experiment exposes the weakness of the block size calculation in Acksis: the network resources are not sufficiently utilized when the traffic load is light. We are starting to address this problem by implementing an optimal block size calculation and will evaluate it in Chapter 5.

## 4.6 Discussions

Acksis can provide a higher delivery rate and delivery speed even in a high contention and collision scenario than other hop-by-hop transport protocols with in-network storage that do not employ sufficient storage management. This is achieved by the storage management mechanisms of Acksis: block size calculation, checking duplicates before caching a packet and removing cached packets that have been confirmed by their destinations.

There are opportunities to improve Acksis. In order to avoid overflow, the block size of Acksis is pre-calculated with respect to the high traffic load scenarios, which leads to storage underutilization in low traffic load situations. Because the small block size needs more rounds to completely deliver a flow, the delivery speed is reduced. We optimize block size decision making to improve the network resource utilization in Chapter 5. This optimization can also improve the scalability of Acksis, and help to ameliorate congestion control by allowing more blocks from a

single source in flight. It is hard to model the intermittent connectivity in TOSSIM. We evaluate the performance of Acksis on a delay-tolerant simulator in Chapter 5.

# Chapter 5

# Dynamic Acknowledgement-assisted Storage Management Transport Protocol

## 5.1 Protocol Overview

*Dynamic Acknowledgement-assisted Storage Management Transport Protocol (Dacksis)* [43] is a significant extension to Acksis [42]. The main purpose of Dacksis is to provide an energy efficient transport protocol for ICDT-WSNs that provide both congestion control and end-to-end reliability. Based on this goal, Dacksis transmits on a block basis, and forgoes the link-layer per-packet acknowledgements to reduce the energy consumption spent on overhead. The main extension of Dacksis over Acksis is that the block size in Dacksis is dynamically decided before every block transmission in order to sufficiently utilize node storage. In addition, Dacksis employs hop-by-hop transport and in-network storage to reduce the number of end-to-end loss recovery. Dacksis also provides node storage management without introducing new overhead, so energy spent on confirmed and duplicate packets is also reduced. In order to provide a quick response to detected congestions, Dacksis applies backpressure to provide hop-by-hop congestion control.

Dacksis employs six control messages:

**RQST:** a message broadcasted by a node that has packets to send to request the backpressure of

its neighbors.

**RRPLY:** a message containing backpressure of a node. It is sent by a RQST receiver back to the RQST sender.

**FIN:** a message containing the sequence number of the packets in a block. It is sent by a block sender to indicate the end of the block.

**RPLY:** a message containing the sequence number of the lost packets in a block. It is sent by a FIN receiver to a FIN sender to provide hop-by-hop reliability.

**E2ERPLY:** a message generated by a sink after all packets in a block have been received by the sink. It is used to confirm packets and clean node storage.

**RFIN:** a message generated by a source node after the source node transmitted a block. It contains the sequence number of the packets in the block. RFIN is used in virtual retransmission [38] to provide end-to-end reliability.

Figure 5.1 gives an overview of Dacksis. Nodes stay in *Idle* state until an event triggers an action. In Figure 5.1, red actions belongs to the congestion control mechanism, yellow actions are in the storage management mechanism, and cyan actions are parts of the reliability mechanism. These mechanisms will be explained in detail in the rest of this section.

## 5.2 Congestion Control

Dacksis provides congestion control through a reactive hop-by-hop mechanism:

**Reactive next hop decision:** a node that has packets to forward broadcasts RQST to get the back-pressure of its neighbors. If there are RRPLYs received by the node, the node picks the neighbor with the shortest queue length as the next hop.

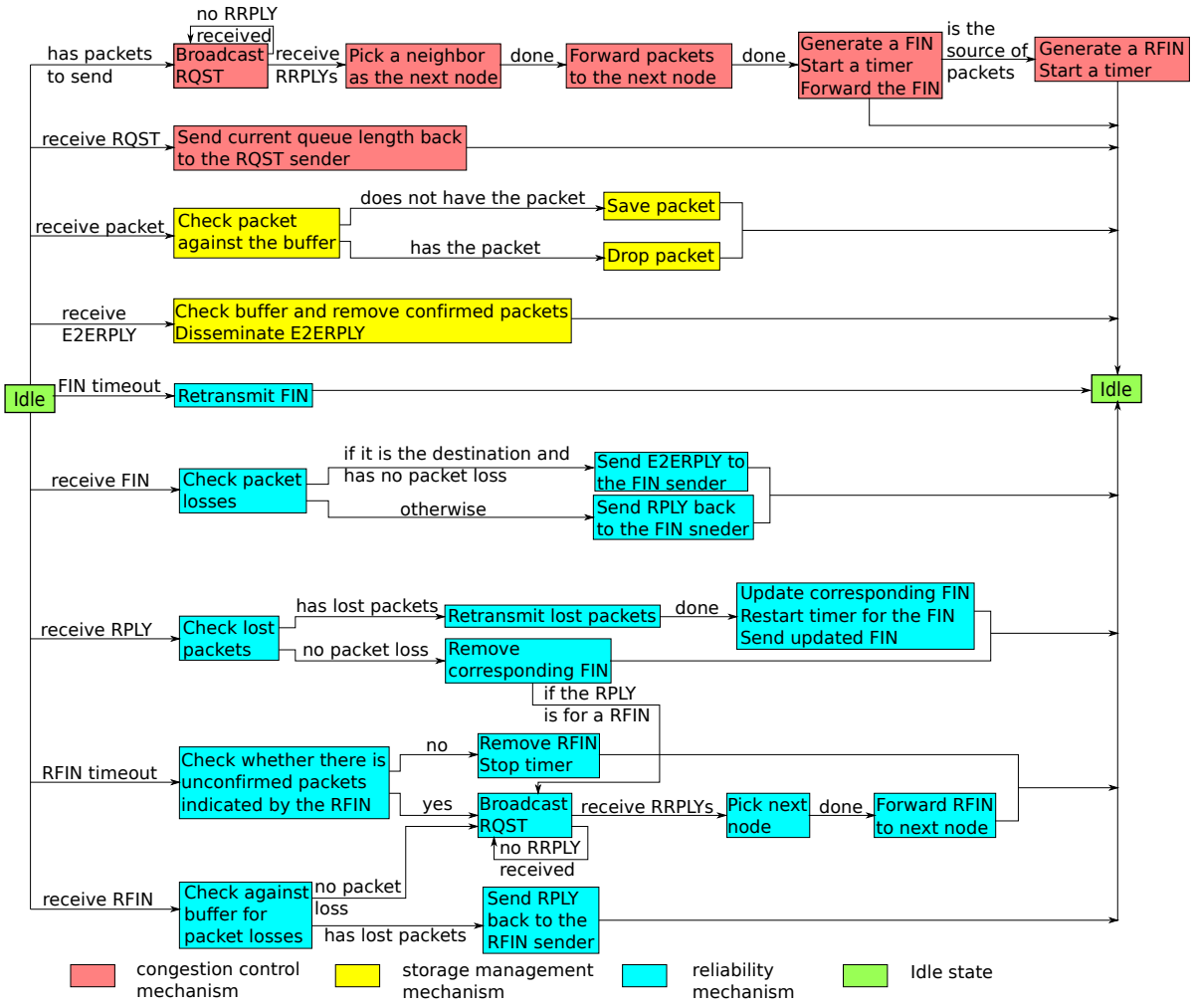



Figure 5.1: Dacksis overview.

Through the reactive next hop selection, Dacksis makes a local decision to forward packets to the direction with less traffic. In order to reduce the energy spent on per-packet acknowledgements, Dacksis transmits packets in blocks. The block size is dynamically decided as discussed in Subsection 5.3. Once the block size is decided, the node forwards a block of packets in burst to the next hop. After the node completes forwarding the block of packets, it generates a FIN with the sequence number of the packets in the block and forwards the FIN to the next hop to indicate the end of a block transmission. Meanwhile, the node starts a timer for the FIN. If the node is the source node of the packets it just sent, it generates a RFIN for these packets and starts a timer for the RFIN.

Unlike Acksis, Hop, and CTP, which use periodic beacons to acquire neighbor information, Dacksis broadcasts RQSTs when needed and collects RRPLYs to get the information. The reactive requests acquire more up to date neighbor information at the cost of an increase in the delay resulting from the neighbor information collection. Additionally, the reactive requests reduce the traffic load and energy consumption from the overhead when the data transmission requests in a network are less frequent. However, it can also increase the traffic load and energy consumption through heavy RQSTs and RRPLYs transmissions when data transmission requests are frequent. The option of using cached neighbor information is a to address this issue. A way to reduce unnecessary transmission for neighbor information is discussed in Chapter 7.

To support the dynamic block size calculation and reactive next hop decision, after a node receives a RQST, it sends a RRPLY with its current queue length and available buffer size back to the RQST sender and starts a timer for the RRPLY. It is possible that a node receives several RQSTs from its neighbors. In order to prevent the buffer overflow caused by multiple senders competing for the same next hop, a node sends a RRPLY to only one RQST sender at a time in a first come first serve manner. Then the node can reply to other RQSTs in the following three cases:

- If the node is the next hop of the RQST sender, the node can reply to other RQSTs after the block transmission between it and the RQST sender is complete.
- If the node is not the next hop of the RQST sender, the node can reply to other RQSTs after it overhears transmission from the RQST sender.
- If the node does not overhear the transmission from the RQST sender, it can reply to other RQSTs after the timer set for the RRPLY expired.

## 5.3 Block Size Calculation

Dacksis employs a **dynamic block size calculation** that a sender calculates the block size through Equation 5.1 after the sender chooses the next hop. Let $B_D$ be the block size, $A$ be the

available buffer size of the next hop, and $W$ be the number of packets to be forwarded by the sender. The block size equals to the number of packets to be sent if the available buffer size of the next hop is larger than the number of packets to be sent. Otherwise, the block size equals to the available buffer size of the next hop.

$$B_D = \begin{cases} A, & A \leq W \\ W, & A > W \end{cases} \tag{5.1}$$

Dynamic decision on the block size is more efficient than the worst case consideration block size [42]. There are three major reasons supporting this argument. First, the worst case consideration block size allows only one block per source node in flight, which increases the waiting time to forward a flow when the number of packets in a flow is larger than the block size. This problem can get worse in intermittently connected scenarios where the future topology of a networks is unpredictable. Second, the worst case consideration block size underutilizes node storage when traffic load is light. Finally, the worst case block size consideration is less energy efficient, since more energy is spent on overhead for the reliability guarantee.

## 5.4 Storage Management

Storage management of Dacksis utilizes two mechanisms:

**Duplication checking:** a node checks the received packet against the buffer first. If the packet is already in its buffer, the node drops the packet; otherwise, the node saves it.

**Buffer cleaning:** a node checks the received E2ERPLY against the buffer. If there are packets in its buffer indicated by the E2ERPLY, remove these packets from the buffer.

When a node receives a packet, it applies the duplication checking to decide whether to save the received packet or not. In this way, Dacksis can shorten the queue length and reduce the energy spent on transmitting duplicate packets.

When a node receives an E2ERPLY, it applies the buffer cleaning first to eliminate confirmed packets. In this way, Dacksis can save node space for more unconfirmed packets and reduce the energy spent on transmitting confirmed packets. After the buffer cleaning, the node disseminates the E2ERPLY if there are new confirmed packets listed in the E2ERPLY.

## 5.5 Reliability

The reliability mechanism of Dacksis is composed of two levels: hop-by-hop and end-to-end. Hop-by-hop reliability is guaranteed by FIN and RPLY, and end-to-end reliability is provided through *virtual retransmission* [38]. Hop-by-hop reliability combined with in-network storage can reduce the number of end-to-end retransmission when using virtual retransmission to provide end-to-end reliability, since lost packets can be retransmitted by intermediate nodes.

When a node receives a FIN, it checks the packets indicated by the FIN against the buffer for packet losses. If there are packets that are not saved in the node's buffer, the sequence number of these packets are included in the RPLY forwarded back to the FIN sender. If the FIN sender does not receive a RPLY before the FIN timer expires, the sender will retransmit the FIN. After the FIN sender receives a RPLY, it checks whether there are sequence numbers of lost packets in the RPLY. If there are no lost packets, the node goes back to idle state. If there are lost packets, the FIN sender retransmits lost packets followed by an updated FIN with the sequence number of the retransmitted packets. This process continues until there is no packet loss. The completion of the process indicates that a complete block of packets has been stored in the RPLY sender.

After the timer for a RFIN expires, the node checks whether there are packets indicated by the RFIN stored in its buffer. If no packet indicated by the RFIN exists in the node's buffer, after removing the RFIN and stopping the timer for the RFIN, the node goes back to idle state. Otherwise, a virtual retransmission is triggered. The node broadcasts a RQST for the RFIN, picks the next node from the RPLY senders and forwards the RFIN to the next node. If there is no RPLY received by the node after a certain period of time, the node will rebroadcast the RQST. After a

node received a RFIN, it will check the RFIN against the buffer to see whether all packets indicated by the RFIN are stored in its buffer. If the node has all packets indicated by the RFIN, it forwards the RFIN to the next node by broadcast a RQST first. If there are some packets indicated by the RFIN not stored in the node's buffer, the node sends a RPLY with the sequence number of these missed packets back to the RFIN sender for retransmission.

## 5.6 Theoretical Analysis

As a protocol for intermittently connected wireless sensor networks, Dacksis retains the beneficial features of Acksis, such as storage management and reliability. However, in order to be more efficient, Dacksis extends the block size calculation mechanism of Acksis. The main differences between Dacksis and Acksis are:

- In Acksis, the block size is fixed and pre-calculated according to Equation 5.2

$$B_A = \left\lfloor \frac{\min_{i \in N}(S_i)}{N} \right\rfloor \tag{5.2}$$

  where $B_A$ is the block size, $N$ is the number of nodes in a network, $i$ is an individual node, and $S_i$ is the storage size of a node $i$ [42]. While in Dacksis, the block size is dynamically calculated per transmission according to Equation 5.1.

- In Acksis, a node cannot transmit a block until the node receives the acknowledgement for the previous block from the destination. While in Dacksis, a node can transmit a block as long as the next hop is available and the previous block was successfully forwarded.

Due to these differences, Dacksis takes better advantage of nodes' storage space than Acksis. In this section, we show that Dacksis can achieve better performance than Acksis. Table 5.1 summarizes the terminology and notation used in the rest of this section.

Table 5.1: Notation and terminology

| Variable | Definition |
|---|---|
| $B$ | The number of packets in a block |
| $C_A$ | The number of blocks per flow of packets in Acksis |
| $C_D$ | The number of blocks per flow of packets in Dacksis |
| $F$ | The number of packets in a flow |
| $H$ | The average number of hops for a packet to be delivered to the destination |
| $I$ | The wait time to receive the acknowledgement from the next hop (the idle time of a sender) |
| $M$ | The expected number of neighbors of a node |
| $N$ | The number of nodes in a network |
| $p_d$ | The probability that one of node's neighbors is the destination |
| $p_m$ | The probability of a node meeting with another node |
| $p_t$ | The probability of successful transmision of a packet to the next hop |
| $S$ | The storage size of a node (measured as the number of packets) |
| $T$ | The time to deliver a block to the next hop |
| $\tau$ | The transmission time of a packet |
| $T_A^F$ | The time to deliver a flow of packets to the destination by using Acksis |
| $T_D^F$ | The time to deliver a flow of packets to the destination by using Dacksis |

Let $B$ be the block size, $\tau$ be the transmission time of a packet, $I$ be the node's idle time, waiting for acknowledgement from the next hop, and $p_t$ be the probability of successfully transmitting packets to the next hop. The time to transmit a block of packets to the next hop at step $n$ is:

$$T_n = \sum_{t=1}^{n} \Big( B\tau \sum_{k=0}^{t-1} (1-p_t)^k + tI \Big) (1-p_t)^{t-1} p_t. \tag{5.3}$$

As $n$ approaches infinity, the expected time to deliver a block to the next hop, $T$, is:

$$T = B\tau p_t \sum_{t=1}^{\infty} (1-p_t)^{t-1} \sum_{k=0}^{t-1} (1-p_t)^k + I p_t \sum_{t=1}^{\infty} t(1-p_t)^{t-1}. \tag{5.4}$$

Equation 5.4 can be further simplified to:

$$T = \frac{B\tau}{2p_t - p_t^2} + \frac{I}{p_t}. \tag{5.5}$$

In the situation without any packet loss, the time to deliver a block of packets to the next hop is $T = B\tau + I$. With the probability of successful transmission decreasing, the time to deliver a block of packets to the next hop increases.

Let $N$ be the number of nodes in a network, one of them is the destination. Nodes move randomly, with the probability $p_m$ to meet with another node. The expected number of neighbors of a node, $M$, is:

$$\begin{aligned} M &= \sum_{i=0}^{N} i \binom{N}{i} p_m^i (1-p_m)^{N-i} \\ &= N p_m. \end{aligned} \tag{5.6}$$

Let $m$ be the number of neighbors of a node, the probability $p_d$ that one of the $m$ nodes is the destination is:

$$p_d = \frac{\binom{N-1}{m-1}}{\binom{N}{m}} = \frac{m}{N} \tag{5.7}$$

Packets are delivered to the destination if the destination is in the sender's neighborhood. According to Equation 5.7, the probability that packets are delivered to the destination successfully at step $k$ is $(1-p_d)^{k-1}p_d$. Therefore, the average number of hops for a packet to be delivered to the destination is:

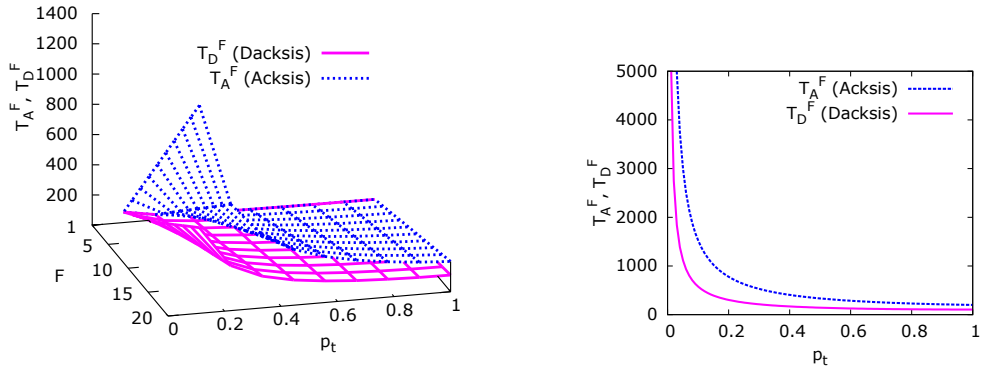
$$\begin{aligned} H &= \sum_{k=1}^{\infty} k(1-p_d)^{k-1}p_d \\ &= \frac{1}{p_d}. \end{aligned} \tag{5.8}$$

Since $H = \frac{1}{p_d}$, and $p_d = \frac{m}{N}$, we can get $H = \frac{1}{p_m}$ when $m = M = Np_m$.

The average time to deliver one block to the destination is $T \cdot H$. Let $F$ be the length of a flow of packets. We assume that every node has the same storage space $S$. The number of blocks per flow in Acksis, $C_A$, and the number of blocks per flow in Dacksis $C_D$ are shown as Equations 5.9 and 5.10.

$$\begin{aligned} C_A &= \frac{F}{\lfloor \frac{S}{N} \rfloor} \\ &= \left\lceil \frac{NF}{S} \right\rceil \end{aligned} \tag{5.9}$$

$$C_D = \begin{cases} 1, & \text{if } S \geq F, \\ \lceil \frac{F}{S} \rceil, & \text{if } S < F. \end{cases} \tag{5.10}$$




(a) Variable flow length

(b) Fixed $F = 20$ packets

Figure 5.2: Plots for $T_A^F$ and $T_D^F$. In both plots, we set $\tau = I = 1$ time unit, $N = 10$, $S = 10$ packets size (each packet has equal size) and $p_m = 0.2$. In (a), we vary $F$ from 1 to 20 packets and $P_t$ from 0 to 1. In (b), we set $F = 20$ packets and vary $P_t$ from 0 to 1.

Let's consider a simple case that there is only one source in the network. The time to deliver a flow of packets with length $F$ to the destination by using Acksis is $T_A^F$ and the time by using Dacksis is $T_D^F$:

$$\begin{aligned} T_A^F &= \left\lceil \frac{NF}{S} \right\rceil \left( \frac{\frac{S}{N}\tau}{2p_t - p_t^2} + \frac{I}{p_t} \right) \cdot H \\ &= \left( \frac{F\tau}{2p_t - p_t^2} + \left\lceil \frac{NF}{S} \right\rceil \cdot \frac{I}{p_t} \right) \cdot H, \end{aligned} \tag{5.11}$$

$$T_D^F = \begin{cases} \left( \dfrac{F\tau}{2p_t - p_t^2} + \dfrac{I}{p_t} \right) \cdot H, & \text{if } S \geq F, \\ \left( \dfrac{F\tau}{2p_t - p_t^2} + \left\lceil \dfrac{F}{S} \right\rceil \cdot \dfrac{I}{p_t} \right) \cdot H, & \text{if } S < F. \end{cases} \tag{5.12}$$

By comparing Equation 5.11 with 5.12, we confirm that Acksis consumes more time to deliver a flow of packets than Dacksis. More specifically, Acksis needs more idle time than Dacksis to deliver a flow of packets to the destination. We plot Equations 5.11 and 5.12 with fixed $N, S, \tau, I$ and $p_m$ and varying $F$ and $p_t$ (Figure 5.2), in order to inspect the performance of Acksis and Dacksis under scenarios with different flow length and probability to successfully deliver packets

Table 5.2: Attribute comparison of competed protocols in Section 5.7

| Protocols | Block Size | Storage Management | End-to-end Reliability |
|---|---|---|---|
| **Hop** | Static | No | Yes |
| **Modified Hop** | Dynamic | No | Yes |
| **CTP** | N/A | N/A | No |
| **Acksis** | Static | Yes | Yes |
| **Dacksis** | Dynamic | Yes | Yes |

to the next hop. We will use a simulation study to analyze the performance under more complicated scenarios in the next section.

## 5.7 Simulation Study

### 5.7.1 Simulator and Experiment Setup

*Contiki* [11] and *Cooja* [50] were used to study the energy consumption and to simulate intermittently connected scenarios. Contiki is an operating system designed for small low-power devices, such as sensor nodes. Cooja is a tool simulating networks of devices running Contiki operating system. Cooja provides features that many real testbeds do not, such as energy measurement, node transmission range control and mobility support. *Bonnmotion* [1] was used to generate node motion patterns that can be loaded into Cooja to simulate intermittently connected scenarios.

We compare Dacksis with Hop [38], Modified Hop, CTP [19], and Acksis [42]. All these protocols are implemented on Contiki. Table 5.2 lists out the key attributes of these protocols. Hop refers to the original protocol that transmits packets in fixed block size without storage management. Hop was designed for wireless mesh networks and it does not provide a block size calculation that fits for wireless sensor networks. We set the block size of Hop the same as the block size of Acksis. Acksis calculates the block size as Equation 5.2. The comparison among

Dacksis, Acksis and Hop highlights the performance impact of the dynamic block size mechanism. Modified Hop, introduced here to facilitate more fair comparison, is our version of Hop protocol that utilizes the same dynamic block size calculation as Dacksis but has no storage management mechanism. The comparison between Dacksis and Modified Hop shows the effect from the storage management to the system performance under different traffic loads. CTP is a well known forwarding protocol for WSNs, and does not provide end-to-end reliability. By comparing with CTP, we study the effect of end-to-end reliability to the system performance under different scenarios.

In this study, we evaluate the performance of these protocols through four metrics: *packet delivery rate*, *packet latency*, *number of hops per packet* and *energy consumption*. Packet delivery rate is the ratio between the number of distinct packets delivered to the sink and the total number of packets sent by all source nodes within a certain period of time. Packet latency is the measure of time elapsed between a packet being sent by the source node and it is arrival at the sink. Number of hops per packet is the count of nodes a packet goes through before it arrives at the sink. Energy consumption is measured through the average transmission time and receiving time spent on delivering one packet to the sink in a network with $N$ nodes (Equation 5.13). Let $E$ be energy consumption, $T_i$ be the total transmission time during the experiment time of a node $i$, $R_i$ be the total receiving time during the experiment time of a node $i$, and $count(pkt)$ be the number of distinct packets arrived at the sinks,

$$E = \frac{\sum_{i=1}^{N} T_i + \sum_{i=1}^{N} R_i}{count(pkt)}. \tag{5.13}$$

We designed four sets of simulation experiments using Cooja: light traffic scenarios, heavy traffic scenarios, varying node densities and varying transmission ranges. The network area is 100 $\times$ 100 meters for all experiments. Nodes move in the network area with the *Random Waypoint* model and 4 m/s speed. Nodes do not run out of energy or fail during the experiments, and have reliable communication if in transmission range. The buffer size of each node is set to 10 packets.

Every source sends a flow of packets in burst. All sources in the network transmit simultaneously. The experiment time of each run is 30 minutes, and the data used in plots are the average values out of 50 runs, with a 95% confidence interval.

### 5.7.2 Experiment Results and Performance Evaluation

#### 5.7.2.1 Light Traffic

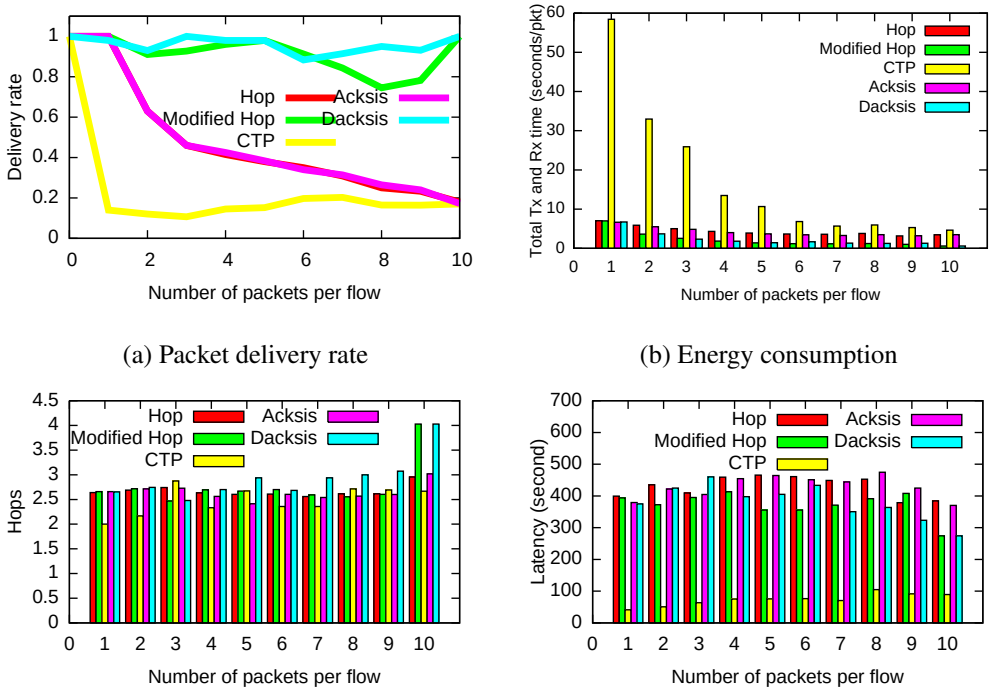
This experiment tests the protocols under light traffic scenarios. In the experiment, there are 10 nodes in the network area, one of the nodes is the source node, and one of the rest nodes is the sink. According to Equation 5.2, the block size of Acksis and Hop is one packet. The source node transmits a flow of packets, and the flow length varies from one to 10 packets. The transmission range of each node is set to 10 meters, so that the network is sparse and intermittently connected. Figure 5.3 shows the results of this experiment.

Figure 5.3(a) shows the delivery rate of these protocols. Dacksis and Modified Hop have higher delivery rates than Acksis and Hop, due to the different block size calculations with these protocols. Acksis and Hop utilize fixed block size, while Dacksis and Modified Hop employ dynamic block size. Let $B_{Fix}$ be the block size of Acksis and Hop, $B_{Dynamic}$ be the block size of Dacksis and Modified Hop. Acksis and Hop need $C^i_{Fix}$ blocks to deliver a flow $i$. Dacksis and Modified Hop need $C^i_{Dynamic}$ blocks to deliver a flow $i$. $F_i$ be the length of a flow $i$. Hence,

$$C^i_{Fix} = \frac{F_i}{B_{Fix}}, \tag{5.14}$$

$$C^i_{Dynamic} = \frac{F_i}{B_{Dynamic}}. \tag{5.15}$$

Since $B_{Fix} = 1$ in this study, $B_{Fix} \leq B_{Dynamic}$, so that $C^i_{Fix} \geq C^i_{Dynamic}$. Moreover, different from Acksis and Hop, Dacksis and Modified Hop allow multiple blocks in flight per source node.




(a) Packet delivery rate

(b) Energy consumption

(c) Number of hops per packet

(d) Packet latency

Figure 5.3: Ten nodes in a 100 $\times$ 100 m area with transmission range 10 m, one of the nodes is the source node, and one of the rest nodes is the sink, the source node transmits a flow of packets, and the flow length varies from one to 10 packets.

As a result, Dacksis and Modified Hop deliver more packets than Acksis and Hop within a certain period of time. Dacksis has a higher delivery rate than Modified Hop. This is because of the storage management mechanism with Dacksis, which shortens the queue length by eliminating duplicate packets and confirmed packets. Therefore, Dacksis can deliver more distinct packets than Modified Hop within a certain period of time. Acksis and Hop have similar delivery rates since they use the same fixed block size. The fixed block size allows only one block in flight per source node, so Acksis and Hop need longer time to deliver a flow. CTP has the lowest delivery rate since it does not provide end-to-end reliability.

Figure 5.3(b) shows the energy consumption of these protocols. Dacksis and Modified Hop have the lowest energy consumptions, followed by Acksis and Hop. CTP has the highest energy consumption. Dacksis and Modified Hop need less number of rounds to deliver a flow than Acksis and Hop, as a result, Dacksis and Modified Hop consume less energy on overhead. Due to CTP's

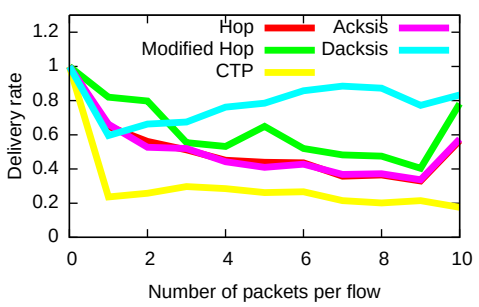
low deliver rate, its energy consumption is higher than other protocols.

CTP has similar number of hops per packet as the other protocols (Figure 5.3(c)), but has a much lower packet latency than the others (Figure 5.3(d)). This is because CTP forwards packets based on the link state that directs packets along the best path to the sink if the path exists, while other protocols forward packets based on the backpressure that alleviates the network congestion but does not guarantee the shortest route to the sink.

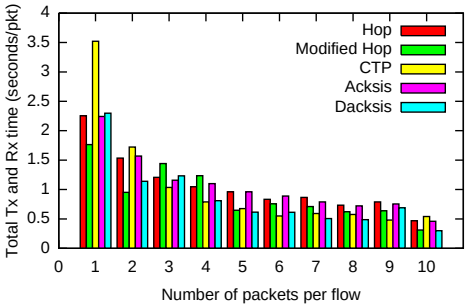
#### 5.7.2.2 Heavy Traffic

This experiment investigates the performance of these protocols under heavy traffic scenarios. In this experiment, there are 10 nodes in the network area, one node is the sink, and the rest are the source nodes. According to Equation 5.2, the block size of Acksis and Hop is one packet. Each source node transmits one flow, and the flow length varies from one to 10 packets. All source nodes transmit simultaneously. The transmission range of each node is set to 10 meter to simulate a sparse and intermittently connected environment. Figure 5.4 shows the results of this experiment.

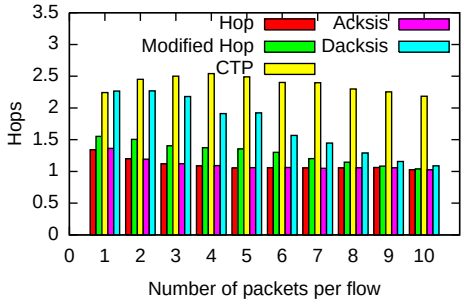
Figure 5.4(a) shows that Dacksis and Modified Hop have the comparatively higher delivery rates, followed by Acksis and Hop, and CTP has the lowest delivery rate. This is because Dacksis and Modified Hop adopt the dynamic block size calculation, Acksis and Hop employee the fixed block size calculation, and CTP does not provide end-to-end reliability. The detail discussion is given in Subsection 5.7.2.1. With the flow length increasing, the delivery rate of Dacksis increases. This is due to Dacksis relying on backpressure to direct packets in order to alleviate network congestion. Backpressure can route packets more efficiently when the traffic in the network is enough. Modified Hop has a higher delivery rate than Dacksis initially, but its delivery rate shrinks compared to Dacksis's when the flow length is larger. The reason is Modified Hop does not provide storage management. When the traffic load is heavy, the lack of storage management leads Modified Hop to transmit more duplicate and confirmed packets, which in turn decreases the number of distinct packets delivered at the sink within a certain period of time.



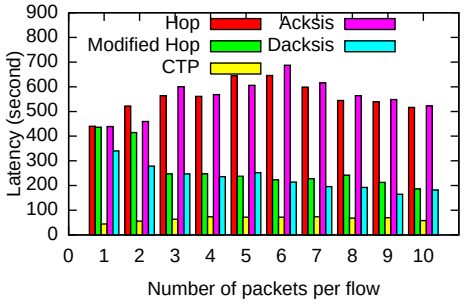

(a) Packet delivery rate




(b) Energy consumption




(c) Number of hops per packet




(d) Packet latency

Figure 5.4: Ten nodes in a 100 $\times$ 100 m area with transmission range 10 m, one node is the sink, the rest are the source nodes, each source node transmits one flow, the flow length varies from one to 10 packets, and all source nodes transmit simultaneously.

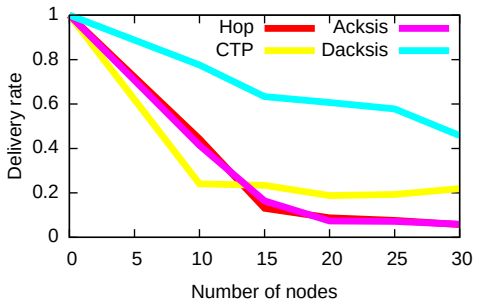
Similar to the light traffic scenarios, Dacksis has lower energy consumption (Figure 5.4(b)). This is because Dacksis utilizes the dynamic block size calculation, and consumes less energy on transmission duplicate and confirmed packets. In Figures 5.4(c) and (d), the number of hops and packet latency of Dacksis decrease when the traffic is getting heavier. This is because there is enough traffic in the network for backpressure to forward packets along better paths. However, the backpressure does not help significantly for Acksis and Hop in the packet delivery rate, number of hops and packet latency when the flow length is large. Because the fixed block size mechanism leads these two protocols underutilize node storage. As shown in Figure 5.4(c), Dacksis has a higher number of hops than Modified Hop, since its storage management lets nodes have more available storage spaces to forward packets. But Acksis does not have a higher number of hops than Hop, because of its static block size calculation allowing only one block in flight per source node. In Figure 5.4(d), Dacksis and Modified Hop have a lower packet latency than Acksis and

Hop, because the dynamic block size calculation reduces the packet queuing time.

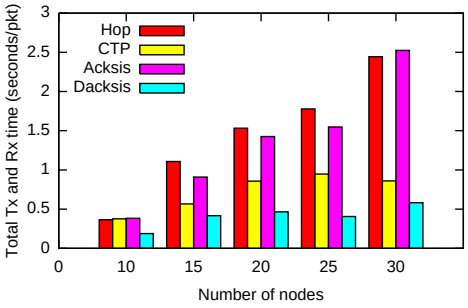
#### 5.7.2.3 Node Density

This experiment varies the node densities in the network area to evaluate protocols under mostly disconnected and mostly connected scenarios. The number of nodes in the network area changes from 10 to 30 nodes, and each node's transmission range is set to 10 meters. One of these nodes is the sink, all the other nodes are source nodes. Each source node transmits a flow of packets, and the flow length is 10 packets. All source nodes transmit simultaneously. Since varying the buffer size of nodes is unlikely to occur in practice, the buffer size of nodes is unchanged in this experiment. As a result, the block size calculation of Acksis is not applicable when the number of nodes are more than 10. In this experiment, we set the block size of Acksis and Hop to one packet and independent of the node density. In this and the next set of experiments, we have not evaluated the performance of Modified Hop, since the main purpose of these two sets of experiments is to inspect the impact from the network connectivity instead of traffic load. Figure 5.5 shows the results of this experiment.

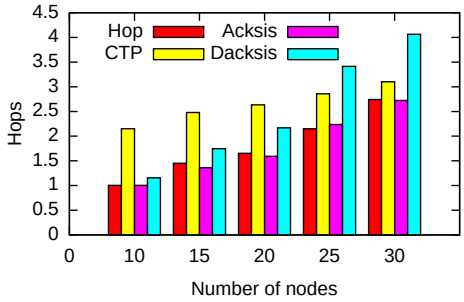
Figure 5.5(a) shows the delivery rates of these protocols: Dacksis still has the highest delivery rate, but the delivery rate decreases while increasing the node density. This is because the number of packets need to be delivered grows when the node density increased. As a result, within a certain period of time, the packets waiting to be delivered increases. Due to the dynamic block size calculation and storage management, Dacksis still has the lower energy consumption than Acksis and Hop regardless of the node density (Figure 5.5(b)). The number of hops and packet latency increase with the node density growing for all protocols (Figures 5.5(c), (d)). Since there are more intermediate nodes between a source and the sink when the network is dense, delivering packets to the sink takes more time.



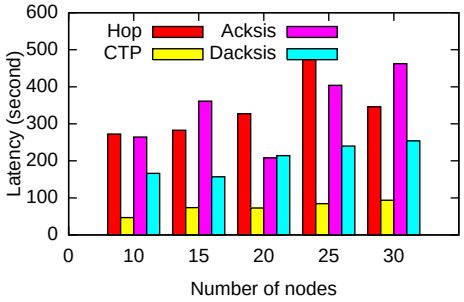

(a) Packet delivery rate




(b) Energy consumption




(c) Number of hops per packet



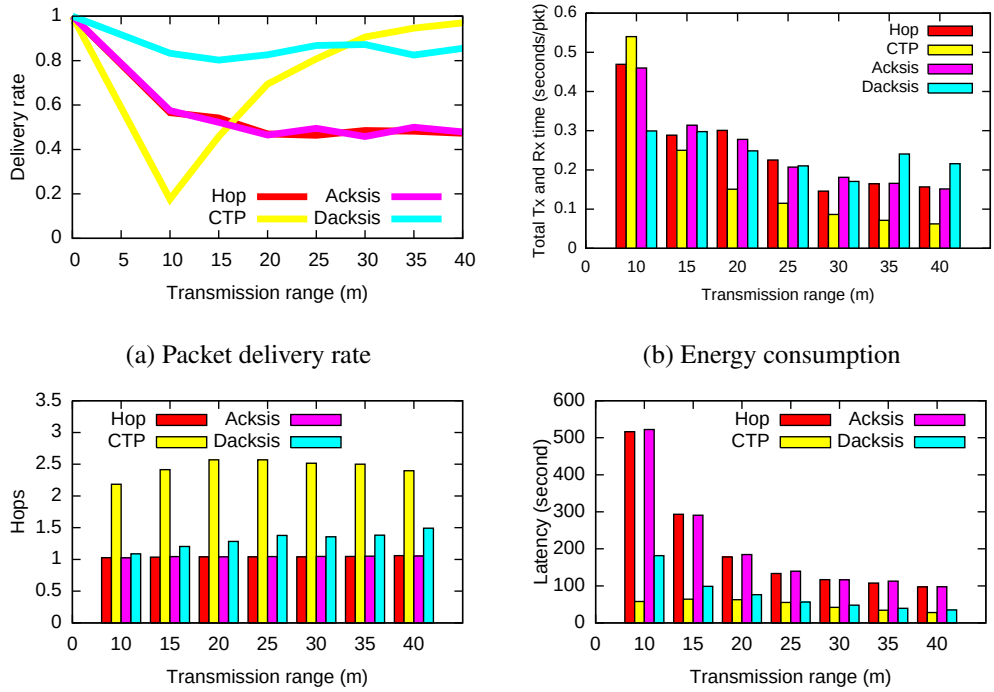

(d) Packet latency

Figure 5.5: The number of nodes in the network area changes from 10 to 30, nodes' transmission range is 10 m, one of these nodes is the sink, all the other nodes are source nodes, each source node transmits a flow of packets, the flow length is 10 packets, and all source nodes transmit simultaneously.

#### 5.7.2.4 Transmission Range

This experiment varies the nodes' transmission ranges to test the performance of these protocols under intermittently connected scenarios and connected scenarios. There are 10 nodes in the network area in this experiments, the transmission range of every node varies from 10 to 40 meters. The block size of Acksis and Hop is one packet according to Equation 5.2. One of these nodes is the sink, the rest are source nodes. Each source node transmits a flow of packets, and the flow length is 10 packets. All source nodes transmit simultaneously. Figure 5.6 shows the results of this experiment.

Figure 5.6(a) shows that CTP delivery rate increases with the node transmission range increasing. When the node transmission range is larger than 30 meters, CTP has a higher delivery




(a) Packet delivery rate

(b) Energy consumption

(c) Number of hops per packet

(d) Packet latency

Figure 5.6: Ten nodes in the $100 \times 100$ m area, the transmission range of every node varies from 10 to 40 m, one of these nodes is the sink, the rest are source nodes, each source node transmits a flow of packets, the flow length is 10 packets, and all source nodes transmit simultaneously.

rate than Dacksis. Since CTP was designed for connected networks, it forwards packets based on the link state and does not provide end-to-end reliability. As a result, when the transmission range of nodes is large enough to form a connected network, CTP can take advantage of its link state forwarding method to deliver packets through the best path, and the impact from lack of end-to-end reliability is minimized. While the delivery rate of Dacksis, Acksis and Hop does not vary much with the node transmission range change. Dacksis, Acksis and Hop forward packets based on backpressure that can be affected by the traffic load, but changes on the transmission range of nodes have no significant impacts on the traffic load. The energy consumption of CTP also decreases with the transmission range increasing, due to the increasing delivery rate of CTP. As with the delivery rate, the number of hops of Dacksis, Acksis and Hop does not fluctuate much with the transmission range increasing (Figure 5.6(c)), since the forwarding method of these protocols, backpressure, is not affected by the transmission range. However, the packet latency of Dacksis,

Table 5.3: Contributions of protocol mechanisms to the performance.

| **Performance Metrics** | **Storage Management** | **Dynamic Block Size** | **End-to-end Reliability** |
|---|---|---|---|
| **Delivery rate** | Increase | Increase | Increase (intermittently connected) |
| **Energy consumption** | Reduce | No impact | Increase (connected) |
| **Number of hops** | Increase | No impact | No impact |
| **Packet latency** | No impact | Reduce | No impact |

Acksis and Hop decreases (Figure 5.6 (d)) when the transmission range is large. This is because the time spent on waiting for connection is reduced when the transmission range increases.

## 5.8 Contributions of Protocol Mechanisms

Table 5.3 lists out the contributions of the protocol mechanisms to the performance of the protocols. Storage management increases the delivery rate and the number of hops per packet, and reduces the energy consumption. Storage management shortens the queue length of a node by eliminating duplicate and confirmed packets. As a result, the queuing time of a packet and energy spent on transmissions of duplicate/confirmed packets are reduced. Furthermore, the nodes can have more available buffer space to accept packets for routing. Therefore, Dacksis has on average higher delivery rate, lower energy consumption even thought it exhibits larger number of hops per packet than Modified Hop.

The dynamic block size calculation increases the delivery rate, and reduces the energy consumption and packet latency. Since the dynamic block size calculation more efficiently utilizes the available buffer space of nodes, a source node can inject more than one block into the network without waiting for the confirmation of the previous block. Therefore, packets can be delivered to the sink more quickly. As a result, Dacksis has on average higher delivery rate and lower packet

latency than Acksis and Hop.

End-to-end reliability increases the delivery rate but also consumes more energy. This is because end-to-end reliability spends energy in retransmission and confirmation. To provide a high delivery rate, retransmission and confirmation are necessary in intermittently connected scenarios, but are less critical in connected scenarios. Hence, we observed Dacksis has higher delivery rate under intermittently connected scenarios and higher energy consumption under connected scenarios than CTP.

## 5.9 Discussions

Dacksis supplies end-to-end reliability, congestion control and storage management. It provides a higher packet delivery rate and lower energy consumption independent of the traffic situations and node density than other hop-by-hop transport protocols adopting in-network storage and blocks as the transmission unit. This is achieved by the storage management and the dynamic block size calculation of Dacksis. The storage management saves node storage for unconfirmed packets and reduces energy spent on transmission duplicate and confirmed packets. The dynamic block size calculation efficiently utilize the free storage space of nodes to deliver packets to the sink more quickly, which is also the main improvement of Dacksis over Acksis.

There exist opportunities to improve Dacksis. Dacksis utilizes backpressure to alleviate congestion, which may introduce long propagation delay because of tossing packets back and forth. This problem can be addressed by using the delivery probability coupled with the the available storage space of nodes. The delivery probability measures the closeness of a node to the destination. Through delivery probability and the available storage space of nodes, traffic can be directed toward the destination on the paths with less congestion. As a result, the propagation delay of data packets can be reduced. This method is discussed in Chapter 6.

# Chapter 6

# Reactive Store-and-Forward Protocol

Cross-layer design, the joint design of networking layers, can improve the energy efficiency of wireless sensor network. In intermittently connection scenarios, reactive mechanism is easier to acquire and maintain routing information than proactive mechanism. Hop-by-hop transport can react to the detected network problems more quickly, and in-network storage can reduce the number of end-to-end retransmission when packet loss happens.

The *reactive store-and-forward protocol (ReSaF)* [40] is a cross-layer protocol that discover route by a reactive conditional-forwarding routing algorithm, and provides end-to-end reliability, congestion control and storage management through hop-by-hop transport, in-network storage. It can provide high delivery rate in an energy efficient way.

## 6.1 Protocol Overview

The basic idea of the reactive store-and-forward protocol (ReSaF) is that a node broadcasts requests when it has packets to forward, the node then picks the next hop from the replies of neighbors. If there is no appropriate next hop, the node backs off and rebroadcasts until an appropriate next node (a node with a higher delivery probability, more available buffer space and energy than the sender) becomes available. If a node does not receive confirmations for the packets it generated and sent after a certain period of time, a *virtual retransmission* will be triggered, as explained in

Section 6.5.

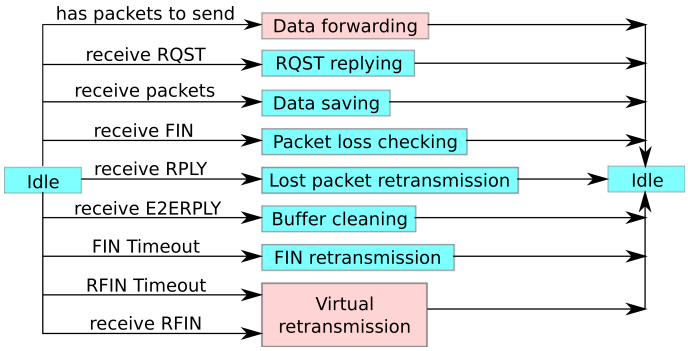



Figure 6.1: ReSaF state diagram.

Figure 6.1 gives an overview of the ReSaF protocol. A node is in the *idle* state until there is an event triggers the node to transit to another state. After finishing the tasks in the new state, the node goes back to the *idle* state. ReSaF utilizes six types of control messages described in Table 6.1.

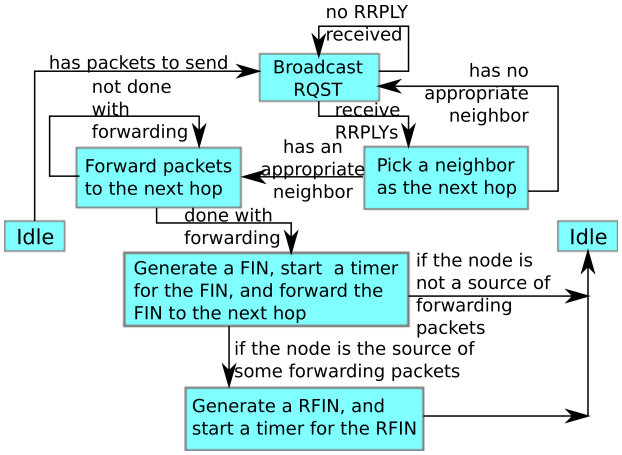



Figure 6.2: Data forwarding

## 6.2 Data Forwarding

When a node has packets to send, it enters into the *data forwarding* state detailed in Figure 6.2. In this state, the node broadcasts a RQST, and waits for RRPLYs. If the node does no receive any RRPLYs within a certain amount time, the node will rebroadcast the RQST later. If the node

Table 6.1: Control Messages of ReSaF

| Control Msg | Description |
|---|---|
| *RQST* | Broadcasted when a node has packets or RFIN to forward. It contains the delivery probability of the node. |
| *RRPLY* | A reply message to a Request. It contains the updated delivery probability, available energy and buffer, and queue length. It is sent back from the Request receiver to the Request sender. |
| *FIN* | Sent after a node finishes forwarding a block of packets to indicate the end of block sending. It contains the sequence number of the packets in the block the node just sent. |
| *RPLY* | After a node receives a FIN, it will check the FIN against its buffer for packet loss. If there are lost packets, the sequence number of the lost packets will be recorded in a RPLY. The node then sends the RPLY back to the sender of the FIN. |
| *E2ERPLY* | When a block reaches at the destination, and there is no packet loss in the block, the destination will disseminate an E2ERPLY to confirm the packets in the block. |
| *RFIN* | This control message is used for virtual retransmission. After a source forwards a block of packets, a RFIN will be generated for the block. If there is a block whose packets are not confirmed totally before a timeout, the RFIN for that block will be forwarded for loss recovery. |

receives RRPLYs, it picks an appropriate neighbor as the next hop according to the Algorithm 6 described in Subsection 6.3. If there is no appropriate neighbor, the node will rebroadcast RQST later. If there is an appropriate neighbor, the node forwards a block of packets to the neighbor. The block size selection is explained in Subsection 6.4. When finished forwarding the block, the node generates a FIN, starts a timer for the FIN, and forwards the FIN to the next hop. If the node is the source of the packets in this block, it creates a RFIN for those packets generated by it, and starts a timer for the RFIN.

## 6.3 Conditional-forwarding Routing Algorithm and Congestion Control

A node enters into the *RQST replying* state, after it receives a RQST. The goal of this state is to send the *routing information* of a node back to a RQST sender. The routing information of a node includes: *Available energy* (the amount of available energy), *Queue length* (the number of packets waiting to be forwarded), *Available buffer* (the size of the available buffer space), and *Delivery probability* (the probability of successful delivery of packets to the sink).

Initially, all nodes except the sink have zero delivery probability. The sink has the delivery probability, one. If a sink node receives a RQST, it sends a RRPLY back to the RQST sender with delivery probability, one. If a regular node receives a RQST, it updates its delivery probability according to Equation 6.1, where $DP_i$ is the delivery probability of the RQST sender, $DP_j$ is the delivery probability of the node, and $DP_j^{'}$ is the node's updated delivery probability. Then the regular node sends a RRPLY back to the RQST sender with the updated delivery probability $DP_j^{'}$ and the remaining routing information.

$$DP_j^{'} = \frac{DP_i + DP_j}{2} \tag{6.1}$$

In order to avoid buffer overflow caused by multiple senders competing for the same next hop, a node can only reply to one RQST at a time. After a node sends a RRPLY to a RQST sender, the node cannot reply to other RQSTs until one of the three conditions is true:

- The block transmission between the node and the RQST sender completes, if the node is the next hop of the RQST sender.
- The node overhears packet transmission from the RQST sender, if the node is not the next hop of the RQST sender.
- A timeout set for the RRPLY sent to the RQST sender expires, if no packets transmission from the RQST sender is overheard by the node.

**Algorithm 6** Conditional-forwarding Routing Algorithm

```
1: BS: best score, S: score, DP: delivery probability
2: QL: queue length, AE: available energy, NH: next hop
3: while node_i has packets to forward do
4:    node_i broadcasts RQST
5:    if node_i receives RRPLYs then
6:       BS = AE_i/QL_i
7:       NH = NULL
8:       for all neighbor_j in the received RRPLYs do
9:          S_j = AE_j/QL_j
10:         if DP_j > DP_i and S_j > BS then
11:            NH = neighbor_j
12:            BS = S_j
13:         end if
14:      end for
15:      if NH ≠ NULL then
16:         DP_i = DP_NH
17:         forward packets to neighbor_j
18:      else
19:         rebroadcast RQST later
20:      end if
21:   else
22:      rebroadcast RQST later
23:   end if
24: end while
```

After a RQST sender receives RRPLYs from its neighbors, it picks the next hop from these neighbors according to the conditional-forwarding routing algorithm (see Algorithm 6). This algorithm employs two routing metrics: delivery probability $DP$ and a routing score $S$ defined as the ratio between the available energy and the current queue length of a node. A larger value of $S$ indicates more available energy and a shorter queue. The next hop is chosen to be the one in the set of nodes with higher delivery than the current node and the best routing score in the set. The conditional-forwarding directs packets on a path with higher delivery probability, less congestion and more available energy.

## 6.4 Block Size Calculation

In ReSaF, per-packet link-layer acknowledgement is disabled, data packets are grouped in to blocks to forward in order to reduce energy consumption. The size of a block is decided dynamically according to Equation 6.2 after a node decides the next hop. Let the available buffer size of the next hop be $AS$, the number of packets to deliver be $PS$, and the block size be $BS$. The block size equals the available buffer size of the next hop if there is not enough space for all packets to be sent, otherwise, the block size equals to the number of packets to be sent.

$$BS = \begin{cases} AS, & AS \leq PS \\ PS, & AS > PS \end{cases} \tag{6.2}$$

## 6.5 Reliability and Storage Management

ReSaF guarantees hop-by-hop reliability through *FIN* and *RPLY*, and provides end-to-end reliability through virtual retransmission. ReSaF adopts *duplicate checking* in the *data saving* state and *packet elimination* in the *buffer cleaning* state to manage in-node storage.

When a node receives data packets, it enters into the *data saving* state. In this state, the node executes the first part of storage management, *duplicate checking* before saving packets. If the packets that are already in the buffer, the node drops the received duplicate packets.

After a node receives a FIN, it enters into the *packet loss checking* state. In this state, a node checks its buffer to see whether all packets indicated by the FIN are in its buffer. If there are lost packets, the node sends a RPLY with the sequence numbers of the lost packets back to the sender of the FIN. If there is no packet loss and the node is the sink, the node sends an E2ERPLY back to the FIN sender. If there is no packet loss and the node is not the sink, the node sends a RPLY back to the sender of the FIN without any sequence number indicating no packet loss.

If a node receives a RPLY, it enters into the *lost packet retransmission* state. In this state, the node starts hop-by-hop retransmission by checking packet loss first. If there is no packet loss,

the node removes the corresponding FIN and stops the timer for the FIN. If this retransmission process is for a RFIN, the node forwards the RFIN to the next hop. Otherwise, the node goes back to the idle state. If there are sequence numbers in the RPLY, the node retransmits the lost packets back to the sender of the RPLY, and updates the sequence numbers of the corresponding FIN to the retransmitted packets' sequence numbers. Then, the node transmits the updated FIN to the RPLY sender after finishing retransmitting lost packets, and reset the timer for the FIN. When the timer of a FIN expires, the node retransmits the FIN.

Once a node receives an E2ERPLY, it enters into the *buffer cleaning* state. In this state, the node executes another part of storage management, *packet elimination*, by removing the confirmed packets indicated by the E2ERPLY from the buffer. If there are packets being removed, the node disseminates the E2ERPLY. Otherwise, the node goes back to the idle state.

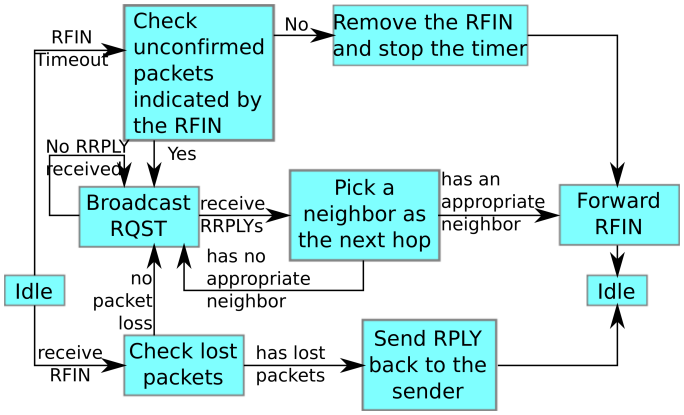



Figure 6.3: Virtual retransmission

When a timer of a RFIN expires or a node receives a RFIN, the node enters into the *virtual retransmission* state. Figure 6.3 shows the virtual retransmission process. When a timer of a RFIN is up, the node checks whether all packets indicated by the RFIN are confirmed. If all packets are confirmed, the node removes the RFIN and stops the timer for the RFIN. Otherwise, the node forwards the RFIN with unconfirmed packets' sequence numbers to the next hop. To forward a RFIN, a node first broadcasts RQST. If there is no RRPLY received, the node will rebroadcast later. If there are RRPLYs received, the node picks the next hop according to Algorithm 6. If there is an appropriate next hop, the node forwards the RFIN to the next hop. Otherwise, the node will

rebroadcast later. When a node receives a RFIN, it checks the buffer to see whether all packets indicated by the RFIN are saved. If the node has all packets, it forwards the RFIN to the next hop. If there are missed packets, the node sends a RPLY with sequence numbers of the missed packets back to the RFIN sender.

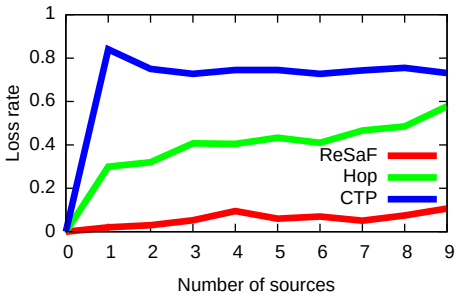
## 6.6 Simulation and Experiments

### 6.6.1 Simulator and Experiment Setup

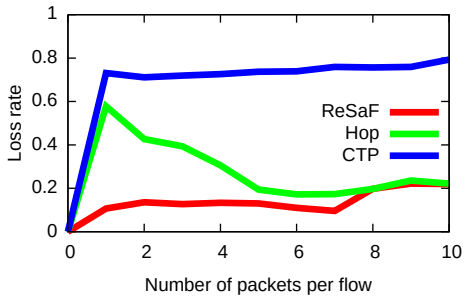
ReSaF has been implemented on Contiki [11] and evaluated on Cooja [50]. Bonnmotion [1] is used to generate intermittently connected scenarios. We compare ReSaF with Hop and Collect Tree Protocol (CTP). Since Hop does not provide storage management, in order to investigate the performance of the conditional-forwarding routing algorithm and backpressure routing, we modify Hop to include the same storage management mechanisms as ReSaF that does not cache duplicate packets and eliminates confirmed packets. CTP is a common WSN protocol for connected scenarios. Comparing with CTP allows us to study the tradeoff between a protocol for intermittently connected WSNs and a protocol for connected WSNs.

We set up three experiments under light traffic scenarios, heavy traffic scenarios and different node densities. Nodes have the transmission range 10 meters and speed 4 m/s. Nodes move according to the *random waypoint* model. The buffer space of a node is set to 10 packets. The network area is $100 \times 100$ m. The simulation time for each experiment is 10 minutes. Every set of experiments has 50 runs, and the results are the average values out of the 50 runs, with a 95% confidence interval. The entire network is mostly disconnected in the experiments for light traffic scenarios and heavy traffic scenarios. In the experiment for different node densities, the entire network varies from mostly disconnected to mostly connected. Therefore, the scalability of the protocols is also investigated through the experiments.

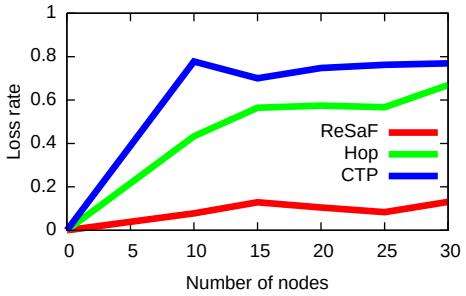
We use four metrics to evaluate the performance: *packet loss rate*, *number of hops per packet*, *packet latency*, and *energy consumption*. The packet loss rate is the ratio between the number of




*(a) Light traffic load scenarios*




*(b) Heavy traffic load scenarios*




*(c) Different node densities*

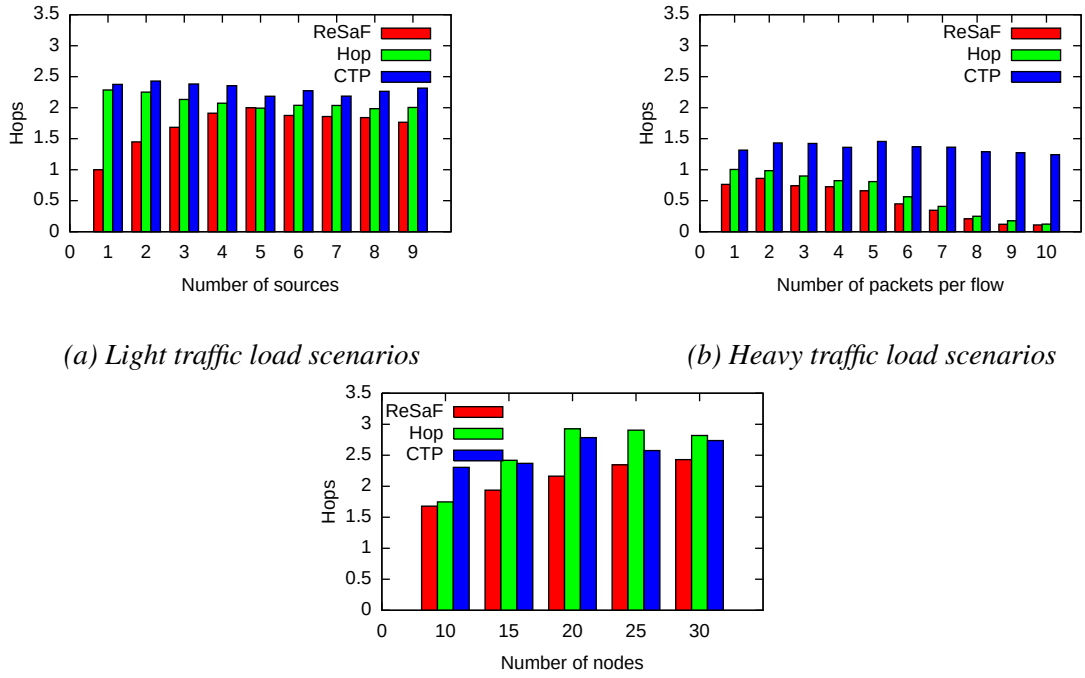
Figure 6.4: Packet loss rate under different scenarios

distinct packets received at the sink and the number of packets sent by all sources. The number of hops per packet is the hop count of a packet from its source to the sink. The packet latency is the time between the moment the packet generated by a source to the time it is received by the sink. The energy consumption is measured as the sum of transmission and reception times of all nodes in the network.

## 6.6.2 Experiment Results and Performance Evaluation

### 6.6.2.1 Light traffic load scenarios

In this experiment we generate 10 nodes in the network area, and one of them is the sink. The number of sources varies from one to nine. Each source sends one packet, all sources transmit simultaneously. This experiment investigate the performance of the protocols in uncongested sparse networks.




*(a) Light traffic load scenarios*

*(b) Heavy traffic load scenarios*

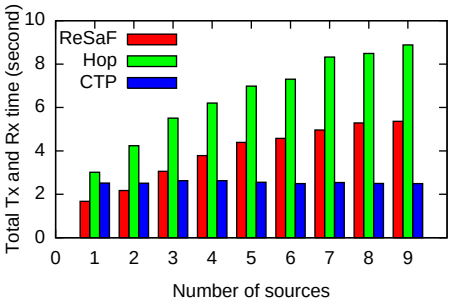
*(c) Different node densities*

Figure 6.5: Average number of hops per packet under different scenarios

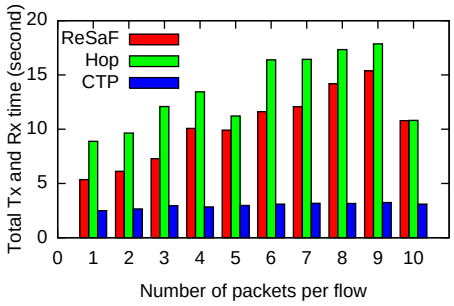
Figure 6.4 (a) shows that ReSaF has the lowest loss rate, followed by Hop, and CTP has the highest loss rate. The high loss rate of Hop is a result of its reliance on backpressure to route packets that performs poorly when there is not enough traffic. Since CTP does not provide effective end-to-end loss recovery, it is less reliable in intermittently connected situations where the packet loss is high.

ReSaF has the smallest number of hops per packet of all three protocols (Figure 6.5 (a)). Because of the conditional-forwarding routing algorithm of ReSaF, the next hop can only be the one with higher delivery probability, more available energy and shorter queue length than s sender, which reduces the forwarding times of packets. Hop forwards packets to the nodes with more available buffer regardless of the delivery probability and available energy, resulting in a larger average number of hops per packet than ReSaF. The routing metric of CTP is the estimated link state, which causes CTP to have the largest number of hops of the three protocols.

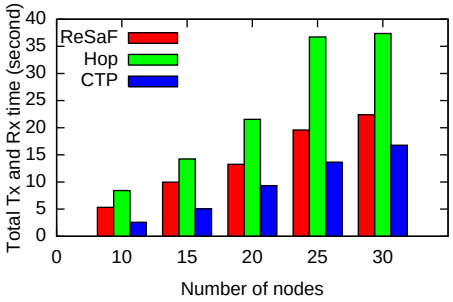
CTP has a lower packet latency than ReSaF and Hop (Figure 6.7 (a)). The reason is that




*(a) Light traffic load scenarios*




*(b) Heavy traffic load scenarios*




*(c) Different node densities*

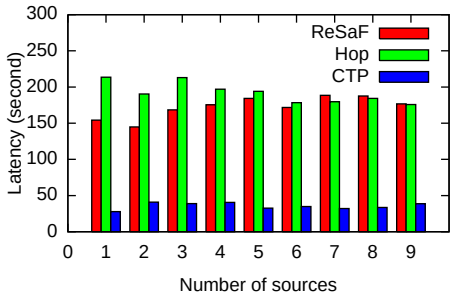
Figure 6.6: Energy consumption (measured as the total transmission and receiving time) under different scenarios

ReSaF needs to wait for the appropriate next hop, while Hop may direct packets along a less optimal path.

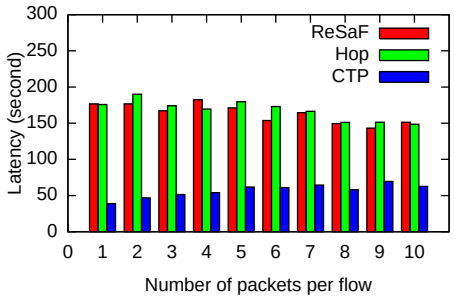
Due to the differences in routing mechanisms, ReSaF uses less energy than Hop (Figure 6.6 (a)). Deliver packets in Hop may need more intermediate nodes, which leads to more energy consumption than ReSaF. CTP has fairly constant energy consumption regardless of the number of source, since CTP drops the packets after a certain number of retransmission.

#### 6.6.2.2 Heavy traffic load scenarios

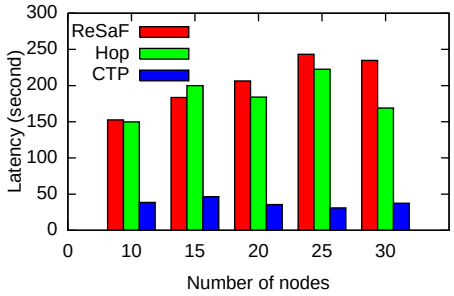
In this experiment we generate 10 nodes in the network area, one of them is the sink. All nodes except the sink are sources, and each source sends one flow. Each flow is sent in burst, and all sources start transmitting at the same time. The flow length is varied from one to 10 packets. This experiment investigates the performance of the protocols in congested sparse networks.




*(a) Light traffic load scenarios*




*(b) Heavy traffic load scenarios*




*(c) Different node densities*

Figure 6.7: Packet latency under different scenarios

From Figure 6.4 (b), we can see that CTP still has the highest packet loss rate. The packet loss rate of Hop decrease with the flow length increasing, while the packet loss rate of ReSaF increase with the flow length increasing. The packet loss rate of Hop demonstrates that Hop performs well when there is sufficient traffic in the network. The packet loss rate of ReSaF shows that ReSaF can deal with light traffic load scenarios better than Hop. When the traffic load is heavy, packets in ReSaF need to wait longer to find an appropriate next hop. This can be explained by Figure 6.5 (b), the number of hops of ReSaF decreases as the flow length increases. The number of hops per packets with Hop is higher than that of ReSaF when traffic is heavy, allowing Hop to deliver packets to the sink more quickly. Similar to light traffic load scenarios, Figure 6.7 (b) shows that Hop and ReSaF have a higher packet latency than CTP. As shown in Figure 6.6 (b), ReSaF has less energy consumption than Hop when flow length is short, and the energy consumption of ReSaF approaches the level of Hop as the flow length increases. This is because when traffic load is high, Hop can get enough pressure to route packets more efficiently, reducing the amount of energy

spent.

#### 6.6.2.3 Different node densities

In this experiment we vary the node density of the network from 10 to 30 while keeping the network area constant. One node is the sink, all the remaining nodes are sources. Every source sends one packet, all sources send at the same time. This experiments tests the scalability of the protocols as the network density increases.

Figure 6.4 (c) shows that ReSaF has lower packet loss rate than Hop and CTP independent of the node density. Same as the first two experiments, ReSaF has the smallest number of hops per packet of the three protocols (Figure 6.5 (c)). As shown in Figure 6.7 (c), ReSaF and Hop have higher packet latency than CTP regardless of the network size. However, Figure 6.6 (c) shows that the energy consumption of CTP increase with the node density growing, this is because CTP needs to spend more energy on routing information maintenance. The energy consumption of Hop and ReSaF increases when the network size growing too, but ReSaF still has less energy consumption than Hop, due to the routing mechanism of ReSaF.

## 6.7 Discussions

ReSaF provides end-to-end reliability, congestion control, storage management, and conditional-forwarding routing algorithm. It can provide a lower loss rate, smaller number of hops per packet and less energy consumption regardless of the traffic load in networks than other hop-by-hop protocols employing in-network storage and backpressure to direct packets. This is achieved by the conditional-forwarding routing mechanism of ReSaF that a node forwards packets only if there is an appropriate next hop that has has the best delivery probability, available buffer and energy in the neighborhood and has higher delivery probability and more available buffer and energy than the sender.

The routing decisions are made based on the delivery probability in ReSaF. The accuracy

of the delivery probability can effect the performance of the protocol. Therefore, a study of the delivery probability is necessary (given in Chapter 7). During our study, we found that the connectedness of a mobile network is not static and uniform. The connectedness of a network may vary from time to time, and the number of clusters and the sizes of these clusters may also change. Therefore, a way to optimized the protocol is to make routing decisions with respect to the current network status locally. The proposed method, connectedness-aware copy-adaptive routing protocol, is introduced in Chapter 7.

# Chapter 7

# Connectedness-Aware Copy-Adaptive Routing Protocol

Single-copy and multiple-copy are two well-known schemes used in intermittently connected networks [69, 65]. Comparing with multiple-copy schemes, single-copy [19, 83] schemes can reduce the resources spent on forwarding duplicate copies, but are less robust to packet loss caused by transmission or node failures. Multiple-copy schemes [75, 47, 21] are more robust to packet loss, but simply flooding packets in a network can lead to resource exhaustions and degrade the network performance. Hence, copy-control is critical especially in resource limited networks.

Many of the existing solutions provide copy-control through setting a threshold $L$. These schemes *spray* $L$ copies of a packet into a network, then stop injecting more copies and let the nodes that carry those copies to either forward these copies to other nodes [68, 56] or wait for a meeting with a destination [67, 22]. Even though applying a user-defined threshold $L$ to control the number of copies of a packet in networks can help in limiting congestion and resource consumption caused by flooding, it is less flexible to handle possible variations in node density or network connectedness.

This work, *Connectedness-Aware Copy-Adaptive Routing Protocol (CACAR)*, aims to achieve a high delivery rate independent of the node density without sacrificing the energy efficiency and

the assistant from GPS.

## 7.1 Protocol Overview

Based on the previous exposition, the desirable design goals of the proposed multiple-copy routing scheme for ICDT-WSNs can be summarized as follows:

- Have a better delivery rate that is higher than the existing multiple-copy routing schemes, independent of the network connectedness and node density.
- Achieve lower delivery latency than multiple-copy routing schemes with/without copy-control, which means an effective copy-control and the appropriate routing decisions are important to an efficient routing scheme.
- Be energy efficient that is spending a small amount of transmission and reception time to deliver a packet under all conditions, which also indicates to perform fewer transmissions on duplicate copies to deliver a packet.
- Be highly scalable that is easily to adapt to variations on network connectedness and node density without complicate recalculation and reconfiguration.

According to the above goals, the novel routing scheme, (CACAR), is able to route packets to the appropriate next hops with adaptive copy-control mechanism. As a result, this proposed scheme could achieve high delivery rate and low delivery latency without sacrificing energy efficiency. The notation used in this chapter is summarized in Table 7.1.

CACAR is composed of two parts:

- *Neighbor discovery*: in WSNs, nodes usually generate beacons periodically to announce their current status to the neighbors, which can be utilized to update routing information. We take advantage of the beacons to maintain the *intimacy* (defined by Definition 7.1.1) of nodes in CACAR. The pseudo code of the intimacy updating algorithm is shown in Algorithm 7.

Table 7.1: Notation Summarization

| Variable | Definition |
|---|---|
| $I$ | the intimacy |
| $I_d$ | the intimacy of the destination |
| $I_{max}$ | the maximum intimacy in a neighborhood |
| $I_{min}$ | the minimum intimacy in a neighborhood |
| $I_{MAX}$ | the maximum intimacy in a network |
| $I_{MIN}$ | the minimum intimacy in a network |
| $d$ | the destination |
| $C$ | the average maximum cluster size |
| $N$ | a network $N$ |
| $\|N\|$ | the number of nodes in a network |
| $c_N$ | the normalized maximum cluster size |
| $F$ | the scale of the intimacy range |

- *Packet forwarding*: nodes in CACAR make forwarding decisions locally based on the network *connectedness* (the details about connectedness are described in Section 7.2). If $node_i$ estimates the network is connected, it forwards packets to the *best node* (defined by Definition 7.1.2) in the neighborhood. Otherwise, $node_i$ forwards packets to all the *better nodes* (defined by Definition 7.1.3) in the neighborhood. The pseudo code for the packet forwarding algorithm is shown in Algorithm 8).

**Definition 7.1.1.** (Intimacy). An intimacy is a metric to measure the closeness of a node to the destination.

We utilize the idea of *Erdős number* in this work to measure the closeness. The destination has the lowest intimacy value, $I_d = I_{MIN} = 0$, and $I_d$ is a constant. The initial intimacy values of all the rest regular nodes equal to $I_{MAX} = \infty$. The intimacy values of the regular nodes decay per

time unit according to Equation 7.1, and get updated according to Algorithm 7.

$$I_i^{'} = I_i + 1. \tag{7.1}$$

A low intimacy value indicates that a node is close to the destination. The intimacy is discussed in detail in Section 7.2.

**Definition 7.1.2.** (Best Node). The best node of $node_i$ is a neighbor of $node_i$ whose intimacy value is the lowest in $node_i$'s neighborhood and is also lower than $I_i$.

**Definition 7.1.3.** (Better Node). A better node of $node_i$ is a neighbor of $node_i$ whose intimacy value is lower than $I_i$.

The set of better nodes of $node_i$ includes the best node of $node_i$.

**Algorithm 7** CACAR: Intimacy Updating

```
1: I_min = I_i
2: for all node_j in node_i's neighborhood do
3:     if I_j < I_min then
4:         I_min = I_j
5:     end if
6: end for
7: if I_min ≠ I_i then
8:     I_i = I_min + 1
9: end if
```

### 7.1.1 Adaptability of Neighbor Discovery

The length of beaconing interval for neighbor discovery is a tradeoff between the accuracy of neighbor information and the energy efficiency [18]. A small interval consumes more energy than a large interval, while provides more up-to-date network information. Therefore, CACAR uses an adaptive beaconing to save energy while trying to get the latest information from neighborhood by extending the Trickle algorithm [36].

Trickle transmits the version number of nodes' information based on a randomized timer. The basic idea of Trickle is that a node suppresses its announcement after the interval expires by

**Algorithm 8** CACAR: Packet Forwarding

```
1: for all packets in node_i do
2:   if node_i estimates the network is connected then
3:     best node = node_i
4:     for all node_j in node_i's neighborhood do
5:       if I_j < I_best node then
6:         best node = node_j
7:       end if
8:     end for
9:     if best node ≠ node_i then
10:      forward the packet to the best node
11:    end if
12:  else
13:    for all node_j in node_i's neighborhood do
14:      if I_j < I_i then
15:        forward a copy to node_j
16:      end if
17:    end for
18:  end if
19: end for
```

doubling the interval up to a maximum value $\tau_h$ if it hears the same version number from another node, and shrinks its interval to a small value $\tau_l$ if it hears a new version number. If all nodes have the same version number, their interval increases exponentially up to $\tau_h$.

CACAR applies Trickle in such a way that the version number gets updated when a node's intimacy gets updated. Therefore, the information that nodes are getting closer to the destination is announced quickly, and energy spent on announcing information for nodes are drifting away from the destination is significantly reduced.

### 7.1.2 Adaptability of Packet Forwarding

CACAR takes advantage of the goodness of multiple-copy coupled with a delivery probability scheme that packets can be delivered more quickly by being forwarded to nodes with higher delivery probabilities. Instead of using delivery probability, CACAR employs the intimacy as a metric to measure the network connectedness and to decide the appropriate next hops. Unlike other multiple-copy schemes that forward copies to all nodes with higher delivery probabilities,

CACAR adopts a copy-control mechanism that is different from much of the existing work presetting the allowed number of copies. The copy-control mechanism of CACAR is based on the network connectedness in a way such that nodes forward packets to the neighbor that is closest to the destination in the neighborhood if the network is estimated as connected, or generate copies to all nodes that are closer to the destination if the network is measured as disconnected. The connectedness is estimated by nodes locally, and is adaptable to the changes on network connectedness or node density.

The hop-level reliability of CACAR is achieved through hop-level acknowledgement. After receiving a data packet, the receiver sends an acknowledgement to the packet sender. If the received packet is a duplicate packet, the receiver sets the *duplicate packet flag* in the acknowledgement to indicate dropping the packet. If the data packet is from a node with lower intimacy value, the receiver sets the *routing loop flag* in the acknowledgement to indicate recalculating intimacy values. After receiving such an acknowledgement, the sender broadcasts a neighbor discovery message with its latest intimacy value for intimacy update. If the receiver has no available space for the incoming packet, it sets the *drop flag* in the acknowledgement. After receiving such an acknowledgement, the sender broadcasts a neighbor discovery message with its current intimacy value later to find another next hop. If no acknowledgment received by the packet sender before a timeout (nodes set a timer for acknowledgement after sending a data packet), the sender retransmits the data packet. After a certain number of retransmissions, the sender drops the packet. Each packet has a time to live (TTL) field. Once the TTL is larger than the maximum value, the packet is dropped.

The above discussion about CACAR leaves an issue on how to estimate the connectedness of a network locally, which is detailed in the following section.

## 7.2 Connectedness Measurement

Due to the high node failures and transmission errors in ICDT-WSNs, multiple-copy routing mechanism is more efficient and robust than single-copy mechanisms. However, too many duplicate copies of packets in a network will lead to unnecessary packet transmissions, energy inefficiency and longer queuing delay. It is difficult to decide the appropriate number of copies of packets for multiple-copy mechanism in intermittently connected scenarios, therefore, we propose a copy-adaptive routing mechanism letting nodes locally decide the number of copies and next hops at each forwarding step according to the connectedness measured by nodes. In the rest of this section, we will introduce the method to measure the connectedness.

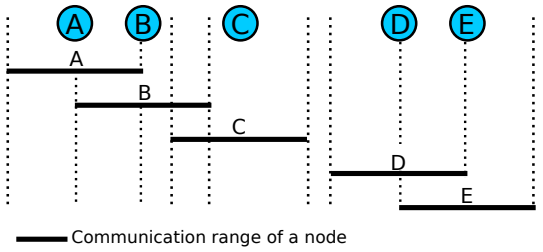



Figure 7.1: In a network with five nodes, A can talk to B and C, B can talk to A and C, C and talk to A and B, D and E and talk to each other.

It is necessary to define a meaningful metric to measure the connectedness before we get into the details. Since there is no agreement on the metric for connectedness in such intermittently connected scenarios, we define the *average maximum cluster size* as a meaningful metric in our study.

**Definition 7.2.1.** (Average Maximum Cluster Size). Average maximum cluster size is the average number of nodes each node can communicate with. This includes the nodes that in the immediate neighborhood and the nodes that can be reached through multi-hop communication. Let $|N|$ be the number of nodes in a network $N$ and $N_i$ be the number of nodes $node_i$ can communicate with.

The average maximum cluster size $C$ is expressed as:

$$C = \frac{\sum_{i \in N} N_i}{|N|}. \tag{7.2}$$

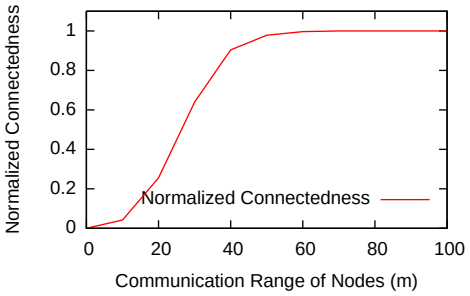
In the example shown in Figure 7.1, the average maximum cluster size of this network is $\frac{2+2+2+1+1}{5} = 1.6$.

To measure the connectedness independent of the network size, we introduce the *normalized maximum cluster size*.

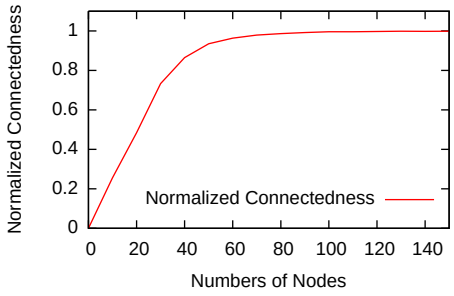
**Definition 7.2.2.** (Normalized Maximum Cluster Size). Normalized maximum cluster size of a network is the value of average maximum cluster size of the network divided by the number of nodes in the network subtracted by one. Let $c_N$ be the normalized maximum cluster size:

$$c_N = \frac{C}{|N| - 1}. \tag{7.3}$$

The normalized maximum cluster size of the network shown in Figure 7.1 is $\frac{1.6}{4} = 0.4$.




*(a) The impact of communication range on $c_N$ of a network with 10 nodes moving in a $100 \times 100$ m area*




*(b) The impact of the number of mobile nodes on $c_N$ of a network in a $100 \times 100$ m area and communication range of 20 m*

Figure 7.2: Normalized Connectedness

The $c_N$ of a network has such features: In a network with a constant number of nodes, larger communication range leads to higher $c_N$ (Figure 7.2 (a)); In a network composed of nodes with a fixed communication range, larger number of nodes lead to a higher $c_N$ (Figure 7.2 (b)).

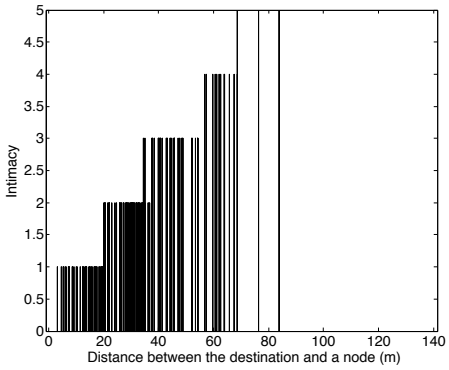
We term the connectedness of a network as:

- If $c_N = 1$, the network is a *connected network*. In this case, every node can communicate with all the rest nodes via the single-hop or multi-hop manner.
- If $c_N < 1$, the network is an *intermittently connected network*. In this case, some nodes are isolated from other nodes.

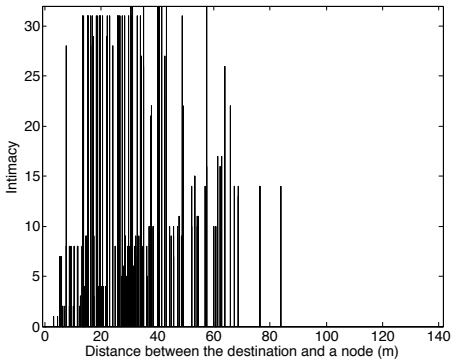
Determining the global network connectedness is difficult in intermittently connected wireless sensor networks. We propose a novel method to estimate the connectedness by using local observations.

Before we proceed, we discuss about the intimacy. We've defined *intimacy* and introduced how to update and decay the intimacy value of nodes in Section 7.1. Intimacy is used to measure the closeness of a node to the destination, and a low intimacy value means a node is close to the destination. The minimum intimacy value in a network is the intimacy value of the destination, since no one other than the destination is closer to itself. The maximum intimacy value in a network is the initial values of regular nodes. Since nodes could move toward to or drift away from the destination in ICDT-WSNs due to the node mobility, it is necessary to increase the intimacy for the nodes that move away from the destination. We increase the intimacy values of these drifting away nodes by one. As described in Definition 7.1.1, the intimacy value decays per time unit, and updates every time a node discovers another one with lower intimacy value. Therefore, the intimacy value of a node keeps growing if the node does not meet with any nodes with lower intimacy values than its own intimacy value during a certain period of time. $I_{MAX}$ and $I_{MIN}$ are shipped with the scheme when nodes are initially launched.

In a connected network where $c_N = 1$, every node can communicate with the destination directly or through the single-hop/multi-hop manner. Therefore, the nodes can be constructed into a tree structure with respect to their intimacy values. The destination is the root of the tree, the nodes that are in the communication range of the root are the children at the first layer of the tree, the nodes that are single-hop away from the destination are the children at the second layer of the




*(a) The intimacy distribution of 150 mobile nodes with communication range 20 m in a* $100 \times 100$ *m area (*$c_N$ *of the network is 1)*




*(b) The intimacy distribution of 150 mobile nodes with communication range 5 m in a* $100 \times 100$ *m area (*$c_N$ *of the network is 0.024)*

Figure 7.3: Intimacy distribution in scenarios with different normalized connectedness

tree, and the nodes that are more hops away from the destination are at lower layers of the tree. The value of intimacy increases per layer in a manner that:

$$I_M = M - 1, \tag{7.4}$$

where $I_M$ is the intimacy of nodes at layer $M$ or $M$ hops away from the destination. For children at layer $M$, their intimacy values fall in to the range $(M-2, M)$.

**Theorem 7.2.1.** *In connected scenarios, the intimacy values of nodes that are* $M$ *hops away from the destination fall in to the range* $(M-2, M)$ *when applying CACAR.*

*Proof.* In connected scenarios, the nodes can be constructed into a tree structure with respect to their intimacy values. According to Algorithm 7, the nodes in the communication range of the destination have the intimacy value $I_d + 1 = 1$, and the nodes in one hop away from the destination have the intimacy value $I_d + 2 = 2$. Therefore, nodes that are $M$ hops from the destination are with the intimacy value $I_d + M - 1 = M - 1$, nodes that are one hop further are with the intimacy value $I_d + M = M$, and nodes that are one hop closer are with the intimacy value $I_d + M - 2 = M - 2$. As a result, the intimacy values of nodes in $M$ hops from the destination fall in to the range $(M-2, M)$. □

The purpose of introducing Theorem 7.2.1 is not to study the connected scenarios, but use it to estimate the connectedness of network. Since Theorem 7.2.1 holds only when $c_N = 1$, when a network is intermittently connected where $c_N < 1$, Theorem 7.2.1 does not hold. Therefore, the range can be used to estimate the connectedness of a network.

Let us use an example to show how well the intimacy range can be held under connected and intermittently connected scenarios. We use the Random Waypoint model in this example. All nodes in a network have the same communication range. The intimacy of nodes grows as the distance between nodes and the destination increases, as shown in Figure 7.3 (a). The number of hops can be transfer to the distant between a node to the destination by applying $H = \left\lceil \frac{D}{R} \right\rceil$, where $H$ is the number of hops, $D$ is the normalized distance converts the geographic distance to the minimum number of hops required to traverse the distance, and $R$ is the communication range of nodes. From this figure, an observation can be made that intimacy values of nodes at $M$ hops from the destination fall into the range $(M-2, M)$. In an intermittently connected network, this property does not hold, as shown in Figure 7.3 (b).

The scale of the intimacy range $(M-2, M)$ is a constant, $F = 2$. To determine the connectedness of a network, we let nodes estimate the connectedness by observing the range of intimacy values of nodes in their neighborhood. Let $I_{max}$ be the maximum intimacy in $node_i$'s neighborhood, $I_{min}$ be the minimum intimacy in $node_i$'s neighborhood, and $\Delta_i$ be the scale of the intimacy range measured by $node_i$:

$$\Delta_i = I_{max} - I_{min}. \tag{7.5}$$

By comparing $\Delta_i$ with $F$, $node_i$ can determine whether the network is connected or intermittently connected. Cases where $\Delta_i \leq F$ are interpreted as an indication that the network is connected. This is because in a connected network the range of the intimacy values of nodes in the neighborhood is $(M-2, M)$. Correspondingly, if $\Delta_i > F$, we conclude that the network is intermittently connected.

The goodness of this connectedness mechanism is that there is no pre-set threshold needed.

Therefore, when the changes on communication range and node density happen, there is no need to reconfigure or recalculate.

Moreover, such connectedness estimation can help to reduce unnecessary copies. As described in Section 7.1, nodes forward packets to the best node if the network is estimated as connected ($\Delta_i \leq F$), which means neighbors are all with high intimacy values or are all with low intimacy values. If all neighbors are close to the destination (all with low intimacy values), packets have a higher possibility to get delivered, hence, forwarding to one next hop can reduce unnecessary traffic. If all neighbors are far from the destination (all with high intimacy values), there is no need to forward packets to equally bad next hops.

## 7.3 Simulation and Experiments

### 7.3.1 Simulation Experiment Setup

CACAR has been implemented on Contiki [11] and evaluated on Cooja [50]. Bonnmotion [1] is used to generate the motion patterns. We compare CACAR with Epidemic routing [75] and the state-of-art routing protocol for wireless sensor network, Collect Tree Protocol (CTP) [20]. CTP is a single-copy protocol but can provide high end-to-end reliability with low energy consumption when a route exists. Comparing with CTP allows us to study the tradeoff between a protocol for intermittently connected WSNs and a protocol for connected WSNs. Epidemic routing protocol is a basic multiple-copy protocol for intermittently connected networks under the assumption that nodes are equipped with infinite buffers. By comparing with Epidemic routing protocol, we can evaluate the advantage of the adaptive copy-control mechanism of CACAR.

We set up two experiments: varying the communication range of nodes and varying the number of nodes in a network. The network area is $100 \times 100$ m. Nodes in this area move according to the Random Waypoint model with speed 4 m/s, and the buffer size is set to 12 packets. One of the nodes is the destination. All the remaining nodes are source nodes transmitting a packet every 30 seconds. The simulation time for each experiment is 10 minutes. Source nodes start transmitting

packets at two minutes after the initiation, and keep transmitting for five minutes. Every set of experiments has 30 runs, and the results are the average values out of the 30 runs, with a 90% confidence interval. The purpose is to study the end-to-end reliability and energy efficiency of CACAR, therefore, we do not investigate the throughput of the protocol.

We use four metrics to evaluate the performance:

**Delivery rate** : the ratio between the number of distinct packets received at the sink and the number of packets sent by all sources

**Number of hops per packet** : the hop count of a packet from its source to the sink

**Packet latency** : the time between the moment the packet generated by a source to the time it is received by the sink

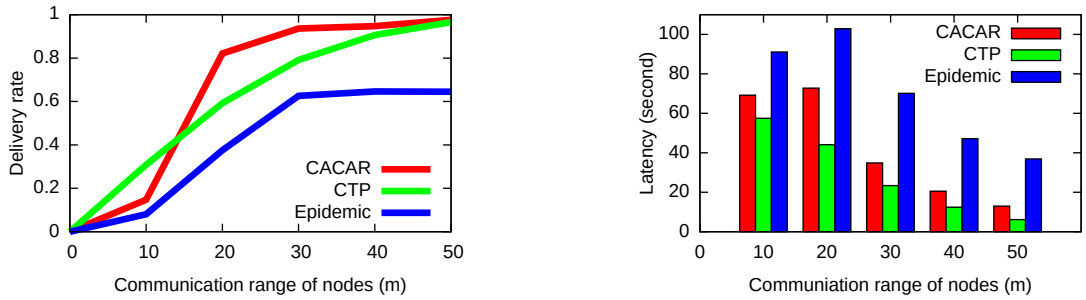
**Energy consumption** : it is measured as the sum of the number of transmission and the number of receptions per delivered packet

### 7.3.2 Experiment Results and Performance Evaluation

#### 7.3.2.1 Variable communication range

This experiment investigates the performance of these protocols under scenarios with different connectedness. In this experiment we generate 10 nodes in a $100 \times 100$ m area and vary the communication range of nodes from 10 m to 50 m while keeping the network area fixed. The $c_N$ of the network varies from 0.042 to 0.978.

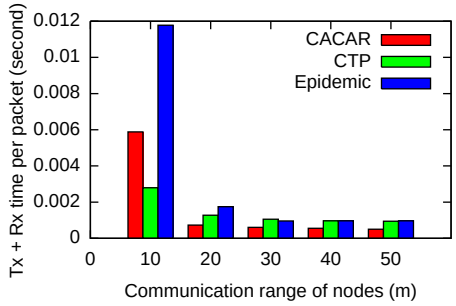
Figure 7.4 (a) shows that the delivery rate of CACAR is higher than CTP and Epidemic in most time. The delivery rate of CACAR is lower than CTP at the communication range of 10 m, since the network is too sparse. Under this scenario, nodes do not get enough contacts to maintain the intimacy and to transmit packets. With the growth of the communication range, nodes can contact more frequently, and the intimacy can be better maintained. According to the routing mechanism of CACAR, the number of nodes forwarding packets to more than one next



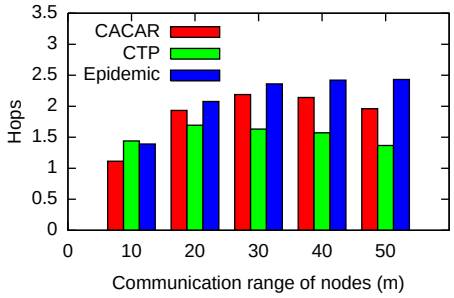

*(a) Delivery rate*




*(b) Packet latency*




*(c) Total Tx and Rx time per packet*

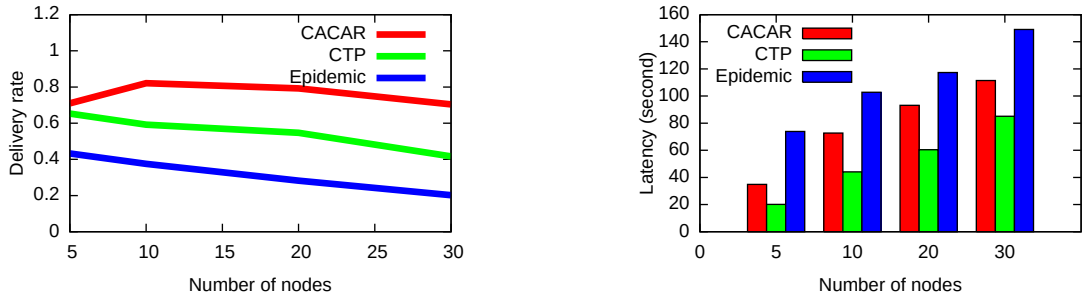



*(d) Number of hops per packet*

Figure 7.4: Ten mobile nodes in a $100 \times 100$ m area, with the communication range varying from 10 m to 50 m.

hops grows as the connectedness decreases. Therefore, CACAR is more robust than CTP when the communication range is small. The delivery rate of Epidemic increases with the growth of the connectedness. However, Epidemic's delivery rate is still lower than the other two protocols. This is because Epidemic routing assumes that the nodes have infinite buffers. Nodes in WSNs typically have limited storage space, hence, the multiple-copy mechanisms like Epidemic result in a significant packet loss due to buffer overflow.

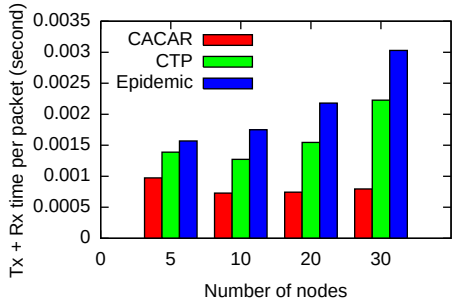
In CACAR, nodes only forward packets to the neighbors with better intimacy values than themselves. Comparing with the route metric of CTP, intimacy is less sensitive and changes less frequently. If there are no appropriate next hops to forward, the senders will hold the packets till the appropriate next hops become available. Hence, the packet latency of CACAR is larger than the latency of CTP. Epidemic has a higher latency than CACAR, since nodes in Epidemic have the average longer sending queue than nodes in CACAR (Figure 7.4 (b)).



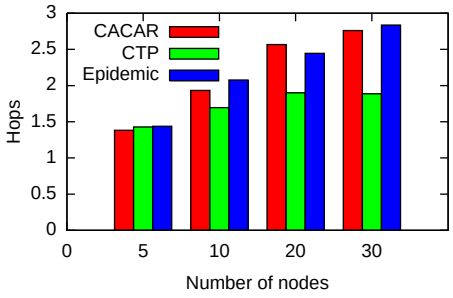

*(a) Packet loss rate*




*(b) Packet latency*




*(c) Total Tx and Rx time per packet*




*(d) Number of hops per packet*

Figure 7.5: Mobile nodes with communication range 20 m in a $100 \times 100$ m area, varying the number of nodes in the network from 5 to 30.

The routing metric in CTP is more sensitive than intimacy in CACAR. Once a node's routing metric gets updated in CTP, all the nodes relying on this node need to update their routing metrics. Therefore, a series of messages will be sent out for routing metric maintenance. As a result, CTP needs larger numbers of transmissions and receptions than CACAR to maintain routing metric under the scenarios where the network topology changes frequently. Moreover, CACAR has more packets delivered than CTP during the same period of time. Hence, CACAR spends less transmission and reception time per packet than CTP (Figure 7.4 (c)). CACAR consumes less energy than Epidemic, since CACAR transits less number of duplicate copies of a packet than Epidemic.

Epidemic transmits a packet to every nodes that are without a copy of the packet, regardless of the nodes' delivery probabilities. Conversely, CTP transmits a packet to the next hop with the highest probability to deliver the packet. As a result, CTP has the lower number of hops per packet than Epidemic. The transmission strategy of CACAR is in between of Epidemic and CTP. If the

connectedness of a network is measured as intermittently connected, the nodes in CACAR transmit a copy of a packet to every nodes with better intimacy values in the neighborhood. Therefore, the number of hops per packet of CACAR is also in between of Epidemic and CTP, as shown in Figure 7.4 (d).

#### 7.3.2.2 Variable node density

The experiment investigates the performance of these protocols under different densities and different traffic loads. In this experiment we vary the number of nodes in a constant network area from 5 to 30, while keeping the communication range of 20 m. The $c_N$ of the network varies from 0.1666 to 0.709, and the number of source nodes varies from 4 to 29 accordingly.

Figure 7.5 (a) shows that the delivery rates of CACAR, CTP and Epidemic decrease with the growth of the number of nodes. This is because the traffic load increases with the growth of the number of nodes. However, CACAR still has a higher delivery rate than CTP and Epidemic due to the different routing mechanisms they employ. The routing mechanism of CACAR is more robust than CTP, and more efficient than Epidemic in intermittently connected scenarios (detailed in Subsection 7.3.2.1).

The packet latencies of CACAR, CTP and Epidemic also increase with the growth of number of nodes (Figure 7.5 (b)), since the average routing paths are getting longer in networks with more nodes. Because of the same reason explained in Subsection 7.3.2.1, CACAR has longer packet latency than CTP, and shorter latency than Epidemic.

As shown in Figure 7.5 (c), CACAR consumes less energy per packet than CTP and Epidemic independently of the number of nodes in a network. The reason is explained in Subsection 7.3.2.1. The energy consumption of these protocols increases with the growth of number of nodes. However, the increasing of CACAR is slower than CTP and Epidemic. This is because CTP needs more transmission and reception to maintain routing tables in dense networks. While for Epidemic, the growth of number of nodes leads to a significant increasing in the number of copies to transmit. Therefore, its energy consumption also increases significantly.

The number of hops per packet of the three protocols increases with the growth of the node density, as shown in Figure 7.5 (b). This is because the average path length grows with the node density increasing.

## 7.4 Discussions

CACAR, a copy-adaptive routing protocol, can provide higher delivery rate without scarifying energy efficiency in intermittently connected scenarios regardless of the traffic load. This is achieved by the connectedness-aware copy-adaptive routing mechanism of CACAR that a node forwards one copy of a packet to the neighbor with the best intimacy value in the neighborhood if the node estimate the network is connected; otherwise, the node forwards a copy of a packet to each neighbor with better intimacy value than the node itself. CACAR allows nodes to perform copy-control and to make routing decision locally based on their local estimation.

# Chapter 8

# Regional Information Dissemination Protocol

The previous work targets at the transmissions between the source nodes and destinations, in which case the source nodes know which nodes are the destinations. However, there are scenarios where the destinations are unknown to the source nodes when they are trying to send packets. The information sent by the source nodes is only interested to nodes in the *region of interest (ROI)*. Due to the mobility of nodes, the nodes in the ROI are highly unstable.

In this chapter, we first propose the network model of regional information dissemination networks. To better describe the mechanisms used in this type of networks for efficient dissemination, the basic epidemic flooding is introduced first, and then we propose several flooding-based mechanisms reducing unnecessary transmissions.

## 8.1 Network Model

Regional information dissemination networks are mission-oriented networks, where all nodes in a network cooperator together to achieve a common task assigned to the network. Hence, the efficiency goal of this type of networks is to reduce the energy consumption from the network perspective instead of from a node perspective.

The regional information dissemination networks operate in a fixed area, where a set of static targets are randomly distributed. A set of mobile nodes move in the area according to a mobility model, detect targets and communicate locally when necessary. Once a node detects a target, it informs other nodes in the ROI of the target. Nodes that receive a target information act as relay nodes to inform other nodes in the ROI of the target via local communication. A ROI of a target is a circle area centering at the target. The target information is only necessary to the nodes in the ROI of the target, not critical to the nodes outside of the ROI. We assume that nodes are aware of their locations, the network area is significant larger than the ROI, and the radius of ROI is larger than the communication range of a node. Consequently, multi-hop relay is necessary to inform nodes in the ROI of a target.

Due to the node mobility, the network is intermittently connected: the contemporaneous path from a node detecting a target to a node in the ROI does not always exist. Even if it exists, the path is highly unstable that may break while being discovered. The mobile nodes can enter and exit the ROI anytime. As a result, the destinations of the target information are uncertain to the nodes that detect the targets. The dissemination mechanism should not rely on the designated information destinations, and should be designed from the perspective of *arbitrary destinations*.

## 8.2 Epidemic Flooding

*Epidemic flooding* [75] lets nodes transmit a copy of a target information to every node that does not have a copy of the target information, after the nodes detect the target or receive a copy of the target information (shown in Algorithm 9).

**Algorithm 9** Epidemic Flooding

```
1: if node_i has a copy of a target information TI_m then
2:   for all node_j in node_i's neighborhood do
3:     if node_j does not have a copy of TI_m then
4:       node_i forward a copy of TI_m to node_j
5:     end if
6:   end for
7: end if
```

In Epidemic flooding, there is not need to know the destination information, a node with a copy of the target information simply forward a copy to other nodes without a copy of the target information in its communication range. Eventually, all nodes in the network will be aware of the target information.

## 8.3 Region Restricted Flooding

Although the epidemic flooding can let the nodes in the ROI of a target be aware of the existence of the target, it generates a significant amount of unnecessary transmissions to inform the nodes outside of the ROI that are not interested in the information. This amount of unnecessary transmission increases with the growth of the network density. Therefore, reducing this amount of unnecessary transmissions is critical to an efficient dissemination protocol.

*Region restricted flooding* lets the nodes in the *restricted region* (see Definition 8.3.1) of a target execute epidemic flooding; The nodes outside of the restricted region are not allowed to transmit the target information (shown in Algorithm 10).

**Algorithm 10** Region Restricted Flooding

```
1: if node_i has a copy of a target information TI_m and node_i is in the restricted region then
2:    for all node_j in node_i's neighborhood do
3:       if node_j does not have a copy of TI_m then
4:          node_i forward a copy of TI_m to node_j
5:       end if
6:    end for
7: end if
```

**Definition 8.3.1.** (Restricted Region). The restricted region of a target is a circle area centered at the target, which can be seen as an enlarged or shrunk ROI of the target. The radius of the restricted region, $R_\alpha$, is calculated as:

$$R_\alpha = \alpha R, \tag{8.1}$$

where $R$ is the radius of the ROI, and $\alpha \sim 1$. The lower bound of $\alpha$, $\alpha_{min}$, is calculated as:

$$\alpha_{min} = 1 - \frac{r_t}{R}, \tag{8.2}$$

where $r_t$ is the communication range of nodes.

Therefore, when $\alpha = \alpha_{min}$, the nodes located at the border of the restricted region, $node_a$, can still provide transmissions to the nodes in the ROI but outside of the restricted region of the target. $node_b$, as shown in Figure 8.1.

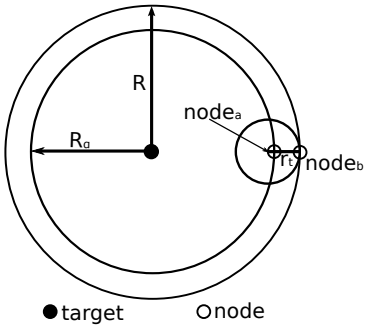



Figure 8.1: Lower bound of the restricted region

The restricted region can be controlled by varying the value of $\alpha$. As shown in Figure 8.2, when $\alpha = 1.0$, the restricted region equals to the ROI.

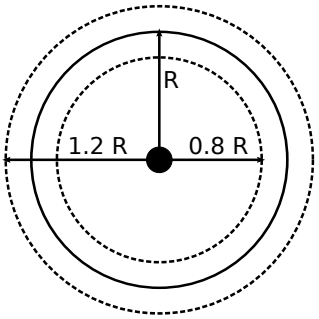



Figure 8.2: Restricted regions for varying $\alpha$

## 8.4 Probability Based Flooding

*Probability based flooding* is a flooding-based protocol that lets a node with a copy of a target information transmit the information to other nodes that are not aware of the target with a probability, $p \in (0, 1]$ (shown in Algorithm 11). When $p = 1$, the probability based flooding turns to the epidemic flooding.

**Algorithm 11** Probability Based Flooding

```
1: if node_i has a copy of a target information TI_m then
2:   for all node_j in node_i's neighborhood do
3:     if node_j does not have a copy of TI_m then
4:       generate a random number, rand ∈ (0, 1)
5:       if rand < p then
6:         node_i forward a copy of TI_m to node_j
7:       end if
8:     end if
9:   end for
10: end if
```

Probability based flooding reduces the transmissions via the control of the growth rate of the total number of transmissions in a network. Although it can reduce the number of transmissions to inform nodes in the ROI of a target, probability based flooding lets all nodes in a network be aware of the target information eventually. The transmissions that happen outside of the ROI of a target are not eliminated in probability based flooding.

## 8.5 Region Restricted Probability Based Flooding

*Region restricted probability based flooding* is a hybrid protocol of the region restricted flooding and probability based flooding. In region restricted probability based flooding, the nodes in the restricted region of a target transmit the target information to other nodes that are not aware of the information with a probability, $p \in (0, 1]$ (shown in Algorithm 12).

When $p = 1$, the region restricted probability based flooding turns to the region restricted flooding. In another word, we can consider the region restricted flooding as a special case of the

**Algorithm 12** Region Restricted Probability Based Flooding

1: **if** $node_i$ has a copy of a target information $TI_m$ and $node_i$ is in the restricted region **then**
2:  **for all** $node_j$ in $node_i$'s neighborhood **do**
3:  **if** $node_j$ does not have a copy of $TI_m$ **then**
4:  generate a random number, $rand \in (0, 1)$
5:  **if** $rand < p$ **then**
6:  $node_i$ forward a copy of $TI_m$ to $node_j$
7:  **end if**
8:  **end if**
9:  **end for**
10: **end if**

region restricted probability based flooding.

In addition to control the growth rate of the number of transmissions, unlike probability based flooding, region restricted probability based flooding reduces the unnecessary transmissions outside of the ROI of a target by applying the restricted region. As a result, in region restricted probability based flooding, transmissions only happen in the restricted region with a certain probability.

## 8.6 Simulation and Experiments

### 8.6.1 Simulator and Experiment Setup

The epidemic flooding (F), region restricted flooding (R), probability based flooding (P) and region restricted probability based flooding (RP) have been implemented on Contiki OS [11] and evaluated on Cooja [50]. Bonnmotion [1] is used to generate intermittently connected mobile scenarios.

In this study, we set one target at the center of a $100 \times 100\ m$ network. The radius of the ROI of the target is set to $R = 25\ m$. Nodes in the network moves according to the *Random Waypoint* model, and the transmission range and sensing range of nodes are set to 10 $m$. To inspect the effect from the parameters to the performance of protocols, we vary the values of $\alpha$ and $p$ in region restricted flooding, probability based flooding and region restricted probability based flooding.

The value of $\alpha$ in region restricted flooding varies from 0.6 (calculated via Equation 8.2), to 1.2. The value of $p$ in probability based flooding varies from 0.125 to 0.5. For the region restricted probability based flooding, we vary $p$ from 0.125 to 0.5 while keeping $\alpha$ equals to 1, and inspect the performance when $p$ equals to 0.125 and $\alpha$ equals to 0.6. In order to study the scalability of the protocols, we vary the network density from 5 to 40 nodes. The network varies from mostly disconnected to partially connected. For each density, we set 50 runs, each run is 10 minutes, and the results are the average values out of the 50 runs, with a 95% confidence interval.

In the investigation of the fixed network density, we use three metrics to evaluate the protocols:

**Informed Ratio** : the percentage of nodes in the ROI that are informed (shown in Equation 8.3)

**Total Cost** : the total number of transmissions of all node in a network

**Cost** : the rate between the total cost and the number of informed nodes in the ROI (shown in Equation 8.4)

$$\text{Informed Ratio} = \frac{\text{num. of informed nodes in the ROI}}{\text{num. of nodes in the ROI}} \times 100\% \tag{8.3}$$

$$\text{Cost} = \frac{\text{Total Cost}}{\text{num. of informed nodes in the ROI}} \tag{8.4}$$

In the investigation of the varied network density, we take two steps: evaluate the performance at time 60 seconds, and evaluate the performance when the informed ratio reaches at 80%. We use the relative cost instead of the total cost in this investigation for a comparison to show the trends as the network scales. In the second step, we use the dissemination latency instead of the informed ratio to inspect the dissemination speed. The relative cost and dissemination latency are defined as:

**Relative Cost** : the average number of transmissions per node (shown in Equation 8.5)

**Dissemination Latency** : the time needed to inform 80% of nodes in the ROI

$$\text{Relative Cost} = \frac{\text{Total Cost}}{\text{num. of nodes in a network}} \tag{8.5}$$

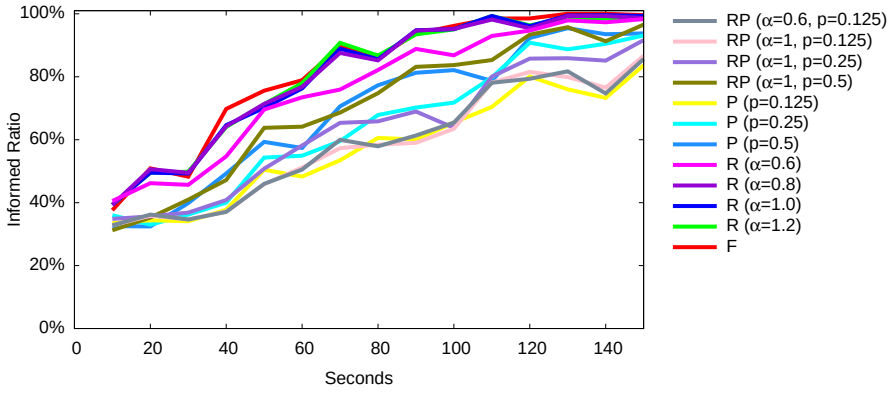
### 8.6.2 Experiment Results and Performance Evaluation

#### 8.6.2.1 Fixed Network Density

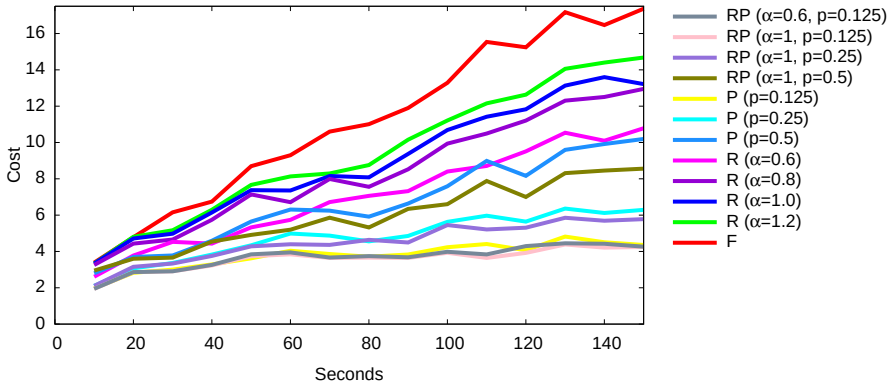
In this experiment, we set the network density to 10 nodes, and investigate the performance every 10 seconds for 150 seconds since the target has been detected by the first node.

As shown in Figure 8.3, the informed ratio, cost and total cost increase with the time goes by, independent of the values of $p$ and $\alpha$. The epidemic flooding has the highest informed ratio, cost and total cost than others, followed by the region restricted flooding. Larger $\alpha$ leads to higher informed ratio, higher cost and more transmissions. When $\alpha$ in [0.8, 1.2], restricted region flooding has very close informed ratio to the epidemic flooding, while the cost of restricted region flooding is much lower than the epidemic flooding. This is because region restricted flooding reduces the unnecessary transmissions outside of the ROI, which can be observed from Figure 8.3 (c) that the region restricted flooding has a smaller number of transmissions than the epidemic flooding.

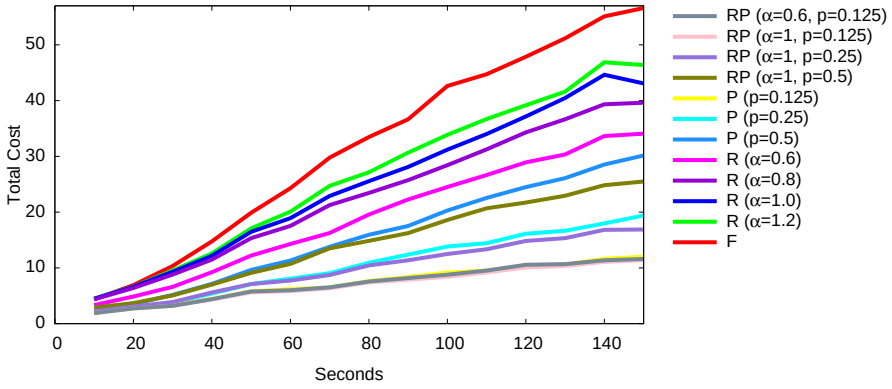
We can also observe from Figure 8.3 that the probability based flooding and region restricted probability based flooding have smaller total cost than the epidemic flooding and region restricted flooding. Smaller $p$ leads to less number of transmissions. With the same $p$ value, region restricted probability based flooding generates less transmissions than the probability based flooding. With the same $p$ value, the differences on the total cost of region restricted probability based flooding with different $\alpha$ values are very small. Therefore, in the region restricted probability based flooding, $p$ is the more impactful parameter.




*(a) Percentage of informed nodes in the ROI*




*(b) Average number of transmissions to inform a node in the ROI*




*(c) Total number of transmissions in the network*

Figure 8.3: Evolution of information propagation and costs from 0 to 150 seconds for a network with 10 nodes and an area of $100 \times 100\ m$

#### 8.6.2.2 Varied Network Density - Step I

In this experiment, we vary the network density from 5 to 40 nodes, and investigate the performance at time 60 seconds since the target has been detected by the first node.

As shown in Figure 8.4 (a), region restricted flooding has higher informed ratio than probability based flooding. In the region restricted probability based flooding, those with higher $p$ values can inform more nodes in the ROI of the target at 60 seconds. This is because more frequent transmissions can disseminate the information into a network more quickly.

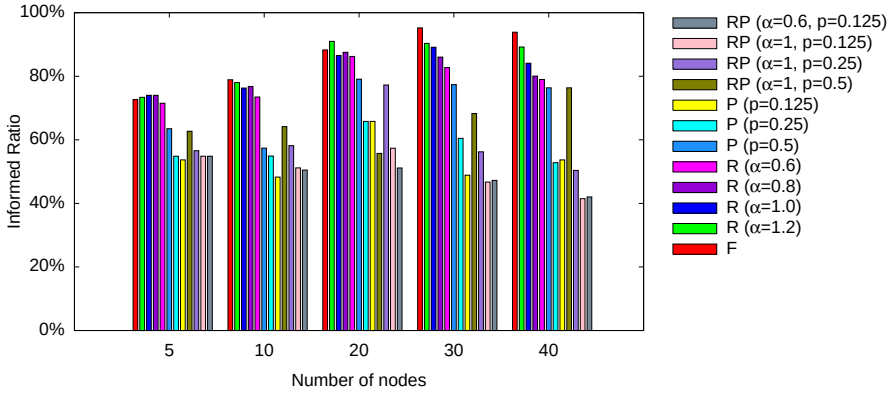
Figure 8.4 (b) shows that the differences between the costs of region restricted flooding, probability based flooding and region restrict probability based flooding and the cost of the epidemic flooding increase with the growth of the network density. This is because the unnecessary transmissions of the epidemic flooding grow significantly with the increase of the network density.

From Figure 8.4 (c), we can observe that probability based flooding can better control the growth rate of the number of transmissions than the epidemic flooding. However, the unnecessary transmissions outside of the ROI of the target still increases with the growth of the network density, which is especially prominent when $p$ equals to 0.5. The region restricted flooding can control the unnecessary transmissions better than probability based flooding, since the number of transmissions of region restricted flooding does not keep growing with the increase of the network density. The region restricted probability based flooding still consumes the least transmissions independent of the network density.

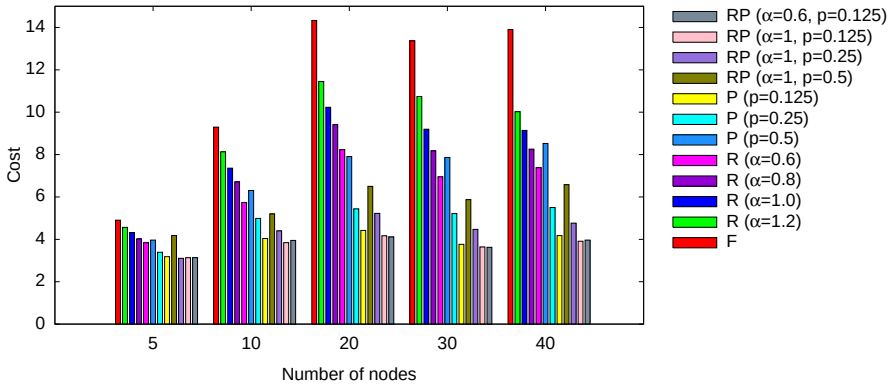
#### 8.6.2.3 Varied Network Density - Step II

In this experiment, we vary the network density from 5 to 40 nodes. We investigate the informed ratio per 10 seconds, and record the dissemination latency, cost and relative cost when the informed ratio first reaches at 80%.

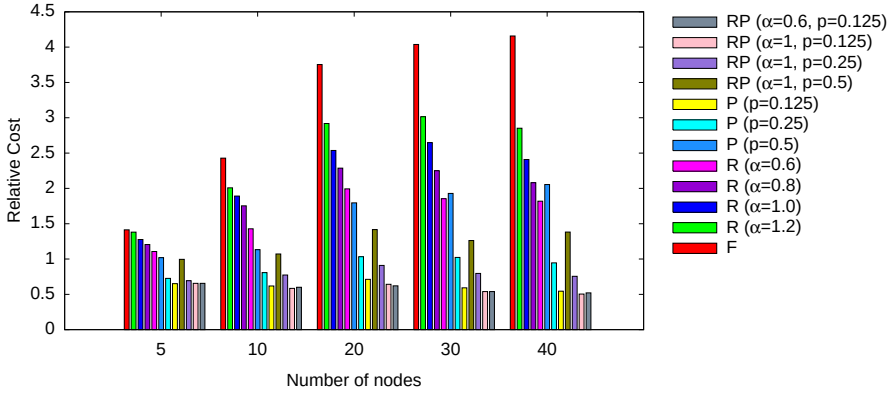
Figure 8.5 (a) shows that the dissemination latency of those flooding based protocol decreases with the network density increase, and after 20 nodes, the dissemination latency starts to increase. This is because, when the network density is low, the opportunity to meet and communicate with




*(a) Percentage of informed nodes in the ROI*

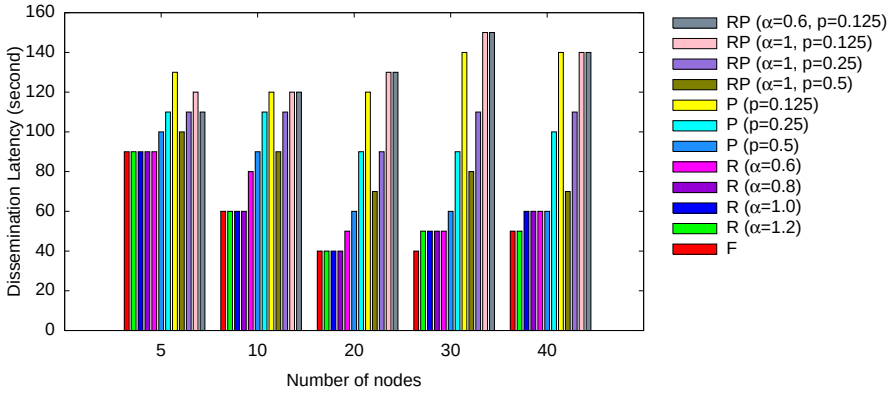



*(b) Average number of transmissions to inform a node in the ROI*

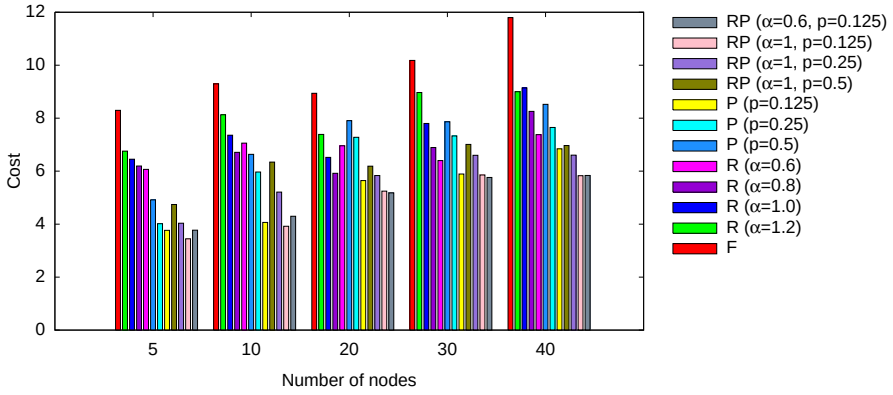



*(c) Average number of transmissions per node in the network*

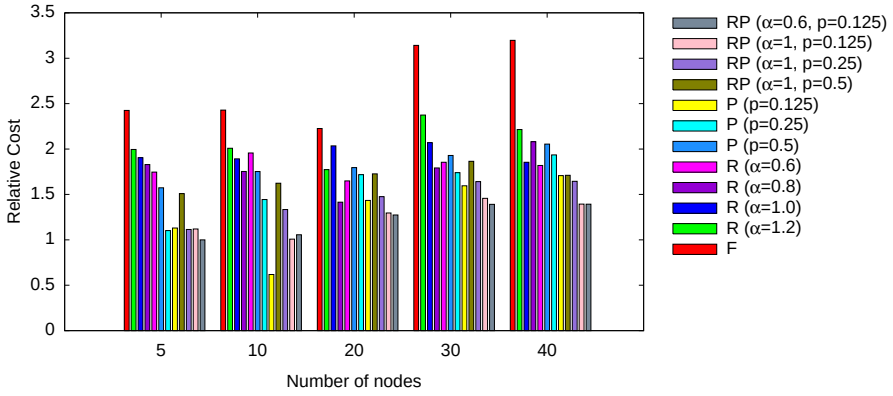
Figure 8.4: Information propagation and costs at time 60 seconds for a network with 5 – 40 nodes and an area of $100 \times 100\ m$




*(a) Time to inform 80% of nodes in the ROI (observed at 10 second intervals)*




*(b) Average number of transmissions to inform a node in the ROI*




*(c) Average number of transmissions per node in the network*

Figure 8.5: Information propagation and costs when 80% of nodes in the ROI were informed for a network with 5 – 40 nodes and an area of $100 \times 100\ m$

other nodes is small, therefore, to disseminate information to the nodes in the ROI needs more time. While with the growth of the network density, the number of nodes in the ROI of a target increases too. As a result, the time to inform the nodes in the ROI of a target increases. For the probability based flooding and region restricted probability based flooding, the opportunity to communicate with other nodes is much smaller than the epidemic flooding and region restricted flooding. Consequently, the dissemination latency of these two protocols is longer than epidemic flooding and region restricted flooding.

As shown in Figure 8.5 (b) and (c), the epidemic flooding has the highest cost, since it generates more transmissions to disseminate information to the nodes in the ROI of a target. The cost of region restricted flooding increases slightly with the growth of the network density. The region restricted flooding can provide the competitive dissemination latency as the epidemic flooding with much less cost. The region restricted probability based flooding can achieve a informed ratio with the lowest cost.

Moreover, the low $\alpha$ value does not always lower the cost. Higher $p$ leads to higher cost. Depending on the value of $p$, the probability based flooding is not always better than the region restricted flooding. For the region restricted probability based flooding with the same $p$, it is not always the case that smaller $\alpha$ leads to lower cost. These observations reflect that the value of $p$ and $\alpha$ should adapt to the network density to achieve the optimum performance, which is currently under investigation.

## 8.7 Discussions

Region restricted probability based flooding employs region restriction and probability based flooding to disseminate the information to the nodes in the ROI. It can provide quick dissemination speed to inform nodes in the ROI with lesser cost by varying the parameter $p$ and $\alpha$. This is achieved by the restricted region mechanism to reduce the unnecessary transmissions to inform nodes outside of the ROI that are not interested in the disseminated information. The relationship

between the value of $p$ and $\alpha$ and the network density is left to be explored.

# Chapter 9

# Conclusions

In this dissertation, we in-depth study the routing, flow-control, error-control and storage management in ICDT-WSNs. Specifically, we develop new energy efficient approaches to solve these problems without a request on large storage space.

## 9.1 Routing

In our study, routing has been divided into two groups: information transmissions with known destinations and information dissemination with arbitrary destinations. To transmit information with known destinations, the performance can be improved by taking advantage of the utility levels of nodes, such as intimacy, queue length, free storage space and available energy level. The study [40] shows that the solution, ReSaF, which employs the utility levels of nodes can achieve higher delivery rate and smaller number of hops per packet than other hop-by-hop protocols that take advantage of the the in-network storage and backpressure to direct packets.

Moreover, we notice that the connectedness of a network varies with the communication range of nodes and the number of nodes in a network. Duplicate packet transmissions are necessary in disconnected networks, but can degrade the performance and energy efficiency in connected networks. It is important to let communication protocol be adaptive to the network connectedness with respect to the number of duplicate transmissions. Our study shows that copy-adaptive routing

protocol, CACAR, can provide higher delivery rate without scarifying energy efficiency in intermittently connected scenarios regardless of the traffic load. This is achieved by the connectedness-aware copy-adaptive routing mechanism, which allows nodes to perform copy-control and to make routing decisions locally based on the measured connectedness.

The second group, information dissemination, aims to disseminate the information to the nodes in the ROI. The information is not critical to the nodes outside of the ROI. Due to the node mobility, nodes in the ROI are highly unstable. In this dissertation, we concentrate on the scenario that the ROI is a circle area with a static target as the center. Our study indicates that region restricted probability based flooding can provide quick dissemination speed to inform nodes in the ROI with smaller number of transmissions than other flooding based schemes. This is achieved by reducing the unnecessary transmissions to inform nodes outside of the ROI that are not interested in the disseminated information.

## 9.2 Flow-control

Flow-control in ICDT-WSNs is via hop-by-hop manners. Nodes with packets to forward measure or estimate congestion and make forwarding decisions locally. As a result, flow-control in ICDT-WSNs is usually coupled with routing.

We investigate two flow-control methods: backpressure and the utility level of nodes. Backpressure directs packets only based on the available space of the neighbors. Taking advantage of the utility level of nodes, packets are forwarded mainly based on the intimacy values, but the available space of neighbors is also take into consideration. The study [40] indicates that using the utility level of nodes can achieve higher delivery rate, shorter packet latency and less energy consumption.

## 9.3 Error-control

In our study, error-control mainly focuses on the loss recovery for packet loss happened during transmission and due to the overflow of storage or queue. Error-control has two levels: hop level loss recovery and end-to-end level loss recovery. Hop level loss recovery is achieved by hop-level acknowledgement, and end-to-end level loss recovery is achieved vi in-network storage and virtual retransmission.

According to our study [42, 43], communication protocols with error-control can provide higher delivery rate than those without error-control in ICDT-WSNs.

## 9.4 Storage management

The purpose of storage management in this dissertation is to maximize the utility of storage. Therefore, we aim to save space for more important packets by removing unnecessary ones. Our work focuses on duplicate checking and unnecessary packet elimination. Nodes execute in-network storage only when the received packets are not in the storage. For those packets that are already confirmed by the destinations, nodes eliminate them from the storage.

We investigate these storage management mechanisms in [42, 43], the results show that the protocols that employ in-network storage with the storage management perform significantly better than those without storage management.

# Bibliography


[1] "BonnMotion," http://net.cs.uni-bonn.de/wg/cs/applications/bonnmotion/, 2013.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128601003024

[3] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad Hoc Networks*, vol. 3, no. 3, pp. 257–279, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870505000168

[4] I. F. Akyildiz and E. P. Stuntebeck, "Wireless underground sensor networks: Research challenges," *Ad Hoc Networks*, vol. 4, no. 6, pp. 669–686, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870506000230

[5] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 12, no. 2, pp. 222–248, quarter 2010.

[6] R. Bartoš, S. G. Chappell, R. J. Komerska, M. M. Haag, S. Mupparapu, E. Agu, and I. Katz, "Development of routing protocols for the solar-powered autonomous underwater vehicle (SAUV) platform," *Wireless Communications and Mobile Computing*, vol. 8, no. 8, pp. 1075–1088, Aug. 2008.

[7] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 307–320. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182838

[8] K. Bur, P. Omiyi, and Y. Yang, "Wireless sensor and actuator networks: Enabling the nervous system of the active aircraft," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 118–125, July 2010.

[9] K. Cao, A. Kinoshita, Y. Takaki, C. Ohta, and H. Tamaki, "Efficient urban broadcast protocol for v2v communications with relay control," in *Vehicular Networking Conference (VNC), 2013 IEEE*, Dec 2013, pp. 24–30.

[10] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," http://www.ietf.org/rfc/rfc4838.txt, 2007.

[11] Contiki, "Contiki Operating System," http://www.contiki-os.org/, 2012.

[12] E. M. Daly and M. Haahr, "The challenges of disconnected delay-tolerant MANETs," *Ad Hoc Networks*, vol. 8, no. 2, pp. 241–250, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870509000791

[13] M. G. Dipankar Raychaudhuri, Ed., *Emerging Wireless Technologies and the Future Mobile Internet*. Cambridge University Press, 2011.

[14] M. Domingo, "A context-aware service architecture for the integration of body sensor networks and social networks through the IP multimedia subsystem," *IEEE Communications Magazine*, vol. 49, no. 1, pp. 102–108, january 2011.

[15] DTNRG, "Delay tolerant networking research group," http://www.dtnrg.org, 2012.

[16] S. Duquennoy, F. Österlind, and A. Dunkels, "Lossy links, low power, high throughput," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11. New York, NY, USA: ACM, 2011, pp. 12–25. [Online]. Available: http://doi.acm.org/10.1145/2070942.2070945

[17] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 Conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/863955.863960

[18] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 16:1–16:49, Dec. 2013.

[19] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: ACM, 2009, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644040

[20] ——, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. ACM, 2009, pp. 1–14.

[21] S. Grasic, E. Davies, A. Lindgren, and A. Doria, "The evolution of a DTN routing protocol - PRoPHETv2," in *Proceedings of the 6th ACM Workshop on Challenged Networks*, ser. CHANTS '11. New York, NY, USA: ACM, 2011, pp. 27–30. [Online]. Available: http://doi.acm.org/10.1145/2030652.2030661

[22] X. Guan, M. Chen, and T. Ohtsuki, "Epidemic theory based H + 1 hop forwarding for intermittently connected mobile ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, no. 1, pp. 1–10, 2012. [Online]. Available: http://dx.doi.org/10.1186/1687-1499-2012-76

[23] V. Gungor and G. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 4258–4265, oct. 2009.

[24] L. Guo, R. Beyah, and Y. Li, "SMITE: A stochastic compressive data collection protocol for mobile wireless sensor networks," in *Proceedings of IEEE INFOCOM, 2011*, 2011, pp. 1611–1619.

[25] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proceedings of IEEE Wireless Communications and Networking Conference, 2006. (WCNC 2006)*, vol. 1, april 2006, pp. 228–235.

[26] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 180–191. [Online]. Available: http://doi.acm.org/10.1145/1098918.1098938

[27] O. D. Incel, L. van Hoesel, P. Jansen, and P. Havinga, "MC-LMAC: A multi-channel MAC protocol for wireless sensor networks," *Ad Hoc Networks*, vol. 9, no. 1, pp. 73–94, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870510000624

[28] G. Isbitiren and O. Akan, "Three-dimensional underwater target tracking with acoustic sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 8, pp. 3897–3906, 2011.

[29] Y. Iyer, S. Gandham, and S. Venkatesan, "STCP: a generic transport layer protocol for wireless sensor networks," in *Proceedings of the 14th International Conference on Computer Communications and Networks, 2005. (ICCCN 2005)*, October 2005, pp. 449–454.

[30] B. Ji, C. Joo, and N. Shroff, "Exploring the inefficiency and instability of back-pressure algorithms," in *Proceedings of IEEE INFOCOM, 2013*, 2013, pp. 1528–1536.

[31] E. Jones, L. Li, J. Schmidtke, and P. Ward, "Practical routing in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 943–959, Auguest 2007.

[32] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," *SIGPLAN Notices*, vol. 37, pp. 96–107, October 2002. [Online]. Available: http://doi.acm.org/10.1145/605432.605408

[33] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, and I. Stoica, "Flush: a reliable bulk transport protocol for multihop wireless networks," in *Proceedings of the 5th international conference on embedded networked sensor systems*, ser. SenSys '07. New York, NY, USA: ACM, 2007, pp. 351–365. [Online]. Available: http://doi.acm.org/10.1145/1322263.1322296

[34] P. Levis and D. Gay, *TinyOS Programming*. Cambridge University Press, 2009.

[35] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 126–137. [Online]. Available: http://doi.acm.org/10.1145/958491.958506

[36] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks," in *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, ser. NSDI'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2.

[37] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani, "Block-switched networks: a new paradigm for wireless transport," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*. Berkeley, CA, USA: USENIX Association, 2009, pp. 423–436. [Online]. Available: http://dl.acm.org/citation.cfm?id=1558977.1559006

[38] ——, "Block-switched networks: a new paradigm for wireless transport," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, April 2009, pp. 423–436.

[39] M. Li and Y. Liu, "Underground structure monitoring with wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, ser. IPSN '07. ACM, 2007, pp. 69–78. [Online]. Available: http://doi.acm.org/10.1145/1236360.1236370

[40] Y. Li and R. Bartos, "Energy efficient reactive store-and-forward protocol for intermittently connected networks," in *Global Telecommunications Conference (GLOBECOM 2013), 2013 IEEE*, 2013.

[41] ——, "A survey of protocols for intermittently connected delay-tolerant wireless sensor networks," *Journal of Network and Computer Applications*, vol. 41, pp. 411 – 423, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804513002087

[42] Y. Li, R. Bartos, and J. Swan, "Transport protocol with acknowledgement-assisted storage management for intermittently connected wireless sensor networks," in *Distributed Computing and Networking*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7730, pp. 57–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35668-1_5

[43] ——, "Dacksis: An efficient transport protocol with acknowledgment-assisted storage management for intermittently connected wireless sensor networks," *Pervasive and Mobile Computing*, vol. 13, no. 0, pp. 272 – 285, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574119214000522

[44] Y. Li and W. Wang, "Horizon on the move: Geocast in intermittently connected vehicular ad hoc networks," in *Proceedings of IEEE INFOCOM, 2013*, 2013, pp. 2553–2561.

[45] W. Liao and C. Huang, "SF-MAC: A spatially fair MAC protocol for underwater acoustic sensor networks," *Sensors Journal, IEEE*, vol. 12, no. 6, pp. 1686–1694, 2012.

[46] C.-M. Lien, S.-Y. Lee, T.-Y. Ho, D.-N. Yang, and Y.-W. Hong, "Information dissemination with epidemic routing in energy harvesting wireless sensor networks," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 2815–2821.

[47] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, July 2003. [Online]. Available: http://doi.acm.org/10.1145/961268.961272

[48] J. Liu, X. Jiang, H. Nishiyama, and N. Kato, "Multicast capacity, delay and delay jitter in intermittently connected mobile networks," in *Proceedings of IEEE INFOCOM, 2012*, 2012, pp. 253–261.

[49] S. Nelson, M. Bakht, and R. Kravets, "Encounter-based routing in DTNs," in *INFOCOM 2009, IEEE*, April 2009, pp. 846–854.

[50] F. Österlind, "A sensor network simulator for the Contiki OS," Swedish Institute of Computer Science, SICS Report, 2009.

[51] J. Paek and R. Govindan, "RCRT: rate-controlled reliable transport for wireless sensor networks," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, ser. SenSys '07. New York, NY, USA: ACM, 2007, pp. 305–319. [Online]. Available: http://doi.acm.org/10.1145/1322263.1322293

[52] ——, "RCRT: Rate-controlled reliable transport protocol for wireless sensor networks," *ACM Transactions on Senensor Networks*, vol. 7, pp. 20:1–20:45, October 2010. [Online]. Available: http://doi.acm.org/10.1145/1807048.1807049

[53] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 134–141, 2006.

[54] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 95–107. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031508

[55] J. Polastre, R. Szewczyk, C. Sharp, and D. Culler., "The mote revolution: Low power wireless sensor network devices," in *IEEE HotChips 16*, Auguest 2004.

[56] A. T. Prodhan, R. Das, H. Kabir, and G. C. Shoja, "TTL based routing in opportunistic networks," *Journal of Network and Computer Applications*, vol. 34, pp. 1660–1670, September 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jnca.2011.05.005

[57] M. Quwaider, J. Rao, and S. Biswas, "Body-posture-based dynamic link power control in wearable sensor networks," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 134–142, July 2010.

[58] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," *Wireless Networks*, vol. 12, pp. 63–78, February 2006. [Online]. Available: http://dx.doi.org/10.1007/s11276-006-6151-z

[59] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu, "Z-MAC: a hybrid MAC for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 511–524, June 2008. [Online]. Available: http://dx.doi.org/10.1109/TNET.2007.900704

[60] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '06. New York, NY, USA: ACM, 2006, pp. 190–201. [Online]. Available: http://doi.acm.org/10.1145/1132905.1132927

[61] P. Salvo, M. De Felice, A. Baiocchi, F. Cuomo, and I. Rubin, "Timer-based distributed dissemination protocols for VANETs and their interaction with MAC layer," in *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*, June 2013, pp. 1–6.

[62] H. Seferoglu and E. Modiano, "Diff-Max: Separation of routing and scheduling in backpressure-based wireless networks," in *Proceedings of IEEE INFOCOM, 2013*, 2013, pp. 1555–1563.

[63] J. M. Soares and R. M. Rocha, "CHARON: Routing in low-density opportunistic wireless sensor networks," in *Proceedings of Wireless Days (WD), 2009 2nd IFIP*, dec. 2009, pp. 1–5.

[64] J. M. Soares, M. Franceschinis, R. M. Rocha, W. Zhang, and M. A. Spirito, "Opportunistic data collection in sparse wireless sensor networks," *EURASIP Journal on Wireless Communications Networking*, vol. 2011, pp. 6:1–6:20, January 2011. [Online]. Available: http://dx.doi.org/10.1155/2011/401802

[65] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 63–76, feb. 2008.

[66] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[67] ——, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[68] ——, "Spray and Focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2007. (PerCom Workshops '07)*, March 2007, pp. 79–85.

[69] ——, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 77–90, February 2008. [Online]. Available: http://dx.doi.org/10.1109/TNET.2007.897964

[70] N. Tezcan and W. Wang, "ART: an asymmetric and reliable transport mechanism for wireless sensor networks," *International Journal of Sensor Network*, vol. 2, pp. 188–200, April 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1359004.1359009

[71] N. Thompson, S. C. Nelson, M. Bakht, T. Abdelzaher, and R. Kravets, "Retiring replicants: congestion control for intermittently-connected networks," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1118–1126. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833686

[72] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Communications Magazine*, vol. 43, no. 3, pp. S27–S32, March 2005.

[73] O. Tonguz, N. Wisitpongphan, and F. Bai, "DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks," *Wireless Communications, IEEE*, vol. 17, no. 2, pp. 47–57, April 2010.

[74] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wirel. Netw.*, vol. 8, no. 2/3, pp. 153–167, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013763825347

[75] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Department of Computer Science, Duke University, Tech. Rep. Technical Report CS-2000-06, April 2000.

[76] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 171–180. [Online]. Available: http://doi.acm.org/10.1145/958491.958512

[77] M. Vuran and I. Akyildiz, "XLP: A cross-layer protocol for efficient communication in wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 11, pp. 1578–1591, November 2010.

[78] P. Wang, Z. Sun, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz, "On network connectivity of wireless sensor networks for sandstorm monitoring," *Computer Networks*, vol. 55, no. 5, pp. 1150–1157, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610003592

[79] S. Wang, X. Wang, X. Cheng, J. Huang, and R. Bie, "The tempo-spatial information dissemination properties of mobile opportunistic networks with Levy Mobility," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, June 2014, pp. 124–133.

[80] X. Wang, T. Yu, and Y. Xu, "Lower bound for node buffer size in intermittently connected wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 754–766, 2013.

[81] Y. Wang and H. Wu, "Delay/fault-tolerant mobile sensor network (DFT-MSN): A new paradigm for pervasive information gathering," *IEEE Transactions on Mobile Computing*, vol. 6, no. 9, pp. 1021–1034, September 2007.

[82] N. Wisitpongphan, O. Tonguz, J. Parikh, P. Mudalige, F. Bai, and V. Sadekar, "Broadcast storm mitigation techniques in vehicular ad hoc networks," *Wireless Communications, IEEE*, vol. 14, no. 6, pp. 84–94, December 2007.

[83] M. Xiao, J. Wu, C. Liu, and L. Huang, "Tour: Time-sensitive opportunistic utility-based routing in delay tolerant networks tour: Time-sensitive opportunistic utility-based routing in delay tolerant networks TOUR: Time-sensitive opportunistic utility-based routing in delay tolerant networks," in *INFOCOM 2013, IEEE*, 2013.

[84] Y. Xu and X. Wang, "Fundamental lower bound for node buffer size in intermittently connected wireless networks," in *Proceedings of IEEE INFOCOM, 2011*, 2011, pp. 972–980.

[85] J. Yang, P. Guo, T. Jiang, and K. Zhang, "SRCR: A novel MAC protocol for underwater acoustic networks with concurrent reservation," in *Communications (ICC), 2012 IEEE International Conference on*, 2012, pp. 435–439.

[86] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOM 2002)*, vol. 3, 2002, pp. 1567–1576.

[87] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRG-4S8TBBT-1/2/b242d2fd1f6d2cf5c6fce0a24c4cb029

[88] J. Zhang, V. Gauthier, H. Labiod, A. Banerjee, and H. Afifi, "Information dissemination in vehicular networks via evolutionary game theory," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 124–129.

[89] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *IEEE Communications Surveys Tutorials*, vol. 8, no. 1, pp. 24–37, quarter 2006.

[90] S. Zhou and P. Willett, "Submarine location estimation via a network of detection-only sensors," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 3104–3115, june 2007.