Performance of Data Aggregation for Wireless Sensor Networks

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Jie Feng

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

This thesis focuses on three fundamental issues that concern data aggregation protocols for periodic data collection in sensor networks: *which* sensor nodes should report their data, *when* should they report it, and should they use *unicast* or *broadcast* based protocols for this purpose.

The issue of *when* nodes should report their data is considered in the context of real-time monitoring applications. Such applications can require high sampling rates and low-delay forwarding of the sensor values to a sink node at which the data is to be further processed. Since aggregation requires that some sensor data be delayed at intermediate nodes, however, while waiting for other data to be received, a key issue in the context of real-time monitoring is how to achieve effective aggregation with minimal forwarding delay. Previous work has advocated synchronous aggregation, in which a node's position in the aggregation tree determines when it transmits to its parent. The first part of this thesis shows that asynchronous aggregation, in which the time of each node's transmission is determined adaptively based on its local history of past packet receptions from its children, outperforms synchronous aggregation by providing lower delay for a given end-to-end loss rate.

Second, new broadcast-based aggregation protocols are designed and evaluated. They minimize the number of packet transmissions, relying on multipath delivery rather than automatic repeat request for reliability. The performance of broadcast-based aggregation is compared to that of unicast-based aggregation, in the context of both real-time and delay-tolerant data collection. For real-time applications, this work investigates whether such protocols can achieve lower collection delays and support higher sampling rates than conventional aggregation protocols, while performance evaluation for delay-tolerant data collection is focused on reliability. The results suggest that when packet loss is random, broadcast-based protocols can yield significantly improved performance in some real-time data collection scenarios, specifically when sensor data can be aggregated into packets of size that is independent (or largely independent) of the number of values being aggregated.

Broadcast-based aggregation can yield significantly better performance than unicast-based aggregation for both real-time and delay-tolerant data collection when packet loss follows a two-state Gilbert error model.

Finally, in the context of applications in which coverage of some monitored region is to be maintained, this thesis investigates the potential benefits of dynamically, rather than semi-statically, determining the set of nodes reporting their data. In such applications, sensor nodes are often deployed more densely than would minimally be required. With a semi-static approach, node scheduling protocols are deployed to reduce energy consumption and prolong network lifetime by putting redundant nodes to sleep. Node scheduling approaches, however, may leave part of the monitored region uncovered if node failures happen and a replacement node is not woken up immediately. Unicast and broadcast-based coverage-preserving data aggregation protocols in which nodes dynamically determine during each round of data collection whether they should transmit their data, or whether the set of neighbouring nodes that have already transmitted is sufficient to provide coverage, are designed and evaluated. The performance of the proposed protocols is compared to that of data collection protocols relying on node scheduling. Results suggest that the proposed broadcast-based protocol can greatly improve reliability, at the potential cost of increased traffic volume owing to non-minimal selection of transmitting nodes. For real-time data collection that is willing to trade reliability for improved data collection delay, broadcast-based data aggregation with node scheduling is able to achieve lower delay with moderate loss of coverage.

Acknowledgements

I would like to thank the many people who made this thesis possible.

First of all, I would like to extend my deep and sincere gratitude to my supervisors, Dr. Derek Eager and Dr. Dwight Makaroff, for their guidance and support through out my Ph.D program. This thesis would not have been possible without their enlightening instruction, encouragement, patience, and knowledge.

I wish to thank the wonderful office staff members of my department. I am especially grateful to Jan Thompson, our graduate correspondent, for assisting me in many different ways. I'd also like to thank Brian Gallaway and Greg Oster for their professional and timely technical support.

I would like to thank my committee members, Dr. Thomas Kunz (external), Dr. Brian Daku (cognate), Dr. Grant Cheston, Dr. Nathaniel Osgood, and Dr. Raymond Spiteri, for their valuable comments on my thesis. The comments really helps to improve the quality of my thesis. Thank Dr. Kevin Stanley, a professor in our research group, for providing feedbacks on my thesis.

Special thanks to my pet cats, huhu (little tiger) and xixi (Sisi). Their accompany brings enormous joy to my daily life.

Finally, my deepest appreciation to my mother, my husband, my sister, and my brother. Their love and support enabled me to complete this work. My mom always says that a person can go anywhere in the world without fear if he/she masters math, physics, and chemistry. I wonder whether she would still encourage me to attend university in Beijing if she knew that I would go so far away from her. Thank my sister and my brother for taking care of our mom for me.

Contents

List of Tables

List of Figures

List of Acronyms

ACK	ACKnowledgement
ADDA	Application Dependent Data Aggregation
AID	Aggregation ID
AIDA	Application Independent Data Aggregation
AIMD	Additive Increase Multiplicative Decrease
ARQ	Automatic Repeat reQuest
CCP	Coverage Configuration Protocol
CNS	Center at Nearest Source
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	arrier Sense Multiple Access with Collision Detection
CTS	Clear to Send
DAA	Data-Aware Anycast
DAG	Directed Acyclic Graph
DCTC	Dynamic Convoy Tree-based Collaboration
DSFC	Delay Sensitive Feedback Control
EPAS	Energy-efficient Protocol for Aggregator Selection
EPCR	Emergent Packet Channel Reservation
EWMA	Exponentially Weighted Moving Average
GIT	Greedy Incremental Tree
HEED	Hybrid Energy-Efficient Distributed clustering
LDAS	Lightweight Deployment-Aware Scheduling
LEACH	Low-Energy Adaptive Clustering Hierarchy
LEACH-C	LEACH-Centralized
MAC	Medium Access Control
MCS	Minimum Coverage Set
MFS	Multi-level Fusion Synchronization
NP-hard	Non-Deterministic Polynomial-Time Hard
OGDC	Optimal Geographic Density Control
PEAS	Probing Environment and Adaptive Sensing
PEGASIS	Power Efficient data GAthering protocol for Sensor Information Systems
QoS	Quality of Service
RTS	Request to Send
RW	Randomized Waiting
SPT	Shortest Path Tree
TAG	Tiny AGgregation
TCP	Transmission Control Protocol
ToD	Tree on Directed acyclic graph
UCS	Unequal Clustering Size
WSN	Wireless Sensor Network

Glossary

802.11

A family of standards developed by the Institute of Electrical and Electronics Engineers (IEEE) for wireless local area networks (WLANs).

Directed acyclic graph

A directed acyclic graph is a directed graph that has no directed cycles.

Duplicate insensitive aggregation function

An aggregation function is duplicate insensitive if and only if multiple occurrences of the same value when input into the function do not affect the result.

Duplicate sensitive aggregation function

An aggregation function is duplicate sensitive if multiple occurrences of the same value when input into the function produce different aggregation results.

Medium access control

In networks using broadcast channels, a key issue is how to determine who gets to transmit when multiple nodes compete for the channel. This issue is handled by a sublayer of the data link layer called medium access control (MAC) sublayer.

Steiner tree

Given an edge-weighted graph G = (V, E, w) and a subset S $\subseteq$ V of required vertices, a Steiner tree is a subtree of G that includes all vertices of S.

Minimum Steiner tree

Given an edge-weighted graph G = (V, E, w) and a subset S $\subseteq$ V of required vertices, a Steiner tree is called the minimum Steiner tree when the sum of the weights of the edges in the subtree is minimum among all possible Steiner trees.

Network simulator version 2

Network simulator version 2 (ns2) is a discrete event simulator for networking

research. It is a free open source simulator.

Quantile

Quantiles are values that divide a set of ordered data into equal proportions; examples are median, quartile, and decile.

Chapter 1

Introduction

Wireless sensor networks have caught the attention of the research community in recent years. These networks are composed of small sensor nodes that integrate sensing, wireless communication, and computation. Each sensor node has limited processing capability, storage capacity, and communication bandwidth. Unlike the Internet, sensor networks are generally application specific, in which multiple nodes cooperate to fulfill a common task.

As each individual sensor node is inherently resource-constrained, sensor networks depend on the coordination of a large number of nodes to carry out their tasks [21]. In many cases, such as wild animal tracking, the nodes are deployed in areas without external power supply or an infrastructure for communication, or they may even be deployed in inhospitable physical environments where human operation is difficult. A typical way to deploy a sensor network in a harsh environment would be dropping the sensor nodes from a plane [21, 87]. The nodes rely on their limited resources to survive. Meanwhile, to keep down the cost of network deployment and maintenance, the sensor nodes must be able to organize themselves and work unattended because it is impractical to configure each node manually. Since communication is a major source of energy consumption, sensor nodes usually self-organize into a multi-hop wireless network to avoid long-range communication [2].

A broad class of applications is possible with wireless sensor networks. Potential applications include habitat and environmental monitoring, detection and prediction [5, 12, 43, 48, 63, 77, 83, 92], target tracking [3, 34, 68, 80, 81, 94], infrastructure monitoring [28, 50, 71, 90], health applications [36, 45, 79], structural health monitoring [8, 49, 103], and military applications [35, 84]. As sensor networks can work

unattended with low deployment and maintenance cost, they enable scientists to conduct research in areas that could not previously be reached.

While sensor networks offer significant advantages for many scientific and commercial applications, the constraints on sensor nodes raise new challenges. The most important constraint on sensor nodes is their limited power supply. Sensor nodes are usually battery powered so they can operate in environments without any infrastructure. It is impractical to locate each node and replace its battery when it runs out of energy, especially when the network is deployed in wild environments. Power conservation has been well recognized by the sensor network community as a critical factor to prolong the network lifetime. Other constraints on sensor nodes include low computation and storage capability, limited wireless communication range, and susceptibility to physical damage.

The design of sensor networks is influenced by many factors. Akyildiz *et al.* give an outline of some important design factors in their survey on sensor networks [2]:

- **Fault tolerant:** sensor nodes may run out of battery power, or may be blocked or physically damaged. Fault tolerance requires that the functionalities of the network not be interrupted due to node failures.

- **Scalability:** A sensor network may contain hundreds or thousands of sensor nodes. Protocols for sensor networks require good scalability.

- **Production cost:** Since sensor networks may require the coordination of a huge number of nodes, the cost of each node should be kept low to make deployment feasible.

- **Hardware constraints:** Hardware design of sensor nodes are constrained by many factors, including size (the node may need to fit into a matchbox-sized module), low energy consumption, low cost, and the ability to work unattended.

- **Sensor network topology:** To deploy a large number of nodes densely, sensor nodes can be either thrown out in a mass, or placed one by one by humans or robots. Network topology changes can occur due to node movement (in the

case of mobile sensor nodes), temporary or permanent node failures. Additional nodes may be deployed to replace malfunctioning nodes.

- **Environment:** Sensor networks may be deployed and work unattended in harsh, inhospitable environments without any infrastructure for communication or power supply.

- **Transmission media:** Nodes in sensor networks communicate with each other over wireless media.

- **Power consumption:** Sensor nodes are equipped with a limited power source. Replenishment of power sources is impossible in many scenarios. Power management and power conservation are important to prolong network lifetime.

A huge amount of work has been done on sensor networks. Routing schemes for sensor networks have been developed and evaluated [7, 30, 38, 40, 46, 53, 108]. Different reliable transport protocols have been proposed, as well as congestion control strategies [18, 44, 70, 98, 97]. Supporting technologies such as time synchronization and quality of service (QoS) have been also extensively studied [19, 25, 29, 51, 57, 58]. Efforts have also been made to study link quality in sensor networks [6, 89, 114].

Different technologies have also been proposed to reduce energy consumption in sensor networks. Data aggregation reduces energy consumption by combining data from different nodes to suppress redundancy and traffic volume [26, 27, 30, 37, 61, 86]. Node scheduling saves energy by putting nodes to sleep when they are not required to participate in the sensing and communication tasks [54, 93, 99, 104]. Energy efficient Medium Access Control (MAC) protocols have also been proposed for sensor networks, which save energy by cycling nodes between active and sleep states [60, 76, 91, 96, 105].

Aggregation protocols for different applications, such as monitoring and periodic data collection [61, 69, 86], dynamic event detection[23], and target tracking[112], have been proposed. This dissertation focuses on data aggregation approaches for periodic data collection in sensor networks that monitor an area of interest. These

networks require that samples be collected periodically and transmitted back to a sink node where data is to be gathered and processed. As the network is usually densely deployed, data produced by different nodes may be redundant. On the other hand, in many cases, it is the summaries, instead of the raw data, that are valuable to the researchers. Network traffic and energy consumption can be reduced by aggregating data from various nodes as it is being forwarded to the sink.

The performance of data aggregation protocols can be measured by different metrics, such as energy cost, data forwarding delay, and end-to-end loss rate. Real-time monitoring applications aim to retain a "current" view of the monitored area, and require effective aggregation with minimal forwarding delay. For example, in battlefield surveillance, sensor nodes are deployed to detect and track moving military targets such as tanks and other vehicles [9]. In this case, the data collected from the sensor network should remain as current as possible. In general, low data forwarding delay is important in any application requiring response to external phenomenon in real-time [1]. Similarly, 100% reliable delivery from each sensor is an important goal in some applications, while others can tolerate moderate levels of loss. Furthermore, some applications require data from only a subset of sensors covering some region of interest.

Timing plays a key role in effective aggregation, since aggregation requires that some sensor data be delayed at intermediate nodes, while waiting for other data to be received. Existing aggregation protocols can be classified as synchronous or asynchronous based on their timing control strategies. New asynchronous aggregation protocols are proposed in this work, the performance of which is evaluated and compared to that of synchronous aggregation protocols. The results provide useful insight for real-time monitoring applications.

Both unicast and broadcast transmission can be used for data collection in sensor networks [32, 61, 86]. A unicast packet is received and processed by the intended recipient only. Unicast transmission relies on Automatic Repeat reQuest (ARQ) for reliability, but packet recovery may fail when transient/persistent link failure happens. Broadcast-based data collection takes advantage of the inherent redundancy

of the broadcast medium in sensor networks for reliability [32, 66]. When a packet is transmitted using broadcasting, all nodes within the transmission range of the sender receive the packet if there are no link errors. One problem with broadcast-based aggregation, however, is that when the same packet is received and aggregated by multiple nodes, incorrect results may be produced for duplicate-sensitive aggregation functions. For example, "count" is a duplicate-sensitive aggregation function. The same sensor reading should be only counted once. Both unicast-based and broadcast-based aggregation protocols have been proposed. Previous work regarding broadcast-based aggregation has been focused on broadcast-based aggregation protocol design for duplicate-sensitive aggregation functions [32, 61, 66, 67]. This work proposes new broadcast-based aggregation protocols and compares them to unicast-based aggregation in terms of data forwarding delay, end-to-end loss rate, and traffic volume.

Sensor networks are often more densely deployed than would minimally be required to improve network robustness against node failures. To improve energy efficiency and prolong network lifetime, node scheduling protocols can be deployed to put redundant nodes to sleep. Such node scheduling based approaches, however, may leave some of the monitored area unattended when a node fails and other nodes are not woken up to replace it right away. Instead of putting redundant nodes to sleep and performing aggregation over the working nodes, an alternative approach is to keep all nodes awake and let each node dynamically determine for each round whether cancelling its transmission would have any impact on area coverage. The third part of this thesis investigates unicast-based and broadcast-based coverage preserving data aggregation protocols and compares their performance to that of unicast and broadcast-based aggregation relying on node scheduling.

1.1 Research Scope

This dissertation studies data aggregation protocols for static wireless sensor networks that require data to be collected periodically by the nodes. Furthermore,

aggregation of sequences of sensor values from the same node (temporal aggregation) is not considered. Finally, sensor networks in which paths to the sink node are multiple hops are assumed. Data aggregation in flat networks is out of the scope of this dissertation.

1.2 Research Goals

The main goal of this dissertation is to study the performance of different data aggregation protocols for periodic data collection in sensor networks. Three fundamental issues are addressed. The first issue deals with timing control, i.e., when should the nodes transmit. The second issue concerns how packets are transmitted. Aggregation protocols can use unicast or broadcast transmission. The third issue deals with which nodes should transmit. For dense networks, it may not be necessary to collect samplings from all nodes to meet the service requirement of the applications.

- Timing control has a great impact on aggregation efficiency and data forwarding delay. Existing aggregation protocols can be classified as synchronous or asynchronous, based on their timing strategies. This dissertation proposes asynchronous aggregation protocols that use more aggressive methods for determining when a node should transmit to its parent. The performance of the proposed protocols is evaluated and compared to that of synchronous aggregation.

- The second part of this dissertation examines broadcast-based aggregation protocols that minimize the number of packet transmissions, relying on multipath delivery rather than ARQ for reliability, and compares them to unicast-based aggregation in the context of both real-time and delay-tolerant monitoring.

- For dense networks with a significant number of redundant nodes, node scheduling protocols can be deployed to put redundant nodes to sleep, and thus, aggregation is performed over the working nodes. An alternative approach is coverage preserving aggregation, in which each node dynamically determines

during each round of data collection whether its data should be transmitted, or whether its sensing area is covered by neighbouring nodes that have already transmitted. The third part of this dissertation examines the performance of coverage preserving aggregation and data aggregation based on node scheduling.

Synchronous vs. asynchronous aggregation

Aggregation protocols for periodic data collection sensor networks usually transmit over a certain structure, for example, a tree rooted at the sink [61, 62, 86]. Aggregation is performed as data is being forwarded to the sink at intermediate nodes. To achieve efficient data aggregation, intermediate nodes are required to delay their transmissions until packets routed through them have been received.

Timing is crucial for real-time data aggregation protocols which aim to achieve effective aggregation with minimal forwarding delay. These networks require a high sampling rate and a low delay in forwarding data to the sink so as to maintain a current "view" of the environment being monitored. For example, during a chemical leak, nodes with chemical sensors are deployed to collect real-time information about the location and concentration of hazards in the polluted area [65].

According to the timing strategies used, aggregation protocols can be classified as synchronous or asynchronous. Synchronous aggregation divides time into a sequence of time intervals for each data collection period and assigns a different time interval to a particular set of sensor nodes [61, 86]. The nodes are synchronized in the way that nodes along the path to the sink are assigned with successive intervals such that nodes farthest away from the sink transmit first.

In asynchronous aggregation, each node makes its own timing decision based on local packet reception history. As timing control can be classified as synchronous or asynchronous, a natural question is how these two approaches compare to each other in the context of real-time monitoring. Previous work has advocated synchronous aggregation [86]. However, it is observed that the timing control strategy of the asynchronous protocol used for comparison is quite straightforward and may result in

long delay [86]. In this work, improved asynchronous data aggregation protocols with more aggressive timing models are developed. A thorough performance comparison is conducted between synchronous and asynchronous aggregation protocols.

Unicast-based vs. Broadcast-based aggregation

Both broadcast and unicast-based aggregation protocols have been proposed for sensor networks [61, 62, 67, 86]. Nath and Bibbons propose synopsis diffusion, a broadcast-based aggregation protocol, for duplicate-sensitive aggregation functions in sensor networks [67]. Another two broadcast-based aggregation protocols proposed by Gobriel *et al.* [32] and Motegi *et al.* [66] also focus on how to support duplicate-sensitive aggregation when the same data is received by multiple nodes.

This dissertation examines new broadcast-based protocols that minimize the number of packet transmissions and rely on multipath delivery for reliability. It is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates including the same sensor value. For example, when only the maximum sensor value is needed at the sink, the same sensor value can be aggregated by multiple nodes without any impact on the final result. No particular aggregation function is assumed; instead, two extreme cases are considered. In one case, it is assumed that sensor data from different nodes can be aggregated into fixed-size packets. In the other, it is assumed that the required packet size increases linearly with the number of values included in the aggregate.

The performance of the proposed broadcast-based protocols is compared to that of unicast-based aggregation in the context of both real-time and delay-tolerant monitoring. For real-time scenarios, this work investigates whether broadcast-based protocols can be employed to maximize the achievable sampling/collection rate and minimize the collection delay with some modest amount of data loss. For both real-time and delay-tolerant applications, end-to-end loss rate and energy consumption are examined.

Aggregation with node scheduling vs. coverage preserving aggregation

To tolerate faults such as node failures or link failures, sensor nodes may be deployed much more densely than would minimally be required. This level of redundancy can lead to unnecessary congestion and battery depletion for nodes that may not need to transmit. Node scheduling protocols prolong network lifetime by putting redundant nodes to sleep [54, 93, 100, 104]. One key issue in node scheduling is how to pick nodes to achieve the desired area coverage.

Data aggregation can be performed over the active nodes selected by a node scheduling protocol. One potential drawback of node scheduling based approaches, however, is that area coverage may not be complete when node and/or communication failures happen and replacement node(s) is not woken up immediately. This work examines an alternative approach for efficiently maintaining area coverage in dense sensor networks. In the proposed approach, nodes dynamically determine during each round of data collection whether they should transmit their data, or whether the set of neighbouring nodes that have already transmitted is sufficient to provide coverage.

Both unicast-based and broadcast-based coverage preserving aggregation protocols are proposed, and their performance is compared to that of unicast and broadcast-based aggregation relying on node scheduling. The results provide useful insight for applications that aim to periodically collect at a sink node sensing data that covers the region of interest.

1.3 Contributions

This dissertation investigates protocol design and performance issues concerning periodic data aggregation for sensor networks. In particular, performance comparison between synchronous and asynchronous aggregation, broadcast and unicast-based aggregation, and coverage preserving aggregation and aggregation based on node scheduling is investigated. The main contributions of this work are as follows:

Synchronous vs. asynchronous aggregation

Aggregation protocols for periodic data collection can be categorized as synchronous and asynchronous according to the timing strategies used in the protocols. Previous work has shown that synchronous aggregation outperforms asynchronous aggregation [86]. In this work, improved asynchronous aggregation protocols are proposed, the performance of which is evaluated and compared to that of synchronous aggregation in the context of real-time monitoring. The results show that with more aggressive timing strategies, asynchronous aggregation outperforms synchronous aggregation by providing lower delay for a given end-to-end loss rate. Performance comparisons are also conducted among the asynchronous protocols. The results show that adaptation of timeout values based on a weighted average of history information from multiple rounds is preferable to adaptation based only on the immediately previous round. The results also show that randomizing the transmission times of leaf nodes to avoid congestion at the beginning of each round, and the duration of the randomization interval, have a great impact on delay and end-to-end loss rate. A method is proposed for adaptively determining the duration of the randomization interval.

Unicast-based vs. broadcast-based aggregation

New synchronous and asynchronous broadcast-based aggregation protocols are proposed and compared to unicast-based aggregation using simulation. The results suggest that for real-time data collection, broadcast-based protocols can yield significantly improved performance in some scenarios when packet loss is random, especially when sensor data can be aggregated into packets of size that is independent (or largely independent) of the number of values being aggregated. For the two-state Gilbert error model, broadcast-based aggregation yields significantly better performance than unicast-based aggregation for both real-time and delay-tolerant data collection. Preliminary experimental results from a real sensor network are also presented.

Aggregation with node scheduling vs. coverage preserving aggregation

Rather than putting redundant nodes to sleep, this work investigates aggregation approaches that integrate aggregation with coverage preservation. In coverage preserving aggregation, nodes dynamically determine during each round of data collection whether they should transmit their data, or whether neighbouring nodes that have already transmitted are sufficient to provide the desired coverage. Both unicast and broadcast-based coverage preserving data aggregation protocols are designed, and their performance is compared using simulation to that of unicast and broadcast-based data aggregation protocols relying on node scheduling. The results suggest that the proposed broadcast-based coverage preserving protocol can greatly improve reliability (yielding an order of magnitude reduction in uncovered area, in some cases), at the potential cost of increased traffic volume/energy cost owing to non-minimal selection of transmitting nodes. For real-time data collection that is willing to trade reliability for reduced data collection delay, broadcast-based aggregation with node scheduling is able to achieve lower delay with moderate end-to-end loss in most scenarios.

1.4 Thesis Organization

The remainder of this dissertation is organized as follows. Chapter 2 reviews prior work on data aggregation. Chapter 3 proposes new asynchronous aggregation protocols and presents a thorough performance comparison between synchronous and asynchronous aggregation. Chapter 4 considers data aggregation based on broadcast, and gives a performance comparison between unicast and broadcast-based aggregation. Chapter 5 presents new coverage preserving aggregation protocols, and compares the performance of the proposed protocols with that of aggregation protocols relying on node scheduling. Chapter 6 summarizes the dissertation and outlines future directions.

Chapter 2

Background

2.1 Wireless Sensor Networks

A wireless sensor network is a network of small sensor nodes that are capable of computation, wireless communication, and sensing. Raghunathan *et al.* identify four subsystems in a sensor node: a computing subsystem, a communication subsystem, a sensing subsystem, and a power supply subsystem [75]. The computing subsystem provides intelligence to the sensor node with a microprocessor or microcontroller. In addition to the microcontroller unit, a typical sensor node is equipped with volatile and non-volatile storage, and I/O capabilities to support sensors [39]. The communication subsystem consists of a short-range radio for wireless communication. The sensing subsystem uses one or more sensors to take measurements of properties of the physical environment. The power supply subsystem provides power to the rest of the node.

Fig. 2.1 shows a picture of a MICAz mote from the Crossbow company[1]. The mote is about the size of a matchbox and is powered by two AA batteries.

Networking is a central component of sensor networks [39]. For sensor networks, the communication protocol stack consists of the physical layer, data link layer, network layer, transport layer, and application layer [2, 106]. The physical layer is concerned with transmitting bit streams over the communication medium. The data link layer is concerned with the data transmissions between two nodes. The design of these two layers is influenced by the fact that wireless medium is broadcast in nature.

[1]Crossbow: http://www.xbow.com

Figure 2.1: MICAz Mote

A packet transmission is received by all nodes that are within the transmission range of the sender, when there are no link errors. When the sender wants to send a message to a specific node, the message is transmitted using *unicast*. A unicast packet carries the address of the destination node. All neighbors of the sender, except the destination node, discard the packet. A *broadcast* packet, on the other hand, carries a special broadcast address. A broadcast packet is processed by all nodes that receive it. This addressing issue is handled by the MAC sublayer of the data link layer. All packets received at the physical layer are forwarded to the MAC layer. Packets destined to nodes other than the receiver itself are discarded by the MAC layer.

In wireless networks, a collision happens when two or more nodes, within the range of some receiving node, transmit at the same time, and the packets are corrupted at the receiving node by the overlapping signals. The MAC layer coordinates the access of the nodes to the shared channel to avoid collisions. A huge number of MAC protocols have been proposed for sensor networks [60, 91, 105]. An important class of these MAC protocols is the Carrier Sense Multiple Access (CSMA) protocols. In CSMA protocols, a node listens to the medium when it is ready to transmit data;

this is called *carrier sensing*. The node is allowed to transmit if the medium is found idle. If the medium is found busy, the node defers its transmission to a later time.

CSMA protocols cannot eliminate packet collisions. When nodes inside the network are able to listen while sending, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) can be used to improve CSMA performance [55]. To reduce the cost of a collision, the node stops transmitting if it detects another signal during its transmission. In wireless networks, however, it is not possible to listen while transmitting. Therefore, it is impossible to implement collision detection. In wireless communication, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is used as an improvement of pure CSMA. The RTS/CTS (Request to Send/Clear to Send) scheme is one of the techniques for collision avoidance. Before transmitting its data, the sender broadcasts an RTS control message and the destination node responds with a CTS. The data transmission is started after the RTS/CTS handshake. All other nodes that heard the RTS or CTS refrain from transmitting to avoid packet collision [4, 55].

Wireless communication is well known for its unpredictable performance [114]. Packet losses due to link errors are common in sensor networks. In unicast transmissions, ARQ can be implemented at the MAC layer to improve data transmission reliability. When a data packet is successfully received, an ACK (ACKnowledgement) is sent by the receiver to inform the sender about the reception. If the sender does not receive any acknowledge after a certain period of time, the sender retransmits the data packet until an acknowledgment is received or a predefined number of retransmissions has been reached. Note that usually acknowledgements are not used in broadcast transmissions.

Two error models are commonly used to simulate packet loss in wireless networks, the independent random error model and the two-state Gilbert error model. The first error model simulates packet losses that are independent and uncorrelated. However, measured data from actual communication links indicates that link errors often occur in isolated bursts [56, 17]. The two-state Gilbert error model simulates bursts of errors over communication links. The model is based on the use of a

Markov chain with two states: the good state and the bad state. In the good state, transmission is error-free. In the bad state, the link only has a certain probability of transmitting a packet/bit correctly [56, 17]. State transition happens after the node stays in one state for a certain amount of time.

In multi-hop sensor networks, the network layer is responsible for routing the packets from the sources to the destinations. Different routing protocols have been proposed for sensor networks [40, 53, 108]. The transport layer aims to provide reliable and cost-effective packet delivery from the source to the destination. Congestion control is an important part of the transport layer [44, 97]. When the nodes offer more load to the network than it can handle, congestion builds up and the quality of service of the network deteriorates dramatically. The application layer contains support for the specific application of the sensor network and protocols that are commonly needed. Akyildiz *et al.* examine three possible application layer protocols: sensor management protocol, task assignment and data advertisement protocol, and sensor query and data dissemination protocol [2].

Sensor nodes are usually powered by batteries. Power consumption is a big issue in sensor networks. Previous research has shown that communication is considerably more energy costly than computation. Transmitting a single bit may cost hundreds or even thousands of times as much energy as executing a single instruction [47, 74]. Based on this basic observation, data aggregation has been proposed to improve energy efficiency in sensor networks [27, 37, 61, 86]. The main idea is to perform in-network processing to reduce communication cost.

2.2 Data Aggregation

Sensor networks are often densely deployed to cover the area of interest. Data produced by different neighbouring nodes may be highly correlated and redundant. For event detecting applications, the same event may be detected and reported by multiple nodes. For data collection applications, it is often the summaries (or aggregates), instead of the raw sensor readings, that are valuable to the researchers or

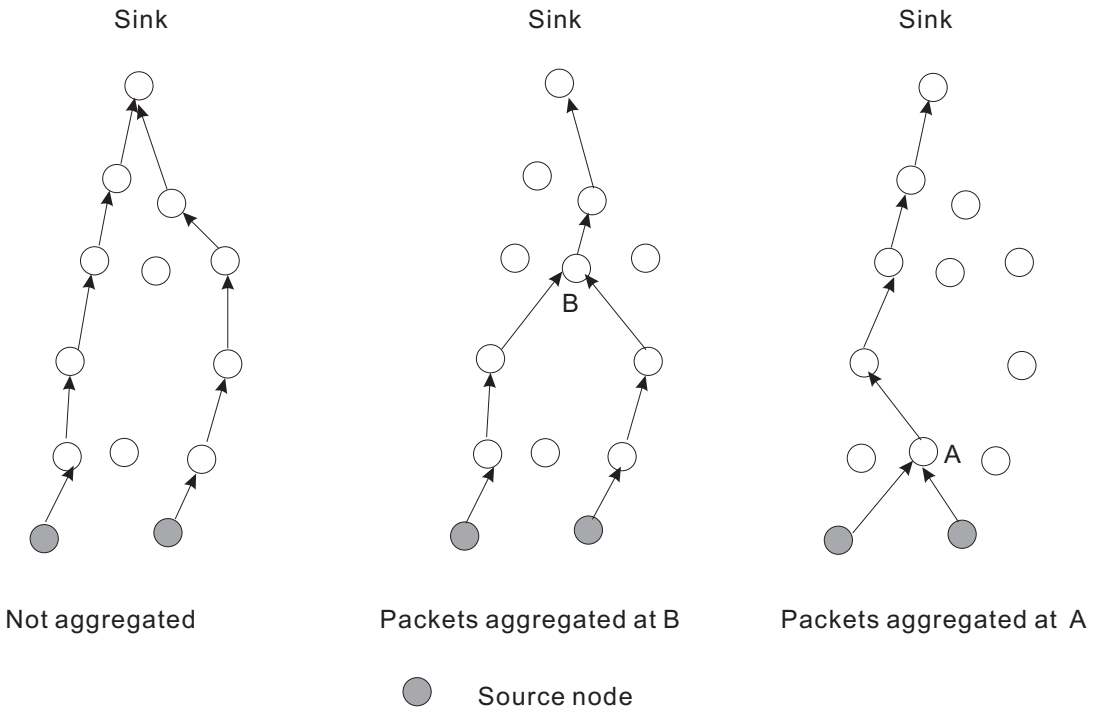


Figure 2.2: Impact of Routing on Data Aggregation

other users. As data transmission often uses more energy than data processing, energy consumption can be greatly improved by reducing the amount of transmissions through in-network processing or data aggregation [23].

Data aggregation has been extensively studied in the literature. Fasolo *et al.* identify three basic ingredients in data aggregation: *suitable networking protocols*, *effective aggregation functions*, and *efficient ways of representing the data* [24]. A well-designed routing protocol is the most important ingredient for data aggregation. Traditional routing protocols typically adopt the shortest-path routing method. In sensor networks, however, data centric routing, in which routing is done based on the content of the data packets, is often adopted to promote data aggregation. Fig. 2.2 illustrates how routing impacts data aggregation. In Fig. 2.2, it is assumed that the aggregation results of data from different nodes can be included in a single packet.

In data collection applications, such as environment and habitat monitoring, nodes collect samples periodically and transmit sensed data to a sink node [12, 63, 92]. Aggregation is performed as packets are being forwarded to the sink. Aggregation protocols for periodic data collection are usually based on network structures such

as clusters and trees. As the nodes that participate in the task stay relatively stable, the cost to construct and maintain the structure is low.

In event-based applications, source nodes change dynamically with the situation. Zhang *et al.* examine aggregation with a dynamically-configured tree for event-based applications [112, 113]. The overhead to build and maintain the tree structure, however, can be costly. A structure-free aggregation protocol is proposed by Fan *et al.* to avoid the overhead for the maintenance of the underlying structure [22].

Aggregation protocols based on multi-path routing have been proposed [32, 37, 61, 66, 67]. Considering that wireless medium is broadcast in nature, broadcast-based data aggregation protocols are proposed to exploit the inherent redundancy of the broadcast medium in sensor networks [32, 61, 66, 67]. He *et al.* use a hybrid of both unicast and broadcast transmission in application independent data aggregation [37].

In addition to where to send a packet, when to send out the packet also has significant impact on aggregation efficiency and data forwarding delay. Intermediate nodes may have to delay their transmissions while waiting for more data to aggregate. Ideally, a node waits until it hears from everyone from whom it should hear, and then sends out its aggregation result immediately to avoid excessive delay. Unfortunately, even if the node knows which nodes are going to send to it, packets from the senders may be lost due to link errors. A longer waiting time may yield nothing but longer delay. For event-based applications, events happen randomly inside the monitored area. When a node detects an event or receives an event report, it has no idea who else may have detected the same event and may send a report to it. The node may receive nothing at all if it decides to wait longer. Thus, waiting for all expected transmissions does not always work.

Real-time sensor networks require as many samples be aggregated as possible, while ensuring timely arrival of the final aggregates at the sink. The number of samples included in the final aggregates may be sacrificed to meet the delay constraint.

Aggregation protocols can be classified as synchronous or asynchronous based on their timing control schemes. Many aggregation protocols found in the literature use synchronous timing control, which assigns different time intervals to nodes according

their hop count to the sink [61, 66, 67]. The nodes are synchronized in such a way that nodes further away from the sink transmit earlier. Two asynchronous aggregation protocols, periodic simple and periodic per-hop aggregation, are proposed by Solis and Obraczka [86].

The second ingredient, aggregation functions, can be categorized as duplicate sensitive or duplicate insensitive. An aggregation function is duplicate insensitive if and only if multiple occurrences of the same value as input to the function do not affect the result. Functions such as "maximum" and "minimum" are duplicate insensitive, while "count" is not. Duplicate sensitivity is an important factor for aggregation protocol design. Reliability can be improved by forwarding packets along multiple paths when the aggregation result is duplicate insensitive. The broadcast nature of wireless communication can be utilized to forward the same packet to multiple nodes without multiple transmissions. For duplicate-sensitive aggregation functions, such as "count", however, aggregating the same sensor reading at multiple nodes leads to incorrect results. To take advantage of the existence of multiple parents, a simple solution is proposed in TAG (Tiny AGgregation) [61]. To support duplicate-sensitive aggregation functions, an aggregation result is decomposed and broadcast to multiple parents. Two other broadcast-based aggregation protocols proposed by Motegi *et al.* and Gobriel *et al.* support duplicate-sensitive functions by ensuring that the same packet is aggregated by only one of the nodes that receive the packet [32, 66].

Simple aggregation functions such as "maximum", "minimum", and "count" produce a single value as the aggregation result, which can be included in fixed-sized packets. More complicated functions include "median", "count distinct", "histogram", and more [61]. To get an accurate result for "median" and "count distinct", the nodes need to keep track of all distinct sensor readings. As a result, the message size grows linearly with the number of distinct sensor readings in the network. Approximation schemes are proposed by Shrivastava *et al.* to handle quantile-related aggregation functions [82].

Existing data aggregation protocols can also be classified as application de-

pendent data aggregation (ADDA) and application independent data aggregation (AIDA). In application-specific sensor networks, it is efficient to use application level information in data aggregation to eliminate redundancy and minimize the number of transmissions. However, He *et al.* argue that ADDA has limitations due to its lack of adaptation and reliance on application-specific decisions, and propose AIDA for multiple-purpose sensor network systems [37]. AIDA works as a general lossless aggregation scheme between the MAC layer and the network layer. Aggregation is accomplished by combining multiple outgoing network units into a single AIDA payload to reduce the overhead incurred during channel contention and acknowledgment. As the original data can be recovered from the aggregates, no information is lost in AIDA.

The third ingredient in data aggregation deals with data presentation. Shrivastava *et al.* propose an aggregation scheme for quantile-related queries, in which each sensor aggregates the data it receives from other sensors into a fixed size message [82]. A data structure called Quantile Digest or q-digest is devised to store the data.

Aggregation protocols collect data from active nodes and perform aggregation on the collected data to reduce traffic volume. When redundant nodes are common in the network, however, it may not be necessary to collect data from every node. To prolong network lifetime, different node scheduling protocols have been proposed for dense sensor networks [54, 93, 99]. The main idea is to keep just enough nodes awake to provide the desired area coverage, while putting the rest to sleep to save energy.

In dense networks with a significant number of redundant nodes, node scheduling protocols can be used to cycle nodes between active and sleep states. One disadvantage of such node scheduling approaches, however, is that part of the monitored area may be left unattended when a node fails and is not replaced immediately. An alternative approach is explored in this work, in which each node dynamically determines for each round whether enough data has been forwarded towards the sink by neighbouring nodes to provide coverage, and whether it should cancel its transmission.

In the following sections, different aggregation protocols are briefly reviewed. Section 2.3 focuses on unicast-based data aggregation protocols, which are further classified as cluster-based, tree-based, structure-free, semi-structured, and data aggregation based on other structures. Broadcast-based protocols are reviewed in section 2.4. Node scheduling protocols are discussed in section 2.5. Section 2.6 summarizes this chapter.

2.3 Unicast-based Data Aggregation Protocols

Packets can be transmitted by unicast or broadcast in data aggregation protocols. Both unicast and broadcast-based aggregation protocols are found in the literature. Different unicast-based aggregation protocols are briefly reviewed in this section. Broadcast-based aggregation protocols are reviewed in the next section.

In unicast-based aggregation protocols, nodes in the network usually form certain structures such as trees and clusters. Aggregation is performed at intermediate nodes as data is transmitted to the sink over the structure. The tree structure is the most frequently used structure in unicast-based data aggregation. It is also the structure that is used in the new unicast-based aggregation protocols proposed in this work. In addition to the tree structure, the use of other structures for data aggregation, such as clusters, chains, and grids, has been extensively studied. Structure-free data aggregation protocols have also been proposed for event-based applications with dynamic source nodes [22, 85]. In this section, unicast-based aggregation protocols based on different network structures are briefly reviewed.

Cluster-based Data Aggregation

In cluster-based data aggregation, nodes are grouped into clusters, with one cluster head for each cluster. Members of a cluster send packets to their cluster head via single-hop or multi-hop communication; the cluster head aggregates received data, and forwards the results to the sink, also via single-hop or multi-hop communication. In Fig. 2.3, cluster members communicate with their cluster heads directly; the

cluster heads also communicate with the sink directly. Important design issues in clustering systems include energy consumption balancing, cluster head placement, and determining the optimal number of cluster heads.

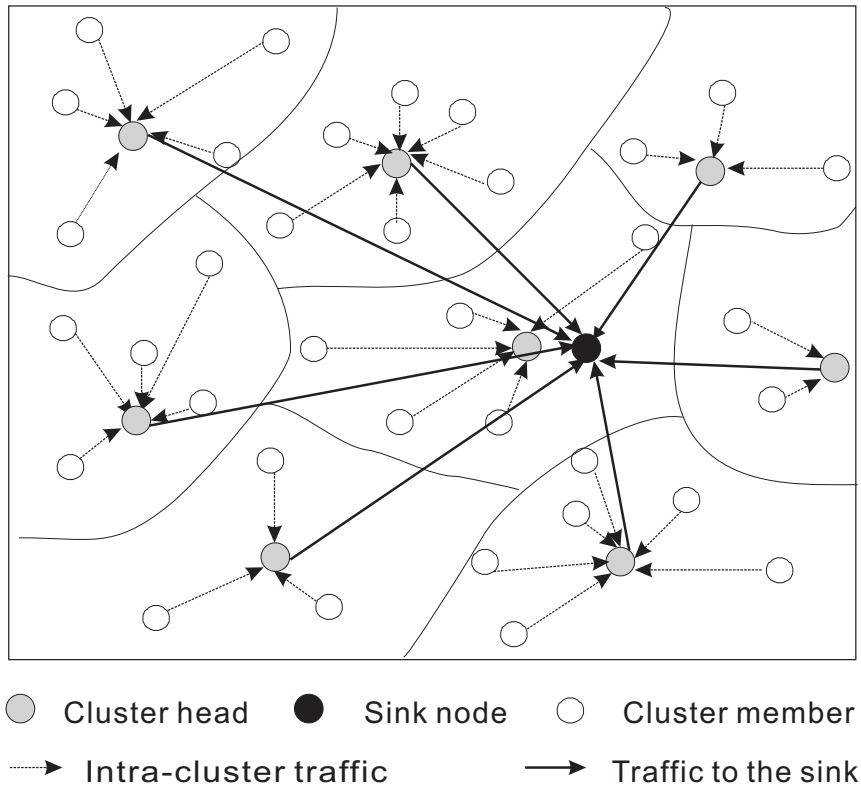


Figure 2.3: Cluster-based Data Aggregation

LEACH (Low-Energy Adaptive Clustering Hierarchy) is a cluster-based data aggregation protocol [40]. In LEACH, nodes are assumed to be able to adjust their power level to achieve different transmission ranges. Single-hop communication is used between cluster members and their cluster heads. A predetermined fraction of the nodes elect themselves as cluster heads based on a randomization function, and broadcast a message to the whole network. The other nodes decide which cluster they should join based on the signal level of the messages that they received. Each cluster head creates a TDMA (Time Division Multiple Access) schedule and assigns a different time slot to each cluster member. After a cluster head hears from all its cluster members, it aggregates the received data and transmits the result directly to the base station via CDMA (Code Division Multiple Access) to avoid interfer-

ence between clusters. To avoid excessive energy consumption at the cluster heads, LEACH rotates cluster head positions among nodes. Simulation results suggest that there is an optimal value for the number of cluster heads that minimizes the energy consumption in the network, but that problem is not studied analytically in LEACH.

Cluster head election and cluster formation are performed in a distributed way in LEACH and may result in poor cluster head placement. The same authors propose LEACH-C (LEACH-Centralized) as an improved version, in which cluster formation is controlled in a centralized manner by the sink based on positions and power levels of the nodes [41].

Younis *et al.* study the problem of clustering nodes in an efficient way to prolong network lifetime, and propose a distributed cluster formation protocol called HEED (Hybrid Energy-Efficient Distributed clustering) [107]. In HEED, cluster heads are selected periodically based on the node's residual energy and a secondary parameter, such as the node's proximity to its neighbours. Simulation results show that HEED is able to distribute cluster heads well among the nodes and prolong network lifetime.

LEACH and HEED consider clustering networks with homogeneous nodes, and have the cluster heads communicate with the sink via single-hop communication. Cluster head rotation is used in the protocols to avoid excessive energy consumption at the cluster heads. Soro *et al.* consider heterogeneous networks with two types of nodes, sensor nodes and more powerful cluster head nodes [88]. They consider the scenario in which the cluster heads form a backbone network and route aggregate results to the sink using multi-hop communication over the backbone. Cluster heads closer to the sink have a higher energy consumption as they serve to relay the packets from cluster heads farther away, resulting in unbalanced energy consumption among the cluster heads. Unequal Clustering Size (UCS) is proposed in the paper to provide uniform energy dissipation. UCS solves the problem by varying the size of the clusters with respect to the expected relay load. The UCS approach is extended to homogeneous networks in which packets from the cluster heads are forwarded to the sink over a backbone formed by the cluster heads. Performance evaluation in the paper shows that UCS is able to achieve more uniform energy dissipation for

homogeneous networks as well.

Mhatre and Rosenberg consider heterogeneous networks in which sensor nodes may use single-hop or multi-hop communication to communicate with their cluster heads, and cluster heads communicate with the sink directly with long range transmissions [64]. Through a cost-based analysis, they study the design problem of determining the optimal number of cluster head nodes, the optimal transmission method within a cluster (single-hop or multi-hop), and the required battery energy of cluster members and cluster heads. A hybrid communication mode is proposed to balance energy consumption of nodes in the same cluster. In the proposed communication mode, cluster members use single-hop and multi-hop communication alternatively to communicate with their cluster heads. Nodes farther away from the sink spend more energy on transmissions because of longer transmission range when single-hop communication is used. When multi-hop communication is used, however, the nodes closest to the sink have higher energy consumption because of their relaying burden.

Chen *et al.* study how to determine the optimal number of cluster heads to minimize total energy consumption in a homogeneous network [14]. After the optimal number is derived, a distributed cluster head selection protocol called Energy-efficient Protocol for Aggregator Selection (EPAS) is presented to achieve the desired number of cluster heads. EPAS consists of two phases. Let k be the optimal number of cluster heads. In the first phase, each sensor independently chooses to be a cluster head with probability p_1, $p_1 \in [0, k/n]$. Suppose that each cluster head has a fixed coverage radius of b meters. In the second phase, each sensor that is not covered by a cluster head declares itself to be a cluster head with probability p_2. How to pick p_1 and p_2 is discussed in the paper. By careful choice of p_1 and p_2, the expected number of aggregators after phase 2 is k . Each node that is not a cluster head selects the closest cluster head as its cluster head.

In cluster-based data aggregation, data from the cluster members may be held back at the cluster head to be aggregated with data from other source nodes. For applications that demand real-time delivery, there is a tradeoff between waiting delay and aggregation accuracy. Each cluster head waits for a certain period of time,

aggregates the packets it has received for a round, and sends out the aggregation result. Packets that arrive after the cluster head has transmitted for that round are not aggregated. Peng *et al.* proposed DSFC (Delay Sensitive Feedback Control) to solve the tradeoff in cluster-based aggregation [72]. In addition, an inter-cluster congestion avoidance scheme, EPCR (Emergent Packet Channel Reservation) is proposed.

In DSFC, the ratio of the number of packets aggregated by the cluster head to the total number of packets received by the cluster head for a round is defined as the aggregation scale. For each round, each cluster head calculates the error of the aggregate it transmitted, i.e., the difference between the aggregate it transmitted and the aggregate of all received packets for that round. The error is then used to adjust the required aggregation scale for the next round. The aggregation scale is increased if the error is bigger than a specific value. The cluster head has to wait for the packets from all its cluster members when the aggregation scale is set to 1, and the aggregation waiting delay is the longest in this case.

When an emergency event, for example, a forest fire, happens inside the network, nodes that detect the event generate a large number of report packets. EPCR is devised to accelerate the transmission of emergency reports. It is assumed that only cluster-head nodes take part in inter-cluster routing. When a cluster head receives an emergency report, it informs other cluster head nodes along the path to the base station that regular packets can be decelerated or transmitted along other paths. EPCR alleviates channel contention and accelerates the transmission of emergency reports.

Tree-based Data Aggregation

In tree-based data aggregation, a tree rooted at the sink is formed, as shown in Fig. 2.4. Aggregation is performed at intermediate nodes as packets are transmitted along the tree towards the sink.

TAG is a tree-based aggregation scheme for aggregating data with a periodic traffic pattern [61]. In TAG, an aggregation tree is formed with the sink as the root. Each node in the tree gathers data periodically and sends a data packet to its parent.

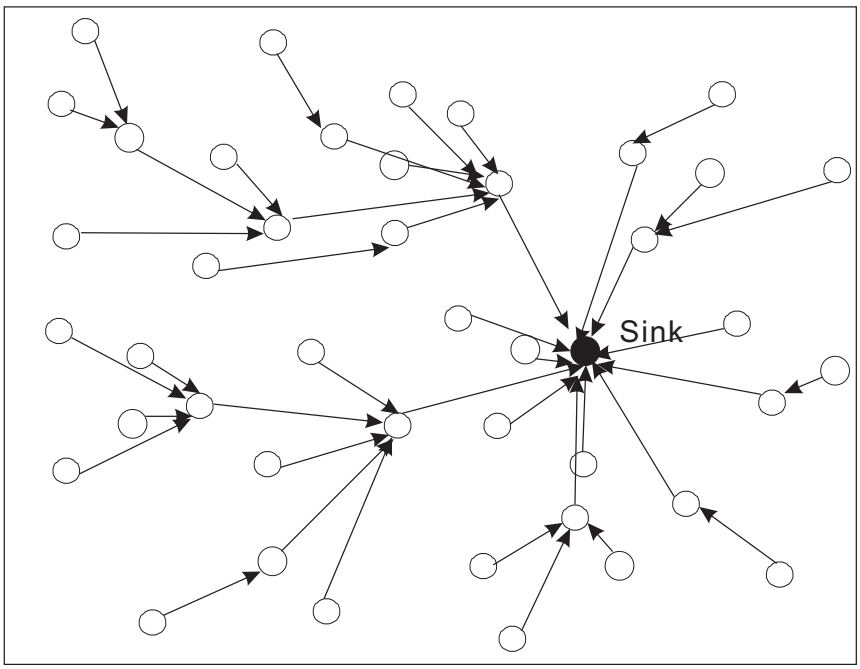


Figure 2.4: Tree-based Data Aggregation

Aggregation is carried out as the data packets are forwarded to the sink.

Data is to be collected every sampling duration in TAG, and a sampling duration is divided into time intervals. A node specifies the interval during which it will be receiving packets from its children, and includes the interval in the query that it forwards to its children. After a child receives the query, it forwards the query to its own children, setting its receiving interval to be ahead of that of its parent. Obviously, the interval should be long enough for the transmissions of all children of a parent, and for the processing of the associated data. A large interval, however, will lead to long delay. The duration of the interval is environment specific. A simple method is used to select the duration of the interval, which is the duration of the sampling duration divided by the depth of the tree, assuming that the depth is known.

Timing in TAG relies on assigning nodes at different tree levels to different time intervals so that nodes at the bottom of the tree transmit first. TAG can be categorized as synchronous data aggregation as it requires synchronization between nodes of adjacent tree levels. A synchronous protocol similar to TAG is used as the basis

for comparison in the following three chapters. In both protocols, nodes are assigned to different transmission intervals according to their hop count to the sink. In TAG, a node is expected to transmit during its own interval, but when exactly the node transmits is not explained. In this work, it is observed that randomizing the transmissions over the interval yields better performance than when all nodes at the same tree level attempt to transmit at the same time. The packet transmissions during the same interval are randomized in the synchronous protocol used for comparison.

The impact of timing in periodic tree-based aggregation is studied by Solis *et al.* [86]. Three timing models are compared in the paper: periodic simple aggregation, periodic per-hop aggregation, and periodic per-hop adjusted aggregation. In periodic simple aggregation, aggregation is performed at regular time intervals, with the aggregation period equal to the data generation period. Each node aggregates all data it has received since its last transmission, and sends out the result at the end of each data generation period. The timing control in periodic simple aggregation is very simple and easy to be implemented, but may result in increased delay. Suppose that node A forwards its data to node B. If node B times out right before A transmits, A's data will be delayed at B until B transmits again. As both A and B transmit every data generation period duration, A's data will always be delayed at B for a time duration close to the data generation period duration.

In periodic per-hop aggregation, the maximum time a node waits before aggregating and transmitting out its result is equal to the data generation period. The node aggregates and sends out its result immediately if the node hears from all children before the maximum waiting time expires. Periodic per-hop adjusted aggregation improves over periodic simple aggregation, but it may still result in increased delay due to the lack of adaption in its timing control strategy. In periodic per-hop adjusted aggregation, nodes decide when to aggregate and send according to their position in the aggregation tree.

The performance of the three timing models is evaluated in terms of energy consumption, data accuracy, data freshness, and overhead. The results show that timing plays a crucial role in aggregation performance for periodic data collection,

and periodic per-hop adjusted aggregation outperforms the other two schemes.

Among the three timing models studied by Solis *et al.*, periodic per-hop aggregation is a synchronous aggregation scheme. The key is still to schedule nodes based on their hop count to the sink node and form a "data wave" towards the sink. Periodic simple aggregation and periodic per-hop aggregation fall into the category of asynchronous aggregation. In Chapter 3, asynchronous aggregation protocols with more aggressive timing control strategies are proposed and compared with synchronous data aggregation in the context of real-time monitoring. In the proposed protocols, each node adjusts its transmission time based on past packet reception history. The results show that the proposed asynchronous aggregation protocols outperforms synchronous aggregation.

A tree-based aggregation protocol called MFS (Multi-level Fusion Synchronization) is proposed by Yuan *et al.* for event-based applications [109]. In MFS, a leaf node sends out an event report to its parent immediately when it detects an event. An internal node initiates its timer when an event is detected and sends out a START message to all its neighbours. Upon receiving a START message, an internal node starts its timer if the timer is not already set and broadcasts another START message to its own neighbours. The process is similar to flooding a message to the whole network. When there is no packet loss in the network, the internal nodes that need to participate in forwarding the event report to the sink set their timers at almost the same time that the event happens. The duration of the timer is set according to the node's position in the tree. For a node k hops away from the sink, the timer is set to $(M_{AX} - k\Delta)$, where and Δ are protocol parameters.

The lowest latency that MFS is able to achieve is approximately equal to M_{AX}, when all pertinent internal nodes trigger their aggregation timers at almost the same time that the relevant event occurs (ignoring propagation delays). In the worst case, none of the pertinent internal nodes receive the START message (due to collisions or link errors), and the latency can be $DM_{AX} - (D-1)D\Delta/2$, where D is the tree depth. MFS incurs long latency for events that happen close to the sink, especially when the network is large. When a node one hop away from the sink detects an

event, it still has to wait $(M_{AX} - \Delta)$ before sending out its event report.

Zhang *et al.* propose DCTC (Dynamic Convoy Tree-based Collaboration) for detecting and tracking a mobile target in sensor networks [112, 113]. The key in DCTC is a tree structure called the convoy tree, which includes only nodes surrounding the target. The tree changes dynamically as the target moves around. The root of the convoy tree collects information from the source nodes, generates a report, and forwards it to a single sink or multiple sinks.

The main challenge in DCTC is how to reconfigure the tree in an efficient way as the target moves. New nodes need to be added to the tree, and existing tree nodes may need to be pruned. The root may have to be replaced as the target moves away from it. Tree expansion and pruning schemes are proposed in the paper, as well as tree reconfiguration schemes.

Aggregating data from a sparse set of nodes in a wireless sensor network is studied by Gao *et al.* [30]. Each participating node knows that there are other nodes that are collecting data, but no global information about those nodes is available. The authors investigate how participating nodes discover each other in a distributed way, and how these nodes form an aggregation tree to promote data aggregation. Nodes are assumed to be uniformly deployed in a region. It is also assumed that nodes on the boundary are connected to an external high-speed network. The base station is located in the external network, which collects and processes the aggregates. Each node uses a pseudo-random hash function on its node ID to generate a priority value for itself. The base station has the highest priority. It is assumed that the nodes are loosely synchronized.

Gao *et al.* devise a probe protocol for node discovery [30]. The participating nodes initiate the probe process at roughly the same time by sending out probe packets in the four directions, north, south, west, and east, as illustrated in Fig. 2.5. Suppose node w receives a probe message from node p. It ignores the message if the message has been received before. Otherwise, node w records the message. If node w received probe messages from other nodes before, and among these nodes from which node w has heard, there is someone with higher priority than node p, then the

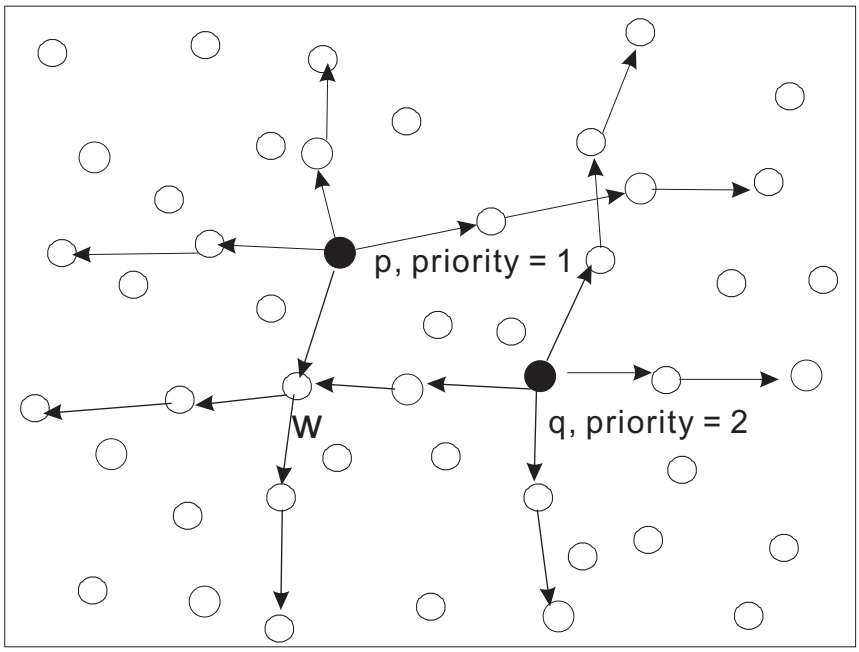


Figure 2.5: The Probe Process

one with the highest priority becomes a candidate for p's parent. If node w finds a parent candidate for p, a recall process is started to inform other nodes, including node p itself, to stop propagating p's probe. The recall message also informs node q about its parent candidate. Node p may receive multiple recall messages from several nodes, and in that case, the parent candidate included in the first recall that node p receives becomes p's parent. If node p's priority is higher than all nodes whose probe messages have been received by w, node w retransmits p's probe message in the same direction the message came in. Eventually, an aggregation tree rooted at the base station is built, since the base station has the highest priority among the participating nodes. Note that the aggregation tree is a logical structure. The parent of a participating node is either another participating node or the base station.

The impact of tree construction algorithms is studied by Krishnamachari *et al.* [52]. Assuming that each node only transmits a single packet, the minimum number of transmissions is achieved with a minimum Steiner tree, which is NP-hard (Non-Deterministic Polynomial-Time Hard) to find in general. Three suboptimal schemes

are proposed in the paper as alternatives: Center at Nearest Source (CNS), Shortest Path Tree (SPT), and Greedy Incremental Tree (GIT). In CNS, the source node nearest to the sink serves as an aggregation point. All other source nodes send their data to the aggregation point directly. In SPT, each source node sends its data to the sink along the shortest path. The paths merge when they overlap. In GIT, the tree only contains the shortest path between the sink and the closest source node at the beginning. Then the source node closest to the current tree is connected to the tree. This process repeats until all source nodes are connected to the tree.

The impact of tree construction on data aggregation is evaluated in terms of energy saving and delay through simulation. Two source placement models are considered: the event-radius model, in which the source nodes are located within a certain range of an event point, and the random-source model, in which the source nodes are randomly selected from the network nodes. Energy saving from aggregation is measured by the number of transmissions. Delay incurred by aggregation is measured by examining the hop count difference between the nearest source and the farthest source node, since data from nearer nodes may be held back in order to be aggregated with data from source nodes that are farther away. The results show that data aggregation can greatly reduce data transmissions, but incurs non-negligible delay. GIT is showed to have the best performance among the three schemes.

Structure-Free Data Aggregation

For event-based applications with dynamic source nodes, a fixed network structure may result in poor aggregation efficiency. An example is shown in Fig 2.6. While node A and Node B are within each other's transmission range, their event reports take completely different routes to the sink. If node B sends its event report to node A, and node A aggregates the event report with its own and forwards the aggregate to the sink, then only four transmissions are needed to report the event to the sink, assuming that the event reports can be aggregated into a single packet.

While a fixed structure may yield poor aggregation performance, changing the structure dynamically, however, may cause high overhead. Fan *et al.* explore the

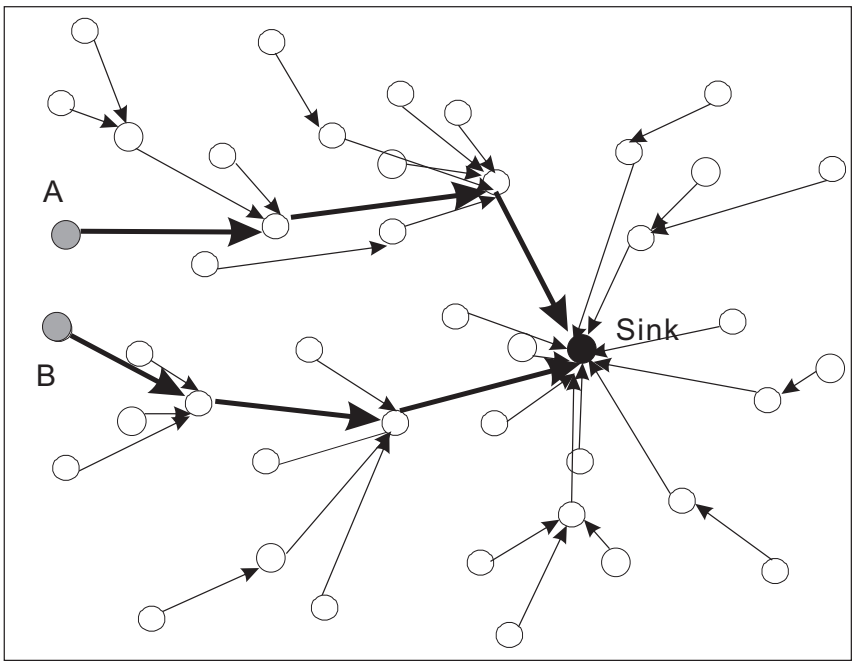


Figure 2.6: Tree-based Data Aggregation Yields Poor Performance for Event-Based Application

potential of structure-free data aggregation [22]. Two mechanisms are proposed in their structure-free protocol to improve spatial and temporal convergence: Data-Aware Anycast (DAA) and Randomized Waiting (RW). As there is no pre-configured structure, routing decisions are made on the fly through DAA. When a node has something to send, it broadcasts an RTS with an Aggregation ID (AID) to its neighbours. The measurement timestamp is used as the AID in the paper. Packets with the same AID can be aggregated. Therefore, packets that are generated at the same time can potentially be aggregated.

To improve spatial convergence, the nodes' neighbours are assigned different priority to respond with a CTS. Neighbours that have packets with the same AID and are closer to the sink have the highest priority, followed by neighbours with packets with the same AID, but farther away from the sink. Neighbours that have no packets with the same AID, and are farther away from the sink, are assigned the lowest priority. To decide its priority to respond with a CTS, each node must be able to

determine its relative distance to the sink.

When a node detects an event, or receives an event report with an specific AID, the node has no idea whether any packets with the same AID are being forwarded to it. It has to decide independently how long it should wait for more packets to aggregate. A simple approach is to assign a fixed waiting time to each node and have nodes closer to the sink wait longer. This approach, however, may result in a long delay for events that happen in spots close to the sink. To improve temporal convergence, RW is used in the proposed structure-free aggregation. The main idea is to randomly delay a node's transmission to improve the chance of aggregation without incurring long delay in large networks.

While structure-free aggregation does not have any overhead in structure construction and maintenance, it does not guarantee aggregation of all packets from a single event. Energy is wasted when unaggregated packets are forwarded to the sink, especially when the event happens far from the sink. To further improve energy efficiency, Fan *et al.* propose a semi-structured aggregation protocol, which is introduced below.

Semi-structured Data Aggregation

To benefit from the strengths of the structured and the structure-less approaches, Fan *et al.* propose a semi-structured protocol based on DAA [23]. In the semi-structured approach, DAA and RW are used to aggregate packets locally. When DAA cannot achieve further aggregation, packets are transmitted over a structure called Tree on Directed acyclic graph (ToD) instead of being forwarded to the sink directly. The goal is to make packets related to the same event meet early and get aggregated as they are forwarded over the structure.

ToD is composed of multiple shortest path trees. To construct a ToD, nodes must know their geographic locations. The whole network is divided into square cells with side length greater than the maximum diameter of the event area. These cells are grouped into First-level clusters (F-clusters), and then the F-clusters are grouped into Second-level clusters (S-clusters). Each F-cluster has a cluster head called F-

aggregator, and each S-cluster also has a cluster head called S-aggregator. When a packet cannot be aggregated by DAA any more, it will be sent to the F-aggregator that covers the cell of the sender. The F-aggregator aggregates packets related to the same event, and decides whether the result should be forwarded to the sink directly or to a S-aggregator for further aggregation, based on the location of the event.

When geographic information is available to each node, the ToD structure can be built without any overhead. This semi-structured approach achieves efficient aggregation in large scale networks by getting packets aggregated near the event location.

Data Aggregation based on Other Structures

In LEACH, multiple cluster heads communicate directly with the sink over a long transmission range [40]. Energy consumption can be further reduced if only one node communicates with the sink using long range transmission. Lindsey *et al.* propose a chain-based aggregation protocol called PEGASIS (Power Efficient data GAthering protocol for Sensor Information Systems) to improve energy efficiency [59]. As in LEACH, PEGASIS makes the same assumption that nodes in the network are able to adjust their transmission power level and communicate with the sink directly. A greedy chain formation scheme is proposed in PEGASIS, which requires global knowledge of the network at each node. After the chain is formed, each node has two neighbours along the chain. Nodes at the end of the chain start transmission first. When a node receives a packet from its neighbour on the chain, it aggregates the data with its own and transmits the aggregation result to the other neighbour on the chain. Eventually, the packets arrive at a leader node, which transmits the final aggregate to the sink directly. Nodes take turn in becoming the leader node to balance their energy usage.

In addition to the structures mentioned above, grid-based aggregation protocols are also found in the literature [95]. The basic idea is to divide the monitored area into grids and select one node inside each grid as the data aggregator, as seen in Fig. 2.7. Nodes inside the grid send their data to the aggregator, which then aggregates

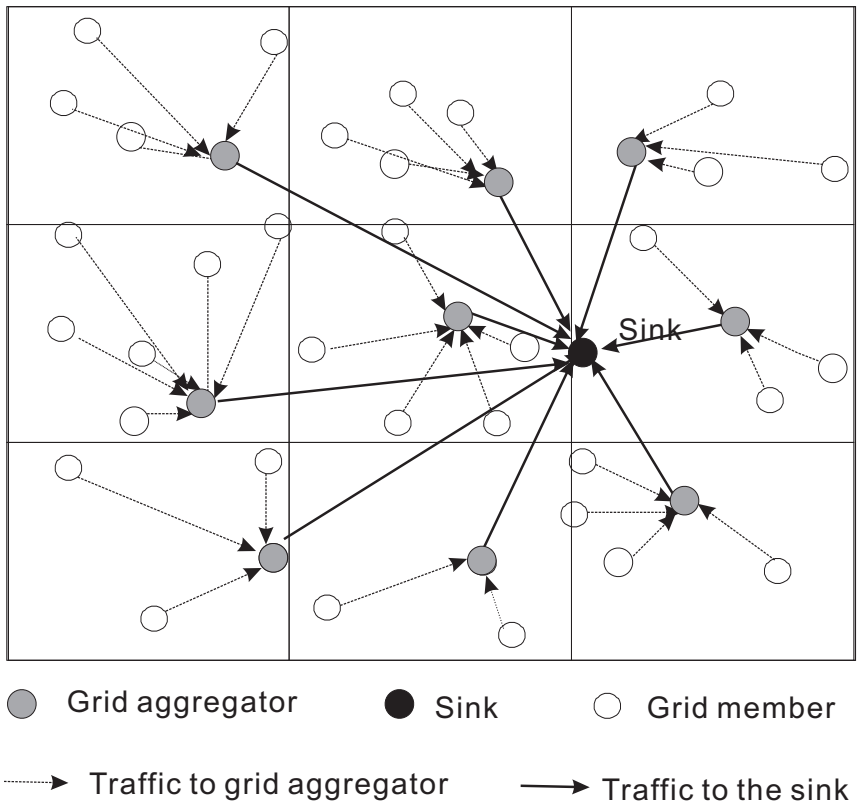


Figure 2.7: Grid-based Data Aggregation

the received information and sends the aggregation results to the sink node.

2.4 Broadcast-based Data Aggregation Protocols

Since nodes often have more than one neighbour in sensor networks, one single broadcast is enough to transmit a packet to multiple nodes in broadcast-based aggregation. One problem that comes with the benefit is that aggregating the same data at multiple nodes yields incorrect results for duplicate-sensitive aggregation functions.

Broadcast-based aggregation protocols have been proposed for duplicate-sensitive aggregation functions in sensor networks. As the same broadcast may be received by multiple nodes, these protocols must ensure that concurrent processing of the same data at multiple nodes does not produce wrong results. One common strategy is to guarantee that only one of the recipients aggregates the data [66]. Nath and Gibbons solve the problem by converting data into duplicate-insensitive synopses, which are

small-sized digests of the data [67].

Existing broadcast-based data aggregation protocols are briefly reviewed below. Most of these protocols are focused on protocol design for duplicate-sensitive aggregation functions. New broadcast-based data aggregation protocols are proposed in Chapter 4. Instead of focusing on duplicate-sensitive aggregation functions, it is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates including the same sensor value, either because the aggregation function is duplicate insensitive or because the sink is able to filter duplicates. The performance comparison between the new broadcast-based protocols and their unicast-based counterparts aims to investigate whether broadcast-based aggregation has any performance advantage in the context of real-time and delay-tolerant data collection.

Value Splitting

Value splitting is proposed by Madden *et al.* as a means of leveraging the available redundancy of the network topology to increase reliability [61]. While nodes are organized into a tree structure in TAG, the authors observe that a node can easily find multiple parents that are one step closer to the sink. The value splitting scheme considers nodes with two possible parents. To support duplicate-sensitive aggregation functions, aggregation values are split into two parts at child nodes. A single broadcast is enough to deliver the partial value to both parents of a child node when the aggregation results can be linearly split into two equal values.

A simple statistical analysis is done in the paper. The performance of value splitting is compared to that of a single-parent broadcast-based scheme based on the aggregation function "count". The results show that value splitting yields lower variance for the same expected value. The performance of the two aggregation schemes is also evaluated through experiments, and the results show that value splitting outperforms the single-parent broadcast-based scheme.

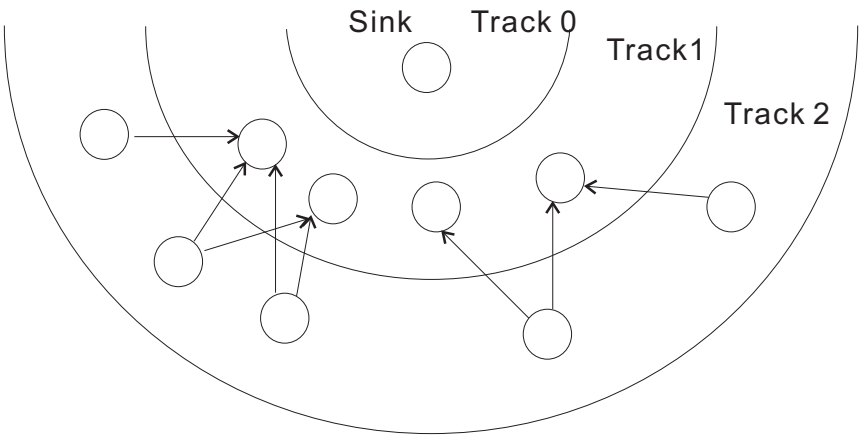


Figure 2.8: Track Topology

RideSharing

Gobriel *et al.* propose a fault tolerant broadcast-based aggregation protocol, RideSharing, for duplicate-sensitive aggregation in sensor networks [32]. Two mechanisms are proposed: cascaded RideSharing and diffused RideSharing. Both schemes organize the nodes in a track graph. The sink is located in track 0. Nodes one hop away from the sink are located in track 1, and so on, as seen in Fig. 2.8. Nodes in track n may have multiple parents in track $n-1$. Among these parents, one is selected as primary parent and the others serve as backup parents.

A bit vector is included in the packet. When the transmission between a child and its primary parent fails, the primary parent indicates which child is missed in the bit vector included in its packet. Cascaded RideSharing and diffused RideSharing differ in the way that an error is corrected. In cascaded RideSharing, error correction is completed by one of the backup parents, while in diffused RideSharing, multiple backup parents cooperate in error correction by aggregating a share of the missing value.

In cascaded RideSharing, each node has a correction sequence, which may be an ascending order of node ID. The backup parents decide whether to correct an error based on their orders in a correction sequence. The first backup parent in the sequence tries to correct the error first. If the correction is successful, the first backup parent sets the bit vector in its packet to indicate that the error has been

corrected. The next backup parent takes action to correct the error if and only if the backup parent ahead of it in the sequence fails to do so.

Multiple backup parents collaborate in error collection in diffused RideSharing. Each backup parent only aggregates a share of the missing value. To calculate its own share of the missing value, each backup parent keeps a counter of the remaining parents, including the node itself and the backup parents from which it has not heard. If a backup parent is not able to aggregate its share, the remaining backup parents adjust their share to compensate for the missing part. A backup parent in diffused RideSharing has to be able to hear the other parents to maintain an accurate estimate of the number of the remaining parents, while a backup parent in cascaded RideSharing only needs to be in the communication range of the preceding parent in the correction sequence. Parent clique is proposed as an enhancement in the paper, in which a child only selects nodes that can hear each other as its parents. This may limit the number of nodes that a child can select as its parents. When a node in track n has only one neighbour in track $n-1$, the node selects parents from its own track.

Transmission order plays an important role in RideSharing. A child should transmit before its parents. Backup parents should transmit after the primary parent. The backup parents also need to transmit in a certain order to increase the probability of error correction. TDMA or prioritized contention based schemes are mentioned in the paper as possible solutions. The impact of timing on latency is not evaluated.

One problem with RideSharing is that nodes in track 1 (nodes one hop away from the sink), have only one parent in track 0, which is the sink itself. At least one node has only one parent even if co-tracking is adopted. Meanwhile, the aggregation results generated by the nodes in track 1 may contain a large amount of data, the loss of which may have a significant impact on the end-to-end loss rate. In the performance evaluation, links between track 1 and the sink are assumed to be error-free. This assumption makes the results less convincing.

DAG-based Data Aggregation

Motegi *et al.* propose a multi-parent broadcast-based aggregation protocol based on DAG (Directed Acyclic Graph) [66]. To support duplicate-sensitive functions, the proposed protocol assigns a destination node to each node. Data generated by a node will not get aggregated until packets containing that data arrive at the node's destination node.

Before data collection starts, the sink broadcasts a request message (REQ) to the entire network, which includes a parent node list (the list is empty for the sink), the hop count to the sink (in the case of the sink itself, its hop count is set to zero), and other parameters about the query, such as data collection frequency. The parent node list includes the IDs of the parent nodes that the node has learned from the received REQs. When the node rebroadcasts the received REQ, it includes its parent node list in the REQ. At the end of the REQ flooding process, each node has a list of parents and grandparents.

Each node selects its destination node based on information about its possible parents and grandparents. If the node has multiple parent candidates, it checks whether the parent candidates have any common parent. If some of its parent candidates share a common parent, the node chooses the common parent of these parent candidates as its destination node, and these parent candidates become the node's parents. In the case that no two parent candidates share the same parent, the node selects one of these candidates as both its parent and its destination node.

The proposed DAG-based protocol uses similar timing control as TAG [61], in which each node decides when to transmit based on its distance to the sink. When a leaf node sends out its packet, it puts its sampling value and its destination node ID in the packet. A parent node may take one of the three actions on the value it receives from its child: aggregate, forward, and discard. If the node is the destination node for the value, it aggregates the value with its own value and other possible values. If the destination node for the value is one of the receiver's parents, the value is forwarded without being aggregated. The value is simply discarded if the destination node of

the value is neither the receiver itself nor the receiver node's parent.

Synopsis Diffusion

Instead of avoiding aggregating the same data at multiple nodes to support duplicate-sensitive aggregation functions, synopsis diffusion, a broadcast-based aggregation protocol, solves the problem by decoupling aggregate computation from message routing [67]. In the proposed protocol, data is represented by duplicate-insensitive synopses, which are small-sized digests of the data. Sensor readings are converted to synopses by a *synopsis generation function*, and two synopses can be fused into a new synopsis by a *synopsis fusion function*. A *synopsis evaluation function* is used at the sink to translate the synopses into meaningful data. Neither the order in which the data from different nodes is aggregated nor the number of times the same data is aggregated has any impact on the final aggregate at the sink.

Obviously, the three functions are key parts of the protocol and have to be designed for each different aggregation function. The authors show the functions for "count", "sum", "count distinct", "most popular items" in the paper.

AIDA

AIDA is an application-independent lossless aggregation proposed by He *et al.* for multiple-purpose sensor network systems [37]. AIDA is not a pure broadcast-based aggregation protocol; it uses both unicast and broadcast at the MAC layer for packet transmissions.

AIDA works between the network layer and the data link layer. Aggregation is performed by combining multiple outgoing network units into a single AIDA payload to reduce the overhead incurred during channel contention and the number of acknowledgments. AIDA saves energy for several reasons. First, channel contention cost is reduced as multiple network units are merged into one AIDA payload. Second, fewer MAC control messages are sent in AIDA. Finally, for sensor network platforms that use fixed packet sizes, such as TinyOS [42], AIDA can reduce unnecessary padding, although fixed sizes could prevent aggregation too.

Three different aggregation strategies are proposed in AIDA: the fixed scheme, the on-demand scheme, and the dynamic feedback scheme. In the fixed scheme, aggregation is conducted when the number of network units to be aggregated reaches a certain value or when a timer goes off. The fixed scheme may cause high delay when traffic load is low, since the node has to receive enough network units, or wait until timeout before aggregating the received data and sending out the result. In the on-demand scheme, aggregation is done whenever the MAC is available for transmission and there are network units to be transmitted. The dynamic feedback scheme works by adjusting the number of network units required for aggregation according to the network condition. When workload is high, the value is increased to raise the degree of aggregation. The dynamic feedback scheme performs the best among the three schemes in terms of delay and energy saving.

2.5 Node Scheduling

Area coverage is a common requirement in sensor network applications [11, 93, 111]. In these applications, sensor nodes must be deployed over some geographic region such that each point within the region is covered by the sensing capability of at least one node. When sensor networks are densely deployed, the existence of redundant nodes leaves the researchers room to improve the system design of sensor networks. Node scheduling protocols have been proposed to prolong network lifetime. In node scheduling protocols, nodes are cycled between active and sleep states to reduce energy consumption.

Coverage preservation is one major concern in node scheduling protocols. After some nodes are put to sleep, remaining nodes must be able to preserve the original area coverage, or achieve the desired coverage as required by the application. In particular, security surveillance requires that each point of the monitored area be covered by some minimum number of sensor nodes. For environmental monitoring applications, data collected should reflect the status of the whole monitored area.

In dense monitoring sensor networks, node scheduling protocols can be imple-

mented to cycle nodes between active and sleep states. The active nodes collect and transmit data to a sink node. Data aggregation is performed at intermediate nodes to reduce energy consumption. Different node scheduling protocols are briefly reviewed below. The performance of data aggregation relying on node scheduling is evaluated and compared to coverage preserving data aggregation in Chapter 5. In coverage preserving aggregation, each node dynamically determines during each round of data collection whether it should transmit its data, or whether its sensing area is covered by neighbouring nodes that have already transmitted.

Sensor Coverage model

The most frequently used coverage model assumes a deterministic sensing range [54, 93, 99, 100, 104]. A point p is covered by a node n if their distance is less than the node's sensing range S; i.e., node n's coverage region is bounded by a circle with radius S. Xing *et al.* use a probabilistic detection model in which the probability of detecting an object is a function of the distance between the object and the node [101].

Node Scheduling with Deterministic Coverage Preservation

Deterministic coverage preservation ensures that the selected nodes meet the desired coverage requirement as long as the network is able to support it. Thus, location information is essential for deterministic coverage preserving algorithms. It is very difficult to guarantee that the selected subset of nodes can maintain the original or the desired sensing coverage without location information.

Tian *et al.* use a sponsored sector mechanism to select active nodes in their coverage preserving node scheduling protocol [93]. In the proposed protocol, each node keeps the location information of its *off-duty sponsors*, i.e., neighbours within the node's sensing range S. The node may have other neighbours whose sensing areas overlap with the node's sensing area, i.e. neighbours within the $(S, 2S)$ range, but they are not considered the node's off-duty sponsors. Based on the location information of the off-duty sponsors, the node computes its sponsored sector with

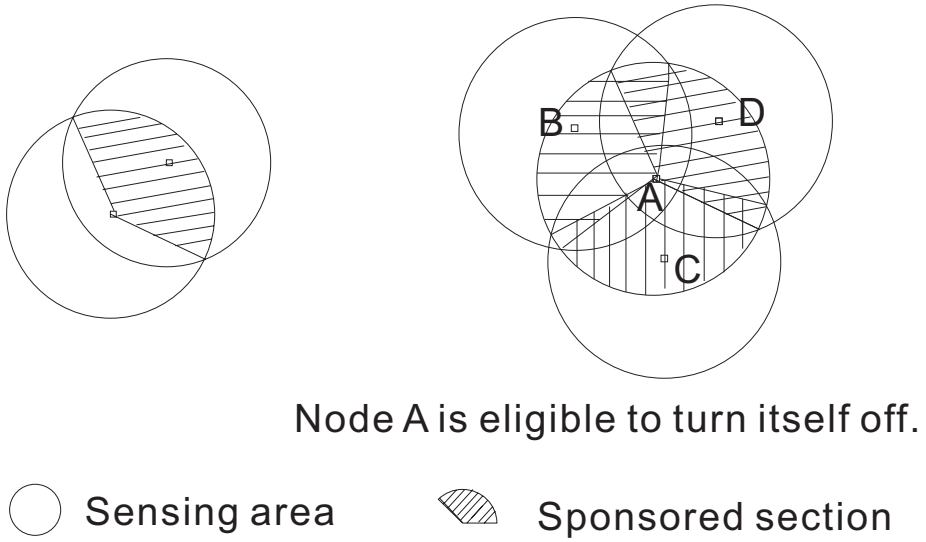


Figure 2.9: Sponsored Sector Mechanism

each off-duty sponsor, which is the sector covered by both the sponsor and the node itself. The node is eligible to turn itself off only if the union of its sponsored sectors covers the node's whole sensing area, as seen in Fig. 2.9.

Operation is divided into rounds in the proposed protocol. Each round begins with a self-scheduling phase, followed by a sensing phase. During the self-scheduling phase, nodes decide whether to turn themselves off based on their eligibility. A random backoff strategy is used to avoid simultaneous actions at different nodes. Obviously, the sponsored sector mechanism is relatively conservative in determining a node's eligibility to turn off as the neighbours within the $(S, 2S)$ range are not considered the node's off-duty sponsors. There might still be redundant active nodes in the network.

Wang *et al.* combine connectivity with coverage in their work [99]. The network is said to to be K-covered or have K-coverage if each point in the monitored area is covered by at least K nodes. They prove that 1-coverage of a convex area implies connectivity if the communication range is at least twice the sensing range, i.e., if the nodes are able to cover the whole monitored area, then the nodes must be connected if the communication range is at least twice the sensing range. Fig. 2.10 shows a simple scenario in which the communication range is shorter than the sensing range. In Fig. 2.10, the monitored area is covered, but the nodes are disconnected.

It is further proven that a convex area is K-covered, if all intersection points between two sensing circles, and all intersection points between the sensing circle of any sensor and the boundary of the area, are at least K-covered. At the same time, K-coverage implies K-connectivity when the communication range is at least twice the sensing range ($K \geq 1$) (K-connectivity means that at least K nodes have to be removed in order to partition the network into disconnected parts). A coverage configuration protocol named CCP (Coverage Configuration Protocol) is presented to provide different coverage degree for sensor networks. In CCP, each node keeps the location information of the nodes within a distance of twice of its sensing range. A node is eligible to turn itself off if all intersection points within its sensing range are at least K-covered. The authors combine CCP with a connectivity-preserving protocol to ensure both K-connectivity and K-coverage when the communication range is less than twice the sensing range.

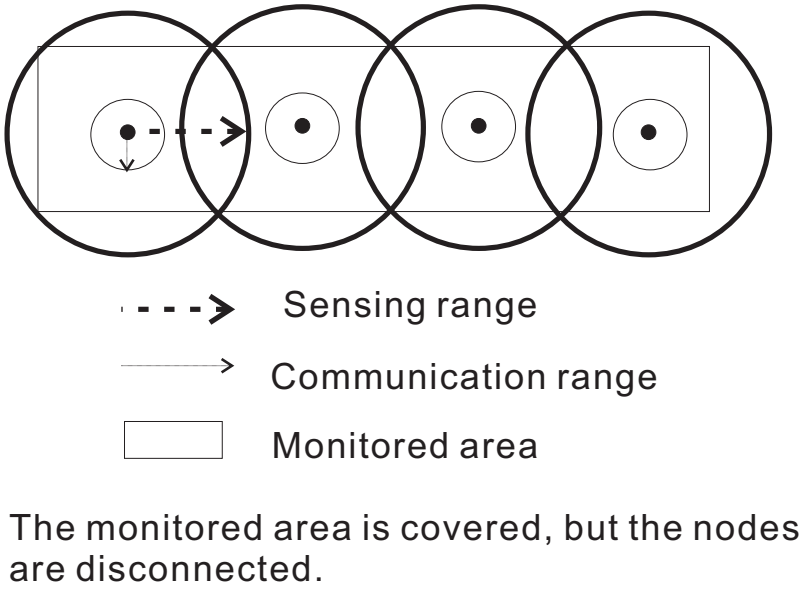


Figure 2.10: Connectivity and Coverage

Zhang and Hou present a similar analysis of the relation between coverage and connectivity in their work [111]. It is observed that an area is completely covered, if all intersection points of two sensing circles, and all intersection points of one sensing circle and the boundary of the area, are covered by a third sensing circle.

The authors investigate how to minimize the number of nodes that completely cover the area of interest. An important notion called *overlap* is defined in the paper. The overlap of a point inside the monitored area is the number of nodes that cover that point minus 1. The overlap of a point outside the area is the number of nodes that cover that point. The overlap of the whole area covered by the nodes is the integral of the overlaps of all points covered by the nodes. Zhang and Hou prove that minimizing the number of active nodes is equivalent to minimizing the overlap of the active nodes if all nodes completely cover the region of interest and have the same sensing range.

Furthermore, Zhang and Hou also present a distributed node scheduling protocol, OGDC (Optimal Geographic Density Control), which aims to minimize the number of working nodes by minimizing the overlap of the working nodes [111]. When the communication range is less than twice the sensing range, 1-coverage does not mean connectivity, i.e., nodes that are able to achieve 1-coverage may not be connected. An extension of OGDC is proposed to solve the problem.

Among the three protocols mentioned above (sponsored sector, CPP, and OGDC), OGDC works the best to reduce redundancy but only considers 1-coverage. Sponsored sector is also focused on 1-coverage and is conservative in turning nodes off. Simulation-based performance evaluation in OGDC shows that OGDC outperforms CPP. In all three protocols, each node relies on location information of its sensing neighbours to determine its eligibility to turn itself off. The simulation results in OGDC show that CPP fails to achieve the expected coverage when the network is highly dense. The operation of CPP is divided into rounds, and nodes decide whether to stay awake at the beginning of each round. CPP relies on message exchanges to keep the nodes updated on their neighbours' location information. As packet losses due to collisions are common in a dense sensor network, a node may fail to refresh its information and make a wrong decision to turn itself off. OGDC also needs to exchange information at the beginning of each round, but its message exchange overhead is much lower than CPP.

Node Scheduling with Statistical Coverage Preservation

While location information of the neighbours enables a node to decide precisely whether all points within its sensing range are covered by it neighbours, information exchange and storage may incur high overhead, especially in dense networks. Moreover, location information may be unavailable in the network.

Wu *et al.* assume that no location information is available and propose LDAS (Lightweight Deployment-Aware Scheduling) to provide statistical guarantees on sensing coverage [100]. LDAS assumes that the nodes are randomly deployed with uniform distribution in the network. Nodes located outside a node's sensing circle are not considered as that node's sensing neighbours even if their sensing areas overlap with that of the node.

In an earlier work, the same authors prove that if a node's sensing area can be completely covered by other nodes, then at least three neighbours and at most five neighbours are needed to cover the node's sensing circle [31]. This result suggests that just a few neighbours may be enough to completely cover a node's sensing area or cover a large proportion of the node's sensing area with high probability.

For LDAS, Wu *et al.* show that it is expensive to provide complete coverage as 11 sensing neighbours are needed to achieve a 90% probability of covering a node's sensing area completely. If moderate loss of area coverage is tolerated, however, only 5 sensing neighbours are required to cover 90% the node's sensing area, which means the network density can be significantly reduced.

All nodes in LDAS stay in one of the three states: on-duty, ready-to-off, and off-duty. The finite state machine is shown in Fig. 2.11. Nodes in the on-duty state are responsible for sensing and communication. Meanwhile, each on-duty node keeps track of the number of its working neighbours. When the number exceeds a threshold computed based on the coverage requirement, the node randomly selects some of its working neighbours to turn off and sends tickets to them. A node enters the ready-to-off state when it collects enough tickets. After staying in the ready-to-off state for a random back-off time, the node turns itself off and sleeps for a certain

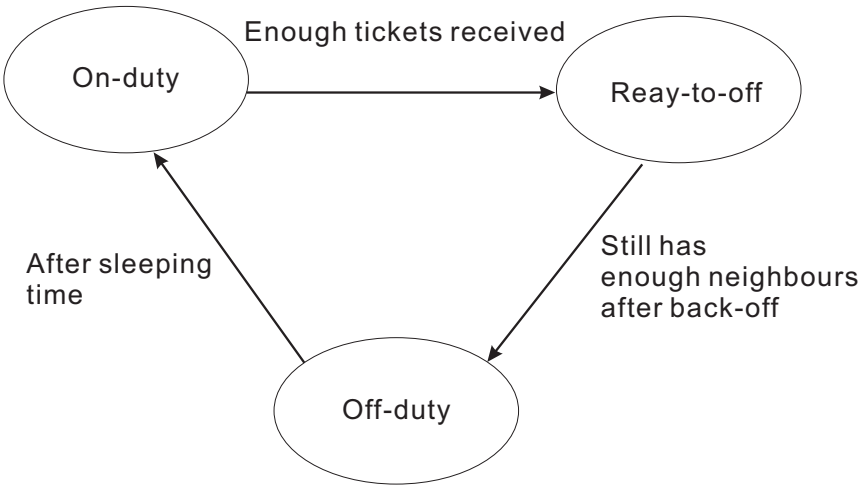


Figure 2.11: Finite State Machine of LDAS

sleeping time if it still has enough working neighbours.

PEAS (Probing Environment and Adaptive Sensing) is another node scheduling protocol that does not assume the knowledge of location information [104]. PEAS aims to keep a required number of nodes alive in the presence of node failures. All nodes start from the sleeping mode and wake up after an exponentially distributed random time. When a node wakes up, it broadcasts a PROBE message with a predefined transmission/probing range. The node continues to be awake only if no one replies to it. Otherwise, it goes to sleep for another random time. The probing range can be adjusted to control the number of working nodes. As the result of PEAS, working nodes are separated by a minimum distance of the probing range.

2.6 Summary

The chapter provides a brief introduction on wireless sensor networks and data aggregation. Different unicast and broadcast-based data aggregation protocols are reviewed. Most prior work has focused on the functionality of data aggregation. This dissertation focuses on performance optimization issues in data aggregation. Existing node scheduling protocols are also reviewed in this chapter. The performance of data aggregation relying on node scheduling is used for comparison in the performance evaluation in Chapter 5.

Chapter 3

Synchronous vs. Asynchronous Data Aggregation

3.1 Motivation

This chapter considers data aggregation in the context of sensor networks supporting real-time monitoring, specifically real-time monitoring systems where sensor data is sampled *periodically* and forwarded to a single *sink* node. Different real-time monitoring applications place different constraints on how "current" a view must be maintained of the monitored area [1]. Protocols able to support a more current view enable new or enhanced monitoring applications. A primary performance metric for aggregation protocols for this domain, therefore, is how current a view they can support, as determined by the sustainable sampling rates and the data delivery delays.

Aggregation protocols for sensor networks with periodic traffic usually transmit sensor values over a tree or cluster topology, rooted at the sink [40, 61, 62, 86]. Previous work has advocated synchronous aggregation protocols, in which a sequence of time intervals are statically defined for each *round* (collection of one set of sensor values), with each interval dedicated to transmissions from particular sensor nodes [86]. Recall that TAG is an example of an aggregation service using synchronous aggregation [61]. In TAG, each node, beginning with the sink node, informs its children in the aggregation tree of the interval during which it will be receiving data. A child's transmission interval is fixed as the receiving interval of its parent, and the child's own receiving interval is chosen as the immediately preceding interval.

Thus, all of the sensors at the i^{th} level of the aggregation tree, $1 \leq i \leq H$, share transmission interval $H - i$, where H denotes the height of the tree and where the first interval in a round is numbered as interval zero. All intervals are of identical duration. A potential disadvantage of synchronous aggregation is increased delay, since the interval duration must be conservatively chosen so as to provide a high probability that each node will be able to successfully transmit its data to its parent prior to the end of its transmission interval. A second potential disadvantage is that the constraints imposed on node transmission times may result in suboptimal use of spatial multiplexing.

Solis and Obraczka have described and evaluated two asynchronous aggregation protocols, called periodic simple and periodic per-hop aggregation [86]. In periodic simple aggregation, each node waits for a period of time equal to the round duration, aggregating all of the data received from its children over that period, before transmitting to its parent. This approach does not provide low delay; in fact, data generated during one round may not be received at the sink for a number of rounds equal to the height H of the aggregation tree.

Periodic per-hop aggregation is similar to periodic simple aggregation in that nodes may wait for a period of time equal to the round duration before transmitting to their parent, but each node may transmit earlier if data is received and aggregated from all children prior to the end of the round. Again, this approach may result in long delays, with the data generated in one round not being received at the sink until some subsequent round. These simple asynchronous protocols were found to yield poorer performance than synchronous aggregation in the paper [86].

In this chapter, improved asynchronous aggregation protocols are designed through use of more aggressive methods for determining when a node should transmit to its parent. If a node receives data from all of its children prior to sending its own data to its parent, in a given round, all of this data is aggregated and sent to the parent at that point. A node will also transmit to its parent if the time it has been waiting for its children exceeds an adaptively determined timeout value. In this case, any "late arrivals" from its children are simply dropped. The choice of the timeout value

is critical, since a long timeout value may cause excessive delay, while substantial data loss may be incurred if the timeout value is too short. In the proposed protocols, timeout values are adaptively determined based on local history of past packet receptions.

The new asynchronous protocols, as well as synchronous aggregation, are evaluated using simulation. Asynchronous aggregation is found to outperform synchronous aggregation. Performance comparisons of the asynchronous protocols show that adaptation of timeout values based on a weighted average of history information from multiple rounds is preferable to adaptation based only on the immediately previous round. It is also found that randomizing the transmission times of leaf nodes to avoid congestion at the beginning of each round has a great impact on the performance of asynchronous aggregation. In the proposed protocols, the transmission times of leaf nodes are randomized over an interval called the randomization interval. A method is proposed for adaptively determining the duration of the randomization interval.

The remainder of this chapter is organized as follows. The new asynchronous aggregation protocols are presented in Section 3.2. Section 3.3 presents simulation results evaluating the performance of the new asynchronous protocols and of synchronous aggregation. Section 3.4 concludes the chapter.

3.2 Asynchronous Aggregation

The main goal of this chapter is to design asynchronous aggregation protocols that maximize aggregation efficiency by ensuring that as much aggregation occurs as possible, while still providing timely arrival of aggregation results at the sink. Three asynchronous protocols are proposed in the following subsections, beginning with the simplest of these protocols, and then making enhancements that yield improved performance as shown by the performance results in Section 3.3.

The proposed protocols run above the network layer. Aggregation is performed as data packets are forwarded to the sink. The union of the routes to the sink forms

an aggregation tree with the sink as its root node. For simplicity, it is assumed that a node can aggregate data from its subtree, together with its own data, into a fixed-size packet.

3.2.1 Basic Asynchronous Aggregation Protocol

In the basic asynchronous protocol, each non-leaf node sets a timeout in each round that establishes the maximum time it will wait to receive data from its children. The timeout value is determined adaptively, based on the timings of packet receptions from its children in the immediately preceding round. The node transmits its data packet for this round to its parent (aggregating its own data with whatever it has received from its children) either when it hears from all of its children, or when the timeout expires, whichever occurs first.

For simplicity, it is assumed that all nodes agree on the same base time T_0 defining the beginning of the first round. (In Section 3.3.7, however, it is shown that the proposed protocols are tolerant of substantial variability in the values of T_0 used at different nodes.) To avoid concurrent transmissions, each node i (other than the sink node) picks a random value r_i between 0 and R at T_0, where R is a protocol parameter. At time $T_0 + r_i$, node i sends a packet containing its sensor data for the first round to its parent (r_i is uniformly distributed).

At each subsequent round j, each node i that is a leaf in the aggregation tree sends a packet containing its sensor data at time $T_0 + r_i + (j-1)\tau$, where τ is the time between successive sensor readings at each node (i.e., the reciprocal of the sensor sampling rate). Each non-leaf node operates as follows. Let L_i^j denote the time by which non-leaf node i receives the last packet for round j. Let TO_i^j denote the timeout for round j at node i. Node i sets its timeout for the second round to $TO_i^2 = L_i^1 + \tau$. Note that the sequence number for each round can be recycled so that only a few bits are necessary to keep track of the current round.

For round $j + 1 > 2$, the timeout of node i is updated as follows:

1. If node i received data packets for round j from all of its children before

TO_i^j, it sets the timeout for round $j+1$ to $TO_i^{j+1} = L_i^j + \tau + e$ (since a packet from each child should arrive approximately once every time τ). The protocol parameter e allows for some variance in the times at which packets are successfully transmitted.

2. If the timer for round j went off before node i received packets from all of its children, node i sets the timeout for round $j+1$ to $TO_i^{j+1} = TO_i^j + \tau$. If node i receives one or more packets for round j after time TO_i^j ("late arrivals"), it updates TO_i^{j+1} to $L_i^j + \tau$. Such late arrivals have been received too late to be aggregated in node i's round j transmission to its parent, and are simply dropped, since only up-to-date data is of interest in real-time monitoring.

Fig. 3.1 depicts how the basic asynchronous aggregation protocol works. In scenario 1, node 2 received the data packet for round j from node 1, its only child, before TO_2^j. The timeout of node 2 is set to $TO_2^{j+1} = L_2^j + \tau + e$ for round $j+1$, where L_2^j is the time by which node 2 received the packet for round j from node 1. In scenario 2, node 1's packet for round j was dropped, and node 2 transmitted at timeout. The timeout of node 2 is set to $TO_2^{j+1} = TO_2^j + \tau$ for round $j+1$. In scenario 3, node 2's timer for round j went off before node i received the packet from node 1. The packet from node 1 was received at time L_2^j as a late arrival. The timeout of node 2 is set to $TO_2^{j+1} = TO_2^j + \tau$ when node 2 transmitted for round j, and then updated to $TO_2^{j+1} = L_2^j + \tau$.

The choice of the protocol parameter e impacts the timeliness of the arrivals of data packets at the sink, and the number of late arrivals at the intermediate nodes in the aggregation tree. If e is set too small, timeouts may be set too aggressively, and data packets that experience normal variability in transmission times may arrive after the expiry of the respective timeout and be dropped. When e is set too large, latency may build up as nodes wait for data packets that will never be received owing to transmission failures. The experiments show, however, that e can be set to a fixed value that yields good performance over a wide range of conditions. In contrast, tuning the parameter R according to the particular network scenario can

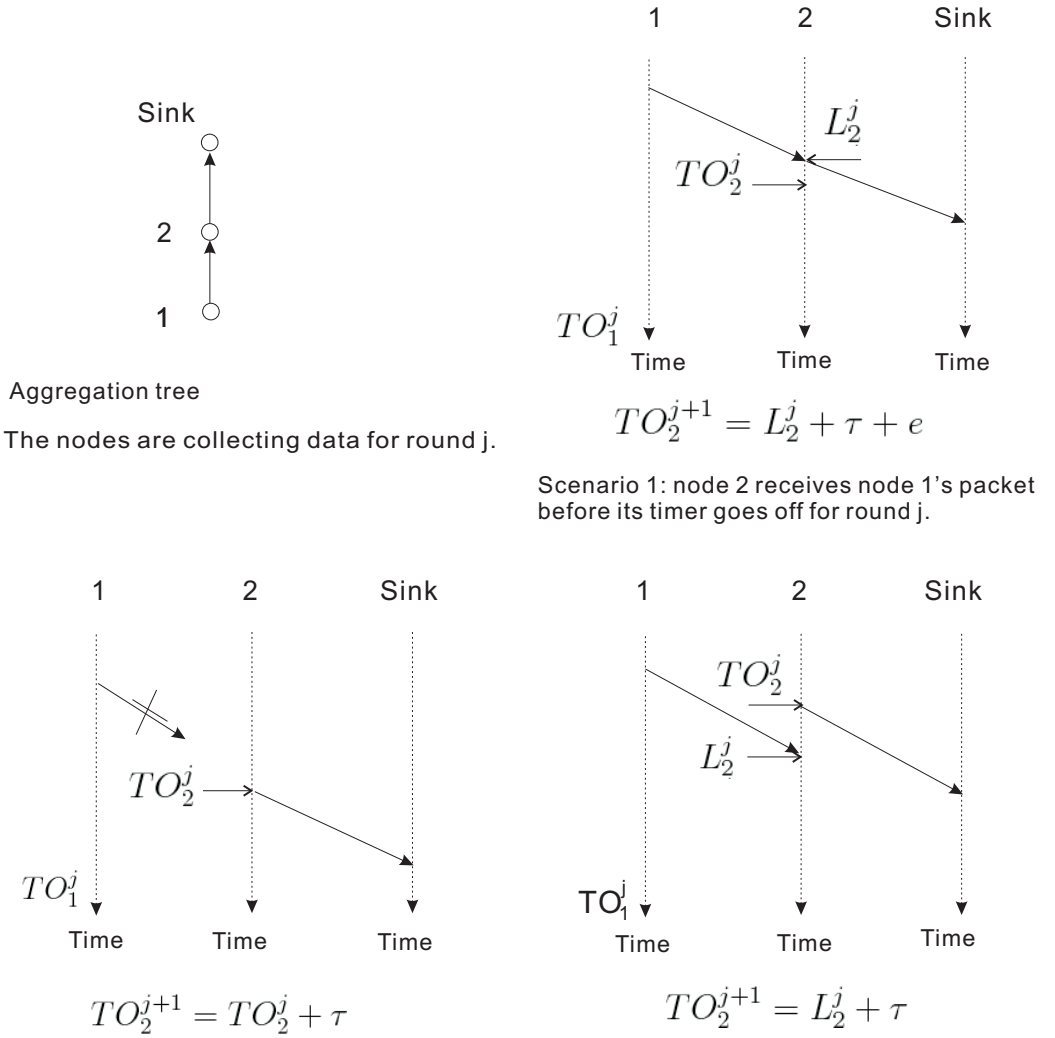


Figure 3.1: Basic Asynchronous Aggregation

yield substantial improvements in performance. A method is proposed in Section 3.2.3 to adaptively determine the value for R.

The above protocol supports adaptivity to dynamics in the topology of the aggregation tree, as long as nodes have some mechanism for dynamically altering their set of child nodes and their parent when necessary. The timing of transmissions can be quickly adjusted according to the above rules.

3.2.2 Asynchronous Aggregation Protocol with EWMA

While the basic protocol is straightforward, it may cause a "timeout chain" phenomenon under certain circumstances. Suppose, for example, that a node 1 has only one child, node 2, and only one grandchild, node 3. Suppose that node 2 receives the packet for round j from node 3 at time L_2^j, and sets its timeout for round $j+1$ to $L_2^j + \tau + e$. Suppose further that the aggregate sent by node 2 arrives at node 1 after a transmission delay d, causing node 1 to set its timeout for round $j+1$ to $L_2^j + d + \tau + e$. If the round $j+1$ packet from node 3 is not successfully received by node 2, node 2 will time out and send its packet for round $j+1$ at time $L_2^j + \tau + e$, as shown in Fig. 3.2. If the transmission delay of this packet exceeds d, node 1 will also time out, causing the packet to be a late arrival and be dropped.

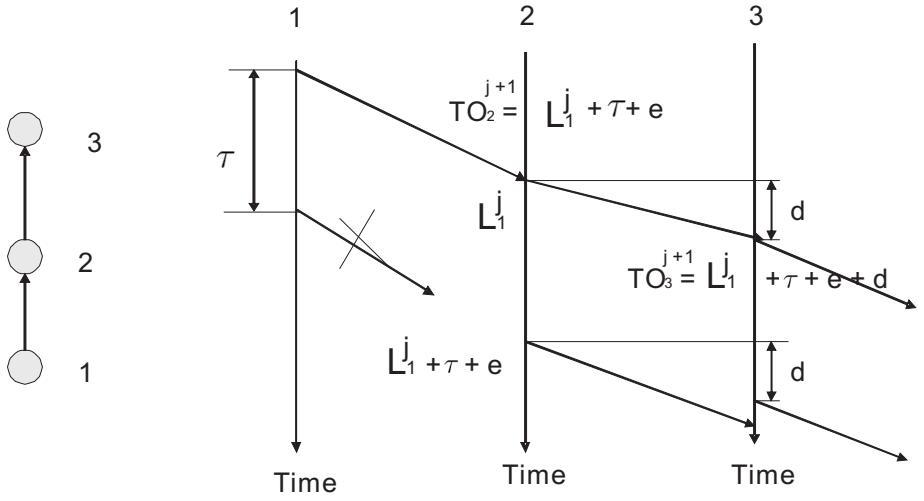


Figure 3.2: Timeout Chain

The main reason for the above phenomenon is that the timeout for the next round at a node is set too aggressively when packets are received from all children prior to timeout expiry. In the second asynchronous protocol, an Exponentially Weighted Moving Average (EWMA) strategy is used to adjust the timeout value in this case. Specifically, if a node i heard from all of its children before timeout in round j, it sets its timeout for round $j+1$ to $TO_i^{j+1} = (1-\delta)(TO_i^j + \tau) + \delta(L_i^j + \tau + e)$. The parameter δ controls how quickly the protocol reacts to changes in the network.

The above adjustment method bears some similarity to the Additive Increase Multiplicative Decrease (AIMD) algorithm in the Transmission Control Protocol (TCP). Both react slowly to "good news" while aggressively to "bad news". In this dissertation, the timeout for the next round is adjusted cautiously when packets are received from all children prior to timeout expiry, but more aggressively when there is a late arrival.

3.2.3 Adaptive Asynchronous Aggregation Protocol

It is important to randomize the transmission times of leaf nodes to avoid congestion at the beginning of each round. The parameter R controls the duration of the randomization interval. Recall that leaf node i sends its packet for round j at time $T_0 + r_i + (j-1)\tau$, where r_i is a random value between 0 and R. Choosing an appropriate value of R requires balancing the risk of congestion (if R is set too small) versus increased delay (if R is set too large). The best value is network dependent. In the third asynchronous aggregation protocol, R is determined adaptively and the r_i value for each node is changed when R is changed.

The adaptive protocol attempts to keep R within a certain range relative to the delay D from the beginning of a round, until the last of the data from that round is received at the sink. Intuitively, if R is large relative to D, a substantial portion of the delay might be due to the randomization delay at the leaf nodes. In this case, delay might be reduced if R was made smaller. If, on the other hand, R is small relative to D, R might be fruitfully increased, so as to better spread out the transmissions of the leaf nodes.

Specifically, the protocol maintains an EWMA average of the delay D, using $D_{ave} = \alpha D_{ave} + (1-\alpha)D^*$, where D^* is the latest measurement for the delay D and α is a parameter determining the weight given to the previous value of the average. Suppose the desired range of R/D_{ave} is $[\beta - \Delta, \beta + \Delta]$. When the sink observes that R/D_{ave} is out of range, it updates R as follows. If $R/D_{ave} < \beta - \Delta$, R is updated to $R = D_{ave}(\beta + \Delta)$. If $R/D_{ave} > \beta + \Delta$, R is updated to $R = D_{ave}(\beta - \Delta)$. The adjustment is more aggressive than setting the new value to

$R = D_{ave}\beta$ when R/D_{ave} is out of range. After a new R is selected, each leaf node picks a new random value between 0 and the new R as its randomization delay and adjusts its timeout value accordingly. The adjustment propagates to the parent nodes through packet reception information. Under normal conditions (no network congestion), after each adjustment, D_{ave} converges to a new value when the network stabilizes. If the new R is larger than the old value, the new D_{ave} is likely to be larger than the old value too. (Figures showing the impact of R can be found in Appendix A.) Eventually, the ratio R/D_{ave} falls into the range of $[\beta - \Delta, \beta + \Delta]$ as a result of the adjustment(s). Randomizing the transmission times of leaf nodes is to avoid congestion at the beginning of each round. When the nodes offer more load to the network than it can handle, congestion is inevitable, and randomizing the transmission times of leaf nodes cannot avoid congestion in this case. With suitable parameter value selections, changes to R with this protocol would be relatively rare. To inform the nodes about these changes, a simple method is to flood the new value throughout the network. Another possible solution is to piggyback the new value in the acknowledgements. No particular technique is modeled for communicating changes in R from the sink to the other network nodes in this dissertation.

3.3 Comparative Evaluation

3.3.1 Synchronous Aggregation

The synchronous aggregation protocol used here for comparison uses a synchronization structure similar to that in TAG [61] and the cascading timeout protocol [86]. In particular, it is assumed that each node i knows its hop count to the sink, h_i, and accordingly chooses its transmission interval within each round, as shown in Fig. 3.3. Let I be the duration of the interval and the tree height be H. For each round j, node i picks a random value r_i^j between 0 and $\lambda I, 0 \leq \lambda \leq 1$, aggregates the data it has received for this round and sends out its packet at $T_0 + \tau(j-1) + (H - h_i)I + r_i^j$. Randomizing transmissions over λI yields better performance than when all nodes

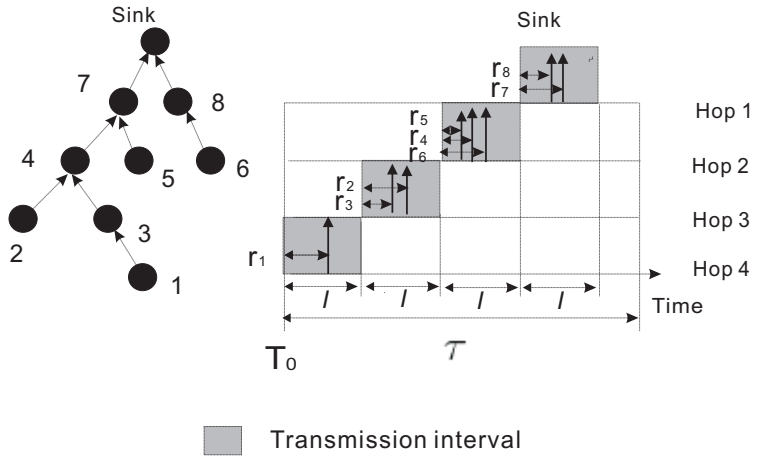


Figure 3.3: Synchronous Aggregation

at the same tree level attempt to transmit at the same time. Parameter λ is set to 0.8 in all experiments. Alternative values were tried, but did not yield better performance (see Fig. A.2). A packet is dropped if it arrives after the parent has sent out its aggregate for that round, as in the asynchronous protocols.

The duration of the interval is the decisive performance factor once the network configuration is fixed. In the performance evaluation experiments, different interval durations are used to explore the best achievable performance.

3.3.2 Goals, Metrics, and Methodology

The performance of the asynchronous protocols and the synchronous protocol is evaluated through ns2 (network simulator version 2) simulation[1]. The primary metrics of the performance evaluation are the end-to-end loss rate and the maximum data age. The end-to-end loss rate is equal to the ratio of the number of samples not included in the aggregates arriving at the sink to the total number of samples the nodes generate. The maximum data age is a measure of how old the data at the sink can be by the time the next samples arrive, and is approximated as τ plus the *maximum data collection delay*, where the maximum data collection delay for each round is the maximum delay from when a sensor value is captured, until the corresponding aggregate arrives at the sink. The average of the maximum data ages

[1]http://www.isi.edu/nsnam/ns/

over multiple rounds is plotted in the figures. An additional metric for which some results are presented is the average number of MAC layer data packet transmissions per round, which yields insight into relative energy usage. Other possible metrics that are not measured include the average data age or the median data age. The maximum data age is chosen because it reflects how old the data at the sink can be in the worst case.

Different sensor networks are generated by randomly scattering nodes in square areas with different sizes. The sink is the node closest to the center of the network unless otherwise stated, and the aggregation tree is constructed as a shortest path tree. All figures except Fig. 3.7, 3.12, and 3.13 show results for a sensor network with 160 nodes deployed over a 250 meter by 250 meter area. Fig. 3.4 shows how the 160 nodes are spread inside the area. The physical layer packet loss rate is specified as a simulation input parameter. The uniform random error model is used for all experiments except those in which the two-state Gilbert error model is used to simulate channel errors (Section 3.3.7).

Deciding on an appropriate range for the physical layer loss rate is a particularly challenging issue. Opinions vary on the characteristics of the wireless channel under different operating characteristics. Zhao and Govindan place approximately sixty MICA motes in a chain topology, with one node at the head of the chain as the sender and all other nodes as receivers [114]. Measurements are taken from three different environments: an office building (a 2 meter by 40 meter hallway), a natural habitat (a 150 meter by 150 meter segment of a local state park), and an empty parking lot (150 meter by 150 meter). The results show heavy tails in the distributions of packet loss at the physical layer. The indoor experiments show that half of the links experience more than 10% packet loss at the physical layer, and a third more than 30%. Such variability shows the existence of a *gray area* within the communication range of a node. Packet loss rate is high for the receivers that are located in the grey area of the sender, and the area is almost a third of the communication range in some environments. Packet reception also varies significantly over time in the grey area. As pointed out by Zhao and Govindan, the actual behavior of these environments

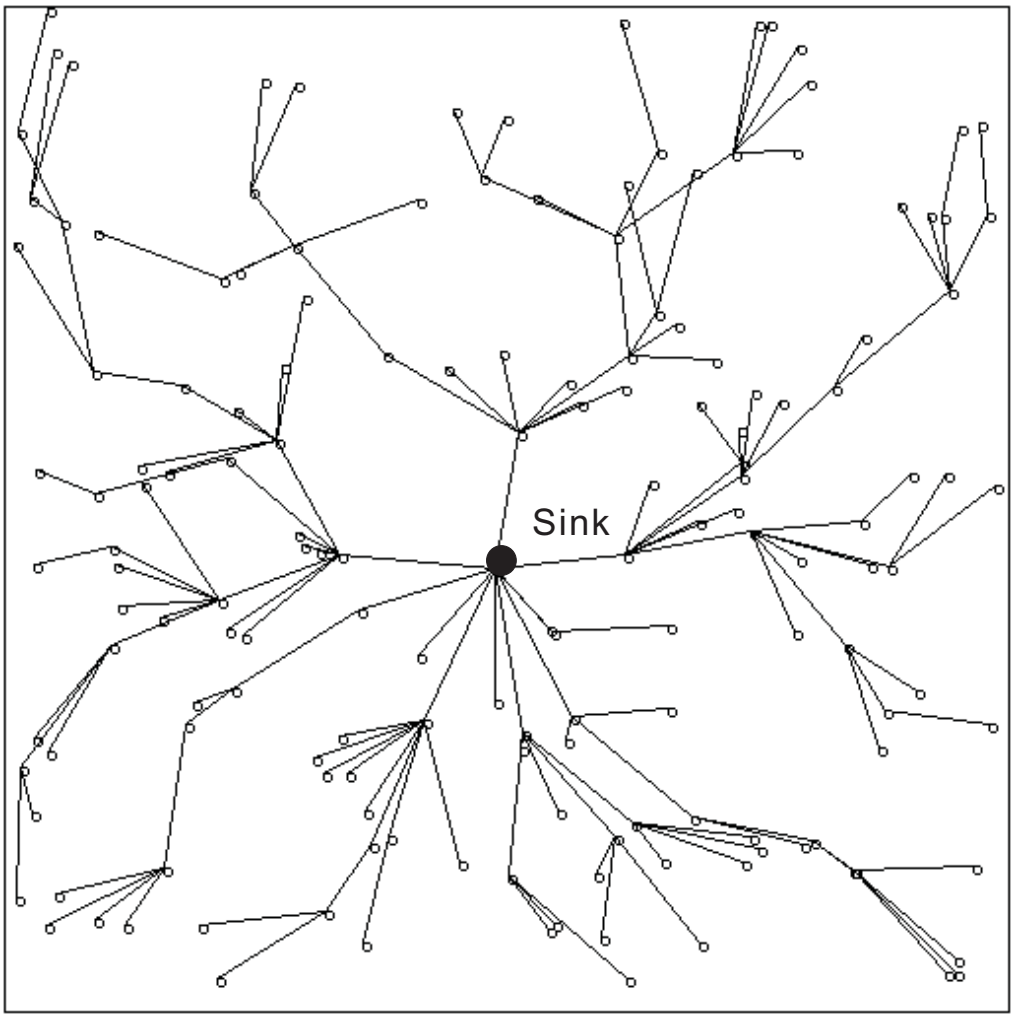


Figure 3.4: Sensor Field with 160 Nodes in a 250m by 250m Area

is likely to be even worse than that reported in their paper, since the experiments were done during quiet times, for example, late at night in the indoor environment.

Marina *et al.* collect measurements in both indoor and outdoor environments to examine the characteristics of the IEEE 802.15.4 communication channel [73]. The indoor measurements are done in a softly partitioned office environment without serious obstacles for signal propagation. The results show that for the indoor environment, packet loss rate is less than 3.5% at 32 meter distance. However, the indoor packet loss rate increases dramatically with longer distance, from 10% to 90% when the distance increases from 40 meters to 70 meters. Considering that the indoor measurements are collected in a relatively benign environment, worse link quality is

expected in enviroments with more interference, for example, a long hallway, where multipath reflections from the walls can be serious. For the outdoor scenario, the results show that the packet loss rate increases from 10% to 90% as the distance increases from 70 meters to 160 meters. The outdoor experiments are done with a fixed packet size of 20 bytes. Worse packet loss rate is expected with larger packets.

The impact of physical layer loss rate on the aggregation protocols is studied in the performance evaluation. When a large number of nodes are randomly deployed in an area, especially in a harsh environment, the communication links are expected to show high variability. Many links may have high loss rate due to increased interference from the surrounding, such as passing animals, or because the receiver is within the gray area of the communication range of the sender. The variability of the link quality is not modelled in this dissertation. Instead, it is assumed that all nodes have the same loss probability. Three different physical layer loss rates, 10%, 20%, and 30% are used in the simultions when packet loss is assumed to be randomly distributed. Future work will consider differing loss patterns at each node.

An 802.11 MAC layer is simulated, without RTS/CTS [102], with a transmission range of 40 meters and rate of 2Mbps. This matches closely with the non-beacon mode of 802.15.4 [20]. In the non-beacon mode, 802.15.4 MAC protocol is basically a CSMA/CA protocol. In the simulations, data packet size is fixed at 52 bytes. A data packet is retransmitted up to three times before being discarded if an ACK is not received.

For all protocols except adaptive asynchronous aggregation, each simulation run lasts 1,150 sampling period durations, with the initial 200 and the last 50 sampling period durations removed from the measurements. With a high sampling rate, congestion may build up in the networks. The first 200 rounds allow the network to warm up. Transmission delay increases dramatically when network congestion happens. The sampling value of one round may arrive at the sink more than ten rounds later after the beginning of its own round. Sampling values generated for the last 50 rounds are taken out from the measurements. For adaptive asynchronous aggregation, each simulation run lasts 1,550 sampling period durations, with the initial

600 and the last 50 sampling period durations removed from the measurements. The simulations with adaptive asynchronous aggregation last longer to allow time for the computation of the randomization delay and the convergence of the network. Here the warm-up and cool-down times are conservatively chosen to ensure the correctness of the measurements. It actually takes much less time for the network to stabilize. The results reported below are averaged over 5 simulation runs.

3.3.3 Parameter Analysis

Table 3.1 lists the parameters that are used in the performance evaluation. While the asynchronous protocols have more protocol parameters than the synchronous protocol, experimental results show that e and δ can be easily fixed at 0.1 seconds and 0.05 respectively for all network settings. For adaptive asynchronous aggregation, α, β, and Δ are fixed at 0.875, 0.7, and 0.15 respectively. This section will look at the effects of the following parameters: e, R and I. More figures showing the impact of the protocol parameters are shown in Appendix A.

First consider the parameter e, Fig. 3.5 plots performance results of asynchronous aggregation with EWMA for the sensor network with 160 nodes randomly deployed in a 250 meter by 250 meter area (see Fig. 3.4). The sampling period duration τ and R are fixed at 0.5 seconds and 0.3 seconds, respectively. Fig. 3.5 shows that fixing e at 0.1 seconds yields good performance for different physical layer loss rates. Experiments are also conducted with different sensor networks and the results show that good performance is achieved for all sensor networks when e is 0.1 seconds. Thus, e will be set to 0.1 seconds for the remaining experiments.

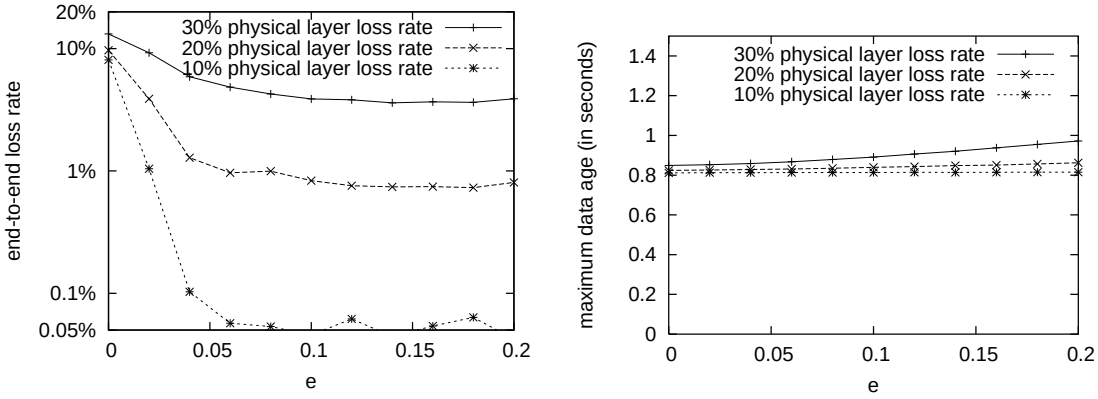
Fig. 3.6 plots performance results of asynchronous aggregation with EWMA for different R. When R equals 0, all leaf nodes attempt to transmit at the beginning of each round. More packets are dropped because of collisions. The end-to-end loss rate drops as the duration of the randomization interval gets longer. When the duration of the randomization interval is long enough that the number of collisions are negligible, increasing R only leads to longer delay. Fig. 3.6 shows that the choice of R has a great impact on the performance of asynchronous aggregation. The value

τ	The time between successive sensor readings at each node.
R	The duration of the randomization interval for the asynchronous protocols.
PLR	The physical layer loss rate for the uniform random error model.
N	The number of nodes inside the network.
S	The size of the square area inside which N nodes are scattered.
α	Protocol parameter used in adaptive asynchronous aggregation to calculate the EWMA average of the delay D. It determines the weight given to the previous value of the average. The value of α is fixed at 0.875 unless otherwise stated.
β	Protocol parameter for adaptive asynchronous aggregation. The desired range of R/D_{ave} is $[\beta - \Delta, \beta + \Delta]$. The value of β is fixed at 0.7 unless otherwise stated.
Δ	Protocol parameter for adaptive asynchronous aggregation. The desired range of R/D_{ave} is $[\beta - \Delta, \beta + \Delta]$. The value of Δ is fixed at 0.15 unless otherwise stated.
e	Protocol parameter for all three asynchronous aggregation protocols. The value of e is fixed at 0.1 seconds unless otherwise stated.
δ	Protocol parameter used in asynchronous aggregation with EWMA and adaptive asynchronous aggregation to calculate the timeout for the next round. It controls how quickly the protocol reacts to changes in the network. The value of δ is fixed at 0.05 unless otherwise stated.
λ	Protocol parameter for synchronous aggregation. The transmission times of the nodes at the same tree level are scattered over λI for each round. The value of λ is fixed at 0.8 in all experiments.

Table 3.1: Simulation Parameters for Chapter 3

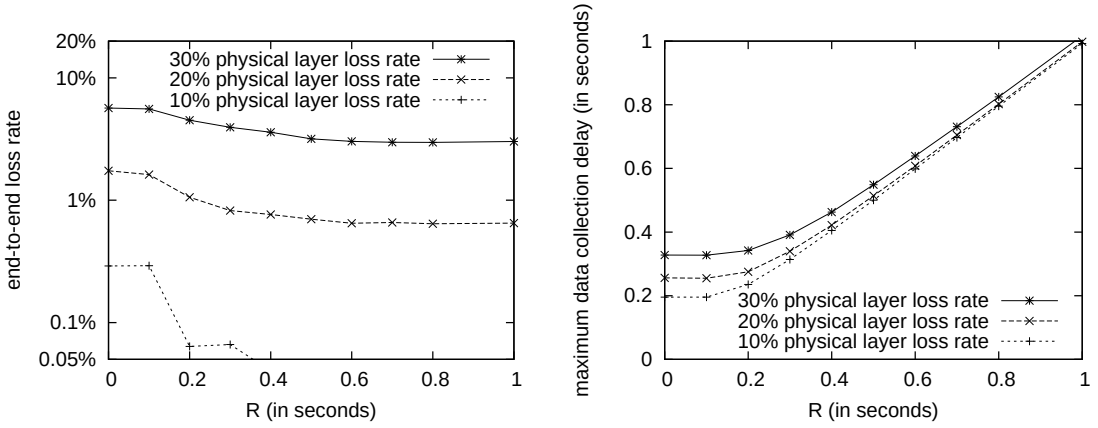
of R is adaptively adjusted in adaptive asynchronous aggregation.

The performance of the synchronous protocol is very sensitive to the choice of the duration of the interval. When the duration of the interval is too short, not all nodes at the same tree level can get their packets through to the next level during their transmission interval. Packets that arrive after the receiver has sent out its own packet are not aggregated. In addition, when the transmission interval is short, more packets get dropped because of collision as they contend to transmit during the interval. Increasing I decreases the end-to-end loss rate at first, but after I is large enough, no more improvement on the end-to-end loss rare can be made. At this stage, increasing I only leads to increased maximum data age.



(a) End-to-End loss rate (b) Maximum Data Age

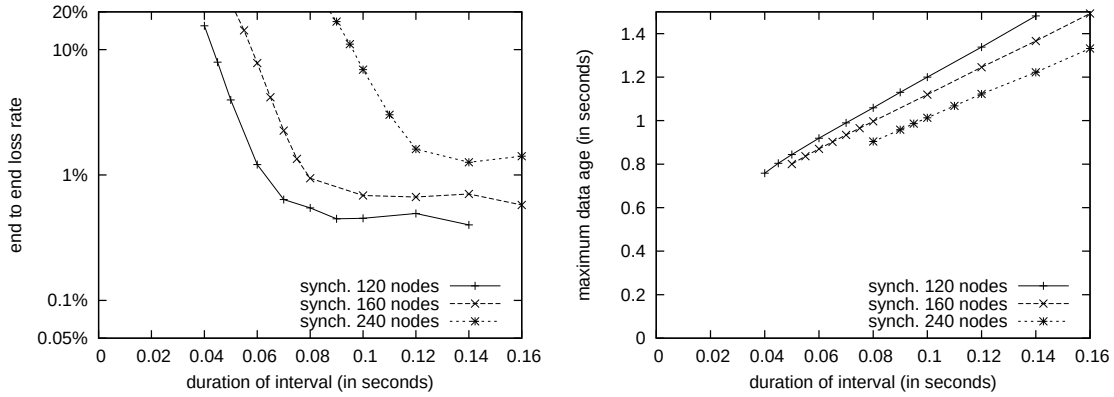
Figure 3.5: Impact of Parameter e ($\tau = 0.5$ sec., N = 160, S = 250m $\times$ 250m, R = 0.3 sec.)



(a) End-to-End Loss Rate (b) Maximum Data Collection Delay

Figure 3.6: Impact of Parameter R on Asynchronous Aggregation with EWMA ($\tau = 5$ sec., N = 160, S = 250m $\times$ 250m)

Fig. 3.7 plots performance results of synchronous aggregation for sensor networks with 120, 160, or 240 nodes randomly deployed in a 250 meter by 250 meter area. The sampling period duration is set to 0.5 seconds, and the physical layer loss rate is fixed at 20%. Fig. 3.7 shows that setting I to 0.08 seconds yields good performance for sensor networks with 120 and 160 nodes, but results in high end-to-end loss rate for the sensor network with 240 nodes. Setting I to 0.12 seconds decreases the end-to end loss rate for the sensor network with 240 nodes, but results in excessive maximum data age for the other two sensor networks. The results suggest that aggregation tree structure has a great impact on the choice of I. In practice, it may be difficult



(a) End-to-End Loss Rate

(b) Maximum Data Age

Figure 3.7: Impact of Parameter I ($\tau = 0.5$ sec., S = 250m $\times$ 250m, R = 0.3 sec., PLR = 20%)

to set this parameter in a manner yielding consistently good performance.

3.3.4 Principal Performance Comparison

Experiments for this section use the sensor field with 160 nodes randomly deployed in a 250 meter by 250 meter area. All figures in this section, except Fig. 3.11, plot results for 20% physical layer loss rate. Three sampling rates are used in the experiments; τ equals to 0.25, 0.5, and 0.75 seconds.

For basic asynchronous and asynchronous aggregation with EWMA, R is varied from 0 to τ in the simulations. For each R, the end-to-end loss rate and the maximum data age of these two protocols are measured. The impact of R has been studied in Section 3.3.3. In Fig. 3.6, the maximum data age and the end-to-end loss rate are plotted in two separate figures. It can be useful to see what the end-to-end loss rate is for a certain maximum data age, but this is hard to determine from Fig. 3.6(a) and Fig. 3.6(b).

For synchronous aggregation, the sampling rate is fixed, and I is varied in the simulations. The impact of I has been shown in Section 3.3.3. Similarly, when the maximum data age and the end-to-end loss rate are plotted in two separate figures, as seen in Fig. 3.7, it is hard to tell what the end-to-end loss rate is for a certain maximum data age. To address this issue, all figures in the rest of this chapter plot

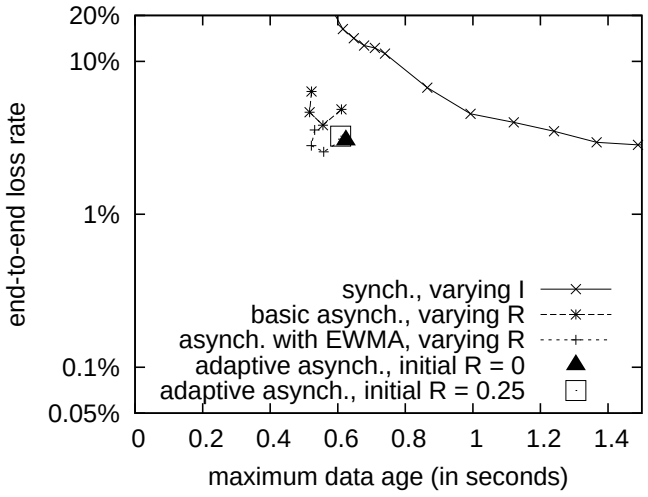
the end-to-end loss rate or the average number of MAC layer data packets per round as a function of the maximum data age.

Fig. 3.8 (a), (b), and (c) plot performance results of the synchronous and asynchronous protocols at three different sampling rates. Each point for basic asynchronous and asynchronous aggregation with EWMA is achieved at a specific R. For $\tau = 0.25$ seconds, $R \in \{0, 0.1, 0.2, 0.25\}$. For $\tau = 0.5$ and 0.75 seconds, $R \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75\}$ respectively. Similarly, each point for the synchronous aggregation protocol is achieved at a specific I. Two different values, 0 and τ, are used as the initial value for R in adaptive asynchronous aggregation. For all three sampling rates, asynchronous aggregation achieves lower maximum data age than synchronous aggregation for a given end-to-end loss rate. When τ equals 0.25 seconds, the asynchronous aggregation protocols are able to lower the maximum data age by 50% for 4% end-to-end loss rate.

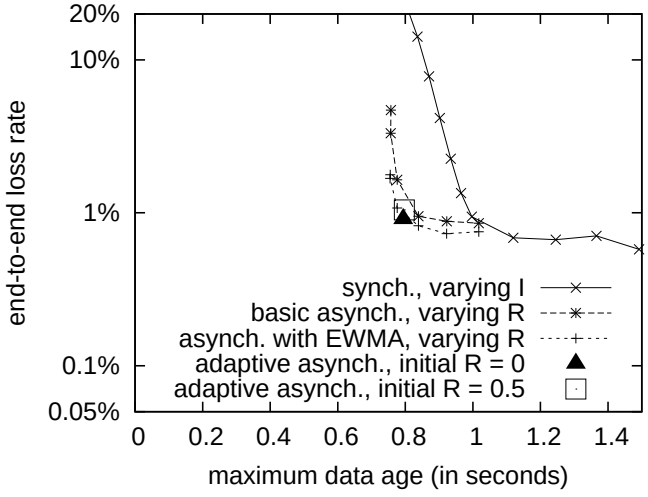
The performance of basic asynchronous and asynchronous aggregation with EWMA is very close when τ is 0.5 and 0.75 seconds. Fig. 3.8 shows that both protocols achieve an end-to-end loss rate close to 1% at similar maximum data age. At such a low end-to-end loss rate, almost all packet transmissions are triggered when a parent hears from all children. The timeout strategy does not have much impact. When τ is 0.25 seconds, the end-to-end loss rate gets higher and the difference between the two protocols becomes bigger as more packet transmissions are triggered by timeouts.

The choice of R has a great impact on the performance of asynchronous aggregation. When R is set to 0, all leaf nodes try to transmit at the very beginning of each round and may cause a lot of packet collisions. Increasing R reduces the end-to-end loss rate at first, but when the duration of the randomization interval is longer enough that the number of collisions are negligible, increasing R only leads to longer delay. Fig. 3.8 shows that for different initial values of R, adaptive asynchronous aggregation is able to adjust R to a good value so that the maximum data age is close to the minimum value while the end-to-end loss rate is kept low.

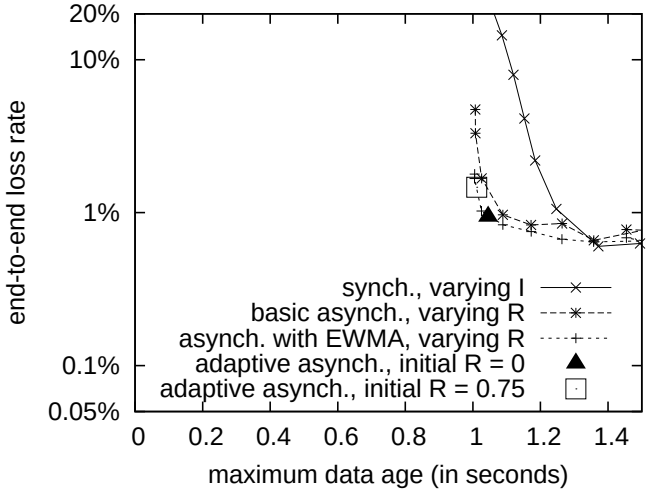
For adaptive asynchronous aggregation, the initial value of R is set to 0 and τ in the simulations. The value of R is adaptively adjusted during the simulations so



(a) $\tau = 0.25$ seconds



(b) $\tau = 0.5$ seconds



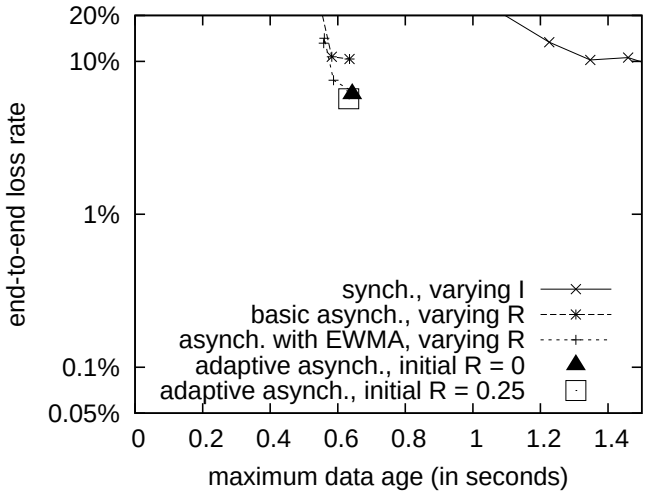
(c) $\tau = 0.75$ seconds

Figure 3.8: Performance with Sink at Center (N = 160, S = 250m × 250m, PLR = 20%)

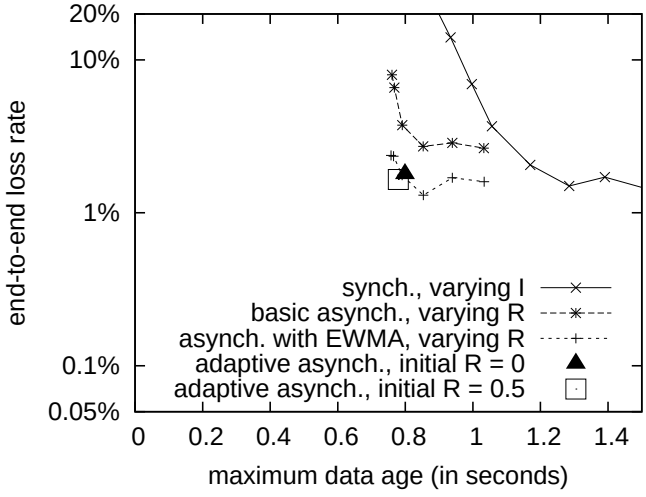
that the ratio of R to D_{ave} is within the range of $[\beta-\Delta, \beta+\Delta]$ (Parameters β and Δ are fixed at 0.7 and 0.15 in the simulations). Given different initial values for R, the final R may converge to different values, as long as the sink observes that the ratio of R to D_{ave} is within the desired range. As seen in Fig. 3.8, for $\tau = 0.25$ seconds and 0.5 seconds, the performance of adaptive asynchronous aggregation converges to basically the same point for the two initial values. For $\tau = 0.75$ seconds, adaptive asynchronous aggregation adjusts R to two different values for the two initial values, yielding slightly different performance for each initial value.

Fig. 3.9 shows the performance of the considered protocols with alternative sink placement. The same sensor network as for Fig. 3.8 is used, but with the sink at the corner. Fig. 3.9 (compare with Fig. 3.8) shows that the maximum data age of the synchronous protocol substantially increases when the sink is located at the corner, while that of the asynchronous protocols stays around the same. A close look at the tree structure shows that the aggregation tree with the sink at the corner is more than twice as long as the one with the sink in the center, but the maximum number of nodes at the same tree level is quite similar in both cases. Thus, for synchronous aggregation, a similar interval duration is required in both cases, but twice as many intervals are required when the sink is at the corner.

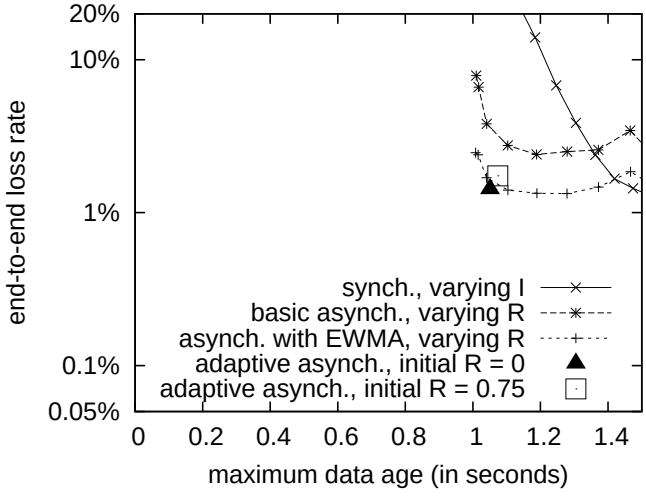
Another observation from Fig. 3.8 and Fig. 3.9 is that with the same 0.5 and 0.75 second sampling periods, asynchronous aggregation with EWMA performs much better than basic asynchronous aggregation with the sink at the corner. For $\tau = 0.75$, asynchronous aggregation with EWMA is able to reduce the end-to-end loss by 40% to 70% for a given maximum data age. The reason for the difference can be traced back to the tree structure as well. When the sink is located at the corner, the sink only has four children and only one of these children has three children. Moreover, only one of these three grandchildren of the sink has its own children. The performance of basic asynchronous aggregation is very susceptible to the "timeout chain" phenomenon mentioned in Section 3.2.2 with such a tree structure. The number of late arrivals now differs enough to make a more significant difference in the end-to-end loss rate. The relative difference between basic asynchronous and asynchronous



(a) $\tau = 0.25$ seconds



(b) $\tau = 0.5$ seconds



(c) $\tau = 0.75$ seconds

Figure 3.9: Performance with Sink at Corner (N = 160, S = 250m $\times$ 250m, PLR = 20%)

aggregation with EWMA, however, does not vary much with different sink placement when τ is 0.25 seconds. This is because packet losses due to congestion now play an important role in the network. The loss caused by the defects of basic asynchronous aggregation is less dominant. With the sink at the corner, adaptive asynchronous aggregation is still able to adjust R to a good value so that the maximum data age is close to the minimum value while the end-to-end loss rate is kept low, as seen in Fig. 3.9.

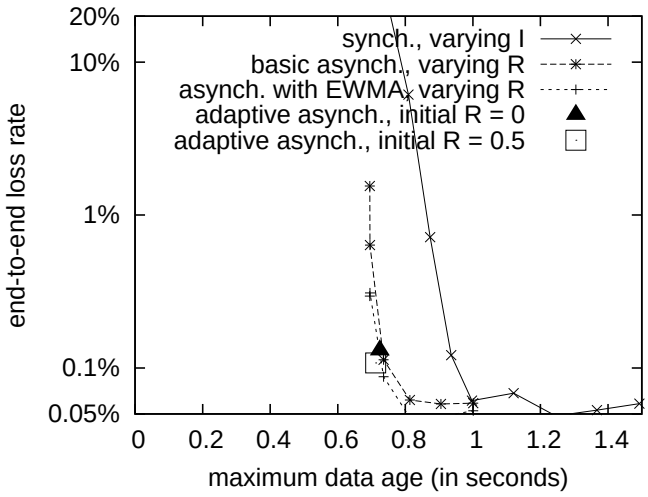
3.3.5 Impact of Physical Layer Loss

All figures in this section show results for the sensor network with 160 nodes deployed over a 250 meter by 250 meter area. The sampling period duration τ is fixed at 0.5 seconds in all experiments. Fig. 3.10 plots performance results of the protocols for 10% and 30% physical layer loss rate; all other figures in this section plot results for 20% physical layer loss rate.

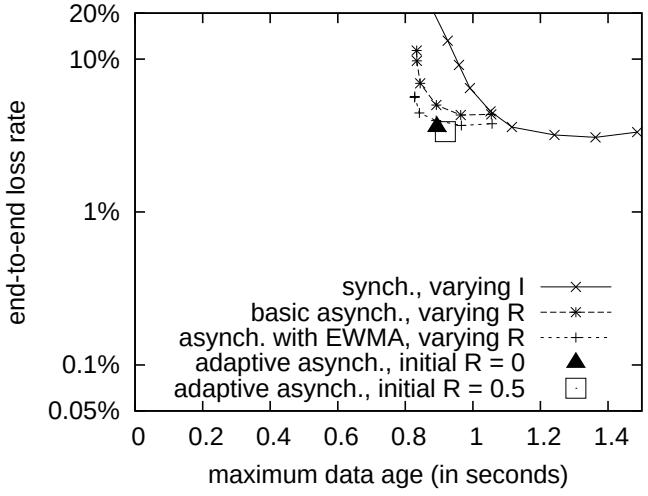
Fig. 3.10 plots performance results with 10% and 30% physical layer loss rate. The maximum data age and the end-to-end loss rate of both synchronous and asynchronous aggregation get worse as the physical layer loss rate increases. Comparing Fig. 3.10(a) with Fig. 3.10(b), the results show that physical layer loss rate has a greater impact on end-to-end loss rate than on the maximum data age. With 10% loss rate, the protocols can achieve an end-to-end loss rate as low as 0.05%. With 30% loss rate, the end-to-end loss rate is about 3% in the best case. The asynchronous protocols can provide a maximum data age around 0.7 seconds with 10% physical layer loss rate. The number inceases by just 0.1 seconds to around 0.8 seconds with 30% physical layer loss rate. Asynchronous aggregation outperforms synchronous aggregation for both physical layer loss rates.

The value of R is adaptively adjusted in adaptive asynchronous aggregation to keep R/D_{ave} within a certain range. For 10% physical layer loss rate, the maximum data age of adaptive asynchronous aggregation is very close to the minimum value, as seen in Fig. 3.10(a). Comparing with the adaptive protocol, basic asynchronous and asynchronous aggregation with EWMA are able to achieve lower end-to-end

loss rate with increased maximum data age. For 30% physical layer loss rate, Fig. 3.10(b) shows that the end-to-end loss rate of adaptive asynchronous aggregation is very close to the minimum value. Basic asynchronous and asynchronous aggregation with EWMA are able to achieve lower maximum data age with higher end-to-end loss rate in this case. For both physical loss rates, adaptive asynchronous aggregation is able to ajust R to achieve a good tradeoff between the end-to-end loss rate and the maximum data age.



(a) PLR = 10%

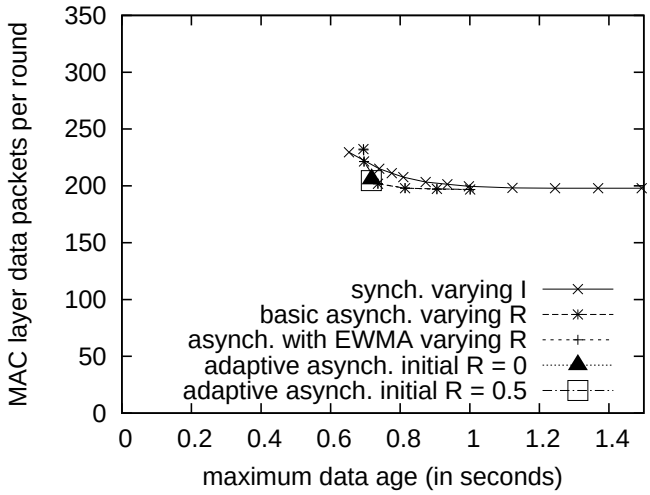


(b) PLR = 30%

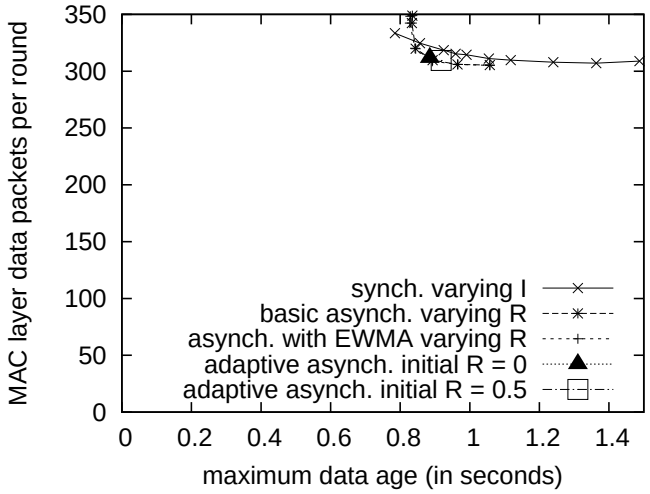
Figure 3.10: Impact of Physical Layer Loss Rate (τ = 0.5 sec., N = 160, S = 250m $\times$ 250m)

Fig. 3.11 shows the average number of MAC layer data packet transmission per

round of the considered protocols with 10% and 30% physical layer loss rate. Fig. 3.11 shows that the performance improvements shown in Fig. 3.10 are achieved without impact on the number of MAC layer packet transmissions. The number of MAC layer data packet transmissions is very similar with all of the considered protocols, and in fact even slightly better with the asynchronous protocols when the end-to-end loss rate is low.



(a) PLR = 10%



(b) PLR = 30%

Figure 3.11: Average Number of MAC Layer Data Packet Transmissions per Round (τ = 0.5 sec., N = 160, S = 250m $\times$ 250m)

3.3.6 Impact of Density

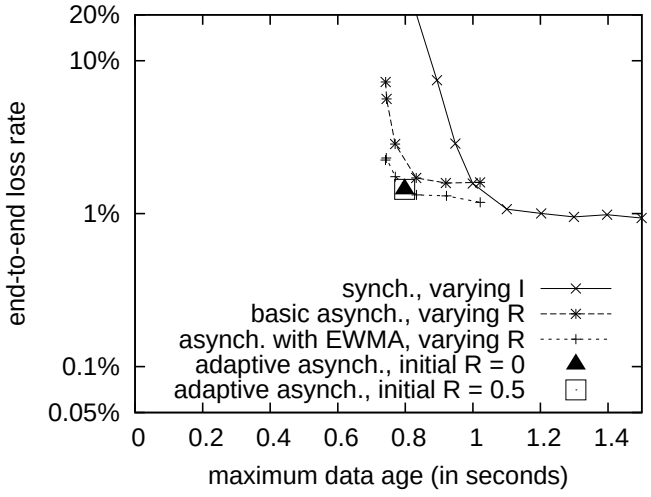
To study the impact of area size, two sensor networks are generated with 160 nodes randomly deployed in a 200 meter by 200 meter area and a 350 meter by 350 meter area. Fig. 3.12 shows performance results for these two sensor networks. When the number of nodes is fixed, the performance of the asynchronous protocols is not very sensitive to the size of the area. For the synchronous protocol, the average data collection delay increases as the tree gets longer and skinnier with a lower density. The maximum data age increases accordingly. The performance improvement asynchronous aggregation achieves over synchronous aggregation increases as the size of the area increases (density decreases), as shown in Fig. 3.12. For 2% end-to-end loss rate, the asynchronous protocols are able to lower the maximum data age by around 20% and 10% for the sensor network with a 350 meter by 350 meter area and a 200 meter by 200 meter area, respectively.

Two more sensor networks are generated by randomly placing 120 and 240 nodes in a 250 meter by 250 meter area, respectively. When the size of the area is fixed, the maximum data age for all of the considered protocols increases as the number of nodes increases. As seen in Fig. 3.13, the asynchronous protocols outperform the synchronous protocol for different numbers of nodes. Again, the performance improvement asynchronous aggregation achieves over synchronous aggregation increases as the number of nodes (density) decreases.

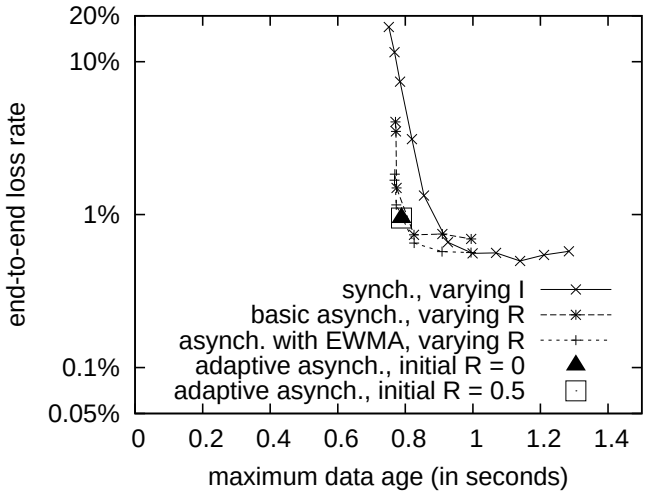
3.3.7 Other Factors

Two more factors that were examined were clock shift between sensor nodes and packet error distribution over time. The remainder of this section describes the experiments and evaluation of the effect of these factors.

For both synchronous and asynchronous aggregation, it was assumed that there is a common base time T_0 that defines the beginning of the first period at all sensor nodes. This assumption is relaxed in Fig. 3.14 by assuming that there is some variable clock shift away from this common base time, so that different nodes consider



(a) S = 350m × 350m

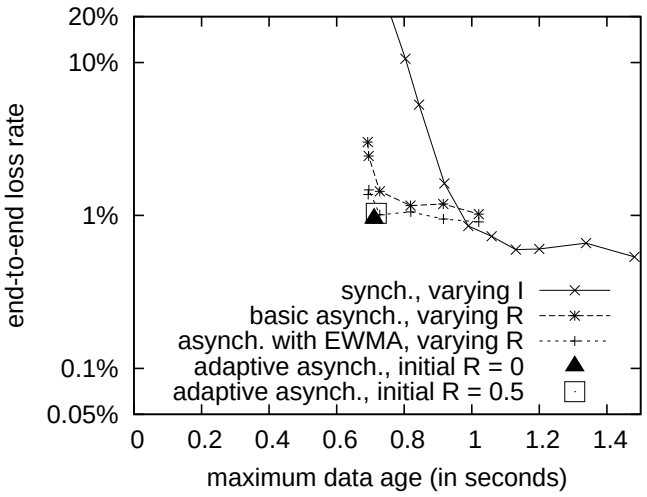


(b) S = 200m × 200m

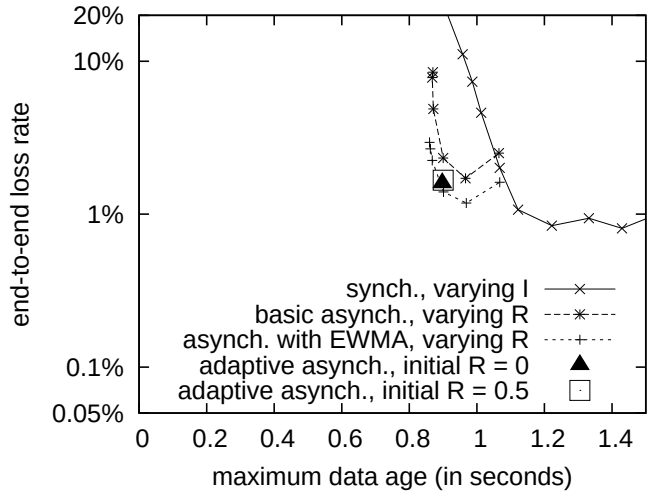
Figure 3.12: Impact of Size of Area (τ = 0.5 sec., N = 160, PLR = 20%)

the first period to begin at somewhat different times. The clocks of the nodes shift uniformly within a range of [−50 ms, 50 ms]. Fig. 3.14 (compare with Fig. 3.8(b)) shows that the asynchronous protocols are much more tolerant of clock shift than the synchronous protocol.

Fig. 3.16 considers the impact of a more bursty physical layer packet loss process on the relative performance of the aggregation protocols. As illustrated in Fig. 3.15, the two-state Gilbert error model is used, with a "good" state in which there is no physical layer packet loss, and a "bad" state in which there is a 40% physical layer packet loss rate. When in each state, after a time duration of 5 seconds, a state



(a) N = 120



(b) N = 240

Figure 3.13: Impact of Number of Nodes (τ = 0.5 sec., S = 250m $\times$ 250m, PLR = 20%)

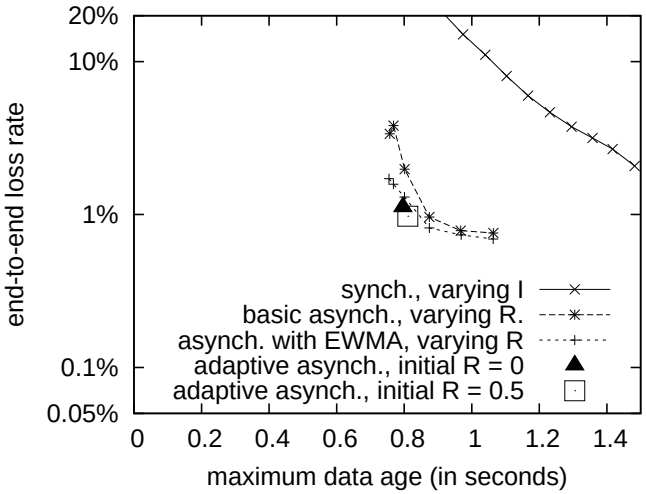


Figure 3.14: Performance with Clock Shift ($\tau = 0.5$ sec., N = 160, S = 250m $\times$ 250m, PLR = 20%)

transition decision is made with 20% probability of moving to the other state. Each node independently transits between the two states. As seen in the figure, the relative performance of the various protocols is qualitatively consistent with that observed in the earlier experiments. With the Gilbert model, the asynchronous aggregation protocols are still able to achieve lower loss rate for a given end-to-end loss rate. Similar results have been obtained with other settings of the Gilbert error model parameters.

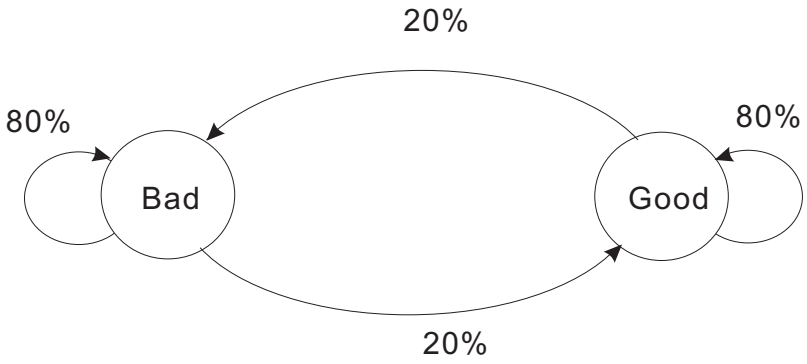


Figure 3.15: Two-State Gilbert Error Model (Discrete Time Model)

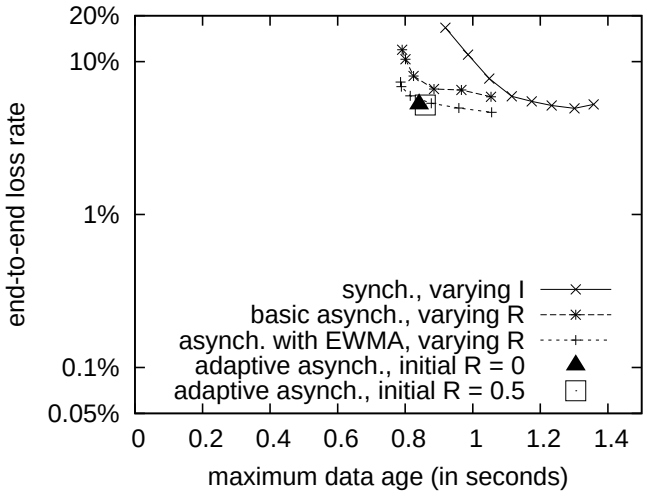


Figure 3.16: Performance with Two-state Gilbert Error Model ($\tau = 0.5$ sec., N = 160, S = 250m $\times$ 250m)

3.4 Summary

This chapter presents three asynchronous protocols and compares them against each other and against synchronous aggregation for the context of real-time monitoring systems. Simulation results show that asynchronous aggregation outperforms synchronous aggregation in its ability to keep data "current" while achieving a low end-to-end loss rate. Results also show that the per-node transmission adaptation strategy is crucial in asynchronous aggregation. Asynchronous aggregation with EWMA outperforms basic asynchronous. Randomizing the transmission times of leaf nodes to avoid congestion at the beginning of each round, and the duration of the randomization interval, have a great impact on the performance of asynchronous aggregation. In adaptive asynchronous aggregation, the duration of the randomization interval is adaptively adjusted to achieve a good tradeoff between the maximum data age and the end-to-end loss rate.

Chapter 4

Unicast-based vs. Broadcast-based Data Aggregation

4.1 Motivation

This chapter considers both real-time and delay-tolerant monitoring applications that collect data periodically. Unlike real-time monitoring, delay-tolerant monitoring can tolerate a long delay, but may demand a low end-to-end loss rate.

Most prior aggregation protocols, including the protocols studied in Chapter 3, use unicast transmission. These protocols achieve reliability using an acknowledgement/retransmission (ARQ) facility, as provided by the link layer for example. The prior work concerning aggregation protocols using broadcast transmission has focused on protocol mechanisms for "duplicate sensitive" aggregation [15, 32, 66], in which the sink must never receive multiple aggregates that include the same sensor value (or the same share of a sensor value, if "value splitting" [61] is employed).

This chapter examines broadcast-based aggregation mechanisms that minimize the number of packet transmissions, and rely on multipath delivery rather than ARQ for reliability. When a data packet is transmitted using unicast, at lease two MAC layer transmissions are required for each successful data packet transmission, one for data packet transmission and one for acknowledgement. MAC layer packet retransmission is used for packet recovery in unicast transmission. The sender retransmits the data packet until an acknowledgment is received or a predefined number of retransmissions has been reached. In the proposed broadcast-based aggregation protocols, each node broadcasts at most twice for each round. The second broadcast is

important to improve reliability.

In the proposed broadcast-based protocols, it is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates that include the same sensor value, because aggregation is duplicate insensitive (e.g., only the maximum sensor value is needed), or duplicates can be filtered. For example, when each aggregate is a concatenation of sensor values, the sink can pick up duplicates and ignore them.

Both synchronous and asynchronous broadcast-based aggregation protocols are developed. They take advantage of the fact that multiple downstream nodes may be potential receivers of a single broadcast transmission, and of the ability of a node to listen to and analyze transmissions from neighbouring nodes so as to determine which (if any) have included that node's broadcast data in their own transmissions. The latter ability is used as a substitute for acknowledgements. In addition to the reliability improvement that arises from potential multipath delivery to the sink, improved reliability is achieved through use of two-phase protocols in which a node may repeat (once) its broadcast.

The new broadcast-based aggregation protocols are evaluated and compared with the unicast-based protocols mostly using simulation in the context of both real-time and delay-tolerant data collection. For real-time scenarios, this chapter investigates whether broadcast-based protocols can be employed to maximize the achievable sampling/collection rate and minimize the collection delay with some modest amount of data loss. This chapter also considers the question of whether such protocols can achieve better performance in these respects than unicast-based protocols. For delay-tolerant applications, the performance comparison is focused on energy cost and end-to-end loss. Both the uniform random error model and the two-state Gilbert error model are used to simulate packet loss in the performance comparison.

Rather than assuming some particular aggregation function, two extreme cases are considered. In one of these, it is assumed that sensor data can be aggregated into packets of size that is independent of the number of values being aggregated. In the other, the required packet size is assumed to increase linearly with the number of values included in the aggregate. The first case applies with duplicate-insensitive ag-

gregation functions such as "maximum". Multiple occurrences of the same sampling value as input to the function "maximum" do not affect the aggregation result. This case would also apply with duplicate-sensitive aggregation functions such as "count", when the sensor readings and aggregates can be encoded into duplicate-insensitive data. Recall that an aggregation protocol called synopsis diffusion is introduced in Chapter 2 [67]. In synopsis diffusion, data is represented by duplicate-insensitive synopses. These synopses are small-sized digests of the data and can be included in a fixed size packet. For duplicate-sensitive aggregation functions such as "count", the original sensor readings and the aggregation results are turned into duplicate-insensitive synopses. The sink can compute the final aggregation result from the synopses it received, no matter how many times the same synopsis is aggregated at multiple nodes. The second case applies with application independent data aggregation [37], where different sensor readings are concatenated and included in the aggregation results. Which case more closely reflects reality may depend on the extent to which sensor values and identifiers can be encoded in highly-compressed form in the aggregates.

With the uniform random error model, the results show that for a fixed required packet size, broadcast-based protocols are able to support higher sampling/collection rates with lower collection delays than unicast-based protocols. The performance improvements are particularly pronounced for synchronous aggregation. When the required packet size is assumed to increase linearly with the number of values included, however, no significant performance advantage are found with broadcast-based protocols. With the two-state Gilbert error model, broadcast-based protocols yield considerably lower end-to-end loss rate than unicast-based protocols, as MAC layer retransmission becomes ineffective in loss recovery.

The remainder of the chapter is organized as follows. Section 4.2 proposes new synchronous and asynchronous broadcast-based aggregation protocols. Section 4.3 presents simulation results evaluating the new protocols in comparison to the unicast-based aggregation protocols. Section 4.4 provides initial experimental results. Section 4.5 concludes this chapter.

4.2 Broadcast-based Aggregation

The main goal here is to design broadcast-based aggregation protocols that minimize the number of packet transmissions and rely on multipath delivery rather than ARQ for reliability. Similar to the unicast-based aggregation protocols in Chapter 3, broadcast-based aggregation protocols can also be classified as synchronous and asynchronous based on their timing schemes. New synchronous and asynchronous broadcast-based aggregation protocols are proposed in this section.

4.2.1 Synchronous Broadcast-based Aggregation

The proposed synchronous broadcast-based aggregation is designed based on the synchronous unicast-based protocol described in Section 3.3.1. With synchronous aggregation protocols, within each round of communication, all sensor nodes at the same distance (number of hops) from the sink are given the same interval of time in which to transmit. Nodes farther away from the sink are given earlier transmission intervals, so as to allow their data to be aggregated with that of nodes closer to the sink before these latter nodes make their own transmissions.

In the synchronous broadcast-based aggregation protocol, nodes are organized into a ring topology [67], as shown in Fig. 4.1. The sink is the only node that is located in ring 0, nodes one hop away from the sink are in ring 1, etc. As in the unicast-based protocol, nodes in different rings are allotted different time intervals within each round of communication for their transmissions. It is assumed that each node i knows its hop count h_i to the sink and the maximum hop count H in the tree, and accordingly chooses its transmission interval within each round. As before, the duration of the period between sensor readings is denoted by τ, and the interval duration by I. In a tree topology, each node forwards its data to its parent. In the ring topology, node i in ring h_i may forward its data to multiple nodes in ring $h_i - 1$, as long as these nodes are within node i's transmission range.

Unlike the synchronous unicast-based protocol, in the proposed broadcast-based protocol each interval is divided into a first and second phase with durations λI and

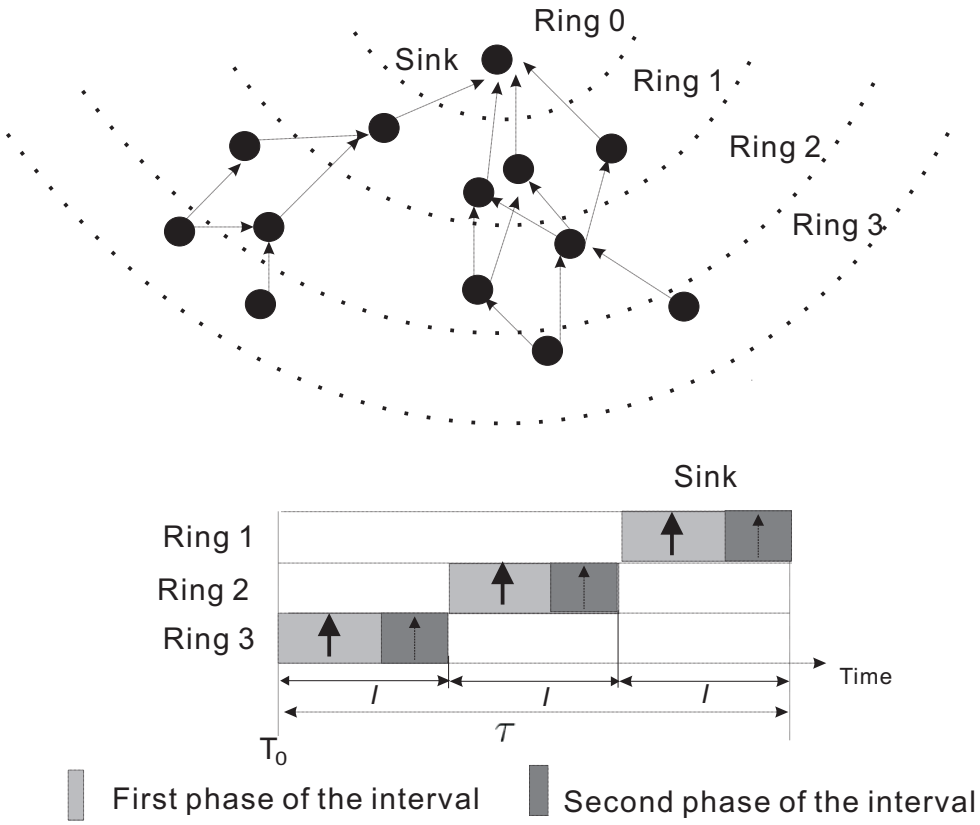


Figure 4.1: Synchronous Broadcast-based Aggregation

$(1 - \lambda)I$, respectively. Each broadcast packet includes a bit vector indicating the nodes whose data is aggregated in the packet. Some applications may require the corresponding node ID of each value the sink collects to keep track of the changes that happen at the location where the node is deployed. The bit vector is not an overhead in this case, and the unicast-based protocols also have to carry such a bit vector. One method to implement the bit vector is to pre-configure the nodes with a running number starting at 0, then only one bit is needed for each node. If such preconfiguration is not doable, instead of carrying a long node ID, such as the 64-bit ID for mote-class devices, bloom filter, a space-efficient randomized data structure can be used to encode node information [10]. When the node ID for each value is not needed by the sink, the overhead of the bit vector can be further reduced through network partition. The bit vector only carries the IDs of the nodes that are located in the same network partition of the sender. The construction of the bit vector is not modelled in this work.

Each node (except for the sink) broadcasts a packet containing (possibly aggregated) sensor data during the first phase of its interval in each round. Nodes may also make a second broadcast during the second phase, as described below. To keep the data consistent, each node only collects one sampling value during each round, i.e., all data is collected when the node makes its first broadcast. Nodes aggregate all of the data they have received from broadcasts for the current round (including broadcasts from neighbouring nodes in the same ring), for their own broadcasts.

Specifically, in each round j, node i picks a random value $r1_i^j$ between 0 and $(\lambda - 0.1)I$, aggregates the data from the broadcasts it has received for this round, and makes its own first broadcast at time $T_0 + \tau(j-1) + (H - h_i)I + r1_i^j$. For nodes that happen to choose a random value close to $(\lambda - 0.1)I$, there is a $0.1I$ interval that allows their packets to get through before the second phase starts. Here it has been assumed that all nodes agree on the same base time T_0 defining the beginning of the first round. All neighbouring nodes that heard node i's first broadcast for round j include node i's data in the broadcasts they make later for that round. Node i then picks another random value $r2_i^j$ between 0 and $(1 - \lambda - 0.1)I$. A broadcast is made in the second phase of the round, at time $T_0 + \tau(j-1) + (H - h_i)I + \lambda I + r2_i^j$, if, by this time, node i has not heard a broadcast transmission from some other node in the same ring that has included node i's data. There is a gap of $0.1I$ separating the second phase from the beginning of the next interval. Similar to the unicast-based protocol, randomizing the transmissions within each phase yields better performance than when all of the nodes in the same ring attempt to transmit at approximately the same time.

The second broadcasts are important to improve reliability. For nodes with few neighbours, or nodes that are the last among the nodes in the same ring to make their first phase broadcast, a second broadcast increases the likelihood that their data is received by at least one node in the next ring. This two-phase strategy can reduce the overall end-to-end loss rate significantly, at minimal additional cost in terms of numbers of transmissions.

As described previously, this work makes no assumptions regarding the type of

aggregation that is performed, but rather attempts to model a range of scenarios in the performance experiments through consideration of two extreme cases with respect to how packet size grows with the number of aggregated values.

4.2.2 Asynchronous Broadcast-based Aggregation

With synchronous aggregation, the time interval during which each node transmits is statically determined. In contrast, in the asynchronous aggregation protocols considered here, each node adaptively determines when to transmit based on its history of past packet receptions from its children. The "adaptive asynchronous" unicast-based aggregation protocol proposed in Chapter 3 is used as a basis for the new asynchronous broadcast-based aggregation protocol.

In the synchronous unicast-based aggregation protocol used in TAG, aggregation is performed over a tree structure, with one parent for each node except the sink [61]. It is observed in TAG that a node could potentially transmit to multiple parents, each one hop closer to the sink, depending on the density of the network. A "value splitting" aggregation protocol is proposed based on that observation, where each node may two parents. In the broadcast-based protocol proposed in this section, nodes may similarly have up to two parents[1], but the same sensor value (rather than only distinct shares of a sensor value) may be aggregated at each parent node and forwarded on up the tree. The maximum number of parents that each node is allowed to have is limited to two. When a node has multiple neighbours closer to the sink, two of them are selected as the node's parents. Allowing each node to have more parents increases reliability, but results in increased delay as an parent node has to wait for data from more children. Allowing some nodes to broadcast twice as described later in this section, together with this 2-parent strategy, yield good performance in terms of both reliability and delay.

Such multipath routing in broadcast-based aggregation can yield improved reliability, but requires that nodes be able to receive multiple aggregates including the

[1]When a node has only one neighbour closer to the sink, it can choose a sibling that has multiple neighbours closer to the sink as its second parent.

same sensor value. This can be handled when aggregation is duplicate insensitive, or when aggregation is performed in such a way that it is possible to recognize and filter out duplicates.

As with the synchronous broadcast-based protocol, each node (except the sink) broadcasts either once or twice during each round. Similarly, all nodes collect their sampling values only once during each round to keep the data consistent. Each broadcast may include not only data from its children, but also data overheard from broadcasts from other neighbours in the tree. Each broadcast packet includes a bit vector indicating the nodes whose data is aggregated in the packet. A non-leaf node makes its first broadcast during a round either when it has received the data from each of its children (either directly from a broadcast by that child, or indirectly via a broadcast from some other child), or upon timeout. Timeouts are established in a similar manner as in the unicast-based asynchronous protocol described in Section 3.2.2. A second broadcast is made if and only if the node does not hear a broadcast transmission from any other node that includes its data, or if it receives additional data from its children, before a second timeout occurs.

In each round of the asynchronous broadcast-based protocol, a timeout is set at each non-leaf node establishing the maximum time the node will wait to receive packets from its children in the aggregation tree. The timeout value is determined adaptively, based on the timings of packet receptions from the child nodes in the previous rounds. The node transmits its packet (aggregating its own data with whatever it has received from its children) either when packets have been received from all children, or when the timeout expires.

As before, it is assumed that all nodes agree on the same base time T_0 defining the beginning of the first round. As in the asynchronous unicast-based protocol, each node i (other than the sink node) picks a random delay r_i between 0 and R, where R is a protocol parameter. At time $T_0 + r_i$, node i broadcasts a packet containing its own data for the first round.

In each subsequent round $j \geq 2$, each leaf node i makes its first broadcast at time $T_0 + r_i + (j-1)\tau$, aggregating its own data and any other data it has overheard

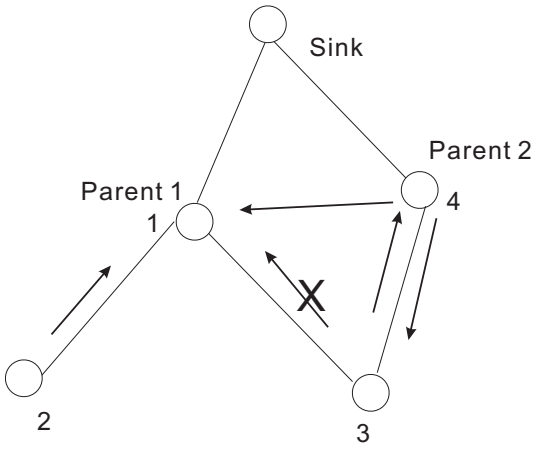


Figure 4.2: Asynchronous Broadcast-based Aggregation

from other broadcasts. Note that there is no second broadcast for the first round. Let A_i^j denote the time at which a non-leaf node i receives all of the data from its children for round j that it will receive during this round. Let L_i^j denote the time at which non-leaf node i receives the last of the first broadcasts from its children for round j that it will receive during this round. Note, A_i^j might not equal L_i^j, as the parent node may not be able to receive all first broadcasts of its children. Data from its children may come with the second broadcasts of the children, or the broadcasts of the parent's other neighbours. Fig. 4.2 illustrates a simple scenario in which A_1^j does not equal L_1^j.

Let TO_i^j denote the timeout for the first transmission of round j at non-leaf node i. TO_i^2 is set to $L_i^1 + \tau$. After each round j, $j > 1$, TO_i^{j+1} is set as follows:

1. If node i received the data for round j from all of its children prior to time TO_i^j, its timeout for round $j+1$ is set to $TO_i^{j+1} = (1-\delta)(TO_i^j + \tau) + \delta(A_i^j + \tau + e)$, similar to in the asynchronous unicast-based protocol.

2. If the first transmission timeout for round j expires before node i receives the

data from all of its children, its timeout for round $j+1$ is tentatively set to $TO_i^{j+1} = TO_i^j + \tau$. If node i receives one or more first-time broadcasts for round j from its children subsequent to the expiry of its timeout for that round, it updates TO_i^{j+1} to $L_i^j + \tau$. Data from these packets has been received too late to be aggregated in node i's first broadcast for round j, but can be included in node i's second broadcast.

As noted above, starting from round 2, a second broadcast is made following the first broadcast, if the node does not hear a broadcast transmission from any other node that includes its data, or if it receives additional data from its children, prior to a second timeout. The second timeout is set to a time duration $e/2$ following the time of the first broadcast. As in the asynchronous unicast-based protocols in Chapter 3, the protocol parameter e allows for some variance in the times at which packets are successfully received. When the second broadcast is transmitted after a time duration $e/2$ following the time of the first broadcast, there is a good chance that the second packet is received by the node's parent before the parent sends out its first broadcast.

It is important to randomize the transmission times of leaf nodes to avoid congestion at the beginning of each round. The parameter R controls the duration of the randomization interval. A similar strategy as in the adaptive asynchronous unicast-based aggregation protocol in Chapter 3 is adopted to adjust the value of R so that it is kept within a certain range, as measured relative to the EWMA average D_{ave} of the delay D from the beginning of a round until the last of the data from that round is received at the sink. In the simulation implementation, D_{ave} is measured as $D_{ave} = \alpha D_{ave} + (1-\alpha)D^*$, where D^* is the latest measurement of the delay D, and α is a smoothing factor that determines the weight given to the old value. The adaptive protocol attempts to keep the value of R/D_{ave} in the interval $[\beta-\Delta, \beta+\Delta]$, where β and Δ are protocol parameters. When R/D_{ave} moves out of this range, the value of R is updated (and sent by the sink to all of the sensor nodes) as follows. If $R/D_{ave} < \beta - \Delta$, R is updated to $R = D_{ave}(\beta + \Delta)$. If $R/D_{ave} > \beta + \Delta$, R is updated to $R = D_{ave}(\beta - \Delta)$.

4.3 Simulation-based Performance Evaluation

4.3.1 Goals, Metrics, and Methodology

The performance of the broadcast-based protocols is evaluated through ns2 simulation. The adaptive asynchronous unicast-based aggregation protocol (described in Section 3.2.3) and the synchronous unicast-based aggregation protocol (described in Section 3.3.1) are used for comparison. The primary performance metrics are the following: (1) the end-to-end loss rate (the ratio of the number of sensor readings not included in the aggregates arriving at the sink to the total number of readings the nodes generate), (2) the maximum data age (τ plus the maximum data collection delay), (3) the average number of MAC layer packet transmissions per round (for unicast, including both ACK and data packet transmissions), and (4) the average number of bytes transmitted per round. The last metric yields insight into relative energy usage. As in chapter 3, the maximum data age is a measure of how "stale" the data received at the sink from one round can become, before that for the next round is received. The maximum data age reported in the figures is the average of the maximum data ages over multiple rounds.

Sensor fields are generated by randomly scattering nodes in square areas. The sink is the node closest to the center of the network. Three sensor networks are used in performance evaluation, with 120, 160, and 240 nodes randomly deployed in a 250 meter by 250 meter area, respectively. All sections, except Section 4.3.2.3 and 4.3.3.2, show results for the sensor network with 160 nodes. Note that these three sensor networks are also used in the performance evaluation in Chapter 3. Fig. 3.4 in Section 3.3.2 shows the topology of the sensor network with 160 nodes.

An independent random error model is used for all experiments except those in which the two-state Gilbert error model is used to simulate channel errors (Section 4.3.3.4 and 4.3.2.5). The physical layer packet loss rate is specified as a simulation input parameter for the uniform random error model. Similar as in Chapter 3, an 802.11 MAC layer is simulated for unicast-based protocols, without RTS/CTS [102],

with a transmission range of 40 meters and rate of 2Mbps. Different maximum numbers of MAC layer retransmissions are simulated for when the sender fails to receive an ACK. The same transmission range and data rate are used for the broadcast-based protocols in the simulations. The protocols are evaluated with both fixed and increasing packet sizes. Fixed-size packets are 52 bytes. Otherwise the packet size grows linearly with the number of values aggregated in the packet at a rate of 4 bytes per sensor value. The performance comparison does not rely on the choice of the number of bytes for each sensor value. The total number of the sensor values that are included in the aggregates is the decisive factor in this case. Whether it is 4 bytes or 2 bytes per sensor value, the results will yield the same insight. Table 4.1 lists the parameters that are used in the performance evaluation.

As in Chapter 3, for the synchronous aggregation protocols, each simulation run lasts 1,150 sampling period durations, with the initial 200 and the last 50 sampling period durations removed from the measurements. For the asynchronous aggregation protocols, each simulation run lasts 1,550 sampling period durations, with the initial 600 and the last 50 sampling period durations removed from the measurements. As explained in Chapter 3, the simulations with asynchronous aggregation last longer to allow time for the computation of the randomization delay and the convergence of the network. Unlike the results in Chapter 3, each result reported below is from a single simulation run. It is observed that the results averaged over multiple runs are very similar to the results from one simulation run. (This is also true with the results in Chapter 3.) It is not necessary to run the simulations multiple times.

4.3.2 Performance for Real-time Data Collection

In this section, the performance of the protocols at different sampling rates is examined. The results provide useful insight for real-time data collection that requires high sampling/collection rate and low collection delay, and may tolerate some modest amount of data loss.

τ	The time between successive sensor readings at each node.
R	The duration of the randomization interval is initialized to 0 for both unicast and broadcast-based asynchronous aggregation protocols.
PLR	The physical layer loss rate for the independent random error model.
N	The number of nodes inside the network.
S	The size of the square area inside which N nodes are scattered. All sensor networks in this chapter are deployed in a 250 meter by 250 meter area.
α	Protocol parameter used in asynchronous unicast and broadcast-based aggregation to calculate the EWMA average of the delay D. It determines the weight given to the previous value of the average. The value of α is fixed at 0.875 unless otherwise stated. The value of α is fixed at 0.875 unless otherwise stated.
β	Protocol parameter for asynchronous unicast and broadcast-based aggregation. The desired range of R/D_{ave} is $[\beta-\Delta, \beta+\Delta]$. Unless otherwise stated, the value β is fixed at 0.7 and 0.6 for fixed and increasing packet size, respectively.
Δ	Protocol parameter for asynchronous unicast and broadcast-based aggregation. The desired range of R/D_{ave} is $[\beta-\Delta, \beta+\Delta]$. The value of Δ is fixed at 0.15 unless otherwise stated.
e	Protocol parameter for asynchronous unicast and broadcast-based aggregation. Unless otherwise stated, the value of e is fixed at 0.06 seconds and 0.12 seconds for asynchronous broadcast-based aggregation with fixed packet size and increasing packet size, respectively. For asynchronous unicast-based aggregation, e is set to 0.1 seconds for fixed packet size and 0.15 seconds for increasing packet size.
δ	Protocol parameter used in asynchronous unicast and broadcast-based aggregation to calculate the timeout for the next round. It controls how quickly the protocol reacts to changes in the network. Protocol parameter for . The value of δ is fixed at 0.05 unless otherwise stated.
λ	Protocol parameter for synchronous unicast and broadcast-based aggregation. The transmission times of the nodes at the same tree level are scattered over λI for each round. The value of λ is fixed at 0.8.
τ_b	The average duration of the bad state in the two-state Gilbert model.
P_b	Percentage of time that is spent in the bad state in the two-state Gilbert model.
I	The duration of the transmission interval for synchronous unicast and broadcast-based aggregation protocols. The value of I is set to τ/H unless otherwise stated, where H is the maximum hop count H in the tree.

Table 4.1: Simulation Parameters for Chapter 4

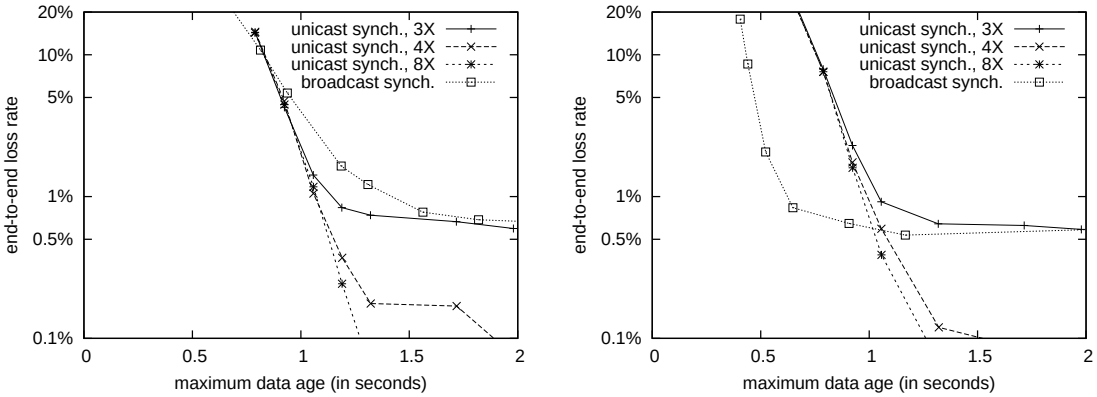
4.3.2.1 Principal Performance Comparison

Experiments for this section use the sensor field with 160 nodes randomly deployed in a 250 meter by 250 meter area. The physical layer loss rate is fixed at 20% in all experiments for this section; the impact of physical layer loss rate is considered later in section 4.3.2.2. In the figures, 3X, 4X, and 8X denote up to 3, 4, and 8 MAC layer retransmissions. Values for τ are chosen manually within the range of [0.15, 1.12] seconds. For each protocol, the chosen values yield good coverage of the x / y region being plotted in the figures. Note that there is some variation in the particular values chosen from this range for different protocols. This variation in the choice of τ also exists in the simulations in Sections 4.3.2.2, 4.3.2.3, and 4.3.2.5.

Chapter 3 shows that for asynchronous unicast-based aggregation, all parameters except R can be fixed to certain values that yield good performance for a broad range of network configurations. Through experimentation, it is found that the same conclusion holds for asynchronous broadcast-based aggregation as well. In all the simulations whose results are reported in this chapter, e is fixed at 0.12 seconds and 0.06 seconds for asynchronous broadcast-based aggregation with increasing packet size and fixed packet size, respectively. For asynchronous unicast-based aggregation, e is set to 0.15 seconds for increasing packet size and 0.1 seconds for fixed packet size. Parameter β is fixed at 0.7 and 0.6 for fixed and increasing packet size, respectively. Other common parameters for both asynchronous unicast-based and broadcast-based aggregation, δ, α, and Δ are fixed at 0.05, 0.875, and 0.15 respectively. Figures showing the impact of the protocol parameters can be found in Appendix B.

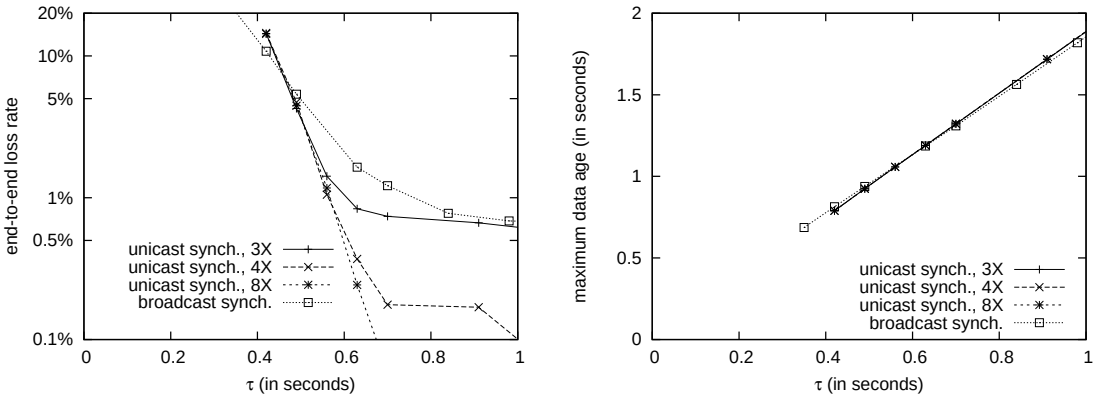
Fig. 4.3 plots performance results of the synchronous aggregation protocols. The results for the different points on each curve are generated by varying the sampling period duration τ, and measuring the resulting maximum data age and end-to-end loss rate. For synchronous aggregation, λ is set to 0.8, and the interval duration is set to τ divided by the maximum hop count to the sink. To better explain the results, Fig. 4.4 plots performance results of the synchronous protocols as a function of τ, for increasing packet size. Both Fig. 4.3(a) and Fig. 4.4 plot the results from

the same set of simulations. As τ gets smaller, the end-to-end loss rate starts to grow as more packets are lost because of collisions. In addition, the interval duration gets shorter as τ decreases. When the interval is too small, nodes in the same ring cannot make their transmissions within their own transmission interval. Packets that arrive after the receiver has sent out its partial aggregate are not aggregated, and the end-to-end loss rate increases sharply. In all figures that plot the end-to-end loss rate as a function of the maximum data age, the point with the highest end-to-end loss rate on the curve corresponds to the smallest τ (the highest sampling rate).



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.3: Performance of Synchronous Unicast and Broadcast-based Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 20%)

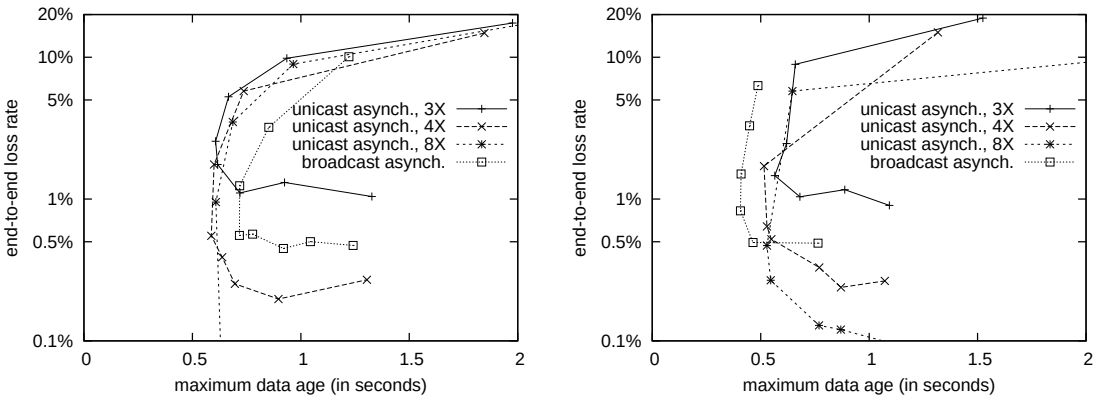


(a) End-to-End Loss Rate (b) Maximum Data Age

Figure 4.4: Performance of Synchronous Aggregation (real-time, N = 160, S = 250m $\times$ 250m, PLR = 20%, increasing packet size)

Fig. 4.5 plots performance results of the asynchronous aggregation protocols.

Similarly, the results for the different points on each curve are generated by varying τ, and measuring the resulting maximum data age and end-to-end loss rate. The initial value for R is 0 for the asynchronous protocols. To better explain the results, Fig. 4.6 plots the end-to-end loss rate and the maximum data age of the asynchronous protocols as a function of τ, for increasing packet size. Both Fig. 4.5(a) and Fig. 4.6 plot the results from the same set of simulations. As τ gets smaller, the end-to-end loss rate starts to grow as more packets are lost because of collisions. When the sampling rate gets too high, congestion starts to build up in the network. With asynchronous aggregation, not only does the loss rate increase sharply when the sampling rate is too high, but the maximum data age also increases, since nodes increase their timeout values (and therefore their delays before sending their aggregates) to reflect the late arrival times of packets that have been retransmitted. This explains why the curves turn sharply and move to the upper right corner in Fig. 4.5.

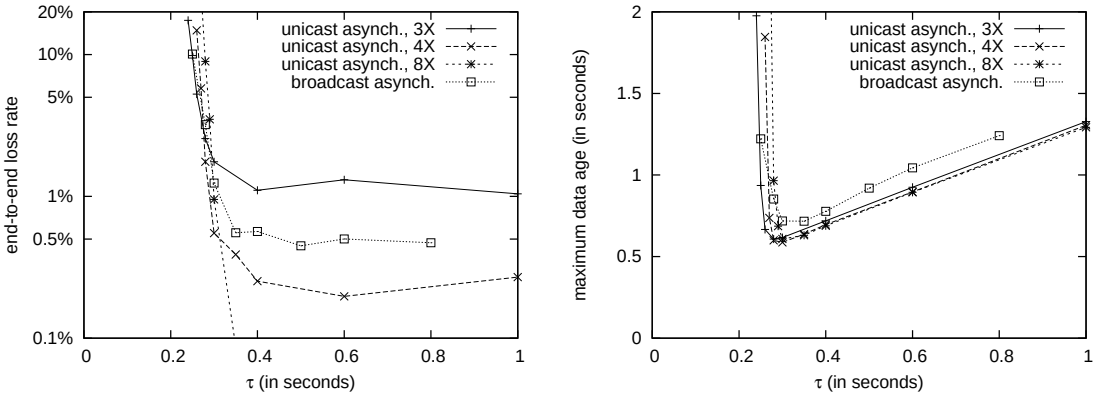


(a) Increasing Packet Size

(b) Fixed Packet Size

Figure 4.5: Performance of Asynchronous Unicast and Broadcast-based Aggregation (real-time, varying τ, N = 160, S = 250m × 250m, PLR = 20%)

The figures show that both synchronous and asynchronous broadcast-based aggregation are outperformed by their unicast-based counterparts for increasing packet size. For fixed packet size, both synchronous and asynchronous broadcast-based aggregation are able to achieve lower maximum data age than their unicast-based counterparts. In broadcast-based aggregation, each node broadcasts at most twice for each round. In unicast-based aggregation, however, a successful unicast packet



(a) End-to-End Loss Rate

(b) maximum data Age

Figure 4.6: Performance of Asynchronous Aggregation (real-time, N = 160, S = 250m × 250m, PLR = 20%, increasing packet size)

transmission requires at least two MAC layer transmissions, one for the data packet and one for the acknowledgement. When data from different nodes can be aggregated into a fixed-size packet, the unicast-based aggregation protocols are likely to generate more traffic, and larger traffic volume leads to longer delay. For increasing packet size, However, the broadcast-based aggregation protocols are likely to generate more traffic and suffer older data age, since the same sample value may be carried in multiple packets. This intuition is confirmed by the traffic analysis in Section 4.3.2.4.

Comparing Fig. 4.3 and Fig. 4.5, it appears that a broadcast-based approach is most promising for synchronous (rather than asynchronous) aggregation. For fixed packet size, synchronous broadcast-based aggregation outperforms synchronous unicast-based aggregation with up to three MAC layer retransmissions for different sampling rates. When up to 4 and 8 MAC layer retransmissions are permitted, synchronous unicast-based aggregation is able to achieve lower end-to-end loss rate than synchronous broadcast-based aggregation when τ is large enough. For real-time data collection that is willing to accept moderate end-to-end loss rate for better data freshness, broadcast-based synchronous aggregation is able to achieve much lower maximum data age than its unicast-based counterpart in the case of fixed packet size. For example, for 2% end-to-end loss rate, synchronous broadcast-based

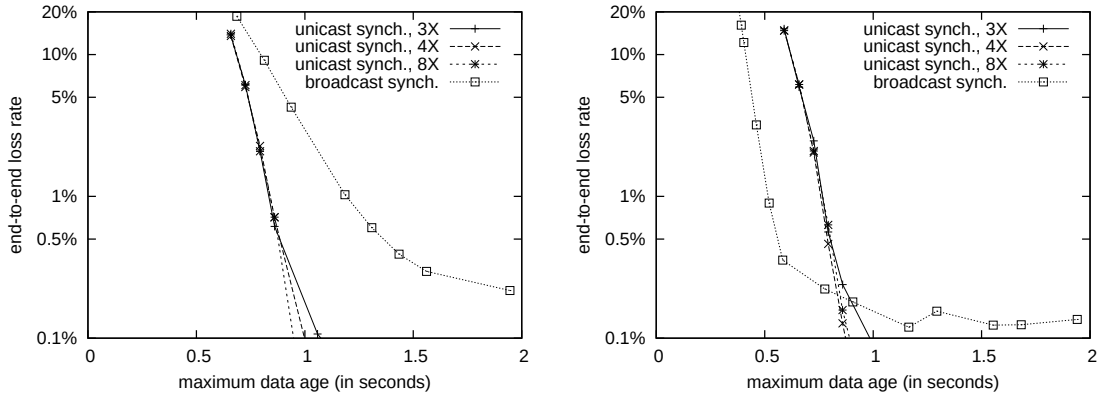
aggregation is able to lower the maximum data age by about 40%, as seen in Fig. 4.3(b). The performance improvement of broadcast-based asynchronous aggregation, however, is not that significant. For fixed packet size, it is only able to lower the maximum data age by about 20%, as seen in Fig. 4.5(b).

4.3.2.2 Impact of Physical Layer Loss Rate

Experiments for this section use the sensor network with 160 nodes randomly deployed in a 250 meter by 250 meter area. Fig. 4.7 and Fig. 4.8 plot performance results of the synchronous protocols for 10% and 30% physical layer loss rate, respectively. Values for τ are chosen manually within the range of [0.13, 1.19] seconds.

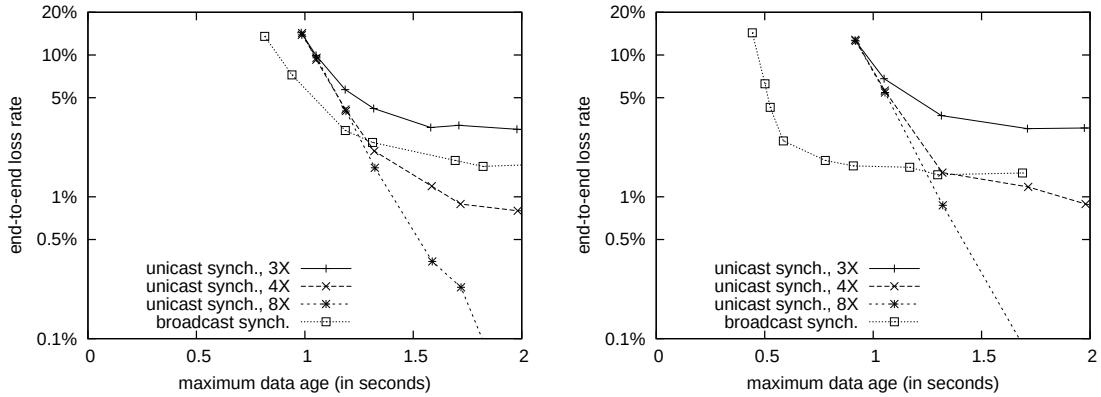
Aggregating the same data at different nodes improves reliability, but also increases the packet size for broadcast-based aggregation with increasing packet size. When the physical layer loss rate is 10%, the cost of the retransmissions is relatively modest in unicast-based aggregation. As seen in Fig. 4.7(a), synchronous unicast-based aggregation outperforms synchronous broadcast-based aggregation for increasing packet size. For 20% physical layer loss rate, synchronous unicast-based aggregation also outperforms synchronous broadcast-based aggregation for increasing packet size, as shown in Fig. 4.3(a). As the physical layer loss rate increases, however, the cost of the retransmissions in unicast-based aggregation becomes more substantial. Fig. 4.8(a) shows that with 30% physical layer loss rate, synchronous broadcast-based aggregation is able to achieve lower maximum data age than the synchronous unicast-based protocol for increasing packet size.

For fixed packet size, 4.8(b) shows that the synchronous broadcast-based protocol is able to lower the maximum data age by about 50% with moderate end-to-end loss rate (5% end-to-end loss rate, for example). Fig. 4.7(b), Fig. 4.3(b), and Fig. 4.8(b) plot performance results of the synchronous protocols for 10%, 20%, and 30% physical layer loss rate, respectively. The results suggest that the relative performance of synchronous broadcast-based aggregation improves as the physical layer loss rate gets higher. The reason behind this can be traced back to traffic volume. As mentioned before, in broadcast-based aggregation, each node broadcasts



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.7: Impact of Lower Physical Layer Loss Rate on Synchronous Aggregation (real-time, varying τ, N = 160, PLR = 10%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.8: Impact of Higher Physical Layer Loss Rate on Synchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 30%)

at most twice for each round regardless of the loss probability. In unicast-based aggregation, however, the number of retransmissions increases as the loss rate gets higher, and the delay increases accordingly as more traffic is generated. As seen in Fig. 4.7(b), Fig. 4.3(b), and Fig. 4.8(b), the physical layer loss rate has a greater impact on the maximum data age of unicast-based aggregation than on that of broadcast-based aggregation. As a result, the relative performance of synchronous broadcast-based aggregation improves as the physical layer loss rate gets higher

Fig. 4.9 and Fig. 4.10 plot the performance of the asynchronous protocols for 10% and 30% physical layer loss rate, respectively. Fig. 4.9(a), Fig. 4.5(a), and

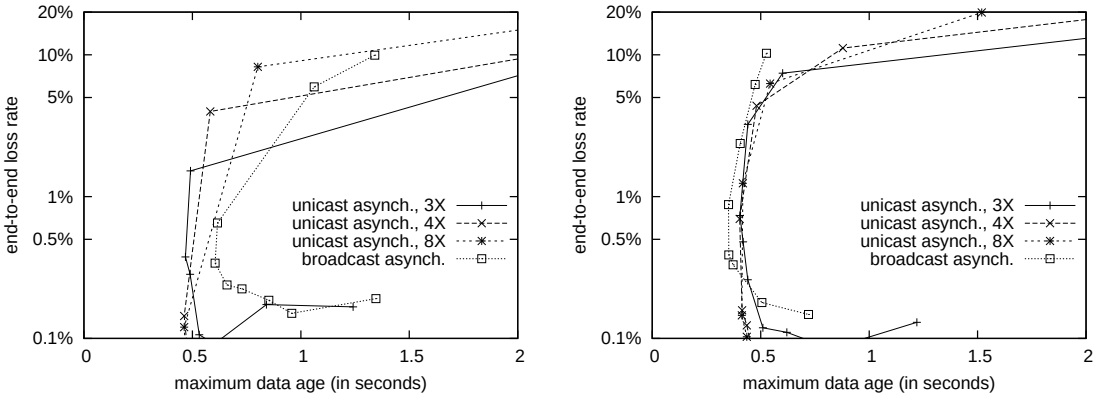
Fig. 4.10(a) show that asynchronous broadcast-based aggregation is outperformed by asynchronous unicast-based aggregation for all three physical layer loss rates in the case of increasing packet size. For fixed packet size, asynchronous broadcast-based aggregation is able to achieve lower maximum data age for all three physical layer loss rates, but the performance gain is not as significant as with the synchronous broadcast-based aggregation protocol. Similarly, the relative performance of asynchronous broadcast-based aggregation improves as the physical layer loss rate increases.

4.3.2.3 Impact of Density

Fig. 4.11 and Fig. 4.12 plot performance results of the synchronous aggregation protocols, for the sensor fields with 120 nodes and 240 nodes over a 250 meter by 250 meter area, respectively. Values for τ are chosen manually within the range of [0.13, 1.5] seconds.

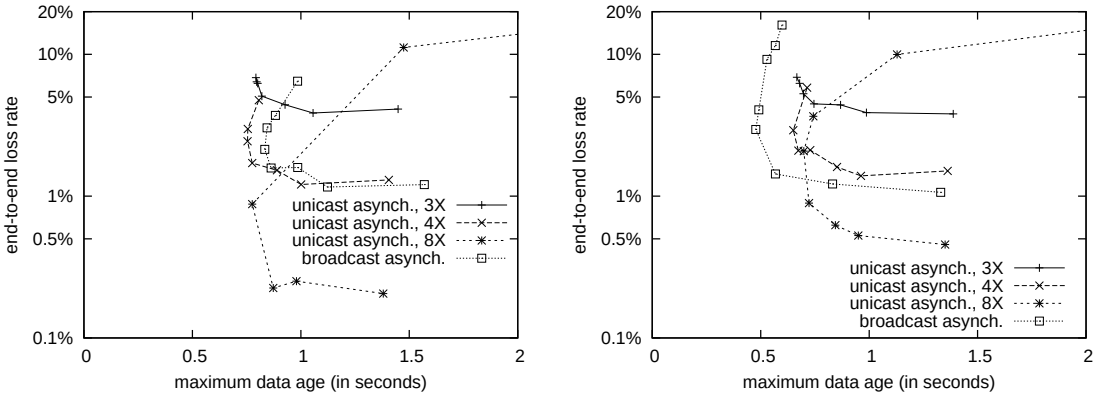
As the network density increases, the broadcast-based protocols are expected to achieve higher reliability as a broadcast is likely to be received by more nodes. However, for broadcast-based aggregation with increasing packet size, larger packets are produced as the same data is aggregated by more nodes, which means a longer packet transmission time and greater network congestion. Meanwhile, packet loss recovery is quite feasible for unicast-based protocols with the packet loss rate of 20% that is used for these figures. As the result of these two effects, Fig. 4.12 shows that synchronous broadcast-based aggregation is outperformed even more by synchronous unicast-based aggregation for increasing packet size, when the number of nodes (and density) is increased. For fixed packet size, however, performance with the synchronous broadcast-based protocol improves as network density gets higher, but performance with synchronous unicast-based aggregation degrades as network density increases. The relative performance of synchronous broadcast-based aggregation improves with increased density.

Fig. 4.13 and Fig. 4.14 show the performance of the asynchronous aggregation protocols, for the sensor fields with 120 nodes and 240 nodes, respectively. Similar to



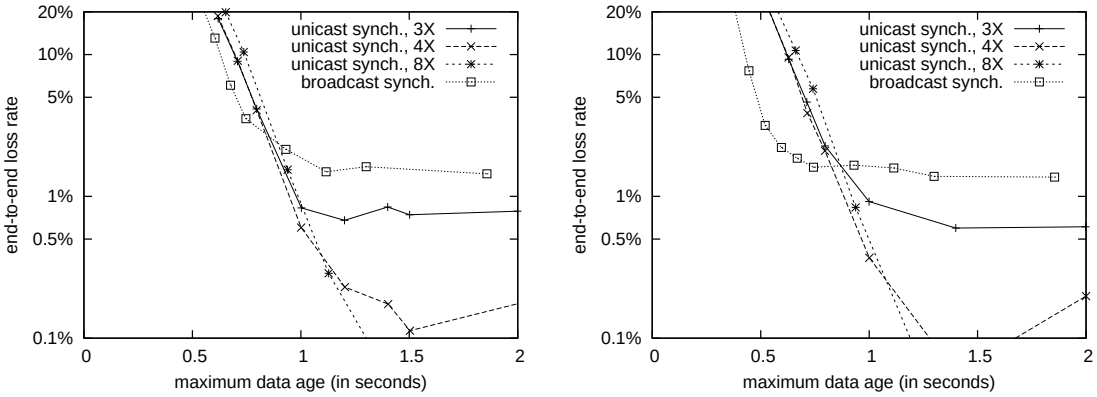
(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.9: Impact of Lower Physical Layer Loss Rate on Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 10%)



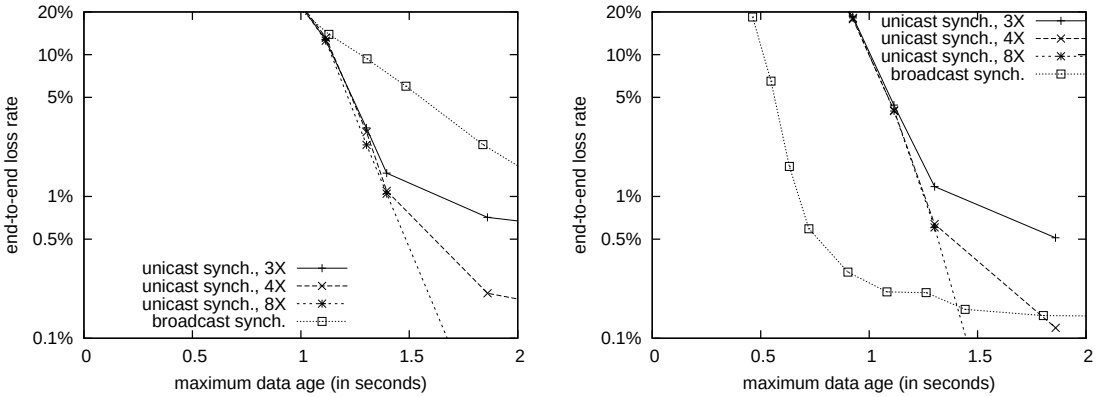
(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.10: Impact of Higher Physical Layer Loss Rate on Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 30%)



(a) Increasing Packet Size (b) Fixed Packet Size

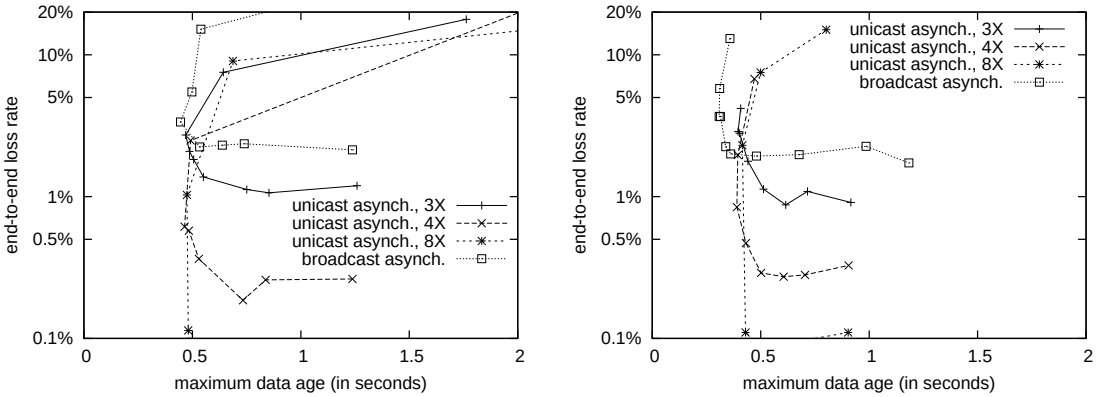
Figure 4.11: Impact of Lower Density on Synchronous Aggregation (real-time, varying τ, N = 120, S = 250m $\times$ 250m, PLR = 20%))



(a) Increasing Packet Size (b) Fixed Packet Size

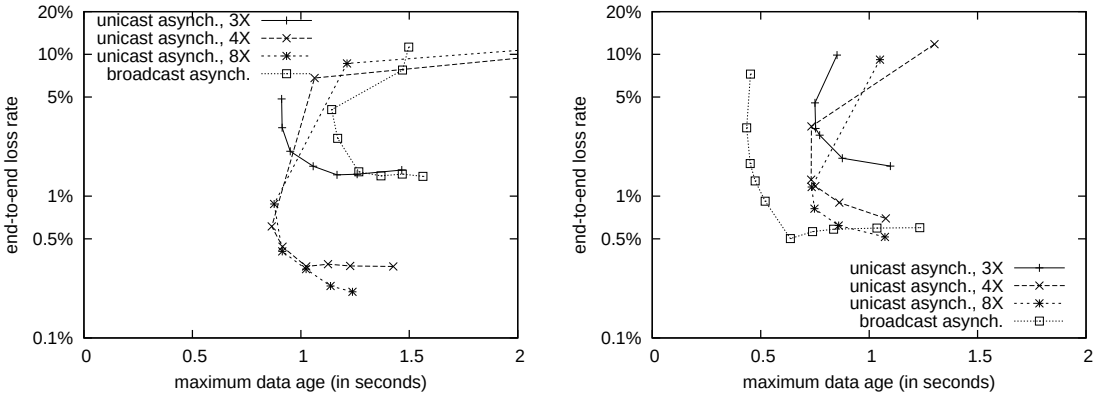
Figure 4.12: Impact of Higher Density on Synchronous Aggregation (real-time, varying τ, N = 240, S = 250m $\times$ 250m, PLR = 20%)

synchronous broadcast-based aggregation, for increasing packet size, asynchronous broadcast-based aggregation is outperformed even more by asynchronous unicast-based aggregation when the number of nodes (and density) is increased. For fixed packet size, however, the relative performance of asynchronous broadcast-based aggregation improves with higher density. For 1% end-to-end loss rate, asynchronous broadcast-based aggregation is able to lower the maximum data age by about 30% with higher density, as seen in Fig. 4.14(b).



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.13: Impact of Lower Density on Asynchronous Aggregation (real-time, varying τ, N = 120, S = 250m $\times$ 250m, PLR = 20%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.14: Impact of Higher Density on Asynchronous Aggregation (real-time, varying τ, N = 240, S = 250m $\times$ 250m, PLR = 20%)

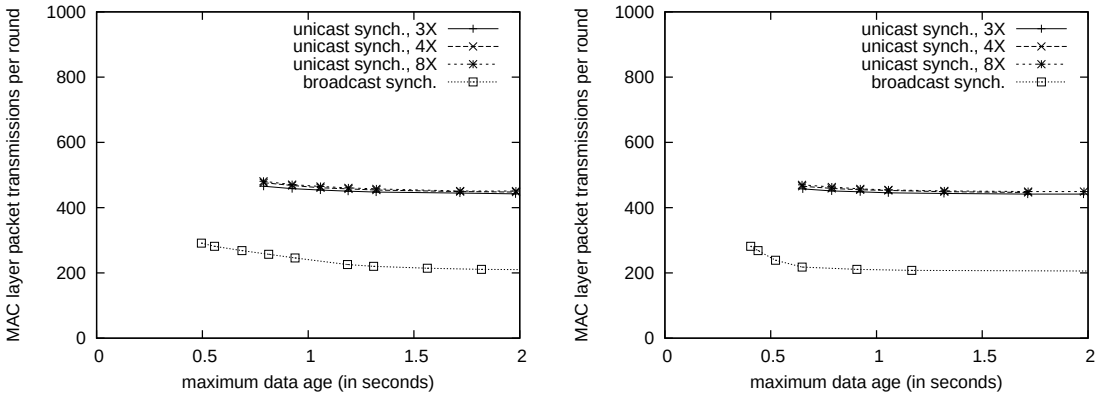
4.3.2.4 Traffic Volume

Figures in this section plot performance results for the sensor network with 160 nodes over a 250 meter by 250 meter area. The physical layer loss rate is fixed at 20% in the simulations. The results reported in this section and Section 4.3.2.1 are computed based on the same set of simulations. Section 4.3.2.1 reports the maximum data age and the end-to-end loss rate, and this section reports traffic volume.

Fig. 4.15 and Fig. 4.16 plot the average number of MAC layer packet transmissions per round and the average number of bytes transmitted per round with the synchronous aggregation protocols, respectively. In the broadcast-based protocols, each node transmits at most twice for each round. In the unicast-based protocols, however, a successful unicast packet transmission requires at least two MAC layer transmissions, one for the data packet and one for the acknowledgement. As seen in Fig. 4.15, synchronous broadcast-based aggregation sends fewer packets per round than synchronous unicast-based aggregation.

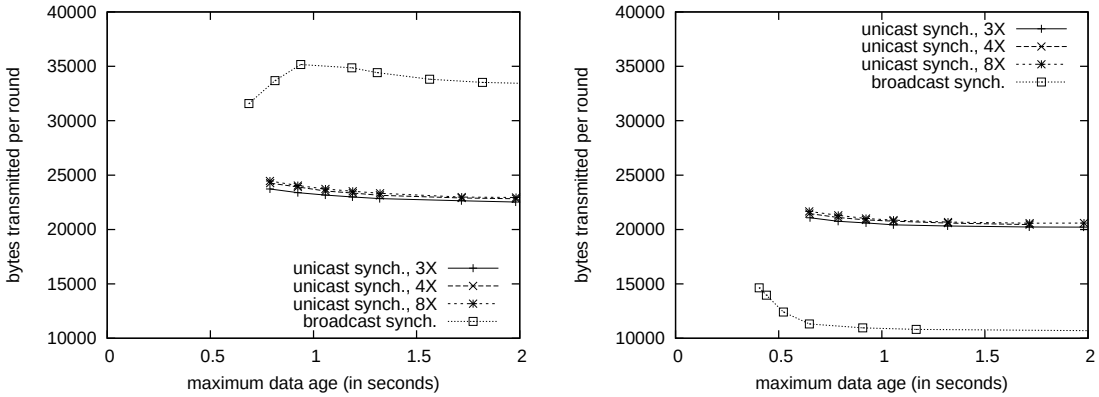
The duration of the transmission interval gets shorter as the sampling period duration τ gets shorter. More packets collide when the nodes in the same ring contend for the channel during the shortened transmission interval. For synchronous unicast-based aggregation, more MAC layer retransmissions are made for packet loss recovery. For synchronous broadcast-based aggregation, more nodes broadcast twice because they do not hear a broadcast that includes their own data from any other node in the same ring. This explains why for both unicast and broadcast-based aggregation, more packet transmissions are made as the sampling rate gets higher.

In broadcast-based aggregation, a node aggregates all data included in the broadcasts it receives from its neighbors. The same sensor value may be included in the aggregates of different nodes. For increasing packet size, while synchronous broadcast-based aggregation sends fewer MAC layer packets per round than synchronous unicast-based synchronous aggregation, a larger volume of data is produced by synchronous broadcast-based aggregation. This helps to explain why synchronous broadcast-based aggregation yields poorer performance in this case.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.15: MAC layer Packet Transmissions per Round of Synchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 20%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.16: Bytes Transmitted per Round of Synchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 20%)

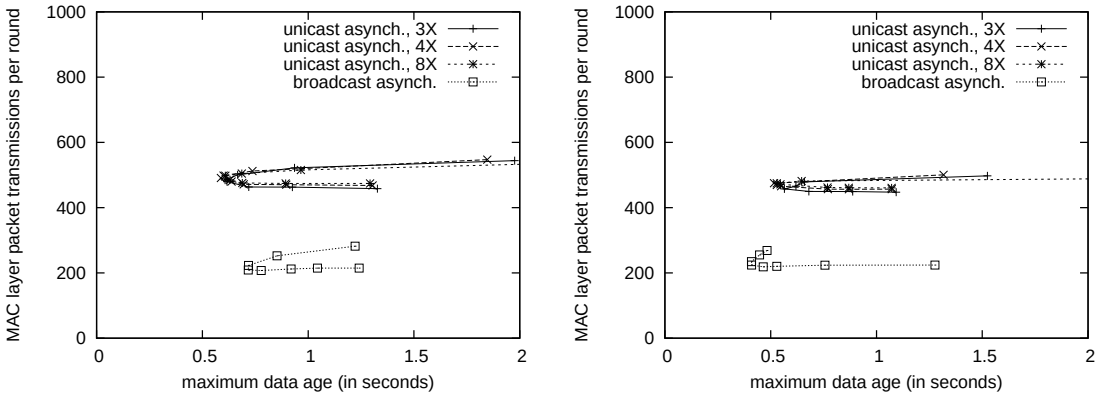
Fig. 4.3(a) shows that the end-to-end loss rate gets higher as the sampling rate gets higher. The end-to-end loss rate for the point with the lowest maximum data age in Fig. 4.16(a) are above 20%. Note that the corresponding results for that point are not plotted in Fig. 4.3(a). As more packets are dropped, the redundancy caused by multiple routing is reduced for broadcast-based aggregation, and the average number of values inluded in each aggregate decreases (packet size decreases accordingly in the case of increasing packet size). While the average number of MAC layer packet transmissions per round keeps increasing as the sampling rate gets higher, for increasing packet size, the average number of bytes transmitted per round increases

first and then starts to drop as the sampling rate gets higher, as seen in Fig. 4.16(a).

Fig. 4.17 and Fig. 4.18 plot the average number of MAC layer packet transmissions per round with the asynchronous aggregation protocols as a function of the maximum data delay and τ, respectively. Similarly, in asynchronous broadcast-based aggregation, each node broadcasts at most twice for each round. In asynchronous unicast-based aggregation, a successful unicast packet transmission requires at least two MAC layer transmissions. For both fixed and increasing packet size, asynchronous broadcast-based aggregation sends fewer packets per round than unicast-based asynchronous aggregation.

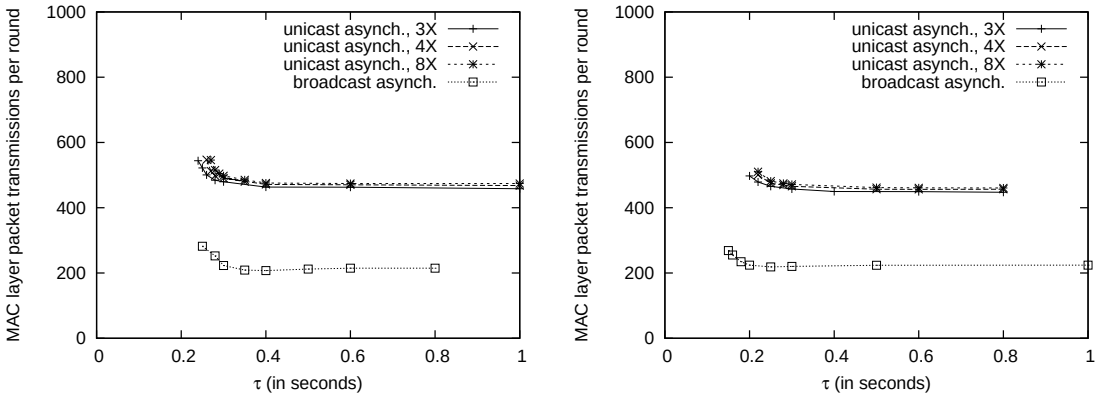
Fig. 4.18 shows that more packets are transmitted as the sampling rate gets higher. For asynchronous unicast-based aggregation, more MAC layer retransmissions are made to recover packet losses due to collisions. For asynchronous broadcast-based aggregation, more nodes transmit twice because they do not hear a broadcast transmission that includes their own data from any other node.

Fig. 4.19 and Fig. 4.20 plot the average number of bytes that are transmitted per round with the asynchronous aggregation protocols as a function of the maximum data delay and τ, respectively. While for both fixed and increasing packet size, fewer MAC layer packets are transmitted by asynchronous broadcast-based aggregation, a larger volume of data is produced by asynchronous broadcast-based aggregation in the case of increasing packet size, owing to the fact that the same sampling value may be included in multiple aggregates. This helps to explain why in Fig. 4.5, asynchronous broadcast-based aggregation yields poorer performance than its unicast-based counterparts for increasing packet size. Fig. 4.5(a) and Fig. 4.19(a) show that the highest end-to-end loss rate for the points plotted in these two figures is around 10%. In Fig. 4.16(a), it is observed that the average number of bytes transmitted per round with synchronous broadcast-based aggregation increases first and then starts to drop as the sampling rate gets higher in the case of increasing packet size. This phenomenon is not observed in Fig. 4.19(a) because the end-to-end loss rate is not high enough to reduce the redundancy caused by multiple routing for asynchronous broadcast-based aggregation in this case.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.17: MAC layer Packet Transmissions per Round of Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m × 250m, PLR = 20%)

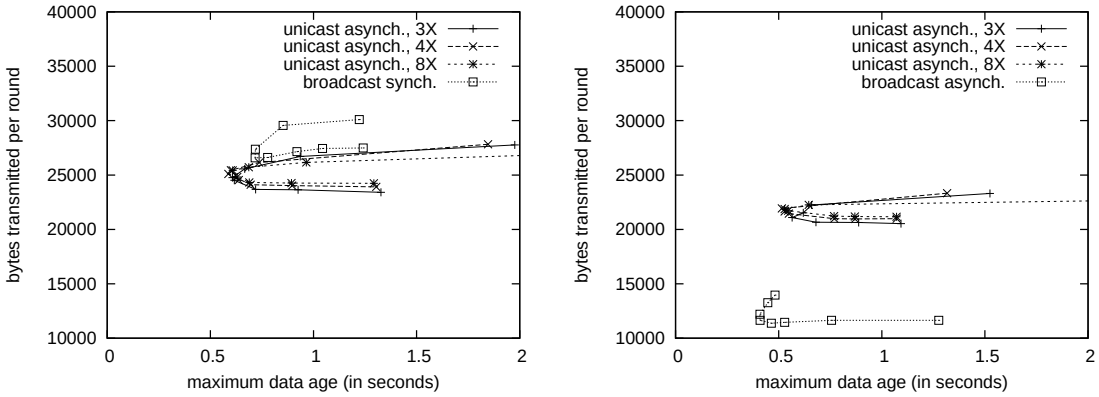


(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.18: MAC layer Packet Transmissions per Round of Asynchronous Aggregation as a Function of τ (real-time, N = 160, S = 250m × 250m, PLR = 20%)

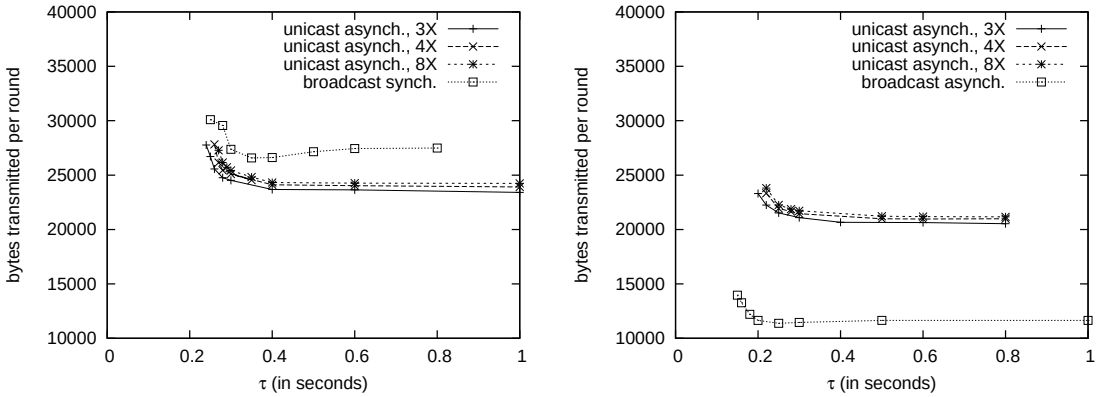
4.3.2.5 Impact of Two-State Gilbert Error Model

The two-state Gilbert error model as seen in Fig. 4.21 is considered in this section. At each point in time, each node (other than the sink) is in one of two states, independently of all other nodes. In the "good" state, all packets are received correctly unless they collide with other packets. In the "bad" state, no packets are received correctly. The time a node spends in a state before transiting to the other state (the sojourn time) is exponentially distributed. The sink is assumed to be a special node, which always stays in the "good" state.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.19: Bytes Transmitted per Round of Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m $\times$ 250m, PLR = 20%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.20: Bytes Transmitted per Round of Asynchronous Aggregation as a Function of τ (real-time, N = 160, S = 250m $\times$ 250m, PLR = 20%)

Parameterizing the Gilbert model in Fig. 4.21 requires fixing the average sojourn times in the good and bad states. The ratio of the average sojourn time in the bad state (denoted by τ_b) to the sum of these times determines the percentage of time spent in the bad state (denoted by P_b). Note that for given values of P_b and τ_b, the average sojourn time in the good state can be calculated as $\tau_b \times (100 - P_b)/P_b$. For all results reported in this section, P_b is fixed at 20% and τ_b is fixed at 0.5 seconds.

Values for τ are chosen manually within the range of [0.13, 1.4] seconds. Fig. 4.22 plots performance results for synchronous broadcast-based aggregation. The results show that synchronous unicast-based aggregation suffers high end-to-end loss

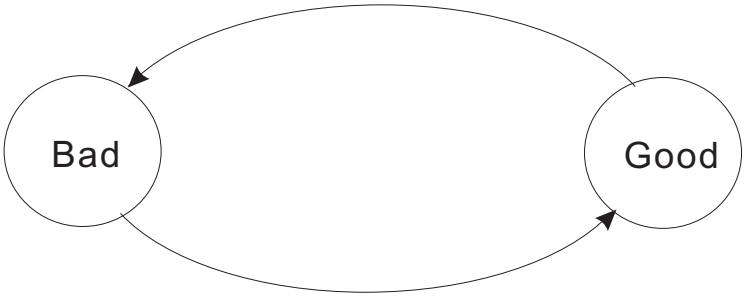
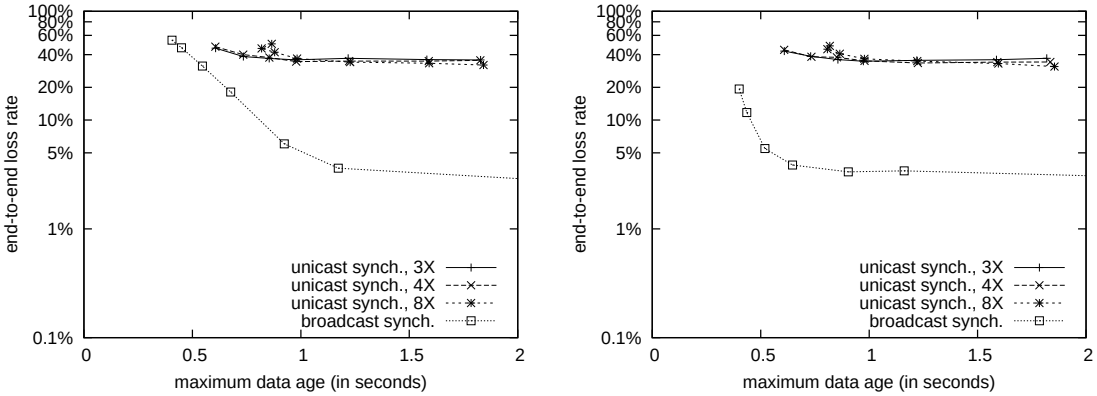


Figure 4.21: Two-State Gilbert Error Model (Continuous Time Model)



(a) Increasing Packet Size

(b) Fixed Packet Size

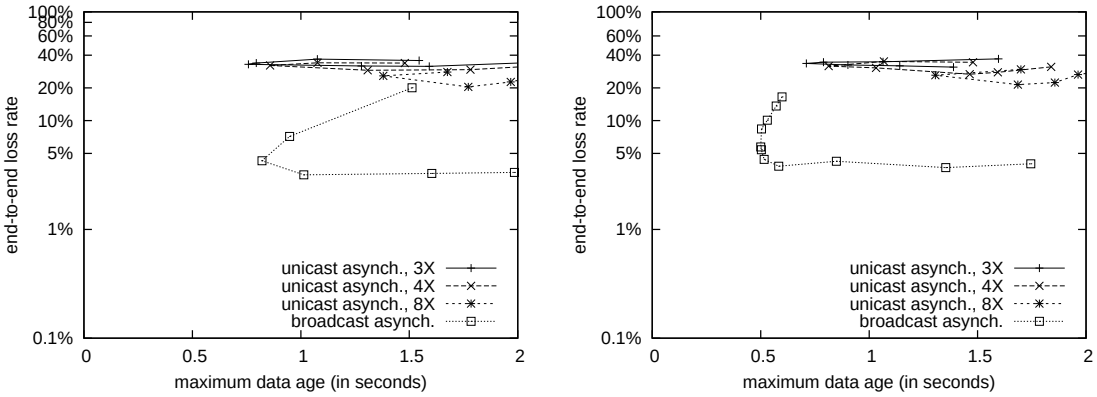
Figure 4.22: Impact of Two-state Gilbert Error Model on Synchronous Aggregation (real-time, varying τ, N = 160, S = 250m × 250m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)

rate in this case, because link layer retransmissions are ineffective in loss recovery. Synchronous broadcast-based aggregation outperforms synchronous unicast-based aggregation for both fixed and increasing packet size.

Fig. 4.23 plots performance results for asynchronous broadcast-based aggregation. Similarly, asynchronous unicast-based aggregation suffers high end-to-end loss rate in this case. Asynchronous broadcast-based aggregation outperforms asynchronous unicast-based aggregation for both fixed and increasing packet size.

4.3.2.6 Broadcast-based Synchronous vs. Broadcast-based Asynchronous

Fig. 4.24 plots performance results of broadcast-based synchronous and broadcast-based asynchronous aggregation. As shown in Chapter 3, the performance of the synchronous protocols is very sensitive to the choice of I. In practice, it may be



(a) Increasing Packet Size

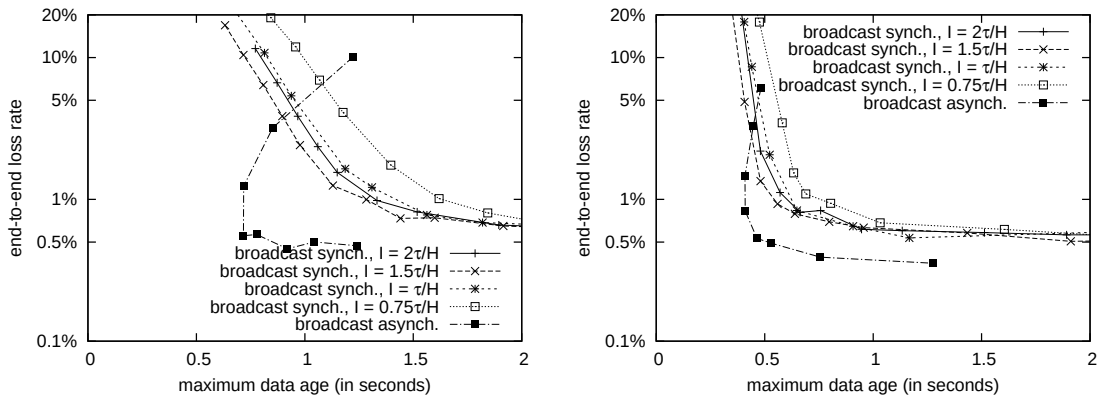
(b) Fixed Packet Size

Figure 4.23: Impact of Two-state Gilbert Error Model on Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m × 250m, P_b = 20%, τ_b = 0.5 sec.)

difficult to set this parameter in a manner yielding consistently good performance. In the performance comparison between synchronous broadcast and unicast-based aggregation, the interval duration is set to τ/H for both protocols. The synchronous protocols may not perform the best when I is set to τ/H, but it is fair to both of them. When comparing the performance of broadcast-based synchronous and asynchronous aggregation, however, fixing I to τ/H may be unfair to the synchronous protocols. To explore the potential of broadcast-based synchronous aggregation, extra simulations are run with $I = 0.75\tau/H, 1.5\tau/H$, and $2\tau/H$. Fig. 4.24 shows that for both increasing and fixed packet size, broadcast-based asynchronous aggregation is still able to achieve lower maximum data age for a given end-to-end loss rate, which is consistent with the results in Chapter 3.

4.3.2.7 Summary of Results

The proposed synchronous and asynchronous broadcast-based aggregation protocols are evaluated and compared with their unicast-based counterparts in the context of real-time data collection in Section 4.3.2. Sections 4.3.2.1 to 4.3.2.4 present performance results with the independent random error model. The results show that for aggregation with fixed packet size, broadcast-based aggregation is able to achieve lower maximum data age than unicast-based aggregation in most scenarios, partic-



(a) Increasing Packet Size

(b) Fixed Packet Size

Figure 4.24: Broadcast-based Synchronous vs. Broadcast-based Asynchronous Aggregation (real-time, varying τ, N = 160, S = 250m × 250m, $PLR = 20\%$)

ularly in the case of synchronous protocols. However, for increasing packet size, the broadcast-based protocols are outperformed by their unicast-based counterparts in most scenarios. Performance results with the two-state Gilbert error model are presented in Section 4.3.2.5. Both synchronous and asynchronous broadcast-based aggregation yields significantly better performance than their unicast-based counterparts when packet loss follows the two-state Gilbert error model.

4.3.3 Performance for Delay-Tolerant Data Collection

To evaluate the performance of the aggregation protocols for delay-tolerant data collection, a sufficiently large τ is chosen so that network contention is minimal. Maximum data age is not a major concern in delay-tolerant applications. Performance comparison between unicast and broadcast-based aggregation is focused on the end-to-end loss rate and traffic volume.

4.3.3.1 Principal Performance Comparison

Figures in this section show performance results for the sensor network with 160 nodes over a 250 meter by 250 meter area. Fig. 4.25 plots the end-to-end loss rate of the synchronous protocols, for different physical layer loss rates. The figure is

for fixed packet size; the figure for increasing packet size looks very similar and is omitted. Packet loss due to contention is rare when τ is sufficiently large. Packet size does not have much impact on end-to-end loss rate and the average number of MAC layer packet transmissions per round in this case. The average number of bytes transmitted per round, however, is still greatly influenced by packet size.

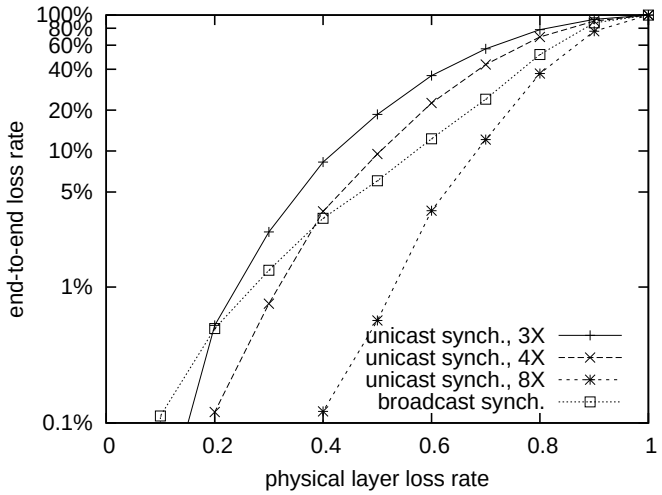


Figure 4.25: Performance of Synchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m, fixed packet size)

As seen in Fig. 4.25, synchronous broadcast-based aggregation outperforms synchronous unicast-based aggregation with up to 3 MAC layer retransmissions when physical layer loss rate is higher than 20%. It also outperforms synchronous unicast-based aggregation with up to 4 MAC layer retransmissions when physical layer loss rate is higher than 40%. However, retransmission is effective in packet recovery when packet loss is uniformly distributed. Synchronous unicast-based aggregation is able to yield lower end-to-end loss rate for different physical layer loss rates when up to 8 MAC layer retransmissions are allowed.

Fig. 4.26 shows the performance of the asynchronous protocols for different physical layer loss rates. As before, the figure is for fixed packet size; the figure for increasing packet looks very similar and is omitted. Asynchronous unicast-based aggregation with up to 8 MAC layer retransmissions consistently outperforms asynchronous broadcast-based aggregation for different physical layer loss rates.

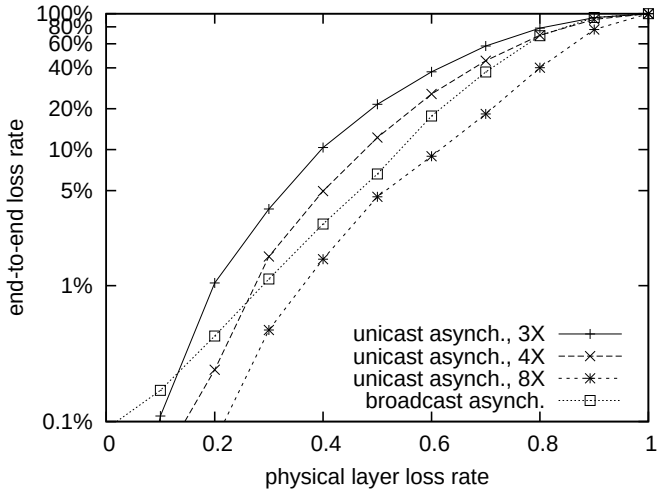


Figure 4.26: Performance of Asynchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m, fixed packet size)

4.3.3.2 Impact of Density

Simulations for this section are done with the two sensor fields with 120 and 240 nodes randomly deployed in a 250 meter by 250 meter area. Only results for fixed packet size are shown; results for increasing packet size are quite similar.

Fig. 4.27 and Fig. 4.28 show the performance of the synchronous aggregation protocols, for the sensor fields with 120 and 240 nodes, respectively. Synchronous broadcast-based aggregation relies on multi-path routing for reliability. Lower density means that a node has fewer neighbors to receive its broadcasts and carry its data in their aggregates. Fig. 4.27 shows that synchronous broadcast-based aggregation is consistently outperformed by synchronous unicast-based aggregation with up to 4 and 8 MAC layer retransmissions for different physical layer loss rates. The relative performance of synchronous broadcast-based aggregation improves with higher density. For the sensor field with 240 nodes, synchronous broadcast-based aggregation consistently outperforms synchronous unicast-based aggregation with up to 3 MAC layer retransmissions for different physical layer loss rates. It also outperforms synchronous unicast-based aggregation with up to 4 MAC layer retransmissions when the physical layer loss rate is higher than 30%.

Fig. 4.29 and Fig. 4.30 plot performance results of the asynchronous aggregation

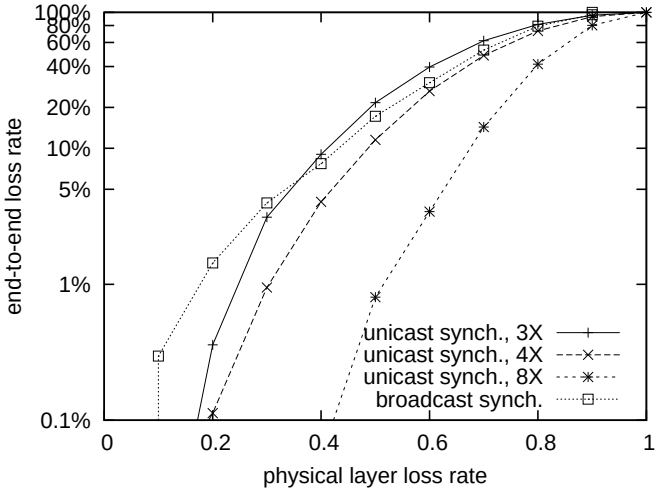


Figure 4.27: Impact of Lower Density on Synchronous Aggregation (delay-tolerant, N = 120, S = 250m × 250m, fixed packet size)

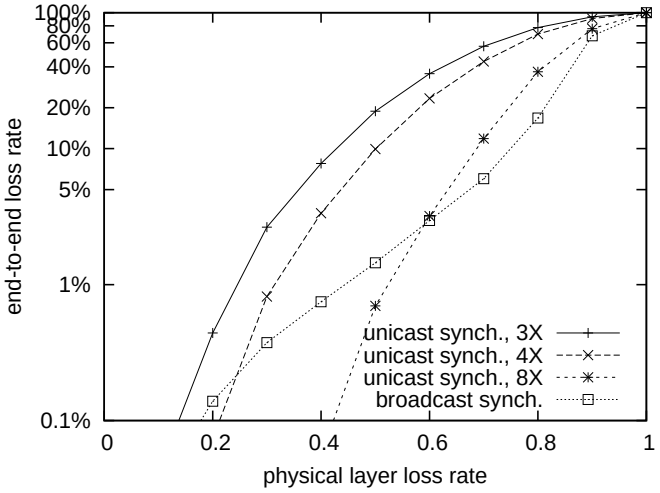


Figure 4.28: Impact of Higher Density on Synchronous Aggregation (delay-tolerant, N = 240, S = 250m × 250m, fixed packet size)

protocols, for the sensor fields with 120 and 240 nodes, respectively. Similar to synchronous broadcast-based aggregation, the relative performance of asynchronous broadcast-based aggregation improves as the network density increases.

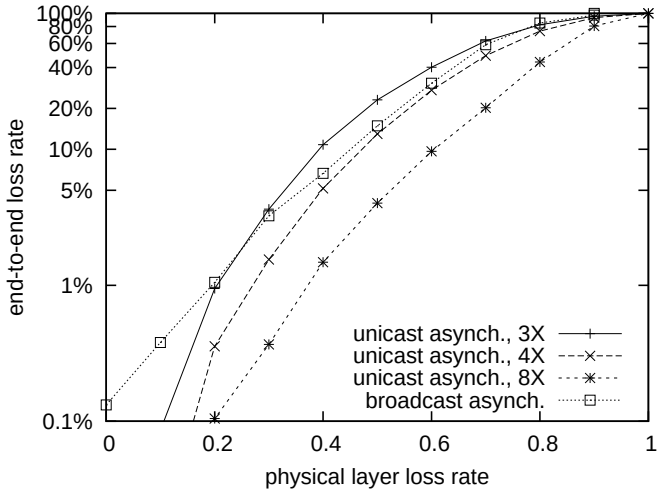


Figure 4.29: Impact of Lower Density on Asynchronous Aggregation (delay-tolerant, N = 120, S = 250m × 250m, fixed packet size)

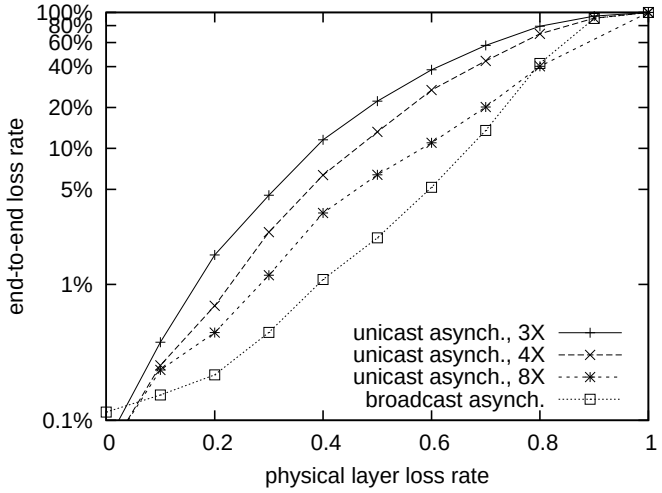


Figure 4.30: Impact of Higher Density on Asynchronous Aggregation (delay-tolerant, N = 240, S = 250m × 250m, fixed packet size)

4.3.3.3 Traffic Volume

Figures in this section show performance results for the sensor network with 160 nodes over a 250 meter by 250 meter area. Fig. 4.31 shows the average number of MAC layer packets that are transmitted per round with the synchronous aggregation

protocols, for fixed packet size. The figure for increasing packet size is very similar. As described before, each node broadcasts at most twice per round in broadcast-based aggregation. In unicast-based aggregation, at least two MAC layer packet transmissions are required for each successful unicast transmission. For all physical layer loss rates, synchronous broadcast-based aggregation sends fewer MAC layer packets per round than synchronous unicast-based aggregation.

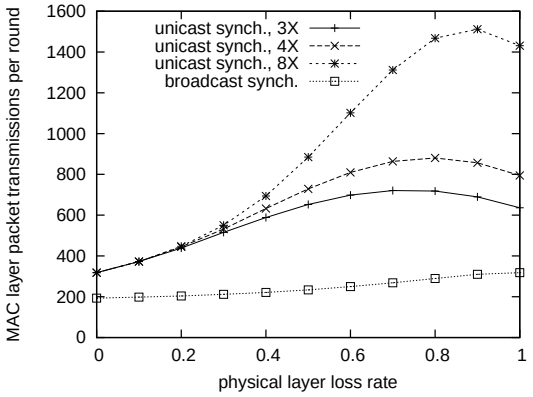
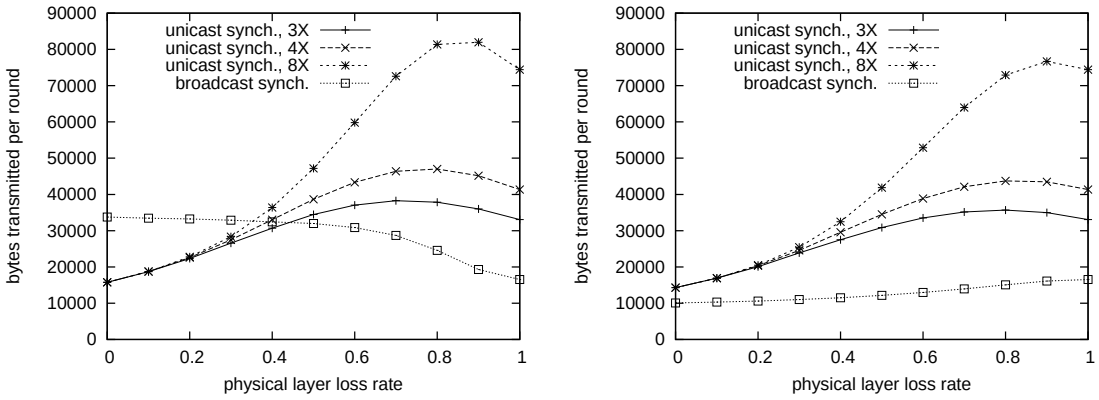


Figure 4.31: MAC layer Packet Transmissions per Round of Synchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m, fixed packet size)



(a) Increasing Packet Size

(b) Fixed Packet Size

Figure 4.32: Bytes Transmitted per Round of Synchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m)

Fig. 4.32 shows the average number of bytes that are transmitted per round with the synchronous aggregation protocols. For increasing packet size, the data volume with synchronous broadcast-based aggregation is larger than with the other protocols when the physical layer loss rate is lower than 40%, owing to cases in which

the same sensor value is redundantly included in multiple aggregates. Note that the data volume with this protocol keeps decreasing as the loss probability increases, since the growing packet loss decreases this redundancy. For fixed packet size, the traffic volume with synchronous broadcast-based aggregation is consistently lower for all loss rates.

Fig. 4.33 shows the average number of MAC layer packets that are transmitted per round with the asynchronous aggregation protocols, for fixed packet size. The figure for increasing packet size is very similar. Fig. 4.34 plots the average number of bytes that are transmitted per round with the asynchronous aggregation protocols, for both fixed and increasing packet size. The results are quite similar with those for synchronous aggregation.

4.3.3.4 Impact of Two-State Gilbert Error Model

Broadcast and unicast-based aggregation using the two-state Gilbert error model are evaluated in this section. The proportion of time each node spends in the bad state is fixed at 20%. The average sojourn time in the bad state is varied to model a range of scenarios. With very short sojourn times, independent random packet loss is modelled. With long sojourn times, link outages and partial node failures are modelled.

Fig. 4.35 and Fig. 4.36 plot performance results of the synchronous and asynchronous aggregation protocols, respectively, for the sensor field with 160 nodes over a 250 meter by 250 meter area. Only results for fixed packet size are shown; results for increasing packet size are very similar. For long average sojourn time in the bad state, broadcast-based aggregation yields much lower end-to-end loss rate than the unicast-based aggregation protocols, because MAC layer retransmissions are not effective in packet recovery in this case. As seen in Fig. 4.35 and Fig. 4.36, both synchronous and asynchronous unicast-based aggregation protocols yield around 40% end-to-end loss rate with long average sojourn time in the bad state (2 seconds, for example). For a very small average sojourn time, such as 0.0001 seconds in the figures, the end-to-end loss rate increases, since a node is likely to enter the bad state

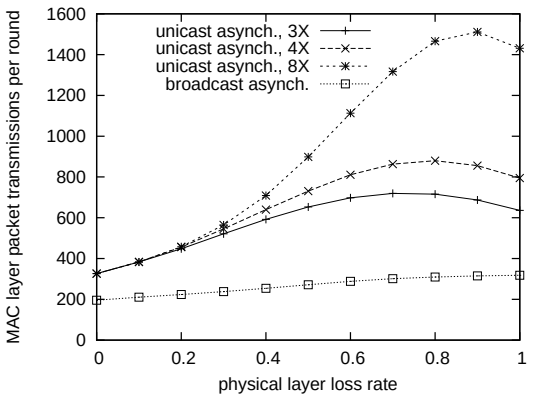
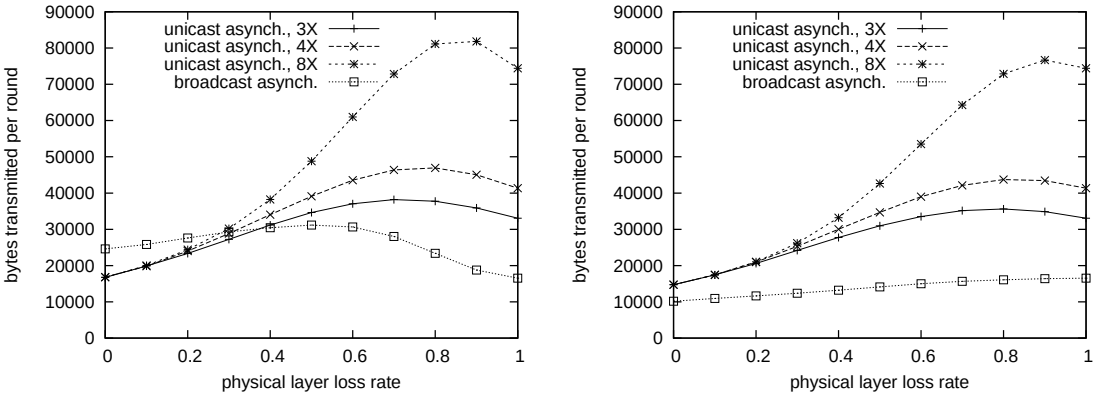


Figure 4.33: MAC layer Packet Transmissions per Round of Asynchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m, fixed packet size)



(a) Increasing Packet Size

(b) Fixed Packet Size

Figure 4.34: Bytes Transmitted per Round of Asynchronous Aggregation (delay-tolerant, N = 160, S = 250m × 250m)

at least once while receiving a packet, causing the loss of that packet.

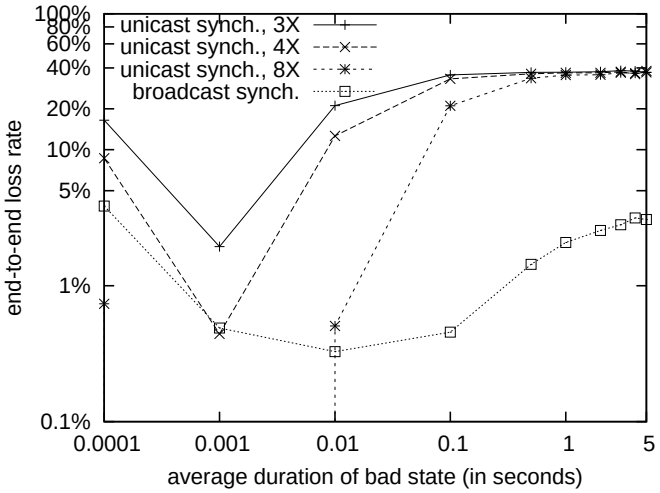


Figure 4.35: Performance of Synchronous Aggregation for Two-state Gilbert Error Model (delay-tolerant, N = 160, S = 250m × 250m, P_b = 20%, fixed packet size)

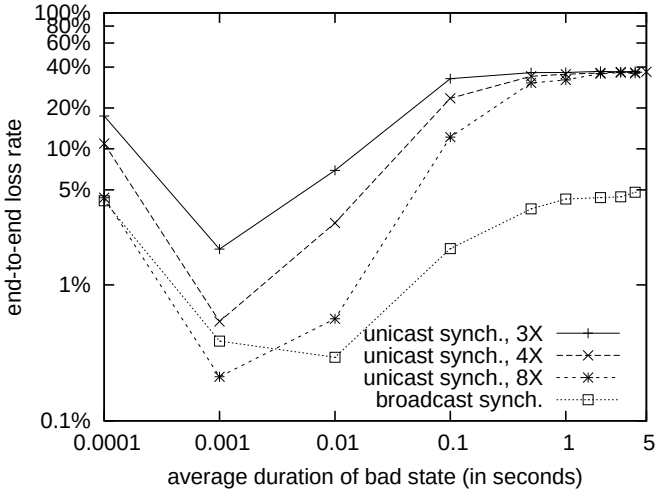
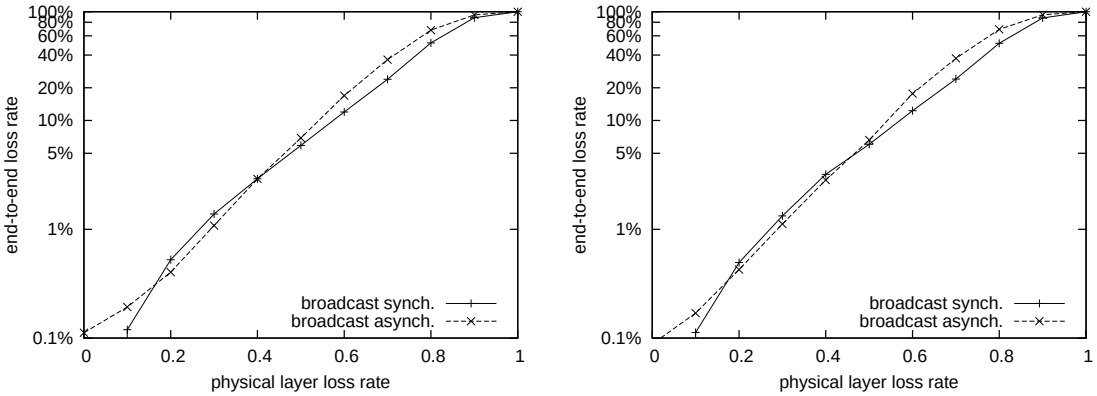


Figure 4.36: Performance of Asynchronous Aggregation for Two-state Gilbert Error Model (delay-tolerant, N = 160, S = 250m × 250m, P_b = 20%, fixed packet size)

4.3.3.5 Broadcast-based Synchronous vs. Broadcast-based Asynchronous

Fig. 4.37 shows that broadcast-based synchronous and asynchronous aggregation protocols yield similar performance for different physical layer loss rates. The main reason to use asynchronous timing control is to lower packet forwarding delay at

intermediate nodes. The broadcast-based asynchronous protocol does not have much advantage in the delay-tolerant scenario.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 4.37: Broadcast-based Synchronous vs. Broadcast-based Asynchronous Aggregation (delay-tolerant, varying τ, N = 160, S = 250m × 250m)

4.3.3.6 Summary of Results

Section 4.3.3 evaluates the performance of the protocols in the context of delay-tolerant data collection. Sections 4.3.3.1 to 4.3.3.3 present performance results with the independent random error model. The results show that when packet loss is random, the unicast-based protocols are able to improve reliability by increasing the maximum number of MAC layer retransmissions. The broadcast-based protocols relies on multipath routing for reliability, and their relative performance improves with higher density. The broadcast-based protocols transmit fewer packets per round than the unicast-based protocols. For fixed packet size, the broadcast-based protocols also transmit fewer bytes per round than the unicast-based protocols. For increasing packet size, traffic volume with broadcast-based aggregation decreases as the physical layer loss rate get higher, while more traffic is generated by unicast-based aggregation as the physical layer loss rate increases. Performance results with the two-state Gilbert error model are presented in Section 4.3.3.4. Both synchronous and asynchronous broadcast-based aggregation yield significantly improved reliability than their unicast-based counterparts, because MAC layer transmission is not

effective in loss recovery in this case.

4.4 Preliminary Experimental Results

This section reports on results from some preliminary experiments using implementations of the synchronous protocols on Crossbow MICAz motes. A picture of a MICAz mote is shown in Section 2.1 of Chapter 2. The MICAz is a 2.4 GHz, IEEE/ZigBee 802.15.4 compliant [20], board for wireless sensor networks. The main features of MICAz are listed in Table 4.2.

Program Flash Memory	128K bytes
Measurement Serial Flash	512K bytes
Configuration EEPROM	4K bytes
Frequency Band	2400 MHz to 2483.5 MHz
Transmit Data Rate	250 kbps
Outdoor Range	75 m to 100 m
Indoor Range	20 m to 30 m

Table 4.2: Features of MICAz

Operating systems for MICAz include TinyOS[2], Contiki[3], and Nano-RKT[4]. This dissertation uses TinyOS, a free and open source operating system designed for wireless embedded sensor networks. The TinyOS operating system and its applications are written in NesC[5], a dialect of the C programming language. TinyOS is component-based. A TinyOS application consists of one or more components. TinyOS code and program code are statically linked together, and compiled into a small binary.

Two programs were developed for each aggregation protocol. One program is intended to run on the sink node, and the other program is for nodes other than the sink. The MICAz mote that acts as the sink is connected to a MIB600, an Ethernet

[2]TinyOS: http://www.tinyos.net
[3]Contiki: http://www.sics.se/contiki
[4]Nano-RK: http://www.nano-rk.org
[5]Introduction to NesC, http://nesc.sourceforge.net/

Gateway produced by Crossbow. Through the gateway, the sink is able to receive commands issued from a computer in an Ethernet network, and report data collected from the sensor network to the computer in the Ethernet network. Transmissions from the sink to the other sensor nodes, for command/control purposes, are done by broadcasting with high power/long transmission range so that all nodes are able to receive them. For both the synchronous unicast and broadcast-based aggregation protocols, the program code for the sink has around 340 lines. The program code for nodes other than the sink has around 450 lines and 500 lines for synchronous unicast and broadcast-based aggregation, respectively. For diagnosis and debugging purposes, the programs support supplemental functions such as log reading, log writing, and LED (Light Emitting Diode) control.

For the experiments whose results are reported here, 24 MICAz motes are deployed in a lecture theatre with desks and chairs fixed to the floor. The area over which the motes are deployed is approximately 9 meters by 9 meters. The motes are fixed to the backs of the chairs, with one mote at the corner as the sink. Note that the chairs are not at the same height. The positions of the nodes are generated by a program by randomly selecting the 24 chairs from the 120 chairs inside the area. The transmission power level of the motes is set to 3, yielding an approximate transmission range of 2.5 meters to 3 meters. (However, the actual connectivity region around each node is highly irregular, as has been observed in previous studies [13].) In all experiments with unicast-based aggregation, the same aggregation tree shown in Fig. 4.38 is used, as built using a simple flooding algorithm. The aggregation tree has 7 hops. A much shorter tree is expected if the sink is located in the center area. In Chapter 3, the impact of sink placement is studied, and the results for both sink placement methods, sink in the center or sink at the corner, are consistent (Section 3.3.4). As there are only 24 motes in the experiments, a mote at the corner is selected as the sink to provide a multi-hop network with the largest potential number of hops.

For synchronous broadcast-based aggregation, a ring topology is formed with nodes i hops away in ring i. By default, the MICAz uses CSMA/CA at the MAC

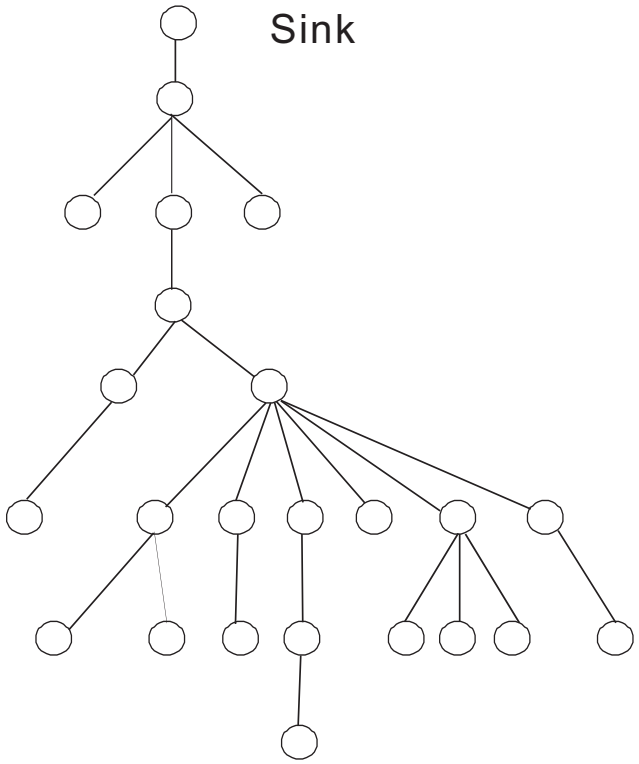


Figure 4.38: Aggregation Tree

layer, without packet retransmission. For unicast communication, the automatic retransmission functionality is activated at the motes and the maximum number of retransmissions is set to 3, 4 and 8 in the experiments. A fixed payload size of 28 bytes is used.

Multiple sets of experiments were carried out, with the experiments in each set using a range of values for the sampling period duration τ. Values for τ are chosen manually within the range of [0.125, 5] seconds in the experiments. For all protocols, the point with the highest end-to-end loss rate corresponds to the smallest τ, as seen in Fig. 4.39. Each single experiment runs for 200 sampling rounds. Fig. 4.39 shows the measured performance of the synchronous protocols from one set of these experiments. Results from the other sets of experiments show similar results.

Despite the performance variability seen in Fig. 4.39, the experimental results appear to be quite consistent with the simulation results shown in Fig. 4.3(b), 4.7(b), 4.8(b), 4.11(b), and 4.12(b). The performance variability can be explained by the differing characteristics of the experimental and simulated networks. As the sam-

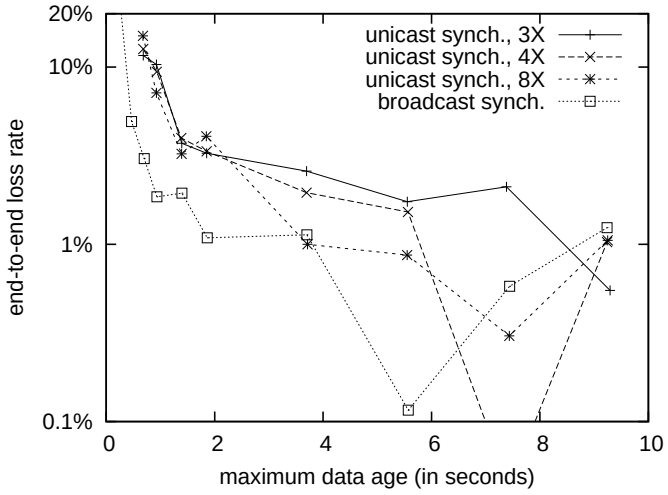


Figure 4.39: Experimental Results for Synchronous Aggregation (real-time, varying τ, N = 24, S = 9m $\times$ 9m, fixed packet size)

pling rate gets higher, the end-to-end loss rate starts to increases, as seen in Fig. 4.39. Unlike the results from simulations, the experimental results show substantial variability, especially at low sampling rates. Packet loss in the experiments comes from two main sources, contention and physical layer link error. Packet loss due to contention plays a key role in the end-to-end loss at high sampling rates. As τ increases, the amount of packet loss from contention decreases consistently and physical layer link error becomes the main reason for packet losses. In the experimental environment, the physical layer link loss is highly bursty, and communication may fail even with large numbers of retransmission attempts.

End-to-end loss rate is influenced by which nodes experience packet loss and their relative position in the aggregation tree. As can be seen in Fig. 4.38, there are several key nodes in the aggregation tree, the loss of whose packets results in the loss of the data of most nodes. Since each experiment only lasts for 200 sampling rounds, packet losses at those key nodes, even in just one or two rounds, can have a substantial impact on the measured end-to-end packet loss rate.

In Section 4.3, the performance of the protocols is evaluated through simulations. Compared to the testbed, the simulated networks have a much larger scale. A direct comparison to validate the simulation against the testbed would be quite beneficial to confirm the simulation setup. However, such a direct comparison is not easy

to implement due to the following two main reasons. First, in contrast to a fixed communication range as in the simulations, the connectivity region around each node is highly irregular in the experiments. Asymmetric links are also observed in the testbed. Simulating the testbed requires an accurate connectivity graph. Second, packet loss has to be modelled separately for each link. Prior work has showed the existence of a gray area within the communication range of a node [114]. The receivers in the gray area may lose 90% of the packets from the sender. Moreover, extreme burstiness in packet loss is observed for relatively long periods of time. Packet recovery may fail even with 8 MAC retransmission attempts. Simulating the testbed requires that measurements be colllected to characterize the links among the nodes. This can be an interesting furture work.

4.5 Summary

In this chapter, new synchronous and asynchronous broadcast-based aggregation protocols are proposed and compared with their unicast-based counterparts. When packet loss is random, the results show that for aggregation with fixed packet size, broadcast-based aggregation is able to achieve lower maximum data age than unicast-based aggregation in most real-time data collection scenarios, particularly in the case of synchronous protocols. However, for increasing packet size, the results do not show much performance advantage with broadcast-based protocols.

Broadcast-based aggregation yields significantly better performance than unicast-based aggregation for both real-time and delay-tolerant data collection when packet loss follows the two-state Gilbert error model. Preliminary experimental results from a real sensor network deployment are also reported in this chapter, and the results are quite consistent with the simulation results.

Chapter 5

Coverage Preserving Data Aggregation

5.1 Motivation

Area coverage is a common requirement in sensor network applications [11, 111]. One approach to achieving area coverage is through optimized placement of a minimal number of sensors [16, 110]. Often, however, it is not feasible to optimize placement, and in any case substantial redundancy may be desired owing to the possibility of sensor node and/or communication failures. For these reasons, sensor nodes may be deployed much more densely than would minimally be required. For example, security surveillance applications may require that each point of the monitored area be covered by at least k nodes ($k \geq 2$).

Node scheduling protocols may be used to conserve energy and prolong network lifetime [33, 54, 93, 99, 100, 104]. A potential drawback of node scheduling approaches, however, is that there may be a significant delay between (persistent or transient) node and/or communication failures and subsequent wake-up of replacement node(s). During this time area coverage may not be complete. One way to address this problem is to choose the set of active nodes so as to provide redundant coverage, with each point covered by k active nodes for some parameter $k > 1$ [54]. Choosing a greater number of active nodes, however, increases energy use and (possibly) network traffic load and contention. Note that even for $k = 2$, in failure-prone environments, there may be a significant probability of a loss of area coverage that could have been avoided if additional nodes had been chosen to be active rather than asleep.

This chapter proposes an alternative approach for efficiently maintaining area

coverage in dense sensor networks. This approach is applicable in contexts in which the objective is to periodically collect, and transmit to a sink node, sensing data that covers the region of interest. In the proposed approach, rather than being (semi-)statically scheduled, nodes *dynamically* determine during each round of data collection whether they should expend the energy required to transmit their data, or whether the set of neighbouring nodes that have already transmitted is sufficient to provide coverage.

In comparison to having all nodes collect and transmit their data, the proposed approach substantially reduces network traffic load and contention. Energy savings depend on the energy cost of data transmission in comparison to that of a potentially receiving but non-transmitting sensor node state. Techniques have been proposed that are able to substantially decrease the energy cost of the latter state [78].

Compared to node scheduling approaches, the simulation results that are presented in Section 5.4 suggest that the proposed approach can greatly improve reliability (yielding an order of magnitude reduction in uncovered area, in some cases), at the potential cost of increased traffic volume owing to non-minimal selection of transmitting nodes.

Both unicast and broadcast-based data aggregation protocols implementing this approach are proposed and compared to aggregation protocols relying on node scheduling. For the broadcast-based protocols considered in this chapter, it is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates including the same sensor value, either because aggregation is duplicate insensitive (e.g., only the maximum sensor reading is needed), or duplicates can be filtered (each aggregate is a concatenation of sensor values). The proposed protocols adopt the synchronous timing control strategy. Nodes are assigned to different transmission intervals based on their distance to the sink. The design of an asynchronous coverage preserving aggregation protocol is not feasible owing to the fact that the transmitting nodes change dynamically each round.

In contrast to implementing a node scheduling protocol and using an independent aggregation protocol that operates among the active nodes, protocols that integrate

aggregation and coverage preserving are designed and evaluated in this chapter. In the proposed protocols, which nodes transmit during each data collection round is dynamically determined. Both unicast-based and broadcast-based protocols of this type are proposed. In the unicast-based protocol, an aggregation tree is formed among all nodes, with the sink as the root, as in many previous aggregation protocols [26, 61]. In each round, the interior tree nodes aggregate the data received from their child nodes, and forward their aggregate packets towards the sink. The leaf nodes rely on overhearing to determine whether or not their sensing area has been covered by neighbouring nodes that have already transmitted during the round, and thus each such node may or may not transmit. The key problem in the design of such a protocol is how to dynamically determine node transmission orderings that are "efficient" in that nodes crucial to achieving area coverage transmit earlier, and so enable other nodes that hear their transmissions to remain silent. Similar to the synchronous broadcast-based aggregation protocols in Chapter 4, the transmission interval is divided into two phases. Nodes crucial for coverage are scheduled to the first phase so as to give the other nodes a better chance to cancel their transmissions.

Note that with the unicast-based coverage preserving protocol, failure of one of the interior tree nodes results in the loss of all of the data from the corresponding subtree. The broadcast-based protocol proposed in this chapter addresses this weakness by using a ring topology [67] instead of a static tree structure. As with the unicast-based protocol, the key design problem is that of dynamically determining the transmission orderings in a coverage-aware manner. A similar two-phase transmission scheme is adopted in the proposed broadcast-based protocol, with nodes crucial for coverage scheduled to the first phase.

Using simulation, the new protocols are compared with conventional unicast-based and broadcast-based data aggregation protocols relying on node scheduling. The comparisons are conservative with respect to the amount of generated network traffic in the node scheduling approach, since the approach assumes the optimal selection of a minimal coverage set. In practice, this optimal protocol would not be possible, owing at least to the need for cycling nodes between active and sleep

states, rather than statically choosing a minimal coverage set to remain active at all times. The performance of the protocols is evaluated for both an independent random error model and a two-state Gilbert error model. For both error models, the proposed broadcast-based protocol is found to provide greatly improved reliability in some cases, at the potential cost of increased traffic volume. The unicast-based protocol, in contrast, is found to yield similar reliability as unicast-based aggregation with node scheduling.

The remainder of the chapter is organized as follows. Section 5.2 presents new unicast-based and broadcast-based coverage preserving aggregation protocols. Section 5.3 describes the baseline protocols relying on node scheduling against which the performance comparisons are made. Simulation results evaluating the performance of the new protocols in comparison to that of the baseline protocols are given in Section 5.4. Section 5.6 concludes the chapter.

5.2 Coverage Preserving Aggregation

It is assumed that sensor readings are made periodically with period duration τ. Sufficient data is to be returned to the sink each round, so that for each point in the monitored region, the sink receives the data from at least one sensor whose sensing range covers that point.

The proposed "coverage preserving" aggregation protocols integrate data collection using aggregation together with dynamic determination of the nodes from which data should be collected. In each round, each node (other than the sink) will transmit only if there exists any point within its sensing area that is not covered by those neighbouring nodes that have already transmitted, or if the node must forward data received from other nodes (e.g., is an interior node in an aggregation tree). The goal is to minimize the number of transmitting nodes while ensuring no (or minimal) loss of area coverage.

It is assumed that each node knows its own location and the locations of its neighbouring nodes. As in previous work [54, 93, 99, 100, 104], a deterministic

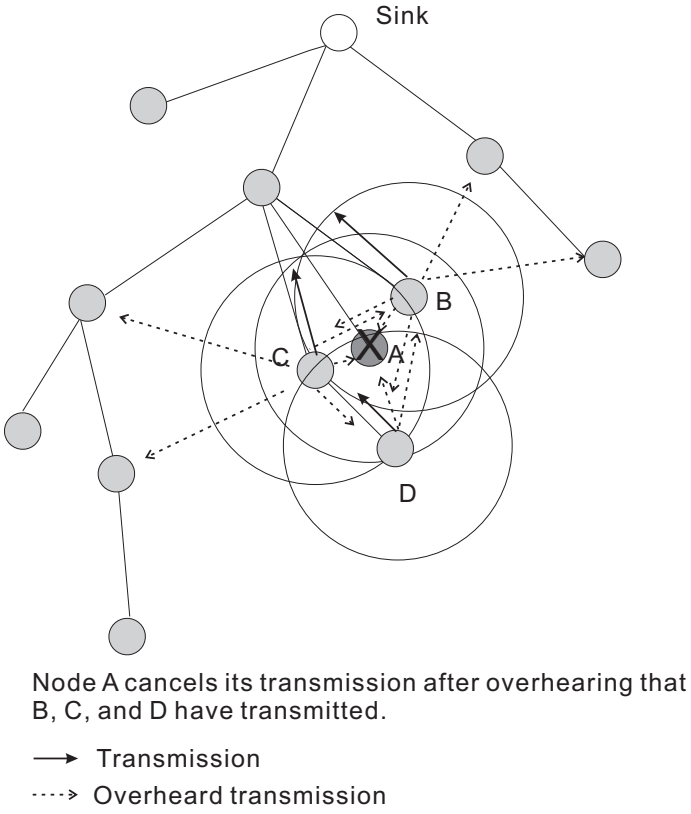


Figure 5.1: Unicast-based Coverage Preserving Aggregation

sensing range is assumed. A point p is covered by a node n if the distance between p and n is less than the node's sensing range S; i.e., node n provides coverage of a region bounded by a circle with radius S. It is assumed that the transmission range of each node is sufficiently long to reach all nodes whose sensing area overlaps with its own. Sections 5.2.1 and 5.2.2 present unicast and broadcast-based coverage preserving aggregation protocols, respectively.

5.2.1 Unicast-based

The proposed unicast-based protocol is tree-based, with the union of the routes from the sensors to the sink forming an aggregation tree with the sink as its root node, as shown in Fig. 5.1. In this figure, as well as in Fig. 5.2, circles indicate sensing ranges, and it is assumed that the transmission range is twice the sensing range. When transmission range is at least twice the sensing range, a node is able to hear all the nodes whose sensing range overlaps with its own. Furthermore, 1-coverage

of a convex area implies connectivity if the transmission range is at least twice the sensing range, i.e., when the communication range is at least twice the sensing range, the nodes must be connected if the nodes are able to cover the whole monitored area [99]. To simply this work, it is also assumed in the performance evaluation that the transmission range is twice the sensing range, in which case the nodes that cover the whole monitored area must be connected. As in TAG [61], nodes at different levels of the tree are assigned to different transmission intervals. All intervals are of identical duration I. The transmission interval of node i is chosen as interval $H - h_i$, where h_i denotes the hop count to the sink from node i and H denotes the maximum hop count (i.e., the tree height).

In the initial round of data collection, each node i picks a random value r_i^1 between 0 and $0.8I$, aggregates the data it has received for this round, and schedules its packet transmission for time $T_0 + (H - h_i)I + r_i^1$. Here it has been assumed that all nodes agree on the same base time T_0 defining the beginning of the first round. The random values are used to spread out the data transmissions across most of the respective interval, with a gap providing separation from the next interval. (Spreading the transmissions over $0.8I$ yields good performance, see Section 3.3.1.) At the scheduled transmission time, node i actually transmits only if it is an interior (i.e., non-leaf) node, or if it has not overhead data transmissions, or acknowledgements of data transmissions, from a set of nodes that cover node i's sensing area. Otherwise, node i cancels its transmission.

For each subsequent round $j, j > 1$, each node i divides its respective interval into two phases, of durations $0.8f_iI$ and $0.8(1 - f_i)I$, respectively, with a gap of $0.1I$ between the phases and a gap of $0.1I$ separating the second phase from the beginning of the next interval. (The value of f_i is dynamically adjusted, as described below.) Nodes that must forward data received from other nodes, and nodes that are dynamically determined to be important to achieving area coverage, schedule their transmissions for their first phase. In particular, since interior tree nodes must forward data received from their children, they transmit during their first phase. Specifically, an interior node i picks a random value r_i^j between 0 and

$0.8f_iI$, aggregates the data it has received for round j, and sends out its packet at time $T_0 + (j-1)\tau + (H - h_i)I + r_i^j$.

The action taken by a leaf node depends in part on what happened during the previous round $j-1$. Specifically, if, by the end of round $j-1$, a leaf node i has overheard data transmissions or acknowledgements of data transmissions, from a set of nodes that cover node i's sensing range, then node i schedules its transmission for round j to be during its second phase. The timeout value is set to time $T_0 + (j-1)\tau + (H - h_i)I + r_i^j + (0.8f_i + 0.1)I$, where r_i^j is a random value between 0 and $0.8(1-f_i)I$. Otherwise, node i schedules its transmission to be during its first phase, at time $T_0 + (j-1)\tau + (H - h_i)I + r_i^j$, where r_i^j is a random value between 0 and $0.8f_iI$. At the scheduled transmission time, node i actually transmits only if it has not overhead, during the current round j, data transmissions, or acknowledgements of data transmissions, from a set of nodes that cover node i's sensing range. Otherwise, node i cancels its transmission for that round.

The relative durations of the phases into which node i divides its respective interval (as determined by the value of f_i) are dynamically determined so as to match the anticipated relative traffic volumes in node i's neighbourhood, using an EWMA strategy. For round 2, f_i is chosen as 1. Let F_i^j and S_i^j denote the number of first-phase and second-phase data transmissions that node i hears during round $j \geq 2$. (Each data packet includes a bit indicating which phase the transmitting node was in when the transmission occurred.) At the end of round j, node i takes as its updated value of f_i a weighted average of the old value of f_i, and the measured fraction of first-phase transmissions during that round: $(1-\delta)f_i + \delta(F_i^j/(F_i^j + S_i^j))$, where δ (set to 0.125 for the simulations of Section 5.4) is a parameter determining the weight given to the most recent measurement. The idea is that the proportion of time for the first phase should be kept consistent with the expected proportion of the traffic generated for the first phase. If more traffic is generated for the first phase than the second phase, then the first phase should last longer than the second phase. In coverage preserving aggregation, the number of nodes that transmit per round varies a lot with different network settings. It is difficult to manually determine

the duration of the two phases in a manner yielding consistently good performance. The adaptive strategy solves the problem by dividing the transmission interval using history information of traffic volumes. The results concerning the number of nodes that transmit per round are reported in Section 5.4.

5.2.2 Broadcast-based

The broadcast-based coverage preserving aggregation protocol organizes nodes into a ring topology [67], rather than tree. As shown in Fig. 5.2, the sink is the only node that is located in ring 0, nodes one hop away from the sink are in ring 1, and in general nodes h hops away are in ring h. As in the unicast-based protocol, nodes in different rings are allotted different time intervals within each round of communication for their transmissions, with a constant interval duration I. Denoting the maximum hop count from the sink by H, each node i in ring h_i transmits in interval $H - h_i$. Each node i aggregates all of the data it has received from broadcasts for the current round (from neighbouring nodes in ring $h_i + 1$, as well as from neighbouring nodes in the same ring h_i that transmitted earlier within the interval) for its own broadcast. Each broadcast packet includes a bit vector indicating the nodes whose data is aggregated in the packet. Note that this protocol assumes it is acceptable for the sink to receive multiple aggregates including the same sensor value.

For each round (including the first), each node i divides its respective interval into two phases, of durations $0.8 f_i I$ and $0.8(1 - f_i)I$, respectively, with a gap of $0.1I$ between the phases and a gap of $0.1I$ separating the second phase from the beginning of the next interval. For the first round, f_i is chosen as 0.9. For subsequent rounds, the value of f_i is updated as in the unicast-based protocol. To improve reliability, in some cases a node may transmit in both phases. (Since broadcast is being used, there are no link layer acknowledgements.)

A scheduled transmission by a node i is cancelled if *both* of the following conditions hold at the time the transmission was to have occurred:

- for each node k in ring $h_i + 1$ from which that node i has heard a broadcast

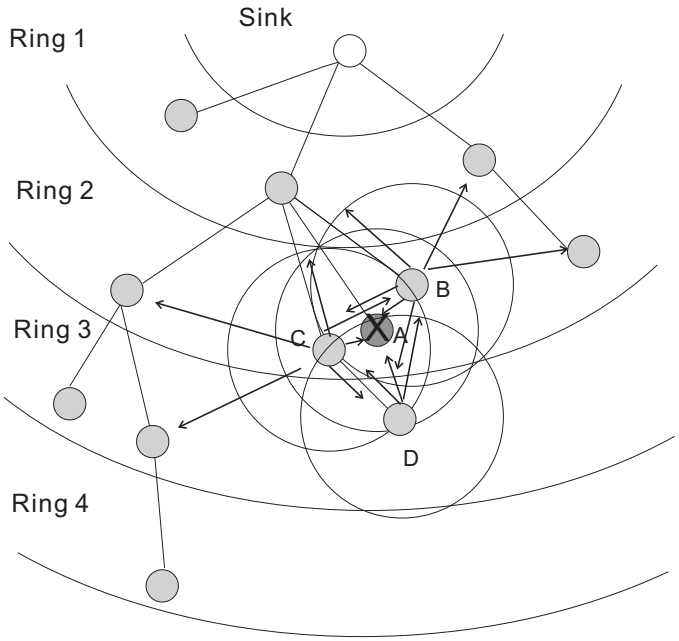


Figure 5.2: Broadcast-based Coverage Preserving Aggregation

for the current round, node i has also heard a broadcast from some other node in ring h_i that has included node k's sensor data in its aggregate;

- the set of other nodes whose data node i has aggregated for the current round (from the broadcasts it has received) cover node i's sensing area. Node i knows whose data it has from the bit vectors carried by the received broadcasts.

The first of these conditions is needed to ensure that for each node in a ring $h_i + 1$, there is at least one node in ring h_i that forwards its data.

Nodes schedule their transmissions as follows. For the first round, each node i picks a random value $r1_i^1$ between 0 and $0.8f_iI$, and schedules a broadcast for time $T_0 + (H - h_i)I + r1_i^1$ within the first phase of its respective interval. If this first broadcast is made (i.e., one or both of the above conditions did not hold), another random value $r2_i^1$ is picked between 0 and $0.8(1 - f_i)I$, and a second broadcast is scheduled (for improved reliability) for time $T_0+(H-h_i)I+(0.8f_i+0.1)+r2_i^j$ within the second phase. This second broadcast is made only if one or both of the above conditions does not hold, *and* only if node i has not heard a broadcast from some

other ring h_i node, subsequent to node i's first phase broadcast, that includes its data from this broadcast. Similar to the broadcast-based protocols in Chapter 4, a node only collects one sampling value during each round to keep the data consistent.

For each subsequent round j, $j > 1$, nodes schedule transmissions much as described above for the first round, with the exception that a transmission is scheduled for the first phase of the respective interval *only if* one or both of the above bulleted conditions did not hold at the end of the previous round. Otherwise, a transmission is scheduled (only) for the second phase.

Broadcast-based coverage preserving aggregation is more resilient to persistent node failure than unicast-based coverage preserving aggregation. In unicast-based coverage preserving aggregation, the failure of a parent node results in the loss of all of the data from the corresponding subtree, and all data from its children is lost before the child nodes find a new parent. In broadcast-based coverage preserving aggregation, a node may have multiple neighbours in the next ring closer the sink. Even if all these neighbours fail, the node's data may still be received and forwarded to the sink by its neighbours in the same ring before it switches to a new ring.

5.3 Data Collection with Node Scheduling

The performance of the proposed protocols is compared to that obtained by using node scheduling together with conventional unicast-based and broadcast-based data aggregation protocols. Rather than evaluate any particular practical node scheduling protocol, an optimization algorithm is applied to find a minimal coverage set; i.e., a smallest set of active nodes such that the sensing area they cover is the same as that of the set of all nodes. Data collection using aggregation is then performed over the nodes in this minimal coverage set. The specific aggregation protocols used are the unicast-based synchronous aggregation protocol described in Section 3.3.1, and the broadcast-based synchronous aggregation protocol described in Section 4.2.1.

Section 5.3.1 describes how a minimal coverage set is found. Sections 5.3.2 and 5.3.3 discuss the unicast and broadcast-based data aggregation protocols that

are used for comparison with the proposed protocols.

5.3.1 Minimal Coverage Set

The minimal coverage set (MCS) problem is formulated as a binary integer programming problem in Matlab[1]. The region over which the sensor nodes are deployed is divided into small cells. Letting m denote the number of cells and n the number of sensor nodes, the sensing area coverage of the nodes is encoded in an $m \times n$ matrix A, where $A_{i,j} = 1$ when cell i is covered by node j, and otherwise $A_{i,j} = 0$. A cell is considered to be "covered" by a node if the center of the cell is within the node's sensing range. This approximation may result in a smaller than actual minimal coverage set, but the impact of discretization can be made negligible by choosing a sufficiently small cell size.

Let b be a vector with m entries, such that $b_i = 1$ when cell i is covered by any of the nodes and zero otherwise. Note that the value of b_i is set to 1 if *any* of the nodes inside the network covers cell i. The value of b_i is set to 0 only if *none* of the nodes covers that cell. Let f be a vector with n entries each set to 1. A minimal coverage set is then given by a binary integer vector x that minimizes $f^T x$ under the constraint $Ax \geq b$, where node i is included in the set if $x_i = 1$. Node i is excluded from the set if $x_i = 0$.

In Matlab, x = bintprog(f,A,b) solves the binary integer programming problem of minimizing $f^T x$ such that $Ax \leq b$. To solve the minimum coverage set problem, x = bintprog(f,-A,-b) is used.

5.3.2 Unicast-based Aggregation

The unicast-based aggregation protocol described in Section 3.3.1 is used for comparison with the coverage preserving aggregation protocols. The protocol matches the coverage preserving unicast-based protocol proposed in Section 5.2.1, except for the protocol elements concerning coverage. An aggregation tree is used, but now just

[1] http://www.mathworks.com/

including the nodes in the minimal coverage set. Nodes at different levels of the tree are assigned to different intervals within each round of communication, as previously. Each round, nodes schedule their transmissions in their respective intervals exactly as is done in the first round in the coverage preserving protocol. Unlike in that protocol, there is no protocol mechanism for cancelling a scheduled transmission.

5.3.3 Broadcast-based Aggregation

The broadcast-based aggregation protocol described in Section 4.2.1 is used in the performance comparison presented in Section 5.4. Similar to the case of unicast-based aggregation, the broadcast-based aggregation protocol in Section 4.2.1 matches the coverage preserving broadcast-based protocol proposed in Section 5.2.2, except for the protocol elements concerning coverage. The nodes in the minimal coverage set are organized into a ring topology, with nodes in different rings assigned to different transmission intervals within each round of communication, as previously.

Each interval is divided into a first and second phase, but unlike in the coverage preserving protocol, the durations of these phases are determined by a fixed parameter λ (chosen as 0.8 in the experiments whose results are presented here). All nodes (in the minimal coverage set) transmit in the first phase, and some nodes also transmit in the second phase as well, for improved reliability.

5.4 Comparative Evaluation

Performance is evaluated for both real-time and delay-tolerant applications using ns2 simulations. The metrics considered are the following: (1) the average percentage of uncovered area in each round of data collection, (2) the average number of MAC layer packet transmissions per round (for unicast, including both initial data packet transmissions, and link layer retransmissions and acknowledgements), (3) the average number of nodes that transmit per round, and (4) the average number of bytes transmitted per round. The last three metrics yield insight into relative energy usage. The uncovered area in each round is the area that is not covered by the

nodes whose data is included in the aggregates that are successfully received at the sink. For real-time applications, the maximum data age is measured, which shows how "stale" the data received at the sink from one round can become before that for the next round is received. Recall that the maximum data age for each round is calculated as τ plus the maximum data collection delay. The figures plot the average of the maximum data ages over multiple rounds.

Node area coverage is determined as described in Section 5.3.1, based on a division of the region into cells. Sensor fields are generated by randomly scattering nodes in square areas. The node closest to the center of the area is selected as the sink.[2] Section 5.4.1.1, Section 5.4.1.2, Section 5.4.2.1, and Section 5.4.2.2 show results for a sensor network with 320 nodes deployed over a 250 meter by 250 meter area. The cell size that is used to determine node area coverage is 2 meters by 2 meters, giving 15625 cells in total for this network. With all 320 nodes, about 99.4% of the area is covered. The size of a minimal coverage set (covering the same area) is 96 nodes. Section 5.4.1.3 and 5.4.2.3 shows performance results for both lower and higher density networks.

As in Chapter 3 and Chapter 4, An 802.11 MAC layer is simulated for the unicast-based aggregation protocols, without using RTS/CTS [102], assuming each node has a transmission range of 40 meters and data rate of 2 Mbps. Different maximum numbers of link layer retransmissions (3 and 8) are simulated for when the sender fails to receive an acknowledgement. The same transmission range and data rate are used for the broadcast-based protocols. All nodes have a sensing range of 20 meters. As the transmission range is twice the sensing range, it is guaranteed that the nodes in the minimum coverage set are connected.

As in Chapter 4, no specific assumptions are made regarding the type of aggregation that is performed; instead, two extreme cases are considered with respect to how packet size grows with the number of aggregated values. In one of these, it is assumed that sensor data can be aggregated into packets of size that is independent

[2]When evaluating aggregation using a minimal coverage set, the sink is selected from only these nodes (optimistically for this approach).

of the number of aggregated values (chosen as 52 bytes in the simulations reported here). In the other, required packet size is assumed to increase linearly with the number of aggregated values (at 4 bytes per value). Similarly, 3X, 4X, and 8X denote up to 3, 4, and 8 MAC layer retransmissions in the figures.

Section 5.4.1 and 5.4.2 evaluate protocol performance for delay-tolerant and real-time data collection, respectively. Two error models are considered: independent random error model and two-state Gilbert error model. Physical layer packet loss occurs independently for each packet, with a fixed probability, and there are no node failures in independent random error model. The two-state Gilbert error model can reflect longer link outages and partial node failures, depending on the parameter settings. The impact of network density on performance is explored in Section 5.4.1.3 and 5.4.2.3, for delay-tolerant and real-time data collection, respectively.

Table 5.1 lists the parameters that are used in the performance evaluation.

τ	The time between successive sensor readings at each node.
PLR	The physical layer loss rate for the independent random error model.
N	The number of nodes inside the network. All sensor networks for the performance evaluation have 320 nodes.
N_{MCS}	The number of nodes in the minimum coverage set of a sensor network.
S	The size of the square area inside which N nodes are scattered.
λ	Protocol parameter for unicast and broadcast-based aggregation with minimum coverage set. The transmission times of the nodes at the same tree level are scattered over λI for each round. The value of λ is fixed at 0.8.
τ_b	The average duration of the bad state in the two-state Gilbert model.
P_b	Proportion of time that is spent in the bad state in the two-state Gilbert model.

Table 5.1: Simulation Parameters for Chapter 5

Similar to the simulations with the synchronous protocols in Chapter 3 and Chapter 4, for all protocols, each simulation run lasts 1,150 sampling period durations, with the initial 200 and the last 50 sampling period durations removed from the measurements. As in Chapter 4, the results reported below are based on one simulation run.

5.4.1 Performance for Delay-Tolerant Data Collection

This section evaluates the performance of the aggregation protocols for delay-tolerant applications. The sampling period duration τ (and corresponding interval duration I, chosen as τ divided by the maximum hop count to the sink) is chosen sufficiently large that network contention is minimal.

5.4.1.1 Performance for Independent Random Error Model

Figures in this section show the performance of both the proposed coverage preserving aggregation protocols, and the conventional aggregation protocols as applied to a minimal coverage set, for the independent random error model. Each figure plots one of the four performance metrics as a function of the physical layer packet loss rate (expressed as a loss probability).

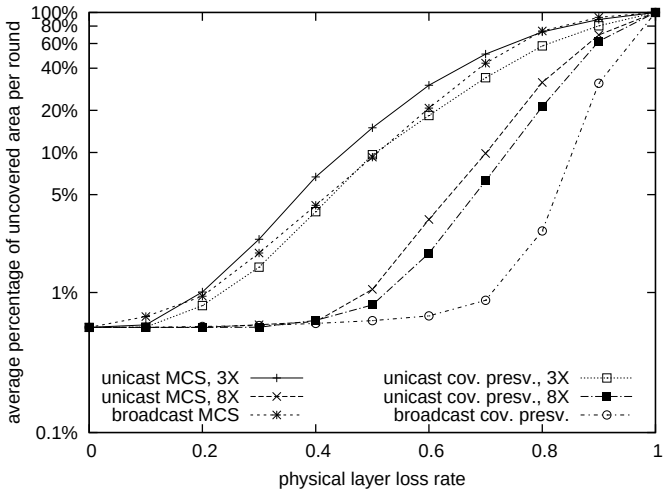


Figure 5.3: Average Percentage of Uncovered Area per Round for Independent Random Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, fixed packet size)

Fig. 5.3 shows the average percentage of uncovered area in each round, for the case in which the packet size is fixed. Results for when the packet size increases with the number of aggregated values are very similar. As the loss probability increases beyond 0.1, broadcast-based coverage preserving aggregation significantly outperforms broadcast-based aggregation with minimal coverage set, and all of the

unicast-based protocols with a link layer retransmission limit of 3. Broadcast-based coverage preserving aggregation begins to outperform the unicast-based protocols with a link layer retransmission limit of 8, as well, once the loss probability exceeds 0.4. Broadcast-based coverage preserving aggregation is able to keep more than 99% of the area covered at 70% physical layer loss rate. These results are explained by the fact that as the packet loss rate increases, the broadcast-based coverage preserving protocol correspondingly increases the number of nodes transmitting their sensor data in each round, as seen in Fig. 5.4. Note in this figure that the number of transmitting nodes is fixed at 96 for the minimal coverage set protocols.

Unicast-based coverage preserving aggregation, however, yields only modestly improved coverage in comparison to unicast-based aggregation with minimal coverage set. The relatively poor performance in comparison to that with broadcast-based coverage preserving aggregation shows the impact of packet loss by the interior tree nodes. With the unicast-based protocols, packet loss at one of the interior tree nodes results in the loss of all of the data from the corresponding subtree.

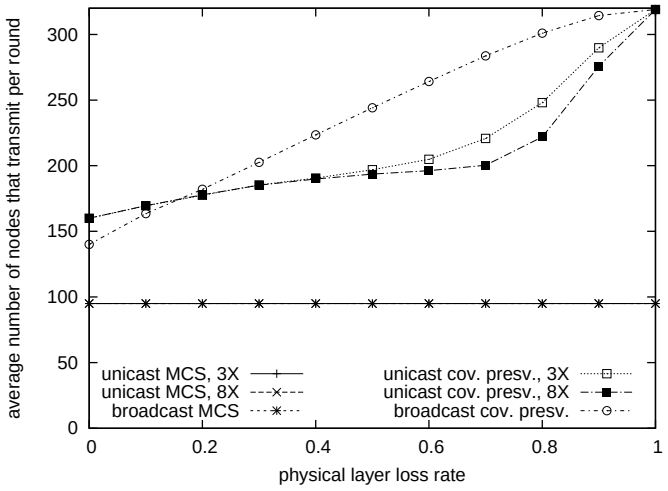


Figure 5.4: Nodes that Transmit per Round for Independent Random Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, fixed packet size)

Fig. 5.5 shows the average number of MAC layer packet transmissions per round (for unicast, including link layer retransmissions and acknowledgements), for the case in which the packet size is fixed. Results for increasing packet size are very similar.

As seen in Fig. 5.5, the broadcast-based protocols send relatively few packets; this is since each node transmits its data at most twice with these protocols, and since nodes do not send ACKs. Note that more packets are sent with coverage preserving aggregation than when using the corresponding protocol with minimal coverage set, even for a packet loss rate of 0. This reflects the fact that a greater number of nodes transmit with this approach, as seen in Fig. 5.4. The figure for increasing packet size is very similar.

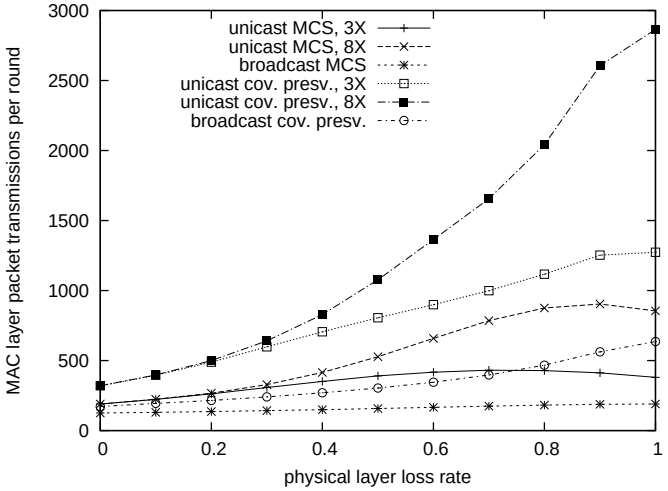


Figure 5.5: MAC Layer Packet Transmissions per Round for Independent Random Error Model (delay-tolerant, N = 320, N_{MCS} = 96, S = 250m × 250m, fixed packet size)

Fig. 5.5 shows that for unicast-based aggregation with minimum coverage set, the average number of MAC layer packet transmission per round drops when the physical layer loss rate reaches 80% (for up to 3 retransmissions) and 90% (for up to 8 retransmissions). Fig. 5.6 and Fig. 5.7 show that while the average number of MAC layer data packets per round keeps increasing as the physical layer loss rate gets higher, the average number of ACK packets per round drops when loss probability is high enough. When a packet is transmitted using unicast, the destination node is expected to send an ACK to the sender when the packet is received. When the loss probability reaches 60% or even higher, fewer packets arrive at the their destination nodes. As a result, the number of ACK transmissions drops. For unicast-based aggregation with minimum coverage set, the decreasing number of ACK packets per

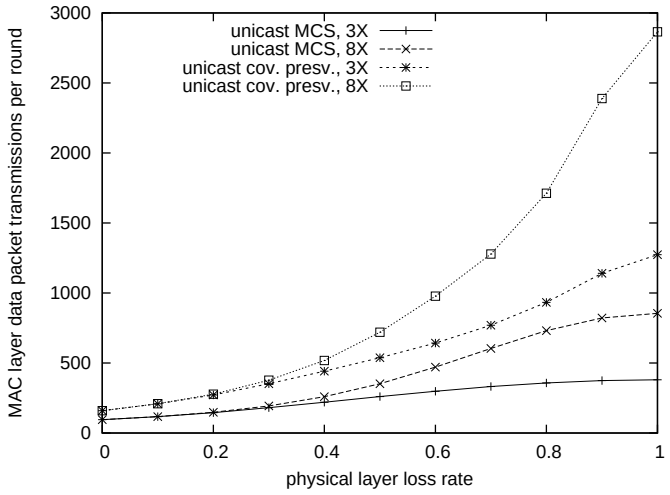


Figure 5.6: MAC Layer Data Packet Transmissions per Round for Independent Random Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, fixed packet size)

round eventually causes the total number of MAC layer packets per round to drop. For unicast-based coverage preserving aggregation, more nodes transmit as the loss probability get higher, and the average number of MAC layer packet transmission per round increases accordingly.

The average number of bytes transmitted per round is shown in Fig. 5.8 for both the fixed and increasing packet size cases. For increasing packet size, the data volume with broadcast-based coverage preserving aggregation is typically considerably larger than with the other protocols, owing to cases in which the same sensor value is redundantly included in multiple aggregates. The data volume with this protocol substantially decreases as the loss probability exceeds 0.7, since the growing packet loss decreases this redundancy.

5.4.1.2 Performance for Two-state Gilbert Error Model

This section reports performance results for both the proposed coverage preserving aggregation protocols, and the conventional aggregation protocols as applied to a minimal coverage set, for a two-state Gilbert error model.

Similar to the two-state Gilbert error model that is used in the performance comparison in Section 4.3, there is a "good" state and a "bad" state in the error

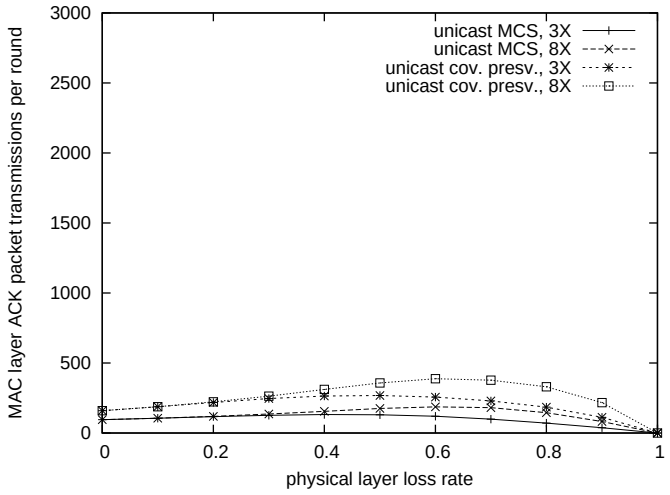
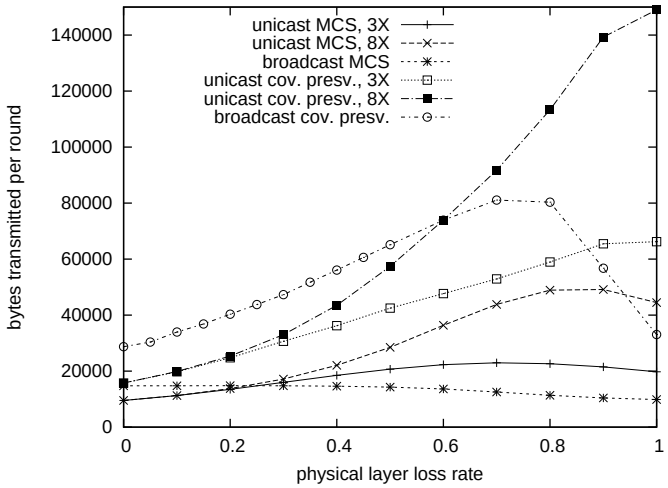


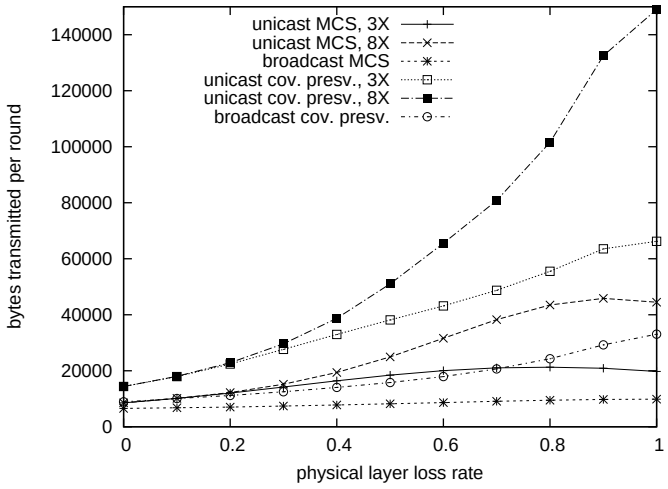
Figure 5.7: MAC Layer ACK Packet Transmissions per Round for Independent Random Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, fixed packet size)

model. When the node is in the "good" state, all packets are received correctly unless they collide with other packets. No packets are received when the node stays in the "bad" state. It is assumed that the sink is a special node, which always stays in the "good" state. The time a node spends in a state before transiting to the other state (the sojourn time) is exponentially distributed. By varying the average sojourn time, a range of scenarios can be modelled, from independent random packet loss (with very short sojourn times) to link outages and partial node failures (with long sojourn times).

Figs. 5.9 to 5.12 show performance results for the case in which the packet size is fixed. In Fig. 5.9, each node spends 20% of its time in the bad state, and the average sojourn time in that state (and the corresponding average sojourn time in the good state) is varied. As seen in the figure, broadcast-based coverage preserving aggregation is able to provide much better coverage in the case of long duration failures than are the unicast-based protocols. This is since the latter protocols rely on link layer retransmissions, which are ineffective in this case. Broadcast-based coverage preserving aggregation also yields better coverage than broadcast-based aggregation with minimal coverage set, since the former protocol can often rely on other nodes to compensate for the nodes in the bad state. Note that for a very small



(a) Increasing Packet Size



(b) Fixed Packet Size

Figure 5.8: Bytes Transmitted per Round for Independent Random Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m)

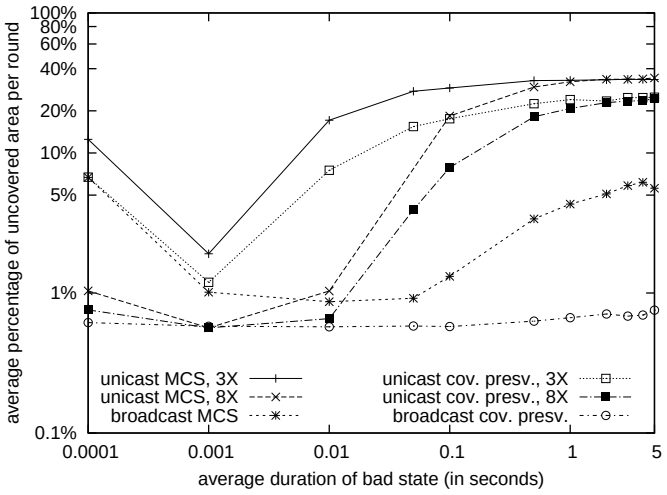


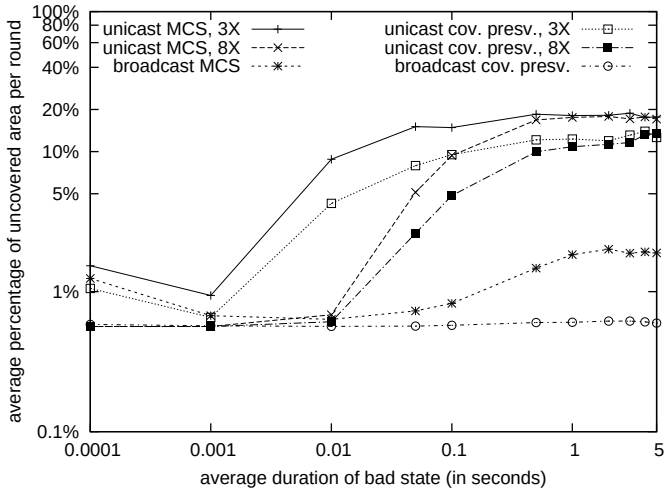
Figure 5.9: Average Percentage of Uncovered Area per Round for Gilbert Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, $P_b = 20\%$, fixed packet size)

average sojourn time, such as 0.0001 seconds in the figure, the average percentage of uncovered area in each round increases, since a node is likely to enter the bad state at least once while receiving a packet, causing the loss of that packet.

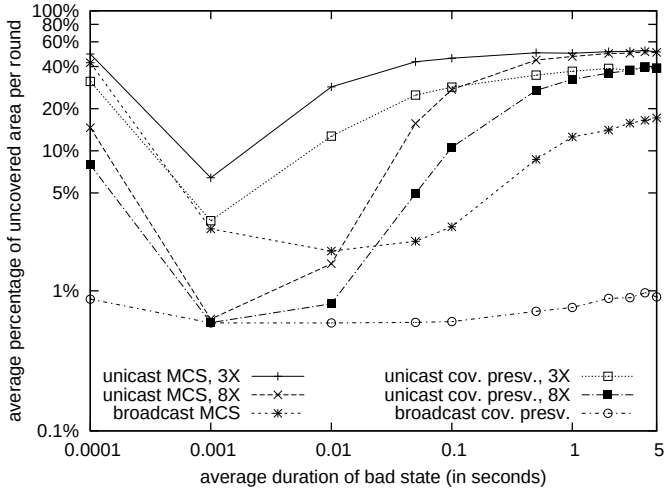
Qualitatively similar results are seen in Fig. 5.10 for different values of the proportion of time each node spends in the bad state. Note that the reliability advantage of broadcast-based coverage preserving aggregation, compared to the other protocols, increases as the reliability of the network decreases.

Fig. 5.11 and Fig. 5.12 plot the number of nodes that transmit per round and the average number of MAC layer packet transmissions per round, respectively. In these two figures, as well as Fig. 5.13, the proportion of time each node spends in the bad state is fixed at 20%.

Fig. 5.13 shows the average number of bytes transmitted per round, for both the fixed and increasing packet size cases. As is the case for the independent random error model, although broadcast-based coverage preserving aggregation transmits relatively few packets per round, in the case of increasing packet size these packets can become quite large, yielding a higher number of bytes transmitted. However, its relative performance with respect to this metric generally improves as the average state sojourn time in the Gilbert model increases.



(a) 10% of Time in Bad State



(b) 1/3 of Time in Bad State

Figure 5.10: Impact of Percentage of Time in Bad State for Gilbert Error Model (delay-tolerant, N = 320, $N_{MCS} = 96$, S = 250m × 250m, fixed packet size)

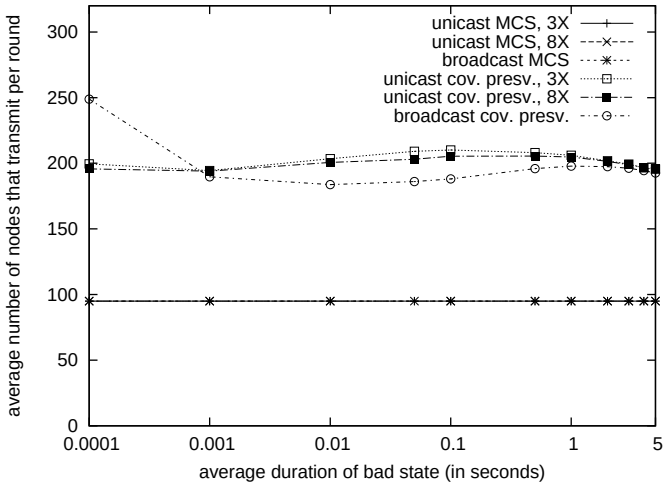


Figure 5.11: Nodes that Transmit per Round for Gilbert Error Model (delay-tolerant, N = 320, N_{MCS} = 96, S = 250m × 250m, P_b = 20%, fixed packet size)

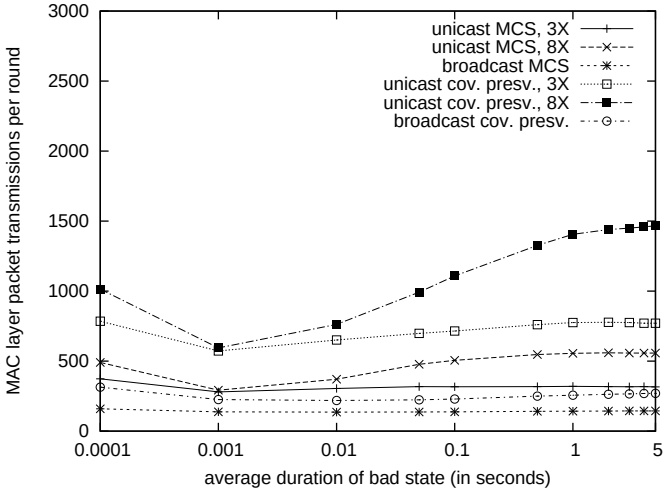
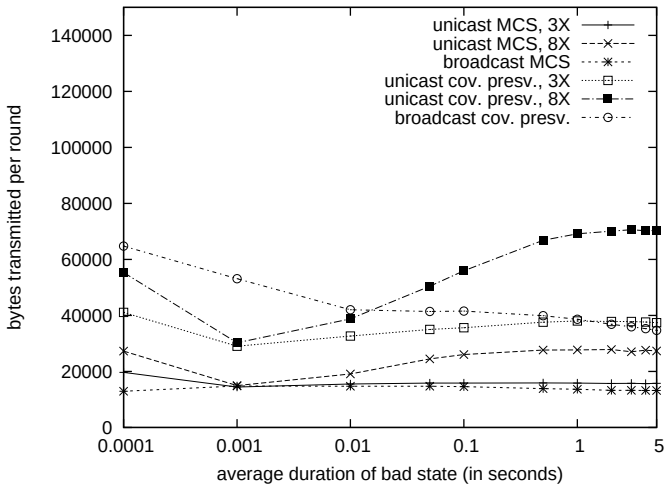
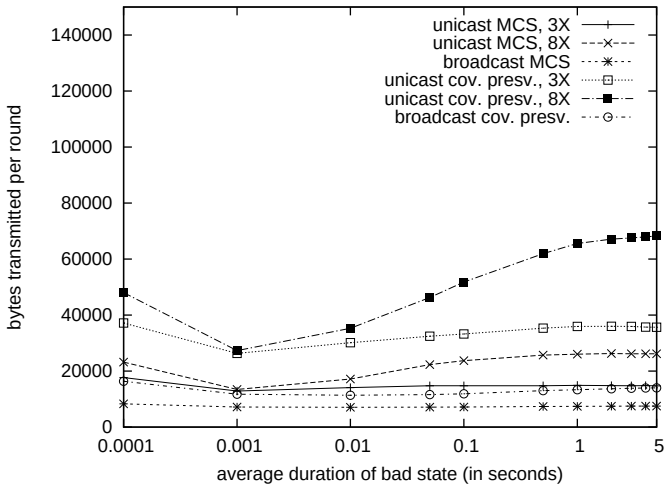


Figure 5.12: MAC Layer Packet Transmission per Round for Gilbert Error Model (delay-tolerant, N = 320, N_{MCS} = 96, S = 250m × 250m, P_b = 20%, fixed packet size)



(a) Increasing Packet Size



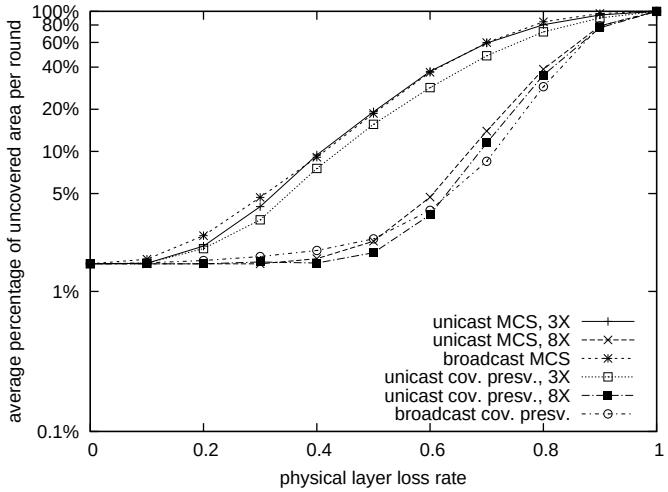
(b) Fixed Packet Size

Figure 5.13: Bytes Transmitted per Round for Gilbert Error Model (delay-tolerant, N = 320, N_{MCS} = 96, S = 250m × 250m, P_b = 20%)

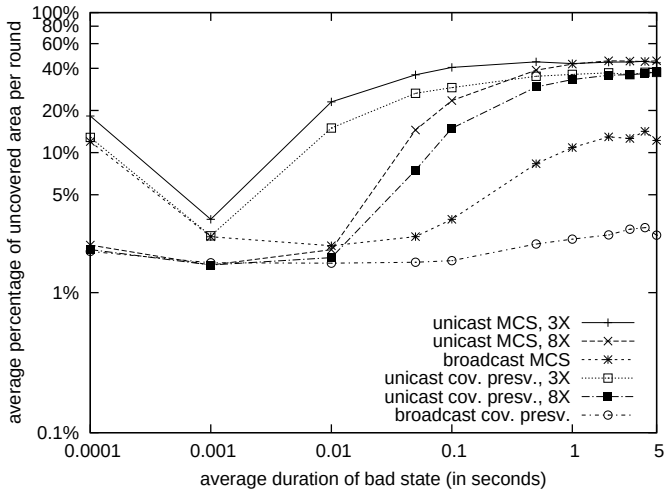
5.4.1.3 Impact of Network Density

This section reports performance results for sensor networks with lower density and higher density, respectively, than that considered previously. As with the previous sensor network, there are 320 nodes, but the area over which these nodes are scattered is 300 meters by 300 meters in Fig. 5.14, and 200 meters by 200 meters in Fig. 5.15. Only figures for fixed packet size are shown in this section; related figures for increasing packet size are very similar.

Given that the cell size is 2 meters by 2 meters, the sensor field with a 300 meter



(a) Independent Random Error Model

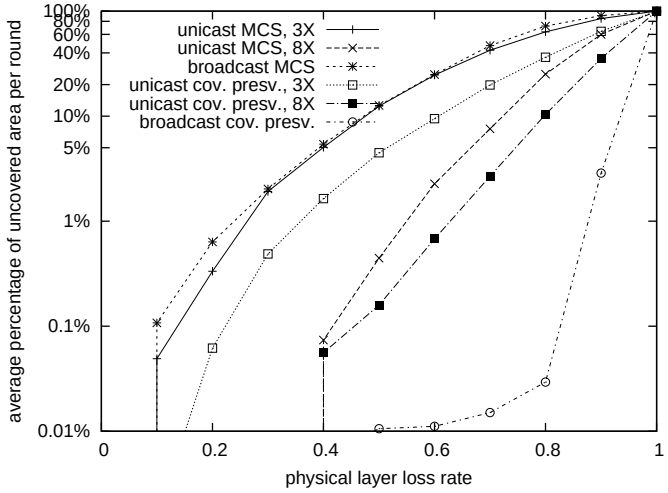


(b) Gilbert Error Model (20% of time in bad state)

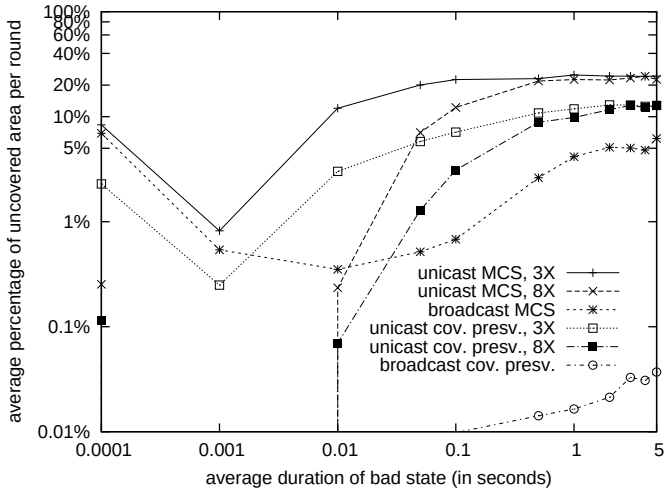
Figure 5.14: Performance for Lower Density Network (delay-tolerant, N = 320, N_{MCS} = 139, S = 300m × 300m, fixed packet size)

by 300 meter area has 22,500 cells in total. With all 320 nodes, about 98.4% of the monitored area is covered. The sensor field with a 200 meter by 200 meter area has 10,000 cells. With all 320 nodes, 100% of the monitored area is covered. With higher density, broadcast-based coverage preserving aggregation is able to achieve more than 99.9% coverage in all simulations with the two-state Gilbert model. The other protocols are also able to get more than 99.9% of the monitored area covered for some of the simulation parameters. Therefore, in Fig. 5.15, the y-scale is changed to plot the performance results. In Fig. 5.15(b), unicast-based coverage preserving

aggregation with up to 8 MAC layer transmissions yields 0% uncovered area when the average sojourn in the bad state is 0.001 seconds. That point cannot be plotted as the y-axis is shown in log scale.



(a) Independent Random Error Model



(b) Gilbert Error Model (20% of time in bad state)

Figure 5.15: Performance for Higher Density Network (delay-tolerant, N = 320, N_{MCS} = 54, S = 200m × 200m, fixed packet size)

Comparing the results in Figs. 5.14 and 5.15 to those in Figs. 5.3 and 5.9 (for area of 250 meters by 250 meters), the main observation is the dramatic impact of density on the relative performance of the protocols. In particular, the higher the density, the greater the reliability improvements with broadcast-based coverage preserving aggregation.

5.4.1.4 Summary of Results

In Section 5.4.1, the proposed unicast and broadcast-based coverage preserving aggregation protocols are evaluated and compared with data aggregation relying on node scheduling in the context of delay-tolerant data collection. Section 5.4.1.1 and Section 5.4.1.2 present performance results with the independent random error model and the two-state Gilbert error model, respectively. Performance results for networks with higher and lower density are presented in Section 5.4.1.3. When packet loss is random, for the network with 320 nodes over a 250 meter by 250 meter area, broadcast-based coverage preserving aggregation yields better coverage than all the other protocols when the physical layer loss rate is above 40%. For the network with lower density, broadcast-based coverage preserving aggregation and the unicast-based protocols with up to 8 MAC layer retransmissions yield similar performance. The reliability improvements with broadcast-based coverage preserving aggregation increase as the network density gets higher. Broadcast-based coverage preserving aggregation yields significantly better coverage when packet loss follows the two-state Gilbert error model. In all scenarios, the unicast-based coverage preserving aggregation protocol yields similar reliability as unicast-based aggregation minimum coverage set.

5.4.2 Performance for Real-time Data Collection

This section evaluates the performance of the aggregation protocols for real-time applications that aim to keep a "current" view of the monitored area. Data forwarding delay and the maximum data age are crucial for such applications.

Simulations are run with different sampling period duration τ (and corresponding interval duration I). The maximum data age and the other metrics are measured for each τ. All figures in this section, except Fig. 5.22 and Fig. 5.29, are created by plotting one of the other four performance metrics as a function of the maximum data age. Fig. 5.22 and Fig. 5.29 plot the average number of bytes that are transmitted per round as a function of τ. In all figures that plot the percentage of uncovered

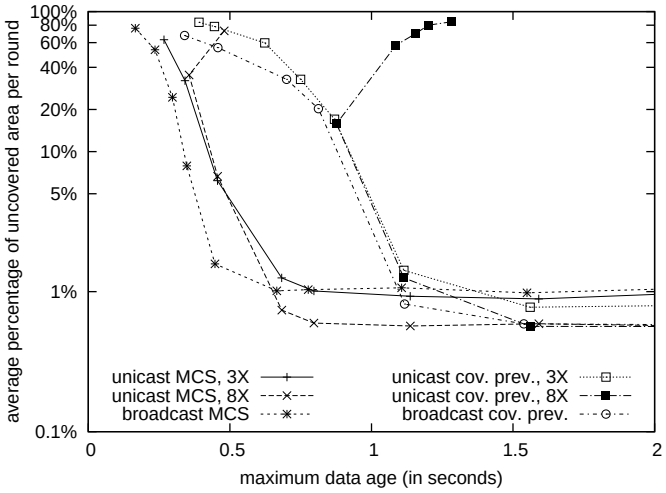
area as a function of the maximum data age, the point with the highest percentage of uncovered area corresponds to the highest sampling rate. The percentage of uncovered area decreases for lower sampling rates.

5.4.2.1 Performance for Independent Random Error Model

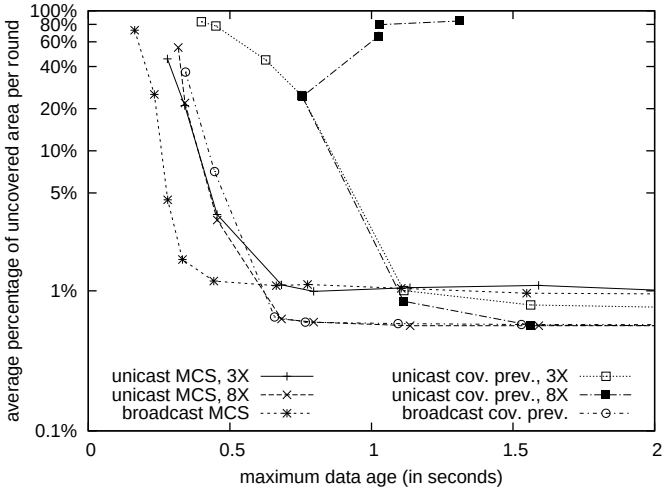
This section presents performance results with the independent random error model. Values for τ are chosen manually within the range of [0.09, 1.2] seconds. Note that there is some variation in the particular values chosen from this range for different protocols. For each protocol, the chosen values yield good coverage of the x / y region being plotted in the figures.

Figs. 5.16 plots performance results of the protocols with 20% physical layer loss rate. To better explain the results, Fig. 5.17 plots the average percentage of uncovered area per round and the maximum data age of the protocols as a function of τ, for increasing packet size. Both Fig. 5.16(a) and Fig. 5.17 plot performance results from the same set of simulations. For both fixed and increasing packet size, broadcast-based aggregation with minimal coverage set is able to achieve lower maximum data age for moderate loss of coverage. For around 2% uncovered area, broadcast-based aggregation with minimal coverage set is able to reduce the maximum data age by around 40% for fixed packet size. The performance improvement with broadcast-based aggregation with minimal coverage set decreases for increasing packet size. Note that broadcast-based aggregation with minimal coverage set is able to get at most 99% of the area covered for different maximum data age.

As τ gets smaller, the average percentage of uncovered area per round starts to increase for all protocols. The interval duration gets shorter as τ decreases. More packets are lost because of collisions. When the duration of the transmission interval is too short, nodes that share the same interval cannot get their packets through within their own transmission interval. Packets that arrive after the receiver has sent out its partial aggregate are not aggregated. For the coverage preserving protocol, as more packets are lost or arrive late, more nodes start to transmit because by the time their timers go off, they cannot hear or overhear from enough neighbours that



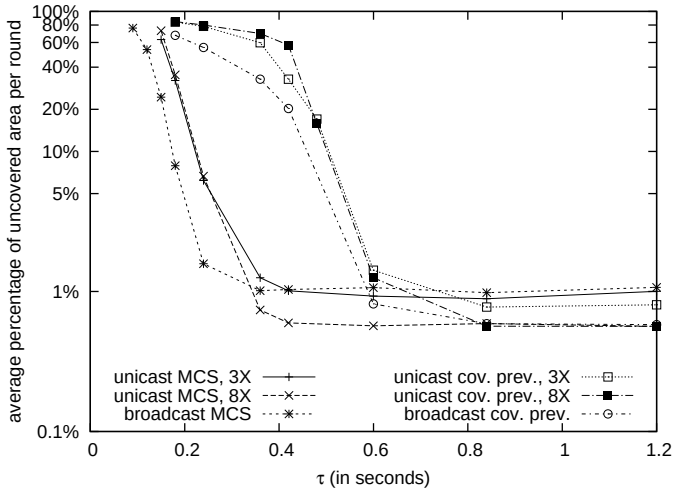
(a) Increasing Packet Size



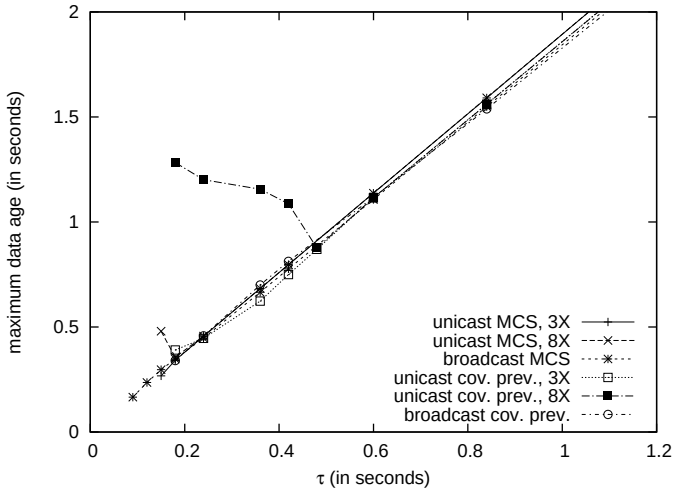
(b) Fixed Packet Size

Figure 5.16: Average Percentage of Uncovered Area per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 20%)

cover their sensing area completely, making the situation even worse.



(a) Average Percentage of Uncovered Area per Round



(b) Maximum Data Age

Figure 5.17: Performance as a Function of τ for Independent Random Error Model (real-time, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 20%, increasing packet size)

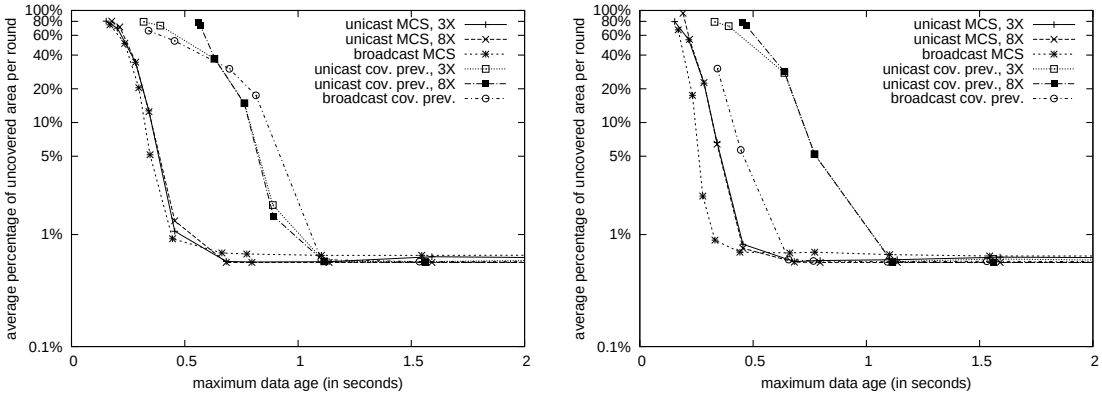
Network congestion starts to build up as the sampling rate gets higher. When up to 8 MAC layer retransmissions are allowed, some packets manage to get through to the sink after repeated retransmissions, with long transmission delay. As seen in Fig. 5.17, for the unicast-based protocols with up to 8 MAC layer retransmissions, the maximum data age starts to increase when the sampling rate is high enough.

Comparing the results in Figs. 5.16(a) to those in Fig. 5.16(b), an observation is the dramatic impact of packet size on the relative performance of the protocols.

For fixed packet size, broadcast-based coverage preserving aggregation outperforms unicast-based coverage preserving aggregation. Its performance is very close to that of unicast-based aggregation with minimal coverage set. Packet size has a significant impact on broadcast-based coverage preserving aggregation; the maximum data age of broadcast-based coverage preserving aggregation increases greatly in the case of increasing packet size. For 5% uncovered area, the maximum data age of broadcast-based coverage preserving aggregation doubles with increasing packet size. This reflects the fact that the same sampling value may be carried in the aggregates of multiple nodes. The performance of broadcast-based aggregation with minimal coverage set also gets worse for increasing packet size, although not as much as broadcast-based coverage preserving aggregation, because only 96 nodes are transmitting in broadcast-based aggregation with minimal coverage set. Compared with the broadcast-based protocols, the unicast-based protocols are influenced by packet size too, but the impact is insignificant, especially for unicast-based aggregation with minimal coverage set. This explains why the performance improvement with broadcast-based aggregation with minimal coverage set decreases in the case of increasing packet size.

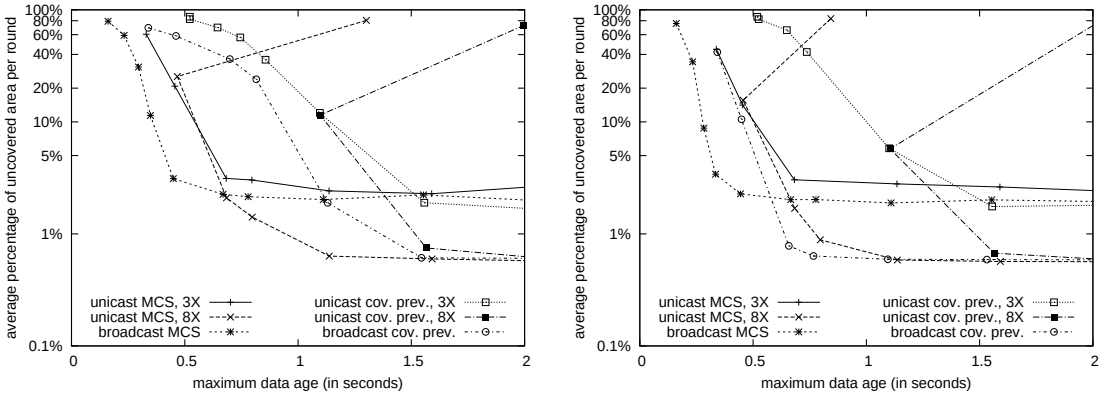
The performance of the protocols for 10% and 30% physical layer loss rate is shown in Fig. 5.18 and Fig. 5.19, respectively. The relative performance of broadcast-based coverage preserving aggregation improves as the physical layer loss rate get higher. For fixed packet size, broadcast-based coverage preserving aggregation is outperformed by both unicast and broadcast-based aggregation with minimum coverage set when the physical layer packet loss rate is only 10%. Packet loss can be easily recovered by MAC layer retransmissions for 10% physcial layer loss rate. Broadcast-based coverage preserving aggregation does not have much advantage in this case. With fixed packet size, broadcast-based coverage preserving aggregation outperforms all unicast-based protocols for 30% physical layer loss rate, as the cost for loss recovery gets higher in this case. The relative performance of broadcast-based coverage preserving aggregation gets much worse in the case of increasing packet size, owing to the fact that the same sampling value may be included in the aggregates

of multiple nodes.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.18: Average Percentage of Uncovered Area per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 10%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.19: Average Percentage of Uncovered Area per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 30%)

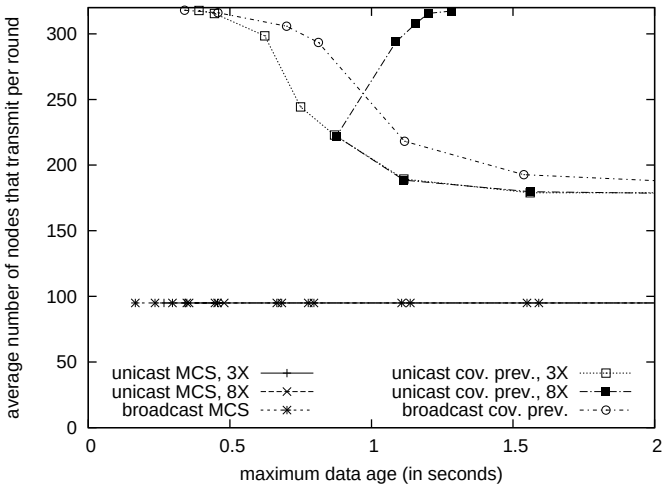
The relative performance of broadcast-based aggregation with minimal coverage set improves with higher physical layer loss rate. For increasing packet size, the performance of unicast and broadcast-based aggregation with minimum coverage set is very close when the physical layer loss rate is 10%. The cost for packet recovery is low for unicast transmissions in this case, and broadcast-based aggregation with minimal coverage set has to pay the price for carrying the same sampling value in multiple aggregates. When the physical layer loss rate is 30%, broadcast-based

aggregation with minimal coverage set outperforms unicast-based aggregation with minimal coverage set for both fixed and increasing packet size. As seen in Fig. 5.16, Fig. 5.18 and Fig. 5.19, unicast-based coverage preserving performs the worst for all three loss rates, except in Fig. 5.18(a).

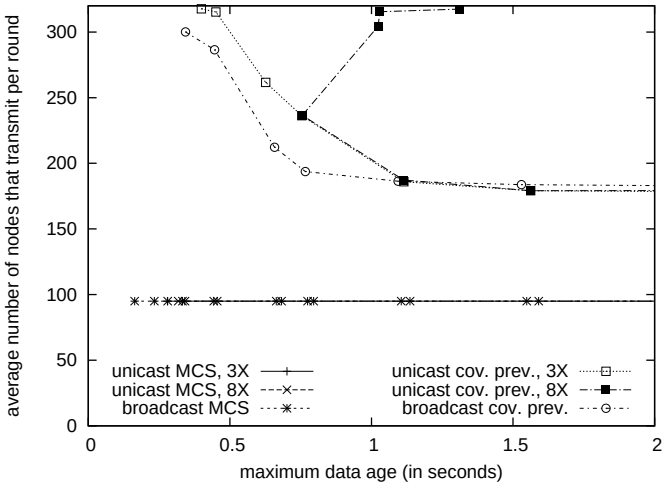
Fig.5.20 plots the average number of nodes that transmit per round for different sampling rates and 20% physical layer loss rate. As the sampling rate gets higher, more packets get dropped due to collisions. At the same time, more packets arrive after the receivers have transmitted for that round, because the transmission interval is so short that packets sent by the nodes in the same ring cannot get through during their transmission interval. For both unicast and broadcast-based coverage preserving aggregation, more nodes transmit because they cannot hear or overhear transmissions from enough nodes that cover their sensing area by the time their timers go off, until all 320 nodes transmit. The number of transmitting nodes is fixed at 96 for the protocols with the minimal coverage set.

Fig. 5.21 shows the average number of bytes transmitted per round as a function of the maximum data age, for 20% physical layer loss rate. The average number of bytes transmitted per round as a function of τ is plotted in Fig. 5.22 to better explain the results.

Traffic volume per round is limited by the sampling period duration τ and network capacity. When duration τ is long enough for all traffic to get though, the amount of traffic generated for each round decides the average number of bytes transmitted per round. As the sampling rate gets higher, more packet collisions happen as the nodes contend for the channel during their shortened transmission interval. For coverage preserving aggregation, more nodes start to transmit as the sampling rate increases, as seen in Fig. 5.20. If no packets are dropped due to queue overflow, more traffic is generated and transmitted for each round. (The duration τ may not be long enough to get all the traffic for one round through.) The network eventually reaches its maximum capacity as the sampling rate keeps getting higher, and traffic transmitted per round is decided by τ in this case. This explains why in Fig. 5.22(a), the number of bytes transmitted per round of the coverage preserving protocols increases first

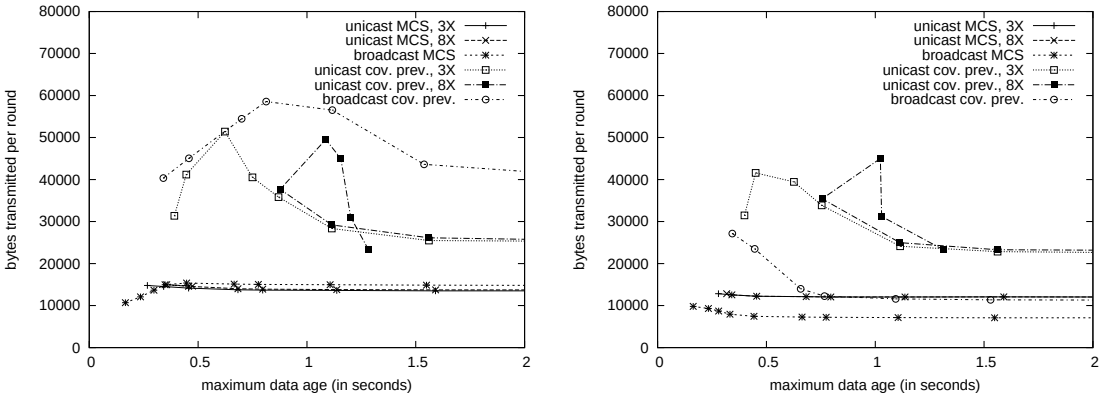


(a) Increasing Packet Size



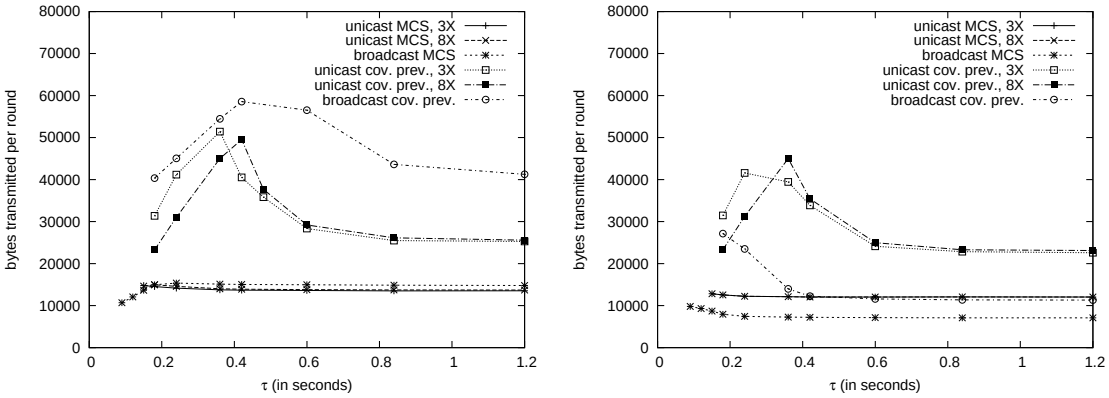
(b) Fixed Packet Size

Figure 5.20: Nodes that Transmit per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 20%)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.21: Bytes Transmitted per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 20%)



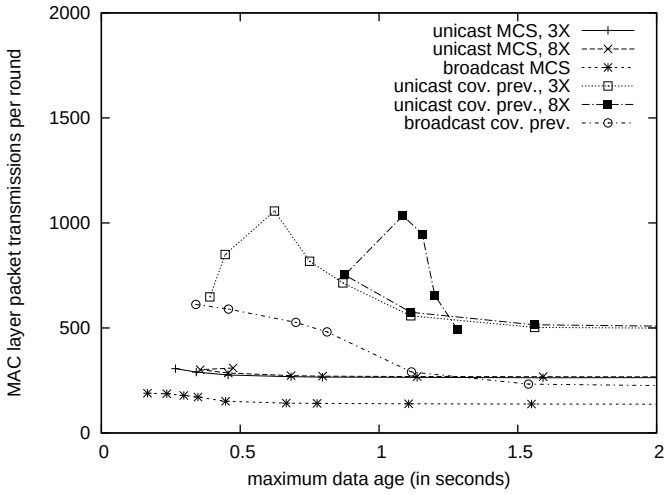
(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.22: Bytes Transmitted per Round as a Function of τ for Independent Random Error Model (real-time, N = 320, N_{MCS} = 96, S = 250m × 250m, PLR = 20%)

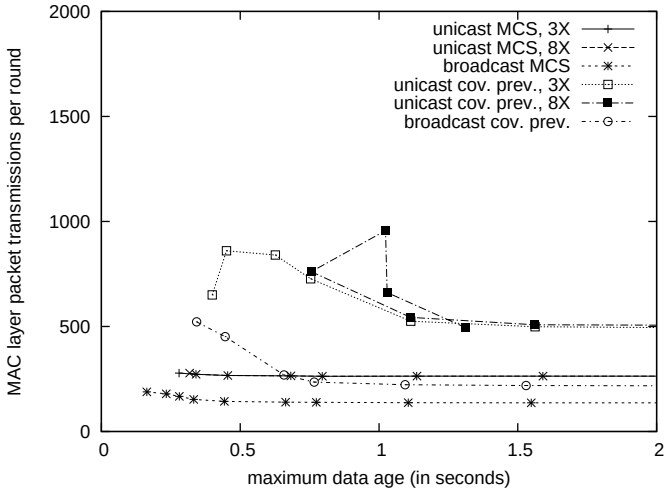
and then decreases as τ gets smaller. Similar phenomena are also observed in Fig. 5.22(b).

The average number of MAC layer packet transmissions per round is plotted in Fig. 5.23, for 20% physical layer loss rate. For increasing packet size, broadcast-based coverage preserving aggregation has the highest traffic volume, as seen in Fig. 5.21(a), although it transmits fewer packets than unicast-based coverage preserving aggregation, as seen in Fig. 5.23(a). This reflects the fact that the same value may be included in multiple packets in broadcast-based coverage preserving aggregation.

For fixed packet size, the curves for the average number of MAC layer packet transmissions per round, as seen in Fig. 5.23(b), are very similar with those for the average number of bytes transmitted per round, as seen in Fig. 5.21(b).



(a) Increasing Packet Size



(b) Fixed Packet Size

Figure 5.23: MAC Layer Packet Transmissions per Round for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m $\times$ 250m, PLR = 20%)

5.4.2.2 Performance for Two-state Gilbert Error Model

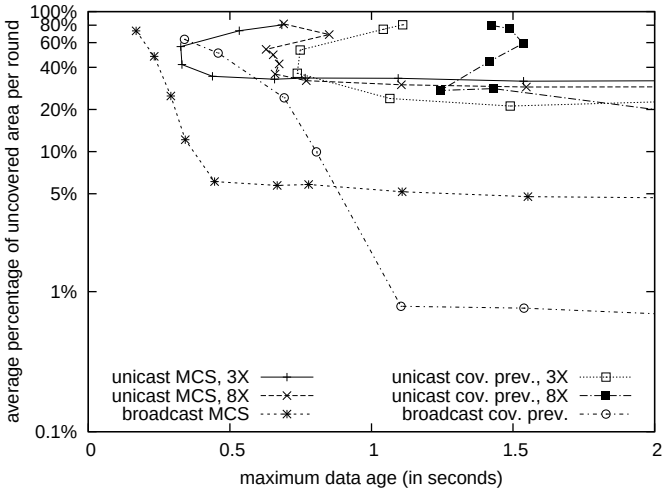
The section presents performance results with the two-state Gilbert error model. Different values are taken from the range of [0.09, 1.2] seconds for τ. Similar to the

simulations in Section 5.4.2.1, there is some variation in the particular values chosen from this range for different protocols.

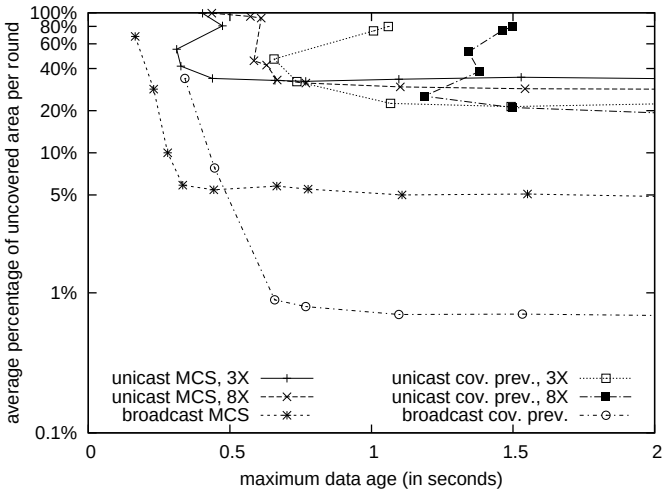
Fig. 5.24 plots performance results of the proposed coverage preserving protocols and the conventional aggregation protocols with minimum coverage set, for a two-state Gilbert error model with 20% of time in the bad state. As described in Section 5.4.1.2, there is a "good" state and a "bad" state in the error model. When the node is in the "good" state, all packets are received correctly unless they are corrupted by collisions. No packets are received when the node stays in the "bad" state. It is assumed that the sink always stays in the "good" state. The sojourn time in each state is exponentially distributed. The average sojourn time in the bad state is fixed at 0.5 seconds for all results that are reported in this section.

Both unicast-based coverage preserving aggregation and unicast-based aggregation with minimum coverage set yield poor coverage, as seen in Fig. 5.24, because MAC layer retransmission is not effective in packet recovery in this case. For real-time data collection that is willing to trade coverage for lower maximum data age, broadcast-based aggregation with minimum coverage set is able to achieve lower maximum data age with moderate loss of coverage, especially in the case of increasing packet size. For around 5% uncovered area, broadcast-based aggregation with minimum coverage set can lower the maximum data age by more than 40% in the case of increasing packet size, as seen in Fig. 5.24(a). Broadcast-based coverage preserving aggregation performs the second best in this case. It is outperformed by broadcast-based aggregation with minimum coverage set in terms of the maximum data age, but with increased data age, it is able to lower the percentage of uncovered area significantly.

Results for the two-state Gilbert error model with 10% and 1/3 of time in the bad state are shown in Fig. 5.25 and Fig. 5.26, respectively. For 10% time in the bad state, broadcast-based aggregation with minimum coverage set is the best choice if the system is willing to accept moderate loss of coverage (less than 20%, for example) for better data freshness. The performance improvement with broadcast-based aggregation with minimum coverage set is substential in the case of increasing packet



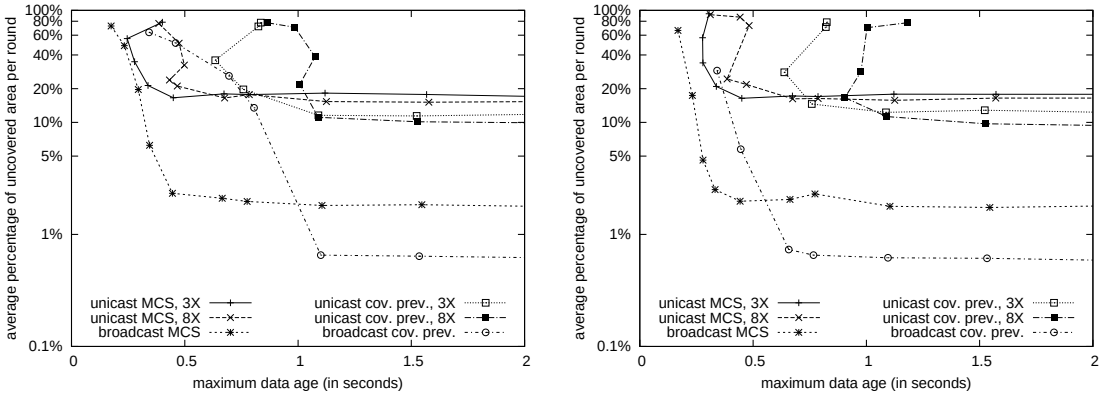
(a) Increasing Packet Size



(b) Fixed Packet Size

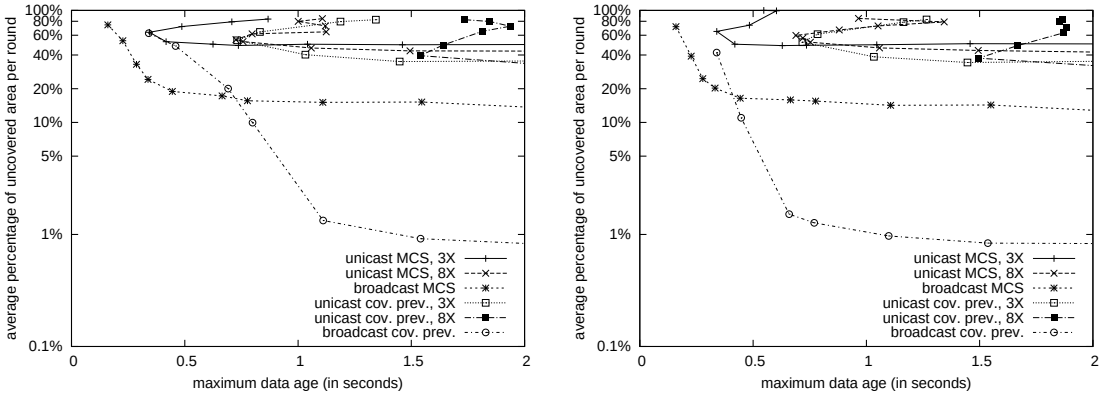
Figure 5.24: Average Percentage of Uncovered Area per Round for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)

size, as seen in Fig. 5.25(a). Broadcast-based coverage preserving aggregation outperforms the unicast-based protocols in the case of fixed packet size. For increasing packet size, with up to 3 MAC retransmissions, the unicast-based protocols yield substantiallly lower maximum data age than broadcast-based coverage preserving aggregation at around 20% loss of coverage, but broadcast-based coverage preserving aggregation outperforms the unicast-based protocols for less than 15% loss of coverage.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.25: Average Percentage of Uncovered Area per Round for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 10\%$, $\tau_b = 0.5$ sec.)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.26: Average Percentage of Uncovered Area per Round for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 1/3$, $\tau_b = 0.5$ sec.)

For 1/3 time in the bad state, if the system is willing to accept around 15% or even

higher percentage of uncovered area, broadcast-based aggregation with minimum coverage set can provide lower maximum data age. Only broadcast-based coverage preserving aggregation can achieve 90% or higher coverage in this case. Broadcast-based coverage preserving aggregation also outperforms the unicast-based protocols for both fixed and increasing packet size.

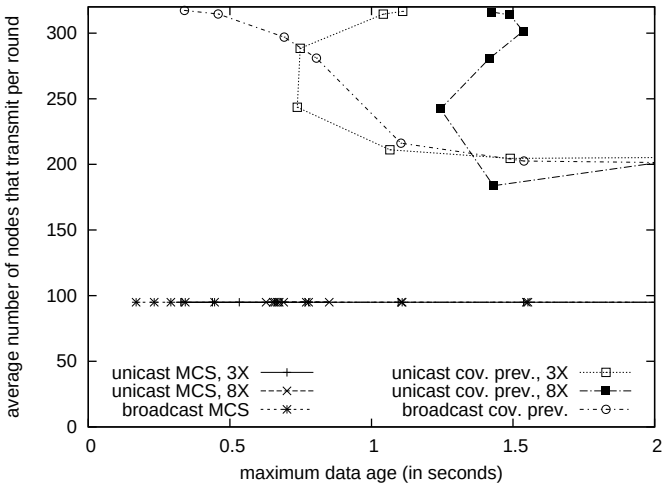
Fig. 5.27 to 5.30 plot performance results for the two-state Gilbert error model with 20% of time in the bad state. Fig. 5.27 plots the average number of nodes that transmit per round. For coverage preserving protocols, more nodes transmit as the sampling rate gets higher, until all nodes inside the network transmit.

Fig. 5.28 and Fig. 5.29 show the average number of bytes transmitted per round as a function of the maximum data age and τ, respectively. For both fixed and increasing packet size, broadcast-based aggregation with minimum coverage set transmit the fewest bytes. As the sampling rate gets higher, more traffic is generated inside the network. The network eventually reaches its maximum capacity, and the traffic transmitted per round is decided by τ in this case. This explains why for increasing packet size, the average number of bytes transmitted per round of all protocols starts to drop as the sampling rate gets higher. This also explains why for fixed packet size, the curves of the unicast-based protocols move downwards as τ decreases.

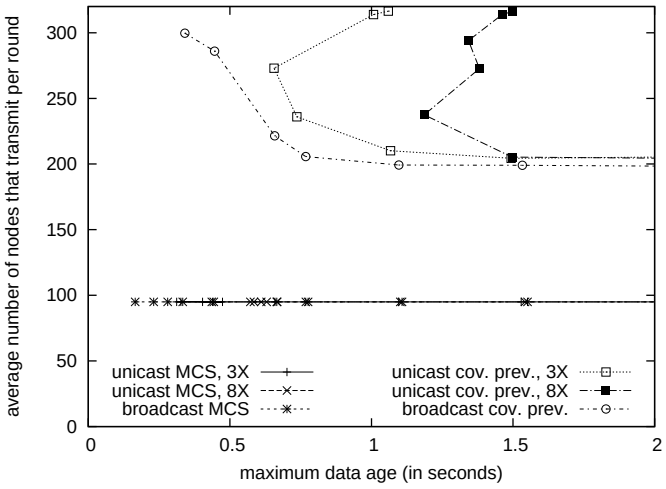
The average number of MAC layer packet transmissions per round is shown in Fig. 5.30. For increasing packet size, although broadcast-based coverage preserving transmits fewer packets than unicast-based coverage preserving with up to 3 retransmissions, as seen in Fig. 5.30(a), it transmits more bytes than the latter, as seen in Fig. 5.28(a), because the same sensor value may be redundantly included in multiple aggregates. For fixed packet size, the curves in Fig. 5.30(b) is quite similar with those in Fig. 5.28(b).

5.4.2.3 Impact of Network Density

The two sensor networks used in Section 5.4.1.3 are used here to evaluate the impact of density in real-time scenarios. In one sensor field, 320 nodes are randomly scattered

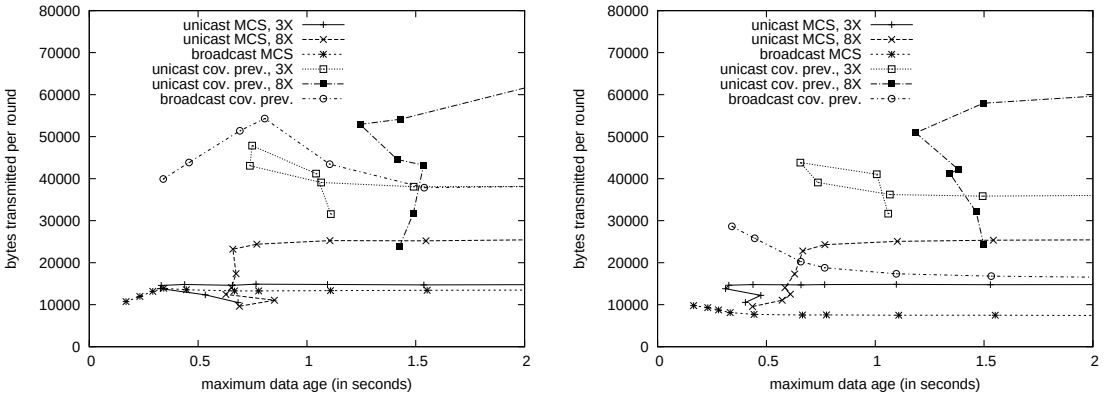


(a) Increasing Packet Size



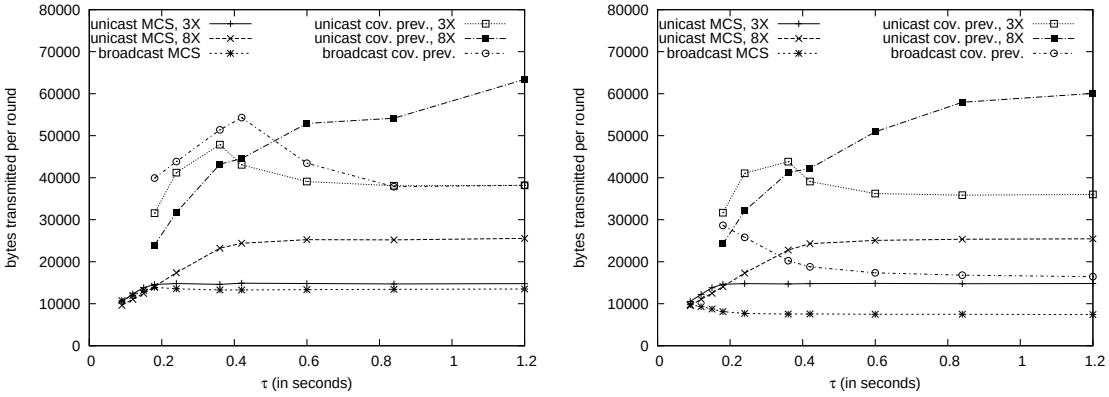
(b) Fixed Packet Size

Figure 5.27: Nodes that Transmit per Round for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)



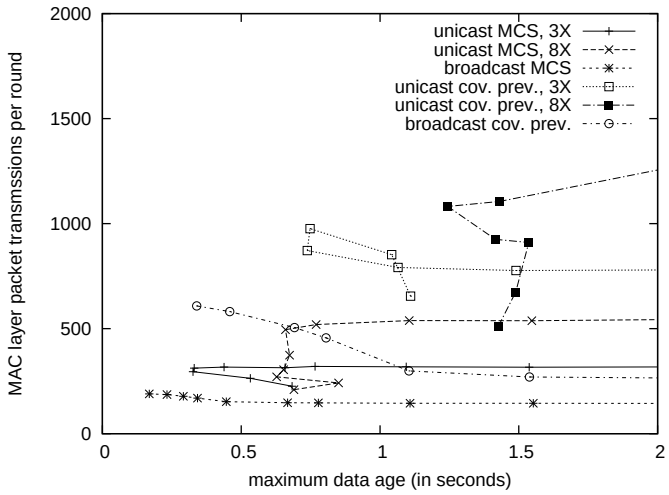
(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.28: Bytes Transmitted per Round for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)

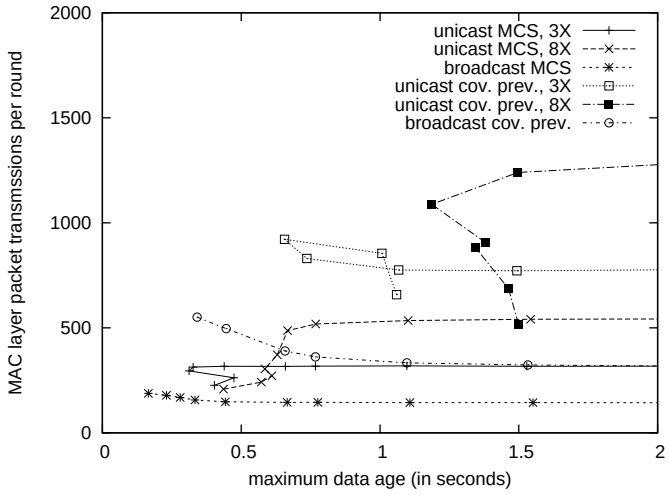


(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.29: Bytes Transmitted per Round as a Function of τ for Gilbert Error Model (real-time, N = 320, $N_{MCS} = 96$, S = 250m $\times$ 250m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)



(a) Increasing Packet Size



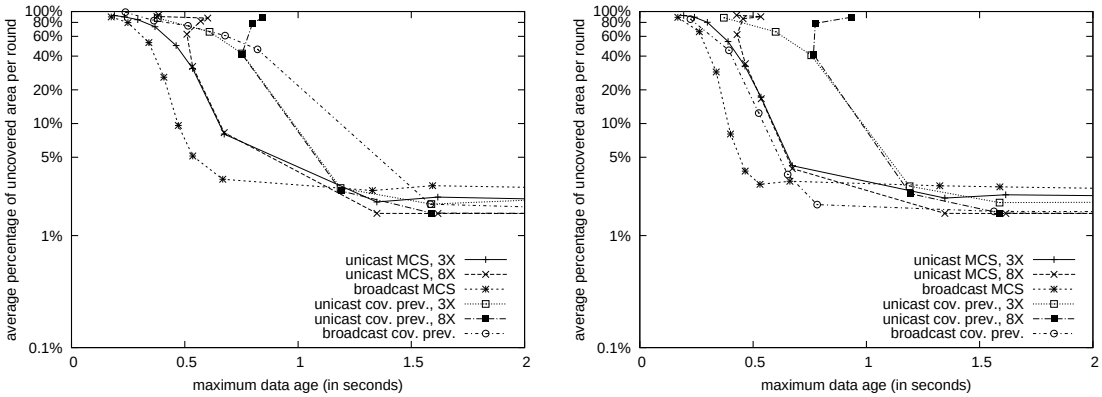
(b) Fixed Packet Size

Figure 5.30: MAC Layer Packet Transmissions per Round for Gilbert Error Model (real-time, varying τ, N = 320, N_{MCS} = 96, S = 250m × 250m, P_b = 20%, τ_b = 0.5 sec.)

over a 300 meter by 300 meter area. In the other, 320 nodes are randomly scattered over a 200 meter by 200 meter area. Values for τ are chosen manually within the range of [0.06, 1.4] seconds. Similarly, there is some variation in the particular values chosen from this range for different protocols.

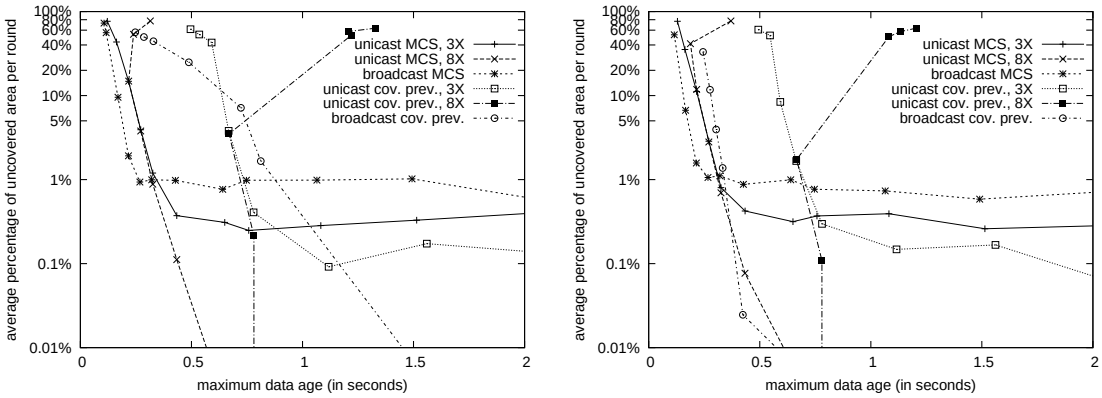
Figs. 5.31 and 5.32 plot performance results for sensor networks with 300 meter by 300 meter area and 200 meter by 200 meter area, respectively. The independent random error model is used in the simulations, with 20% physical layer loss rate. As in Section 5.4.1.3, the y-scale of Fig. 5.32 is changed to plot the results with less than 0.1% uncovered area. For real-time data collection systems that are willing trade coverage for data freshness, broadcast-based aggregation with minimum coverage set is able to achieves lower minimum data age with moderate loss of coverage.

Figs. 5.33 and 5.34 plot performance results with the two-state Gilbert model for the sensor networks with 300 meter by 300 meter area and 200 meter by 200 meter area, respectively. The average sojourn time in the bad state is fixed at 0.5 seconds. In Fig. 5.34, the y-scale is changed to plot results with percentage of uncovered area below 0.1%. All unicast-based aggregation protocols yield poor coverage, especially with lower density, as MAC layer retransmissions are not effective in packet recovery in this case. Broadcast-based aggregation with minimum coverage set and broadcast-based coverage preserving aggregation perform better than the unicast-based protocols. With lower density, broadcast-based aggregation with minimum coverage set can achieve 90% coverage in the best case. If the system is willing to accept 10% or even higher loss of coverage, broadcast-based aggregation with minimum coverage set is able to provide lower maximum data age, especially in the case of increasing packet size. With higher density, broadcast-based aggregation with minimum coverage set is able to provide around 95% coverage in the best case. For systems that are willing to accept 5% or more uncovered area, broadcast-based aggregation with minimum coverage set can provide lower maximum data age. For real-time systems that demand high area coverage, broadcast-based coverage preserving aggregation can reduce the loss of coverage significantly, for example, from 5% to less than 0.1% in the case of higher density, at the cost of increased data age.



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.31: Impact of Lower Density for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 139, S = 300m × 300m, P_b = 20%, τ_b = 0.5 sec.)

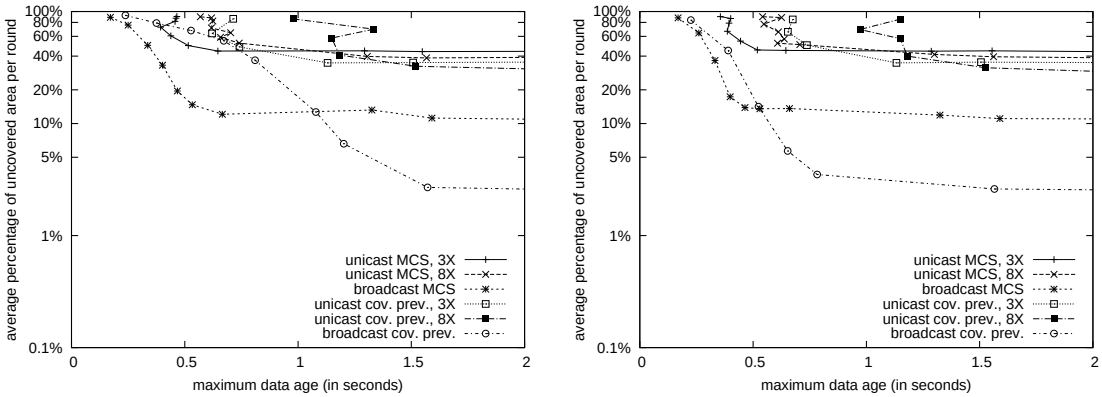


(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.32: Impact of Higher Density for Independent Random Error Model (real-time, varying τ, N = 320, N_{MCS} = 54, S = 200m × 200m, P_b = 20%, τ_b = 0.5 sec.)

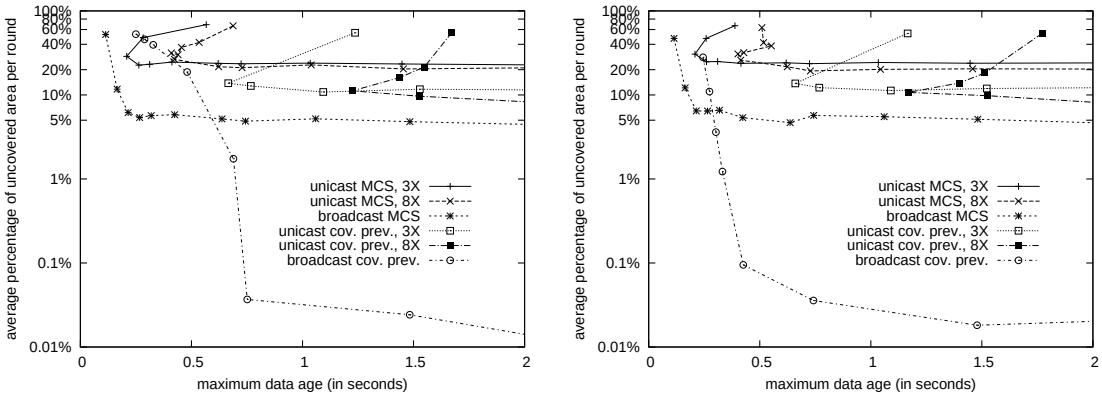
5.4.2.4 Summary of Results

In Section 5.4.2, the performance of the protocols is evaluated in the context of real-time data collection. Section 5.4.2.1 and Section 5.4.2.2 present performance results with the independent random error model and the two-state Gilbert error model, respectively. Performance results for networks with higher and lower density are presented in Section 5.4.2.3. For real-time data collection applications that are willing to trade moderate coverage loss for better data freshness, broadcast-based aggregation with minimum coverage set is found to yield lower maximum data age



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.33: Impact of Lower Density for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 139$, S = 300m × 300m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure 5.34: Impact of Higher Density for Gilbert Error Model (real-time, varying τ, N = 320, $N_{MCS} = 54$, S = 200m × 200m, $P_b = 20\%$, $\tau_b = 0.5$ sec.)

than the other protocols in most scenarios. For real-time systems that demand low loss of area coverage, broadcast-based coverage preserving aggregation can lower the percentage of uncovered area significantly, especially when packet loss follows the Gilbert error model.

5.5 Discussion

Energy consumption varies with different hardware platforms. Table 5.2 and table 5.3, respectively, list the current draw of MICAz and MICA2 from the Crossbow

company. Due to the lack of a general energy model, energy consumption of the aggregation protocols is not measured in this work.

While the energy consumption in different states varies from one device to another, the transmit mode usually has the highest energy consumption, followed by the receive mode[3]. Nodes expend substantially less energy when in a sleep mode. By putting nodes to sleep, node scheduling protocols are able to reduce energy consumption significantly.

19.7mA	Receive mode
11 mA	TX, -10 dBm
14 mA	TX, -5 dBm
17.4 mA	TX, 0 dBm
20 μA	Idle mode, voltage regulator on
1 μA	Sleep mode, voltage regulator off

Table 5.2: Current Draw for MICAz

27mA	Transmit with maximum power
10 mA	Receive
1 μA	Sleep mode

Table 5.3: Current Draw for MICA2 with MPR400CB Processor and Radio Platform

Compared with aggregation based on node scheduling, coverage preserving data aggregation has higher energy consumption. Quantitative analysis of the energy consumption of the protocols is left as a future work. Furthermore, there is still room left for both protocols to improve on energy consumption, especially in delay-tolerant scenarios with large τ. The nodes in ring i only need to stay awake during the transmission interval of the nodes in ring $i+1$ (if $i < H$) and their own transmission interval. This can be considered in the future work.

[3]Micaz has higher current draw in the receive mode than in the transmit mode for -10, -5, and 0 dBm.

5.6 Summary

In this chapter, new unicast and broadcast-based aggregation protocols for periodic data collection in dense sensor networks are proposed. The goal of these protocols is to maintain complete area coverage at a low cost with respect to the required sensor node transmissions. The new protocols integrate aggregation together with dynamic selection of which nodes should transmit during each data collection round. The performance of the proposed protocols is compared to that of conventional aggregation protocols relying on node scheduling in the context of both delay-tolerant and real-time data collection.

For delay-tolerant data collection, the following conclusions can be drawn.

- When packet loss follows the two-state Gilbert error model, the broadcast-based coverage preserving protocol is able to achieve substantially improved reliability (in comparison to node scheduling approaches in which only a minimal coverage set of nodes is active), at substantially lower cost than if all nodes transmit during each round.

- When packet loss is random, the broadcast-based coverage preserving protocol is able to achieve substantially improved reliability in some scenarios. With 320 nodes in a 250 meter by 250 meter area, broadcast-based coverage preserving aggregation yields better coverage when the physical layer loss rate is above 40%. The relative performance of broadcast-based coverage preserving aggregation improves with higher network density.

- The unicast-based coverage preserving aggregation protocol does not show much advantage over aggregation relying on node scheduling.

For real-time data collection, the relative performance of the protocols can be summarized as follows.

- If the application is willing to trade moderate coverage loss for better data freshness, broadcast-based aggregation with minimum coverage set can provide

lower maximum data age than the other protocols in most scenarios considered in this chapter.

- With the two-state Gilbert error model, the broadcast-based coverage preserving protocol can achieve substantially improved reliability than broadcast-based aggregation with minimum coverage set, at the cost of longer maximum data age and higher traffic load. The broadcast-based coverage preserving protocol does not show much advantage when packet loss is random.

- The unicast-based coverage preserving aggregation protocol has no performance advantage over aggregation relying on node scheduling.

Chapter 6

Conclusions

This dissertation addresses protocol design and performance issues in data aggregation for periodic data collection in wireless sensor networks. New aggregation protocols are proposed and compared with existing protocols.

A variety of data aggregation protocols have been proposed for wireless sensor networks [27, 37, 61]. An important issue in data aggregation is timing control [86]. For real-time monitoring sensor networks, the amount of time each node spends waiting for data to be aggregated has to be minimized while ensuring effective data aggregation to save energy. Different timing strategies have been devised to promote aggregation [26, 86]. Existing aggregation protocols can be classified as synchronous or asynchronous based on their timing control strategies. New asynchronous aggregation protocols are proposed in this dissertation.

Aggregation protocols can also be classified as unicast-based [61, 86] or broadcast-based [32, 66, 67] according to whether they use unicast or broadcast communication. Unicast transmission relies on ARQ for reliability, and is not robust against node and persistent link failures. Broadcast transmission takes advantage of the inherent redundancy of the broadcast medium to improve reliability. New broadcast-based aggregation protocols are proposed in this dissertation.

Sensor networks are often densely deployed to improve network robustness. To prolong network lifetime, node scheduling protocols have been proposed to save energy by putting redundant nodes to sleep [54, 93, 99]. When packet loss and node failures are prevalent, turning redundant nodes off saves energy at the cost of network reliability. Without node redundancy, a single node failure may cause incomplete coverage if it is not replaced by other nodes right away. In this dissertation, coverage

preserving data aggregation protocols are proposed. In the proposed protocols, each node dynamically decides whether it should participate in data collection for each round. A node cancels its transmission if the data that has already been transmitted by neighbouring nodes is enough to provide the desired coverage.

Performance comparisons between synchronous and asynchronous aggregation, unicast and broadcast-based aggregation, and coverage preserving aggregation and aggregation based on node scheduling are presented in this work. The remainder of this chapter is organized as follows. Section 6.1 summarizes each of the three parts of the thesis. Section 6.2 states the main contributions, and Section 6.3 outlines future research directions.

6.1 Thesis Summary

This dissertation investigates three fundamental issues concerning the performance of data aggregation for periodic data collection in sensor networks: which nodes should transmit, when should the nodes transmit, and should they transmit their packets using unicast or broadcast.

To promote aggregation, intermediate nodes may have to delay their own transmissions to wait for data from other nodes. Synchronous or asynchronous timing control plays a key role in aggregation efficiency and data forwarding delay. Synchronous aggregation works by assigning different transmission intervals to nodes based on their distances to the sink. Synchronous schemes are efficient in enabling data aggregation, however, they may cause increased delay and suboptimal use of spatial multiplexing. With asynchronous aggregation, nodes closer to the sink may transmit earlier than nodes farther away from the sink.

Both synchronous and asynchronous aggregation protocols have been previously proposed. In particular, Solis *et al.* have *advocated* synchronous aggregation [86]. This dissertation proposes improved asynchronous aggregation protocols with more aggressive methods for determining when a node should transmit to its parent. Three asynchronous aggregation protocols are proposed: basic asynchronous aggregation,

asynchronous aggregation with EWMA, and adaptive asynchronous aggregation. In all three protocols, a timeout value gives the maximum time duration that an interior node would wait before aggregating the received data and sending out its data for each round. The timeout values are adaptively adjusted to minimize data forwarding delay. Basic asynchronous aggregation adapts timeout values based on packet reception in the immediately previous round. Asynchronous aggregation with EWMA adapts timeout values based on a weighted average of history information from multiple rounds. In asynchronous aggregation, the transmission times of aggregation tree leaves are spread over a randomization interval to avoid congestion at the beginning of each round. In adaptive asynchronous aggregation, the duration of the randomization interval is adaptively adjusted. Simulation results show that asynchronous aggregation outperforms synchronous aggregation by providing lower maximum data age for a given end-to-end loss rate.

Data collection using data aggregation can use unicast or broadcast communication. Previous broadcast-based aggregation protocols have focused on the design of aggregation schemes for duplicate-sensitive aggregation functions [32, 66]. The second part of this work examines in particular broadcast-based protocols that minimize the number of packet transmissions and rely on multipath delivery rather than ARQ for reliability. It is assumed acceptable for the sink (and intermediate nodes) to receive multiple aggregates including the same sensor value. In addition to the reliability improvement brought by multipath routing, improved reliability is achieved by allowing each node to broadcast twice using a two-phase strategy. Each node decides whether a second broadcast is necessary based on the broadcasts it heard from its neighboring nodes. The node cancels its second transmission if its data has been forwarded towards the sink by some other node. Both synchronous and asynchronous broadcast-based aggregation protocols are proposed. The new broadcast-based protocols are compared to synchronous and asynchronous unicast-based aggregation protocols in terms of maximum data age, end-to-end loss rate, and traffic volume.

Sensor networks are often deployed more densely than would minimally be required. In such cases, node scheduling protocols can be used to determine which

nodes stay active, and which nodes go to sleep so as to conserve energy and prolong network lifetime [100, 104]. A drawback of node scheduling approaches, however, is that there may be a delay between node or communication failure and the subsequent wake-up of replacement node(s). During such a delay, monitoring coverage of some sub-region may be lost.

In Chapter 5, an alternative approach is proposed for use in contexts in which the objective is to periodically collect sensing data from nodes that completely cover a region of interest. In the proposed approach, each node dynamically determines during each round of data collection whether it should transmit its data, or whether its sensing area is covered by neighbouring nodes that have already transmitted. Both unicast and broadcast-based coverage preserving aggregation protocols are proposed. The key design issue is how to dynamically determine the transmission orderings in a coverage-aware manner so that nodes that are crucial for coverage transmit earlier.

The new coverage preserving protocols adopt the synchronous timing control strategy. Each transmission interval is divided into two phases. A two-phase transmission scheme is devised to achieve coverage-aware transmission orderings in the proposed protocols. Nodes crucial for coverage are scheduled to transmit in the first phase. In the unicast-based protocol, the nodes form a tree structure, with the sink as the root. A leaf node cancels its transmission for a round if its sensing area is covered by neighbouring nodes that have already transmitted for that round. The node relies on overhearing to decide which nodes have transmitted. All interior nodes are scheduled to transmit during the first phase of each interval. A leaf node is scheduled to transmit during the second phase of the next round if it cancelled its transmission for the current round. The idea is to schedule nodes that have good chances to cancel their transmissions to the second phase. In the unicast-based coverage preserving protocol, the failure of one interior tree node results in the loss of all of the data from the corresponding subtree. The broadcast-based coverage preserving protocol eliminates the static tree structure and adopts a ring topology instead. Any node whose sensing area is covered by neighbouring nodes that have already transmitted can cancel its broadcast for that round. The node is then scheduled to the second

phase for the next round. The new coverage preserving aggregation protocols are compared using simulation to data collection protocols relying on node scheduling.

6.2 Thesis Contributions

The main contributions of this work are as follows:

- Improved asynchronous aggregation protocols are proposed. In the proposed protocols, the nodes adaptively adjust their transmission times based on packet reception history to achieve efficient data aggregation while minimizing data forwarding delay [26].

- The new asynchronous aggregation protocols are compared with synchronous aggregation in the context of real-time monitoring. Simulation-based results show that asynchronous aggregation outperforms synchronous aggregation by achieving lower maximum data age for a given end-to-end loss rate. In most of the scenarios in the performance evaluation, the asynchronous aggregation protocols are able to lower the maximum data age by 20% to 30% for a given end-to-end loss rate. In the scenarios with 0.25 second period duration and the sink at the center, the asynchronous aggregation protocols are able to lower the maximum data age by 50% for 4% end-to-end loss rate.

- Performance comparison of the new asynchronous aggregation protocols is presented. For a given maximum data age, asynchronous aggregation with EWMA is able to achieve lower end-to-end loss rate than basic asynchronous aggregation, and the performance improvement is significant in some cases. In the scenario with the sink at the corner and $\tau = 0.75$ seconds, asynchronous aggregation with EWMA is able to reduce the end-to-end loss by 40% to 70% for a given maximum data age. Adaptive asynchronous aggregation is able to adjust the duration of the randomization interval to a good value so that the maximum data age is close to the mimimum value and the end-to-end loss rate is kept low.

- New synchronous and asynchronous broadcast-based aggregation protocols that minimize the number of packet transmissions and rely on multipath delivery for reliability are proposed. In addition to the reliability improvement that arises from potential multipath delivery to the sink, improved reliability is achieved by allowing each node to broadcast twice for one round, if the node does not hear its own data forwarded towards the sink by some other node [27].

- The new broadcast-based aggregation protocols are extensively evaluated and compared with unicast-based aggregation protocols. Simulation-based results suggest that when packet loss is random, the new broadcast-based protocols yield significantly improved performance in some real-time data collection scenarios, specifically when sensor data can be aggregated into packets of size that is independent (or largely independent) of the number of values being aggregated. In the scenario with 160 nodes over a 250 meter by 250 meter area and 20% physical layer loss rate, synchronous broadcast-based aggregation with fixed packet size is able to lower the maximum data age by about 40% for 2% end-to-end loss rate. The relative performance of synchronous broadcast-based aggregation gets even better with higher density and higher physical layer loss rate in the case of fixed packet size. When packet loss follows the two-state Gilbert error model, broadcast-based aggregation yields significantly better performance than unicast-based aggregation for both real-time and delay-tolerant data collection. Preliminary experimental results from a real sensor network of MICAz motes are presented in the thesis. The experimental results are consistent with the simulation results.

- Coverage preserving data aggregation protocols that integrate aggregation and coverage preserving are developed for efficiently maintaining area coverage in dense sensor networks. For each round of data collection, each node dynamically determines whether it should transmit its data, or whether the data forwarded by neighbouring nodes is sufficient to provide coverage. A two-phase transmission strategy is developed to dynamically determine node transmission

orderings so that nodes crucial to achieving area coverage transmit earlier to give other nodes a better chance to remain silent.

- The proposed coverage preserving data aggregation protocols are extensively evaluated and compared with data aggregation relying on node scheduling. The results show that broadcast-based coverage preserving aggregation can greatly improve reliability, at the potential cost of increased traffic volume. For example, with the two-state Gilbert error model, broadcast-based coverage preserving aggregation is able to yield an order of magnitude reduction in the percentage of uncovered area when the nodes spend 1/3 of their time in the bad state. For real-time data collection that is willing to accept moderate loss of coverage for better data freshness, broadcast-based data aggregation with node scheduling is found to yield lower maximum data age in most scenarios.

6.3 Future Work

There are many open issues related to the research in this thesis. Some possible future research directions are listed below.

- This work does not assume any particular aggregation function. Performance of the aggregation protocols for some particular type of aggregation, for example, duplicate-sensitive or duplicate-insensitive aggregation, may be investigated.

- Hybrid protocols using both unicast and broadcast transmission may be developed. When nodes are randomly deployed in an area, there may be sparse areas inside the network. Nodes in a sparse area may only have a couple of neighbours that are closer to the sink; these nodes can use unicast transmission with acknowledgement/retransmission to improve packet delivery reliability. Hybrid protocols of synchronous and asynchronous approaches may also be investigated. In Chapter 5, the nodes decide whether to transmit each round. Another possible approach is to let each interior node decide whether it has received enough data, or whether it needs to wait longer for more data. Protocols

implementing this approach may be developed.

- The coverage preserving protocols use observations of transmissions from the previous round, to determine whether a node's transmission will likely be needed in the current round. If so, the transmission is scheduled early so that other nodes may hear it and possibly realize that their own transmissions are not needed. In general, this strategy results in a larger set of nodes transmitting in each round than is minimally necessary. An area of future work concerns hybrid strategies that use dynamic node selection, as in the protocols proposed here, but that use additional topology information to assist in deciding which nodes may be most important to achieving area coverage.

- Independent failure probabilities among nearby nodes and node scheduling with 1-coverage are assumed in the comparison between coverage preserving data aggregation and aggregation based on node scheduling. Performance of data collection based on node scheduling with k-coverage ($k \geq 2$) and correlations between failure events at nearby nodes may be investigated.

- Chapter 4 reports results from some preliminary experiments using implementations of the synchronous unicast and broadcast-based protocols on Crossbow MICAz motes. The asynchronous aggregation protocols and the coverage preserving aggregation protocols may be implemented on MICAz motes (or other sensor devices) to provide more experimental performance results.

- Traffic volume is used in this work to yield insight into energy usage. A specific energy model may be used to investigate energy consumption.

References

[1] Tarek Abdelzaher, Tian He, and John Stankovic. Feedback control of data aggregation in sensor networks. In *Proc. 43rd IEEE Conf. on Decision and Control*, pages 1490–1495, Atlantis, Paradise Island, Bahamas, Dec. 2004.

[2] Ian F. Akyildiz, Welljan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, 2002.

[3] Javed Aslam, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus. Tracking a moving object with a binary sensor network. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 150–161, Los Angeles, CA, 2003.

[4] Amol Bakshi and Viktor K. Prasanna. Structured communication in single hop sensor networks. In *Proc. 1st European Workshop on Wireless Sensor Networks (EWSN '04)*, pages 138–153, Berlin, Germany, Jan. 2004.

[5] Elizabeth A. Basha, Sai Ravela, and Daniela Rus. Model-based monitoring for early warning flood detection. In *Proc. 6th ACM Conf. on Embedded Network Sensor Systems(SenSys '08)*, pages 295–308, Raleigh, NC, Nov. 2008.

[6] Monique Becker, Andre-Luc Beylot, Riadh Dhaou, Ashish Gupta, Rahim Kacimi, and Michel Marot. Experimental study: Link quality and deployment issues in wireless sensor networks. In *Proc. 8th Int'l IFIP-TC 6 Networking Conf. (Networking '09)*, pages 14–25, Aachen, Germany, May 2009.

[7] Roberto Beraldi, Roberto Baldoni, and Ravi Prakash. Lukewarm potato forwarding: A biased random walk routing protocol for wireless sensor networks. In *Proc. 6th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 09)*, Rome, Italy, June 2009.

[8] Maurizio Bocca, Emre I. Cosar, Juho Salminen, and Lasse M. Eriksson. A reconfigurable wireless sensor network for structural health monitoring. In *Proc. 4th Int'l Conf. on Structural Health Monitoring of Intelligent Infrastructure (SHMII-4)*, Zurich, Switzerland, July 2009.

[9] Tatiana Bokareva, Wen Hu, Salil Kanhere, Branko Ristic, Travis Bessell, Mark Rutten, and Sanjay Jha. Wireless sensor networks for battlefield surveillance. In *Proc. the Land Warfare Conf. (LWC '06)*, Brisbane, Australia, Oct. 2006.

[10] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.

[11] Jean Carle, Antoine Gallais, and David Simplot-Ryl. Preserving area coverage in wireless sensor networks by using surface coverage relay dominating sets. In *Proc.10th IEEE Symposium on Computers and Communications (ISCC '05)*, pages 347–352, Washington, DC, June 2005.

[12] Alberto Cerpa, Jeremy Elson, Michael Hamilton, Jerry Zhao, Deborah Estrin, and Lewis Girod. Habitat monitoring: application driver for wireless communications technology. In *Proc. Workshop on Data Communication in Latin America and the Caribbean (SIGCOMM LA '01)*, pages 20–41, San Jose, Costa Rica, Apr. 2001.

[13] Alberto Cerpa, Jennifer L. Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. Statistical model of lossy links in wireless sensor networks. In *Proc. 4th Int'l Symposium on Information Processing in Sensor Networks (IPSN '05)*, pages 81–88, Los Angeles, CA, Apr. 2005.

[14] Yuanzhu Peter Chen, Arthur L. Liestman, and Jiangchuan Liu. A hierarchical energy-efficient framework for data aggregation in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 55(3):789–796, 2006.

[15] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. Approximate aggregation techniques for sensor databases. In *Proc. 20th Int'l Conf. on Data Engineering (ICDE '04)*, pages 449–460, Boston, MA, Mar.–Apr. 2004.

[16] Santpal Singh Dhillon and Krishnendu Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *Proc. IEEE Wireless Communications and Networking Conf. (WCNC 2003)*, pages 1609–1614, New Orleans, LA, Mar. 2003.

[17] Jean-pierre Ebert and Andreas Willig. A gilbert-elliot bit error model and the efficient use in packet level simulation. Technical Report TKN-99-002, Telecommunication Networks Group, Technical University Berlin, Mar. 1999.

[18] Cheng Tien Ee and Ruzena Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 148–161, Baltimore, MD, Nov. 2004.

[19] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36(SI):147–163, Dec. 2002.

[20] Sinem Coleri Ergen. Zigbee/IEEE 802.15.4 summary. www.sinemergen.com/zigbee.pdf.

[21] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proc. 5th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom '99)*, pages 263–270, Seattle, WA, Aug. 1999.

[22] Kai-Wei Fan, Sha Liu, and Prasun Sinha. On the potential of structure-free data aggregation in sensor networks. In *Proc. 25th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM '06)*, pages 1–12, Barcelona, Spain, Apr. 2006.

[23] Kai-Wei Fan, Sha Liu, and Prasun Sinha. Scalable data aggregation for dynamic events in sensor networks. In *Proc. 4th ACM Conf. on Embedded Networked Sensor Systems (SenSys '06)*, pages 181–194, Boulder, CO, Nov. 2006.

[24] Elena Fasoloy, Michele Rossi, and Jorg Widmeand Michele Zorzi. In-network aggregation techniques for wireless sensor networks: a survey. *IEEE Wireless Communications*, 14(2):70–87, 2007.

[25] Emad Felemban, Chang-Gun Lee, Eylem Ekici, Ryan Boder, and Serdar Vural. Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks. In *Proc. 24th Int'l Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM '05)*, pages 2646–2657, Miami, FL, Mar. 2005.

[26] Jie Feng, Derek Eager, and Dwight Makaroff. Asynchronous data aggregation for real-time monitoring in sensor networks. In *Proc. 6th Int'l IFIP-TC 6 Networking Conf. (Networking '07)*, pages 73–84, Atlanta, GA, May 2007.

[27] Jie Feng, Derek Eager, and Dwight Makaroff. Aggregation protocols for high rate, low delay data collection in sensor networks. In *Proc. 8th Int'l IFIP-TC 6 Networking Conf. (Networking '09)*, pages 26–39, Aachen, Germany, May 2009.

[28] James Fogarty, Carolyn Au, and Scott E. Hudson. Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. In *Proc.19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*, pages 91–100, Montreux, Switzerland, Oct. 2006.

[29] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 138–149, Los Angeles, CA, Nov. 2003.

[30] Jie Gao, Leonidas Guibas, Nikola Milosavljevic, and John Hershberger. Sparse data aggregation in sensor networks. In *Proc. 6th Int'l Conf. on Information Processing in SensorNetworks (IPSN '07)*, pages 430–439, Cambridge, MA, Apr. 2007.

[31] Yong Gao, Kui Wu, and Fulu Li. Analysis on the redundancy of wireless sensor networks. In *Proc. 2nd ACM Int'l Conf. on Wireless Sensor Networks and Applications (WSNA '03)*, pages 108–114, San Diego, CA, Sept. 2003.

[32] Sameh Gobriel, Sherif Khattab, Daniel Mosse, Jos Brustoloni, and Rami Melhem. Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions. In *Proc. 3rd Annual Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '06)*, pages 595–604, Reston, VA, Sept. 2006.

[33] Chao Gui and Prasant Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proc. 10th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom '04)*, pages 129–143, Philadelphia, PA, Sept.–Oct. 2004.

[34] Maria Halkidi, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Resilient and energy efficient tracking in sensor networks. *Int.l Journal of Wireless and Mobile Computing*, 1(2):87–100, 2006.

[35] Mohamed Hamdi, Noureddine Boudriga, and Mohammad S. Obaidat. Whomoves: An optimized broadband sensor network for military vehicle tracking. *Int'l Journal of Communication Systems*, 21(3):277–300, 2008.

[36] Feng Han, Ying Chang, Pingyi Fan, and Athanasios V. Vasilakos. Collaborative beamforming and dirty paper coding assisted transmission strategy for e-health wireless sensor networks. In *Proc. 5th Int'l Wireless Communications and Mobile Computing Conf. (IWCMC '09)*, pages 1080–1084, Leipzig, Germany, June 2009.

[37] Tian He, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. AIDA: Adaptive application-independent data aggregation in wireless sensor networks. *ACM Transactions on Embedded Computing Systems*, 3(2):426–457, 2004.

[38] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Proc. 23rd Int'l Conf. on Distributed Computing Systems (ICDCS '03)*, pages 46–55, Providence, RI, May 2003.

[39] John Heidemann and Ramesh Govindan. An overview of embedded sensor networks. Technical Report ISI TR-2004-594, USC/Information Sciences Institute, Nov. 2004.

[40] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. 33rd Hawaii Int'l Conf. on System Sciences (HICSS '00)*, pages 8020–8029, Maui, HI, Jan. 2000.

[41] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1:660–670, Oct. 2002.

[42] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proc. 9th Int'l Conf. on Architectural Support for Programming Languages and Operating systems (ASPLOS '00)*, pages 93–104, Cambridge, MA, Nov. 2000.

[43] Wen Hu, Nirupama Bulusu, Chun Tung Chou, Sanjay Jha, Andrew Taylor, and Van Nghia Tran. Design and evaluation of a hybrid sensor network for cane toad monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 5(1):1–28, 2009.

[44] Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor network. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 134–147, Baltimore, MD, Nov. 2004.

[45] Hongwei Huo, Youzhi Xu, Hairong Yan, Saad Mubeen, and Hongke Zhang. An elderly health care system using wireless sensor networks at home. In *Proc. 3rd Int'l Conf. on Sensor Technologies and Applications (SENSORCOMM '09)*, pages 158–163, Athens/Glyfada, Greece, June 2009.

[46] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. 8th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom '00)*, pages 56–67, Boston, MA, Aug. 2000.

[47] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[48] Jihyoung Kim, Jungsoo Lim, Jonathan Friedman, Uichin Lee, Mani Srivastava, Mario Gerla, and Diego Rosso. Sewersnort: A drifting sensor for in-situ sewer gas monitoring. In *Proc. 6th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 09)*, Rome, Italy, June 2009.

[49] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steve Glaser, and Martin Turon. Wireless sensor networks for structural health monitoring. In *Proc. 4th ACM Conf. on Embedded Networked Sensor Systems (SenSys '06)*, pages 427–428, Boulder, CO, Nov. 2006.

[50] Younghun Kim, Thomas Schmid, Zainul M. Charbiwala, Jonathan Friedman, and Mani B. Srivastava. NAWMS: nonintrusive autonomous water monitoring system. In *Proc. 6th ACM Conf. on Embedded Network Sensor Systems (SenSys '08)*, pages 309–322, Raleigh, NC, Nov. 2008.

[51] Jinkyu Koo, Rajesh Krishna Panta, Saurabh Bagchi, and Luis Montestruque. A tale of two synchronizing clocks. In *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, pages 239–252, Berkeley, CA, Nov. 2009.

[52] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proc. 22nd Int'l Conf. on Distributed Computing Systems Workshops (ICDCSW '02)*, pages 575–578, Vienna, Austria, July 2002.

[53] Joanna Kulik, Wendi Heinzelman, and Hari Balakrishna. Negotiation-based protocols for disseminating information in wireless sensor network. In *Proc. 5th Annual Int'l Conf. on Mobile Computing and Netwroking (MobiCom '99)*, pages 169–185, Seattle, WA, Aug. 2002.

[54] Santosh Kumar, Ten H. Lai, and József Balogh. On k-coverage in a mostly sleeping sensor network. In *Proc. 10th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom '04)*, pages 144–158, Philadelphia, PA, Sept.–Oct. 2004.

[55] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach, Fifth Edition*. Addison-Wesley, 2009.

[56] John J. Lemon. Wireless link statistical bit error model. Technical Report NTIA Report TR-02-394, U.S. Department of Commerce, 2002.

[57] Christoph Lenzen, Philipp Sommer, and oger Wattenhofer. Optimal clock synchronization in networks. In *Proc. 7th ACM Conf. on Embedded Networked Sensor Systems (SenSys '09)*, pages 225–238, Berkeley, CA, Nov. 2009.

[58] Qun Li and Daniela Rus. Global clock synchronization in sensor networks. In *Proc. 23rd Int'l Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM '04)*, pages 564–574, Hong Kong, China, Mar. 2004.

[59] Stephanie Lindsey and Cauligi S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Proc. 2002 IEEE Aerospace Conf.*, volume 3, pages 3–1125 – 3–1130, Big Sky, MT, Mar. 2002.

[60] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proc. 4th Int'l Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN '04)*, Santa Fe, NM, Apr. 2004.

[61] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review*, 36(SI):131–146, Dec. 2002.

[62] Samuel Madden, Robert Szewczyk, Michael J. Franklin, and David Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 49–58, Washington, DC, June 2002.

[63] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless sensor networks for habitat monitoring. In *Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pages 88–97, Atlanta, GA, Sept. 2002.

[64] Vivek Mhatre and Catherine Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Ad Hoc Networks*, 2(1):45–63, 2004.

[65] A. M.Howard, J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proc. 6th Int'l Symposium on Distributed Autonomous Robotics Systems (DARS'02)*, Fukuoka, Japan, June 2002.

[66] Shinji Motegi, Kiyohito Yoshihara, and Hiroki Horiuchi. DAG based in-network aggregation for sensor network monitoring. In *Proc. Int'l Symposium on Applications on Internet (SAINT '06)*, pages 292–299, Phoenix, AZ, Jan. 2006.

[67] Suman Nath, Phillip B. Gibbons, Srinivasan Seshan, and Zachary R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 250–262, Baltimore, MD, Nov. 2004.

[68] Songhwai Oh, Phoebus Chen, Michael Manzo, and Shankar Sastry. Instrumenting wireless sensor networks for real-time surveillance. In *Proc. 2006 IEEE Int'l Conf. on Robotics and Automation (ICRA '06)*, pages 3128–3133, Orlando, FL, May 2006.

[69] P. Padhy, K. Martinez, A. Riddoch, H.L.R. Ong, and J.K. Hart. Glacial environment monitoring using sensor networks. In *Proc. 1st Workshop on Real-World Wireless Sensor Networks (REALWSN '05)*, Stockholm, Sweden, June 2005.

[70] Seung-Jong Park, Ramanuja Vedantham, Raghupathy Sivakumar, and Ian F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *Proc. 5th ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc '04)*, pages 78–89, Roppongi Hills, Tokyo, Japan, May 2004.

[71] Shwetak N. Patel, Thomas Robertson, Julie A. Kientz, Matthew S. Reynolds, and Gregory D. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *Proc. 9th Int'l Conf.*

on Ubiquitous Computing (UbiComp '07), pages 271–288, Innsbruck, Austria, Sept. 2007.

[72] Shaoliang Peng, Shanshan Li, Yuxing Peng, Wen sheng Tang, and Nong Xiao. Real-time data delivery in wireless sensor networks: A data-aggregated, cluster-based adaptive approach. In *Proc. 4th Int'l Conf. on Ubiquitous Intelligence and Computing (UIC '07)*, pages 514–523, Hong Kong, China, July 2007.

[73] Marina Petrova, Janne Riihijarvi, Petri Mahonen, and Saverio Labella. Performance study of ieee 802.15.4 using measurements and simulations. In *Proc. 2006 IEEE Wireless Communications and Networking Conference (WCNC '06)*, pages 487–492, Las Vegas, Apr. 2006.

[74] Greg J Pottie and William J Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[75] Vijay Raghunathan, Curt Schurgers, Sung Park, Mani Srivastava, and Barclay Shaw. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[76] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 181–192, Los Angeles, CA, Nov. 2003.

[77] Maneesha V. Ramesh. Real-time wireless sensor network for landslide detection. In *Proc. 2009 3rd Int'l Conf. on Sensor Technologies and Applications (SENSORCOMM '09)*, pages 158–163, Athens/Glyfada, Greece, June 2009.

[78] Antonio G. Ruzzelli, Raja Jurdak, and Gregory M.P. OHare. Multi-hop rfid wake-up radio: Design, evaluation and energy tradeoffs. In *Proc. 17th Int'l Conf. on Computer Communications and Networks (ICCCN '08)*, St. Thomas, VI, Aug. 2008.

[79] Loren Schwiebert, Sandeep K.S. Gupta, and Jennifer Weinmann. Research challenges in wireless networks of biomedical sensors. In *Proc. 7th Annual Int'l Conf. on Mobile Computing and Networking (MobiCom '01)*, pages 151–165, Rome, Italy, July 2001.

[80] Cory Sharp, Shawn Schaffert, Alec Woo, Naveen Sastry, Chris Karlof, Shankar Sastry, and David Culler. Design and implementation of a sensor network system for vehicle tracking and autonomous interception. In *Proc. 2nd European Workshop on Wireless Sensor Networks (EWSN '05)*, pages 93–107, Istanbul, Turkey, Jan. 2005.

[81] Kuei-Ping Shih, Sheng-Shih Wang, Hung-Chang Chen, and Pao-Hwa Yang. Collect: Collaborative event detection and tracking in wireless heterogeneous sensor networks. *Computer Communications*, 31(14):3124–3136, 2008.

[82] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 239–249, Baltimore, MD, Nov. 2004.

[83] Pavan Sikka, Peter Corke, Philip Valencia, Christopher Crossman, Dave Swain, and Greg Bishop-Hurley. Wireless adhoc sensor and actuator networks on the farm. In *Proc. 5th Int'l Conf. on Information Processing in Sensor Networks (IPSN '06)*, pages 492–499, Nashville, TN, Apr. 2006.

[84] Gyula Simon, Miklos Maroti, Akos Ledeczi, Gyorgy Balogh, Branislav Kusy, Andras Nadas, Gabor Pap, Janos Sallai, and Ken Frampton. Sensor network-based countersniper system. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 1–12, Baltimore, MD, Nov. 2004.

[85] Primoz Skraba, Qing Fang, An Nguyen, and Leonidas Guibas. Sweeps over wireless sensor networks. In *Proc. 5th Int'l Conf. on Information Processing in Sensor Networks (IPSN '06)*, pages 143–151, Nashville, TN, Apr. 2006.

[86] Ignacio Solis and Katia Obraczka. The impact of timing in data aggregation for sensor networks. In *Proc. 2004 Int'l Conf. on Communications (ICC '04)*, pages 3640–3645, Paris, France, Jun 2004.

[87] Wen-Zhan Song, Renjie Huang, Mingsen Xu, Andy Ma, Behrooz Shirazi, and Richard LaHusen. Air-dropped sensor network for real-time high-fidelity volcano monitoring. In *Proc. 7th Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys '09)*, pages 305–318, Kraków, Poland, June 2009.

[88] Stanislava Soro and Wendi B. Heinzelman. Prolonging the lifetime of wireless sensor networks via unequal clustering. In *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS '05) - Workshop 12*, page 236.2, Denver, CO, Apr. 2005.

[89] Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. The β-factor: measuring wireless link burstiness. In *Proc. 6th ACM Conf. on Embedded Network Sensor Systems (SenSys '08)*, pages 29–42, Raleigh, NC, Nov. 2008.

[90] Ivan Stoianov, Lama Nachman, Sam Madden, and Timur Tokmouline. Pipeneta wireless sensor network for pipeline monitoring. In *Proc. 6th Int'l Conf. on Information Processing in Sensor Networks (IPSN '07)*, pages 264–273, Cambridge, MA, Apr. 2007.

[91] Yanjun Sun, Omer Gurewitz, and David B. Johnson. A receiver initiated asynchronous duty cycle mac protocol for dynamic traffic load. In *Proc. 6th ACM Conf. on Embedded Network Sensor Systems(SenSys '08)*, pages 1–14, Raleigh, NC, Nov. 2008.

[92] Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 214–226, Baltimore, MD, Nov. 2004.

[93] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pages 32–41, Atlanta, GA, Sept. 2002.

[94] Hua-Wen Tsai, Chih-Ping Chu, and Tzung-Shi Chen. Mobile object tracking in wireless sensor networks. *Computer Communications*, 30(8):1811–1825, 2007.

[95] Karthikeyan Vaidyanathan, Sayantan Sur, Sundeep Narravula, and Prasun Sinha. Data aggregation techniques in sensor networks. Technical Report OSU-CISRC-11/04-TR60, Ohio State University, 2004.

[96] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 171–180, Los Angeles, CA, Nov. 2003.

[97] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurth. PSFQ: A reliable transport protocol for wireless sensor network. In *Proc. 1st ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 1–11, Atlanta, GA, Sept. 2002.

[98] Chieh-Yih Wan, Shane B. Eisenman, and Andrew T. Campbell. CODA: Congestion detection and avoidance in sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 266–279, Los Angeles, CA, Nov. 2003.

[99] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 28–39, Los Angeles, CA, Nov. 2003.

[100] Kui Wu, Yong Gao, Fulu Li, and Yang Xiao. Lightweight deployment-aware scheduling for wireless sensor networks. *Mobile Networks and Applications*, 10(6):837–852, 2005.

[101] Guoliang Xing, Chenyang Lu, Robert Pless, and Joseph A. O'Sullivan. Cogrid: an efficient coverage maintenance protocol for distributed sensor networks. In *Proc. 3rd Int'l Symposium on Information Processing in Sensor Networks (IPSN '04)*, pages 414–423, Berkeley, CA, Apr. 2004.

[102] Kaixin Xu, Mario Gerla, and Sang Bae. Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks. *Ad Hoc Networks*, 1(1):107–123, July 2003.

[103] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proc. 2nd ACM Conf. on Embedded Networked Sensor Systems (SenSys '04)*, pages 13–24, Baltimore, MD, Nov. 2004.

[104] Fan Ye, Gary Zhong, Songwu Lu, and Lixia Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. 3rd Int'l Conf. on Distributed Computing Systems (ICDCS '03)*, pages 28–37, Providence, RI, May 2003.

[105] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, Dec. 2004.

[106] Jennifer Yicka, Biswanath Mukherjeea, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[107] Ossama Younis and Sonia Fahmy. HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

[108] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.

[109] Wei Yuan, Srikanth V. Krishnamurthy, and Satish K. Tripathi. Synchronization of multiple levels of data fusion in wireless sensor networks. In *Proc. IEEE 2003 Global Telecommunications Conf. (GLOBECOM '03)*, pages 221–225, San Francisco, CA, Dec. 2003.

[110] S.A.R. Zaidi, M. Hafeez, S.A. Khayam, D.C. Mclernon, M. Ghogho, and E.Kim. On minimum cost coverage in wireless sensor networks. In *Proc. Information Sciences and Systems '09*, pages 213–218, Baltimore, MD, Mar. 2009.

[111] Honghai Zhang and Jennifer C. Huo. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc and Sensor Wireless Networks*, 1(1-2):89–123, 2005.

[112] Wensheng Zhang and Guohong Cao. DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE transactions on wireless communications*, 3(5):1689–1701, Sept. 2004.

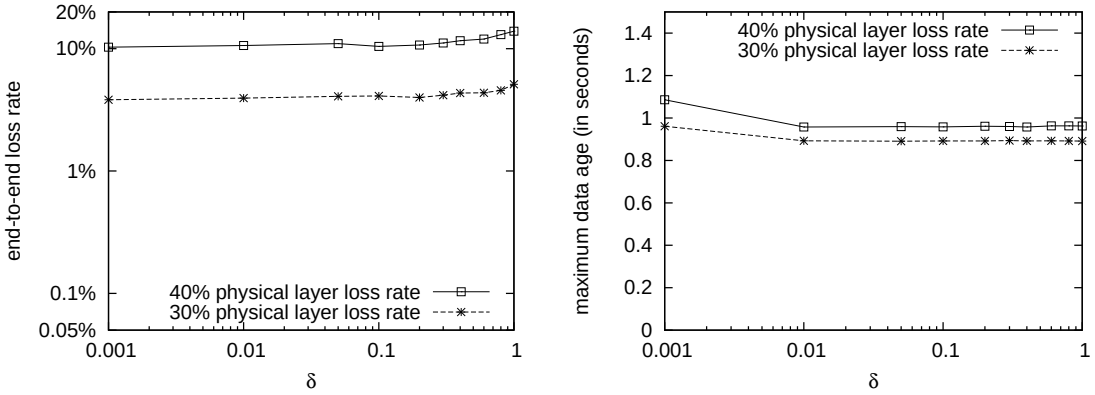
[113] Wensheng Zhang and Guohong Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *Proc. 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM '04)*, pages 2434–2445, Hongkong, China, Mar. 2004.

[114] Jerry Zhao and Ramesh Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. 1st ACM Conf. on Embedded Networked Sensor Systems (SenSys '03)*, pages 1–13, Los Angeles, CA, Nov. 2003.

Appendix A

Parameter Analysis for Chapter 3

Fig. A.1 plots performance results of asynchronous aggregation with EWMA for different δ. Asynchronous aggregation with EWMA degrades to basic asynchronous aggregation when δ is set to 1. The "timeout chain" phenomenon discussed in Section 3.2.2 causes higher end-to-end loss rate. When δ is really small (0.001), the adaption of the timeout values reacts too slowly when packets from all children are received in a round, leading to increased maximum data age. The protocol yields similar performance when δ stays within the range between 0.01 and 0.2. The value of δ is fixed at 0.05 in the simulations unless stated otherwise.

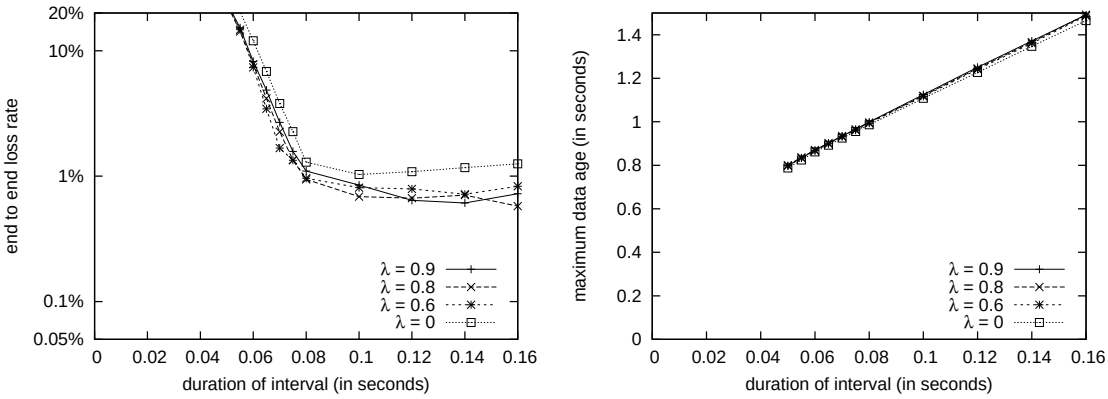


(a) End-to-End Loss Rate

(b) Maximum Data Age

Figure A.1: Impact of Parameter δ on Asynchronous Aggregation with EWMA (τ = 0.5 sec., N = 160, S = 250m × 250m)

Fig. A.2 shows the performance of synchronous aggregation for different α. When all nodes at the same tree level attempt to transmit at the same time, i.e., $\lambda = 0$, more packets are dropped due to collisions, resulting in higher end-to-end loss rate. Randomizing transmissions over $0.8I$ yields better performance than when all nodes at the same tree level attempt to transmit at the same time. The other values, 0.9 and 0.6, do not yield better performance.



(a) End-to-End Loss Rate

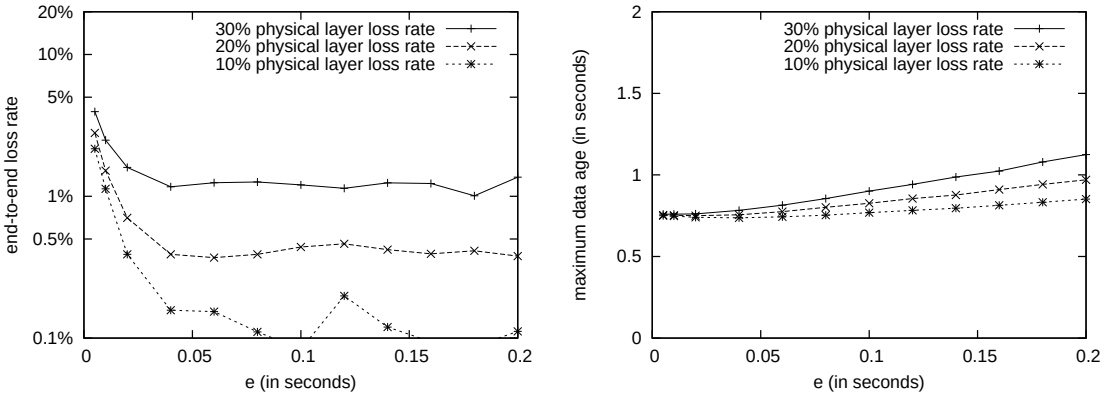
(b) Maximum Data Age

Figure A.2: Impact of Parameter λ on Synchronous Aggregation($\tau = 0.5$ sec., N = 160, S = 250m $\times$ 250m)

Appendix B

Parameter Analysis for Chapter 4

To study the impact of e on asynchronous broadcast-based aggregation, the duration of the randomization interval R is manually fixed to 0.2 seconds. The ability to adaptively adjust R is disabled in the simulations to eliminate possible impact of parameter R on the results. Fig. B.1 and Fig. B.2 plot performance results with fixed and increasing packet size, respectively. For fixed packet size, the results shows that fixing e at 0.06 seconds yields good performance for different physical layer loss rates. For increasing packet size, fixing e at 0.12 seconds yields good performance for different physical layer loss rates.

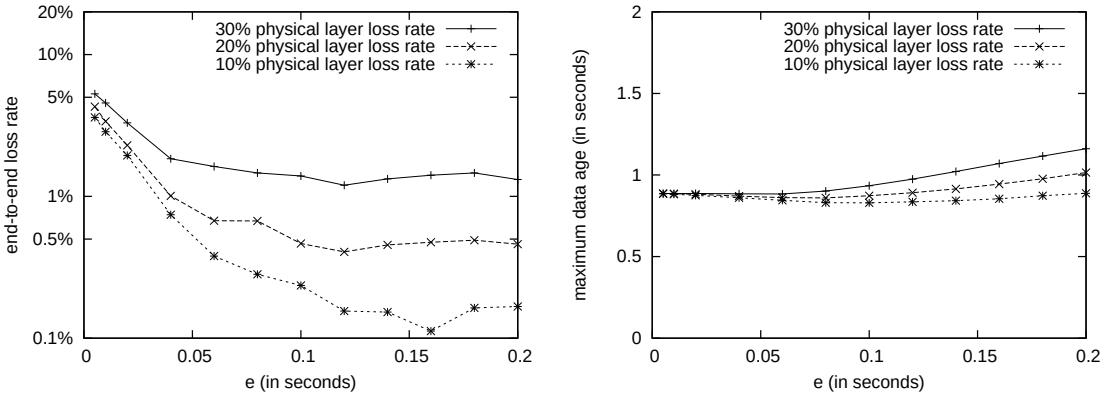


(a) End-to-End Loss Rate

(b) Maximum Data Age

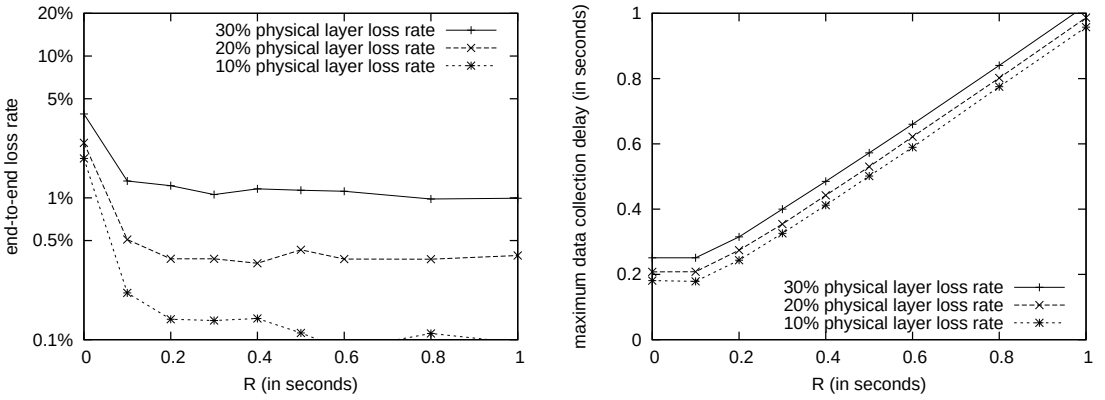
Figure B.1: Impact of Parameter e on Asynchronous Broadcast-based Aggregation ($\tau = 0.5$ sec., N = 160, S = 250m $\times$ 250m, $R = 0.2$ sec., fixed packet size)

To study the impact of R on asynchronous broadcast-based aggregation, the duration of the randomization interval R is manually set to different values. The ability to adaptively adjust R is disabled in the simulations. Fig. B.3 and Fig. B.4 plot performance results of asynchronous broadcast-based aggregation with fixed and increasing packet size, respectively. Parameter R has a similar impact on both asynchronous broadcast and unicast-based Aggregation. When R equals 0, more packets collide with each other because all leaf nodes attempt to transmit at the very beginning of each round. The end-to-end loss rate drops as the duration of the randomization interval gets longer. When R is great enough that the number of collisions are negligible, increasing R only leads to longer delay. Fig. B.3 and Fig. B.4 shows that the choice of R has a great impact on the performance of asynchronous broadcast-based aggregation. For the simulation results that are presented in Chapter 4, the value of R is adaptively adjusted in the proposed asynchronous broadcast-based aggregation protocol.



(a) End-to-End Loss Rate (b) Maximum Data Age

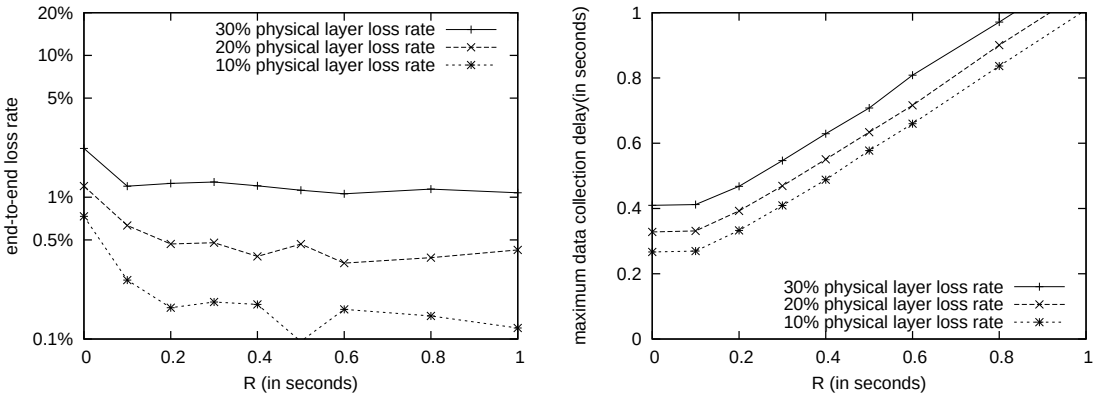
Figure B.2: Impact of Parameter e on Asynchronous Broadcast-based Aggregation (τ = 0.5 sec., N = 160, S = 250m × 250m, R = 0.2 sec., increasing packet size)



(a) End-to-End Loss Rate (b) Maximum Data Collection Delay

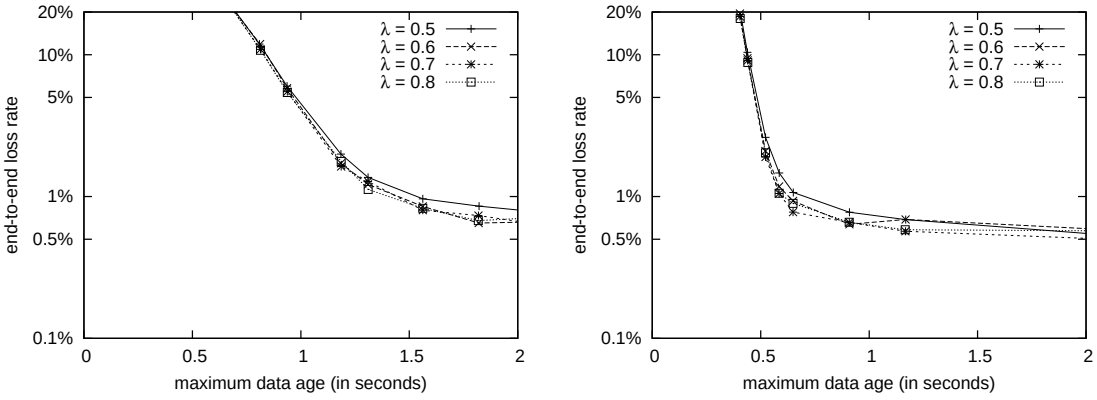
Figure B.3: Impact of Parameter R on Asynchronous Broadcast-based Aggregation (τ = 5 sec., N = 160, S = 250m × 250m, fixed packet size)

Fig. B.5 plots performance results of synchronous broadcast-based aggregation. The different points on each curve are generated by varying the sampling period duration τ, and measuring the resulting maximum data age and end-to-end loss rate. The performance of the protocol is very close when λ is 0.6, 0.7, and 0.8. The performance is starting to get worse when λ is set to 0.5. Parameter λ is set to 0.8 for synchronous broadcast-based aggregation in the performance evaluation in Chapter 4 and Chapter 5.



(a) End-to-End Loss Rate (b) Maximum Data Collection Delay

Figure B.4: Impact of Parameter R on Asynchronous Broadcast-based Aggregation (τ = 5 sec., N = 160, S = 250m × 250m, increasing packet size)



(a) Increasing Packet Size (b) Fixed Packet Size

Figure B.5: Impact of Parameter λ on Synchronous Broadcast-based Aggregation (N = 160, varying τ, S = 250m × 250m, PLR = 20%)