Dynamic multicast and time synchronization in wide-area sensor networks.

DYNAMIC MULTICAST AND TIME SYNCHRONIZATION IN WIDE-AREA SENSOR NETWORKS

by

Qing Ye

A Dissertation
Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Engineering

Lehigh University

November 2007

Approved and recommended for acceptance as a dissertation in partial fullfillment of the requirements for the degree of Doctor of Philosophy.

12/7/2007
Date

[illegible]
Dissertation Director

12/7/2007
Accepted Date

Committee Members:

[illegible]
Liang Cheng

[illegible]
Mooi Choi Chuah

[illegible]
Brian D. Davison

[illegible]
Ted Ralphs

To my wife Rui and my son Eric

For their priceless love and support

Acknowledgments

This research was carried out in the Laboratory Of Networking Group (LONGLAB) at the Department of Computer Science and Engineering, Lehigh University, during 2003-2007. I wish to express my sincere gratitude to all those who have helped me in finishing this study.

I am extremely grateful to my thesis advisor, Professor Liang Cheng. From the very beginning, he taught me to think not only as an engineer but also as a researcher. My minds are more open on not always focusing on finding the solutions but finding ways to identify and analyze the problems. His great enthusiasm in research have made working on this dissertation a wonderful experience. I also value his influence on my vision of my professional and personal developments.

I am indebted to the other members of my committee: Professor Brian D. Davison, Professor Mooi choo Chuah, and Professor Ted Ralphs. They have provided constructive feedbacks and discussions on my research and given helpful advises and comments on my research skills. I would like to thank them for their encouragement and critiques on my dissertation.

I owe my thanks to all members in the LONGLAB for creating a friendly and inspiring atmosphere over these years.

At last, I give my sincere thanks to all my family members. Without your love and support I can achieve nothing.

Contents

List of Tables

List of Figures

Abstract

The event surveillance applications need to collect spatiotemporal event observations from different interested sensing areas. This objective can be achieved by using technologies of Wide-Area Sensor Networks (WASNs), which are emerging wireless networks that have a two-tier infrastructure: *i*) the lower tier consists of multiple Wireless Sensor Networks (WSNs) which are deployed to monitor the interested events; and *ii*) the upper tier consists of mobile data collectors (such as unmanned automatic vehicles) which move around the sensing areas to forward event queries and reports. The mobile data collectors form the so-called Intermittently Connected Networks (ICNs). The connectivity of this infrastructure is unreliable, because *i*) different sensor networks may be deployed in geographically disconnected areas; and *ii*) the end-to-end paths may not exist among mobile data collectors due to node mobility. The key challenge is how to handle the dynamic topology variations in disseminating queries and collecting event reports. Also, globally synchronizing clocks in WASNs is required such that the observed events have accurate timestamps. This dissertation is the first one to address these issues in the overall data communication environment of WASNs.

The first contribution of our work includes a set of dynamic multicast protocols. On-demand Situation-aware Multicast (OS-Multicast) protocol is proposed for ICNs to distribute event queries to the mobile data collectors which can further forward

the queries to the distributed sensor networks. OS-Multicast dynamically builds up the multicast structure according to the current network situations. It forwards data whenever there is a discovered opportunity to reach the destinations. Therefore, OS-Multicast improves the successful message delivery ratio by introducing redundant traffic. OS-Multicast doubles the message delivery ratio than DTBR, the first dynamic multicast protocol for ICNs, when the total unavailability of contacts is more than 70% of the overall system operating time. Priced Track and Transmit (PTNT) protocol is proposed to help mobile data collectors gather event reports from the sensors. By distributedly counting how long the data collectors stay within the neighborhood of each sensor, PTNT is capable of tracking the movement of mobile data collectors without requiring GPS information. PTNT also gives system designers the capability of making a tradeoff between the message delivery efficiency and the average message forwarding delays, by controlling how often the queries are flooded.

The second contribution of our work involves a set of time synchronization protocols. Double-pairwise Time Protocol (DTP) is proposed for ICNs to keep clocks of mobile data collectors ticking at the same speed. By doubling the synchronization messages and applying a new message filter to estimate the relative clock drift, DTP cuts the average time-synchronization error by about half comparing to NTP-Core, which models the synchronization exchange method of NTP which is the Internet time standard for decades. The idea of DTP can also be directly implemented in the current synchronization architecture of NTP. The LEvel Synchronization using Sender, Adjuster and Receiver (LESSAR) is then proposed to synchronize clocks in WSNs. LESSAR is a lightweight approach to addressing the concern of energy consumption in WSN time synchronization by only requiring the adjusters to conduct two-way time message exchange with the time servers. Thus, LESSAR always uses less synchronization traffic than TPSN to achieve similar time accuracy, which is the default time protocol used in TinyOS, i.e. the operating system of off-the-shelf wireless sensors.

In summary, we describe solutions of providing dynamic multicast and time synchronization services in WASNs. Analytical models for performance analyses of some existing solutions and the proposed approaches are also presented in this dissertation.

Chapter 1

Introduction

1.1 Wide-area Sensor Networks

Wide-area Sensor Networks (WASNs) are emerging wireless networks which are viewed as the building components for a broad range of applications to collect spatiotemporal event observations in large areas. The potential applications in WASNs include medical/health care, habitat monitoring, and environmental surveillance.

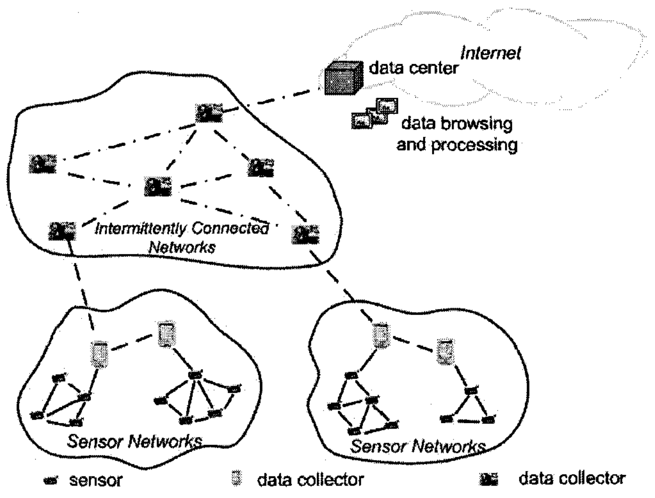

Figure 1.1: The two-tier infrastructure of Wide-Area Sensor Networks.

WASNs consists of two types of nodes: *i*) sensors that are lightweight and battery powered to observe different interesting events using different sensing devices on board; and *ii*) data collectors that are powerful and mobile to explore the distributed sensors and serve as the data carriers to forward the event queries and reports. A two-tier architecture of the wide-area heterogeneous sensor networks is shown in Fig. 1.1, in which the upper tier consists of all the mobile data collectors and the lower tier consists of all the distributed sensors. The upper tier forms Intermittently Connected Networks (ICNs) and the lower tier forms Wireless Sensor Networks (WSNs). We assume that each sensor network has at least one dedicated data collector to collect the event reports. This data collector is also called the data sink or sink. Event reports are first collected by the sinks, then delivered to the central data center via the upper-tier ICNs for browsing, post-processing and logging the events. The data center may be connected to the Internet if needed.

1.1.1 The Upper-tier Intermittently Connected Networks

In the last two decades, much research activities have been conducted for mobile wireless ad-hoc networks (MANET). Unlike the traditional wireless networks such as cellular networks, MANET do not have a pre-defined infrastructure and nodes directly communicate to each other rather than using the access points. Messages are forwarded from node to node until they reach the desired destination. Routing schemes in MANET, such as DSDV[72], AODV[73] and DSR[52], have a common fundamental assumption of that there exists an end-to-end path from the source to the destination. However, wireless links may be intermittently connected as the result of planned or unplanned link unavailability periods. Such link layer challenges are caused by high node mobility, low network density, limited radio ranges, scheduled node unavailability, or unexpected infrastructure disruptions.

Recent research in the Intermittently Connected Networks (also denoted as Disruption Tolerant Networks (DTNs)) [19, 36, 74, 50] addresses these challenges. Communication environments of ICNs feature frequent network partitions between data communication pairs. ICN technologies target on handling data communication in networks where instantaneous end-to-end paths become unstable and inconsistent. In ICNs, messages (called bundles) are transmitted in a store-and-forward fashion as illustrated in Fig. 1.2. A bundle is first stored in the local buffer of its current custodian and then forwarded when the link to the next hop becomes available. Due to the frequent topology variations, the routing decisions at each node may have to vary from time to time to take advantage of data forwarding opportunities. Therefore, dynamic routing approaches that are able to adjust the routing decisions according to the current network situations are required in ICN environment.

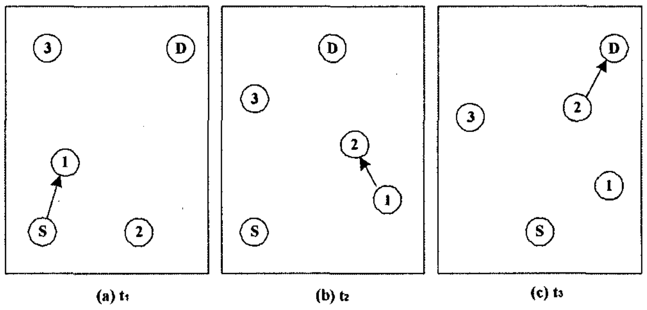

Figure 1.2: Data transmission from node S to D in ICNs, given $t_3 > t_2 > t_1$.

In our designed system, an ICN node could be a bus, a vessel, or an unmanned automatic vehicle that equips two types of radios: one as same as sensor's radio and the other radio that is used to communicate with other ICN nodes.

1.1.2 The Lower-tier Wireless Sensor Networks

The lower-tier Wireless Sensor Networks are composed of tiny wireless nodes that are capable of sensing the changes of the surrounding environments, processing the sensed data and communicating with each other through wireless radios. Sensors are usually deployed in a dense mode and form a connected network to monitor the application-specific events. Each sensor network covers certain interested area and are capable of performing different event surveillance tasks. Therefore, a large area is covered by several sensor networks, which do not need to be connected to each other. The data communication between sensor networks and the central data center is carried out by the mobile data collectors in the upper-tier ICNs.

Compared to the other wireless-capable nodes such as laptops and PDAs, sensors typically have limited power, low computational capability and small size of memory. For example, a commercial version, the Crossbow MPR410, only has a 433MHz Atmel ATMega processor and 128k memory and is powered 2 AA batteries [3]. To prolong the system lifetime, it is required that the protocols designed for sensor networks must be lightweight in terms of computational complexity, storage usage, and energy consumption. The routing schemes designed for MANET or the Internet are too heavy weight to be utilized by WSNs.

Both the ICNs and WSNs can be viewed as subsets of MANET. Table 1.1 compares their differences.

Table 1.1: Comparisons of ICNs, WSNs and MANET

	ICNs	WSNs	MANET
Connectivity	Frequently Disconnected	Connected	Connected
Density	Low	High	Medium
Mobility	High	Low or None	Medium
Propagation Delays	Large	Small	Small
Computation Capability	Good	Bad	Good
Local Storage	Large	Tiny	Medium
Communication Range	Long	Short	Long

1.2 Motivations

Many applications have been proposed using wireless sensors. For example, CodeBlue [62] is a health care project by attaching sensors to the in-patients in a hospital. The sensors monitor the vital medical data such as the heart beat rate and the body temperature of the patients and report the observations to the doctors' PDAs. Alarms are triggered when some attentions from the doctors are needed and data are eventually stored in the database. ZebraNet [53] is a habitat monitoring application to track the traces of a group of zebras in Kenya. The GPS sensors carried by zebras periodically record their positions. The biologists then collect the data to study the migrations and the inter-animal activities of zebras. PoleNet [91] is an environmental surveillance project to study the ice melting speed in the southpole. Sensors are deployed underneath the ice shell to monitor the temperature. Scientists then use small aircrafts that fly over the sensors to collect the event reports.

The technologies utilized in these applications are designed to support a single connected sensor network to perform event surveillance tasks in a small area. However, some challenging issues are raised when the scales of these applications are expanded to deploy thousands of hundreds sensors in multiple areas that are geographically disconnected. For instance, to monitor the bird activities in Indonesia which consists of over 18,000 islands, these issues include: *i*)it is costly and difficult to build up a huge infrastructure to keep all the sensor networks in different islands connected; *ii*) even if the overall network is connected, event reports can hardly be delivered to the data center if the sensors that observe the event are too far away. For example, assume that the packet loss rate over a WSN link is 3%. Then the packets transmitted from a sensor which is 100 hops away from the data center only has less than 5% chance to reach the destination; *iii*) because the capacity of WSN links is very limited, contentions and collisions may frequently happen in the networks when there is a large amount of reports generated by a lot of sensors that observe the events simultaneously ; and *iv*) the life time of those sensors close to the data center may quickly expire since they have to be kept activated to forward the received data.

The ICNs technologies target on providing networking services in communication environments where network partitions frequently occur can be utilized to address these issues. ICN routing schemes take advantage of the node mobility to overcome the disruptions of the end-to-end paths. Messages are forwarded to nodes that either have higher probability to meet the destination in the future or have better knowledge of how to deliver the messages closer to the destination. For example, DakNet [71] is an ICN application to provide internet access for the rural villages in India. The motorcycles that routinely move around different villages act as the mobile data forwarders. Messages are first stored in the hard drives carried by the motorcycles and then forwarded when the motorcycles encounter an access points connected to the Internet. The similar idea can be found in DieselNet [9] project. It constructs a mobile networks consists of city buses moving around Amhurst, MA. An link is set up whenever two buses meet with each other so that messages such as emails can be exchanged.

By combining the benefits of the technologies of both WSNs and ICNs, we propose the architecture of the wide-area sensor networks to support large-scale event surveillance applications which involve multiple sensor networks deployed in several geographically disconnected areas. In our designed system, the ICNs in the upper tier are responsible for relaying messages between the data center and the distributed sensor networks. The mobile ICN nodes forward the event queries/task assignments from the data center to multiple sensor networks. To achieve this, some ICN nodes have two types of radios: the long-range radio for communicating with other ICN nodes and the short-range radio for communicating with sensors. These ICN nodes moves around their dedicated sensor networks and are denoted as the mobile data collectors for the lower-tier sensor networks. They are also responsible of collecting event reports from sensors and unicast the data back to the data center using ICN routing schemes. In the Starbug project [86], it has been proven that having the mobile data collectors store-and-forward the messages is both economy- and energy-efficient to conduct data communication between the data center and the sensor networks.

In such system, we study two research problems: the Dynamic Multicast problem and the Network Time Synchronization problem. A sensory task, which is a spatiotemporal event query, needs to be conducted in two steps: task assignment step and data collection step. In the first step, tasks are generated at the data center and delivered to multiple sensor networks with the help of mobile data collectors. Multicast is an important service to support such one-to-many data communication. After that, even reports need to be collected from sensors that detect the events to the mobile data collectors and then forwarded back to the data center. The way mobile data collectors collect the reports is many-to-many multicast data communication. In this dissertation, we propose a set of dynamic multicast solutions to providing the above services. Moreover, event reports require accurate time stamps for the purpose of logging and processing events correctly. Some applications such as monitoring the temperature changes of volcanos or the vibrations of earthquakes require millisecond-level time accuracy. Therefore, how to provide accurate synchronized global time in WASNs is another essential research component of the event surveillance applications. As discussed above, the challenges of both research problems are the frequent topology disruptions caused by the mobile data collectors and the limited system resources of the wireless sensors. Our solutions to addressing these issues are introduced in later chapters.

1.3 Problem Definitions

1.3.1 The Dynamic Multicast Problem

To define the Dynamic Multicast problem, we have

- V, the set of nodes in the upper-tier ICNs. $V = \{N_0, N_1, ..., N_n\}$, where N_i represents the ith node and N_0 is the data center.

- $E(t)$, the set of the ICN links at time t between nodes in V. We assume that a link e_{ij} from N_i to N_j is a directional link. $e_{ji} \in E(t)$ if $e_{ij} \in E(t)$.

- R, the set of the multicast receivers in the upper-tier which are the mobile data collectors for the lower-tier sensor networks. $R \subset V$.
- $G(t) = (V, E(t))$, a directed graph which represents the time-varying ICNs at time t.
- $W^j(t) = (S^j(t), L^j(t))$, a time-varying graph which represents the jth sensor networks.
- S^j, the set of sensors in the jth sensor network in the lower-tier WSNs. $S^j = \{s_1^j, s_2^j, ..., s_m^j\}$.
- MDC^j represents the set of mobile data collectors in the jth sensor network, $MDC^j \in R$.
- M^j, the set of the sensors in the jth sensor-net to receive the sensory task.
- D, the set of tasks generated at N_0 that must be delivered to all the mobile data collectors in R, and then to be delivered to sensors.
- T_s, the start time of the multicast session in the upper-tier ICNs.
- T_e, the end time of the multicast session in the upper-tier ICNs.
- T_s^j, the start time of the multicast session in the jth WSNs, $\min T_s^j \geq T_s$.
- T_e^j, the end time of the multicast session in the jth WSNs.

It has two correlated subproblems: the Dynamic Multicast problem in the upper-tier ICNs and the Mobile Data Collection problem in the lower-tier WSNs.

We assume that the multicast session in ICNs is divided into $K = |T_e - T_s|$ time slots. Then the ICN DM problem is defined as: given a time-varying network $G(t) = (V, E(t))$, how to select a set of multicast structure $MS(t_1), MS(t_2), ..., MS(t_L)$ to route all the data in D from the source N_0 to all the receivers in R, where $T_s = t_1 < t_2 < ... < t_K = T_e$ and t_i is the ith time slot counted from time T_s. Note that $MS(t_i) \subset E(t_i)$.

In the similar way, the Mobile Data Collection problem in jth WSNs is defined as: given a time-varying network $W^j(t) = (S^j(t), L^j(t))$ and the constraints of system resources, how to select a set of multicast structure $ms^j(t_1), ms^j(t_2), ..., ms^j(t^j_k)$ to route all the reports from the sensors that observed the events in M^j to the mobile data collectors in MDC^j, where $k^j = |T^j_e - T^j_s|, T^j_s = t_1 < t_2 < ... < t^j_k = T^j_e$ and t_i is the ith time slot counted from time T^j_s.

The two multicast problems are similar from their definitions. However, they have different requirements and characteristics which will be discussed in the later chapters.

1.3.2 The Network Time Synchronization Problem

To provide the accurate time stamps for event reports, the clocks in WASNs needs to be synchronized. Network Time Synchronization in WASNs is conducted in a hierarchical way. We assume that the data center always has the correct real time. First of all, the clocks of the mobile data collectors are adjusted to follow the time of the data center. Secondly, the clock of each sensor-net is synchronized to its own data sink. Let t_i represent the reading of local clock of node N_i and let t_0 be the correct real time. Therefore, our objective of the NTS problem is to achieve $|t_i - t_0| < \varepsilon$, where ε is called the time accuracy.

1.4 Organization of This Dissertation

This dissertation is organized as follows. The exiting research results of multicast schemes proposed for the Internet and MANET are summarized in Chapter 2. I describe the designs of the dynamic multicast methods for ICNs and WSNs in Chapter 3, including the network model, the way to construct and maintain the multicast structure and how to handle dynamic membership. In Chapter 4, I propose a novel mathematical model to analyze the time errors of different synchronization approaches for WASNs. The reasons of network synchronization errors are discussed. Based on this analytical model, the error distribution and accuracy limitation of time synchronization are analyzed. This document is concluded in Chapter 5.

Chapter 2

Related Works

2.1 Dynamic Multicast

2.1.1 Multicast in the Internet

To realize in-network group communication, multicast is more efficient than multiple unicast or broadcast in terms of the utilization of network resources. Basically, multicast is defined as the one-to-many or many-to-many data communication. Multicast has become a highly desired service of most vendors' network products and a hot topic of the research community for over decades.

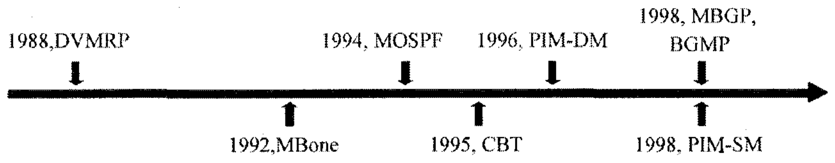

Figure 2.1: History of Internet Multicast Approaches

The thought of IP multicast was first introduced by Steve Deering in 1988 [23]. With implementing this idea into a large scale experiment of audiocast, the first generation of Multicast Backbone (MBone) in the Internet was created by IETF (Internet Engineering Task Force) in 1992 [13]. The connectivity between two MBone hosts was provided by a technique called tunneling, which set up a logic point-to-pint, IP-encapsulated link that may cross several Internet routers [6]. Distance Vector Multicast Routing Protocol (DVMRP) [88], which built up and maintained a tree in a broadcast-and-prune manner, was selected as the multicast approach. After that, a lot of Internet multicast protocols has been proposed. These tree-based approaches could be divided into two categories: source-rooted tree and shared tree. The idea of Multicast Extensions to OSPF (MOSPF) [70] and Protocol Independent Multicast Dense Mode (PIM-DM)[24] belong to the first category in which a shortest path tree is created and preserved from the source to all the intended receivers. However, Core Based Tree (CBT) [7] and Protocol Independent Multicast Sparse Mode (PIM-SM) [26] establishes a shared tree rooted at the rendezvous point (RP) in stead of the source. In this way, the source does not necessarily have to have an explicit list of the group members. All the group Join and Leave messages are handled by the RP. Moreover, several multicast trees could be maintained by the same RP to save more network resources.

2.1.2 Multicast in MANET

The above multicast solutions can be modified and extended in the wireless networks with pre-defined infrastructures. However, they have significant performance degradation in mobile wireless environments. For example, DVMRP could cause the dropping of multicast packets from fast moving hosts since it works based on relatively fixed routing tables [85].

The challenges that multicast design in MANET has to address include: *i*) dynamic topologies: In MANET, the node mobility results in variations of the end-to-end paths. The multicast tree also has to change from time to time and need to be maintained and recovered according to the current connectivity of the discovered paths; *ii*) unreliable links: the typical packet loss rate over wireless links varies from 1% to 30%, depending on the underlying physical and MAC layer designs. But the characteristic value of errors over wired connections is less than 1% [85]. Clearly, a reliable multicast approach that can achieve high degree of the eventual data delivery is required; and *iii*) limited energy: energy consumption is always a big concern for wireless devices. When a wireless node is out of power, it not only means the loss of a potential forwarder in MANET, but also means the loss of data it currently holds. An energy-efficient wireless multicast approach with few overheads is always preferred.

The simplest method to transmit packets to all the receivers is flooding. Flooding does not require any maintenance of a multicast structure so that it has no control overheads. However, the channel overheads of flooding are very large and increase with the increment of the network scale. And it is well known that uncontrolled flooding can cause broadcast storm problem. The overheads of flooding could be reduced by setting an upper bound of the Time-to-Live (TTL) value to control how far a packet is broadcasted. A method called Reverse Path Forwarding (RPF) [23] is used to suppress the redundant traffic by forwarding a packet only if it comes from the shortest path originated at the source. However, both TTL-controlled and RPF-controlled flooding still have a large amount of redundancy and low efficiency when the multicast group is sparse or wireless nodes move fast. Basically, multicast approaches designed for MANET can be categorized as tree-based and mesh-based methods as shown in Fig. 2.2. The difference between these two is whether to maintain a multicast tree or mesh in the wireless networks. The benefit of having a multicast tree is the high message delivery efficiency comes from the structure of a tree. And the benefit of the latter is the high message delivery ratio due to the multiple paths discovered in a mesh. Note that, flooding is a typical mesh-based multicast approach.

Adaptive Tree Multicast [43] is able to switch between the source-rooted tree (like DVMRP) and shared tree (like CBT) for conducting wireless multicast. A receiver can decide whether to register at the source or RP by comparing the length of the shortest paths to them. It tries to be adaptive to the node mobility but has more control overheads from both the source and RP. Ad-hoc Multicasting Routing Protocol (AMRoute) [92] is also tree-based algorithm. It first creates a virtual mesh that only includes the multicast source and receivers as well as the unicasting tunnels between them. Then a multicast tree is constructed from this virtual mesh. AMRoute is independent to the node mobility because it uses the technique of tunneling. As

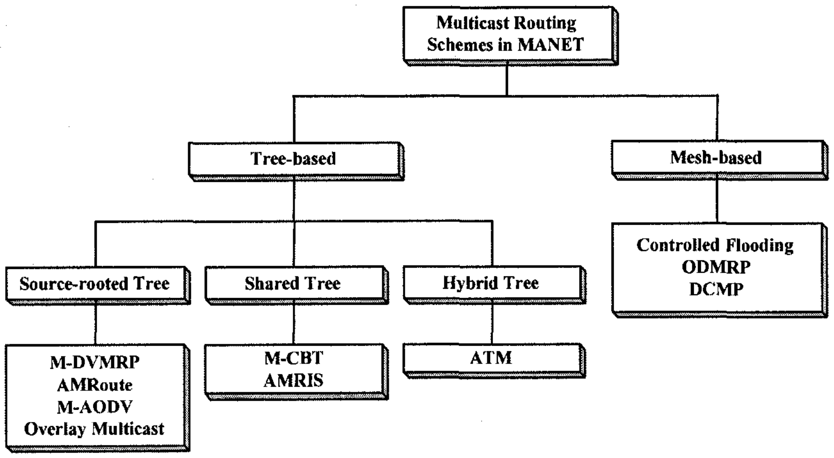

Figure 2.2: Multicast approaches for MANET

long as path between tree members exists in the virtual mesh, the multicast tree needs not to be changed. However, AMRoute may easily build up a non-optimal tree because the topology of virtual mesh is relatively fixed. The selected multicast tree is rarely adjusted even when a receiver may move very close to the source. The idea of Efficient Overlay Multicast proposed in [24] is as similar as the idea of AMRoute. It builds up a multicast tree from a so called virtual overlay but applies the shortest path tree algorithm with a novice way to calculate the cost. Multicast Operations of AODV (MAODV) [75], the multicast extension of the well-known AODV [73] routing protocol, is another tree-based method. It divides the networks into several groups and maintains a multicast tree only among the group leaders. Group leaders are selected along with the RREQ (Route Request) and RREP (Route Response) messages exchanges of AODV. And each receiver only needs to explicitly send Join and Leave messages to its current leader. MAODV is capable of dynamically adjusting the multicast tree. But it introduces more control overheads caused by group leader selections. The typical example of mesh-based multicast approach is On-Demand Multicast Routing Protocol (ODMRP) [76]. Multicast mesh is constructed by backward learning which is performed in the following way: *i*) the source periodically floods Join messages when it has data to send; *ii*) node receives the non-duplicate Join message will forward it till the message reaches a receiver; *iii*) each node assign the neighbor that directly forwards a Join message to itself as the forwarder. Thus a forwarder group, the multicast mesh, is constructed. One benefit of ODMRP is that receivers can inexplicably join or leave the multicast group without notifying the source. And ODMRP is able to achieve high message delivery ratio and small transmission delays. However, it has a large amount of control overheads as the result of the flooding of Join message and the control message exchanges between neighboring nodes to dynamically preserve the forwarding group. Dynamic Core Based Multicast Protocol (DCMP) [21] is also a mesh-based approach proposed for MANET. It is designed for many-to-many communications and the scenarios in which all the multicast sources have the same group of receivers. In DCMP, a core-based mesh is sharing by the multicast sources. Core nodes are dynamically selected to guarantee that each source is at most 2 hops away from a core node. The communication between the multicast sources and the core nodes is conducted by unicasting. And the communication among core-based mesh is performed in the same way as ODMRP. DCMP claims that it achieves much less control overheads than ODMRP. These multicast approaches give a good overview of how to conduct one-to-many data communications.

2.1.3 Multicast in WSNs

There are basically two ways to collecting event repots in large scale sensor networks: either by Push or Pull methods. Sensor nodes can periodically broadcast (push) their observations throughout the networks, no matter how the sinks are moving. Eventually, the data would be transmitted to destinations in a flooding manner. However, a lot of bandwidth is wasted by flooding, especially when events happen frequently. Sensors' observations can also be pulled out of the networks [33], via several unicasting links from data sources to the sinks. Pull method is relatively efficient than Push because it requires fewer nodes involved in data forwarding. However, those unicasting paths easily become invalid when sinks keep changing positions. Comb-Needle [61] is a hybrid push-and-pull approach proposed to support information gathering in WSNs. It first requires the sensors push and store the data into some relay nodes inside the networks. Then data sinks will pull data out of these intermediate relays, rather than retrieving from the original sources, to shorten the data forwarding paths. It balances the efficiency between push and pull by controlling how far the data should be pushed away, based on the frequency of queries and events. However, Comb-Needle is designed for the scenarios in which sinks will not move too fast.

To predict or to approximate the trajectory of moving objects, the target tracking problem is defined and discussed in [59, 94]. A straightforward approach is to attach GPS devices to each nodes, including the mobile data sinks. With the precise location information, the sink's movement can be monitored and well predicted. However, having GPS incurs additional cost and its performance will degrade when sensors are deployed indoor or under foliages [90]. Without having GPS, J. Aslam [29] proposes a scheme of using probabilistic models to estimate the movement of mobile targets. When a target moves, sensors will generate local views of the target's movement, based on the observations of environment (such as the variation of surrounding magnetic field). Each sensor then gives a deduction of whether the target is moving toward itself or moving away from itself. Thus, by collecting the judgments from all the nodes, a picture of the target's current movement could be painted. A probabilistic model is then used to filter all the information to predict the next possible position.

However, the overall estimations have to be done in a centralized way. That incurs delays between the sensors which detect the moving target and the central point which performs the computation. A distributed tracking approach based on acoustic information is proposed in [17]. It divides the sensor nets into several clusters with the use of Voronoi diagram. A static backbone consisting of all the cluster headers is then constructed and the mobile target is tracked cluster by cluster. However, the life time of cluster headers will be much shorter than other nodes because they become the computation and communication hot spots in [17].

Mobicast [31, 32] is a sink-to-sensor multicasting protocol. It studies the scenarios in which a moving sink (such as a soldier or an administrator) keeps querying the sensor nets and retrieving data only from its one-hop-away neighbors. Mobicast focuses on investigating how to forward the queries from the sink into a delivery zone, which consists of sensors located ahead of the sink's movement path. In this way, sensors in the delivery zone can be aware of the incoming data sink and begin to prepare for the data delivery. The size and shape of the delivery zone is dynamically adjusted, along with the sink's movement. Compare to mobicast, mobile multicasting in this paper studies different but more realistic scenarios: a group of mobile sinks monitor several hot spots in the sensor nets. Our study focuses more on the sensor-to-mobile-sink data forwarding.

VLM2 (Very Lightweight Mobile Multicasting) [77] is proposed to support both the sink-to-sensor and sensor-to-sink multicasting in WSN. In VLM2, the mobile sinks periodically floods beacons to the sensor nets. Those beacons are used to help each node update its distance to the mobile sinks so that sensor's observations can be transmitted along the shortest paths. However, VLM2 introduces a lot of control overheads due to the fact of that all the beacons have to be broadcasted throughout the whole networks. TNT and PTNT investigate the similar scenarios as VLM2, with dramatically suppressing the control overheads. The details of our proposed approaches for mobile multicasting are illustrated in the next section.

2.1.4 Multicast in ICNs

Routing in ICNs

The characteristics of occasionally-connected networking in ICNs could be found in many networks that are subject to disruptions and disconnections. For example, the inter-planetary internet designed to support deep-space data transmissions is a typical scenario for ICNs in which wireless links connecting spaceships or space stations may be periodically unavailable. Data communications via LEO satellites among several military bases across different continents is another example as data can be exchanged only when a satellite is visible to a military base. Wireless sensor networks which are deployed in very hash environments may also suffer from frequent network partitions. Moreover, in the networks consisting of highly mobile vehicles without powerful antennas, the duration of an available link between two moving nodes would be very short.

To overcome the frequent disruptions of end-to-end paths, a virtual message switching scheme is proposed in ICN routing designs. Basically, an ICN node is assumed to have a finite local buffer to hold messages that it generates and receives. Asynchronous messages (also denoted as bundles) [36] are forwarded from the source to the destination in a hop-by-hop store-and-forward manner.

The existing ICN routing protocols can be divided into two categories: knowledge-based and probability-based routing. Knowledge-based ICN routing schemes assume that certain information about the networks such as prior knowledge of the link connectivity pattern, the geographic locations, or the node movement schedules have been discovered by the networks. Then, routing decisions are made using Dijkstra-like algorithms to decide when and how a message should be forwarded. ED (Earliest

Delivery), MED (Minimum Expected Delay), EDLQ (Earliest Delivery with Local Queue) and EDAQ (Earliest Delivery with All Queue) methods proposed in [70], MV (Meetings and Visits) routing proposed in [12], and the message ferry routing and control schemes studied in [101, 102, 103] all belong to this category.

Probability-based ICN routing doesn't rely on any prior knowledge of the networks. In general, it requires that when each node encounters another node it estimates the probability of successfully delivering a bundle to the destination by taking the other node as the next hop. A bundle is forwarded when the probability is better than a certain threshold. How to estimate that probability differs in different ways. FC (First Contact) in [51] is the simplest one; it always trusts the first available contact to any neighboring node and forwards the bundles. However, loops may be easily found in FC. Epidemic Routing [84] makes two nodes exchange all not previously seen messages once they encounter each other. Its approach is based on the idea that randomly propagated message will eventually arrive at the destination. PROPHET (Probabilistic ROuting Protocol using History of Encounters and Transitivity) [60] extends the idea of Epidemic Routing by calculating the so-called delivery predictability based on the previous neighbors a node has encountered. A message is forwarded only when the next hop is predicted to have a better chance to meet the destination in the future. However, such prediction could be inaccurate. Spray-and-Wait [81] is another extension of Epidemic Routing. It first sprays a message to a certain number of nodes, and then waiting for one of the copies to be directly forwarded to the destination. In this way, Spray-and-wait is able to reduce the large amount of overhead introduced by Epidemic Routing. Also, different message spraying strategies could be developed and utilized according to the network conditions. MaxProp [9] is one of the latest ICN unicasting methods. Its routing decision is made based on the path likelihood which estimates the delivery probability of choosing one path to forward a bundle. The path likelihood varies according to the changes of ICN environments.

Compared to those knowledge-based routing approaches, the probability-based ICN routing methods often have non-global-optimal route selections because routing is always decided based on local information.

Multicast in ICNs

Multicast for ICNs is a considerably different problem compared to that of Internet and mobile ad hoc networks, because of the intermittent networking connectivity in ICNs, as discussed in Section 3.1.1. Multiple ICN multicast strategies have been proposed to overcome these difficulties and comparing their performance is a major goal of this paper.

To our best knowledge, there is no published paper study investigating how to provide multicast service over Disruption-tolerant Networks. However, certain applications such as the tactic information distribution between the command center and several oversea military bases using satellites do need multicasting to improve the data transmission efficiency. After summarizing the above works, we find that all these efforts are only studying robust multicasting over wireless networks in one or two aspects. *i*) The network robustness is regarded as the reliability of end-to-end data transmission. However, we believe that it's the system capability to tolerate disruptions autonomously. *ii*) There is no investigation taking the energy-efficiency, reliability and support of node mobility in a whole picture for wireless multicasting service. In this proposal, we are going to design a framework to improve the wireless multicasting from these aspects and we also want to find a generic multicasting model to help the research community.

2.2 Network Time Synchronization

A stand-alone hardware clock does not perfectly follow the Coordinated Universal Time (UTC) provided by National Institute of Standards and Technology (NIST), due to the inaccurate vibration frequency of its oscillator, the changes of environmental conditions, or the false configurations of the operating system.

The issue becomes even worse for two distinct clocks in a networking system, because there might be time differences between them from the beginning. Therefore, time synchronization is an essential and critical component for any distributed computer systems. For instance, using a global time *i*) to record and monitor system events with accurate time stamps; *ii*) to achieve collaborations among physically distributed components by activating them at the same time; and *iii*) to improve the network security of authentication and data encryption by taking the synchronized time as part of the public or private keys [68]. The fundamental task of time synchronization is to keep the computer clocks reporting the similar, and ideally the same and correct, time values when required.

There are several straightforward ways to make computer clocks run correctly, by applying a more precious clock board, attaching an antenna that receives time references from radio stations administrated by NIST, or utilizing a GPS receiver that synchronizes with satellites. However, it is too costly to require each computer in a big distributed system consists of thousands of hosts have a GPS to fulfill the timing need. Therefore, the software based approaches are needed to perform the synchronization job in an economic and effective way.

Over the last two decades, Network Time Protocol (NTP) [67, 66] has become the time keeping standard for the Internet because of its robustness and flexibility. NTP sets up a hierarchical infrastructure consists of a large set of primary and secondary time servers. Each host in the Internet synchronize its local clock with its own time server which are then synchronized with the upper-level primary time servers whose clocks are maintained by a trusted time reference or adjusted according to the time readings from GPS. In this way, NTP is able to achieve millisecond-level time accuracy.

2.2.1 Time Synchronization in WSNs

Due to the limited system resources, NTP is too heavyweight to be applied in the wireless sensor networks. To achieve the same time keeping goal, several lightweight synchronization methods have been proposed for WSNs. Post-facto [25] is the simplest method to synchronize a neighborhood of nodes. In Post-facto, every node synchronizes with a third party node that acts as a time beacon which periodically broadcasts time stimulus. When a stimulus arrives, each node then adjusts its local clock with respect to this time reference. Obviously, only sensors that are within the communication range of the beacon can be synchronized. RBS [48] extends the idea of Post-facto. In RBS after receiving a time reference, the receivers record the arrival time with respect to their own local clocks. Then they exchnge these observations to each other to calculate the clock offsets between any two nodes. Therefore, all the nodes are synchronized. Obviously, RBS incurs a large amount of overheads as the result of exchanging time messages. Both Post-facto and RBS can only synchronize the clocks of nodes that are one-hop away from the reference node. Tiny-Sync and Mini-Sync [78] are proposed to conduct the multi-hop time synchronization in WSNs. They require any pair of two neighboring nodes exchange their time information to get the so-called time points. After collecting a series of time points, a tight bound of their time differences is then calculated based on a linear model. In this way, Tiny-Sync and Mini-Sync make the two nodes synchronized. The drawback of Tiny-Sync and Mini-Sync is the large amount of overheads caused by two-way message exchanging between any pair of two neighboring nodes. TPSN [42] borrows the idea of NTP and simplified it for the sensor networks. It does not require the primary and secondary time servers but maintain a hierarchical synchronization structure by organizing sensors into different levels in terms of the distances to the time reference. Synchronization jobs are done from the upper level to the lower level. To achieve that, TPSN requires a steady network topology and symmetric links to guarantee that the time estimations it calculates are accurate. Similar requirement of network connectivity can also be found in TSHL [83], which is the first time protocol proposed for the underwater sensor networks. TSHL is performed in two phases: firstly, it estimates the clock drift using linear regression based on multiple one-way time beacons; secondly, it estimates the clock offset using two-way message exchange similar as TPSN. FTSP [64] is a MAC-layer synchronization approach (TPSN is actually the first one to use MAC-layer time stamping) proposed for wireless sensor networks. It floods time messages in the network and utilizes MAC-layer time stamps to estimate the clock offsets between wireless sensors. During the existing network time synchronization approaches for WSNs, FTSP achieves the best time accuracy as shown in their simulation results.

2.2.2 Time Synchronization in ICNs

Time synchronization in ICNs is considering as a different and difficult problem than that in the Internet and WSNs, because ICNs experience*i*)frequent network partition caused by high node mobility and *ii*) the asymmetric one-way delays caused by different queuing delays. [38] propose an idea called GCS which is able to guarantee that nodes close to the time servers can be more accurately synchronized and faraway nodes are loosely synchronized. GCS discusses the gradient property of synchronizing a distributed wireless networks. However, the design of GCS is still based on NTP and it doesn't address the issue of the frequent topology variations. CTP [45] is the first time protocol which addresses the asymmetric one-way delays of transmitting time messages in wireless networks. It claims that NTP-like approach is inappropriate and transfers the synchronization issue into a global optimization problem which is solved by linear programming. The optimal solution of CTP is a centralized protocol that requires the time offsets of all the nodes must be known. However, this is a very strong assumption for ICNs.

In this document, we conduct time synchronization in WASNs in a hierarchical way: first, all the mobile data collectors are synchronized with the data center; and second, all the sensors are synchronized with their own data collectors. To achieve these goals, we propose Epidemic Time Synchronization(ETS)to address the first issue and LEvel Synchronization using Sender, Adjuster and Receiver (LESSAR) to address the second issue. The details of these protocols are discussed in chapter 4.

Chapter 3

Dynamic Multicast in WASNs

3.1 On-demand Situation-aware Multicast in ICNs

3.1.1 Introduction

Motivations

ICNs(Intermittently Connected Networks) multicast is an important service to deliver messages from a source to a group of receivers. Network designers face some challenges when applying traditional multicast methods proposed for the Internet (e.g., MOSPF [70] and DVMRP [88]) or mobile ad hoc networks (e.g., AMRoute [92] and ODMRP [76]) to ICN environments. First, it is difficult to maintain the connectivity of a multicast structure (tree or mesh) during the lifetime of a multicast session. Second, data transmissions would suffer from many failures and large end-to-end delays because of the disruptions caused by repeatedly broken multicast branches. Third, the traditional approaches are designed with the assumption that the underlying networks are basically connected, which is not true in most ICN environments [100].

To address these issues, several ICN multicast strategies have been proposed, including unicast-based, static-tree-based and dynamic-tree-based multicast strategies. Each strategy has its own advantages and drawbacks due to different design philosophies. And within each strategy, one can design different variations of multicast routing schemes to achieve different QoS (Quality of Service) objectives. The major goal of this study is to study the performances of these multicast strategies in different ICN scenarios.

Our performance metrics include: *i*) *message delivery ratio*, which is defined as the number of unique multicast messages successfully delivered to the receivers over the total number of messages that are expected to be received; *ii*) *message delivery efficiency*, which is the ratio between the unique messages received by the receivers and the total traffic generated in the network; and *iii*) *average message delay*, which is the average of end-to-end message transmission delays. From our investigations and discussions, we aim to help system designers select the appropriate multicast routing scheme to meet the design requirements such as to achieve the most message delivery ratio, the smallest delay, or the best efficiency.

Network Model

ICNs are viewed as an overlay built upon certain underlying networks, such as mobile ad hoc networks as illustrated in Fig. 3.1. Only those nodes that implement ICN functionalities, e.g. the support of sending and receiving bundles, are considered as ICN nodes, while the others are denoted as normal nodes. This overlay that consists of all the ICN nodes is named as the ICN overlay. Different multicast routing strategies that are implemented for ICN nodes are viewed as the functions of the bundle layer in the proposed ICN architecture [14].

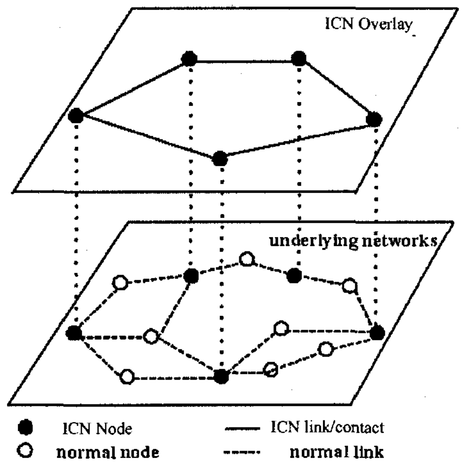

Figure 3.1: An example of the ICN overlay.

An ICN nodes A_j is called a neighbor of A_i if currently there is one end-to-end path connecting them in the underlying networks. We use link e_{ij} to represent the current connectivity from node N_i to N_j in the ICN overlay. The state of e_{ij} is up if and only if A_j is a neighbor of A_i. And the status of e_{ij} becomes down when there is no paths in the underlying networks connecting A_i to A_j at present. It is up again if at least one old path is reconnected or a new path is discovered. ICN unicast routing schemes are utilized in the ICN overlay to deliver a bundle from one ICN node to one of its ICN neighbors. When a ICN node has bundles destined for a neighbor and there is currently no available outgoing link to reach the destination, it will retain the data in its local buffer. We assume that each ICN node has a finite-size buffer for storing bundles.

Problem Definition

To define the multicast problem, we have the following notations:

- V, the set of nodes in the ICN overlay. $V = A_1, A_2, ; A_n$, where a_i represents the ith node and $|V| = n$;
- $E(t)$, the set of ICN links at time t between neighboring nodes in V. $e_{ij}(t) \in E(t)$ is the directed link from A_i to A_j at time t;
- R, the set of the intended multicast receivers. $R \subseteq V$ and $1 < |R| < n$;
- S, the set of the multicast sources that have messages for all the receivers in R. $0 < |S| < n - 1$ (there are at least 1 source and 2 receivers), and $S \cap R = \emptyset$;
- D, the set of data that the sources intend to deliver;
- $G(t) = (V, E(t))$, a directed graph which represents the time varying ICN overlay at time t;
- T_s, the start time of the multicast session;

- T_e, the end time of the multicast session. Let $Te - Ts = L$;

The ICN multicast problem is then defined as: given a time variant network $G(t) = (V, E(t))$, how to find a set of multicast structure $MS(t_0), MS(t_1),, MS(t_L)$ which are the sub-graphs of $G(t_0), G(t_1),, G(t_L)$ where $T_s = t_0 < t_1 < \dot{<} t_L = T_e$, to route the data in D from sources in S to the receivers in R, under the constraints of the current link connectivity in $G(t)$. Those multicast structures could be either tree or mesh which is not guaranteed to be connected in the ICN overlay.

3.1.2 ICN Multicast Strategies

Unicast-based Multicast (U-Multicast) Strategy

The simplest way to perform one-to-many data communication is to send the multicast bundles via multiple unicast operations from the source to each destination. Any ICN unicast routing scheme can be applied and extended to perform this task, with adding additional group information in the header of bundles. Some unicast routing schemes, such as the Epidemic Routing [84] and Spray-and-Wait [80] algorithms, are able to distribute the copies of the same bundles to multiple recipients according to their designs. U-Multicast strategy has the obvious advantage of the least implementation overheads. However, for $|R|$ numbers of receivers registered in a multicast group, U-Multicast requires the source deliver $|R|$ copies of each bundle to these receivers. Thus, the intermediate ICN nodes may repeatedly forward the same copy more than once. It then incurs the transmission overheads which may dramatically lower the message delivery efficiency of U-Multicast when there are many receivers.

Static-tree-based Multicast (ST-Multicast) Strategy

A spanning tree or a steiner tree is the typical structures which is widely used by many multicast protocols. The usage of a tree has been proven to be capable of reducing the transmission overheads in mobile ad hoc networks (MANET) [8]. In ST-Multicast strategy, a tree is constructed and maintained at the source when a multicast session starts. The source first collects all the discovered paths to the receivers and then builds up a smallest cost tree based on such information. By definition, the topology of the tree does not change in the intermediate nodes during the multicast session. And bundles are duplicated at every branching node based on how many downstream neighbors the node has. The overlay multicast approach proposed in [44] in MANET can be extended to be a ST-Multicast routing scheme in ICNs. It creates an additional multicast overlay which includes all the multicast group members in the ICN overlay. Then the tree is built up based on the logical topology in this virtual overlay no matter how the topology in the ICN overlay varies. ST-Multicast strategy is appropriate for some scenarios in which the network disruptions periodically happen in a fixed/scheduled pattern, for instance, the data communication via LEO satellites. In these cases, the discovered tree can act as the multicast backbone for delivering bundles. ST-Multicast strategy has smaller control overheads in terms of maintaining the multicast states than DT-Multicast strategy which will be discussed next. But it loses the flexibility of adjusting the multicast routing decisions according to the topology variations in ICNs.

Dynamic-tree-based Multicast (DT-Multicast) Strategy

DT-Multicast strategy dynamically adjusts the multicast tree to adapt to the current conditions in the networks. In DT-Multicast, each bundle has an associated tree that may change hop-by-hop according to the up/down variations of ICN links. Each node that has a bundle performs the same operations: *i*) to collect the information of the availability of ICN links to update its knowledge of the networks; *ii*) to compute the smallest cost tree based on its latest local view of the ICN overlay; and *iii*) to forward bundles using the discovered multicast tree. In DT-Multicast strategy, a node is capable of changing the multicast structure to take advantage of a newly available path to a destination or to avoid forwarding bundles along those currently broken branches that are discovered in the previous tree. Compared to the other two strategies, DT-Multicast strategy has more implementation and control overheads but is better adaptive to the topology variations in ICNs.

An example of the above three strategies is shown in Fig. 3.2.

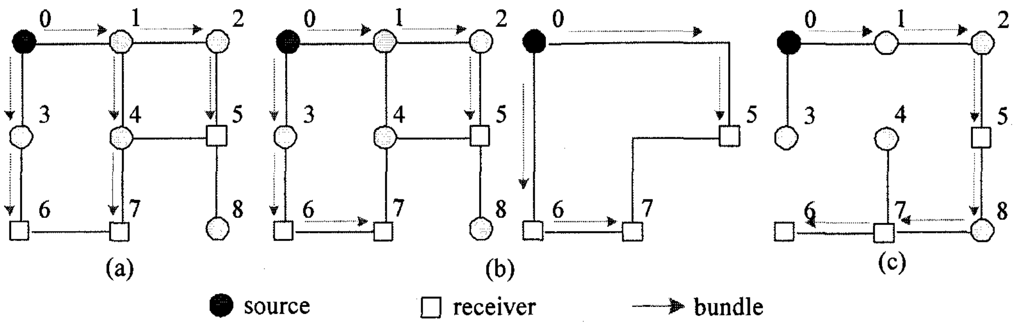

Figure 3.2: Examples of different ICN multicast strategies. Assume that node 0 is the source and nodes 5,6,7 are the receivers. (a) U-Multicast strategy. (b) ST-Multicast strategy: a multicast tree is created from a virtual overlay that only consists of the source and receivers. Then ST-Multicast keeps using this tree to deliver bundles. (c) DT-Multicast strategy: when links $3 \rightarrow 6, 1 \rightarrow 4, 4 \rightarrow 5$ are down, but link $8 \rightarrow 7$ is up, a new multicast tree is dynamically built to take advantage of the newly available link $8 \rightarrow 7$ to forward bundles to receiver node 7 and 6.

3.1.3 Details of DT-Multicast Strategy

In this section, we discuss the details of the DT-Multicast strategy by illustrating two typical DT-Multicast routing schemes: dynamic tree based routing (DTBR) and on-demand situation-aware multicast (OS-Multicast).

Basic Ideas of DTBR & OS-Multicast

DTBR [104] assumes that each ICN node has a certain knowledge oracle containing the schedule or the statistical summary of link up/down information in the ICN overlay. Based on this, the source computes a multicast tree for each bundle and forwards the current message along the tree. There is a receiver list associated with each copy of the bundle. It indicates for which receivers an intermediate node should be responsible. Initially, the list at the source contains all the intended receivers. If the source has more than one downstream node, it will put a new list that only consists of the receivers along that branch into the copy sent to each downstream next hop. Each node that receives a bundle will then re-compute a multicast tree to reach those destinations in the receiver list. This process is repeated hop-by-hop until a copy of the bundle is delivered to a receiver.

OS-Multicast [97] is similar to DTBR as it also builds up a dynamic multicast tree hop-by-hop for each copy of the bundle. However, it doesn't rely on any global knowledge of the network, such as node positions or link up/down schedule. It assumes that the underlying networks is able to record discovered routing information and report the current availability of outgoing links to the ICN multicast agent. In OS-Multicast, there is also a receiver list associated with each bundle. Unlike DTBR, it always contains a full list of all the intended receivers. Therefore, OS-Multicast requires each intermediate node that has a bundle be responsible for delivering the multicast message to all the receivers.

Common Operations of DTBR & OS-Multicast

In this section, we discuss the mutual operations of DTBR and OS-Multicast.

Membership management Each ICN node is assumed to be associated with an *endpoint ID*. A multicast source (or multiple sources) uses a group endpoint ID or an explicit list of the receivers as the destination address for delivering multicast bundles.

Several semantic models of ICN multicast membership have been studied in [104]. Both DTBR and OS-Multicast conform to one of them, the Temporal Membership semantic model, with an explicit receiver list known at the source. When a ICN node intends to join a multicast group, it registers its planned membership period by explicitly flooding a *GROUP_JOIN* message (e.g., node i claims to be interested in the multicast service during the period $[t_{si}, t_{ei}]$, with the start-time t_{si} and the end-time t_{ei}.) into the ICN layer. When the multicast source is informed by the *GROUP_JOIN* message, it puts the membership information into a *receiver_list*, denoted as L_M. For each bundle to be transmitted at time t, a ICN node will check the validity of receivers in L_M. If the membership of a receiver has expired ($t > t_{ei}$) or was not activated ($t < t_{si}$), then that receiver will not be included in the receiver list in the bundle. An intended receiver with expired membership will be removed from L_M. When a receiver wants to leave the multicast session, it could leave silently. But a new *GROUP_JOIN* message is required when this receiver wants to participate in the multicast service again in the future.

Multicast tree construction

As mentioned, DTBR and OS-Multicast collects the information of the ICN overlay by either querying the knowledge oracle or gathering the discovered routing information from the underlying unicasting method. Both compute the multicast tree by the *Dijkstra* shortest path tree algorithm using the hop distance as the cost, at each ICN node involved in the multicast transmission. The multicast structure is calculated dynamically based on the local view of the ICN overlay at each intermediate node. The generic pseudo codes of how DT-Multicast strategies make the multicast routing decisions and how the bundles are forwarded are shown in Fig. 3.3 and Fig. 3.4. *Action* $= 1$ means to forward the bundle and *Action* $= 2$ means to keep a copy of the bundle in the local storage.

```
Algorithm: DT-Multicast Decision
Input:   $G(t) = (V, E(t))$, bundle
output:  $MS_i(t)$, the multicast structure created by node $N_i$ at time $t$,
         $Action_i(\text{bundle}, t)$, how node $N_i$ process bundle at time $t$
 1: foreach receiver $r$ in bundle.$L_M$ do
 2:    if notvalidreceiver($r$, $t$) then
 3:       bundle.$L_M$ = bundle.$L_M$ - $r$
 4:    end if
 5: end
 6: $G'(t)$ = CollectLocalViewofICNLayer( $G(t)$, i)
 7: $Cost_i(t)$ = CalculateHopCounts( $G'(t)$, $i$)
 8: $MS_i(t)$ = Dijksta_ShortestPathTree( $G'(t)$, $Cost_i(t)$, $i$)
 9: $action_i(t) = 0$
10: foreach receiver $r$ in bundle.$L_P$ do
11:    if $r$ is reachable from $N_i$ in $MS_i(t)$ then
12:       bundle.$L_P$ = bundle.$L_P$ - $r$
13:       $action_i(t) = 1$
14:    end if
15: end
16: if $action_i(t) = 1$ and |bundle.$L_P$| > 0 then
17:    $action_i(t) = 2$
18: end if
```

Figure 3.3: Pseudocode of Generic DT-Multicast Strategy.

Multicast state maintainess

To dynamically maintain the tree, each bundle keeps a unique forwarding state, including an upstream list (called L_U) and a pending list (called L_P). The upstream list L_U contains the endpoint ID of ICN nodes a bundle has traversed. The purpose of L_U is to avoid possible routing loops and to reduce redundant traffic. When a bundle arrives, a ICN node creates L_P for that bundle by duplicating the associated receiver list L_M contained in the bundle. So initially $L_P = L_M$. After generating

```
Algorithm: DT-Multicast Action
Input:  $action_i(t)$, $MS_i(t)$, $bundle$
output: how node $N_i$ process $bundle$ at time $t$
 1: if $action_i(t) != 1$ then
 2:     if $|localbuffer_i(t)| <$ maxsize then
 3:         $localbuffer_i(t) = localbuffer_i(t) + bundle$
 4:     else
 5:         managelocalstorage($policy$, $localbuffer_i(t)$, $bundle$)
 6:     end if
 7: end
 8: if $action_i(t) != 0$ then
 9:     $nexthoplist$ = getthenexthopsfromtree($MS_i(t)$)
10:     foreach nexthop $d$ in nexthoplist
11:         $newL_M$ = calculatereceiverlist($bundle.L_M$, $MS_i(t)$)
12:         $newbundle$ = createcopy($bundle$)
13:         $newbundle.L_M = newL_M$
14:         $newbundle.L_P = newL_M$
15:         $newbundle.L_U = newbundle.L_U + N_i$
16:         forwardbundle($d$, $newbundle$)
17:     end
18: end if
```

Figure 3.4: Pseudocode of storing and forwarding bundles using the DT-Multicast strategy.

the multicast tree based on current network situations, it knows which receivers are reachable using this tree. Those covered receivers are then removed from L_P. If the resulting L_P is not empty, this node will put a copy of that bundle into its local buffer and wait for the future opportunities to reach those destinations stored in L_P. Otherwise, the bundle doesn't need to be buffered since all the receivers have been covered from its local view of the ICN overlay.

Each ICN node that has buffered bundles will periodically check if there is any chance to forward the buffered bundles further. If so, it then recalculates a multicast tree to reach the uncovered receivers in LP for each stored bundle. If one receiver is reachable by this new dynamic tree, it would be removed from LP, and a copy of that bundle will be forwarded. A buffered bundle is released when *i*) its lifetime has expired; *ii*) its associated LP is empty; or *iii*) the current buffer management policy decides to discard this bundle to avoid possible buffer overflow. Fig. 3.5 shows the pseudo codes of maintaining the multicast states in DT-Multicast strategy.

```
Algorithm: DT-Multicast Maintaining Operations
Input:  $MS_i(t)$
output: how node $N_i$ maintain the multicast state of buffered bundles
 1: if |$localbuffer_i(t)$| > 0 then
 2:   foreach buffered bundle $b$ in $localbuffer_i(t)$
 3:     foreach receiver $r$ in $b.L_P$
 4:       if $r$ isreachable from $N_i$ by $MS_i(t)$ then
 5:         $d$ = getthenexthop($MS_i(t)$, $r$)
 6:         if $d$ is not in $bundle.L_U$ then
 7:           $b.L_P = b.L_P - r$
 8:           $newL_M$ = calculatereceiverlist($b.L_M$, $MS_i(t)$)
 9:           $newbundle$ = createcopy($b$)
10:           $newbundle.L_M = newL_M$
11:           $newbundle.L_P = newL_M$
12:           $newbundle.L_U = newbundle.L_U + N_i$
13:           forwardbundle($d$, $newbundle$)
14:         end if
15:       end if
16:     end
17:     if |$b.L_P$| = 0 then
18:       $localbuffer_i(t) = localbuffer_i(t) - b$
19:     end if
20: end if
```

Figure 3.5: Pseudocode of maintaining the multicast states associated with each bundle in DT-Multicast.

Differences between DTBR & OS-Multicast

Different DT-multicast routing schemes differ in how the current information of the ICN overlay are collected. DTBR requires that each node has complete knowledge or a summary of the link states in the networks. However, this is difficult to satisfy in most practical applications. Without such a strong requirement, OS-multicast may operate over any ICN unicast method that is able to record historical routing information and detect the status of outgoing links.

Another significant difference between these two DT-Multicast methods is how they treat the receiver list L_M in multicast bundles. In DTBR, LM will be divided into subsets at each branching node in the multicast tree, while in OS-Multicast L_M will not change as bundles are transmitted. This simple modification allows OS-Multicast to utilize some bundle forwarding opportunities which are ignored by DTBR at the cost of additional redundant transmissions.

Consider two receivers R_i and R_j. We denote $G_i(t_1)$ as the set of neighboring ICN nodes of R_i at time t_1. For node N in the set of $G_j(t_1) - G_j(t_1) \bigcap G_j(t_1)$, if it has a bundle then N would deliver the bundle to R_i only, according to DTBR. Suppose that at time $t_2 > t_1$, there occurs a link connecting N and R_j, i.e., $N \in G_j(t_2)$, as the result of the dynamic topology variation in ICNs. Then bundles held by node N at t_2 can not be forwarded to R_j immediately, because N is not required to be responsible for covering R_j before t_2. The opportunity provided by link $N \longrightarrow R_j$ will not be utilized by DTBR until $N's$ upstream node in the tree discovers the new link and puts R_j into the receiver list of bundles forwarded to N. Moreover, bundle forwarding in DTBR stops at receivers R_i and R_j, if they are the leaf nodes of the multicast tree. Therefore, bundles successfully delivered to R_i will never be forwarded to R_j, even if there is currently an available link between R_i and R_j. Such issues with DTBR are overcome in OS-Multicast by keeping L_M the same along the tree.

Table 3.1: Simulation Parameters Without Node Mobility

Parameter	**Value**
Number of nodes	25
Area size	1000m $\times$ 1000m
Topology	5 $\times$ 5 grid
Number of multicast sources	1
Number of multicast receivers	4
Bundle size	512 bytes
Data rate	1bundle/2seconds
Bundle retransmission timer	5 seconds
MAC layer protocol	802.11
Transmission range	250m

However, OS-Multicast has the obvious disadvantage that it introduces much more redundant traffic into the network than DTBR. The performance of OS-Multicast will deteriorate when the traffic rate at the source is high, since redundant copies of delivered bundles will consume scarce wireless bandwidth.

3.1.4 Performance Comparisons

Performance Metrics

To evaluate the performance of different multicast strategies, we implement the ideas of U-Multicast, ST-Multicast, DT-Multicast (including DTBR and OS-multicast) in the ns2 simulator [4].

Results Without Node Mobility & Under Low Traffic Rate

The first set of simulations is conducted under the condition of a low data rate (1 bundle per 2 seconds) at the multicast source and no node mobility. The purpose is to have a general overview about the performance of different multicast approaches. The experiments are conducted by varying the inter-contact duration (defined in the next paragraph) , the local buffer size within ICN nodes, and the scale of the ICN layer. We also apply real-world ICN traces in the simulations. The common parameters of these tests are listed in Table 3.1. To emulate the characteristics of disruption tolerant networks, we have modified the ns2 simulator to manage the link up/down schedules. Different multicast algorithms are implemented in the ICN layer consisting of the ICN nodes randomly selected from 25 nodes in total.

The impact of the inter-contact duration

In this simulation, each link experiences a randomly generated series of up/down periods. The inter-contact duration is the down time between two up time periods. For example, if the up time of a link spans [t_1, t_2], [t_3, t_4], and [t_5, t_6], then [0, t_1], [t_2, t_3], and [t_4, t_5] are the inter-contact durations of this link. In this test, we vary the total length of the inter-contact durations of each link from 10% to 90% of the overall simulation time. 15 out of 25 nodes are randomly selected to be ICN nodes. Each ICN node can keep at most 100 bundles in its local buffer.

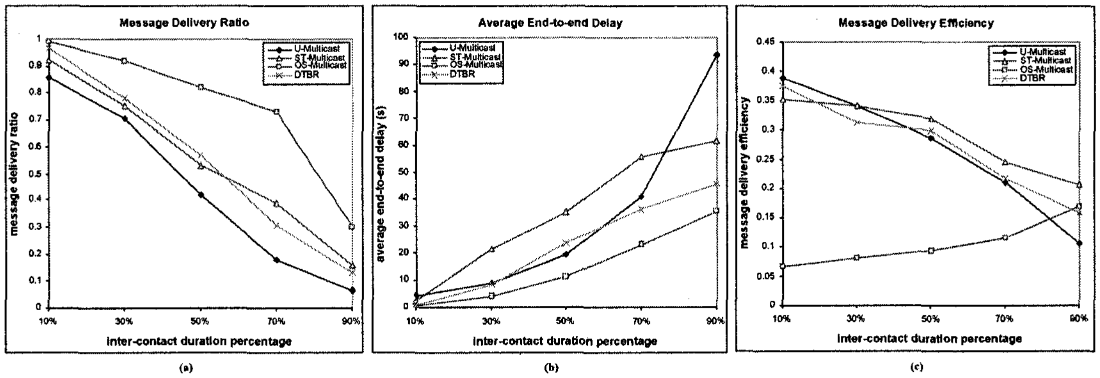

Figure 3.6: Simulation results of varying the inter-contact duration.

Fig 3.6 shows the results. We observe that: *i*) fewer bundles could be delivered to the destinations when the percentage of inter-contact duration becomes larger, i.e., the lack of network connectivity becomes severe; and *ii*) OS-Multicast achieves the best message delivery ratio because of its dynamic nature plus flooding to utilize almost all the opportunistic links to push data closer to the destinations. Fig 3.3(b) depicts the average end-to-end delay performance. Both U-Multicast and ST-Multicast perform worse than two DT-Multicast approaches, because they lack the flexibility to dynamically maintain the multicast structure. OS-Multicast achieves the smallest delay. However, OS-Multicast has the worst efficiency when the percentage of inter-contact duration is small as shown in Fig 3.3(c). Because each intermediate node in the multicast structure of OS-Multicast has the whole list of receivers, multiple copies of the same bundle may be delivered when the network remains well connected. But, as the percentage of inter-contact duration becomes very small, the efficiency of OS-Multicast will benefit from its capability to deliver more bundles than the other three algorithms.

The impact of local buffer size

The size of local storage within each ICN node affects the performance of different multicast approaches. If the size is too small, some bundles have to be discarded due to buffer overflow. Those bundles then immediately lose the chance to be forwarded even when some opportunistic links would be available in the near future.

In this simulation, we fix the percentage of the inter-contact duration to be 70% of the total simulation time and vary the buffer size of each ICN node from holding at most 25 bundles to 100 bundles. As shown in Fig 3.7(a), when ICN nodes have more local storage, more bundles can be delivered to receivers. The average end-to-end delay also increases when the size of local storage increases. This is because with more local storage, some bundles that are dropped in the case of smaller storage could be buffered for longer time until they get a chance to be forwarded. Thus, some of the delivered bundles may have been held in the network and experience

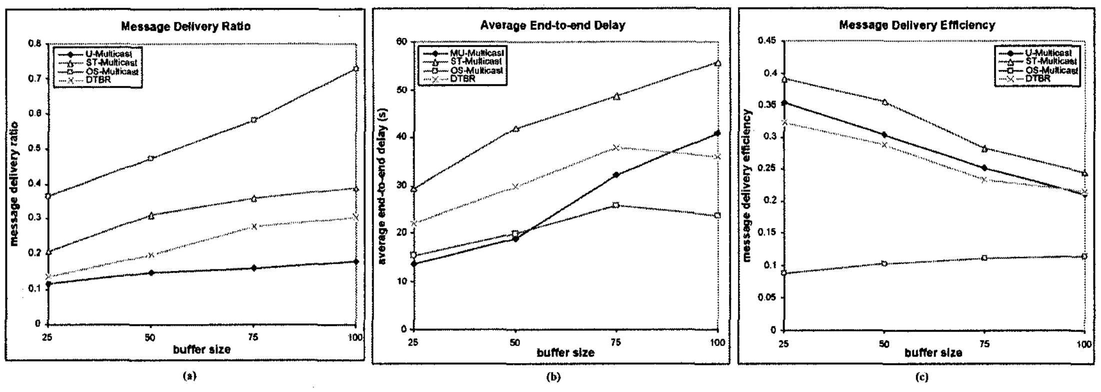

Figure 3.7: Simulation results of varying the local buffer size of each ICN node.

longer queuing delays before reaching their destinations. Fig 3.7(c) shows that for U-Multicast, ST-Multicast and DT-Multicast, the message delivery efficiency decreases when the buffer size increases, because more traffic is introduced into the networks as a result of the retransmissions of more buffered bundles. OS-Multicast has the best message delivery ratio and the smallest delays but the worst efficiency.

The impact of the percentage of ICN nodes

We are also interested in the impact of the scale of the ICN layer. As mentioned earlier, ICN functionalities are only implemented in those nodes that form the ICN layer. Our intuition in this experiment is that with more ICN nodes supporting multicast, the performance of all algorithms would become better.

We vary the percentage of the ICN nodes from 20% to 100% of the total nodes in the network. The results in Fig 3.8 show that the message delivery ratios of all the approaches increase. However, their performances in terms of end-to-end delay and efficiency drop with more ICN nodes. By studying the simulation traces, we find that when more nodes support ICN functionality, generally more potential paths to the receivers are discovered at each intermediate node. This indicates that

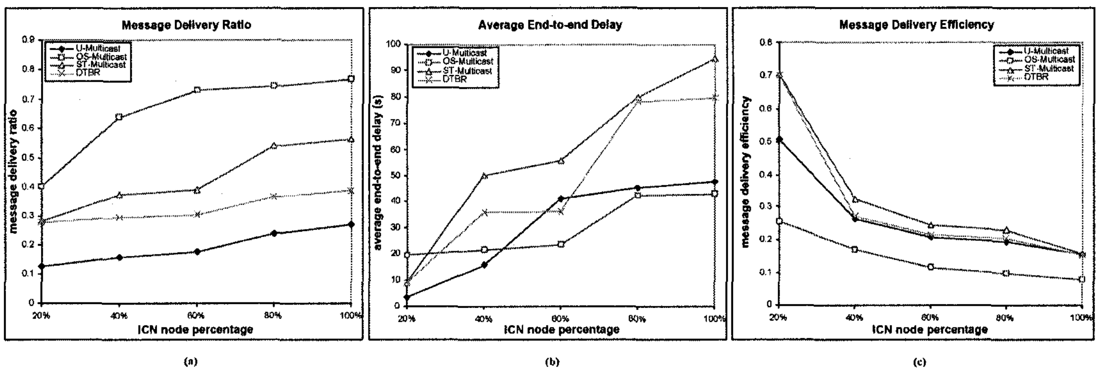

Figure 3.8: Simulation results of varying the scale of the ICN layer.

there are larger possibilities for bundles to be forwarded to the destinations. Thus, it improves the message delivery ratios of different multicast approaches. However, with more ICN nodes, the average lengths of the discovered paths and the average local buffer usage of ICN nodes also increase. Therefore, bundles experience longer queuing delays. It explains why the average end-to-end delays become larger. Moreover, more retransmissions take place as more opportunistic paths are discovered. It incurs more redundant traffic and brings down the efficiencies of all the approaches.

Applying trace data from ICN testbeds

Previously, we observed that different percentages of link up/down durations affect the performances of multicast approaches. In this simulation, we study the impact of different link up/down patterns. In our previous simulations, the probability of having a short inter-contact duration of a link is as same as that of having a long one (i.e., uniformly distributed). However, ICN testbeds/experiments such as PSN (Pocket Switch Networks) [47] and DieselNet [11], have reported that the cumulative distribution function (CDF) of the inter-contact durations approximately follows a power-law distribution. It indicates that in practical cases ICN links are usually up for very short periods of time. Based on their observations, we retrieve the link up/down patterns from the trace files of PSN and apply them into our modified ns-2 simulator to make the inter-contact durations of ICN links follow the same power-law distribution.

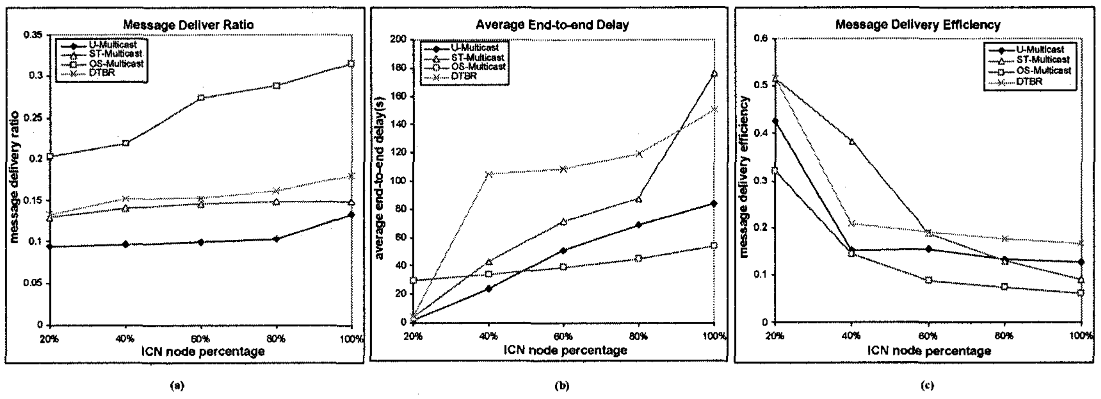

Figure 3.9: Simulation results using the PSN trace.

Fig 3.9 illustrates the results of using the PSN trace. In PSN, wireless devices called iMotes are put into the pockets of the conference attendees at IEEE Infocom 2005. Fig 3.9 shows that all multicast approaches perform much worse in terms of message delivery ratio and end-to-end delays than those shown in Fig 3.7. With the power-law distributed inter-contact durations, it becomes more difficult to deliver bundles when links are repeatedly down for a long period of time. In addition, bundles have to be buffered in the intermediate nodes for longer time to be successfully delivered.

From all these simulations we can conclude that: *i*) DT-Multicast methods should be applied for supporting one-to-many data communication in ICN environments. They perform better than U-Multicast and ST-Multicast, especially when the networking links experience longer inter-contact durations; and *ii*) Comparing to DTBR, OS-Multicast make a trade-off between the message delivery ratio and efficiency. The

Table 3.2: Simulation Parameters With Node Mobility

Parameter	**Value**
Number of nodes	25 (all are ICN nodes)
Area size	2000m × 2000m
Initial Topology	Uniformly randomly distributed
Number of multicast sources	1
Number of multicast receivers	4
Bundle size	512 bytes
Data rate	0.5 ∼ 10bundle/seconds
Bundle retransmission timer	5 seconds
MAC layer protocol	802.11
Transmission range	250m

next section will show that DTBR and OS-Multicast are appropriate for different ICN scenarios.

Results With Node Mobility & Under High Traffic Rate

In practice, ICN nodes like vehicles may move in the network areas rather than being stationary. In this simulation, we apply the mobility patterns retrieved from the ZebraNet trace [89], which records the location information of a group of zebras in Kenya and is regarded as another widely utilized mobility model for studying ICNs. Table II lists the simulation parameters. In this test, all the nodes are configured as ICN nodes and we vary the source traffic rate from 1 bundle per 2 seconds to 10 bundles per second, which is a relatively high traffic rate compared to that in the previous set of simulations. The simulation length is 3,000 seconds and the source only generates bundles during the first 1,000 seconds.

Fig 3.10 shows the results. Not surprisingly, U-Multicast and ST-Multicast deliver very few bundles to the receivers as nodes move, compared to DT-Multicast approaches. Neither can adapt well to the frequent topology variations caused by high node mobility. We also observe that when the traffic rate at the source is high

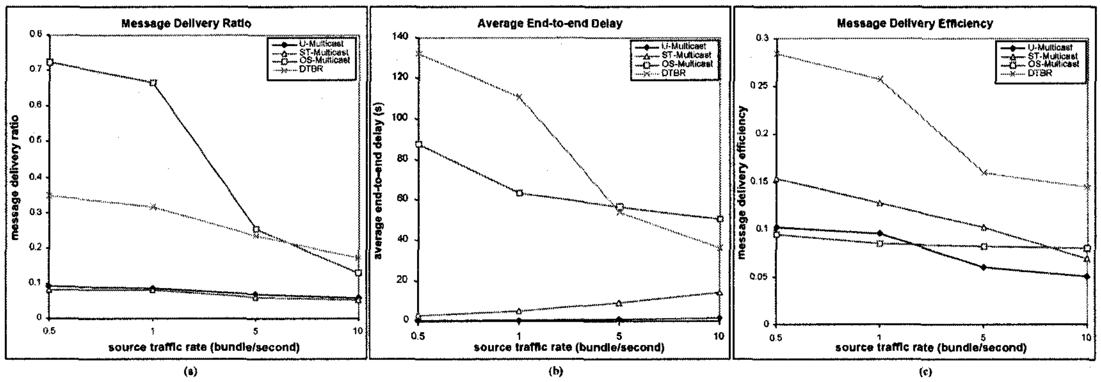

Figure 3.10: Simulation results of varying the source traffic rate with mobile nodes.

(> 5 bundles/second), DTBR outperforms OS-Multicast in all the metrics including the message delivery ratio, efficiency and average end-to-end delays. This is because in OS-Multicast each intermediate node will generate a large amount of redundant traffic to cover all the receivers under high traffic load. They compete with those freshly received bundles for the limited capacity of wireless links to be forwarded to the next hops in the multicast tree. Therefore, fresh bundles have few chances to be forwarded and suffer long queuing delays. This worsens the performance of OS-Multicast. However, when the traffic load is small, the redundancy introduced by OS-Multicast improves its bundle delivery performance because a bundle is considered as delivered when one of its copies reaches each destination. Based on our results, we recommend that system designers select DTBR when the source traffic rate is high to get the best multicast performance.

Other Results

We also investigate the performances of the ICN multicast strategies in the other two cases: stationary nodes with high source traffic rate and mobile nodes with low source traffic rate. The results are shown in Fig. 3.11 and Fig. 3.12 respectively. Both use the same simulation setup as listed in Table II.

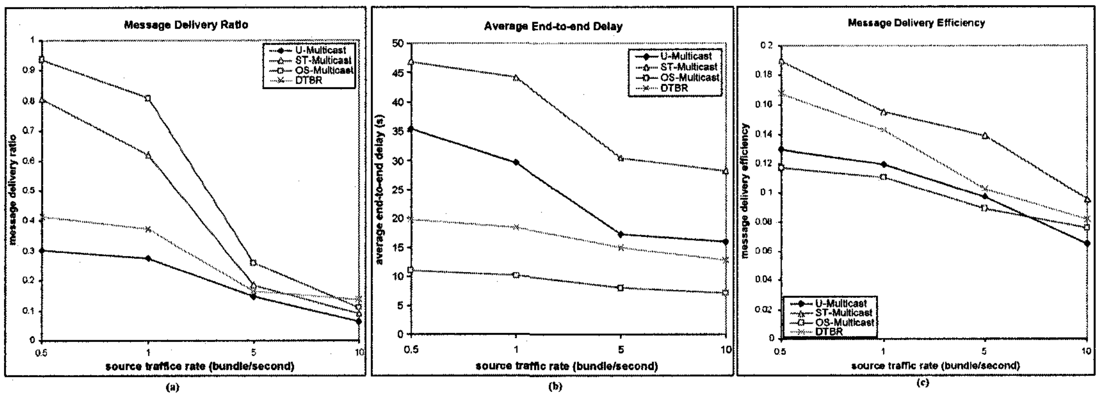

Figure 3.11: Simulation results of varying the source traffic rate with stationary nodes.

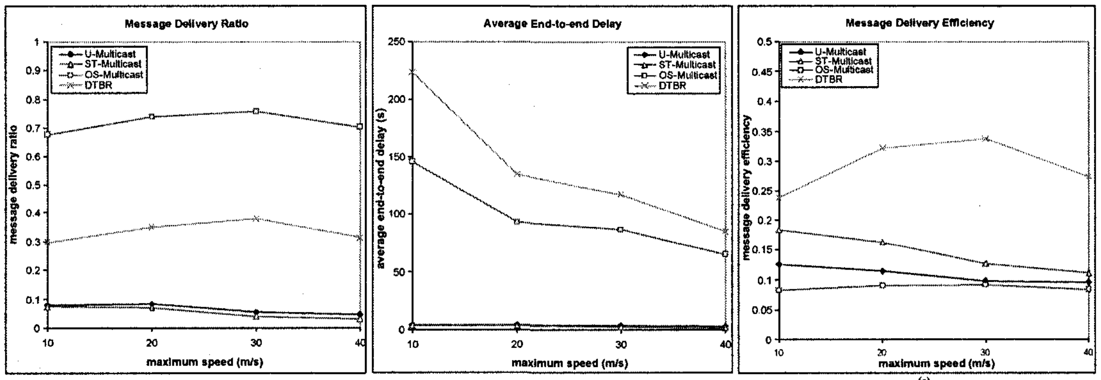

Figure 3.12: Simulation results of varying the maximum moving speed of ICN nodes.

From Fig. 3.11, we get the similar observations as that discussed in Section B: the performance of OS-Multicast drops when the source traffic rate is relatively high. In Fig. 12, we fix the source traffic rate as 1 bundle/2 second but vary the maximum moving speed of ICN nodes from 10m/s to 40m/s. Fig. 3.12 shows that: *i*) the increase of node mobility can help DT-Multicast schemes deliver more bundles to the receivers faster, because it improves the chances of encountering a receiver for each ICN node and reduces the queuing delays; but *ii*) when the node mobility is high, it may worsen the message delivery ratio of DT-Multicast schemes because links in the discovered multicast structure can be more easily broken. How can take advantage of node mobility to assist multicast routing is out of the scope of this paper. Some related discussions about this issue can be found in [58, 99].

3.1.5 Discussion

In this study, we have discussed several multicast strategies including U-Multicast, ST-Multicast, and DT-Multicast that are applicable to disruption tolerant networks. We focus on studying two DT-Multicast routing schemes: DTBR and OS-Multicast, which are able to dynamically adjust the multicast structure in a hop-by-hop manner according to the current network conditions. Performance comparisons among these multicast methods are then done by simulations. Our results show that: *i*) DT-Multicast approaches significantly outperform ST-Multicast and U-Multicast for ICNs, especially when networking links are only up for very short periods of time and nodes may move fast. In most cases, DT-Multicast can achieve the best message delivery ratio with the smallest delays. *ii*) The performances of ICN multicast routing schemes are sensitive to some network configuration parameters, such as the number of ICN nodes within the networks and the local storage size inside each ICN node. In general, we recommend a system designer deploy more ICN nodes with larger buffer to achieve better message delivery ratio. However, there is a trade-off between the performance enhancements and the costs of implementation, deployment, and resources. *iii*) DTBR and OS-Multicast have different advantages. The fundamental design purpose of OS-Multicast is to improve the multicast reliability and the utilization of opportunistic links by introducing more redundancy. The performance gain in terms of message delivery ratio achieved by OS-Multicast comes from the redundant duplicated bundles generated along with the creation of the dynamic multicast structure. Thus, OS-Multicast is more appropriate when the traffic load is small. When the traffic load becomes high, DTBR performs better in terms of the message delivery efficiency.

3.2 Mobile Data Collection in WSNs

3.2.1 Introduction

Wireless Sensor Networks (WSN) consisting of thousands or hundreds of sensors which are able to sense environment data, communicate data by integrated low-power radios, and process the collected data are envisioned to be the building component in many systems. They are expected to be the underlying platform for a broad range of event surveillance applications, such as habitat monitoring [27], civil infrastructure health control [35], health and medical care, and etc.

Sensor networks are event(query)-driven systems. There are multiple data collectors called "data sinks" (or sinks) subscribing to specific data streams and pull the sensed data from the networks. Usually, a sink expresses its interests by broadcasting event quries into the whole network. Only those sensors which detect the interested phenomenon need to report. Sensors are viewed as the data sourcesto push the observed results back to the sinks, via multi-hop wireless data transmission. In practical cases, the sinks could be handheld devices (e.g. PDA) or laptops carried by system administrators, who may move around the sensor nets to perform system management tasks. For example, Bryan Hodgson, the system manager of Lehigh's CSE department, used to deploy some sensors to monitor the temperature of two critical servers in Lehigh's computing cluster to avoid system crash. When the temperature is over certain threshold, Bryan should be notified while he is walking around the building and working on some administrative tasks.

In many other cases, such as the first responder systems [30, 34], there exist practical requirements for the underlying sensor networks to support multiple mobile sinks. In the first responder systems, when a disaster such as the leakage of poisonous chemical gas in the city suddenly happens, a team of emergency rescue experts must rush to the scene to do the damage control and protect the safety of people. Wireless sensor networks could be deployed to monitor the health conditions of victims or certain ingredient in the air. Those critical medical data and environment information must be relayed to the first responders in time, who are moving around the area to help each victim. The one-to-many or many-to-many data communication from stationary sensors to a group of mobile sinks is defined as mobile data collection or mobile multicast in WSNs.

In the mobile data collection problem, the moving sinks first advertise their interests (event queries) to all sensors. For example, sinks may concern about a spatiotemporal event of whether the temperature exceed 110 degree in certain area from 9am to 5pm. Sensors detect that event will report their observations to the mobile sinks, until either the event is over, or the lifetime of the query is expired, or a new query is received to replace the old one. To simplify our algorithm design without losing the correctness, we assume that multiple sinks are interested in the same type of events in the sensor nets.

In this study, we propose Track and Transmit (TNT) which is able to effectively track the movement of mobile sinks and efficiently deliver the data from the sensor nets to the mobile sinks. We also propose Priced Track and Transmit (PTNT) to utilize better (shorter) data forwarding path than TNT. Both approaches are distributed methods and easy to be implemented. They only require each sensor in the networks to forward data based on its own view of the sink's movement. Moreover, TNT and PTNT can significantly suppress the control overheads of querying a sensor net compare to another existing mobile multicast approach VLM2 (Very Lightweight Mobile Multicast), as well as achieve better delivery ratio and acceptable transmission delays.

3.2.2 TNT: Track and Transmit

System Overview

We view a wireless sensor network as a distributed system consisting of lots of ad hoc deployed sensors. Each sensor is equipped with the same type of short-range radio, using the same wireless channel. The link between any pair of connected sensors is bidirectional, i.e., the neighboring nodes can communicate with each other. We also assume that each individual sensor has exactly the same capabilities and functionalities. There isn't any special sensor with more data processing capability or more powerful antenna in our system. That means, TNT and PTNT do not require any central points or cluster headers. The whole network is constructed as a flat structure and no hierarchy is needed. The data sinks are handheld devices equipped with the same radio as wireless sensors. They are carried by humans (administrators or first responders) to query and collect the event observations from the sensor nets. Except the moveable sinks, regular sensors won't change their positions after the initial deployment. The overall sensor nets are well synchronized by time synchronization approaches such as TPSN [42] or FTSP [64]. And we also assume the network is always connected.

Sensors which detect an event that matches sink's queries will send an event report to the mobile sink. These nodes are named as active sensor in this study. A simple scenario of mobile multicast is depicted in Fig. 3.13. To make our further investigation simple and clear, we study the performances of TNT and PTNT using WSNs with gird topology.

Tracking a Mobile Sink

The task of tracking a mobile sink needs the collaborations of the whole sensor networks. The basic idea of TNT is based on a fact of that a moving sink will not

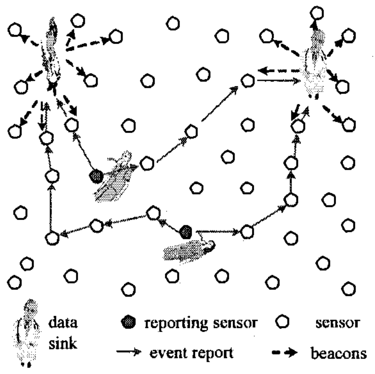

Figure 3.13: An example of mobile multicast.

just jump from one position to another but moves continuously and leaves a trace of its movement in the networks. If we view the whole sensor nets as a time-variable system, the beacons periodically broadcasted by the sink will stamp its movement trace in the networks. For example, node A receives a beacon at time t_1 and node B receives another beacon at time t_2 from the same sink. It implies that during time period $[t_1, t_2]$, the sink somehow moves from A's neighborhood to B's neighborhood.

In TNT, we require each sensor to set up a beacon timer, which is synchronized with the beacon frequency of the mobile sink. Each sensor also maintains a tracking counter, which value is adjusted by its internal beacon timer as the followings:

- When a mobile sink moving around the networks, it does not flood beacons throughout the whole networks but only broadcasts beacons to its current neighbors. A node which receives a new beacon will increase the value of its tracking counter by 1 and resets its beacon timer;

- If the beacon timer got expired and a node does not receive any beacon message, the tracking counter will be decreased by 1. Also, this node will reset the beacon timer for the next period;

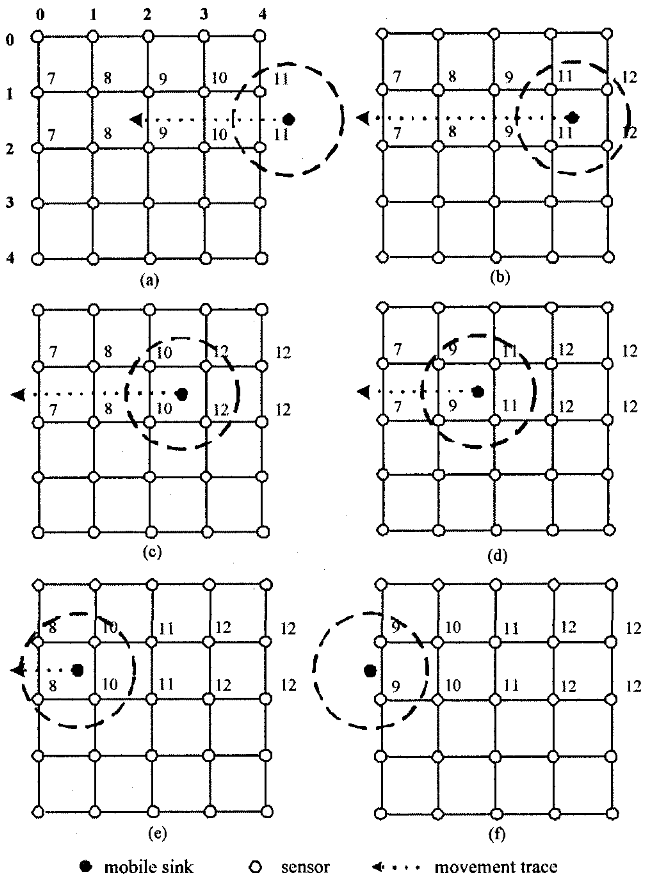

Figure 3.14: An example of how to record mobile sink's movement trace by using the tracking counter information.

- The first beacon of a mobile sink will always be flooded throughout the network, to get the beacon timer of each node synchronized.

By making each node maintain the tracking counter information, we transfer the whole sensor nets into a large database, in which the movement history of a mobile sink is recorded. Fig. 3.14. shows an example of how the tracking counters work a for 5×5 grid network.

In Fig. 3.14, a mobile sink moves through the network from right to left, at a fixed speed. We assume that the initial value of the tracking counter is 10. Cases (a) to (f) illustrate the variations of the tracking counter in this scenario, caused by the movement of the sink. Basically, nodes in row 1 and row 2 will increase their tracking counters when the sink moves around and decrease them when the sink is moving away. Fig. 3.15 shows the details of case (d).

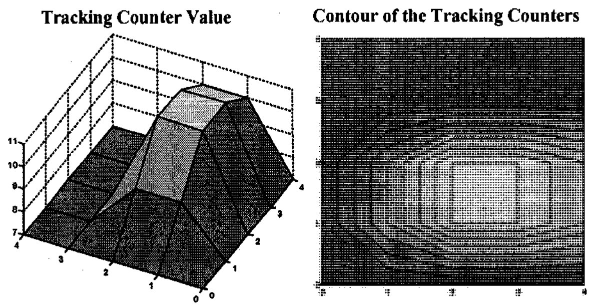

Figure 3.15: Details of tracking counters for case (d).

It is obvious that: *i*) the existence of a mobile sink will cause an increasing trend of the tracking counters, which incurs a peak as shown in Fig.3. This gives us a way to locate the mobile sink by searching along the increasing directions of the tracking counters; and *ii*) Tracking counters in the networks vary continuously along the time line. If we carefully observe the variations of tracking counters step by step, there won't be a peak suddenly occur in one place and then jump to another at the next time. It implies that the moving sink is traceable since its movement is continuous. With these facts in mind, a heuristic data forwarding strategy could be designed to chase the moving sink by looking for the peaks of tracking counters.

Data Forwarding Strategy

In the design of TNT, an identical sequence number is associated with each beacon message sent from the mobile sink to distinguish how fresh a beacon is. TNT does not require each sensor keep the records of all its received beacons but only the freshest one that it receives, i.e., the last time when it encounters the moving sink. Then the sensor's observations can be forwarded in the following way.

Assume node A detects an event that matches the sink's query. Node A then broadcasts its observations to the neighbors along with the following information: the value of A's current tracking counter, the sequence number of the latest beacon it received, and the receiving time of that beacon. When node B, one of node A's one-hop-away neighbors, receives the data packet, it will decide whether it's going to forward the packet or not. Table 3.3 describes the details of the forwarding heuristics in TNT.

Table 3.3: Routing Strategy of TNT

Case	**Tracking Counter**	**Beacon SN**	**Receiving Time**	**Forward or Not**
1	$B > A$	$\cdots$	$\cdots$	Yes
2	$B <= A$	$B > A$	$\cdots$	Yes
3	$B <= A$	$B = A$	$B < A$	Yes
4	$B <= A$	$B < A$	$\cdots$	No

Recall that in TNT each node makes the forwarding decision based on its current knowledge of the sink's movement. Case 1 in table 3.3 represents that, if node B has a bigger tracking counter than A, then in history the mobile sink used to stay longer around B than A. Thus, B forwards A's packet. In case 2, if node A has a bigger tracking counter but the latest beacon received by B is fresher than A, then B also forwards A's packet. That could be the case in which the mobile sink is moving away from A to B, like the sink is moving toward node (1, 3) from node (1, 4) in Fig. 3.15 (b). Case 3 tells that if B used to be nearer to the mobile sink than A (since B received the same beacon earlier), then B forwards A's data. In other cases, B will drop the data packet from A.

Following these heuristics, Fig. 3.16 shows how TNT forwards the observations from node (4, 4) to the current position of the moving sink. TNT can effectively track the sink's movement and successfully route the data to the mobile sink.

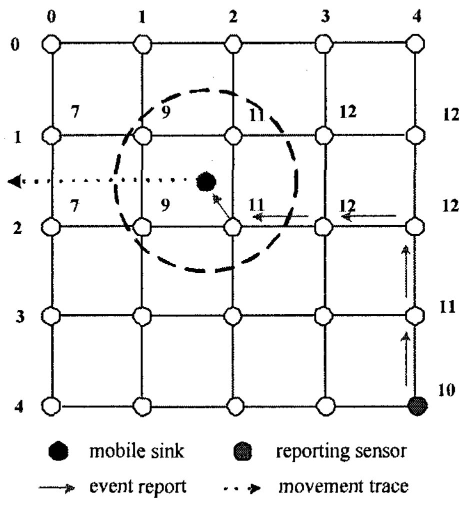

Figure 3.16: An example of data transmission from the sensors to the mobile data sink

3.2.3 Priced TNT: An Improvement of TNT

Issues of TNT

The basic idea of TNT is to transmit sensor's reports along the sink's historical movement trace recorded by the sensor networks. Compared to VLM2 (introduced in Chapter 2), an existing mobile multicast approach for WSN, TNT dramatically suppresses the control overheads of tracking mobile sinks, because TNT does not need to flood each beacon throughout the whole networks. The beacons are only required to be transmitted to the current neighborhoods of the moving sink.

However, TNT suffers longer transmission delays than VLM2, because VLM2 always selects the current shortest paths in the networks. There are also some cases in which TNT does not perform efficiently due to the special movement patterns of the sink.

- The inaccurate peak issue: in TNT, the event reports would be forwarded along the direction where the values of sensor's tracking counter become larger and larger. This reason is that TNT tries to assign more positive values to the current neighbors of the mobile sink while punishing those sensors that do not detect the existence of the sink. However, this greedy strategy may mislead the data forwarding into a wrong direction. An example is shown in Fig. 3.17. The dot line shows the movement path of a mobile sink. After moving around position A for a while, the sink moves across the sensor nets to a new position B. This will leave a tracking counter peak around node (0, 0), (0, 1), (1, 0) and (1, 1), but in fact the sink is now near to node (4, 4), which currently has a smaller tracking counter. Suppose node (2, 2) observes an event and has some data to be sent to the sink. By TNT, some data will be forwarded along the dash line, which follows the movement trace of the sink but is obviously not the best path represented by the real, arrowed line.

- The U-shape trace issue: TNT tries to record and follow the movement trace of a mobile sink. That incurs another issue of the routing efficiency: what if a sink moves along a U-shape trace around the sensor nets. For example in Fig. 3.18, the mobile sink moves from position A to B. By TNT, the data of node (2, 2) would be forwarded using the dash line, which is 11-hops long (node (1, 2) forwards the data because it matches the case 3 in 3.3). But it is obvious that

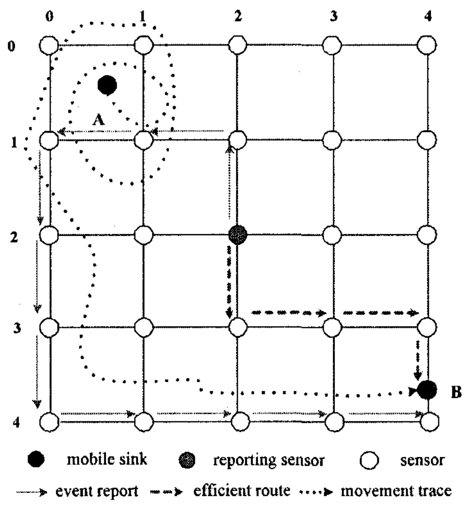

Figure 3.17: An example of the inaccurate peak issue of TNT

the optimal path to relay the data is to follow the solid arrowed line, which is only 3-hops long. In this special case, TNT won't be make node (3, 2) forward the data because its tracking counter keeps decreasing when the sink moves.

VLM2 does not have those issues since it keeps updating the distance from each sensor to the mobile sink. We then combine the benefits of both TNT and VLM2 and propose Priced Track and Transmit (PTNT) to improve the performance of TNT.

The Idea of PTNT

PTNT puts more control information to find better and shorter data forwarding path than TNT. It uses the hop count information as the forwarding price for each sensor node. The forwarding prices are dynamically changed according to the current position of a mobile sink.

Unlike TNT that only floods the first beacon to get all the sensors synchronized, PTNT requires that the mobile sink periodically floods an update beacon throughout

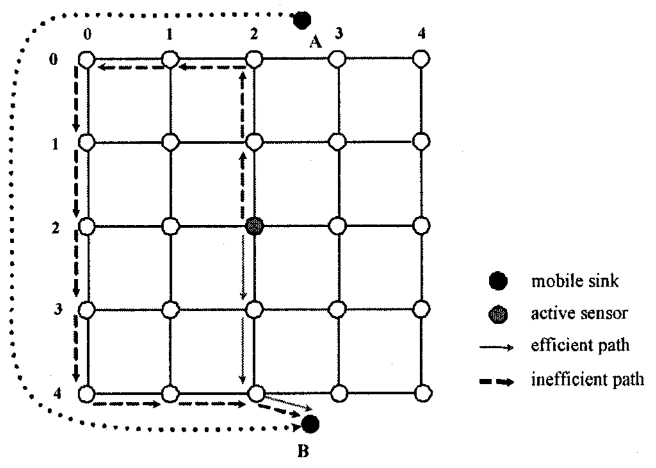

Figure 3.18: An example of the U-shape trace issue of TNT

the whole networks. For example, an update beacon is flooded every ten regular beacons. This special beacon is utilized to refresh the hop counts from sensors to the mobile sink, with the same functionality of beacons in VLM2. Each sensor then takes the current hop count information as the forwarding price to further control its data forwarding behavior.

Besides the tracking counter, the sequence number and receiving time of the latest received beacon, and the hop count information, PTNT also requires that each node maintains a detected flag. The flag detected is set as 1 when a node just receives a beacon before its current tracking timer is expired. Otherwise, the flag detected is 0. The improved forwarding strategy of PTNT is shown in table 3.4.

When node B receives a data packet from node A, if B's detected flag is 1, i.e., the sink is currently one-hop away from node B, then node B will forward the packet at once. Otherwise, node B is going to compare its tracking counter to node A, with the consideration of the forwarding prices. Case 2 and 4 tells that whenever node B has a cheaper forwarding price, node B will forward the packet even if it may have

Table 3.4: Improved Routing Strategy in PTNT

Case	Detected	Tracking Count	Hop Count	Beacon SN	Receiving Time	Forward or Not
1	detected= 1	$\cdots$	$\cdots$	$\cdots$	$\cdots$	Yes
2	detected= 0	$B > A$	$B < A$	$\cdots$	$\cdots$	Yes
3	detected= 0	$B > A$	$B \geq A$	$\cdots$	$\cdots$	No
4	detected= 0	$B \leq A$	$B < A$	$\cdots$	$\cdots$	Yes
5	detected= 0	$B \leq A$	$B \geq A$	$B > A$	$\cdots$	Yes
6	detected= 0	$B \leq A$	$B \geq A$	$B = A$	$B < A$	Yes
7	detected= 0	$B \leq A$	$B \geq A$	$B = A$	$B \geq A$	No
8	detected= 0	$B \leq A$	$B \geq A$	$B < A$	$\cdots$	No

a smaller tracking counter. Case 5 and 6 represents the special cases in which node B has worse tracking counter and longer distance to the sink than node A, but with fresher beacon information. In PTNT, node B will forward the packet because the sink may just move around node B yet a new update beacon has not been flooded to A. Whenever the tracking counters become unavailable or inaccurate due to certain reasons, PTNT will follow the current forwarding price to transmit data. Thus, at least PTNT acts similar as VLM2. This is verified by our simulation results presented in the next section. By applying the idea of PTNT, the inefficient data forwarding issues of TNT could be solved. The pseudo code of PTNT is depicted in Fig. 3.19.

PTNT improves the data forwarding efficiency by using the hop count information to filter the selected forwarding paths. Clearly, if the update beacons are flooded more frequently, the distance information to the mobile sink would become more accurate. An extreme case is that, the sink broadcasts every beacon message throughout the networks and keeps updating the hop counts in real-time. This is exactly how VLM2 works. Theoretically, it can achieve the best forwarding efficiency since the data will be transmitted using the shortest path. However, too much flooding will waste a lot of broadcast energy and bandwidth, which are valuable resources for WSNs. By adjusting the frequency of flooding the update beacons, PTNT is able to make a good

```
ForwardPackets (packet)
{
   if ( this.up = 1){
      ResetHeader(packet,this);
      return true;
   }
   if ( this.trackcount > packet.header.trackcount
        and this.hopcount < packet.header.hopcount){
      ResetHeader(packet,this);
      return true;
   }
   if ( this.trackcount <= packet.header.trackcount
        and this.hopcount < packet.header.hopcount){
      ResetHeader(packet,this);
      return true;
   }
   if ( this.trackcount <= packet.header.trackcount
        and this.hopcount >= packet.header.hopcount){
      if ( this.beaconSN > packet.header.beaconSN){
         ResetHeader(packet,this);
         return true;
      }
      if ( this.beaconSN = packet.header.beaconSN
           and this.recvTime < packet.header.recvTime){
         ResetHeader(packet,this);
         return true;
      }
   }
   return false;
}

ResetHeader (packet, s)
{
   packet.header.trackcount = this.trackcount;
   packet.header.hopcount = this.hopcount;
   packet.header.beaconSN = this.beaconSN;
   packet.header.recvTime = this.recvTime;
}
```

Figure 3.19: Pseudo codes of PTNT

trade-off between the data forwarding efficiency and the system resource usage.

3.2.4 Mobile Multicast Using TNT/PTNT

The above sections introduce the ideas of TNT and PTNT about how to track and transmit the event reports to a single mobile sink. These ideas can be easily extended for the one-to-many and many-to-many data communication of mobile multicast.

Consider a group of mobile sinks collecting data from a sensor network. Each sink will announce its interests of events that may happen in the networks, by broadcasting a query to all the sensors. Sinks have the same interests will register in the same group. We assume each mobile sink already know the group it belongs to. For instance, two first responders may have the same concern about the heart beat rates of patients in a large area. Both of them want to be notified when a patient's heart beat becomes abnormal. They both know that they need to subscribe to the mobile multicast service of the patient's heart beat reports, from their tasks. In a simple word, the mobile sinks will join or leave a group explicitly, by putting its ID and the group ID with the queries.

After receiving the queries, each sensor will maintain a service table containing group information and assign a tracking counter to each sink. The data sources (active sensors) subscribe to the mobile multicast group in an implicit way, by starting to send out event reports to that group. When an event happens, active sensors take the group ID registered to this event as the destination ID and broadcasts a series of event reports out. Each node receives the data will dynamically decide whether to forward the report or not based on its current knowledge of the sink's movement trace. If not, the data will be discarded. Eventually, the data arrives at those mobile sinks which intend to receive the associated events. When the interested event is over or the lifetime of sink's query is expired, the data sources will leave the multicast group and the mobile multicast session will be over automatically.

3.2.5 Performance Evaluation

Simulation Results

To evaluate the performance of TNT, PTNT and VLM2 in this study, we implement them in GloMoSim [2] network simulator. Our performance metrics include *i*) data delivery ratio, which is defined as the unique event reports successfully arrive at the mobile sink over the total reports generated at the sources; *ii*) mobile multicast efficiency, which include two sub-metrics: control overhead ratio and control bytes ratio. The former is the ratio between the total number of beacons sent by the mobile sink and the total number of all the packets generated during the period of one mobile multicast session. The latter is the ratio between the total bytes of control messages including the size of beacons and the sizes of control headers of event reports and the total bytes of the payloads of event reports; and *iii*) average transmission delay, which is the average of the source-to-sink delays for all the sensor's observations.

All simulations have 51 nodes deployed in a 400x400 area: 2 mobile sinks plus 49 stationary sensors. The topology is a grid network built up in a flat structure. All the sensors have implemented the same functionalities. To represent the movement of human (like system administrators or first responders), we set moving speed of the sinks to be between 1m/s and 6m/s (13mph). Each sink has the same radio as the sensor nodes. And the transmission range is set to be 50 meters.

We simulated four scenarios with different movement patterns of the mobile sinks: non-movement, round-robin movement, back-and-forth movement, and random way-point movement, shown in Fig. 3.20. In each scenario, two mobile sinks move into the sensor nets from (0, 0) and (6, 6) respectively. Both sinks are registered in the same mobile multicast group and broadcast one beacon per second. Node (2, 5) and (5, 4) are selected as the data sources. The data generating rate is 2 reports per second. For PTNT, we vary its flooding frequency of update beacons from once per 300 normal

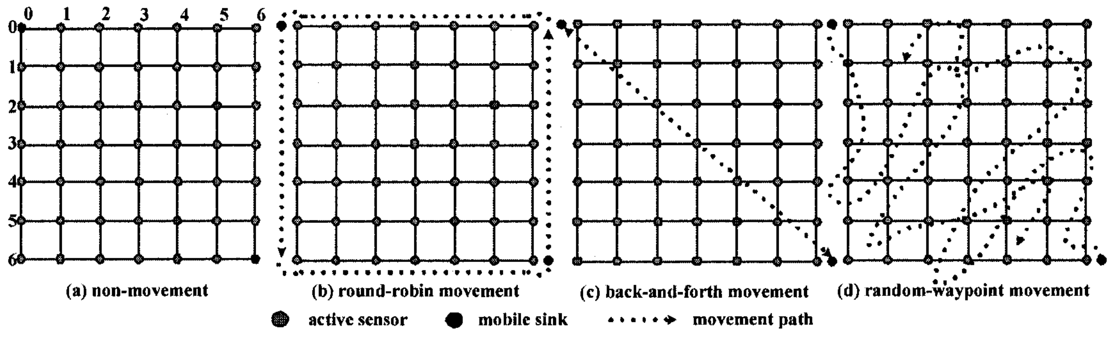

Figure 3.20: Simulation scenarios: different movement patterns of mobile sinks

beacons to once per each beacon. The total simulation time is 300 seconds.

We first illustrate the results of the random waypoint scenario which represents the most general case of mobile multicast in WSN. The other three scenarios are special movement patterns and will be discussed later. In random waypoint scenario, the mobile sinks randomly pick up and move to a destination in the networks. After reaching the destination, sinks may stay a random period of time before it moves to a new place. To eliminate the impacts of different random movement paths, we run simulations 10 times with 10 different sets of waypoints. The final results are the average and are shown in Fig. 9.

In Fig. 3.21(a), it's easy to tell that the data delivery ratios of TNT and PTNT are slightly better than VLM2. The reason is that both TNT and PTNT only selectively flood beacons into the networks, rather than flooding each control message by VLM2. Thus, TNT and PTNT reduce the occurrences of contentions. The high delivery ratios of these two approaches also prove that they are able to effectively track the movement of mobile sinks and forward the data.

Fig. 3.21(b) depicts the average data transmission delay of three approaches. VLM2 has the best performance because it always tries to transmit data along the

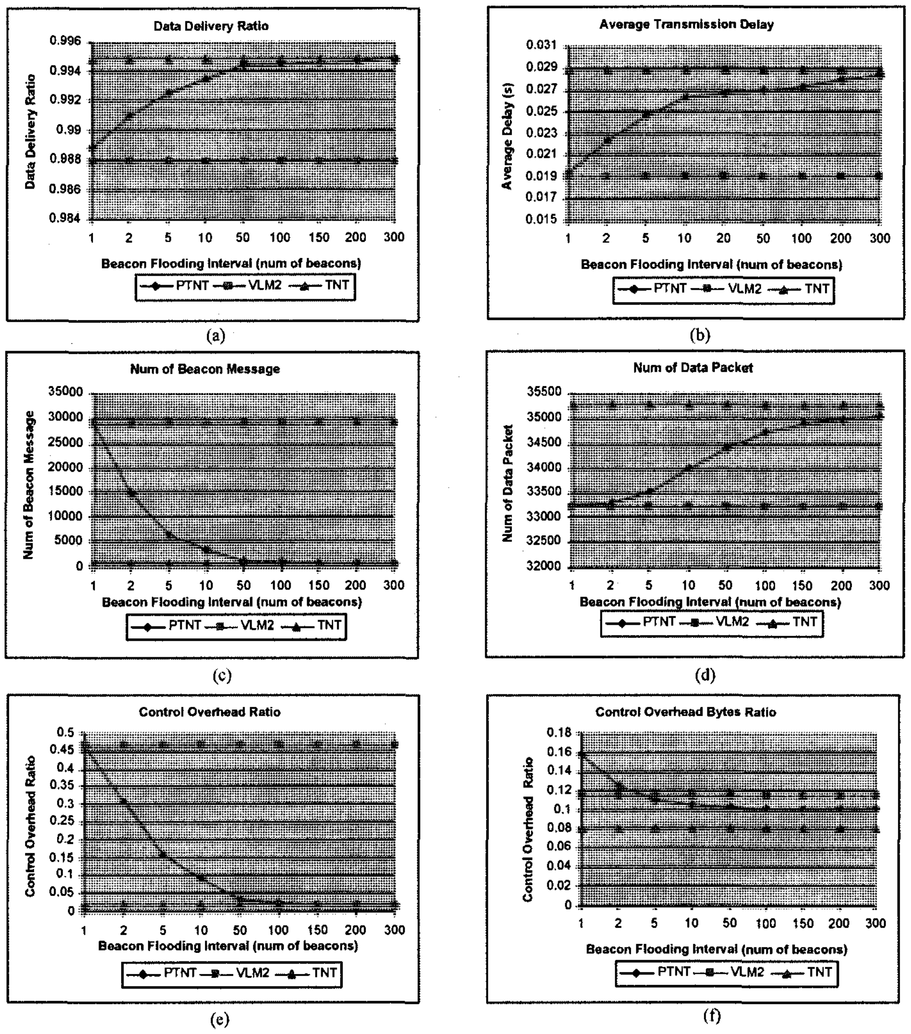

Figure 3.21: Simulation results of the random waypoint scenario

shortest paths. TNT suffers from larger end-to-end delays due to the routing efficiency issues discussed in section 4.

However, PTNT shows the capability of adjusting its performance based on different configurations of the update beacon frequency. It achieves the same performance as VLM2 when it floods all the beacons. When update beacons are generated and broadcasted infrequently, the distance information in each node may be outdated and it has to depend on the tracking counter to decide the data forwarding behaviors. Thus, its performance degrades as similar as TNT. However, Fig. 3.21(e) shows that both TNT and PTNT have much smaller control overhead ratio than VLM2. Although TNT will make more nodes forward data due to its inefficient routing capability (i.e., TNT may take a longer path since it always follows the movement trace of mobile sinks), it only needs to flood the first control beacon. That's why its control overhead ratio is so small. For PTNT, by combining the results of Fig. 3.21(c) and Fig. 3.21(d), it depicts the nice capability of making a trade-off between the control overheads and the data forwarding efficiency. With flooding more update beacons, PTNT can find shorter path than TNT. That means the performance of PTNT is tuneable.

We also illustrate the results of control bytes ratio in Fig. 3.21(f). The header of VLM2 packets is set to be 16 bytes. We implemented the control header of TNT and PTNT as large as 16 bytes and 20 bytes respectively. TNT still has the smallest control bytes ratio. And in most cases, PTNT performs better than VLM2. However, PTNT has larger control overheads in terms of bytes only when it floods more update beacons frequently (e.g., in our simulation flood once per 5 beacons).

In summary, both TNT and PTNT have better data delivery ratios and smaller control overheads than VLM2. TNT has larger end-to-end transmission delays because it may take longer data forwarding paths. However, the performance of PTNT can be tuned by adjusting the flooding frequency of update beacons. We believe that the optimal frequency is highly related to the speed of mobile sinks. The faster the sinks move, the more frequently the forwarding price of each node should be updated.

3.2.6 Analysis of Existing Mobile Data Collection Approaches

Review of the Existing Approaches

In past, several approaches have been proposed to provide the data communication service of using mobile data sinks to collect event reports from stationary sensors. In general, we divide them into two categories: *back-bone based* approach and *flat* approaches according to their routing structure. In the first category, a virtual overlay called routing back-bone is constructed upon the original networks. The shape of the backbone is either a grid or a tree according to different protocol designs. Event queries from mobile sinks and event reports from sensors are all forwarded alone the back-bone before they can be eventually delivered to the destinations. The flat mobile data collection approaches do not have a virtual routing overlay set up in the networks. They view the deployed sensors as a big flat graph. Routing is decided in pure *ad hoc* way, i.e., each sensor has the same functionality to make the routing decision. Our proposed TNT/PTNT approaches belong to this category. The features of the existing mobile data collection approaches are discussed in Table 3.5 and Table 3.6.

Analysis of Simplified Approaches Based on the characteristics of the existing mobile data collection approaches, we give two simplified versions of them. *i*) *Simplified Flat approach*, which always floods the event queries from the mobile sink to the overall networks and always deliver event reports to the mobile sink using the shortest path routing algorithm. *ii*) *Simplified Grid-backbone-based approach*, which divides the entire networks into several sub grids and have all the grid leaders form the virtual backbone. Sink only needs to broadcast the event queries to the leader of the grid it is currently located in. And the grid leader will further forward the query to the sensors along the backbone. Event reports are then forwarded in the reversely using the shortest path routing algorithm.

Table 3.5: Backbone-based Mobile Data Collection Approaches

	Backbone-based Approaches			
	TTDD[95]	SEAD[55]	COMB-NEEDLE[61]	CODE[93]
Assumptions	Stationary sensors and Mobile sink			
Position Info	Yes	Yes	No	Yes
Source:Sink	1:M	1:M	M:1	1:M
Structure	Grid Backbone	Tree Backbone	Comb Backbone	Grid Backbone
Routing	Reverse Shortest Path	Geographic Forwarding	Controlled Flooding	Geographic Forwarding
Tracking	Mobile IP like approach	Mobile IP like approach	Using COMB that moves along with mobile sink	Mobile IP like approach

Table 3.6: Flat Mobile Data Collection Approaches

	Flat Approaches			
	MobiRoute[54]	SAFE[56]	DD[79]	VLM2[77]
Assumptions	Stationary sensors and Mobile sink			
Position Info	No	Yes	No	No
Source:Sink	M:1	1:M	1:1	M:1
Structure	Flat graph	Tree graph	Flat	Flat graph
Routing	Controlled Flooding using different metrics			
Tracking	Mobile IP like approach	Dynamic Tree constructed along with the mobile sink	Gradient set up by controlled flooding	Hop count set up by controlled flooding

To simplified our analysis we assume that:

- the overall networks are a $N \times N$ grid, i.e., in total there are N^2 number of sensors;
- the size of each sub grid is $a \times a$;
- let M represent the scale of sub grids, and $M = \frac{N}{a}$;
- the distance and communication range of each sensor is 1 unit distance;
- there is one mobile data sink moving within the networks with the average speed of V unit distance per second;
- there is one data source located at the coordinator $(N-1, N-1)$;
- the sink broadcasts event queries at the frequency of f_q number of query packets per second. The size of each query packet is S_q bytes;
- the data generation rate at the source is f_d number of data packets per second. The size of each data packet is S_d bytes;

Then we have:

Theroem1: When $M > 1$, using the Simplified Grid-backbone-based approach generates smaller expected traffic of event queries per second than the Simplified Flat approach.

Proof:

The expected traffic of event queries generated using the Simplified flat approach in one second can be calculated as:

$$E(C_{qFlat}) = f_q \times N^2$$

And that using the Simplified Grid-backbone based approach can be calculated as:

$$E(C_{qBackbone}) = f_q \times [a^2 + \frac{2N-2a+\cdots+M\times(2N-2a-(M-1)a)}{M^2} + (2a-2)]$$

$$E(C_{qBackbone}) = f_q \times (a^2 + a + \frac{2N}{3} - 2)$$

Thus:

$$\frac{E(C_{qFlat})}{E(C_{qBackbone})} = \frac{N^2}{a^2+a+\frac{2N}{3}-2)} = 1 + \frac{3(N-1)(N-2)-3a(a-1)+7N}{3a^2+3a+2N-6}$$

Clearly when $a \leq N-1 \Leftrightarrow M > 1, \frac{E(C_{qFlat})}{E(C_{qBackbone})} > 1$ because $N > 0$. ■

Theroem2: When $M > 3$, using the simplified Grid-backbone-based approach generates smaller expected traffic of event reports per second than the Simplified Flat approach.

Proof:

The expected traffic of event reports generated using the Simplified flat approach in one second can be calculated as:

$$E(C_{dFlat}) = f_d \times [\frac{(2N-2)+2\times(2N-2)+\cdots+N\times(2N-2)}{N^2} + V]$$

$$E(C_{dFlat}) = f_d \times [N + V - \frac{1}{N}]$$

And that of using the Simplified Grid-backbone based approach can be calculated as:

$$E(C_{dBackbone}) = f_d \times [(2a-2) + \frac{(2N-2a)+\cdots+M(2N-2a)}{M^2} + \frac{a^2-1}{a} + V - a]$$

$$E(C_{dBackbone}) = f_d \times [3a + \frac{a^2}{3N} = \frac{2N}{3} - 2 + V]$$

Therefore:

$$E(C_{dFlat}) - E(C_{dBackbone}) = f_d \times \frac{N^2+6N-9aN-a^2}{3N} > f_d \times \frac{M^2-3M-1}{3N}$$

Clearly, the above equation is greater than 0 when $M > 3$. ■

3.2.7 Discussion

In this study, we propose the idea of TNT and PTNT, which are lightweight approaches for supporting mobile data collection in WSNs. TNT effectively tracks the moving sinks by requiring sensor networks to collaboratively record the sinks' movement traces. The event reports from sensors are then forwarded along the discovered traces. PTNT improves the data forwarding efficiency of TNT by taking advantage of the shortest path information from sensors to mobile sinks as the forwarding price. Simulations show that both TNT and PTNT are able to suppress the control overheads and achieve better data delivery ratio and acceptable message transmission delays, as compared to VLM2. Moreover, PTNT is capable of making a trade-off between the data forwarding efficiency and the average message delivery delay by controlling how frequent the beacon messages are flooded in the overall networks. When every beacon is flooded, PTNT will always select the shortest path with achieving the smallest delay but the worst efficiency.

Chapter 4

Time Synchronization in WASNs

4.1 DTP: Double-pairwise Time Protocol for Disruption Tolerant Networks

4.1.1 Introduction

Recent research on Intermittently Connected Networks (ICNs) [36, 49, 39, 40, 100, 19], addresses challenges of handling data communications in wireless networks where continuous end-to-end connectivity can not be guaranteed. Planned or unplanned link disconnections in communication environments of ICNs may result in frequent network partitions. Such challenges are caused by high node mobility, low network density, scheduled node unavailability, or unexpected infrastructure disruptions. These challenges may happen in different ICN scenarios, for instance, the sparse mobile ad hoc networks, the interplanetary networks and etc. ICN technologies are viewed as building components to support a broad range of applications such as military battlefields [63], deep-space communications [10] and Internet access in rural areas [51].

Network time synchronization (NTS) is a critical service for any distributed systems, including disruption tolerant networks. In general, a computer clock does not perfectly follow the Coordinated Universal Time (UTC) due to the inaccurate vibration frequency of its internal oscillator, the significant changes of its surrounding environment, or the false configurations of its operating system. The issue of imprecise timing becomes severe in distributed systems because different clocks may tick at different speeds from the beginning and such differences will deteriorate over time and distance [45]. We denote the difference between the reading of a computer clock and the UTC as the *time error*. Thus, the fundamental task of NTS is to keep clocks in the same network reporting time with the smallest achievable time errors when required, which are ideally zero.

NTS service has a wide range of applications in ICNs. It helps monitor the availability and durations of ICN links by logging events with accurate time stamps. It also helps achieve collaborations by activating distributed components simultaneously. Moreover, it helps facilitate network security by providing the synchronized global time as a part of the encryption keys [68].

The Global Positioning System (GPS) is a hardware solution to providing the NTS service. A GPS device must continuously receive time pulses from multiple satellites which have expensive cesium oscillators to minimize the time error of the satellite host. However, GPS devices do not function well in the indoor environments and suffers from performance degradation when there are much rain/snow noises.

The Network Time Protocol (NTP) [66, 67], as a software approach, is the timekeeping standard in the Internet. NTP enables synchronization in a peer-to-peer manner between the clients and servers. It establishes a hierarchical architecture to synchronize clocks within an internetworking system with the so-called primary/secondary time servers. The clocks of these servers are hereby adjusted by the GPS or other trusted time references. NTP is capable of achieving time errors in a range of hundreds of milliseconds when the synchronization operations can be conducted continuously and frequently. Section IV provides detailed descriptions and performance analysis of NTP-Core, which represents the core synchronization message exchange model of NTP.

There is a challenge when directly apply NTP in ICNs: the synchronization operations have to be discontinued when the end-to-end path between the time synchronization peers is broken. When such link disruption happens, the time client must rely on the latest measurement results collected from previous synchronization operations to adjust its local clock, until it can be reconnected to the time server. In some practical scenarios (e.g. the interplanetary networks), it may take hours or days to recover the disrupted ICN links. Therefore, it requires that any NTS approach proposed for ICNs must be able to tolerate the long unavailability of links while achieving the time error as small as possible.

To address this challenge, we propose DTP (Double-pairwise Time Protocol), which is a new software approach to providing the NTS service in ICNs. DTP has the following characteristics: *i*) it achieves approximately half of the time error of NTP-Core (discuess in Section 4.1.3), by utilizing a novel message filter to process time synchronization messages. *ii*) DTP is an extension of NTP and it can be directly deployed in the current NTP's system architecture.

4.1.2 System Models

The Network Model

ICNs are composed of a set of communication hosts that are capable of transmitting messages via the physical links between them in a store-and-forward manner. We view ICNs as a time varying graph $G(t) = (N, E(t))$, where N is the set of nodes and $E(t)$ is the set of the current available links at real time t. Let A_i represent the ith node. Then $N = A_0, A_1, \ldots, A_{n-1}$ and $|N| = n$. A directed link e_{ij} from A_i to A_j exists only when A_j is within the communication range of A_i. We assume that each ICN node has the same data communication capability. Thus, $e_{ji} \in E(t)$ when $e_{ij} \in E(t)$. e_{ji} and e_{ij} are symmetric with the same communication range of R. Note that, $E(t)$ varies along time due to the frequent up or down status changes of links in ICNs. Among all the nodes, we assume that there is at least one node, A_0, has the correct real time. And we denote A_0 as the time server in G. Similar network model can be found in [51, 100]

The Clock Model

Every ICN node has a hardware clock which reads the value from a counter maintained by a timing circuit. The heart of this circuit is a quartz oscillator stimulated by the piezoelectric effect. The accuracy of the oscillator is determined by the size, thickness, and cutting angle of its internal quartz plate as well as the conditions of the surrounding environments, especially the temperature. A standalone computer clock over the real time can be typically regarded as a linear function:

$$Clock_i(t) = R_i \times t + t_{0i} \tag{4.1}$$

where $Clock_i$ is the reading of A_i's clock at real time t, R_i is called the *clock drift* which represents how fast A_i's clock deteriorate away from the UTC, and t_{0i} is the *initial clock offset* which stands for the time error of A_i when system starts. Similar clock models are also proposed in [66, 20, 82].

Thus, A_i's clock can be adjusted correctly if its clock drift and clock offset can be accurately measured, . We regard A_i's clock as perfectly following the UTC with R_i = 1 and $t_{0i} = 0$. In general, R_i is within a linear envelope of the UTC: $|R_i - 1| \leq p$. And the most common value of p of a modern crystal oscillator is between 0 $\sim$ 50ppm (part per million) following the Gaussian distribution [1].

The Delay Model

All the NTS approaches conduct synchronization operations by exchanging time messages between networking nodes to measure their time differences. For transmitting a time message from node A_i to A_j with the sending time t_s according to A_i's clock and the receiving time t_r according to A_j's clock, we have:

$$t_r = R_{ji} \times t_s + t_{0ji} + D_{ji} \tag{4.2}$$

where R_{ji} and t_{0ji} are the *relative clock drift* and *relative clock offset* between these two clocks and D_{ji} is the one-way delays measured by A_j's clock. Fig. 4.1 illustrates this relationship. Note that, when A_j receives the time message at time t_r, the reading of A_i's clock must be t_x.

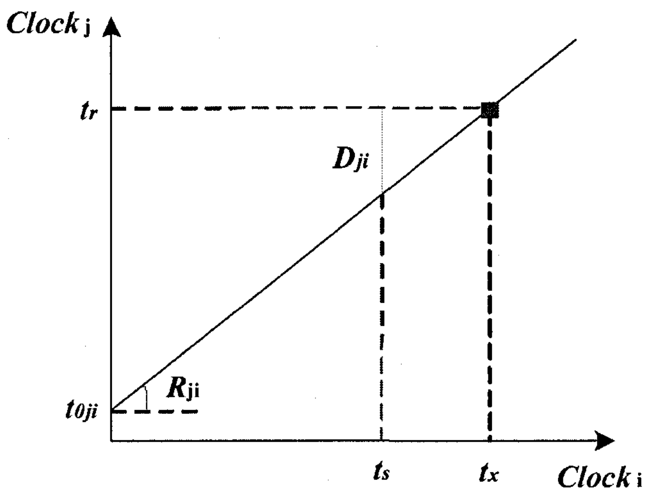

Figure 4.1: The relationship of time instances at nodes A_i to A_j when sending a time message from A_i to A_j

In general, the one-way delays in ICNs consist of two major parts:

- *Propagation delay*, which is the time spent from sending the message out of A_i's network interface to receiving it by A_j's interface.
- *Queuing delay*, which is the time spent for the message staying in the buffer and waiting for accessing A_i's network interface plus the time for A_j to retrieve the reading of the message out of its interface. As explained in [37], the queuing delay is usually larger than the propagation delay in ICNs and is asymmetric over link e_{ij} and link e_{ji}.

4.1.3 Performance Analysis of NTP-Core

NTP (the latest version is NTP4) has been the Internet time standard since 1992. NTP provides the robust time solutions for adjusting computer clocks in the Internet with respect to UTC. Till Aug. 2004, there are over 230 primary time servers and 10-20 millions clients using NTP around the world [65]. NTP actually includes a set of engineered methods, filters and algorithms of designing accurate hardware clocks, providing synchronized network time and supporting secure network authentications. In this study, we only discuss part of NTP's methods which are directly related to our research about how to estimate the clock drift and offset via exchanging synchronization messages between networking nodes. We denote these technologies as NTP-Core. The same abstract model of NTP-Core is also discussed in [46, 45, 38, 42].

Assume that node A_i and A_j are currently connected in the networks. In NTP-Core, A_j synchronizes its clock with A_i by exchanging a series of *ping-pong* messages as illustrated in Fig. 4.2. We call the exchange of a pair of ping-pong messages as a *round* of synchronization operations. NTP-Core keeps a *sliding window* of the latest n (n is the size of the window) rounds of message exchanges and picks up the one with the smallest round trip time (RTT) to conduct the estimations. This rule is called the "*minimum filter*" of NTP-Core.

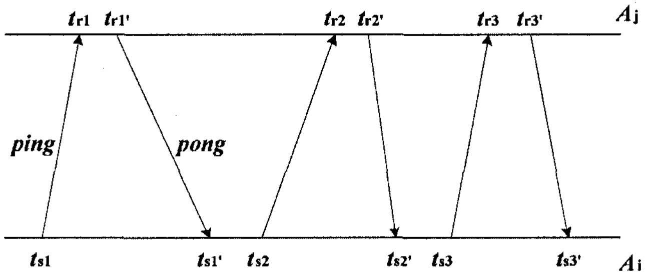

Figure 4.2: NTP-Core requires A_j exchange a series of time probes with A_i and picks up the smallest RTT to estimate the relative clock offset. In this figure, the 3rd message exchange is selected, assuming the size of the sliding window is 3.

Assume that the kth round of the ping-pong messages is chosen, which has the smallest $RTT = [(t_{rk} - t_{sk}) + (t'_{sk} - t'_{rk})]$ in the current window. NTP-Core performs the following calculations to estimate the relative clock offset and the relative clock drift between node A_i and A_j [66, 67]:

$$D'_{ji} = D'_{ij} = \frac{RTT}{2} = \frac{(t_{rk} - t_{sk}) + (t'_{sk} - t'_{rk})}{2} \tag{4.3}$$

$$t'_{0ji} = \frac{(t_{rk} - t_{sk}) - (t'_{sk} - t'_{rk})}{2} \tag{4.4}$$

$$R'_{ji} = 1 \tag{4.5}$$

Fig. 4.3 illustrates the fact of these equations: NTP-Core uses a linear function which goes through the middle point of the line segment AB with the slope fixed to be 1 to approximate the real relationship between A_i's and A_j's clocks. The coordinates of time point A and B are (t_{sk}, t_{rk}) and (t'_{sk}, t'_{rk}) respectively, which are collected from the measurements carried by the kth round of message exchange.

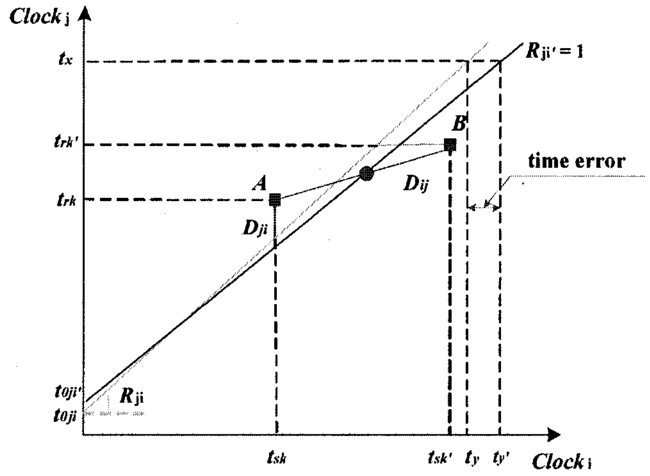

Figure 4.3: NTP's linear approximation of the real time relationship of the time instances between A_i and A_j uses a fixed slop equal to 1.

Based on equation 4.3, it is easy to tell that NTP-Core regards the one-way delays as symmetric over contact e_{ij} and e_{ji}. This is a sound assumption in the Internetworking environments where links are connected with very small error rate. However, it is not valid in ICNs because the queuing delay might be asymmetric over both directions due to the store-and-forward ICN routing mechanism, i.e., every message must be queued first before it is forwarded to the next hop. Another feature of NTP-Core is that, it takes the expected value of the relative clock drift, which is 1, as the approximation of the real value. This simplifies the calculations but it is inappropriate when node A_i and A_j operate in different physical environments. Our analysis in the following study also shows that fixing the clock drift to be 1 will cause larger time error of NTP-Core, when compared to the performance of DTP.

Based on equation 4.4 and 4.5, A_j's clock can then be adjusted in the following way: for any local time reading t_x with respect to A_j's clock, the corresponding estimation is t'_y at A_i as depicted in Fig. 4.3. However, the correct real time should be t_y according to A_i's clock. Therefore, the time error achieved using NTP-Core can be calculated as:

$$Err_{NTP-Core} = \frac{t_x - t'_{0ji}}{1} - \frac{t_x - t_{0ji}}{R_{ji}} \tag{4.6}$$

Clearly, as the sliding window of NTP-Core slides, the choice based on the "minimum filter" will change if a better measurement with smaller RTT than the current one is discovered. Meanwhile, the time error of NTP-Core keeps increasing as the estimated linear function keeps deteriorating away from the real one over time. Let $\triangle t$ be the synchronization interval, i.e. $t_{s(k+1)} - t_{sk} = t_{sk} - t_{s(k-1)} = \triangle t$. By taking equation 4.2 into accounts and simplifying equation 4.6, the maximum time error achieved by NTP-Core is then simplified as:

$$MaxErr_{NTP-Core} = (R_{ji} - 1) \times n\triangle t + \bar{M}_{NTP-Core} \tag{4.7}$$

where $\bar{M}_{NTP-Core}$ is called the residual of NTP's maximum time error and n is the size of the sliding window. Also,

$$\bar{M}_{NTP-Core} = \frac{D_{ij}^{pong} + D_{ji}^{ping} - 2R_{ji}D_{ji}^{ping}}{2R_{ji}} \tag{4.8}$$

where D_{ji}^{ping} and D_{ij}^{pong} represent the one-way delays of transmitting the *ping* and *pong* messages respectively.

From equation 4.7, we can conclude that the maximum time error of NTP-Core is a linear function over the synchronization interval. Considering the feature of the communication environments of ICNs, synchronization can not be conducted when the links between A_i and A_j are broken. Before the links are recovered, A_j must use the latest estimation of R_{ji} and t_{0ji} to adjust its clock and its time error will keep increasing according to equation 4.7. In the next section, we propose DTP, which can be viewed as an extension of NTP-Core in ICNs, to achieve smaller maximum time errors than NTP-Core.

4.1.4 DTP: Double-pairwise Time Protocol

The Idea of DTP

Basically, DTP tries to better estimate the relative clock drift between the client and the time server, rather than fixing it to be 1. The idea of DTP is illustrated in Fig. 4.4. Like NTP, DTP also performs a series of message exchanges between two networking nodes. Rather than exchanging a pair of ping-pong messages, DTP exchanges *pingping-pongpong* messages which are called double-pairwise messages in each round of synchronization operations. The two ping messages are transmitted back-to-back with a small interval d between their sending time instances, where d is a controllable protocol parameter and its impact on the performance of DTP will be discussed later. The purpose of sending back-to-back messages is that they can be queued very closely in the local buffer of A_i. In this way, they will experience the similar queuing delays. The similar ideas of using back-to-back messages are also presented in [69, 18, 57] to measure the available network bandwidth.

To simplify our following analysis without affecting the correctness, we use a

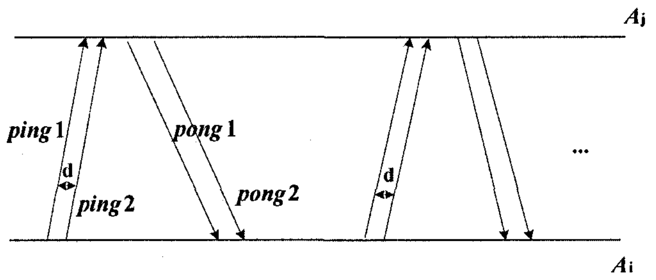

Figure 4.4: The basic idea of DTP: exchange back-to-back *pingping-pongpong* messages with a controllable protocol parameter *d*.

simplified (ideal) version of DTP to study its performance in the rest of this study. This simplified DTP is illustrated in Fig. 4.5. We assume that node A_j always sends back a pong message immediately after it receives a ping message from A_i.

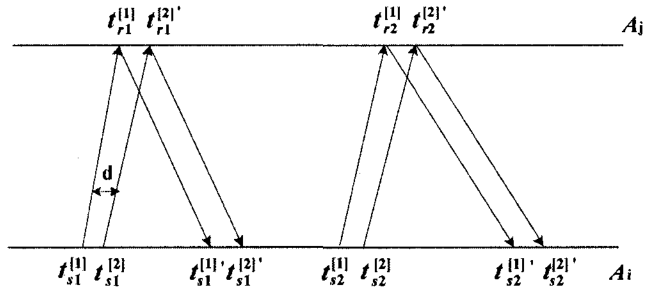
$t_{r1}^{[1]}$ $t_{r1}^{[2]'}$
$t_{r2}^{[1]}$ $t_{r2}^{[2]'}$
A_j
d
$t_{s1}^{[1]}$ $t_{s1}^{[2]}$
$t_{s1}^{[1]'}$ $t_{s1}^{[2]'}$
$t_{s2}^{[1]}$ $t_{s2}^{[2]}$
$t_{s2}^{[1]'}$ $t_{s2}^{[2]'}$
A_i

Figure 4.5: The basic idea of DTP: exchange back-to-back *pingping-pongpong* messages with a controllable protocol parameter *d*

After sending a ping message at $t_{sk}^{[1]}$ to A_j DTP requires that A_i generates and transmits a second ping message at $t_{sk}^{[2]}$, where $t_{sk}^{[2]} - t_{sk}^{[1]} = d$. Comparing the one-way delays of transmitting the back-to-back ping messages, the delay of the second one is slightly larger than that of the first one because it suffers larger queuing delays. The second ping message can be sent only after the first one is transmitted out of the network interface. We can have the same observations from the two pong messages.

DTP uses the same sliding window as NTP. However, it uses the *most similar* *filter* rather than the minimum filter used by NTP-Core:

In the sliding window contains n rounds of *pingping-pongpong* messages,

- calculate the difference between the one-way delays of the two ping messages: $D_{ji}^{ping2} - D_{ji}^{ping1}$;
- calculate the difference between the one-way delays of the two pong messages: $D_{ij}^{pong2} - D_{ij}^{pong1}$;
- calculate the difference between the above two;
- from the current window, select the round of pingping-pongpong messages which has the smallest result calculated in step 3;

Also assume that the kth round is selected after applying the filter to the current window. Then, four time points $(t_{sk}^{[1]}, t_{rk}^{[1]})$,$(t_{sk'}^{[1]}, t_{rk}^{[1]})$,$(t_{sk}^{[2]}, t_{rk}^{[2]})$, and $(t_{sk'}^{[2]}, t_{rk}^{[1]})$ can be collected from the associated time messages. DTP performs the following estimations:

$$t_{0ji'} = \frac{t_{rk}^{[1]} \times (t_{sk}^{[2]} + t_{sk}^{[2]}) - t_{rk}^{[2]} \times (t_{sk}^{[1]} + t_{sk'}^{[1]})}{t_{sk}^{[2]} - t_{sk}^{[1]} + t_{sk'}^{[2]} - t_{sk'}^{[1]}} \tag{4.9}$$

$$R_{ji'} = \frac{2 \times (t_{rk}^{[2]} - t_{rk}^{[1]})}{t_{sk}^{[2]} - t_{sk}^{[1]} + t_{sk'}^{[2]} - t_{sk'}^{[1]}} \tag{4.10}$$

Fig. 4.6 depicts the fact of these equations: DTP chooses a linear function which goes through the two middle points of line segment AB and CD to approximate the real time relationship between clocks of A_i and A_j. Compare to NTP, DTP does not fix the slope of the linear approximation function to be 1 but adjusts it according to the latest measurements. We will prove that such dynamic adjustments can better approximate the time relationship between two clocks of A_i and A_j.

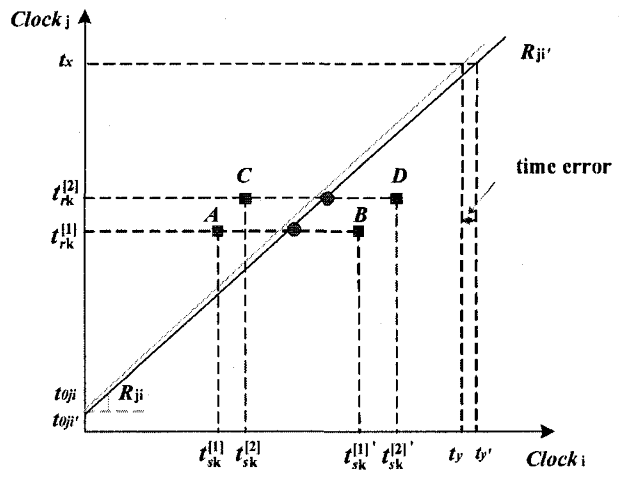

Figure 4.6: DTP's linear approximation of the real relationship of the time instances between A_i and A_j does not assume a fixed slope as 1.

Performance Analysis of DTP

From Fig.4.6, the time error achieved by DTP can be calculated as:

$$Err_{DTP} = \frac{t_x - t_{0ji\prime}}{R_{ji\prime}} - \frac{t_x - t_{0ji}}{R_{ji}} \tag{4.11}$$

The time error achieved using DTP increase as it keeps using the current selection until it selects a better measurement according to the most similar filter. In this case, the maximum time error based on the current measurements is:

$$MaxErr_{DTP} = \frac{[(D_{ij}^{pong2} - D_{ij}^{pong1}) \times R_{ji} - (D_{ji}^{ping2} - D_{ji}^{ping1})] \times n\triangle t}{2R_{ji} \times (R_{ji}d + D_{ji}^{ping2} - D_{ji}^{ping1})} + \bar{M}_{DTP} \tag{4.12}$$

by taking equation 4.9 and equation 4.10 into accounts. And the residual of DTP's maximum time error is:

$$\bar{M}_{DTP} = \frac{D_{ij}^{pong1} D_{ji}^{ping2} - D_{ij}^{pong2} D_{ji}^{ping1} - D_{ji}^{ping1} d + D_{ij}^{pong1} R_{ji} d}{2R_{ji} \times (R_{ji}d + D_{ji}^{ping2} - D_{ji}^{ping1})} \tag{4.13}$$

Comparing to equation 4.7, the maximum time error of DTP is also a linear function over the synchronization interval $\triangle t$. However, we have the following important properties of DTP.

Theorem 1: DTP estimates the relative clock drift R_{ji} more accurately than NTP-Core.

Proof: Let us first calculate the difference of the estimated clock drifts of DTP and NTP-Core to the real R_{ji}. We have

$$Diff_{DTP} = R_{ji'} - R_{ji}$$

$$Diff_{NTP-Core} = 1 - R_{ji}$$

So

$$Diff_{DTP} - Diff_{NTP-Core} = R_{ji'} - 1 = \frac{d - (t_{sk'}^{[2]} - t_{sk'}^{[1]})}{d + (t_{sk'}^{[2]} - t_{sk'}^{[1]})}$$

From equation 4.2, we know that:

$$t_{sk'}^{[2]} = \frac{t_{sk}^{[2]} - t_{0ji}}{R_{ji}} + D_{ij}^{pong2}$$

$$t_{sk'}^{[1]} = \frac{t_{sk}^{[1]} - t_{0ji}}{R_{ji}} + D_{ij}^{pong1}$$

And

$$t_{rk}^{[2]} = t_{sk}^{[2]} \times R_{ji} + t_{0ji} + D_{ji}^{ping2}$$

$$t_{rk}^{[1]} = t_{sk}^{[1]} \times R_{ji} + t_{0ji} + D_{ji}^{ping1}$$

Recall that $t_{sk}^{[2]} - t_{sk}^{[1]} = d$. Thus, $\triangle Diff = Diff_{DTP} - Diff_{NTP-Core}$ can be further simplified as:

$$\triangle Diff = -\frac{(D_{ji}^{ping2} - D_{ji}^{ping1}) \times \frac{1}{R_{ji}} + (D_{ij}^{pong2} - D_{ij}^{pong1})}{d + t_{sk'}^{[2]} - t_{sk'}^{[1]}}$$

The design of DTP guarantees that the second pairwise of ping-pong messages experience additional queuing delays than the first one, i.e., $D_{ji}^{ping2} - D_{ji}^{ping2} > 0$ and $D_{ij}^{pong2} - D_{ij}^{pong1} > 0$. And obviously, the second pong message can only be received after the first one is received, i.e., $t_{sk'}^{[2]} - t_{sk'}^{[1]} > 0$. Therefore, we have

$$Diff_{DTP} - Diff_{NTP-Core} < 0$$

In other words, DTP has better estimation of the relative clock drift than NTP-Core.

■

Theorem 2: The increasing speed of DTP's maximum time error is about half of that of NTP-Core.

Proof: From equation 4.12 and 4.7, we have:

$$ErrRate_{DTP} = \frac{d(MaxErr_{DTP})}{d(\Delta t)}$$

$$ErrRate_{DTP} = \frac{n \times [(D_{ij}^{pong2} - D_{ij}^{pong1})R_{ji} - (D_{ji}^{ping2} - D_{ji}^{ping1})]}{2R_{ji} \times (R_{ji}d + D_{ji}^{ping2} - D_{ji}^{ping1})} \tag{4.14}$$

and

$$ErrRate_{NTP-Core} = \frac{d(MaxErr_{NTP-Core})}{d(\Delta t)} = n \times (R_{ji} - 1) \tag{4.15}$$

Using the most similar filter, DTP guarantees that

$$D_{ij}^{pong2} - D_{ij}^{pong1} \simeq D_{ji}^{ping2} - D_{ji}^{ping1} = \mu > 0$$

where μ is the difference of the one-way delays of transmitting the two back-to-back ping messages, i.e., the additional queuing delay experienced by the second ping message. Then we have:

$$0 < \frac{ErrRate_{DTP}}{ErrRate_{NTP-Core}} = \frac{\mu}{2R_{ji}^2 d + 2R_{ji}\mu} < \frac{\mu}{2R_{ji}\mu} \simeq \frac{1}{2}$$

■

Therefore, DTP is capable of achieving smaller maximum time error than NTP-Core. The advantage of DTP's slower error-increase speed is significant in ICNs where links connecting A_i and A_j may be broken for a long period of time.

Preliminary Results

In order to show the above features of DTP, we implemented it in NS-2 (network simulator 2) [4]. We apply NTP-Core and DTP to the typical 20 nodes IP network topology suggested in [45, 98], as shown in Fig. 4.7.

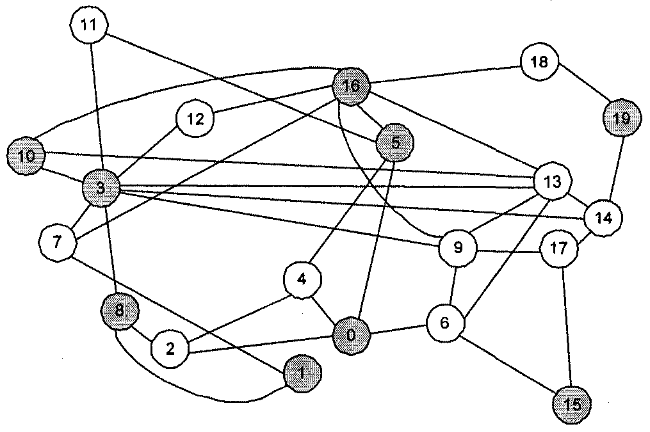

Figure 4.7: Typical 20 nodes IP networks. Dark nodes are the time servers and white nodes are the time clients. In total, we have 9 time servers. There is one server has two time clients.

We organize the 20 nodes into 10 pairs of clients and time servers to conduct network time synchronization. The links between each pair are continuously connected. The simulation time is set to be 5 days and the synchronization interval is 1 minute. The clock drifts of time clients are randomly chosen from 1 to 1+50ppm using Gaussian distribution and the initial clock offsets of 10 clients are randomly chosen from 5 to 10 seconds. The parameter d in DTP is set as 0.01 second. The average of the time errors of these synchronization peers achieved using NTP-core and DTP are illustrated in Fig. 4.8.

Our preliminary results shows that DTP is capable of achieving smaller time errors than NTP with continuous end-to-end connections. And Table 4.1 lists the maximum time error of both synchronization approaches in each simulation day. The maximum

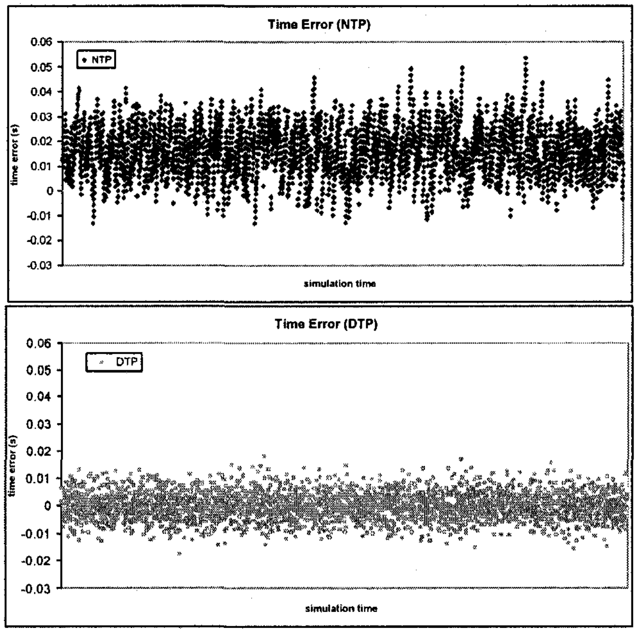

Figure 4.8: Time errors achieved by NTP-Core and DTP. No link disruptions in the networks during the simulation time of 5 days.

Table 4.1: Maximum Time Error in Each Day

Day	NTP-Core (seconds)	DTP (seconds)
1	0.04177	0.01491
2	0.04106	0.01928
3	0.04582	0.01661
4	0.04963	0.01881
5	0.05380	0.01746

time error of DTP is 19.3ms while that of NTP-Core is 53.8ms.

4.1.5 Performance Evaluation

Applying the ICN Traces

To evaluate the performance of DTP and NTP-Core in the communication environments of ICNs, we modify the NS-2 simulator to manage the ICN link up and down schedules. We use the same 20 nodes topology and in Fig. 6, with applying the trace files collected from the existing ICN testbeds. These ICN traces include the Pocket Switch Networks (PSN) trace [47] and the DieselNet trace (without using the throwboxes) [11]. We reduce the simulation time to be 1 day.

In the PSN experiment, wireless devices called *iMotes* are put into the pockets of the conference attendees at IEEE INFOCOM 2005. A link is up only when two people encounter with each other in the same conference room. It is reported that the inter-link/inter-contact durations in PSN trace follow the power-law distribution, i.e., links have more chances to be down for longer period of time. We follow the same distribution and vary the total link unavailable time (i.e., total inter-link/iter-contact durations) from 10% to 90% of the overall simulation time. Our results are shown in Fig. 4.9. We observe that: *i*) the time errors of DTP and NTP-Core increase when ICN links are down for longer time, which is explained by the equation 4.7; *ii*) the maximum time error achieved using DTP is less than half of that of NTP-Core; and *iii*) DTP also outperforms NTP-Core in terms of achieving smaller mean values of the time errors.

The advantage of DTP becomes significant when applying the DieselNet traces. DieselNet is a outdoor ICN experiment in which ICN devices called Bricks are mounted on buses moving around Amherst in Massachusetts. There are 30 buses running in a 17 square mile area. A link is available only when two buses encounter in the route. Compared to the PSN traces, the inter-link/inter-contact duration in DieselNet is much longer. Our statistical results show that the total inter-contact durations vary from 98.43% to 99.99% of the total system operating time. The network connectivity of the DieselNet is much worse than that of the PSN.

Our results in Fig. 4.10 show that: *i*) the maximum time error of NTP-Core increases quickly when the DieselNet traces are applied. When links are down during 99.99% time in 1 day, the maximum time error achieved using NTP-Core is 1.122

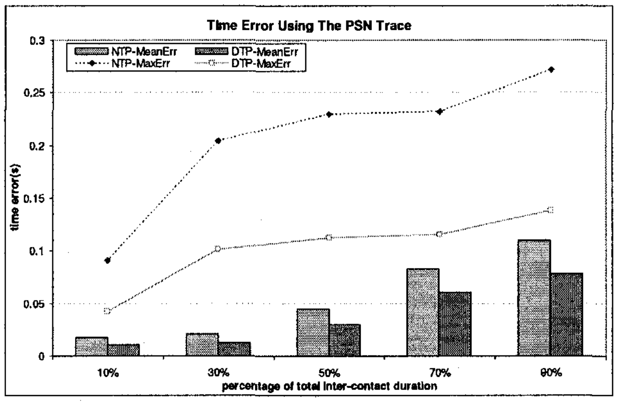

Figure 4.9: The maximum and mean time errors of DTP and NTP-Core using the PSN traces.

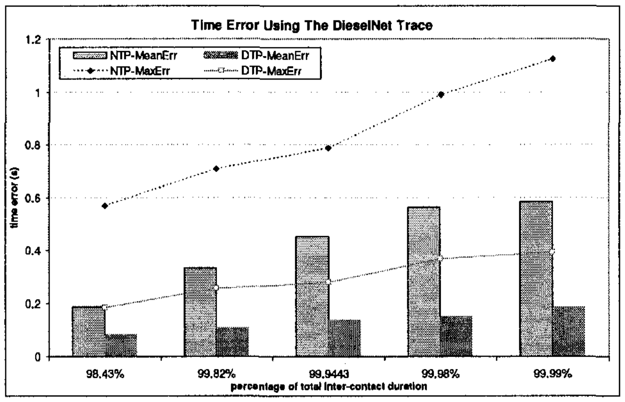

Figure 4.10: The maximum and mean time errors of DTP and NTP-Core using the DieselNet traces.

second. However, if no synchronization operation is taken, the maximum time error will become 3.114 second (by taking 1+36ppm as the clock drift and the initial clock offset is 0 second); *ii*) DTP can achieve much smaller maximum time errors than

NTP-Core. DTP outperforms NTP-Core especially when links are down for longer period of time.

We also study the impact of the protocol controllable parameter d on the performance of DTP. We use the PSN trace and manage the total inter-contact durations to be 70% of the overall simulation time. From the results illustrated in Fig. 4.11, we can tell that i) the maximum time error of DTP decreases as d increases, which matches their relationship in equation 4.12; and ii) DTP's maximum time error follows a long-tail distribution over d. The performance of DTP does not improve much as d is larger than 0.1 second, under the current simulation configurations. Also, when d is larger than 1 second, it loses the physical meaning of generating two back-to-back ping messages any more.

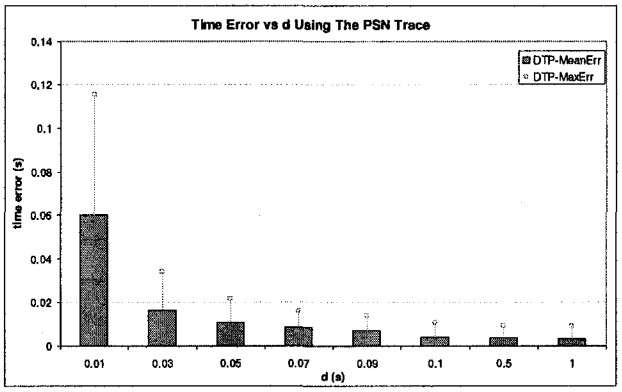

Figure 4.11: The impact of d on DTP's time error.

4.1.6 Introducing the Node Mobility

In this simulation, we randomly deploy 20 ICN nodes and make them move following the mobility patterns retrieved from the ZebraNet trace [89]. ZebraNet trace records the location information of a group of zebras in Kenya and is regarded as a widely utilized mobility model in ICNs. We randomly pick up 10 pairs of synchronization peers from these 20 nodes and collect the average of their synchronization results.

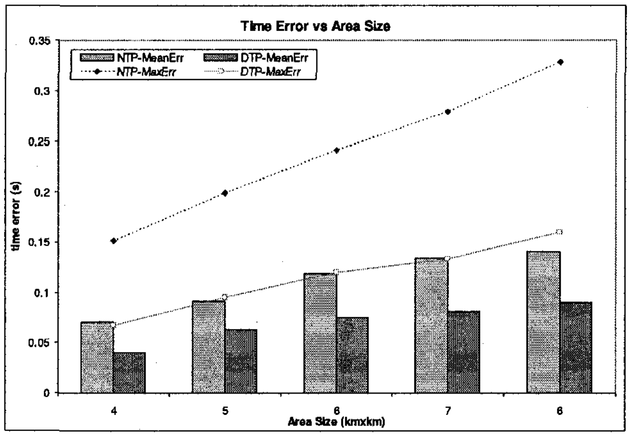

Figure 4.12: The impact of the network area size on time errors.

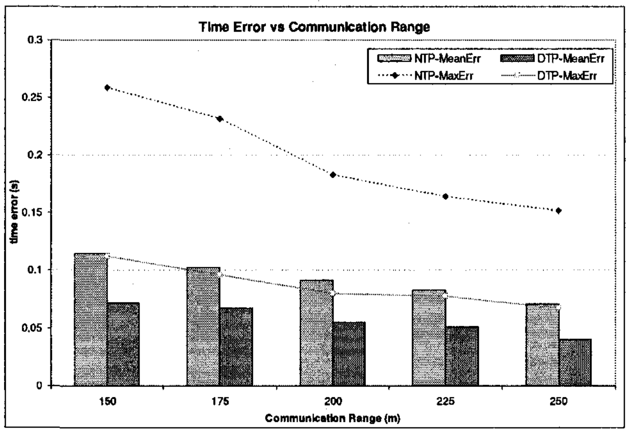

Figure 4.13: The impact of the communication range on time errors.

We first fix the communication range of each node to be 250m and vary the area size from 4 to 8 square kilometers. As illustrated in Fig. 4.12, the maximum time error of both NTP and DTP increases as the network area becomes larger. It is more difficult for nodes to encounter with another when they move following the same mobility pattern in bigger area. Thus the links are down for longer period of time and the time errors increase.

We then fix the size of the network area as 2000m2000m and vary the data communication range of each node from 150m to 250m. Clearly, when the transmission range increases it is easier for a client to detect the time server in its neighborhood. As a result, the total inter-contact durations of the synchronization peers are decreased. As shown in Fig. 4.13, the performances of NTP and DTP can be improved if ICN nodes are equipped with long-range radios. We also observe that DTP's maximum time error is still less than that of NTP with introducing node mobility.

4.1.7 Discussion

The communication environments of ICNs feature frequent contact up/down variations. In ICNs, the network time synchronization operations by exchanging time messages have to stop when links connecting the clients to the time server are broken. In this study, we first prove that the maximum time error achieved using NTP-Core quickly increases in a linear way when links are down for a long period of time. We propose DTP as an extension of NTP-Core in ICNs, which achieves approximately half of the maximum time error than that of NTP-Core. DTP has small implementation cost and can be easily implemented using the architecture of NTP. The benefit of DTP comes from the additional synchronization messages it collects by using the most similar filter. Compared to NTP-Core, DTP doubles the number of messages in the sliding window. For highest accuracy we believe that low-level timestamping is essential. However, to access MAC-layer timestamping requires cross-layer design and optimization, which currently are not addressed in the IETF ICN architecture [28]. In our next design of DTP, we intend to utilize low-level timestamping similar as FTSP to further improve DTP's synchronization accuracy.

4.2 LESSAR: LEvel Synchronization using Sender, Adjuster and Receiver for Wireless Sensor Networks

4.2.1 Introduction

In the previous study, we provide a solution of synchronizing clocks of mobile data collectors in the upper-tier ICNs to the data center in our two-tier system architecture of WASNs. In this study, we focus on studying how to keep the clocks of sensors in the lower-tier Wireless Sensor Networks(WSNs) tick at the same speed.

The resource-limitation constraints of sensor nodes, the ad-hoc network topology of WSNs, and the broad range of applications impose specific requirements on protocol design of time synchronization in WSNs:

- The approach must be light-weighted and efficient. The underlying reason for this requirement is that sensor nodes have small amount of memory, limited battery power, low CPU computing strength, and slow data bus. These hardware constraints lead to the requirement of a light-weighted time synchronization protocol with small storage occupation, low computation complexity, and little energy consumption. In other words, the protocol should be storage-efficient, computation-efficient, and especially energy-efficient so that a sensor node is capable of supporting the time synchronization protocol as a basic functional module.

- The approach must be self-configurable. Because sensor nodes may be deployed in ad-hoc manner, they should not be constrained by synchronizing their local clocks with some specific or fixed sources of time references. The WSN should be able to autonomously maintain a proper structure to achieve time synchronization.

- The approach must be tunable. Different applications have different requirements of timing precision. For example, detecting the speed of a truck moving across a bridge requires the observing time with the accuracy of second while detecting temperature changes during an explosion or earthquake observation needs time accuracy of microsecond. Generally, the better accuracy a time synchronization protocol achieves, the more synchronization packet it requires and thus the more energy consumption and computation complexity it has. Tunable synchronization algorithm is desirable to make a trade-off between its synchronization accuracy and energy consumption.

4.2.2 Analytical Model

Timing Errors of a Single Node

In general, time difference between the timing result of a local clock on a node X and the Coordinated Universal Time (UTC) comes from the imperfect match between the UTC and the output frequency of its internal hardware oscillator, which provides a vibrating frequency to keep X's local clock continuously running. The oscillator's frequency depends on its natural properties and the surrounding physical environment of this node. In practice, the relationships between the clock time t_x generated by the oscillator and the UTC t satisfies the following function:

$$t_x = a_x t + t_{0x} + Drift_x(t) \tag{4.16}$$

where a_x is the clock skew caused by the natural frequency differences between X's oscillator and the ideal standard ones and t_{0x} is the initial time offset. $Drift_x$ is the rambled clock drifts caused by the variations of environment conditions such as temperature. If the clock of X perfectly followed the UTC time t, a_x would be equal to 1 and both t_{0x} and $Drift_x$ would be zero. Fig.4.14 presents one case described by Eq. 4.16.

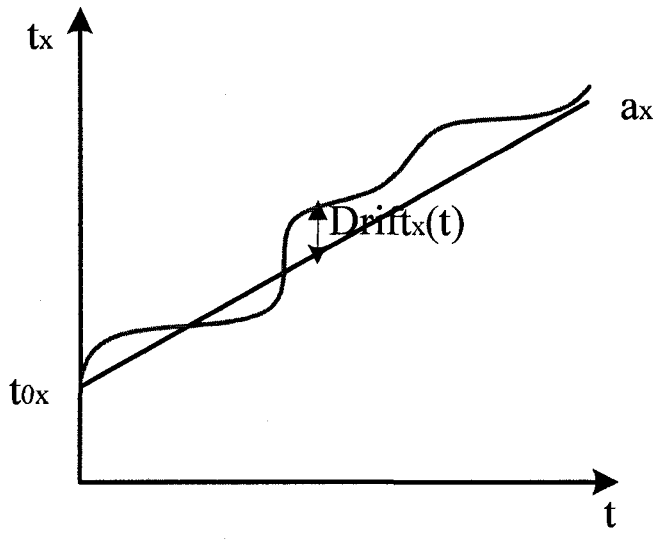

Figure 4.14: The relationship between local clock and UTC.

Clock drifts shown in Fig.4.14 is an environment-related phenomenon and it is hard to be modeled accurately. Even the relationships between the clock drifts and environmental factors are perfectly known, it is still an overly strong assumption for general time synchronization methods that the accurate drifts can be estimated by measuring the environment properties in real-time and deducing the drifts from the drift-environment relationships.

In fact, the clock drift is additive to the overall errors of a node's local clock. If the sampling rate is high enough, the environmental factors will not change much so that the curve with drift (e.g., Fig.4.14) can be approached very well by a sequence of line-segments. In each segment, a linear regression method can be applied to achieve time synchronization. In our application scenarios, the sampling rate is generally high enough, such that I limit the data sets of linear regression to the most recent ones in the following discussion. Therefore, Eq. 4.16 can be replaced by:

$$t_x = a_x(n)t + t_{0x}(n), nT < t < (n+1)T \tag{4.17}$$

where T is the sampling rate. The clock drifts in nth period of sampling time are hidden in the values of $a_x(n)$ and $t_{0x}(n)$.

Timing Errors between Two Networking Nodes

Traditional time synchronization algorithm requires a client follow the clock of a server. Consider two networking nodes shown in Fig.4.15. Node S acts as the sender that sends out a sequence of reference packets with time stamps to the receiver node R.

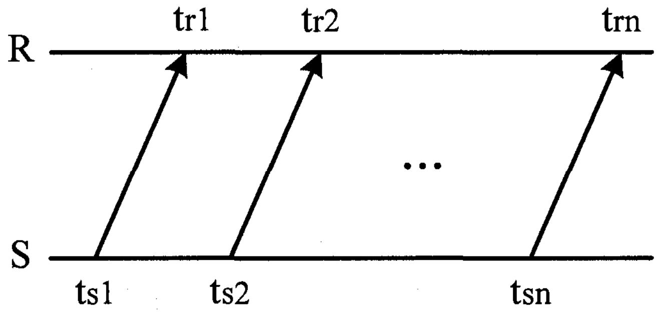

Figure 4.15: The procedure of one-way message transmission for time reference packets.

According to Eq. 4.17, the relationship between the time at the sender S, denoted as t_s, and the time at the receiver R, denoted as t_r, can be initially expressed as follows:

$$t_r = a_{rs}(n)t_s + t_{0rs}(n) \tag{4.18}$$

Along the message transmission path, there are four major delays between the sending time and receiving time of a synchronization packet from the sender to the receiver:

- *Processing delay* (P part): the time spent at node S to prepare and process the packet, and the time for the microprocessor to transfer the packet to its networking component;
- *Accessing delay* (A part): the delay for the packet waiting for accessing the wireless channel;
- *Propagation delay* (D part): the time spent from sending the packet out from the wireless radio at node S to receiving it by the wireless radio at node R;
- *Receiving delay* (R part): the time for node R to process the packet and get the reading of its local clock.

I combine these four parts together and name it as PADR delay. In practical, values of processing, accessing, and receiving delays (the P, A, and R parts) depend on the current workload of node R and S, the implementation of their Media Access Control (MAC) layer protocols, as well as the capability of their microprocessors and communication radios. The propagation delay (i.e. the D part) can be calculated as the physical distance between S and R divided by the speed of light. Clearly, these delays will affect the accuracy of synchronization algorithms since they make the linear regression functions move away from the accurate positions. Thus, Eq. 4.18 should be extended as follows when using this one-way message exchanging approach to synchronize two networked nodes:

$$t_r = a_{rs}(n)t_s + t_{0rs}(n) + PADR_{sr} \tag{4.19}$$

Our Analytical Model

From the above discussion, if two events simultaneously take place at time t_r (observed at node R) and t_s (observed at node S), the point (t_s, t_r) must be located in the line, namely accurate time line, with the slope equal to a_{rs} illustrated in Fig.4.16. Suppose at time t_{s1}, node S sends a reference packet with time stamp t_{s1} to node R and R receives this packet at t_{r1} with the respect to its own local clock. At the same time of t_{r1}, the time at node S must be t'_{s1}, derived from the cross-point of t_{r1} and the accurate time line, then mapped it to S's clock. Clearly, after receiving the second reference packet containing time stamp t_{s2} at t_{r2}, node R is able to get its own practical time line that goes through the points $(ts1, tr1)$ and $(ts2, tr2)$. The differences between the practical time line and the accurate time line are the PADR delays from S to R. To improve the accuracy of time-synchronization algorithms, it is necessary: (i) to give an estimation of the PADR errors in order to get an estimated time line by decreasing estimated PADR delays from the practical time line. This estimated time line can be taken by node R to estimate the accurate time at reference node S when an event happens; and (ii) to make the estimated time line as close to the accurate time line as possible in order to improve the accuracy of synchronization algorithm.

PADR Estimation

PADR delays between node S and node A can be estimated by the simple two-way message exchange mechanisms depicted in Fig.4.17. At time t_{s1}, node S sends out a packet that is received by node A at t_{a1} with respect to its local clock. Node A timestamps this packet with t_{a1} and sends it back immediately so that node S catches the answered packet at t'_{s1}. There goes the second round in the same way.

From Eq. 4.19, there are $PADR_{sa}$ delay between t_{a1} and t_{s1} and $PADR_{as}$ delays between t'_{s1} and t_{a1}. Assume that the workloads at the sender S and the node A are

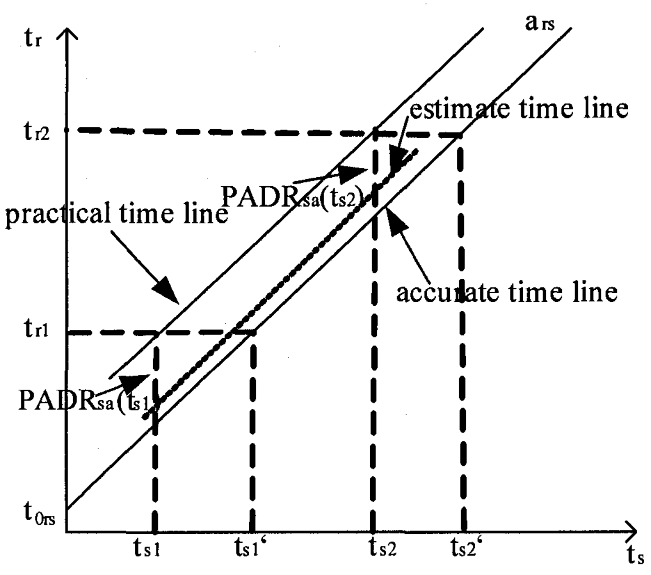

Figure 4.16: The analytical model for time synchronization approaches.

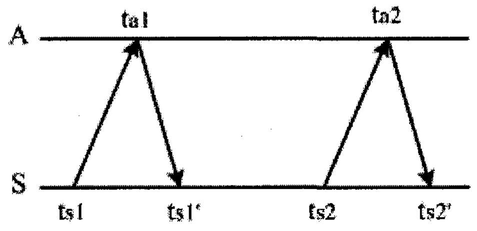

Figure 4.17: The procedure of a Two-way message exchange.

similar to each other, i.e. $PADR_{sa} \approx PADR_{as}$. Fig.4.18 presents the basic idea of two-way message exchange to synchronize the clocks of these two nodes. By locating the four points (t_{s1}, t_{a1}), (t'_{s1}, t_{a1}), (t_{s2}, t_{a2}), and (t'_{s2}, t_{a2}), we can get one practical time line goes through the pair of (t_{s1}, t_{a1}) and (t_{s2}, t_{a2}) that construct the upper bound line of the accurate time line with the difference of $PADR_{sa}$, and another one goes through (t'_{s1}, t_{a1}) and (t'_{s2}, t_{a2}), that constructs the lower bound line with the

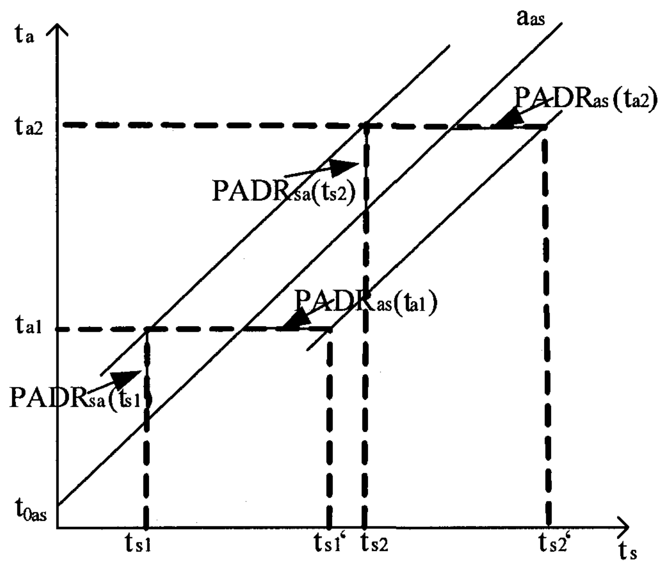

Figure 4.18: The upper bound and lower bound of $PADR$.

difference of $PADR_{as}$ respectively. Based on those four known time points, we can easily calculate the slopes and offsets of the two practical time lines as:

$$Slope_{upper} = \frac{t_{a2} - t_{a1}}{t_{s2} - t_{s1}} \tag{4.20}$$

$$Offset_{upper} = \frac{t_{a1}t_{s2} - t_{a2}t_{s1}}{t_{s2} - t_{s1}} \tag{4.21}$$

$$Slope_{lower} = \frac{t_{a2} - t_{a1}}{t'_{s2} - t'_{s1}} \tag{4.22}$$

$$Offset_{lower} = \frac{t_{a1}t'_{s2} - t_{a2}t'_{s1}}{t'_{s2} - t'_{s1}} \tag{4.23}$$

Let's take the difference between the upper bound line and lower bound line at time t_{s1} and t_{s2} as $Differ(t_{s1})$ and $Differ(t_{s2})$.

Theorem: $Differ(t_{s1}) \approx 2 \times PADR_{sa}(t_{s1})$, $Differ(t_{s2}) \approx 2 \times PADR_{sa}(t_{s2})$.

Proof: From Eq. 4.20, 4.21, 4.22, and 4.23, we get:

$$Differ(t_{s1}) = \frac{(t'_{s1} - t_{s1})(t_{a2} - t_{a1})}{(t'_{s2} - t'_{s1})} \tag{4.24}$$

$$Differ(t_{s2}) = \frac{(t'_{s2} - t_{s2})(t_{a2} - t_{a1})}{2(t'_{s2} - t'_{s1})} \tag{4.25}$$

Observe that the points (t_{s1}, t_{a1}), (t'_{s1}, t_{a1}), (t_{s2}, t_{a2}), and (t'_{s2}, t_{a2}) satisfy the following equations:

$$t_{a1} = a_{as}(n)t_{s1} + t_0(n) + PADR_{sa}(t_{s1}) \tag{4.26}$$

$$t_{a2} = a_{as}(n)t_{s2} + t_0(n) + PADR_{sa}(t_{s2}) \tag{4.27}$$

$$t_{a1} = a_{as}(n)t'_{s1} + t_0(n) - PADR_{as}(t_{a1}) \tag{4.28}$$

$$t_{a2} = a_{as}(n)t'_{s2} + t_0(n) - PADR_{as}(t_{a2}) \tag{4.29}$$

Since $PADR_{sa}(t_{s1}) \approx PADR_{as}(t_{a1})$ and $PADR_{sa}(t_{s2}) \approx PADR_{as}(t_{a2})$, we then have:

$$Differ(t_{s1}) \approx 2PADR_{sa}(t_{s1}) \frac{a_{as}(n)t_{s2} - a_{as}(n)t_{s1} + PADR_{sa}(t_{s2}) - PADR_{sa}(t_{s1})}{a_{as}(n)t_{s2} - a_{as}(n)t_{s1} + 2PADR_{sa}(t_{s2}) - 2PADR_{sa}(t_{s1})} \tag{4.30}$$

In practice, the synchronization interval, i.e. the difference between t_{s2} and t_{s1}, is much larger than the difference between two PADR delays. Therefore $Differ(t_{s1}) \approx 2PADR_{sa}(t_{s1})$.

Following the same way, we get $Differ(t_{s2}) \approx 2PADR_{sa}(t_{s2})$.

Thus,

$$PADR_{estimated}(t_{s1}) = \frac{(t'_{s1} - t_{s1})(t_{a2} - t_{a1})}{2(t'_{s2} - t'_{s1})} \tag{4.31}$$

$$PADR_{estimated}(t_{s2}) = \frac{(t'_{s2} - t_{s2})(t_{a2} - t_{a1})}{2(t'_{s2} - t'_{s1})} \tag{4.32}$$

If we define $\hat{t}_{a1} = t_{a1} - PADR_{estimated}(t_{s1})$ and $\hat{t}_{a2} = t_{a2} - PADR_{estimated}(t_{s2})$, then the estimated time line can be drawn by passing through $(t_{s1}, \hat{t}_{a1})$ and $(t_{s2}, \hat{t}_{a2})$.

4.2.3 Error Limit Analysis

Based on the results of the discussions and analysis above, all nodes can calculate PADR delay after getting every four points in two loops of two-way message exchange. During the period from the end of the second loop to the end of the third one, a node can only estimate the current PADR delay by locating a point in the extension of the newest estimated time line based on information of the last two loops. The difference between the estimated PADR delay from history information and the real accurate one is the synchronization error that a time synchronization algorithm has.

The PADR delay can be regarded as the summation of a constant part $PADR_c$ and a random part $PADR_r$. Deterministic components in the PADR delay that can be determined by the hardware implementation of nodes and network deployment, e.g., the type of hardware chips, the protocol of MAC layer, the capability of communication channels, the distance between two nodes, etc., are included into $PADR_c$. Other non-deterministic components form $PADR_r$. Thus the $PADR_c$ is a constant that does not affect the shape of the distribution curve of the synchronization error, although it does affect the absolute value. The random $PADR_r$ is the one that affects the error distribution. In the following discussion, I focus on analyzing the synchronization accuracy related to the random $PADR_r$ and assume that the constant $PADR_c$ can be accurately estimated. To make the analysis clear, I study $PADR_r$ in two cases: bounded and unbounded ones.

Bounded $PADR_r$

If the absolute value of $PADR_r$ is limited by a bound b, i.e. $|PADR_r| \leq b$, the maximum error would be $3b$.

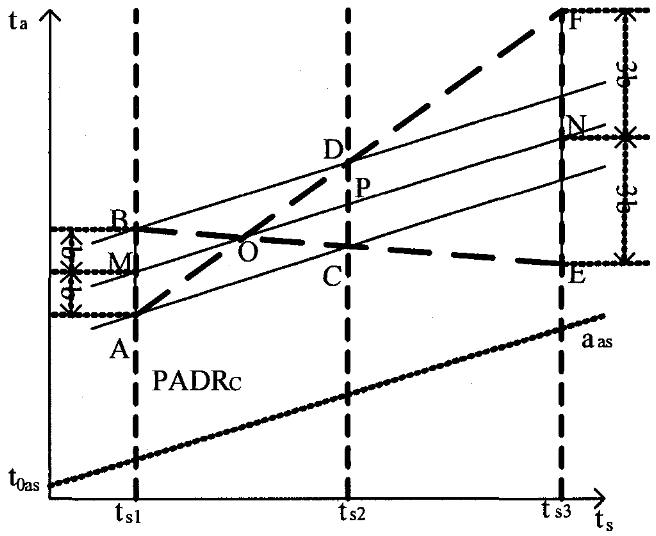

Figure 4.19: The estimate of time synchronization errors when $PADR_r$ is bounded.

The analysis is illustrated in Fig.4.19. Line BD, AC and MN are parallel with the accurate time line. The distance between BD and MN and that between AC and MN are +b and -b respectively, and the distance between MN and the accurate time line is the constant $PADR_c$. The line BD and the line AC represent the upper bound and lower bound of PADR delays due to the random $PADR_r$.

When node A receives the packet with time stamp t_{s1}, the accurate PADR at that time would be located in the segment BA. In the same way, the accurate PADR corresponding to the packet with time stamp t_{s2} would be located in segment DC. According to the above discussions, the points with the estimated PADR must be located in the extension of the lines that go through the latest time points. After drawing two assistant lines AD and BC, it is easy to find that the worst case happens at point F or E, in which I get the maximum PADR estimation errors. Assume that node S sends out the packets in a fixed time interval, i.e. $t_{s3} - t_{s2} = t_{s2} - t_{s1}$. It is easy to find that $|MO| = |OP| = 0.5 \times |PN|$. Therefore the maximum estimation error is $|FN| = |NE| = 3b$, due to the features of the parallel lines. Furthermore, the error distribution function of the points in segment EF can be described:

$$P(e > a) = \begin{cases} 0 & (a > 3b) \\ \int_{\frac{a-b}{2}}^{b} f(x) \int_{-b}^{2x-a} f(y)dydx & (3b \geq a > b) \\ \int_{\frac{a-b}{2}}^{\frac{a+b}{2}} f(x) \int_{-b}^{2x-a} f(y)dydx + \int_{\frac{a+b}{2}}^{b} f(x)dx & (b \geq a \geq -b) \\ \int_{-b}^{\frac{a+b}{2}} f(x) \int_{-b}^{2x-a} f(y)dydx + \int_{\frac{a+b}{2}}^{b} f(x)dx & (-b > a \geq -3b) \\ 1 & (-3b > a) \end{cases} \tag{4.33}$$

in which, e is the random variable, $f(x)$ is the distribution function of the random $PADR_r$, and a is the distance from the accurate time point to point N.

Let us take the uniform distribution as an example. If $f(x) = \frac{1}{2b}$ when $-b < x < b$, the integration of above formulas can be evaluated as follows (presented in Fig.4.20):

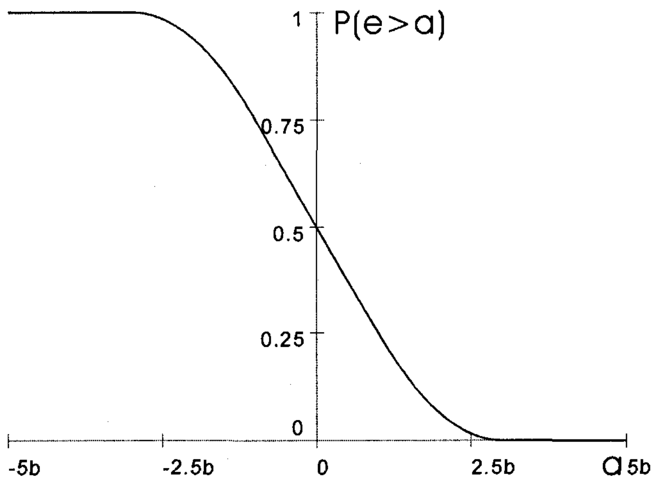

Figure 4.20: A distribution of the errors of upper estimated time line where $PADR_r$ is uniformly distributed.

$$P(e > a) = \begin{cases} 1 & (a > 3b) \\ \frac{1}{16b^2}\left(a^2 - 6ab + 9b^2\right) & (3b \geq a > b) \\ \frac{1}{2} - \frac{a}{4b} & (b \geq a \geq -b) \\ \frac{7}{16} - \frac{1}{16}\frac{a^2}{b^2} - \frac{3}{8}\frac{a}{b} & (-b > a \geq -3b) \\ 0 & (-3b > a) \end{cases} \tag{4.34}$$

Thus, the probability density function of $P(e > a)$can be expressed as follows:

$$f_e^b(a) = -P(e > a)'_a = \begin{cases} \frac{1}{8b^2}(3b - a) & 3b > a > b \\ \frac{1}{4b} & b > a > -b \\ \frac{1}{8}\frac{a}{b^2} + \frac{3}{8}\frac{1}{b} & -b > a > -3b \\ 0 & \text{otherwise} \end{cases} \tag{4.35}$$

Fig.4.19 only proposes the estimation errors of the upper bound line. The other half of estimation error in the lower bound could be analyzed in the same way. The total time synchronization error is nothing more than an average of the self-convolution of these two Identified Independent Distribution (IID) random variables:

$$f_s^b(t) = \int_{-\infty}^{\infty} f_e^{\frac{b}{2}}(x)\, f_e^{\frac{b}{2}}(t-x)\,dx = \begin{cases} \frac{9}{8b} - \frac{9}{8b^2}t + \frac{3}{8b^3}t^2 - \frac{1}{24b^4}t^3 & 2b < t < 3b \\ \frac{11}{24b} - \frac{1}{8b^2}t - \frac{1}{8b^3}t^2 + \frac{1}{24b^4}t^3 & b < t < 2b \\ \frac{5}{12b} - \frac{1}{4b^3}t^2 + \frac{1}{12b^4}t^3 & 0 < t < b \\ \frac{5}{12b} - \frac{1}{4b^3}t^2 - \frac{1}{12b^4}t^3 & -b < t < 0 \\ \frac{11}{24b} + \frac{1}{8b^2}t - \frac{1}{8b^3}t^2 - \frac{1}{24b^4}t^3 & -2b < t < -b \\ \frac{9}{8b} + \frac{9}{8b^2}t + \frac{3}{8b^3}t^2 + \frac{1}{24b^4}t^3 & -3b < t < -2b \\ 0 & \text{otherwise} \end{cases} \tag{4.36}$$

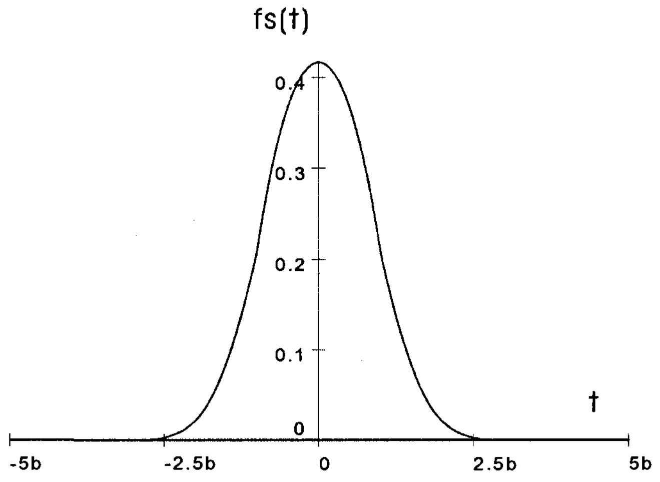

Figure 4.21: An error distribution of estimated PADR where $PADR_r$ is uniformly distributed

Fig.4.21 presents the final error distribution curve when $PADR_r$ follows a uniform distribution.

Unbounded $PADR_r$

If the absolute value of $PADR_r$ is unbounded in $[0,\infty)$, I can follow the similar analysis method and get the new distribution function as following:

$$P(e>a)=\begin{cases}\int\limits_{\frac{a}{2}}^{\infty} f(x)\int\limits_{0}^{2x-a} f(y)dydx & a>0\\ \int\limits_{0}^{\infty} f(x)\int\limits_{0}^{2x-a} f(y)dydx & a<0\end{cases} \tag{4.37}$$

In fact, the cases of the unbounded $PADR_r$ in the real world only happen when node fails.

4.2.4 LESSAR Protocol

Based on the analytical model I proposed, We propose a light-weighted time synchronization protocol, namely LEvel Synchronization by Sender, Adjuster and Receiver (LESSAR), which is able to achieve the accuracy limitation while keep characteristics of low power consumption, affordable storage requirement, and small computation complexity by dramatically reducing the packets transmissions overheads. In order to demonstrate the effectiveness of the proposed analytical model, I study this protocol as an example to illustrate the usefulness of the proposed analytical model.

Observations

In order to design a time-synchronization approach that satisfies the accuracy requirement while keeps the overhead consumption as low as possible, we study the behavior of existing time-synchronization approaches in terms of synchronization error and control packet exchanging procedure. LESSAR is designed according to several key observations based on the analysis. We observed that:

- The constant part of PADR can easily cancel out each other by time-synchronization algorithms that use two-way message exchange approaches. Therefore, two-way message exchange approaches can generate more accurate time-synchronization than one-way message exchange approaches. As a result, in LESSAR, two-way message exchange approach is applied to help nodes to eliminate the influence of PADR delays. Under this condition, the error of estimated PADR delay equals to $|\frac{PADR_{sa}(t_{s1})-PADR_{as}(t_{a1})}{2}|$. If $PADR_{sa}(t_{s1}) \approx PADR_{as}(t_{a1})$, the error equals to zero.

- In general WSNs, accessing delay dominate the PADR delay. The accessing delay is composed of queuing delay, carrier sense time, channel back-offs, and time to transmit preamble bytes. Although by increasing the priority of time-synchronization packets, the queuing delay and channel back-off delay can be significantly decreased, the remaining accessing delay is still in the magnitude of macro-second to milli-second. while the magnitudes of common processing delay, propagation delay, and receiving delay are shorter than 1 macro-second. By applying faster processing units, using separate channels for time-synchronization, measuring time during radio operations, their value can be further reduced.

- When the propagation delay is not the dominant term in the PADR delay, The relationship between the value of PADR and the distance between node S and node R is weak. Values of processing delay, accessing delay, and receiving delay depend on the current workload of node R and S, the implementation of their Media Access Control (MAC) layer protocols, as well as the capability of their microprocessors and communication radios. In a homogeneous network, the distributions of processing delay, accessing delay, and receiving delay of one node pair are similar to those of other node pairs. Therefore, the estimated PADR between node S and node R_1 has the same distribution as the PADR estimation between node S and node R_2. As a result, a proper estimated PADR for one pair of nodes can be applied to other pairs in the network. Consequently, in LESSAR, instead of estimating PADR delay between all node pairs, we estimate PADR delays of several selected node pairs and use the estimation results for synchronize other node pairs. This indirect PADR estimation approach can generate almost similar estimation accuracy while significantly reduce the amount of traffic overhead.

Level Discovery

LESSAR maintains a global time in a WSN by organizing the system into levels. Level discovery is performed at the initial time when the network is deployed. A gateway node is the root of the network. It is assigned to be the level 0 node in the structure and broadcasts a level discovery packet to its neighbors. The nodes who receive this packet are assigned to be level 1 and broadcast their own level discovery packet containing the new level values to their neighbors. A node may receive several level discovery packets and it always accepts the one with the lowest level as its ancestor and takes this value plus 1 as its own level. Then the broadcasting operation continues. Finally, all nodes are constructed in a hierarchical network topology. When a new node joins an established network, it broadcasts a level request packet to ask its neighbors for their current level values. From those responses, it chooses the smallest one plus 1 as its own level. The children of a node can discover the failure of a node when the timer of keep-alive messages from that node expires. These nodes then broadcast level request packet and redo the level discovery progress again. In this way, the whole network can be set up in a hierarchical structure. A node at an upper level means it is closer to the gateway node in terms of number of hops.

In LESSAR, nodes will be synchronized level by level. Each node believes that the clocks in its upper level are more accurate than its local clock and try to synchronize with them. It accepts the time sync packets from parent node and drops all other time sync packets from nodes in its lower level and peers in the same level. Finally, the whole WSN will follow the clock of the gateway node, which could be adjusted by NTP in the upper-layer service network or obtained directly from GPS signals.

LESSAR Algorithm in Two Adjacent Levels

We observed that there is correlation in the PADR errors across multiple receivers of the same sequence of synchronization packets. Like NTP, LESSAR estimates the PADR delays by the well-known two-way message exchange method. However, to reduce the amount of synchronization packets across the networks, only one node is required to do two-way message exchange to estimate PADR delays for its peers in the same level. In this way, LESSAR is still able to achieve the accuracy limitation but decrease the number of time synchronization packets.

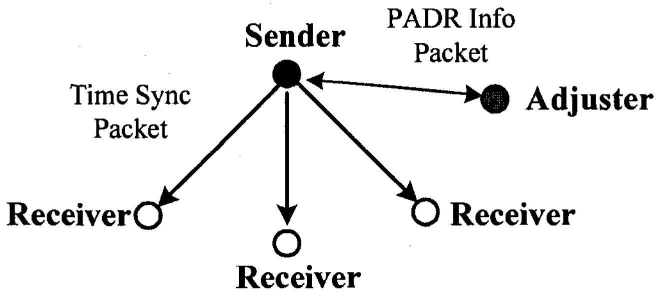

Figure 4.22: The two-level structure of LESSAR

Let us consider a two-level localized view of WSN, which is presented in Fig.4.22. The nodes at level $i+1$ are receivers and the node at level i is their sender. LESSAR selects one node from the receivers as an adjuster for these receivers. All the receivers try to synchronize their local clocks with the sender based on the PADR delays estimated by the two-way message exchanges between sender and adjuster.

The sender periodically broadcasts the time stamped time sync packet to all the receivers and the adjuster. Each node records the receiving time according to its local clock when it receives the packets. Only the adjuster puts its timestamps into the PADR info packet and sends it back to the sender. After two rounds of such operations, the sender has two answers from the adjuster and is able to estimate the PADR delays by Eq. 4.30 and 4.31. Then, the sender puts the estimations into the next time sync packet and broadcasts it to all the receivers. Because the adjuster is selected from the receivers in the same level, they all get the same sequence of time sync packets from the same sender and work in the similar physical environment, the differences of their PADR delay would be small enough to be taken as part of $PADR_r$ described in our above discussions. In this way, any receiver can take the adjuster's PADR estimation as its own to synchronize its local clock with the sender.

Adjuster Selection in the Overall Network

Selecting an adjuster for each level is straightforward: LESSAR always chooses one of the senders in the lower level as the adjuster for its upper level. The gateway node, root of the network, is the sender of level zero. At the edge level, the sender with the longest lifetime or the maximum amount of remaining battery power will be selected as the adjuster. There are existing studies on how to build a network tree [16, 15]. Therefore, LESSAR can view the wireless network based on hierarchical tree structure. In this tree structure, each none-leaf node acts as the sender for its lower level and the adjuster candidate for the current level. Only these nodes perform two-way messages exchanging jobs to estimate the PADR. All the leaf nodes are receivers and synchronize their local clocks by every time sync packet received from their parent nodes.

Synchronization Method and Errors

For one receiver, it draws the estimated time line based on the PADR estimation value from its adjuster in the latest two time sync packets and synchronizes its local clock. Fig.4.23 illustrates this approach.

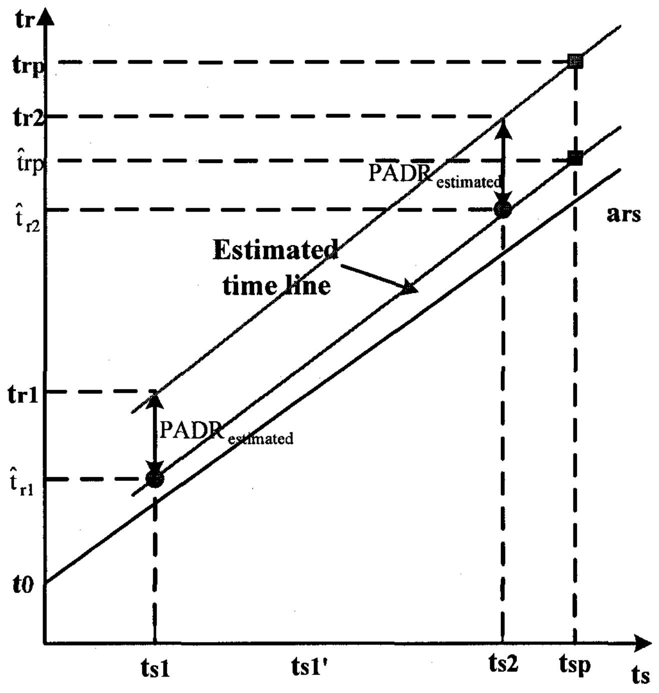

Figure 4.23: The procedure of receiver R estimating the accurate t_{sp} when its local clock is t_{rp} by using the latest estimated time line generated from the previous estimations of the PADR delays

The estimated time line goes through the points $(t_{s1}, \hat{t}_{r1})$ and $(t_{s2}, \hat{t}_{r2})$ with:

$$\hat{t}_{r1} = t_{r1} - PADR_{estimated}(t_{s1}) \tag{4.38}$$

$$\hat{t}_{r2} = t_{r2} - PADR_{estimated}(t_{s2}) \tag{4.39}$$

If there is an event happening at t_{rp} (and t_{sp} according to S's clock), the receiver would time stamp this event as $\hat{t}_{rp}$:

$$\hat{t}_{rp} = t_{rp} - \frac{PADR_{estimated}(t_{s1}) * (t_{r2} - t_{rp}) + PADR_{estimated}(t_{s2}) * (t_{s1} - t_{r1})}{t_{r2} - t_{r1}} \quad (4.40)$$

From Eq. 4.40, it can be found that the accuracy that LESSAR can achieve depends on the difference between the PADR delays estimated by the adjuster and the accurate PADR delays this receiver should have when it receives a synchronization packet from a sender. There exist two sources of errors: (i) the difference between the constant part $PADR_c$ of the adjuster and the receiver; (ii) the difference between the random part $PADR_r$ of the adjuster and the receiver. If the adjuster and the receiver are perfectly the same, the first part of the error is the difference between their distances to the sender, which is denoted as $\triangle Distance/C$, where C is the speed of light. Take Berkley Mote as an example, the maximum $\triangle Distance$ of two nodes is only several hundred feet, and $\triangle Distance/C$ is less than 100 ns. So the dominant error affects the accuracy of LESSAR is the random $PADR_r$.

4.2.5 Performance Evaluations

We implemented and tested LESSAR with OMNeT++ [5]. To mimic the behaviors of practical event sensing applications, he takes the hardware specifications of Berkeley Mote to model nodes in our simulation. The random $PADR_r$ are assumed to follow uniform distribution as bounded $PADR_r$. In addition, the distances between nodes in any two levels are randomly assigned from 10 feet to 100 feet.

We have tested the effect of the random $PADR_r$to the synchronization accuracy. Fig.4.24 presents the error distribution of both the simulation results and Eq. (4.16) when $PADR_r$ follow uniform distribution, $f(x) = \frac{1}{100E-6}$ where $-50us \leq x \leq 50us$.

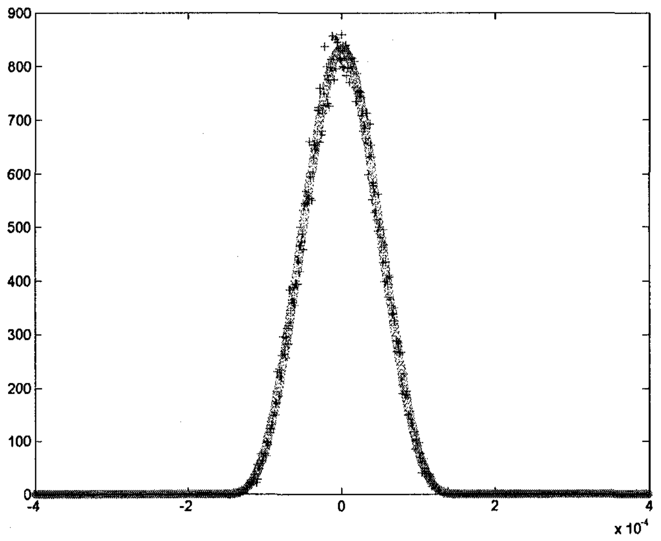

Figure 4.24: An error distribution of simulation results (in blue cross) and theoretical analysis results (in red circle) with uniformly bounded $PADR_r$. The x-axis represents the scale of the errors and the y-axis represents the number of results distributed in each scale

In this figure, the interval for the sender sending out time sync packets is 1s, and the number of samples is 100,000, i.e. the simulation time is about 1 day and 4 hours. It has been observed that the simulation result matches our theory analysis very well. The occurred maximum error of LESSAR is 149.87us that is very close to the theory result $3b$, which is 150us.

A quantile-quantile plot, shown as Fig.4.25, is also utilized to compare the distribution of the above two curves in Fig.4.24. The curve in Fig.12 is nearly a line with the slop that equals to 1. That also proves that the two distributions of the simulation results and the theory analysis are the same. This result justifies that our theoretical analytical model can be used to study the performance of time synchronization protocols. Although only LESSAR has been studied in the simulation, our model can be easily applied to other existing synchronization protocols with linear

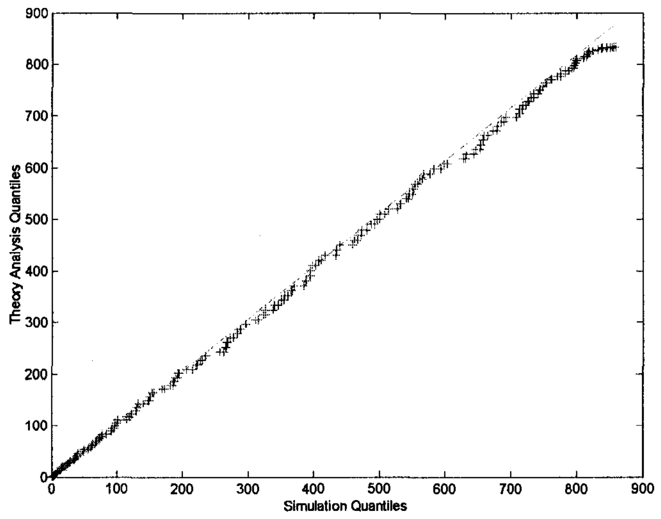

Figure 4.25: The quintile-quintile relationship of the simulation and theoretical analysis results

assumptions.

4.2.6 Discussion

In this study, we proposed lESSAR, a light-weight time synchronization approach to synchronizing clocks of the lower-tier wireless sensor networks in WASNs. Network time synchronization is done by exchanging time messages between time client and time server. Compared to the existing solutions, LESSAR has a new way to collecting clock information by only requiring selected nodes rather than all the nodes to perform two-way message exchanges. In this way, LESSAR can significantly reduce the traffic of synchronization messages by sacrificing the synchronization accuracy. The correctness of LESSAR is based on an assumption of that sensors in the same level in the networks are deployed in the similar physical environment, which is true in most WSNs applications. Further comparisons between LESSAR and the existing synchronization approaches for WSNs can be found in [96].

Chapter 5

Conclusions

5.1 Conclusions and Major Contributions

In this dissertation, I proposed a set of dynamic multicast and time synchronization protocols for supporting the spatiotemporal event surveillance applications in Wide-Area Sensor Networks (WASNs). WASNs have a two-tier infrastructure: *i*) the lower tier consists of multiple Wireless Sensor Networks (WSNs) which can be deployed in interested sensing areas that may be geographically disconnected; and *ii*) the upper tier that consists of mobile data collectors which forward sensory task assignments and event reports back-and-forth between the central data center and the distributed sensor nets. The data collectors form the so-called Intermittently Connected Networks (ICNs). The major contributions include:

- I proposed OS-Multicast, a new one-to-many data communication approach for ICNs to distributing event queries from the data center to multiple mobile data collectors. OS-Multicast dynamically builds up the multicast structure according to the current network situations. It forwards data whenever there is a discovered opportunity to reach the destinations. Our results show that OS-Multicast doubles the message delivery ratio than DTBR, which is the first dynamic multicast approach proposed for ICNs, when the total unavailability of contacts is more than 70% of the overall system operation time. OS-Multicast is actually a controlled-flooding approach. The performance gain achieved by OS-Multicast comes from the additional redundant duplicates of messages generated along with the creation of the dynamic multicast structure in OS-Multicast. Therefore, OS-Multicast has worse message deliver efficiency compared to DTBR. We recommend system designers choose OS-Multicast to provide multicast service in ICNs when the source traffic rate is low.

- I proposed PTNT for mobile data collectors to collect the observed event reports from the sensor networks. PTNT is simple to be implemented in sensor nodes and is capable of tracking the movement trace of mobile sinks without requiring GPS information. PTNT gives system designers the flexibility of adjusting the event query broadcast frequency so that they can make a tradeoff between the message delivery efficiency and average message delay. The performance of PTNT highly rely on how frequent the queries are flooded to the overall networks so that the distance from each sensor to the current positions of the mobile data collectors can be updated. When every query is flooded, PTNT will always select the shortest path to forward messages. In this case, it achieves the smallest delay but has the worst message delivery efficiency caused by flooding the queries.

- I proposed DTP to provide synchronized clocks in ICNs. It is important to have accurate timestamps for both the spatiotemporal event queries and the reports. DTP is a distributed time synchronization approach. It reduces the average time-synchronization error by about half comparing to NTP-Core, which models the fundamental synchronization message exchange method of NTP which is the Internet time standard for decades. The performance gain achieved by DTP comes from the additional time information it collects by doubling the number of synchronization messages in each message exchange sliding window, as well as by applying a new way of performing linear regression of the synchronization messages. DTP requires can also be implemented in the current architecture of NTP with minor modifications.

- I also proposed LESSAR to keep clocks in WSNs synchronized with the data collectors. LESSAR is a lightweight approach to addressing the concern of limited energy in WSNs by only requiring the selected nodes, i.e. the adjusters, rather than requiring all the nodes to conduct two-way time message exchange with the time servers. Thus, LESSAR always uses less synchronization traffic than TPSN to achieve similar time accuracy, which operates using massive two-way synchronization message exchanges among sensors. TPSN is the default time protocol used in TinyOS, i.e. the operating system of off-the-shelf wireless sensors. However, LESSAR sacrifices the synchronization accuracy because of using the clock estimations of the selected adjusters to adjust clocks of the other nodes. The correctness of LESSAR is based on assumptions of that the same sensor nodes have the very similar oscillator and nodes located in the same level in the networks are deployed in the similar physical environment. These assumptions are true in most WSNs applications.

Performance evaluations of the above protocols are conducted either by simulations or by theoretical analyses based on the simplified network models. Thus, the correctness of our study is limited by the correctness of the simulation scenarios and the simplified models. In this dissertation, a lot of efforts are done to improve the correctness of our results such as by applying the trace files collected from real-world ICN testbeds to study the dynamic multicast problem and by using the widely accepted linear clock model to study the time synchronization problem. However, it has been reported in [41, 87] that there are several difficulties and unsolved issues of simulating the Internet and the mobile ad hoc wireless networks. Moreover, [22] argues that the radio propagation model used in ns2 is inappropriate to reflect the real-world scenarios. Also, the linear clock model is correct only when clocks are operated in room temperature. How to further improve the correctness of the performance study in this dissertation is an important part of our future work.

5.2 Future Work

To extend the research in this dissertation, we will study the optimal scheduling issue of the dynamic multicast problem in WASNs, which targets on delivering the multicast data to destinations in the smallest work span. Actual implementation of PTNT in commercial wireless embedded sensors needs to be done to study how to tune the protocol parameters to achieve the best performance in terms of the joint-optimization of the message delivery efficiency and the average delays. We will also implement the idea of DTP using the current architecture of NTP to investigate its performance in the real-world experiments.

Bibliography

[1] Crystal oscillator general information and product selection. *http://www.oscilent.com/catalog/Category/crystaloscillator.htm.*

[2] Glomosim network simulator. *http://pcl.cs.ucla.edu/projects/glomosim/.*

[3] Mica2 datasheet. *http://www.xbow.com/Products/productsdetails.aspx?sid=93.*

[4] Network simulator 2. *http://www.isi.edu/nsnam/ns/.*

[5] Omnet++ simulator. *http://www.omnetpp.org/.*

[6] K.C. Almeroth. The evolution of multicast: From the mbone to interdomain multicast to internet2 deployment. *IEEE Network*, 14:10–20, January 2000.

[7] X Tony Ballardie, Paul Francist, , and Jon Crowcroft. Core based tree: An architecture for scalable inter-domain multicast routing. In *preceeding of ACM SIGCOM'95*, pages 85–95, 1995.

[8] R. Bhatia and L. Li. Characterizing achievable multicast rates in multi-hop wireless networks. In *Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networks (MobiHoc'05)*, May 2005.

[9] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. *IEEE Infocomm'06*, April 2006.

[10] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking - an approach to interplanetary internet. *IEEE Communications Magazine*, June 2003.

[11] B. Burns, O. Brock, and B. Neil Levine. Autonomous enhancement of disruption tolerant networks. In *Proceedings of IEEE International Conference on Robotics and Automation*, May 2006.

[12] B. Burns, O. Brock, and B.N. Levine. Mv routing and capacity building in disruption tolerant networks. In *Proceedings of IEEE INFOCOM*, March 2005.

[13] Stephen Casner and Stephen Deering. First ietf internet audiocast. *ACM Computation Communication Review*, pages 92–97, July 1992.

[14] V. Cerf. Delay tolerant network architecture. *draft-irtf-dtnrg-arch-04.txt*, December 2006.

[15] A. Cerpa and D. Estrin. Ascent: adaptive self-configuring sensor networks topologies. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2002)*, New York, NY, USA, June 2002.

[16] J. Chen, J. Postel, and D. Subramanian. A new approach to routing with dynamic metrics. In *Proceedings of the IEEE INFOCOM '99*, New York, NY, March 1999.

[17] W. Chen, J. C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, 2003.

[18] L. Cheng and I. Marsic. Accurate bandwidth measurement in xdsl service networks. *Computer Communications (Elsevier)*, 2002.

[19] M. Chuah, L. Cheng, and B.D. Davison. Enhanced disruption and fault tolerant network architecture for bundle delivery (edify). In *Proceedings of IEEE Globecom*, November 2005.

[20] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3, 1989.

[21] Subir Kumar Das, B. S. Manoj, and C. Siva Ram Murthy. A dynamic core based multicast routing protocol for ad hoc wireless networks. In *Proceedings of the third ACM international symposium on Mobile ad hoc networking & computing (MobiHoc'02)*, 2002.

[22] Calvin Newport David Kotz and Chip Elliott. The mistaken axioms of wireless-network research. *Dartmouth College Computer Science Technical Report TR2003-467*, 2003.

[23] Stephen Deering and D. Cheriton. Multicast routng in datagram internetworks and extended lans. *ACM Transcations on Computer Systems*, pages 85–111, May 1990.

[24] Stephen Deering, D.Estrin, D. Farinacci, V. Jacobson, G. Liu, and L. Wei. Pim architecture for wide-aree multicast routing. *IEEE/ACM Transcation on Networking*, pages 153–162, April 1996.

[25] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *Proceedings of the 2001 International Parallel and Distributed Processing Symposium (IPDPS'01)*, pages 1965–1970, San Francisco, CA, USA, April 2001. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing.

[26] D. Estrin and et al. Protocol independent multicast sparce mode (pim-sm): Protocol specification. *IETF RFC 2362*, 1998.

[27] A. Cerpa et al. Habitat monitoring: Application driver for wireless communications technology. In *Proceeding of 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[28] Cerf. V. et al. Delay-tolerant network architecture. *IETF RFC 4838*, 2007.

[29] J. Aslam et al. Tracking a moving object with a binary sensor network. In *Proceedings of ACM 1st Conference on Embedded Networked Sensor Systems (SenSys'03)*, 2003.

[30] Lorincz et al. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing, Special Issue on Pervasive Computing for First Response*, 2004.

[31] Q. Huang et al. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *Proceedings of IEEE Information Processing in Sensor Networks (IPSN'03)*, 2003.

[32] Q. Huang et al. Spatiotemporal multicast in sensor networks. In *Proceeding of ACM Sensys'03*, 2003.

[33] S. Madden et. al. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data.* ACM Press, 2003.

[34] V. Kumar et al. Robot and sensor networks for first responders. In *Proceedings of IEEE Pervasive Computing*, 2004.

[35] V. Mehta et al. A bluetooth based sensor network for civil infrastructure health monitoring. *Wireless Networks*, 10(4), 2004.

[36] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of SIGCOMM'03*, August 2003.

[37] K. Fall. Messaging in difficult environments. *Intel Research Berkeley, IRB-TR-04-019*, December 2004.

[38] Rui Fan and Nancy Lynch. Gradient clock synchronization. In *Proceedings of the Twenty-third Annual ACM PODC.* ACM Press, 2004.

[39] S. Farrell and V. Cahill. Delay and disruption tolerant networking. 2006.

[40] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald. When tcp breaks: Delay- and disruption-tolerant networking. *IEEE Internet Computing*, 10, 2006.

[41] S. Floyd and V Paxson. Difficulties in simulating the internet. *ACM Trasction on Networking*, 2001.

[42] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of IEEE Sensys'03*, Los Angel, CA., November 2003.

[43] Mario Gerla, Ching-Chuan Chiang, and Lixia Zhang. Tree multicast strategies in mobile, multihop wireless networks. *Mobile Networks and Applications*, 4:193–207, 1999.

[44] C. Gui and P. Mohapatra. Efficient overlay multicast for mobile ad hoc networks. In *Proceedings of WCNC*, 2003.

[45] O. Gurewitz, I. Cidon, and M. Sidi. Network time synchronization using clock offset optimization. In *Proceedings of 11th IEEE International Conference on Network Protocols (ICNP'2003)*, pages 212–221, November 2003.

[46] O. Gurewitz, I. Cidon, and M. Sidi. Network time synchronization based on clock offset optimization. *IEEE Trasaction On Networking*, 2006.

[47] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proceeding of Sigcomm Workshop in DTN*, August 2005.

[48] L. Girod J. Elson and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'2002)*, Boston, MA, December 2002.

[49] S. Jain, M. Demmer, R. Patra, and K. Fall. Using re-dundancy to cope with failures in a delay tolerant network. In *Proceedings of SIGCOMM'05*, August 2005.

[50] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. In *Proceedings of SIGCOMM'05*, August 2005.

[51] S. Jain and R. Patra K. Fall. Routing in a delay tolerant networking. In *Proceedings of SIGCOMM'04*, August 2004.

[52] D.B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. *In Mobile Computing*, pages 153–181, 1996.

[53] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proceddings of International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107, San Jose, CA, October 2002.

[54] Luo Jun, Panchard Jacques, Piorkowski Michal, Grossglauser Matthias, and Hubaux Jean-Pierre. Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (Systems track) - DCOSS'06*, June 2006.

[55] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon. Minimum energy asynchronous dissemination to mobile sinks inwireless sensor networks. In *Proceddings of the 1st ACM SenSys, 2003(Sensys'03)*, Los Angeles, November 2003.

[56] S. Kim, S. H. Son, J. A. Stankovic, S. Li, and Y. Choi. Safe: A data dissemination protocol for periodic updates in sensor networks. *IEEE Communications Letters*, 2004.

[57] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proceeding of ACM SIGCOMM'00*, Augest 2000.

[58] S-J. Lee, W. Su, and M. Gerla. Wireless ad hoc multicast routing with mobility prediction. *ACM/Kluwer Mobile Networks and Applications*, 2001.

[59] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2), March 2004.

[60] A. Lindgreny, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proceedings of the First International Workshop on Service Assurance with Partial and Intermittent Resources (SAPIR 2004)*, Augest 2004.

[61] Xin Liu, Qingfeng Huang, and Ying Zhang. Combs, needles, haystacks: Balancing push and pull for discovery in largescalesensor networks. In *Proceeding of the 2nd International Conference on Embedded Networked Sensor Systems (Sensys 04)*, 2004.

[62] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proceddings of International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[63] R. Malladi and D.P. Agrawal. Current and future applications of mobile and wireless networks. *Communications of the ACM*, 45, 2002.

[64] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[65] D. Mills. Network time protocol (ntp) general overview. *http://www.cis.udel.edu/ mills/database/brief/overview/overview.ppt*, May 2007.

[66] D.L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[67] D.L. Mills. Network time protocol (version 3) specification, implementation and analysis, (rfc) 1305. In *Internet Engineering Task Force.* March 1992.

[68] D.L. Mills. Cryptographic authentication for real-time network protocols. *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 45:135–144, 1999.

[69] N. Minar. A survey of ntp network. *http://www.media.mit.edu/ nelson/research/ntp-survey99/*, 1999.

[70] J. Moy. Multicast extensions to ospf. *IETF RFC 1584*, 1994.

[71] Alex Pentland, Richard Flecher, and Amir Alexander Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, pages 4–9, January 2004.

[72] C. E. Perkins and P. Bhagwat. Highly dynamic destination sequenced distance vector routing (dsdv)for mobile computers. *Proceedings of Acm Sigcomm'94*, pages 234–244, 1994.

[73] C.E. Perkins and E.M. Royer. Ad-hoc on demand distance vector routing. *Proceedings of The 2nd IEEE Workshop on Mobile Computing Systems And Applications*, pages 90–100, 1999.

[74] Disruption Tolerant Networks program. *http://www.darpa.mil/ato/solicit/dtn/*, November 2006.

[75] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the Fifth Annual*

ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'99), pages 207–218, Seattle, Washington, August 1999.

[76] William Su Sang Ho Bae, Sung-Ju Lee and Mario Gerla. The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks. *IEEE Network*, 7:70–77, January 2000.

[77] A. Sheth, B. Shucker, and R. Han. Vlm2: A very lightweight mobile multicast system for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC) 2003*, 2003.

[78] M. L. Sichitiu and C. Veerarittiphan. Simple accurate time synchronization for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'2003)*, New Orleans, LA, March 2003.

[79] Fabio Silva, John Heidemann, Ramesh Govindan, and Deborah Estrin. Directed diffusion for wireless sensor networking. *USC Tech Report-04*, 2004.

[80] T. Spyropoulos, K. Psounis, and C. Raghavendra. Performance analysis of mobility-assisted routing. In *Proceedings of the Seventh ACM international Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '06)*, Florence, Italy, May 2006.

[81] T. Spyropoulos, K. Psounis, and C.S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceeding of Sigcomm'05 Workshop in DTN*, Augest 2005.

[82] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *Journal of ACM*, 34, July 1987.

[83] Affan Syed and John Heidemann. Time synchronization for high latency acoustic networks. In *Proceeding of IEEE INFOCOM'06*, April 2006.

[84] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. *Duke Technical Report CS-2000-06*, July 2000.

[85] Upkar Varshney. Multicast over wireless networks. *Communications of ACM*, 45:31–37, December 2002.

[86] Iuliu Vasilescu, Keith Kotay, Daniela Rus, Matthew Dunbabin, and Peter Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proceddings of ACM Sensys'05*, pages 154–165, San Diego, CA, November 2005.

[87] Jonghyun Kim Vinay Sridhara and Stephan Bohacek. Models and methodologies for simulating mobile ad- hoc networks. In *Proceddings of The 3rd IEEE International Workshop on Mobility Management and Wireless Access*, November 2005.

[88] D. Waitzman, C. Partridge, and S. Deering. Distance vector multicast routing protocol (dvmrp). *IETF RFC 1075*, 1998.

[89] Y. Wang, M. Martonosi S. Jain, and K. Fall. Erasure coding based routing for opportunistic networks. In *Proceeding of Sigcomm Workshop in DTN*, August 2005.

[90] K. Whitehouse, F. Jiang, A. Woo, C. Karlof, and D. Culler. Sensor field localization: a deployment and empirical analysis. *Technical Report UCB//CSD-04-1349*, April 2004.

[91] Terry Wilson and Kelly Carroll. Polenet,http://www.polenet.org/polenetscience.htm.

[92] J. Xie, R.R. Talpade, A. Mcauley, and M.Y. Liu. Amroute: ad hoc multicast routing protocol. *Mobile Networks and Applications*, 7:429–439, 2002.

[93] Hung Le Xuan and Sungyoung Lee. A coordination-based data dissemination protocol for wireless sensor networks. In *Proceedings of the IEEE First International Conference on Sensors, Sensor Networks and Information Processing (ISSNIP04), 2004*, November 2004.

[94] H. Yang and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications (SNPA '03)*, 2003.

[95] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large scale wireless sensor networks. In *Proceddings of the 8th ACM MobiCom, 2002(Mobicom'02)*, Atlanta, September 2002.

[96] Q. Ye and L. Cheng. Time synchronization in wireless sensor networks. In *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks.* Auerbach Publications, August 2005. ISBN: 0849328322.

[97] Q. Ye, L. Cheng, M. Chuah, and B. Davison. Os-multicast: On-demand situation-aware multicasting in disruption tolerant networks. In *Proceedings of the IEEE VTC2006-Spring*, Melbourne, Australia, May 2006.

[98] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM96*, page 594C602, November 1996.

[99] X. Zhang, G. Neglia, and J. Kurose. Performance modeling of epidemic routing. In *Proceedings of the IFIP Networking 2006*, May 2006.

[100] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys and Tutorials*, January 2006.

[101] W. Zhao and M. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proceedings of IEEE Workshop on Future Trends in Distributed Computing Systems*, May 2003.

[102] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proceedings of ACM Mobihoc*, May 2004.

[103] W. Zhao, M. Ammar, and E. Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *Proceedings of IEEE INFOCOM'05*, March 2005.

[104] W. Zhao, M. Ammar, and E. Zegura. Multicasting in delay tolerant networks: semantic models and routing algorithms. In *Proceeding of Sigcomm'05 Workshop in DTN*, August 2005.

VITA

Qing Ye received his B.S. and M.S. degrees in the Automation Department of Tsinghua University, Beijing, China in 1999 and 2002, respectively.

Since January 2003, he has been working towards his Ph.D. at Lehigh University under the guidance of Professor Liang Cheng in the Laboratory Of Networking Group (LONGLAB). Since then he has conducted research in the field of wireless sensor networks and wireless disruption tolerance networks.

During his dissertation study, he has published more than 10 technical papers. He is a student member of IEEE.