uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Richard Werner Nelem Pazzi

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Design and Performance Evaluation of Routing and Streaming Protocols for Wireless Multimedia Sensor Networks

TITRE DE LA THÈSE / TITLE OF THESIS

Azzedine Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Hossam Hassanein

Emil Petriu

Hussein Mouftah

Shikharesh Majumbar

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Design and Performance Evaluation of Routing and Streaming Protocols for Wireless Multimedia Sensor Networks

by

Richard Werner Nelem Pazzi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Computer Science
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-50750-6
Our file Notre référence
ISBN: 978-0-494-50750-6

Abstract

The potential applications enabled by unattended small wireless sensor devices, capable of collecting information from the environment and organizing themselves in an ad hoc manner to relay data to a remote observer, have motivated the research work presented in this thesis dissertation. Wireless sensor network (WSN) applications can range from military, surveillance and emergency response to health care, traffic control, and vital body conditions monitoring. In addition, the recent development of low-cost wireless multimedia hardware has enabled the integration of multimedia capabilities into WSNs. The research of wireless multimedia sensor networks (WMSNs) is a relatively new field and it is envisioned that it will enhance several existing classes of WSN applications and motivate the development of new and ambitious ones.

In this thesis dissertation, we focus upon the design and performance evaluation of a suite of protocols for wireless multimedia sensor networks: data routing in wireless sensor networks and interactive multimedia streaming in a wireless multimedia sensor networks. To this end, we propose the design of low-latency and reliable routing protocols for WSNs that aim simplicity and efficiency, and we propose and evaluate a hybrid mobile data gathering protocol for WSNs that should not only reduce data delivery latency, but also contribute to alleviating the bottleneck problem at static data aggregation points. We also propose a suite of protocols that enable interactive multimedia streaming over wireless multimedia sensor networks for thin mobile devices. We present our protocols, discuss their implementation, and report on their performance evaluation based on an extensive set of simulation experiments.

List of Publications

The following publications by the author are relevant to the work in this thesis.

Journals:

- Pazzi, R. W. and A. Boukerche. "Mobile Data Collector Strategy for Delay-Sensitive Applications over Wireless Sensor Networks". *Elsevier's Computer Communication Journal*, Volume 31, Issue 5, pp. 1028-1039, March 2008.
- Boukerche, A. and Pazzi, R. W. "An End-to-End Virtual Environment Streaming Solution over Heterogeneous Networks for Thin Mobile Devices". Accepted in the *Elsevier's Computer Communication Journal*, 2008.
- Pazzi, R. W. and A. Boukerche. "Implementation, Measurement and Analysis of an Image-Based Virtual Environment Streaming Solution for Thin Devices over Wireless Networks". Accepted in the *IEEE Transactions on Instrumentation and Measurements Journal*, 2008.
- Boukerche, A., Pazzi, R. W, Araujo, R. "Fault-tolerant wireless sensor network routing protocols for the supervision of context-aware physical environments". *Journal of Parallel and Distributed Computing*, v. 66, n. 4, pp. 586-599, April 2006.

Conference papers:

- Boukerche, A. and Pazzi, R. W. "A Peer-to-Peer Approach for Remote Rendering and Image Streaming in Walkthrough Applications". *In Proceeding of the IEEE International Conference on Communications* - (ICC'07), pp. 1692-1697, 2007.
- Boukerche, A. and Pazzi, R. W. 2007. "Lightweight Mobile Data Gathering Strategy for Wireless Sensor Networks". *In Proceedings of the* 9^{th} *IFIP/IEEE International Conference on Mobile and Wireless Communication Networks* - MWCN, 2007.
- Boukerche, A. and Pazzi, R. W. 2006. "Remote rendering and streaming of progressive panoramas for mobile devices". *In Proceedings of the* 14^{th} *Annual ACM international Conference on Multimedia* - (MULTIMEDIA'06), pp. 691-694, 2006.
- Boukerche, A., Pazzi, R. W, Huang, T . "A Protocol for Interactive Streaming of Image-based Scenes over Wireless Ad-hoc Networks". *In Proceedings of the IEEE International Conference on Communications* (ICC'06), v. 8. pp. 3621-3626, 2006.
- Boukerche, A., Pazzi, R. W., Huang, T. "A Client-Server Approach to Enhance Interactive Virtual Environments on Mobile Devices over Wireless Ad Hoc Networks". *In: European Conference on Parallel Computing (EURO-PAR'06), Lecture Notes in Computer Science* (LCNS), v. 4128. p. 981-991, 2006.

- Boukerche, A. and Pazzi, R. W. "Performance evaluation of a streaming based protocol for 3D virtual environment exploration on mobile devices". *In Proceedings of the* 9^{th} *ACM international Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems* - (MSWiM'06), pages 20-27, 2006.

- Boukerche, A. and Pazzi, R. W. "Scheduling and buffering mechanisms for remote rendering streaming in virtual walkthrough class of applications". *In Proceedings of the* 2^{nd} *ACM international Workshop on Wireless Multimedia Networking and Performance Modeling* - (WMuNeP'06), p. 53-60, 2006.

- Boukerche, A., Pazzi, R. W., Kazem, F. "Design and Implementation of a Rate Control Mechanism for Image-based Virtual Exploration over Wireless Networks". In Proceedings of the 31^{st} IEEE Conference on Local Computer Networks, p. 905-912, 2006.

- Boukerche, A., Pazzi, R. W., Araujo, R. "HPEQ A Hierarchical Periodic, Event-driven and Query-based Wireless Sensor Network Protocol". *In Proceedings of The IEEE Conference on Local Computer Networks* 30^{th} *Anniversary* (LCN'05), p. 560-567, 2005.

- Boukerche, A., Huang, T., Pazzi, R. W. "A real-time transport protocol for image-based rendering over heterogeneous wireless networks". *In Proceedings of the* 8^{th} *ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM'2005), pp. 333-340, 2005.

Acknowledgements

It is difficult to put into words my gratitude to my friend and Ph.D. supervisor, Dr. Azzedine Boukerche. His enthusiasm and motivation have helped to make my Ph.D. experience much more interesting and productive. He has provided me with encouragement, inspiration, priceless advices and friendship, resources and financial support. I would not have been able to succeed in my career without this extraordinary person. Professor Boukerche, I am in debt with you.

I would like to thank all members of the PARADISE Laboratory at the University of Ottawa for their companion and support during all these years. We formed a big family and I will never forget them.

I wish also to thank the support and administrative staff at the University of Ottawa, from fixing systems and delivering hardware to helping me with documents and guidance.

Lastly, and most importantly, I wish to thank my parents, Maria Isabel Nelem Pazzi and Ronald Pazzi, my sister, Danielle Werner Nelem Pazzi, and my wife, Stela Marina Braz Pazzi. I would be lost without their emotional support and love, and for believing in my success. I dedicate this thesis to my parents for giving me life, and to Stela for sharing it with me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Preliminary Considerations

Wireless sensor networks have attracted the attention of either the research community and the industry in the last few years. One of the main reasons for the current research interest and rapid development of wireless sensor networks (WSNs) is the potential pervasive applications enabled by small unattended wireless devices capable of collecting information from the environment and that organize themselves in an ad hoc manner to relay the gathered data to a remote observer or base station. A plethora of applications are envisioned that will exploit these low-cost, large-scale and self-organizing class of wireless ad hoc networks. The applications range from military, surveillance and emergency response to health care, traffic control, and vital body conditions monitoring [6][5]. The main challenges on the design and implementation of wireless sensor networks are the tradeoffs imposed by the different application requirements, sometimes conflicting, thereby making it difficult to come up with a definite network design. For instance, surveillance and emergency systems often require the sensor network to be fast, reliable, and always available. However, timely response, reliability, and extended network lifetime are conflicting requirements, *i.e.*, in order to provide low-latency and reliable sensor data delivery, a large number of nodes must be always on, which consumes more energy

than sensor network solutions for non-delay sensitive applications that receive data periodically. In the later case, the sensor nodes can save energy by turning their radios off and only waking up during data delivery periods, thereby increasing the network lifetime. Another challenge that is taken into consideration in all development layers is that wireless sensor nodes are resource-constrained devices in terms of energy, memory, and processing power. Depending on the application and deployment method, the WSN might be unattended for long periods and replacing batteries may be unfeasible.

Recently, the development of low-cost wireless multimedia hardware, *e.g.*, wireless micro-cameras and multimedia-capable wireless devices, has drawn the attention to the convergence of wireless sensor networks and multimedia streaming capabilities. Wireless multimedia sensor networks (WMSNs), *i.e.*, networked wireless devices capable of capturing, processing and transmitting images, audio and video, besides scalar data, have emerged as an important technology that will enhance existing wireless sensor network applications to a new level [7]. The envisioned applications range from multimedia surveillance sensor networks, advanced emergency response, health care, border protection, augmented reality systems, just to mention a few.

Nonetheless, this anticipated convergence of wireless sensor networks and multimedia streaming faces a dilemma: a sensor network comprises resource limited devices, and multimedia streaming shows bandwidth-hungry requirements. Hence, the core challenge in WMSNs is to provide effective solutions to relay video and other types of media, that are known to require high-bandwidth, over such constrained wireless networks. For instance, the MICAz, Telos, and SmartMesh-XT sensor motes offer a maximum data rate of 250kbps, which is low depending on the multimedia quality requirements of the application. In addition, the bandwidth problem is intensified when multi-hop communications are taken into consideration. In a dense wireless network, *e.g.*, sensor networks, the total capacity of a wireless link is decreased significantly due to interference with the transmissions of nearby nodes [75]. But there is light at the end of the tunnel. The research on multi-tiered network architectures for WMSNs [71][106] is the key compo-

nent to enable the development of applications for WMSNs. For instance, in a multi-tier WMSN, the lower tiers are responsible for sensing and other small tasks, leaving the more complex task, *e.g.*, video recording and encoding, to resource-rich multimedia devices at higher tiers.

1.2 Scope of this Thesis

In this thesis, we focus upon two key areas: data routing in wireless sensor networks and multimedia streaming in a wireless multimedia sensor network. We have observed that most of the existing sensor network techniques are designed considering non-delay-sensitive applications, giving more emphasis to energy savings. In the case of delay-sensitive applications, *e.g.*, surveillance or emergency response systems in indoor environments, low latency and reliability are preferred to energy savings. Therefore, we propose the design of fast and reliable routing protocols that aims at simplicity and efficiency. We also design, implement and test a mobile data gathering approach for wireless sensor networks to alleviate the issue of static single data aggregation points. In the second part of this thesis, we study the concepts of wireless multimedia sensor networks, its challenges and existing architectures, as well as the multimedia streaming research field. Almost all the research on multimedia over sensor networks consider the streaming of video, the so called "video sensor networks". We go further and design an interactive non-linear multimedia streaming technique for WMSNs. Our motivations come mainly from the potential applications that we conceptualize with the convergence of sensor networks and interactive multimedia. The design principles for the development of our interactive multimedia streaming techniques over WMSNs include the use of a low latency data delivery mechanism for sensor networks, the MDC/CPEQ protocol, in which the mobile entities have two roles: they collect the sensory data; and they are the multimedia receivers, *e.g.*, head-mounted display capable of communicating with the sensor network and with a rendering server. The rendering outputs are 2D images that

are streamed to the multimedia receivers. Hence, the interactive multimedia streaming over WMSNs comprises the design of multiple components: a multi-tier architecture, an interactive streaming protocol (ISP), an interactive image-based transport protocol, the image scheduling and buffering mechanisms, and our initial ideas of a virtual path prediction mechanism.

1.3 The Contributions of this Thesis

This thesis contributes with a suite of routing protocols and multimedia streaming for wireless sensor networks. The first protocol, PEQ (Periodic, Event-based, and Query-driven), is the base for the other two protocols and relies on a dissemination tree based on the hop count of the sensor nodes to relay data to a sink in a multi-hop fashion. It provides fault-tolerance by repairing broken routes using only a node's local information.

The second protocol is a cluster-based extension of PEQ, Cluster-based PEQ or simply CPEQ. It uses the residual energy of the nodes in a certain region to select the cluster-head and build its cluster. The cluster-head role is assigned in turns, and each node in the network can become a cluster-head. Thus, there is a periodic selection of new cluster-heads to guarantee that traffic and energy consumption is distributed uniformly among the nodes. CPEQ relies on PEQ to deliver data to a sink in a multi-hop communication pattern, instead of using direct communications like LEACH [53] does, in order to achieve a reasonable scalability.

The third protocol is a mobile data gathering technique called MDC/CPEQ. It inherits all the features of PEQ and CPEQ, plus an extra property: it supports mobility of sinks or "mobile data collectors" (MDCs) in order to alleviate the main problem of all multi-hop routing strategies with static sinks or base stations - the bottleneck at the sink's vicinity. In these terms, the routing protocols for wireless multimedia sensor networks proposed in this thesis are tailored for delay-sensitive applications that require fast response and reliability, but they can be also applied to other classes of applications.

This thesis also contributes with a multi-tier wireless sensor architecture and its protocols that can foster the development of multimedia sensor network applications. In addition, this research proposes a solution to enable detailed interactive multimedia on thin mobile devices for augmented realty applications over wireless multimedia sensor networks and wireless networks in general. As far as the author is concerned, this research work is one of the first attempts in this direction and it is expected that the preliminary discoveries presented here will contribute with the research community by leaving the foundations of an architecture that could be explored and enhanced for future augmented reality systems based on events collected by a wireless sensor network.

The major contributions of this thesis include the design and development of:

- A novel routing protocol for wireless sensor networks in which the routing decisions are performed locally to minimize the traffic. This protocol uses the publish/subscribe scheme that gives the sensor network application more flexibility. For instance, a non-delay-sensitive application can rely on the publish/subscribe mechanism to set a periodic data delivery session that will instruct all sensor nodes to switch their radios to the a sleep mode until the imminence of the data delivery session;

- An efficient and decentralized path recovery mechanism that performs locally to the problem and does not involve flooding or a centralized decision at the sink;

- A cluster-based scheme to minimize traffic and increase reliability. Based on our simulation experiment results, our cluster-based approach shows reasonable scalability and low latency;

- A mobile data gathering scheme for wireless sensor networks. It not only alleviates the single data aggregation point issue, but it also contributes to reducing packet losses, while keeping the latency to a minimum, differently from other passive mobile solutions that increase the latency drastically;

- A multi-tier architecture for wireless multimedia sensor networks and an interactive virtual environment streaming protocol based on remote rendering and image streaming in order to enable high quality virtual explorations on thin mobile devices over wireless channels;
- A buffering and scheduling mechanisms for the streaming of remote rendering outputs in order to fill the gap between video streaming and interactive image-based virtual environment streaming for wireless multimedia sensor networks.

1.4 Thesis Outline

This thesis dissertation begins with a general discussion of the concepts of wireless sensor networks and multimedia sensor networks, its applications, evolution and state-of-the-art wireless sensor hardware technology, network architectures, and research challenges in chapter 2. Related work about data routing in WSNs is introduced in chapter 3, as well as our proposed flat and cluster-based routing protocols and their performance evaluation through simulation experiments. Chapter 4 discusses the use of mobile data gathering approaches in WSNs. Chapter 4 also includes our proposed hybrid mobile/static data gathering technique and reports on its simulation experiments. Chapter 5 refers to the multimedia part of this thesis. It introduces the concepts, challenges, related work and existing solutions to enable interactive multimedia streaming over wireless networks, and presents our interactive multimedia streaming approach, the challenges, our multi-tier architecture for WMSNs, and the application-specific protocols and mechanisms to enable the convergence of sensor networks and interactive multimedia streaming. Finally, chapter 6 concludes this research dissertation with a summary of our schemes, achievements, and our contributions.

Chapter 2

Background Information on Wireless Multimedia Sensor Networks

2.1 Introduction

With the recent developments in wireless networks and inexpensive sensor nodes equipped with wireless radio communication, processors, storage, and battery power, wireless sensor networks (WSNs) have drawn the attention of both the research community and the industry, driven mainly by the promising new applications enabled by networks composed of a large number of small wireless devices that can organize themselves in an ad hoc manner, and are capable of sensing environmental conditions, perform data processing and transmission to a remote base station. Most research in WSNs have concentrated their efforts on developing solutions and applications that have low bandwidth requirements. These non-delay-sensitive applications usually capture and cache the data events, which are transmitted at a later time, thereby exempting the WSNs from providing timely response.

With the development of inexpensive multimedia hardware, *e.g.*, small wireless cameras, wireless multimedia sensor networks (WMSNs) have recently emerged as an essential technology for a variety of applications [7]. A WMSN comprises wireless devices intercon-

nected in an ad-hoc manner and capable of capturing and distributing multimedia. The envisioned applications include video surveillance sensor networks, advanced emergency preparedness, health care, border protection, just to mention a few.

In this chapter, wireless multimedia sensor networks are introduced along with the applications and challenging open research issues in the area.

2.2 Applications of Multimedia Sensor Networks

Several potential applications for WMSNs are envisioned that are capable of capturing and distributing multimedia contents from the monitored physical environment. Potential WMSNs applications are briefly introduced in this section.

2.2.1 Smart Surveillance Networks

Existing surveillance systems will be enhanced by the use of wireless video sensor networks, in which each sensor node will be equipped with a tiny video camera and a wireless radio transceiver. These nodes will cooperate with each other in an ad-hoc fashion in order to deliver multimedia content to a remote command center. Several applications are envisioned, specially for emergency response situations, in which a wireless multimedia sensor network will assist emergency responders by providing them with a better view of the target area and by helping them to trace more efficient strategies in real-time, whether to rescue victims or to fight terrorists or criminals. Video surveillance network systems can be used for tracking and monitoring of troops and enemy activities [43]. Moreover, the deployment of WMSNs in buildings, factories and public places will foster important new applications including the monitoring of suspicious behavior, enhanced security systems for people tracking, etc.

2.2.2 Intelligent Traffic Control and Management

WMSNs will not only provide entertaining multimedia content, but they will also assist in traffic control by monitoring traffic and finding better routes, video streaming services that offer traffic conditions to vehicle occupants. It can also enhance the services offered by existing traffic control systems by controlling the phase of traffic lights [27]. The convergence of vehicular ad-hoc networks (VANETs) and WMSNs will enable a variety of multimedia applications to be delivered directly to vehicle occupants and it will introduce new forms of vehicle safety [91]. For instance, events can be sensed by a vehicle, processed in order to extract useful information, and disseminated to other vehicles [73]. Furthermore, multimedia sensors will be capable of monitoring traffic flows to detect violations [7]. Moreover, other potential applications include smart parking, *e.g.*, available parking lots can be discovered, or providing locations of tourist attractions and popular sites [17].

2.2.3 Automated Homes and Health Care Centers

Recent advances in wireless multimedia and sensor networking have opened new opportunities for health-care systems. The integration of the existing specialized medical technology with wireless ad-hoc and sensor networks is gaining momentum and will provide augmented data gathering and real-time response. There are several potential application that will benefit from this integration, *e.g.*, assistance for elderly people or child care. Other projects such as "CodeBlue" at Harvard have extended WSNs for medical applications [32]. Researchers in the CodeBlue project have developed wearable bio-sensors based on the Mica2 [81], MicaZ [83], or Telos [115] sensor node platforms. Vital body data is collected and relayed to remote base stations, *e.g.*, personal digital assistants (PDAs) and laptops, and visualized in real-time by emergency responders and paramedics.

2.3 Wireless Sensor Hardware

Wireless sensor networks are gaining momentum with countless applications and potential market opportunities. We are witnessing an increase demand for small sensor devices with multimedia capabilities. However, several technical challenges have to be overcome in order to popularize wireless multimedia sensor networks. In the last years, significant progress has been made in the development of wireless sensor network hardware. The purpose of this section is to briefly introduce the evolution of the wireless sensor node hardware and the state-of-the-art in this area.

2.3.1 Wireless Sensor Motes: State-of-the-art

In this subsection, we outline the existing hardware technology of wireless sensor motes. In particular, we discuss some of the existing hardware and their features.

The development history of wireless sensor nodes dates back to 1998, when Kris Pister *et. al.*, started the Berkeley's Smartdust project [108]. The main research goal of the Smart Dust project was to integrate a device with processing, sensing, and communication capabilities into a cubic millimeter package. Although this project ended early on in 2001, several research projects have grown out of it. Among these are the Berkeley's Wireless Embedded Systems (WEBS) [119] and the UCLA's Center for Embedded and Networked Sensing (CENS) [21]. The WEBS includes the NEST project, which is a software/hardware platform that aims at accelerating the development and testing of algorithms for WSNs. This platform involves the development of low-cost sensor motes, their operating system, the infrastructure for time synchronization, storage, computing and simulations, and mechanisms for debugging and data visualization [119]. The CENS project at UCLA aims at exploring the fundamental principles and technologies needed to apply embedded sensor networks to a wide range of applications.

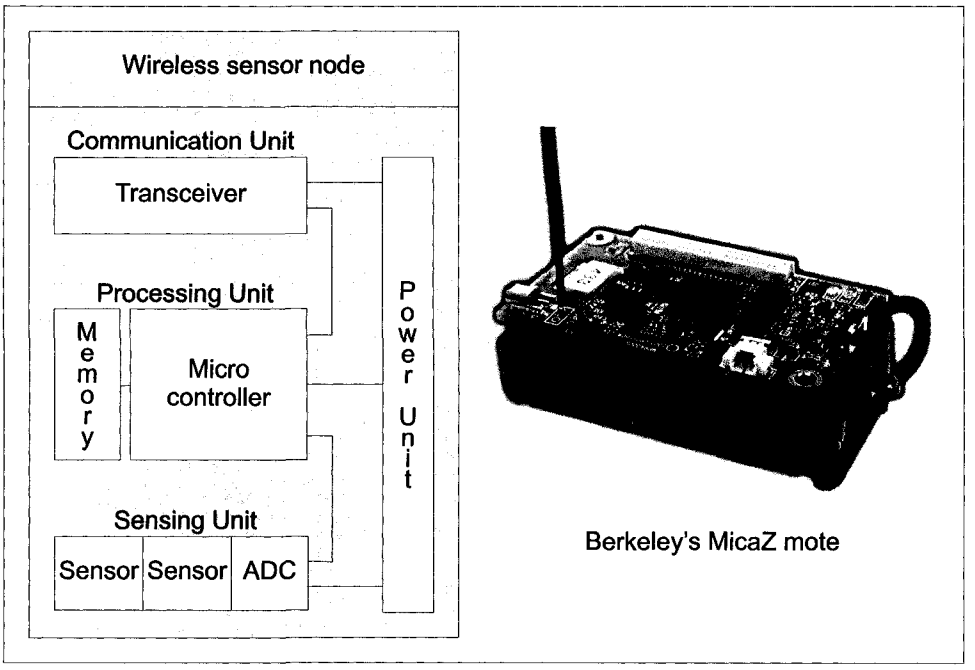


Figure 2.1: The typical wireless sensor node architecture (right: MicaZ mote [83])

Advances of the Wireless Sensor Mote

In this subsection, the existing wireless sensor nodes are discussed briefly in terms of hardware features. A sensor node, also known as a mote, is a lightweight device in a wireless sensor network that is capable of performing some processing, data gathering and communicating with other sensor nodes in the network in an ad hoc manner. The typical architecture of a sensor node is depicted in Figure 2.1. It usually includes a wireless transceiver operated at 916.5MHz or 2.4GHz, IEEE 802.15.4 compliant, a micro-controller running at a few megahertz, a small memory unit of a few kilobytes, one or more sensor units, and a constrained power unit. A sensor mote can even run a simplified operating system like the TinyOS [117]. Developed at the UC Berkeley, the TinyOS is a lightweight, open-source, energy-efficient, operating system that supports large scale wireless sensor networks. Most of the sensor motes support the TinyOS.

A brief description of the existing sensor nodes and their hardware features is given below.

- The weC mote [57]was designed at the UC Berkeley in 1999. It had a micro-controller running at 4 MHz, a program memory of 8 KB, and it had light and temperature sensors. The weC was capable of being programmed remotely through the sensor network. It was also compatible with the TinyOS operating system.

- Developed in 1999, the René node [56] enhanced the capabilities of the weC node by providing an expansion interface for connecting both analog and digital sensors. The René node became very popular and several commercial sensor boards have been developed for it. This node carries 8KB of program memory and can be programmed remotely like the weC mote. Its wireless channel operates at 916.5 Mhz and its transmission can reach up to 20 meters indoors.

- The René 2 module [56] is an enhancement of the original René mote. It features twice the memory available in its predecessor, *i.e.*, 16KB of program memory and 1KB of RAM, and an ATmega163 micro-controller. It is capable of switching from sleep mode to active in 36μsec. It shows similar energy efficiency to the original René node.

- The Dot mote [36] was developed in 2000. It is basically a miniaturized version of the René node. It features a complete node including sensor, processing, communication, and a battery resources. With the same processing and communication capabilities of the Rene platform, the Dot platform had 16KB of program memory and 1K of data memory.

- The Mica mote [80] was a step forward in the evolution of sensor motes. It was a reference in research and development of wireless sensor networks. Its CPU, an Atmel Atmega 128L, was clocked at 4 MHz and mainly targeted low-power WSNs. Its 128KB of program memory, 4Kbytes of SRAM and a 4Kbyte EEPROM, coupled with its low-power dissipation, were more than enough to attract the attention of researchers of wireless sensor networks. Moreover, it was also compatible with the

TinyOS. Its radio unit operated in the 916 or 433 MHz bands, with a throughput of 40 kbps. The Mica mote used two simple AA batteries that were able to keep it up and running for almost one year due to a mechanism that alternates between sleep mode and running mode to conserve power. It offered expansion interfaces like the René mote in order to support and ease the integration of a variety of sensor boards.

- The Mica2 [81] uses the same processor as the original Mica mote as well as it features an expansion connector to interface with external sensor boards. Its battery life is similar to the original Mica mote as it uses AA batteries. The Mica2 mote offers 512Kbytes of serial flash and a radio transceiver that operates on either the 433 MHz band or the 868/916 MHz bands, and a throughput of 38.4 kbps at a maximum range of150 meters. The Mica2 mote can also function as a router. It is also controlled by the TinyOS operating system and can be programmed remotely.

- The Mica2Dot hardware [82] is very similar to the Mica2 mote, with the same processing and communication capabilities, but in a coin-size chip. Due to its limited size, the Mica2Dot mote has reduced input/output capabilities. It features a solderless expansion that is handy for attaching external sensor boards.

- The MICAz mote module [83] was designed for enabling low-power sensor networks and it features several new enhancements over the previous motes of the Mica family. The MicaZ radio is now IEEE 802.15.4/ZigBee compliant and operates at 2.4GHz, and it shows an excellent throughput performance for these kind of constrained wireless devices, offering up to 250kbps data rate. It is also compatible with the TinyOS.

- The Telos mote [115] is a low power wireless module for sensor networks. The Telos mote features a USB interface to connect to several commercial sensor boards. Its radio is IEEE 802.15.4 compliant and can interoperate seamlessly with other de-

vices. It can transfer data at 250 kbps up to 125 meters range indoors. It integrates humidity, temperature, and light sensors, and is compatible with TinyOS.

- The Dust Networks' SmartMesh-XT mote [38] is powered by an IEEE 802.15.4 radio with a power amplifier for an extended range of 400 meters outdoors, while operating in the global license-free 2.4 GHz band. The SmartMesh-XT mote achieves high network reliability, utilizes frequency hopping for interference rejection, and have a typical battery life of 1 year, depending on the application. It can also function as a router for easy network integration, installation and maintenance.

2.3.2 Existing Video Sensor Hardware

A new trend in evolution is envisioned were small sensor nodes will be capable of capturing and delivering multimedia content to remote observers or command centers. Several mote architectures for wireless image sensor networks have been proposed in the past. At first, researchers were trying to integrate video cameras and network cards to provide a wireless video sensor node. The video sensor developed for the Panoptes project [44], for instance, is based on the Intel StrongARM 206 MHz embedded platform. The device is approximately 7 inches long, with an IEEE 802.11 compliant radio, and approximately 4 inches wide. The sensor has a Logitech 3000 USB-based video camera, 64 Mbytes of memory, and it is operated by Linux. The complete device including compression and transmission over IEEE 802.11 consumes approximately 5.5 Watts while capturing and delivering video of 320x240 resolution at 18-20 frames per second.

In 2005, Cao et al. [18] proposed an image sensor node based on an ARM7 micro-controller and a CMOS image sensor with 640x480 resolution. It employed a low-power radio transceiver in order to communicate to neighboring notes at transfer rates up to 76Kbps. The Cyclops project [95] proposed an image sensor node that combines the Cyclops image sensor with MICA2 [81] and MICAz [83] motes.

Downes et al. [37] presented a video sensor mote with an ARM7 micro-controller, 2

Mbytes of flash memory, and 2.4 GHz IEEE 802.15.4 radio module. This mote can carry up to four low-resolution image sensors and two CMOS cameras.

The CMUcam3 project [31] is an initiative from the Carnegie Mellon University in providing video capabilities to intelligent sensors. The CMUcam3 is based on the ARM7TDMI RISC processor and it is connected to an Omnivision CMOS video sensor capable of capturing RGB color images of 352x288 pixels. It offers an interface for the Telos [115] sensor mote.

The MeshEye [55] is an energy-efficient video sensor mote architecture developed at the Stanford University. Basically, it borrows the ideas of [37] by proposing a low-resolution stereo-vision system that is capable of calculating the position, range, and size of moving objects. The detection of a moving object will then trigger another camera module in order to acquire high-resolution images. Several applications are envisioned, for instance, intelligent surveillance and home automation.

2.4 Wireless Multimedia Sensor Network Architecture

Due the constrained resources of wireless sensor motes, traditional research on wireless sensor networks has always focused in scalability issues and how to extend the network lifetime. Therefore, traditional sensor networks are mostly based on a flat architecture, with homogeneous sensor nodes that are capable of communicating with their neighboring nodes only. The nodes in a flat network deliver data to a remote base station in a multi-hop fashion. However, flat topologies with ordinary sensor nodes may not be suitable to deliver multimedia content. Basically, there are three variants of multimedia sensor network architectures [7]: a single-tier flattened, a single-tier clustered, and a multi-tier topology, as depicted in Figure 2.2. Figure 2.2(a) shows a single-tier, flat topology in which sensor nodes acquire multimedia content from the environment and deliver the data to a remote sink through a multi-hop route using neighboring nodes. A single-

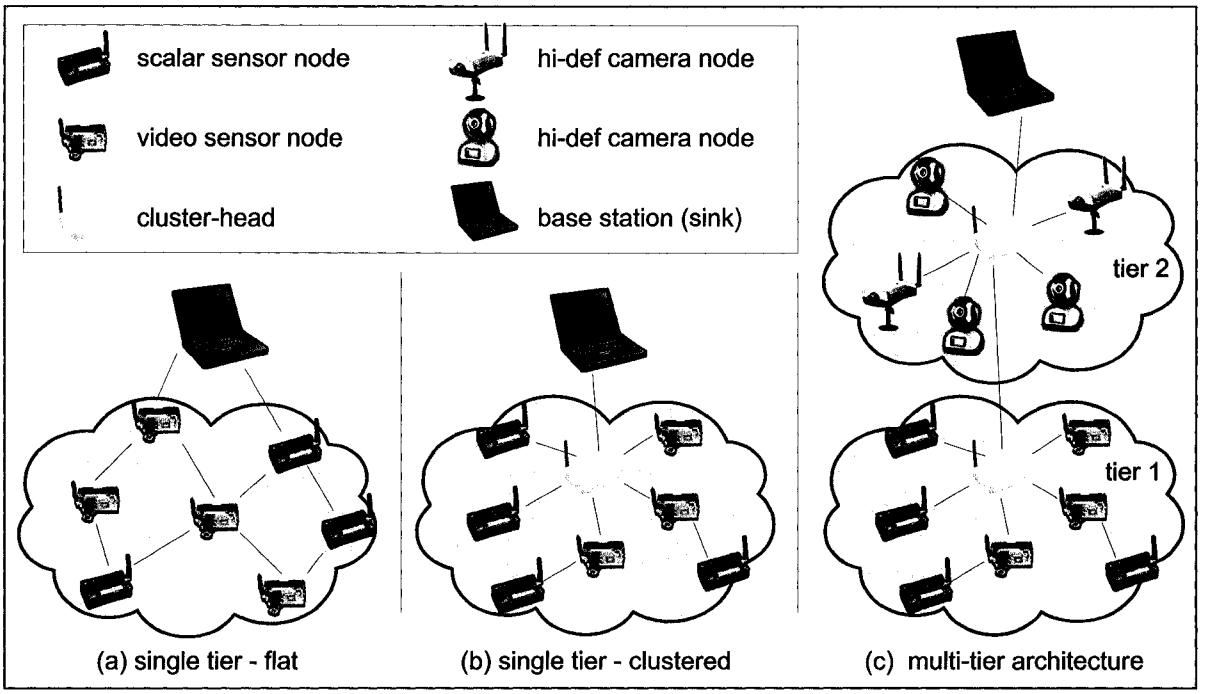


Figure 2.2: The different architectures for WMSNs.

tier, clustered topology is shown in Figure 2.2(b). In this architecture, cluster-heads are powerful nodes capable of complex multimedia acquisition, processing and storage. The nodes in a cluster deliver their scalar or multimedia content to a cluster-head first. The cluster-head will process the data and relay the multimedia content to the sink. The third architecture, as depicted in Figure 2.2(c), employs a multi-tier hierarchy in which heterogenous wireless devices are in charge of performing different tasks, depending on the tier they belong to. Simpler tasks are given to the ordinary, low-power sensor nodes, *e.g.*, detecting temperature, smoke, movement. More complex tasks are left to the resource-rich devices, *e.g.*, capturing images, video, or audio and transmitting them.

2.4.1 Existing Deployments of Multimedia Sensor Networks

In the Panoptes project [44], a scalable video surveillance system is implemented in which Panoptes video sensor motes are utilized. The architecture comprises Panoptes

motes, a video aggregating node, and a client interface. Each video sensor, as discussed in the previous section, is attached to an IEEE 802.11 compliant radio module in order to communicate to other video sensor nodes as well as to the higher tier, *i.e.*, the video aggregating node. In this project, a change detection filtering algorithm was implemented in order to identify events of interest that would trigger the recording of a video sequence. The video is then recorded until motion stops. The video aggregating node can be at any IP-enabled device and will be responsible for video storage and distribution to client interfaces. End-users are able to query the video database remotely.

The SensEye project [71][106] assumes a three-tier architecture in which the lowest tier comprises low-power, resource-constrained sensor platforms, *e.g.*, MICA2 [81] or MICAz [83] motes, and low-resolution Cyclops or CMUcam camera sensors. The second tier comprises PDA-class Stargates [111] nodes equipped with 400MHz XScale processor, higher resolution webcams than the first tier, they run Linux, and each node has two wireless interfaces: an IEEE 802.11 radio that is used communicate with each other node and a 900MHz radio that is used to communicate with the nodes in the lower tier. The last tier comprises high-resolution pan-tilt-zoom cameras connected to resource-rich platforms, *e.g.*, personal computers (PCs). These cameras are used in the SensEye to fine-grain tracking or to fill gaps in coverage. Due to their high energy requirements, the camera nodes in this tier can be woken up on-demand to acquire a high-resolution image only after a new object is detected by a lower tier. This approach can enhance the network lifetime.

2.5 Research Challenges in Multimedia Sensor Networks

This subsection highlights the relevant research challenges in designing architectures for Wireless Multimedia Sensor Networks. The main challenge is to provide effective solutions to relay multimedia content, *e.g.*, audio, video, images, that are known to require

high-bandwidth, over such constrained networks. It is common sense among researchers in this area that the multi-tiered architecture is a feasible approach, and several research projects are adopting it [71][106]. However, a multi-tier WMSN architecture comprises a mix of heterogenous devices and it is tightly coupled with the application. Some applications have multiple objectives, sometimes conflicting, making it hard to come up with a definite sensor network configuration for different application goals. For instance, surveillance and emergency systems often require the sensor network to be reliable and operational for long periods. Thus, it is challenging to provide both high-reliability and extended network lifetime simultaneously. Kulkarni *et. al.*, [70] stated that maximizing lifetime across a multi-tier video sensor network is feasible by taking advantage of the devices at higher tiers, *i.e.*, resource-rich nodes, in order to compensate for the lack of energy resources at lower tiers. Kulkarni *et. al.*, also pointed out the latency issue. For real-time applications, *e.g.*, surveillance and emergency preparedness system, latency is an important metric. Minimizing delay in a flat WMSN architecture involves only the latency of the sensory hardware, *i.e.*, detecting the event and processing analog to digital conversion, and the delay of delivering the event to a remote base station. However, in a multi-tier architecture, multimedia data should be relayed to special entities at higher tiers, for instance, that will perform specific tasks on the data, *e.g.*, image processing, filtering, and compression. Hence, there is a tradeoff between network lifetime and latency.

Another important challenge to overcome is the required bandwidth for multimedia applications. Multimedia content requires high bandwidth that is not available in today's wireless sensor hardware. For instance, the state-of-the-art in sensor motes, *e.g.*, the MICAz, Telos, and SmartMesh-XT motes, offers maximum data rate of 250kbps. The bandwidth problem is multiplied when multi-hop communications are taken into consideration, *e.g.*, wireless sensor networks. The total capacity of a wireless link is significantly decreased due to interference with the transmissions of nearby nodes, as confirmed by performance evaluations [75]. In addition, Sun T. *et. al.*, [113] measured

the effective capacity of IEEE 802.15.4 compliant radios in CSMA-CA. They utilized MICAz motes [83] in a chain topology to evaluate the end-to-end throughput. Their results showed that the total capacity converged to $\frac{1}{4}$ of the one-hop capacity after 4 hops.

Another challenging issue is providing efficient low-power multimedia encoding techniques, as pointed out in [7]. Existing compression schemes rely on complex algorithms and were not designed for resource constrained devices.

Among all the challenging issues, one should give prominence to power consumption. Energy is considered as a critical issue due to the fact that wireless sensor motes are constrained devices and that multimedia content requires complex processing and high transmission rates. This tradeoff poses several issues in WMSNs and, thereby, energy savings must be maximized while providing good quality multimedia services.

2.6 Summary

This chapter introduced briefly the research area of wireless multimedia sensor networks, its applications, existing wireless sensor hardware, trends in the design of sensor motes, existing research on wireless video sensor networks, and discussed the challenging issues in wireless multimedia sensor networks. Although the advances in wireless sensor hardware have demonstrated a significant evolution in the last few years, it is still challenging to deploy wireless multimedia sensor networks, mostly due to the high bandwidth requirements of multimedia contents. For instance, the state-of-the-art sensor motes, *e.g.*, the MICAz, Telos, and SmartMesh-XT motes, offer maximum data rates of 250kbps. This is still one order of magnitude lower than the required for multimedia applications. However, the multi-tier wireless multimedia network architecture seems to be a promising alternative solution.

The next chapter discusses the existing research on routing for sensor networks, presents the proposed routing protocols and reports on their performance evaluation.

Chapter 3

Routing Protocols for Wireless Sensor Networks

3.1 Introduction

Data routing in wireless sensor networks is a challenging issue due to the characteristics that distinguish WSNs from wireless ad-hoc networks [5][93]: existing IP-based routing protocols are not suitable to WSNs; most WSN applications require that data be relayed from multiple source nodes to a single data aggregation point; the WSN traffic contains significant redundancy, because multiple sensor nodes might generate the same event while sensing the same area, *e.g.*, a moving object is detected by several sensor nodes in the field; wireless sensor nodes are constrained devices with limited energy, processing and storage resources. Due to the aforementioned differences, several routing protocols have been proposed for wireless sensor networks [15, 61, 77, 78, 102, 107, 123], among others. This chapter discusses the previous and related work on routing protocols for wireless sensor networks. Section 3.5 presents our proposed protocols, the detailed description of the algorithms along with their performance evaluation experiments and results.

3.2 Related Work

Data routing protocols for wireless sensor networks can be classified into four categories, according to [5]: data-centric, cluster-based or hierarchical, location-based, and quality of service based protocols.

- **Data-Centric Routing Protocols**: In data-centric routing protocols, data is routed based on queries describing the user's interest, rather than host addresses, *i.e.*, the interest is focused in the data gathered by the sensor network rather than in individual node addresses. The protocols in this category require an attribute-based naming scheme that describes the data or information of the sensor network, usually involving attribute-value pairs, *e.g.*, temperature=30, humidity=50, objects=10, etc. For instance, one can query the network for information such as "send me the temperature only when it is greater than 50^0C", or even "give me the location of the person X", etc. Furthermore, there is no global node addresses, *i.e.*, the nodes have only local knowledge of the neighboring node, otherwise it would affect the network scalability. In data-centric routing, the sink or base station is usually responsible for sending queries to and collecting the reports from the sensor network. The SPIN [52] and Directed Diffusion [60, 61] protocols were among the first attempts in data-centric routing for wireless sensor networks. Several Directed Diffusion-based data-centric routing protocols [15, 30, 105] were designed since then.

- **Cluster-based Routing Protocols**: Protocols in this category allow the sensor nodes to organize themselves into clusters, *i.e.*, groups of sensor nodes in which each cluster is ruled by one leader or cluster-head. Thus, resource constrained sensor nodes can be in charge of performing the sensing tasks and transmitting their scalar data to the appropriate cluster-head, usually involving a few hops to relay the data from the sensor node to the cluster-head. The cluster-heads, on the other hand, are resource-rich devices that are capable of performing more complex tasks than ordinary scalar sensor nodes. For instance, a cluster-head node has special

battery and wireless resources and can be used to perform data fusion/aggregation in order to eliminate redundancy. A cluster-head receives data from all the nodes within its cluster and usually relays the processed data to a remote base station or data sink. Therefore, the cluster-based approach can greatly contribute to the system scalability, lifetime, and energy efficiency. Due to its characteristics, a cluster-based architecture usually provides better load and energy distribution than a flat network approach, thereby avoiding potential bottlenecks. Furthermore, the cluster-based routing scheme reduces energy consumption due to the lower amount of data messages that a cluster-head should forward to the sink after it has performed data aggregation/fusion. In some approaches, the cluster-heads are capable of communicating with one another and form the backbone of the wireless sensor network. Several cluster-based approaches were proposed in the literature for WSNs. The LEACH protocol [53] is one of the first attempts to provide a cluster-based routing mechanism for WSNs. Later, a number of cluster-based routing protocols have been developed [77, 78, 112, 125], following the pioneering ideas of the LEACH protocol.

- **Location Based Routing Protocols**: In this category, the location information of the sensor nodes are required for the proper operation of the protocols. Position information can be acquired from nodes equipped with global position system (GPS), radio signal strength triangulation, manually, from a reference point, etc. Location-based routing protocols can utilize the position of nodes, for instance, to turn off redundant nodes in a same area that do not participate in the routing in order to save energy, etc. Examples of location-based solutions for WSNs include the Geographic Adaptive Fidelity (GAF) [120], the Geographical Energy Aware Routing (GEAR) [126], among others.

- **Quality of Service-Based Routing Protocols** The key focus of QoS-based routing protocols is to maintain a balance between energy consumption and data

quality over the wireless sensor network. In particular, the protocol has to satisfy certain QoS requirements, *e.g.*, delay, energy, bandwidth, etc., while delivering data to the sink. However, the characteristics of WSNs pose several challenges to the design of QoS aware routing protocols: resource-constrained devices impose limitations to the complexity of algorithm; traffic in WSNs is usually unbalanced, *i.e.*, packets flow from a large number of sensor nodes to a single data aggregation point; data redundancy is common in WSN and data fusion/aggregation schemes should be considered; sudden topology changes may arise in WSNs due to node failures, interference, or even due to sleep/awake approaches utilized by some algorithms; unbalanced energy consumption should be avoided in order not to deplete the power resources of certain nodes faster than others, thereby causing problems such as network disconnections or it even turn the network completely unusable; and the scalability issue, *i.e.*, a scheme for WSNs should guarantee the QoS requirements of the application scale up with the network simultaneously. The Sequential Assignment Routing (SAR) [109] and SPEED [51] protocols are two examples of QoS-based routing approaches for wireless sensor networks.

The next subsection discuss some of the routing protocols available in the literature of wireless sensor networks.

3.2.1 Sensor Protocols for Information via Negotiation (SPIN)

In 1999, Heinzelman *et. al.*, [52] proposed the family of protocols called SPIN. SPIN focuses on the dissemination of individual sensor data to all the sensor nodes in the network, assuming that all nodes are potential sink or base station nodes. SPIN is based on two design principles: negotiation and resource-adaptation. SPIN nodes negotiate with each other before transmitting data in order to eliminate the transmission of redundant data throughout the network. The negotiation mechanism requires a meta-data naming scheme to describe the information. Each SPIN node has a resource manager that polls

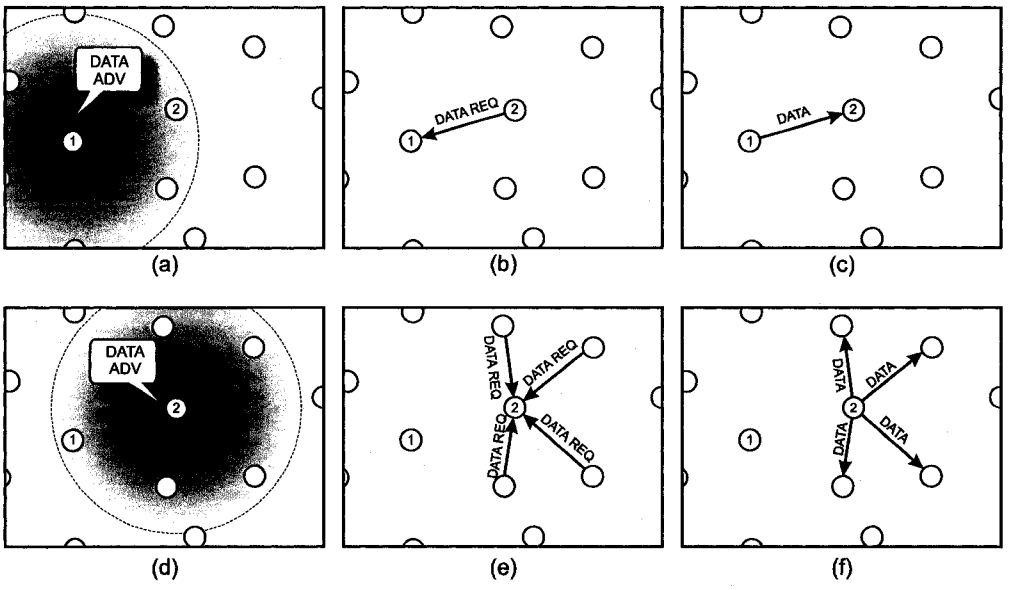


Figure 3.1: SPIN's message exchange mechanism

the node's resources before data can be processed or transmitted, *e.g.*, a sensor node can decide not to perform certain activities when its energy level is low.

An overview of the message exchange operation of the SPIN protocol is shown in Figure 3.1. The SPIN protocol starts when a sensor node gathers new data that it is willing to share. The node will notify its neighbors about its new data by sending an advertisement message that carries the meta-data. If a neighbor is interested in the data, it replies with a data request message and the data is sent to this neighbor node. Thereafter, the node that receives the data will repeat the data advertisement process with its own neighbors. In conclusion, SPIN is a simple protocol that relies on the nodes local information, *i.e.*, the nodes know only their one-hop neighbors. It provides energy savings by almost halving the redundant data when compared to a simple flooding. The drawback of this scheme is that when a node is interested in the data but it is located far away from the source node, and the intermediate nodes between source and destination are not interested in that specific data, the advertisement message may never reach the destination. Therefore, SPIN cannot be directly applied to applications that require real-time and reliable data delivery such as surveillance and emergency preparedness.

3.2.2 Directed Diffusion (DD)

In [60], Intanagonwiwat *et. al.*, proposes the Directed Diffusion (DD) routing protocol for wireless sensor networks. The DD is a data-centric protocol in the sense that the data generated by the sensor nodes are named by attribute-value pairs. Similar to the SPIN protocol, DD aims at aggregating redundant data to minimize the number of transmissions, to save energy and to prolong the network lifetime. The application can inject interest messages in the network through the sink. An interest is a query that contains a list of attribute-value pairs such as name of objects, interval, duration, coordinates, or events, *e.g.*, temperature, pressure, etc. The sink broadcasts interests and each node that receives an interest sets up a gradient that points towards the neighbor sensor node from which it received the interest. A gradient contains information about how to forward data to a specific sink through a forwarding neighbor node, *i.e.*, it is a forwarding link and is characterized by the data rate, duration and expiration time derived from the received interest message. The process of diffusing the interests is based on flooding the network until all sensor nodes have gradients, *i.e.*, paths are established between sensor nodes and the sink. However, multiple paths can be established and even loops will be possible. DD eliminates loops at a later stage and selects the best path by using a reinforcement mechanism. This mechanism is started by the sink, which re-sends the original interest message through the selected path, but with a smaller interval, in order to update the gradients of each node in the path. By doing this, the sink will reinforce the source node to increase its data rate. An example of the operation of the DD protocol is depicted in Figure 3.2. DD also provides a route recovery mechanism. Sensor nodes may fail to work properly due energy depletion, obstacles between source and destination, interference, physical destruction, etc. When a node fails to deliver its data, another route should be discovered. The DD path recovery mechanism basically re-initializes the reinforcement mechanism in order to find other routes. However, the drawback of this scheme is that DD must flood the entire network every time it needs to recover from broken routing paths. Therefore, the sink refreshes and re-sends interest

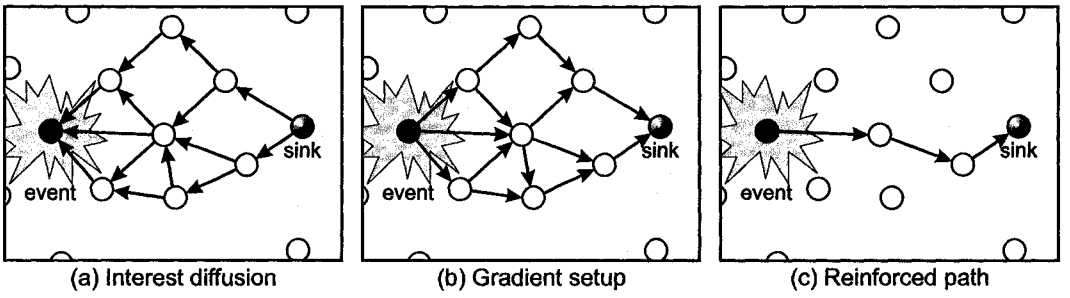


Figure 3.2: The Directed Diffusion protocol.

messages periodically to keep the network failure-free. Another drawback is the extra overhead caused by the reinforcement scheme and the matching of interests with events.

3.2.3 Rumor Routing (RR)

The Rumor Routing (RR) protocol [15] is one among the many variations of Directed Diffusion. DD floods of the entire network to propagate interest messages. The key idea of Rumor Routing is to eliminate the unnecessary flooding by routing queries based on "rumors" about which sensor node has observed the event of interest. The RR protocol operates with agents. Hence, when a sensor node detects an event, it generates an agent that is responsible for propagating the rumor through the nodes along the path it travels. Therefore, the source node will relay its data message to a remote sink and each sensor node in the path will setup a gradient or link so that future event requests to that specific event will be routed through a direct path. For instance, when a a sensor node forwards a data request, the nodes that already know the route to that specific event will reply and properly route the request to the source node. This is how RR avoids flooding the entire network. The authors show through simulation experiments that RR is capable of reaching significant energy savings [15]. However, the drawback the RR protocol is that its performance depends on the number of events that are detected, because the maintenance of agents adds extra overhead when there is a large number of events but a small number of interests from the sink on those events.

3.2.4 Low-Energy Adaptive Clustering Hierarchy (LEACH)

The LEACH protocol [53] was one of the pioneering approaches on the literature about cluster-based routing in wireless sensor networks. Basically, LEACH utilizes the randomized selection of cluster-heads to distribute the energy load among the node in the sensor network. The LEACH protocol has two phases: a set-up phase when the sensor nodes are organized into clusters; and a transmission phase in which data are relayed from the nodes to the cluster-heads and from cluster-heads to the sink. LEACH operates in rounds in which the sensor nodes elect themselves to be cluster-heads at the beginning of a round r starting at time t, with a certain probability $P_i(t) = p$, which is chosen depending on the number of cluster-heads k one might expect. Thus:

$$P_i(t) = \begin{cases} \frac{k}{N - k*(r \bmod \frac{N}{k})} & \text{if } C_i(\mathrm{t}) = 1 \\ 0 & \text{if } C_i(\mathrm{t}) = 0 \end{cases}$$

where N is the total number of nodes in the network. If each node is a cluster once in $\frac{N}{k}$ rounds, all nodes will be cluster-heads the same number of times. $C_i(t) = 1$ indicates that node i is eligible to be a cluster-head at time t, and 0 otherwise.

After cluster-heads are elected, these nodes must broadcast their new status to the other sensor nodes in the network. Thus, each cluster-head broadcasts an advertisement message to the entire network using the CSMA MAC protocol. This will ensure that every single sensor node can become part of a cluster. Each non-cluster-head sensor node determines to which cluster it belongs by choosing the cluster-head that requires the minimum communication energy, based on the received signal strength of the received advertisement message from each cluster-head. After this decision, each sensor node must notify the cluster-head that it will join the cluster. The notification is sent back to the chosen cluster-head using CSMA. Each cluster-head sets up a TDMA schedule and coordinate with all the nodes in its cluster in order to ensure that will not be collisions among data messages. This mechanism also allows the sensor nodes to schedule sleep periods to further minimize energy consumption. The data transmission phase begins

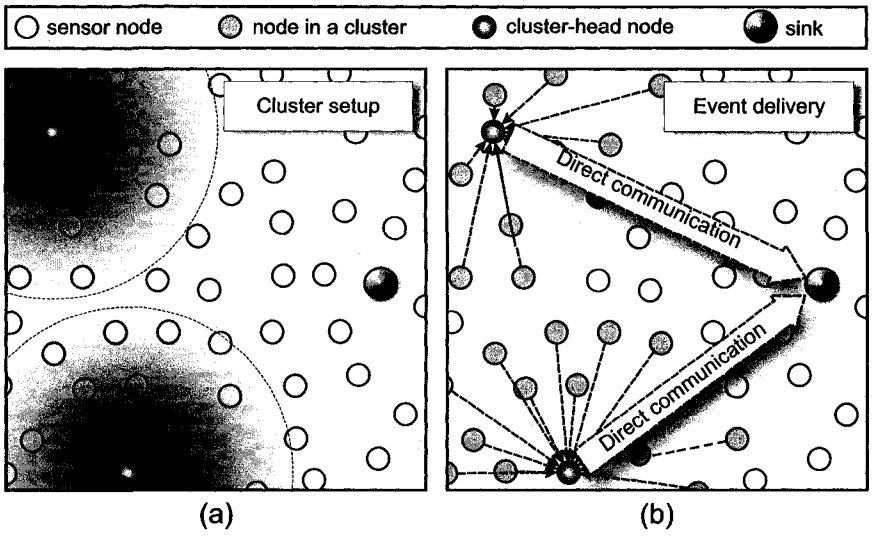


Figure 3.3: The LEACH protocol.

only after these steps are complete. LEACH also performs local data aggregation in the cluster-heads to reduce the amount of data being sent from the clusters to the sink, thereby reducing energy dissipation even further and enhancing network lifetime.

According to the simulation results presented in [53], the sensor nodes reach energy depletion randomly and the dynamic clustering increases the network lifetime. The LEACH protocol is completely distributed and requires no global knowledge of network. However, it uses one-hop communications from sensor nodes to cluster-heads and from cluster-heads to the sink, as exemplified in Figure 3.3. Therefore, the LEACH mechanism faces scalability problems if applied to large networks due to the long distance communications in order to relay the gathered data.

3.2.5 TEEN and APTEEN Protocols

The Threshold-sensitive Energy Efficient sensor Network protocol (TEEN) [77] and the Adaptive Periodic TEEN protocol (APTEEN) [78] are two routing protocols that were mainly inspired by LEACH [53]. The TEEN protocol employs the same clustering model proposed in LEACH, but it was designed to be responsive to sudden changes in the

sensed attributes. To this end, TEEN uses two thresholds: hard and soft thresholds. Basically, each cluster-head node sends the hard and soft threshold values to all sensor nodes in its cluster. The hard threshold is used to reduce the number of transmissions by programming the sensor nodes to transmit data to the cluster-heads only when the sensed attribute is in the range of interest, *i.e.*, only when the value of the sensed attribute is equal to or greater than the hard threshold. The soft threshold is used to check if there was any changes in previous readings of the same attribute, such that the sensor node will only transmit after a major change has occurred. Thus, the soft threshold can further reduce data transmissions. The soft threshold implies that if a small value were used the more accurate the sensor network readings would be. Therefore, there is a tradeoff between energy savings and data accuracy. Since these thresholds are broadcasted every time there is a cluster setup, these thresholds can be modified in each round. The main drawback of the TEEN protocol is that if the thresholds are not reached or not even received, the nodes will never transmit their data.

The APTEEN scheme [78], on the other hand, tries to solve the main drawback of the TEEN protocol and it offers more flexibility to the user needs and the type of the application. Each cluster-head will send the threshold values along with a count time interval. The count time interval is a timer that sets the maximum interval between two successive reports sent by a node, *i.e.*, if a node does not report any data message to the cluster-head for a time period equal to the count time, it is forced to transmit the data. Simulation results have demonstrated that the two protocols outperform LEACH regarding energy consumption and network lifetime. The main drawbacks of the two approaches are the overhead involved in the set up of clusters, actually this is an issue with almost all cluster-based protocols, and the distribution of thresholds values.

3.2.6 Energy Efficient Hierarchical Clustering

Bandyopadhyay, et al. [9] proposed a randomized and distributed hierarchical clustering algorithm for sensor networks. The sensor nodes forward the gathered data to level-1

cluster-heads. The level-1 cluster-heads perform data aggregation/fusion and transmit the data to level-2 cluster-heads, and so on, until the level-n cluster-heads relay their data to the final destination, *i.e.*, the sink.

The algorithm first elects the level-1 cluster-heads, then level-2 cluster-heads, and so on. The level-1 cluster-heads are chosen based on the LEACH algorithm, *i.e.*, each sensor nodes decides to become a level-1 cluster-head with a certain probability p_1), and notifies its new status to the sensor nodes within communication range. Furthermore, the notification message is forwarded to all the sensors nodes within k_1 hops of the advertising cluster-head. Each sensor node that receives the notification joins the cluster of the closest level-1 cluster-head. The remaining sensors, *i.e.*, the nodes that did not received the notification from a cluster-head, become "forced" level-1 cluster-heads. Thereafter, the level-1 cluster-heads elect themselves as level-2 cluster-heads with a certain probability p_2 and broadcast their decision of becoming a level-2 cluster-head. This decision is forwarded to all the sensor nodes within k_2 hops. Each level-1 cluster-head that receive the notifications from level-2 cluster-heads joins the cluster of the closest level-2 cluster-head. The level-1 cluster-heads that did not receive the notification become forced level-2 cluster-heads. The process repeats for the selection of cluster-heads at level 3, 4,..., n. This mechanism will generate a hierarchy of cluster-heads.

This approach achieves better energy efficiency load balancing than previous works. Through simulation experiments, the authors show that the energy consumption decreases as the node density increases in a single level clustering scenario. On the other hand, in a multi-level clustering scenario, the decrease in the node density is followed by a reduction in the energy consumption levels. The drawback of this approach involves the high energy consumption of high level cluster-heads, since these cluster-heads will consume more energy during cluster-head selection, setup, and data forwarding. Consequently, these nodes may deplete their energy resources faster than other nodes in the network, causing instability and network disconnections.

3.2.7 Geographic Adaptive Fidelity (GAF)

The Geographic Adaptive Fidelity (GAF) [120] is a location-based routing protocol designed for wireless ad hoc networks that can be applied in sensor networks. In GAF, the network is divided into a virtual grid. Each node uses its location, *e.g.*, acquired from an embedded GPS receiver, to map itself to a position in the virtual grid. The sensor nodes inside each $r * r$ square of the grid are considered equivalents in terms of routing capabilities. A sample of a virtual grid is depicted in Figure 3.4. Any nodes within the square A can reach any of the square B nodes, and any square B node can reach the nodes in square C. Therefore, the nodes within the same square can organize themselves to decide about their duty cycles such that only one sensor node per square region can be turned on while the others will be put to sleep. As shown in Figure 3.4(b), nodes 1, 2 and 3 can communicate with any of the nodes 4, 5 or 6. Thus, nodes 1, 2 and 3 are considered equivalent and only one of them can be awake. By this, the routing fidelity is guaranteed. The same idea applies to nodes within region B. Nodes 4, 5 and 6 are equivalent and only one of them will be on per cycle.

The size of the squares r is proportional to the radio range R of the nodes such that the maximum distance between two nodes of adjacent square regions is not larger than R. For instance, the distance between the nodes 4 and 9 in Figure 3.4(b) is the largest possible distance between two nodes of adjacent regions, and this distance must not be larger than their transmission range R. Therefore:

$$r^2 + (2r)^2 \leq R^2 \tag{3.1}$$

or

$$r \leq \frac{R}{\sqrt{5}} \tag{3.2}$$

In addition, GAF operates in three states: discovery, active, and sleep. All nodes start in the discovery state, *i.e.*, the nodes turn their radios on and exchange discovery

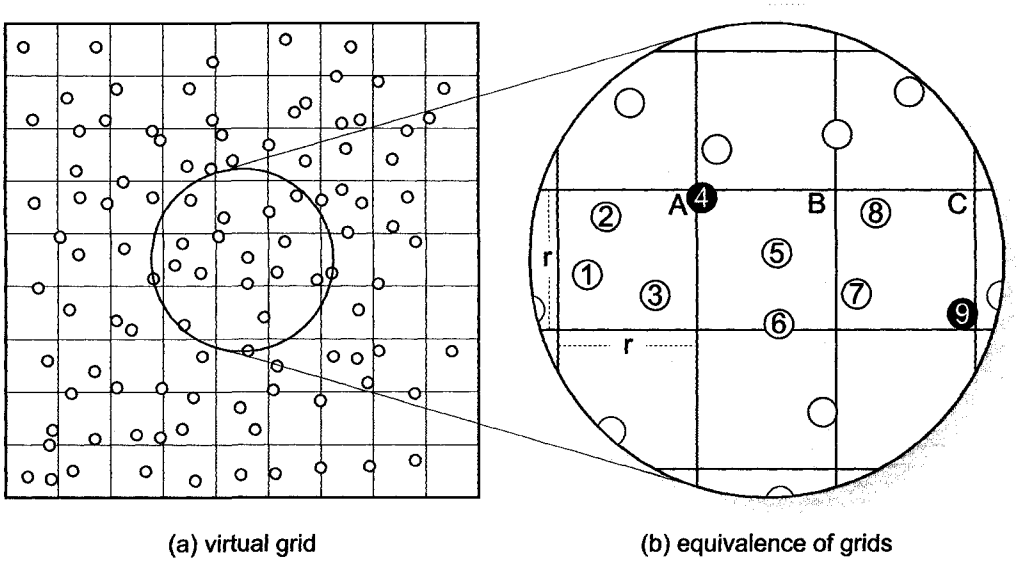


Figure 3.4: Sample of a GAF virtual grid.

messages to find other nodes within the same grid in order to be able to calculate the routing equivalence. A discovery message carries the node ID, which is basically the position of the node in the grid, and a timestamp. A node sets a timer and when this timer expires, the node broadcasts its discovery message and enters the active state. This timer can be suppressed by discovery messages of other nodes. Once in the active state, the node starts another timer that indicates for how long the node will stay active. The node continues to broadcast discovery messages at a specific time interval. An active node will change to sleep mode whenever it can determine that some other equivalent node will be responsible for routing within its square region. Simulation experiments show that GAF performs well regarding energy savings and increases the network lifetime by turning equivalent nodes off.

3.2.8 Geographical Energy Aware Routing (GEAR)

GEAR [126] is a location-based and energy-aware routing protocol that uses geographic information in order to disseminate queries to the appropriate region of the network, without the need of flooding. GEAR uses energy aware and geographically informed

neighbor selection heuristics to route a packet towards the target region [126]. GEAR forwards packets to all the nodes in the target region in two steps: firstly, it forwards the packets towards the specific target region; secondly, the packet is disseminated within the region. In the first step, a node forwards the packet to the nearest neighbor to the target region. If there is no neighbor closer to the destination than the node itself, GEAR picks a node that minimizes some cost value of this neighbor. If the packet has reached the target region, it can be diffused within that region by recursive geographic forwarding. The target region is divided into N sub-regions and N copies of the packet are created. This procedure continues until the current node is the only one inside this sub-region. Under some low density conditions, recursive geographic forwarding sometimes does not terminate. In this case, GEAR uses restricted flooding instead. The main idea of GEAR was to restrict the number of queries by only considering a certain region rather than sending the interests to the entire network like other solutions do, *e.g.*, Directed Diffusion [77]. By simulation experiments, the authors show that GEAR delivers 70% to 80% more packets than the greedy perimeter stateless routing (GPSR) [67] for an uneven traffic distribution.

3.2.9 Sequential Assignment Routing (SAR)

The SAR protocol [109] is one of the pioneering research works that introduced a notion of quality of service in a routing protocol for wireless sensor networks. SAR assumes multiple paths from source to sink nodes and it considers the energy resource, the QoS factors of each path, and the priority level of a packet during its routing decisions. Multiple paths are used in order to avoid the extra overhead of route discovery due to node failures. A tree rooted at the source node is built while avoiding sensor nodes with lower QoS and energy reserves. At the end of the tree construction process, each sensor node will be part of multi-path tree. Each sensor node can decide which neighbor nodes will be used to forward the packets by considering the additive QoS metric and the energy resource for each path. The node estimates the maximum number of packets it can

forward though a specific path without depleting its energy. Hence, the SAR algorithm calculates a weighted QoS metric as the product of the additive QoS metric and a weight coefficient associated with the priority level of the packet. SAR uses a route reconfiguration mechanism that is triggered by the sink periodically, similar to Directed Diffusion. To accommodate local path recovery, SAR uses a handshake procedure between neighboring nodes that is performed by checking the consistency of routing tables between upstream and downstream nodes on each path. The drawback of the SAR protocol is the overhead of maintaining routing tables and states.

3.2.10 Stateless Protocol for Real-Time Communication (SPEED)

The SPEED protocol [51] is a QoS-based routing protocol that provides end-to-end soft real-time communication guarantees. It strives to maintain a desired delivery speed across the sensor network by using an algorithm that considers a feedback control mechanism and a non-deterministic geographic forwarding. Similar to GAF, every single node in SPEED broadcasts a beacon message periodically in order to exchange location information among neighbors. In addition, SPEED uses two types of beacons: a delay estimation beacon and a backpressure beacon. These beacons are used to identify traffic changes. SPEED uses the single hop delay to estimate the load of a node. It utilizes an ACK-based mechanism to compute the delay. SPEED split the neighbor nodes into two groups: one group contains the nodes that have relay speeds larger than a desired threshold; and the other group that cannot reach such speed threshold. A relaying neighbor node is chosen from the fast group of nodes, and the neighbor with the higher relay speed has a higher probability to be chosen. If there are no fast neighbors, a relay ration is calculated based on the Neighbor Feedback Loop (NFL). The main goal of the NFL mechanism is to maintain a single hop relay speed above the desired threshold. A miss ratio of a neighbor is calculated based on the number of misses, *i.e.*, the number of packets that were forwarded to a node with relay speed lower than the threshold, or a packet loss. Thus, NFL attempts to converge the the miss ratio to zero. The algorithm

that chooses the neighbor that can provide the desired speed is called Stateless Non-deterministic Geographic Forwarding (SNGF). The NFL module provides the relay ratio that is fed to the SNGF module. Thereafter, the SNGF selects the node that meets the speed requirement (fast neighbor) based on the delay metric. If such a node cannot be found, it checks the relay ratio. If the relay ratio is less than a random number between 0 and 1, the packet is dropped. When a node fails to find a next hop node, the back-pressure module sends the packet back to the source node so that they can search for new routes. According to the author's simulation results, SPEED reduces the average end-to-end delay by 30% to 40% in a heavy congestion scenario when compared to the other algorithms considered, namely AODV [92] and DSR [64].

3.3 Our Proposed Routing Protocols for Wireless Sensor Networks

3.3.1 Initial Considerations

With the recent developments in wireless networks and multi-functional sensors with digital processing, power supply and communication capabilities, wireless sensor networks are being largely deployed in physical environments for real-time monitoring in different classes of applications [5] [84]. One of the most appealing applications is security surveillance and supervision of context aware physical environments for critical conditions monitoring. In a prison, for instance, it is important to keep a reliable monitoring of the physical environment, especially when emergency situations emerge, such as rebellions of inmates that can lead to incendiary fire conditions and human lives and patrimony losses. In such situations, it is important that information can be "sensed" from the physical environment while the emergency state is in progress, since more precise information can be used by security and rescue teams for operation management and strategic decisions. However, in order to keep the information flowing from the sensors during the

emergency, a wireless sensor network solution has to cope with sensor node failures, *e.g.*, due to malfunction, physical destruction, interferences, etc. Therefore, wireless sensor network solutions for such environments should be fault tolerant, reliable, fast, and provide fast reconfiguration. In terms of energy savings, in a silent monitoring state, sensor nodes can be programmed to notify about events in a periodic fashion, *e.g.*, send temperature every 10 minutes, or event-driven fashion, *e.g.*, send temperature only when above 60^0C. In these cases, the interest may not change for quite a long period. Some existing energy saving solutions take that into consideration and switch some nodes off, leading the nodes to an inactive state - these are waken up only when interest matches the gathered events [60]. On the other hand, in query-based application scenarios, queries can be propagated to sensors arbitrarily, according to the application and/or user's will and some existing energy saving solutions may not be adequate because the transition from inactive state to data transfer state can be costly in terms of energy dissipation when several arbitrary transitions are necessary [24]. Moreover, energy savings and fault tolerance support present conflicting interests when the paths involving inactive nodes have to be set up quickly due to path failure.

There are basically three types applications that should be taken into consideration when designing protocols for WSNs: periodic, event-driven, and query-based.

- *Periodic*: In this model, sensor nodes report the gathered data to the sink periodically. The sensor nodes may cache the data and trigger a delivery mechanism at a pre-specified rate according to the application's parameters. This kind of application is usually non-delay-sensitive but requires some degree of reliability. Examples in this category include: precision agriculture, in which periodic data reports are sent from the field; and data mining applications that requires periodic information to built their data association and rules.

- *Event-driven*: Applications in this category are mostly delay-sensitive, *i.e.*, real-time, and involves critical events. Therefore, low latency is the primary concern

in a routing protocol. In this model, a sensor node gathers the data from the environment and relay to the base station immediately. The data flow in this model is likely to present high rates of redundancy. Therefore, a data aggregation/fusion mechanism should be used. Examples of applications in this category include surveillance and emergency preparedness systems.

- *Query-based*: This category is a hybrid of the previous categories. A query is injected in the network and cached by the sensor nodes. When a sensor node detects an event that matches a query, it will deliver the data to the interested base station or sink. A query may request that data be delivered periodically, *e.g.*, send the humidity of area X every T hours, or it may request that data be delivered immediately, *e.g.*, send the pressure reading when it reaches a threshold. Thus, the query-based model can assume any categories for the delivery of data.

In this section, two novel routing protocols for wireless sensor networks are proposed: The Periodic, Event-driven and Query-based Protocol (PEQ) and its cluster-based extension (CPEQ). The PEQ and CPEQ are publish/subscribe-based schemes that offer fault tolerance and low latency in order to meet the wireless sensor network requirements for delay-sensitive type of applications. PEQ can provide low latency for event notification, fast broken path reconfiguration, and high reliability in the delivery of events with low energy dissipation. Low latency is achieved by the use of the shortest path for the delivery of events. The sensor nodes can trigger a recovery mode when they detect that a neighbor node has failed to forward a packet. In the recovery phase, the sensor nodes find an alternative path locally and cooperatively using a small number of message exchanges. Basically, the sensor network in PEQ is configured initially as a hop tree rooted at the sink. The publish/subscribe paradigm is used to promote the interactions between the sensor node and the sink. The subscription and notification messages are propagated to the sensors through the dissemination tree. In addition, PEQ and CPEQ can be used for periodic data gathering to further minimize energy consumption by setting sleep periods

during the configuration of the dissemination tree.

In order to minimize traffic and latency, a cluster-based version of the PEQ mechanism was designed to relay the gathered data to the sink efficiently by reducing the network traffic and distributing the energy dissipation uniformly among the nodes. In the CPEQ protocol, the nodes with more residual energy are selected as cluster-head nodes that will use multi-hop communication to relay the data to the sink in a scalable manner. The strength of CPEQ lies in its simplicity and effectiveness of its event delivery process. The provision of low latency and reliable communications in the presence of high rates of node failure, and fast subscription of new queries, makes the PEQ and CPEQ algorithms a good choice to support applications in areas ranging from Health care, *e.g.*, monitoring of vital signs, localization of objects and people within health-care facilities, laboratories, etc, to surveillance applications, *e.g.*, military operations, emergency preparedness and response, etc.

This section is organized as follows. The subsection 3.3.2 describes the details of the PEQ algorithm and subsection 3.3.4 discuss its performance evaluation through simulations. Subsection 3.13 describes the CPEQ protocol, and its simulation experiments and results are described in subsection 3.3.6. Finally, subsection 3.3.7 highlights the potential improvements of both protocols and, lastly, the conclusions are given in section 3.4.

3.3.2 The PEQ Algorithm Overview

The main motivation of this protocol is driven by the need to provide the support for all of the following requirements simultaneously: low latency, reliability, and fast path recovery in the presence of failures. Although several interesting solutions have been reported in the literature, they basically do not support all three requirements simultaneously. Moreover, some solutions either require a special hardware or sophisticated processing at the nodes. The basic idea of the PEQ algorithm is to use ordinary motes, with no special hardware and simple algorithms at each node by using only the hop level as the main information to minimize data transmission. In the presence of node failure, a switch to

a fast recovery mode is done by keeping the exchange of information among neighbor as low as possible, differently from other solutions. Therefore, the primary design principles of the PEQ protocol are:

- *Principle 1*: Each sensor node should have local knowledge of the network, thereby providing a scalable and lightweight solution;
- *Principle 2*: Provide the shortest path for a node to reach the sink, thus minimizing the number of transmissions;
- *Principle 3*: Offer a fast and fault-tolerant data delivery. Routes should be recovered in the presence of node failures. A local mechanism that involves a minimum number of nodes is desired, instead of a flooding mechanism;
- *Principle 4*: Employ the publish/subscribe paradigm for submission and matching of queries to the sensor network;
- *Principle 5*: Exploit an overhearing technique in order to provide a reliable routing protocol.

PEQ is a routing protocol that is performed in three steps. The first step comprises the construction of the hop tree. The sink starts the process of building the hop tree by flooding the network with a configuration message. The second step involves the propagation of subscriptions to the sensor nodes. Finally, events can be delivered from the sensors to the sink by using the shortest route. The next sections describe the publish/subscribe paradigm as the mechanism for the sensors/sink interaction, followed by the description of each phase of the PEQ protocol.

The Publish/Subscribe Paradigm for Sensors/Sink Interaction

Wireless sensors networks usually consist of thousands of sensor nodes, each one producing events that are delivered to one or more static sinks. Several communication

paradigms can be used to promote the interaction between sensors and sinks. Examples of such paradigms include message passing, remote invocations, notifications, shared spaces and message queuing. The basic problem with these paradigms is that they fail to promote full decoupling between participants, making the system inflexible and unscalable [41]. Eugster *et. al.*, [42] present an excellent study of these paradigms and compare them to the publish/subscribe paradigm, which has received increased attention because it decouples consumers and producers in time, *i.e.*, publishers and subscribers do not need to be active in the interaction at the same time, in space, *i.e.*, publishers and subscribers do not need to know each other, and in flow, *i.e.*, publishers and subscribers do not need to be synchronized to interact with each other. In the publish/subscribe paradigm, one or more sinks receive event notifications from the sensor network. The sink expresses interest in a certain data by subscribing to sensor nodes.

Initial Configuration of the Sensor Network

In the proposed solution, the sensor nodes do not have global knowledge of the network, *i.e.*, each node knows only a small amount of local information about the neighbor nodes that are within its radio range. The initial configuration of the sensor network involves the setup of a dissemination tree called the hop tree. This mechanism is based on the sink flooding the network with a message that carries a hop value that will be stored, incremented and transmitted to its neighbor nodes. These neighbor nodes store the received hop value, increment it and forward to their neighbors. This process continues until every single sensor node has received its hop level. Thus, the hop level indicates the distance in hops from the sensor node to the sink. In addition, each node will also store the address of the source nodes that are one hop neighbors in order to alternate among the neighbors that will be used as a forwarding node. Due to the broadcast communication of a node, all its neighbors will receive the configuration message. Therefore, one node that has already forwarded a configuration message will most likely receive a configuration message from other neighbor nodes. This can generate loops and message explosion.

Algorithm 1 INITIAL CONFIGURATION

```
▷ Variables:
1: config_pkt                                  {Configuration packet: (hop, sourceID, sinkID, timestamp)}
2: routeTable = ∅                              {Routing table: (hop, destID, sinkID, timestamp)}
3: subTable = ∅                                {Table of subscriptions: (type, criteria, sinkID, timestamp)}
4: hop = nil                                   {Hop count or hop level}
5: sinkID = nil                                {Identification or address of the sink}
6: timestamp = nil                             {Timestamp of a packet}
7: destID = nil                                {Identification or address of the destination node}
8: sourceID = nil                              {Identification or address of the source node}

▷ Functions:
9: clock()                                     {Returns the current system time}
10: lookup(param)                              {Searches a table for the entry param}
11: getLocalAddress()                          {Returns the address of the current node}
12: add()                                      {Adds an entry to the table specified}
13: send()                                     {Transmits the packet}
14: recv()                                     {Action(s) taken when a packet is received}

  Action: {Sink starts the network configuration}
15: config_pkt.hop ← 1;                        {initializes the hop count}
16: config_pkt.sinkID ← getLocalAddress();
17: config_pkt.timestamp ← clock();            {sets the timestamp field}
18: config_pkt.send();.                        {broadcasts config_pkt}

  Action: {recv() - A sensor node receives a packet}
19: if (entry ← routeTable.lookup(config_pkt.sinkID)) then
20:     if (entry.hop > config_pkt.hop) then
21:         entry.hop ← config_pkt.hop;                {Updates local routing table}
22:         entry.destID ← config_pkt.senderID;        {Sets the forwarding node address (destID)}
23:         config_pkt.hop ← config_pkt.hop + 1;       {Increment hop count}
24:         config_pkt.sourceID ← getLocalAddress();   {Sets the source address}
25:         config_pkt.send()                          {Forward config_pkt}
26:     else
27:         config_pkt.drop();                         {Drops the packet silently}
28:     end if
29: else if not (entry) or (entry.hop == config_pkt.hop) then
30:     entry.sinkID ← config_pkt.sinkID;              {Update local parameters}
31:     entry.destID ← config_pkt.senderID;            {Sets the forwarding node address (destID)}
32:     entry.hop ← config_pkt.hop;
33:     entry.timestamp ← config_pkt.timestamp;
34:     routeTable.add(entry);                         {Adds the new entry to the table}
35:     config_pkt.hop ← config_pkt.hop + 1;           {Prepares the config packet}
36:     config_pkt.sourceID ← getLocalAddress();       {Sets the source address}
37:     config_pkt.send();                             {Broadcasts the configuration message}
38: end if
```

Therefore, a set of rules was established as part of the algorithm for the configuration message propagation in order to avoid these issues that cause energy waste and high traffic rates. One of the local rules establishes that when a node receives a configuration message from its neighbor, it compares the hop value of the received message with its local hop value. If the local hop value is greater than the received one, the node updates its hop count, increment this value and retransmit to its neighbors. In the case that the locally stored hop count is smaller or equal to the received hop value, the node does not update its hop count and simply drops the packet. Each node has two tables: a routing table, and a subscription table, as depicted in Algorithm 1. The routing table stores one entry for each sink the node is serving, whereas the subscription table holds the sink's subscriptions. During the initial configuration of the data dissemination tree, the routing table of each node is configured as follows: upon the reception of a configuration message, the sensor node will check if the source of that message can be used as a forwarding node. The decision is based on the hop level of the source node, as shown in Algorithm 1. If the node's local hop count, relative to the sink specified by sinkID, is greater than the hop level carried by the received message, the node will update its routing table and set its forwarding node to be the node from which it received the message. Thereafter, the node increments the hop count and broadcasts the configuration message to its neighbors. Thus, each node in the network will learn and store the source address in its routing table that will be used later on to forward data to the sink. In case the locally stored hop is smaller or equal to the received hop, the node does not update its hop and does not retransmit it. This mechanism is depicted in Figure 3.5(a) and (b). This process continues until every node has a forwarding address in order to reach the sink that started the network configuration.

Propagation of Subscriptions

In the publish/subscribe paradigm, the sink needs to subscribe to the kind of data it is interested in, so that it can be notified about events that the sensor network collects

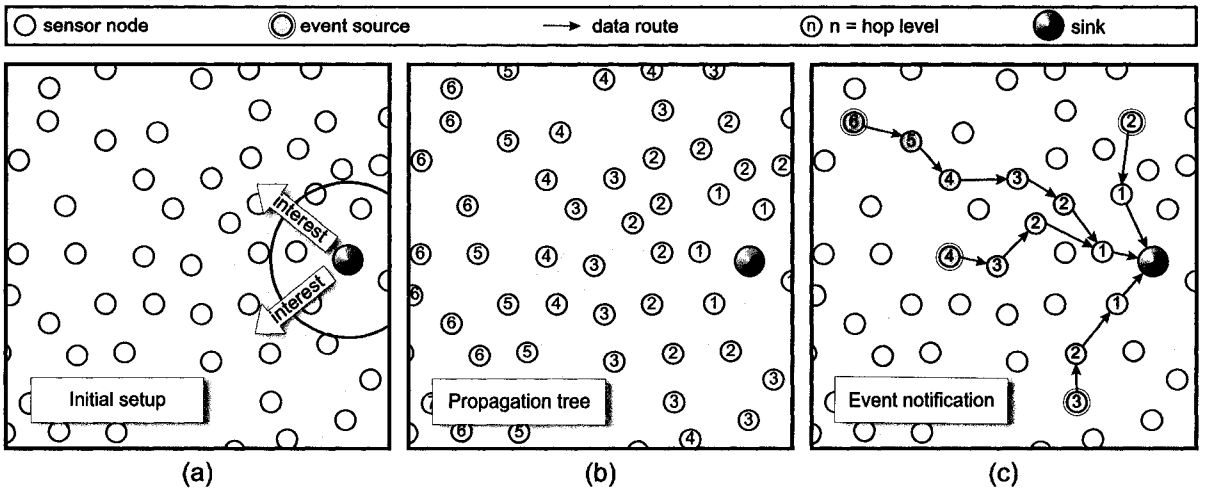


Figure 3.5: The steps of the PEQ algorithm

from the physical environment. For instance, the sink injects a subscription message that carries one or more criteria, *e.g.*, temperature > 60, presence of smoke, etc. If a sensor node can provide the kind of data requested in a subscription, it caches it in its subscription table. After the initial configuration, the only information a node has is the hop level and its forwarding neighbor node. In the absence of any information about which node of the network can satisfy the sink interest, a sensor node will propagate the initial subscriptions by flooding the network with this interest. This is convenient for applications that do not change their interests or that do not query the network very often. For instance, the application sets its criteria for the day, injects the subscription once, and waits for the sensor network reports. Thus, this mechanism is suitable for event-driven class of applications.

As an alternative, the subscription propagation phase can be suppressed by piggy-backing subscriptions into the initial configuration packet. Thus, the broadcasting of a configuration message that carries subscriptions from the sink will not only re-configure the network, *i.e.*, fix any broken paths, but it will also disseminate subscriptions in one restricted flooding flow.

In a periodic application scenario, PEQ can be used to further minimize energy

consumption by setting sleep periods during the configuration of the dissemination tree, or during the propagation of subscriptions. The sink spreads its subscriptions that carry the periods for data gathering. Assuming that the clocks of all nodes are synchronized, the nodes will set sleeping periods when no data is required. For instance, if the sink subscribes its interest that a certain data should be delivered starting at a specific time T. All the nodes in sleep mode will only turn on their radios when the time T is approaching.

Data Delivery to the Sink

When a sensor node gathers data from its sensing module, it first checks its subscription list to determine if there is any registered subscription that matches the data. If a criterion is met, the node will forward the data to the sink that has subscribed. The algorithm is described in Algorithm 2. Basically, the node assembles a data notification packet that contains the following fields: type, value, sinkID, and timestamp. Thereafter, the node checks its routing table and uses the destID address that leads to the sink in order to forward the packet towards the sink. Each sensor node that receives the data notification packet will look up its routing table for an entry that contains the sinkID and will retrieve the destID address to forward the data towards the sink. This process repeats for each node in the path until the data reaches the sink. The actions taken when an intermediate node receives a data packet is shown in Algorithm 3. In addition, each node will randomly select one of the entries it has found in its routing table that leads to the sink. Thus, a node will alternate its forwarding node every time it has a packet to relay to the sink, as depicted in Figure 3.6, in which a sample of the sensor network shows some nodes with multiple choices of forwarding neighbors. For instance, one of the level-3 nodes has three forwarding nodes and it will randomly select which one to use each time it needs to relay a packet. This mechanism avoids depleting the energy of a single node caused by relaying packets through the same path repeatedly.

During the initial configuration phase, each node learns only the addresses of the nodes that are in a previous hop level. This characteristic makes it easy for a node to

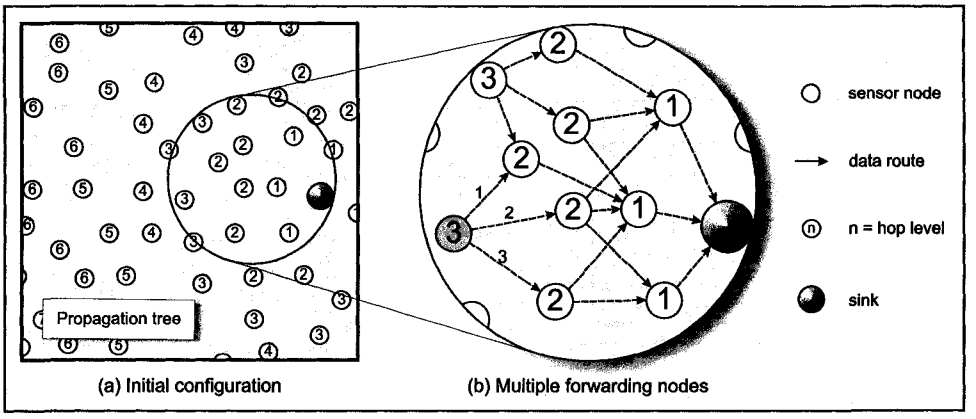


Figure 3.6: A sample of the multiple forwarding nodes in PEQ.

select the shortest path, besides avoiding message loops. Suppose that the sink sends a subscription to the network, and considering that the level-3 node, *i.e.*, the darker one, in Figure 3.6 is the sensor node that produces an event that meets the subscription criterion. Note that the arrows indicate the links that could form alternative paths. The level-3 node has three route choices, level-2 nodes have their own number of routes, and so on. Multiples packets will be delivered through alternate paths. An important feature of the configuration of the hops level can be observed in the notification transmission phase. When a node receives a data notification packet from a neighbor node, it only retransmits the message if the node belongs to a one-hop level higher than the current node. For instance, only level-2 nodes will forward the information received from level-3 nodes.

Recovery from Disrupted Routes

The routes created for from sensor nodes to the sink are the shortest possible and efficient in terms of number of hops. However, if a single node in the path fails, it will cause route disruption, thereby preventing data delivery to the sink. A sensor node may fail due to energy depletion, physical destruction, communication blockage and interference, etc. Several routing algorithms that support path recovery for sensor networks have been proposed in the literature. Some of them are based on periodic flooding mechanisms,

Algorithm 2 DATA DELIVERY

```
▷ Variables:
1: data_pkt                          {Data notification packet: (data, source, destID, sinkID, timestamp)}
2: routeTable = ∅                    {Routing table: (hop, destID, sinkID, timestamp)}
3: subTable = ∅                      {Table of subscriptions: (type, criteria, sinkID, timestamp)}
4: F = ∅                             {F = {f1, f2, ..., fn} is the set of possible forwarding neighbors of a node}
5: hop = nil                         {Hop count or hop level}
6: sinkID = nil                      {Identification or address of the sink}
7: timestamp = nil                   {Timestamp of a packet}
8: destID = nil                      {Identification or address of the destination node}
9: source = nil                      {Identification or address of the source node}

▷ Functions:
10: clock()                          {Returns the current system time}
11: lookup(param)                    {Searches a table for the entry param}
12: getLocalAddress()                {Returns the address of the current node}
13: add()                            {Adds an entry to the table specified}
14: send()                           {Transmits the packet}
15: recv()                           {Action(s) taken when a packet is received}

 Action: {Sensor node has data to deliver}
16: tmp_entry ← subTable.lookup(type);          {Retrieves the entry that matches the data}
17: repeat
18:     entry ← routeTable.lookup(tmp_entry.sinkID);   {Retrieves the sinkID entry}
19:     F ← entry.destID;                       {Stores the multiple forwarding addresses}
20: until (entry ≠ EOF)                         {Prepares the data notification packet}
21: data_pkt.destID ← RANDOM(F);                {Sets the forwarding node randomly from the set F}
22: data_pkt.timestamp ← clock();
23: data_pkt.source ← getLocalAddress();        {Sets the source address with its own address}
24: data_pkt.sinkID ← entry.sinkID;             {broadcasts config_pkt}
25: data_pkt.data ← data;                       {Data gathered from the sensor}
26: data_pkt.send();                            {Sends the data packet}
```

such as the Directed Diffusion [60] and Hill *et. al.*, [56]. These mechanisms rely on the sink to trigger the periodic flooding in order to repair broken paths and to discover new routes to forward traffic around faulty nodes. However, the periodic flooding mechanism is not satisfactory in terms of energy savings due to the extra overhead. Furthermore, these periodic algorithms are unable to route data around failed nodes until the next flooding, thereby it increases the packet drop rate and delay. The PEQ's path repair algorithm was primarily designed based on acknowledgments. It comprised two parts: the failure detection of the forwarding neighbor node, and the selection of a new destination. When a sensor node forwards data to its forwarding neighbor, it simply sends the data packet and sets a timeout timer and waits for the neighbor's acknowledgment. If the node receives the ACK packet from its neighbor, it can infer that the neighbor is working properly. However, this ACK-based path repair scheme introduced extra traffic, packet

Algorithm 3 DATA FORWARDING

```
▷ Variables:
1: data_pkt                          {Data notification packet: (data, source, destID, sinkID, timestamp)}
2: routeTable = ∅                    {Routing table: (hop, destID, sinkID, timestamp)}
3: F = ∅                             {F = {f1, f2, ..., fn} is the set of possible forwarding neighbors of a node}
4: sinkID = nil                      {Identification or address of the sink}
5: destID = nil                      {Identification or address of the destination node}

▷ Functions:
6: lookup(param)                     {Searches a table for the entry param}
7: send()                            {Transmits the packet}
8: recv()                            {Action(s) taken when a packet is received}

 Action: {recv() - An intermediate sensor node receives a DATA packet}
9: repeat
10:    entry ← routeTable.lookup(data_pkt.sinkID);     {Retrieves the sinkID entry}
11:    F ← entry.destID;                               {Stores the multiple forwarding addresses}
12: until (entry ≠ EOF)
13: data_pkt.destID ← RANDOM(F);     {Sets the forwarding node randomly from the set F}
14: data_pkt.send();                 {Forwards the data packet}
```

delay and retransmissions of packets that were properly forwarded by the neighbors but that were not acknowledged due to uplink problems. Therefore, the current route repair mechanism is based on the overhearing technique. Hence, suppose a sensor node sends a data packet to its forwarding neighbor and sets a timer T. Upon receiving the packet, the neighbor node will attempt to forward it to its neighbors towards the sink, as already described in Algorithm 3. If the node overhears its neighbor forwarding the packet within the specified timer T, the node can infer that its neighbor is working. Otherwise, a problem must have occurred with the neighbor and another node should be selected as the new forwarding neighbor. Thus, the node will attempt to select another destination from the set of forwarding neighbors $F = \{f_1, f_2, ..., f_n\}$, where f_i is the address of a forwarding neighbor learned during the setup of the dissemination tree. If there is no forwarding node, *i.e.*, if $F = \emptyset$, the node switches to a neighbor discovery state. The node immediately broadcasts a SEARCH message. Each neighbor that receives the SEARCH packet will reply with a message that carries its hop level and address. Thereafter, the node will create a list N with all the neighbor nodes that replied such that $N = \{n_1, n_2, ..., n_n\}$, where N is sorted by hop level and n_i is the entry for the i^{th} neighbor in order of preference. Hence, the node will choose the neighbor with the lowest hop level as its new forwarding node. If there is no available neighbor n_i with

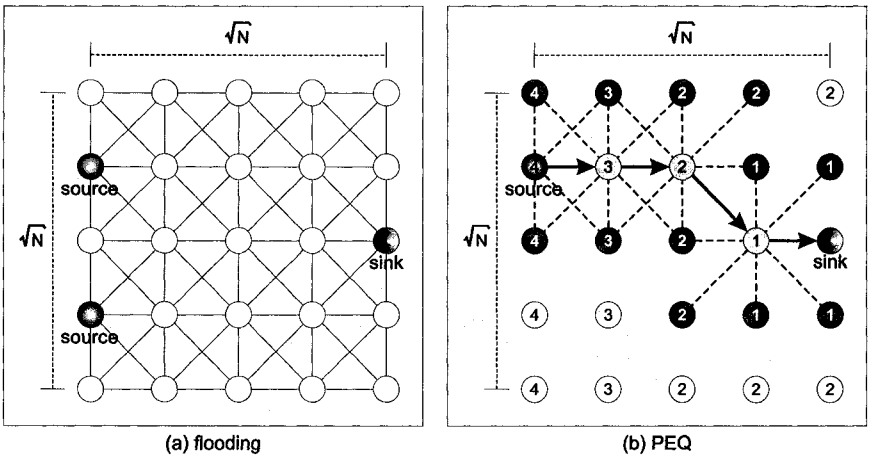


Figure 3.7: Sample of the grid topology.

hop level $h_i \in H$, where $H = \{h_1, h_2, ..., h_n\}$ is the set of hop levels of the nodes in N, that is less or equal to the hop level h_s of the node that started the discovery phase n_s, *i.e.*, $\forall h_i \in H \nexists h_i \mid h_i \leq h_s$, the node will send the packet back to the node from which it received the packet. This node will try to find another route by starting the neighbor discovery phase itself, but excluding the neighbor that backtracked to avoid loops.

If no neighbor replies to the SEARCH message, *i.e.*, $N = \emptyset$, the node is isolated and its transmission will not reach any neighbor. The node will attempt to broadcast another SEARCH message after a timer expires. Another solution would be to reconfigure its radio module to increase its coverage area.

3.3.3 Data Delivery Cost Function

In this section, a theoretical representation of the data delivery cost is discussed for the flooding and the PEQ protocols. Following the ideas presented in [61], a simple idealized setting is investigated, in which N sensor nodes are deployed on a square grid, as seen in Figure 3.7. The links between pairs of nodes that can communicate with each other are shown in Figure 3.7(a). A total of n source nodes are deployed on the left edge of the grid, and the sink is place on the right edge. The cost of transmission and reception of one event from each source to the sink is defined as one unit for transmission and one

unit for reception. We take into consideration more realistic scenarios in the simulation experiments presented in section 3.3.4.

Communication Cost for Flooding the Network

The cost of flooding the network with n events (one event per source) is denoted by:

$$C_f(N,n) = C_{TX}(N,n) + C_{RX}(N,n) \tag{3.3}$$

where N is the total number of sensor nodes and C_{TX} and C_{RX} are the costs of transmission and reception of one event, respectively. The transmission cost C_{TX} is given by:

$$C_{TX}(N,n) = nN \tag{3.4}$$

because each node will transmit the event once (assuming that one event is transmitted in one packet). The reception cost C_{RX} is determined by the number of links in the network. As depicted in Figure 3.7(a), there are $2(\sqrt{N}-1)^2$ diagonal links, $(\sqrt{N}-1)\sqrt{N}$ horizontal links, and $(\sqrt{N}-1)\sqrt{N}$ vertical links. In addition, each node will also receive the same event from all neighbors, *i.e.*, the reception cost per link will be $2 \times n$ events. The total reception cost is given by:

$$C_{RX}(N,n) = 2n[2(\sqrt{N}-1)^2 + 2(\sqrt{N}-1)\sqrt{N}] \tag{3.5}$$

Finally, the total cost of flooding the network with n events is determined by:

$$\begin{aligned} C_f(N,n) &= nN + 2n[2(\sqrt{N}-1)^2 + 2(\sqrt{N}-1)\sqrt{N}] \\ &= nN + 2n[2(\sqrt{N}-1)(\sqrt{N}-1) + 2(\sqrt{N}-1)\sqrt{N}] \\ &= nN + 2n[2(\sqrt{N}-1)(\sqrt{N}+\sqrt{N}-1)] \\ C_f(N,n) &= nN + 4n(\sqrt{N}-1)(2\sqrt{N}-1) \end{aligned} \tag{3.6}$$

Therefore, the complexity of the data delivery cost for flooding is equal to $O(nN)$. The analysis of the PEQ protocol is discussed in the next section.

Data Delivery Cost in PEQ

In PEQ, each node has enough routing information to reach the sink using the shortest path after the configuration phase. A characteristic of the PEQ protocol determines that a node in a certain hop level h_i only forwards its data packets to neighbor nodes at hop level $h_i - 1$. Therefore, a shortest path does not include a vertical link. The data delivery cost $C_P(N, n)$ for PEQ is determined by the number of links on the shortest path from the source to the sink depicted in Figure 3.7(b). The cost of delivering n events to the sink is denoted by:

$$C_p(N, n) = C_{TX}(N, n) + C_{RX}(N, n) \tag{3.7}$$

Roughly put, the transmission cost C_{TX} is determined by the number of transmissions on the shortest path to the sink:

$$C_{TX} = n(\sqrt{N} - 1) \tag{3.8}$$

because only the nodes in the shortest path will retransmit the event. The reception cost C_{RX} is determined by:

$$\begin{aligned} C_{RX} &= 2n(\sqrt{N} - 1) + 6(\sqrt{N} - 1) \\ &= (2n + 6)(\sqrt{N} - 1) \end{aligned} \tag{3.9}$$

Therefore, the total cost of delivering n events in PEQ is denoted by:

$$\begin{aligned} C_p(N, n) &= n(\sqrt{N} - 1) + (2n + 6)(\sqrt{N} - 1) \\ &= (3n + 6)(\sqrt{N} - 1) \end{aligned} \tag{3.10}$$

As can be deducted from equation 3.10, the complexity of the data delivery cost of PEQ is equal to $O(n\sqrt{N})$.

3.3.4 PEQ's Simulation Experiments and Results

Simulation Scenario and Metrics

The PEQ protocol was implemented and evaluated using the ns-2 simulator [87]. The simulation scenarios consist of several sensor fields of different population sizes ranging from 100 to 500 sensor nodes. The nodes were randomly placed on an area of $150m^2$ and a fixed workload of 5 sources and 1 sink were used. The size of the field is increased to keep the node density constant. The radio range of the nodes was set to 20 meters in order to mimic realistic sensor radio modules. The source nodes were placed as farthest away as possible from the sink so that when the network size is increased, the length of the routes to the sink also increases in terms of hops. Therefore, the impact of network size on the performance of the PEQ protocol can be more noticeable than a random placement of the nodes. Node failures are simulated by turning off a fixed fraction of nodes. These nodes were randomly chosen from the sensor field and turned off at a random time and stayed off until the end of the simulation.

Table 3.1: Summary of the PEQ's simulation parameters.

Parameter	**Value**
Simulation time	1000 sec
Simulation area	150 × 150m
Number of nodes N	100-500
Radio range	20m
Number of source nodes	5
Source node data rate	2pkts/s
Bandwidth	34.8Kbits
Packet size	32Bytes
DATA packet size	64Bytes
TX power dissipation	0.01488W
RX power dissipation	0.01250W
Idle power dissipation	0.01236W

The simulation parameters used in this set of experiments are listed in Table 3.1. These parameters are based on the values reported in the Directed Diffusion (DD) research work [60], with some modifications. The source nodes' data rate was chosen to be 2 events per second.

The simulations use the IEEE 802.11 MAC layer implemented in the ns-2 simulator. As reported in [60], this is not a satisfactory choice of MAC layer, since a TDMA-based MAC would be more appropriate than a RTS/CTS approach, in terms of energy efficiency, because an 802.11 radio in idle state consumes as much power as when it is listening to the channel. Therefore, the simulation experiments presented in this chapter follow the ideas of [60] to modify the ns-2 energy model. The energy model is based on the RFM TR1000 [118]. The power values shown in Table 3.1 were extracted from [104].

The event delivery delay and the delivery ratio are critical metrics for the performance evaluation of sensor network protocols for event-driven and query-based application scenarios. Thus, PEQ is evaluated through the following metrics:

- *Average Delay*: Average latency from the moment a data packet is transmitted to the moment it is received at the sink;
- *Average Event Delivery Ratio*: The ratio of the number of received data packets by the number of originally sent;
- *Average Dissipated Energy*: The total power consumption of the network per number of nodes.

The mean is calculated over 10 simulation runs for each metric with 95% of confidence interval.

The Performance of PEQ

The performance results of PEQ are compared the ones of Directed Diffusion (DD). DD is considered as a benchmark by the research community and is the most similar protocol

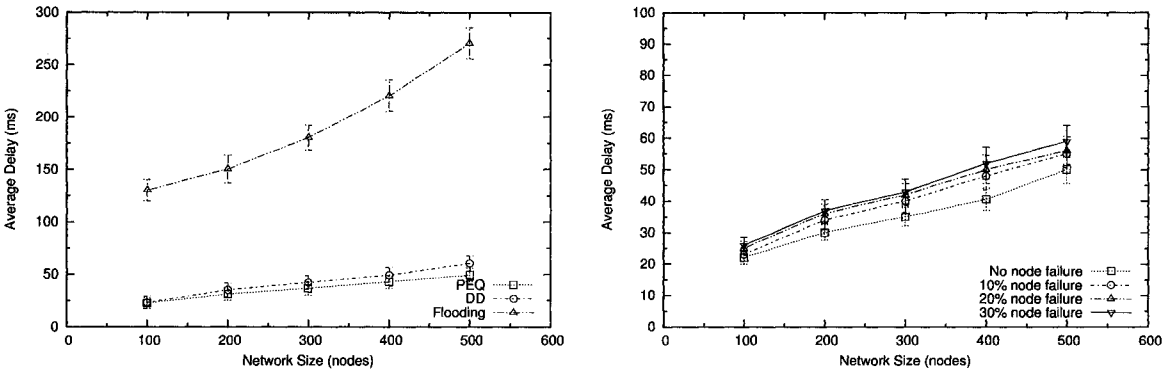
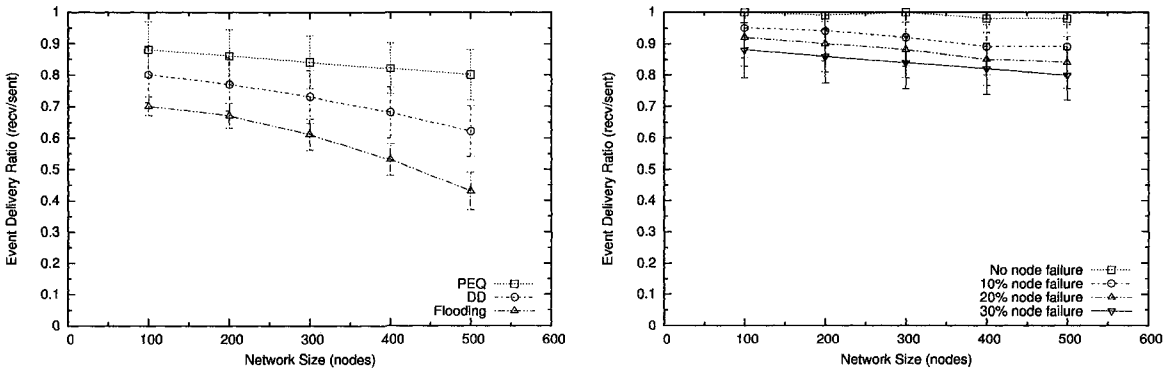


Figure 3.8: End-to-end average delay comparison.

to the PEQ, *i.e.*, both lie in the same category of data-centric routing protocols. PEQ is also compared to a simple flooding routing mechanism, which is the worst case scenario. The average delay is particularly important in event-driven and query-based applications, which demand fast and reliable response times. For instance, a query to track an object or person in a building subject to an emergency situation, or the detection of fire or high temperatures, require fast system response and reliable delivery. Low latency is crucial since an object being tracked can move fast in a few seconds and information about its location may be out of date very quickly, preventing actions that could save lives if taken in a short time. PEQ uses the subscription message to propagate the initial configuration that builds the path to the sink and when the source node receives the subscription, it uses this path to deliver data to the sink. PEQ should at least outperform the flooding mechanism, as can be seen in Figure 3.8(a). PEQ also shows a significant lower delay than Directed Diffusion. The main reason is that DD adds extra overhead through the reinforcement mechanism, which uses multiple paths to send the information to the sink and, sometimes, the path is not the shortest one. The average latency for different node failure rates is depicted in Figure 3.8(b). As network size increases, the the number of hops an event has to travel from source to sink also increases, and so does the average packet delay. Failure rate has an impact in the observed delay, as shown in Figure 3.8(b).



(a) (b)

Figure 3.9: Average packet delivery success rate.

For instance, in a network size of 500 nodes, the average delay increases from 49ms to 58ms, which is an acceptable latency for the event-driven and query-based application scenarios. This increase in delay is due to the fact that, in order to repair a broken path, the algorithm has to find "alive" nodes. As we increase the number of failed node, the discovered paths may become longer, thereby adding extra delay to the packets.

Sensor network reliability can be measured by its average event delivery ratio, which reflects the success ratio of packets transmitted to the sink. Figure 3.9 shows that PEQ is able to maintain a reasonable event delivery ratio even at a high percentage of failed nodes.

Figure 3.10(a) presents the energy consumption of the PEQ, DD, and flooding algorithms. The consumption of PEQ and DD are very similar, with a small difference due to the extra overhead that DD introduces with the reinforcement mechanism. However, in our simulations, the idle state of the radios consume as much energy as receiving packets. Therefore, idle state energy consumption dominates the average calculate in this experiment. Figure 3.10(b) shows the average dissipated energy per node in the presence of different node failure rates.

In this section segment of our simulation, the experiments are conducted with different data rates of 5, 10 and 15 data packets per second in order to investigate the scalability

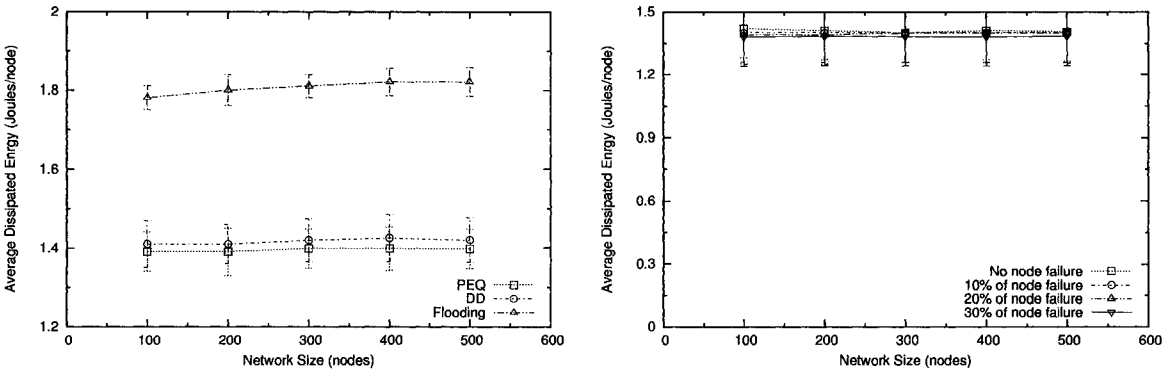
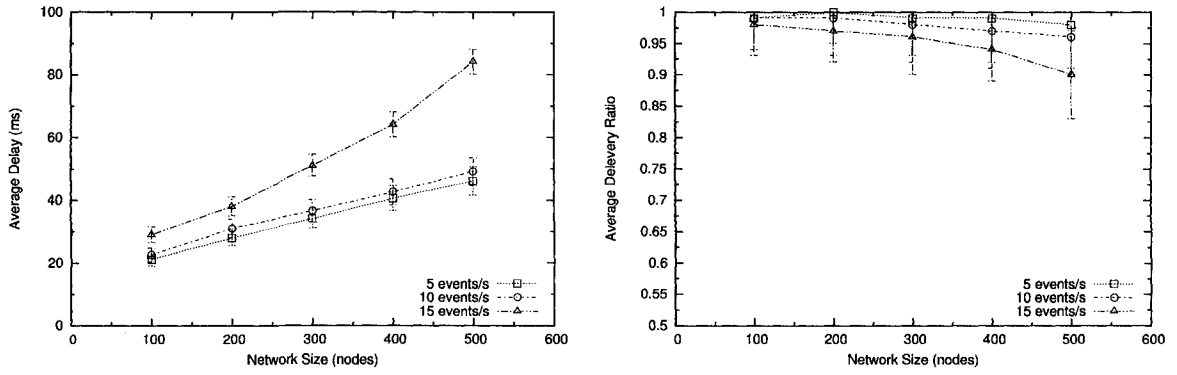


Figure 3.10: Energy consumption per node.

of our approach under different traffic conditions. Figure 3.11(a) shows that an increase in the data rate has raised the latency observed, especially when 15 or more events per second were generated by the sources. This was expected since a rate of 15 events per second generates much more traffic and therefore more packet collisions and losses, as shown in Figure 3.11(b), in which the average delivery ratio decreases when data rate is increased.

The sensor field considered here has only one sink, so that the algorithm could be better evaluated in a traffic jam situation. The nodes closer to the sink have to deal with a great number of packets per second, limiting the performance of the network and impacting on its lifetime. From the experiments, it can be seen that PEQ can provide low latency for event notification, fast broken path reconfiguration, and high reliability in the delivery of the events with low energy dissipation. Low latency is achieved by the use of the shortest path for the delivery of events.

A cluster-based approach CPEQ, based on PEQ, is described in the next section. CPEQ has been designed to minimize latency even further, specially under high traffic such as in the case of an emergency situation, such as detection fire, gas leaks, etc.



(a) (b)

Figure 3.11: End-to-end delay and packet delivery ratio.

3.3.5 The Cluster-Based CPEQ Algorithm Overview

The sensor nodes in a wireless sensor network may generate heavy data traffic, especially in an emergency scenario. For instance, nodes may detect that the temperature of part of a building is increasing; other nodes may detect fire and all these data might be sent to a sink that will triggers immediately some nodes or even video sensors to track people inside a building. The sensor nodes in a certain region may detect the same event and send redundant data to the sink. In order to avoid redundancy, which generates unnecessary traffic and dissipates more energy, a data aggregation method is necessary [54][60]. Thus, a cluster-based approach was devised that groups sensor nodes to efficiently relay the sensed data to the sink. The CPEQ protocol adopts a cluster-based approach where nodes with more residual energy are selected as cluster-head nodes. A cluster-head node builds up a cluster and the nodes within the cluster forward data to the cluster-head. Thereafter, the cluster-head executes data aggregation or fusion on the gathered data and relays the results to the sink. Every node on the network can become a cluster-head for a specific period of time. When the time expires, other nodes are selected a cluster-heads. The main goals of the CPEQ protocol are to distribute the energy dissipation among the nodes uniformly, and to reduce traffic in the network. The

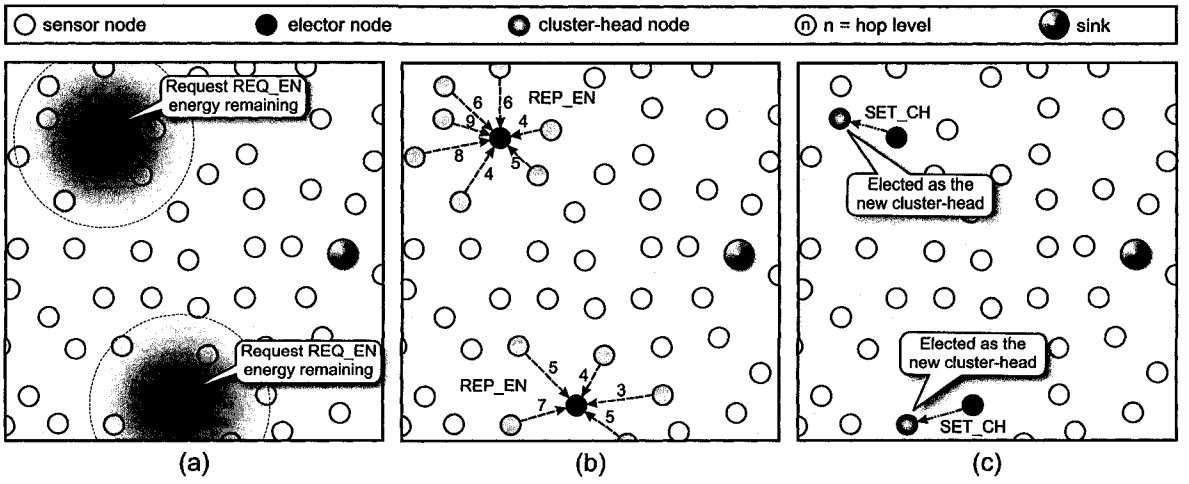


Figure 3.12: Cluster-head selection mechanism.

CPEQ algorithm is performed in five steps: initial configuration; selection of cluster-heads; cluster configuration; data transmission to the cluster-head; and data delivery to the sink. Each step will be discussed in details in the following sections.

Initial Configuration

The sensor nodes in a certain region may detect the same events and send redundant data to the sink. In order to avoid redundancy, which generates unnecessary traffic and dissipates energy, a data aggregation method is necessary. The CPEQ protocol is based on the PEQ mechanisms and it adopts a cluster-based approach in which the nodes with more residual energy are selected as cluster-head nodes. The cluster-head node builds up a cluster and the nodes within its cluster send their events to the cluster-head. Thereafter, the cluster-head executes an aggregation function, e.g., the averaged temperature or to conceal redundant data, before relaying the data to the sink. The CPEQ protocol configures the dissemination tree using the PEQ's algorithm with a simple modification that is the addition of two fields that contain the percentage of nodes that can become cluster-heads, and the duration of the round T_r.

Cluster-Head Selection

The cluster-head selection scheme is based on the ideas presented in LEACH [53], which is a benchmark in cluster-based routing for sensor networks. After the initial configuration, any node in the network can become a cluster-head with a certain probability. For instance, if the desired percentage of cluster-heads is 5%, the probability of a node becoming a cluster-head will be $p = 0.05$. Therefore, Each node generates a random number between 0 and 1 and if this number is less than the probability p, the node will initiate the cluster-head selection mechanism. The "elector" node requests the energy level from its immediate neighbors by sending a REQ_EN packet, as shown in Figure 3.12(a). Each neighbor replies with a REP_EN packet that contains its address and the amount of energy remaining, as can be seen in 3.12(b). The elector node selects the neighbor with the highest residual energy as the new cluster-head by sending it a SET_CH packet, as depicted in Figure 3.12(c). In the example, the nodes with energy levels 7 and 9 are selected as the new cluster-heads. The cluster-head selection scheme is executed periodically, in rounds. A node remains in the cluster-head state for a specific period of time T_r. When T_r expires, the nodes will clean their cluster-related routing information and restart the selection mechanism (see Algorithm 4).

Configuration of the Clusters

The new selected cluster-head node is responsible for notifying its neighbors about its new status. Each cluster-head will start the cluster configuration by broadcasting the notification packet CH_NTF in order to build a cluster. This mechanism acts exactly the same as the initial configuration algorithm of PEQ. In order to limit the size of a cluster, the CH_NTF packet carries a time-to-live field. The result of the cluster configuration scheme can be seen in Figure 3.13(a) for $TTL = 2$. It is possible that a sensor node receives CH_NTF packets from more than one cluster-head. The node will join the cluster from which it received the packet with the lowest TTL. In the case of ties, the node will join the first cluster-head from which it received the notification CH_NTF.

Algorithm 4 CLUSTER-HEAD SELECTION

▷ **Variables:**
1: *REQ_EN* {Energy request packet: (source, destID, timestamp)}
2: *REP_EN* {Energy reply packet: (energy, source, destID, timestamp)}
3: *SET_CH* {Set cluster-head packet: (source, destID, timestamp)}
4: $N = \emptyset$ {$N = \{n_1, n_2, ..., n_i\}$ is the set of nodes that reply REQ_EN}
5: $destID = nil$ {Identification or address of the destination node}
6: T_r {Duration of a round}
7: T_{REQ} {Timer to receive the replies from the neighbor nodes}
8: *prob* {Probability of becoming a "selector" node}

▷ **Functions:**
9: *max(param)* {Returns the maximum value within a set}
10: *send()* {Transmits the packet}
11: *drop()* {Silently drops a packet}
12: *recv()* {Action(s) taken when a packet is received}
13: *random(param)* {Returns a random number}
14: *getLocalAddress()* {Returns the address of the current node}
15: *getEnergy()* {Returns the energy remaining of the current node}
16: *setTimer(time)* {Starts a timer}

Action: {The round starts. Determines if the node will be a "selector" node}
17: **if** ($prob \geq random(1)$) **then**
18: $REQ\_EN.source \leftarrow getLocalAddress()$; {Prepares REQ_EN}
19: $REQ\_EN.destID \leftarrow BROADCAST$;
20: $REQ\_EN.send()$; {Broadcasts the energy request packet}
21: $setTimer(T_{REQ})$; {Sets the timer to receive replies from neighbors}
22: **end if**
23: $setTimer(T_r)$ {Another round starts when this timer expires}

Action: {*recv*() - A node receives a REQ_EN packet}
24: $REP\_EN.energy \leftarrow getEnergy()$;
25: $REP\_EN.destID \leftarrow REQ\_EN.source$; {Sets the destination to be the selector node}
26: $REP\_EN.send()$; {Sends the reply message}

Action: {*recv*() - The "selector" node receives the REP_EN message}
27: $N \leftarrow REP\_EN$; {Stores the addresses of neighbors that replied}
28: **if** $T_{REQ}expired$ **then**
29: $node \leftarrow max(N)$; {Returns the node with more energy}
30: $SET\_CH.destID \leftarrow node.address$; {Sets the destination as the node with more energy remaining}
31: $SET\_CH.send()$; {Sends the SET_CH to the new cluster-head}
32: **end if**

Data Transmission to Cluster-Heads

When a node detects a phenomenon in the environment, the sensed data must be relayed to the cluster-head. The data routing algorithm is the same employed by PEQ when it routes data to the sink, in which nodes use their forwarding addresses that were learned during the configuration step. In the CPEQ protocol, the cluster-head can be thought of as a sink for its cluster. Figure 3.13(b) shows two cluster-heads receiving data from their cluster nodes. The CPEQ protocol also inherits the path repair mechanism from PEQ.

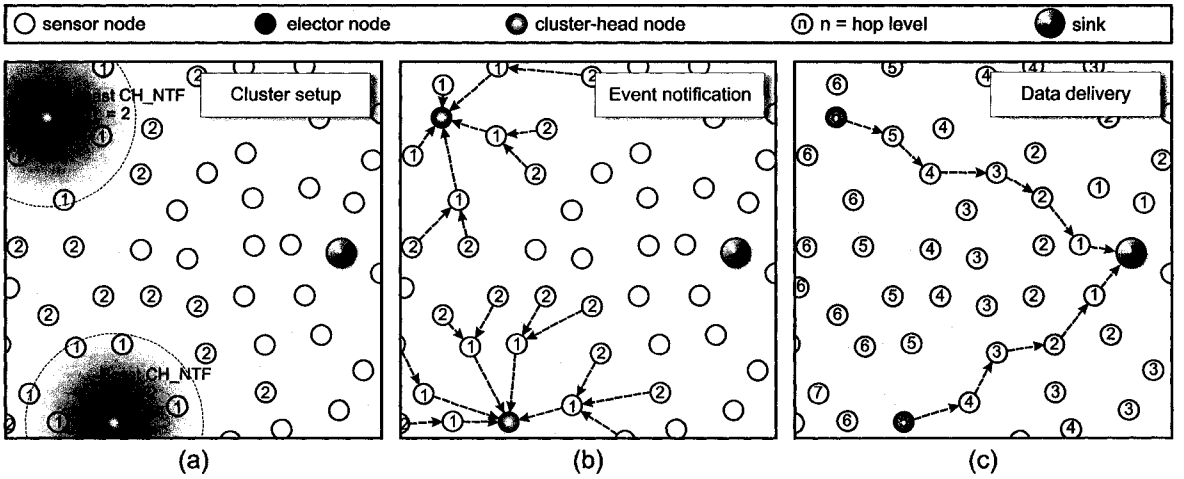


Figure 3.13: Cluster setup and data propagation

Data Transmission to the Sink

After receiving data from the sensor nodes within its cluster, each cluster-head will forward its data to the sink. In the LEACH protocol [53], the communication among cluster-heads and the sink is performed in one hop. Therefore, the cluster heads have to reserve more energy to transmit at long distances. This mechanism works only for small networks because every cluster head can reach the Sink in a small field. However, in large scale networks, cluster-heads might be unable to communicate directly with the sink. The CPEQ protocol, on the other hand, is based on multi-hop communication among cluster heads and the sink. A cluster-head forwards its data to the sink through the shortest path, which was configured during the initial configuration step, as can be seen in Figure 3.13(c). It is possible that some nodes do not belong to any cluster after a cluster configuration phase. In this case, the nodes can use the routes found during the initial configuration and relay data directly to the sink through those routes. Other solutions [22][125] employ cluster-heads with specialized hardware that consume more energy and communication resources. These solutions have high deployment costs and need special care with the distribution of the cluster heads, as they will remain fixed throughout the network lifetime.

3.3.6 Performance Evaluation of the CPEQ Protocol

Simulation Scenarios and Metrics

The CPEQ protocol was implemented in the network simulator ns-2 [87]. A fixed network size of 500 sensor nodes is utilized in the first set of experiments. The data rate of source nodes is est to 2 events per second. The time to live for cluster construction is set to $TTL = 2$, and the percentage of cluster-head nodes in the network is based on LEACH [53], *i.e.*, 5% or $p = 0.05$. The percentage of source nodes in the network ranges from 2% to 16% in order to reflect low and high traffic rates. In addition, the source nodes are randomly selected from the sensor field. The duration of a round T_r is 50 seconds. Table 3.2 summarizes the simulation parameters.

The CPEQ protocol is evaluated through three metrics: average event delay, average event delivery ratio, and average dissipated energy. The PEQ and Directed Diffusion schemes were simulated under the same scenario and parameters and their results compared.

Table 3.2: Summary of the CPEQ's simulation parameters.

Parameter	**Value**
Simulation time	1000 sec
Simulation area	$150m \times 150m - 400m \times 400m$
Number of nodes N	100 - 500
Radio range	20m
Percentage of source nodes	2 - 16%
Percentage of cluster-heads	5%
Round duration T_r	50 sec
Source node data rate	2pkts/s
Time-to-live TTL	2
Bandwidth	34.8Kbits
Packet size	32Bytes
DATA packet size	64Bytes
TX power dissipation	0.01488W
RX power dissipation	0.01250W
Idle power dissipation	0.01236W

The Performance of CPEQ

As outlined in section 3.3.4, PEQ shows low latency for event delivery, specially when the network comprises a small number of source nodes. Therefore, a cluster-based variation of PEQ was designed to deal with a large number of source nodes. In this section, the performance of CPEQ is evaluated and compared to PEQ, Directed Diffusion (DD) and LEACH protocols.

In this first set of simulations, 100 nodes are scattered across a 150m square region and the number of source nodes is increased from 2 to 16% of the total number of nodes.

Recall that the average delay is the end-to-end delay observed between a source node and the sink. As expected, with low data traffic, CPEQ is outperformed by the other protocols due to its extra delay, since each cluster-head must wait to receive data from the nodes within its cluster before performing data aggregation or fusion. In addition, the delivery paths are a bit longer since data flows from sensor nodes to the cluster-head and from cluster-heads to the sinks. As the number of source nodes increases, *e.g.*, in a typical emergency situation, CPEQ is able to sustain a stable delay since it reduces the amount of data packets that are delivered through the network by performing a simple aggregation function on each cluster-head. The sensor nodes in PEQ and Directed Diffusion have to forward all data generated by the source nodes to the sink, thereby contributing to the increase in delay. Figure 3.14(a) shows the stable delay observed in CPEQ and LEACH, and the abrupt increase in delay for PEQ and DD as the percentage of source nodes increases. In CPEQ and LEACH, the dynamics of the clustering configuration, *i.e.*, the selection and notification of cluster-heads, add extra overhead. As observed in Figure 3.14(a), PEQ and Directed Diffusion provide better results in terms of energy dissipation when just a few source nodes are present in the network. LEACH outperforms CPEQ in terms of packet delay since LEACH uses direct communications and CPEQ multi-hop communications.

However, when the number of source nodes increases, CPEQ outperforms PEQ and DD protocols by reducing the data traffic and, consequently, the number of transmissions

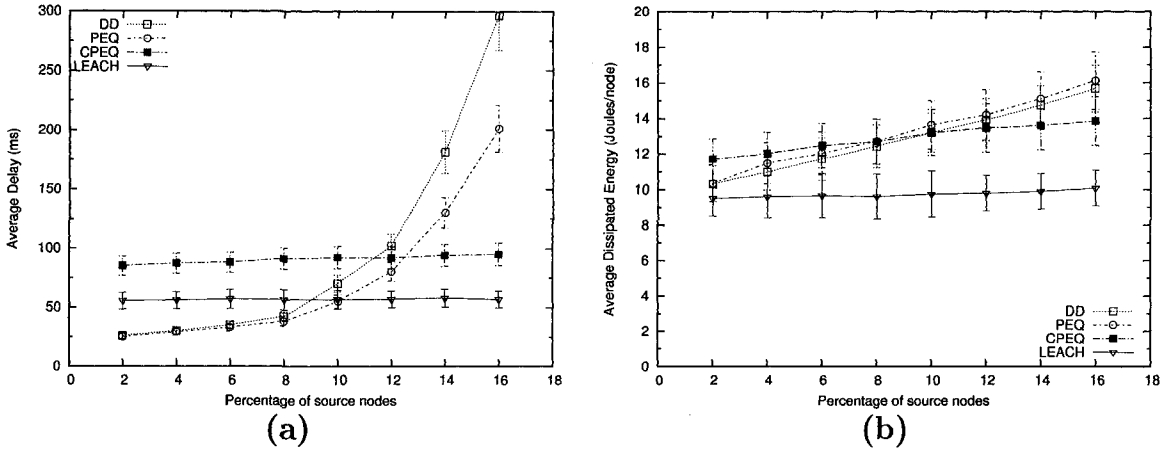


Figure 3.14: Average packet delay and energy consumption.

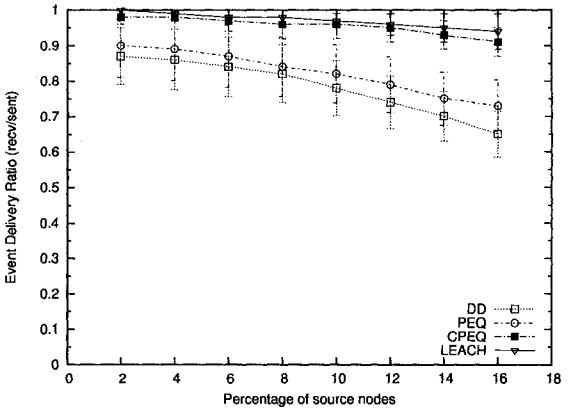


Figure 3.15: Delivery success rate as a function of source nodes.

and receptions. It results in less energy dissipation, as can be seen in Figure 3.14(b). It is worth noting that CPEQ, PEQ and DD do not save energy by turning nodes off, and the fact that idle radio modules spend as much energy as when they are receiving transmissions, the idle time energy utilization absolutely dominates all simulations. LEACH outperforms the other protocols since the cluster-heads set up a TDMA transmission schedule such that the sensor nodes can turn their radios off when they are not transmitting data.

The clustering and aggregation approaches of CPEQ also cause an impact on the packet delivery success. CPEQ behaves similar to the PEQ protocol in a low traffic

scenario. However, CPEQ shows better performance in high traffic scenarios since only the cluster-head nodes relay data to the sink, thereby reducing data traffic and packet collisions, as can be observed in Figure 3.15 that presents the results of data delivery ratio. LEACH shows similar performance to CPEQ.

In the second set of simulation experiments, the percentage of source nodes is fixed in 5%, and the network size ranges from 100 to 500 nodes in order to evaluate the scalability of the protocols under different network sizes. We have set the maximum radio transmission range of a node in LEACH to 150 meters and 20 meters in CPEQ, PEQ and DD. The sensor field area is also increased to keep a constant node density. All the other parameters are the same as in Table 3.2. Figure 3.16(a) shows that increasing the network size, PEQ and Directed Diffusion show higher latency than CPEQ. As mentioned earlier, CPEQ reduces the number of transmitted packets. Thus, it reduces traffic and queue buildups, which affects end-to-end delay and the other metrics. As expected, LEACH outperforms CPEQ by a small lead, but after a certain network size, some cluster-heads in LEACH do not reach the sink. The traffic reduction achieved by the cluster-based protocols has also an impact in the delivery success ratio, as can be seen in Figure 3.16(b), even with an increasing number of source nodes. LEACH clearly does not scale since it uses direct communications to the sink. LEACH is limited to small networks and depends on the radio range of the sensor nodes.

The average dissipated energy is shown in Figure 3.17. CPEQ, PEQ and DD show similar results, with a slightly better performance of CPEQ as the network scales up. LEACH achieves the best performance among the protocols studied. Packet losses are minimized in CPEQ due to its clustering and data aggregation features since not all data packets have to be relayed to the sink.

3.3.7 Final Remarks

Although the CPEQ offers a slightly better energy dissipation than PEQ, it still suffers from the high traffic load at the nodes surrounding the sink(s). One alternative would be

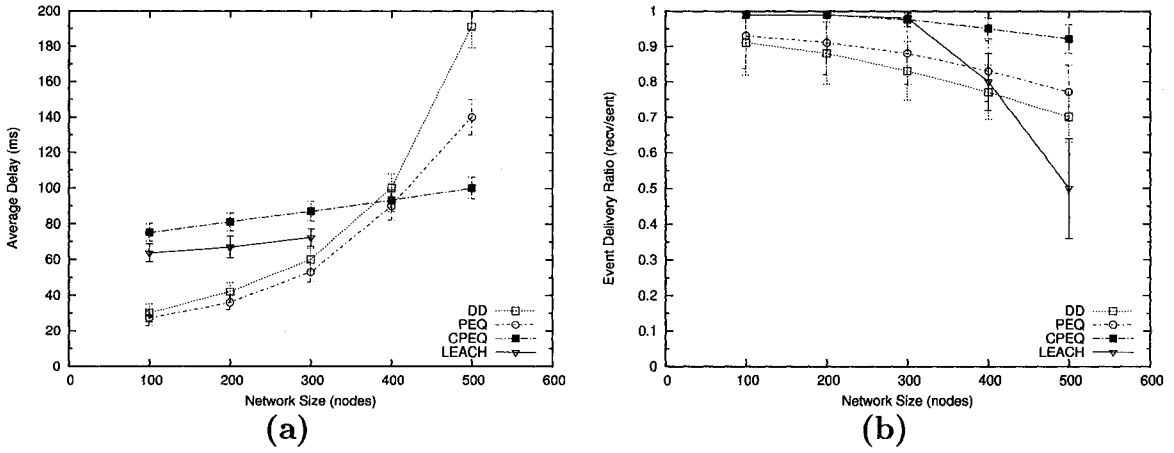


Figure 3.16: Average packet delay and delivery success ratio.

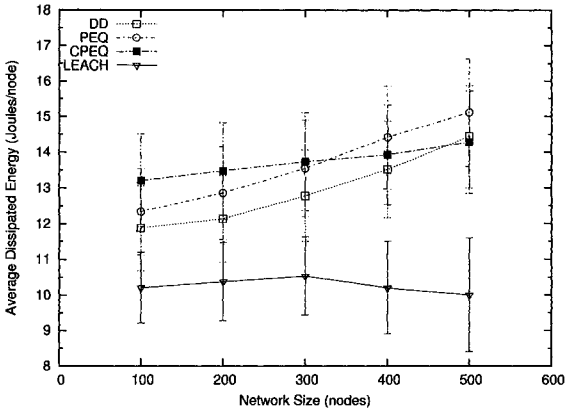


Figure 3.17: Energy consumption per node as a function of network size.

the deployment of several sinks. However, this solution would just postpone the problem, since the sensor nodes would still propagate their data to the sinks, and the nodes around it would suffer from high congestion. The LEACH approach solves the bottleneck at the sink's surroundings since it uses one-hop communications from cluster-heads to the sink. However, high power transmissions are necessary, thereby increasing the complexity of a group of sensor nodes, *i.e.*, the cluster-heads, and so the implementation costs.

An interesting approach is envisioned in which the sinks are mobile nodes. In such approach, the sinks move through the sensor network while gathering the data. There are several research work in the literature about data "mules". However, most of the

existing solutions are designed for non-delay-sensitive application scenarios. In addition, the existing handoff algorithms cannot be applied directly to wireless sensor networks due to the constraint power resources of sensor motes. Therefore, a mobile data gathering approach is proposed in chapter 4, in which a delay-sensitive data gathering solution is discussed and evaluated.

3.4 Summary

Sensor networks are increasingly being used for continuous sensing, event detection, location sensing, as well as micro-sensing in applications areas such as health care, transportation, finance, defense, government, manufacturing, emergency response, just to name a few. One of the most appealing applications is security surveillance and supervision of context aware physical environments that can be subjected to critical conditions such as fire, leaking of toxic gases and explosions. A challenging issue of these networks is how to provide a fast, reliable and fault tolerant routing protocol for query-based, event-driven and periodic sensor networks application scenarios. In this chapter, PEQ was proposed. It is a novel wireless network algorithm, that uses ordinary sensor node hardware with short radio range to meet periodic, event-driven and query-based interests. PEQ uses a small amount of information for the routing mechanism, *e.g.*, the hop level of each node. If a failure is detected, unlike other solutions that use flooding, a PEQ node employs the overhearing technique or it broadcasts a SEARCH message to its neighbors, and receives a reply with their hop level and identification. The neighbor with the lowest hop level is chosen as the new forwarding node. PEQ provides low latency for event notification, dynamic broken path reconfiguration, and high reliability in the delivery of events. Low latency is achieved by the use of the shortest path for the delivery of events. In addition, individual nodes, instead of a sink-based mechanism, trigger the recovery mechanism locally.

In order to decrease latency even further, specially under high network data traffic

conditions, a cluster-based version of PEQ was devised. CPEQ is a cluster-based routing protocol that groups sensor nodes to relay the gathered data to the sink efficiently. In the CPEQ protocol, nodes with more residual energy are selected as cluster-head nodes that relay the data to the sink by distributing energy dissipation among the nodes uniformly. Important metrics were evaluated and compared to the Directed Diffusion paradigm and the PEQ protocol. As the number of nodes increases, the simulation experiments indicate that CPEQ almost halves the packet delay compared to DD and PEQ. The delivery ratio is also improved in CPEQ. As the network size increases, CPEQ shows nearly 97% of delivery success in a network size of 500 nodes, versus 70% for DD and nearly 80% for PEQ in the same conditions. In terms of energy dissipation, CPEQ consumes less energy due to its data clustering feature, which reduces data traffic. The strength of CPEQ/PEQ is its simplicity and effectiveness in the delivery of events, thereby making it as a good candidate to meet constraints and requirements of event delivery in event-driven and query-based applications. LEACH outperforms CPEQ in some metrics. However, LEACH does not scale well and its deployment in large scale networks is unfeasible.

Chapter 4

Mobile Data Gathering Protocols for WSNs

4.1 Initial Considerations

Wireless sensor networks have been attracting the attention of researchers and the industry mainly due to the their potential applications [3][16][86] such as health care [105], soil monitoring for precision agriculture, border protection, marine and space exploration, battlefield and hostile environment surveillance [6], and a variety of training and monitoring applications for emergency preparedness and response.

Wireless Sensor Networks (WSNs) can be seen as a large collection of small wireless devices that can organize themselves in an ad hoc network capable of sensing environmental conditions within their range and have constrained energy, processing and communication resources. Sensor nodes usually transmit their data to a base station. However, a wireless sensor network usually lacks infrastructure and the sensor nodes must organize themselves in order to create routes that lead to a base station. Therefore, WSNs perform multi-hop data propagation in order to relay data to a static base station. Routing for WSNs is a largely studied field, and the reader can refer to [6][12][25][13] for interesting surveys and research work on routing protocols for WSNs.

In large scale WSNs, sensor nodes may use a cluster-based approach in which a cluster-head is responsible for collecting data from its cluster nodes. Hierarchical or cluster-based routing is a well-known approach that aims at providing scalability and energy efficiency in WSNs. A randomized selection of cluster-heads is usually used as a solution to the problem of energy depletion of cluster-head nodes [53]. In this technique, any sensor node can become a cluster-head, and the task of being a cluster-head is alternated periodically in order to provide load balancing. Other cluster-based routing protocols for WSNs include [77][78], among others. In a usual scenario, a WSN is a high density network deployed over a large area of interest. Therefore, several nodes are used in order to deliver a single packet to a remote sink. Furthermore, sensor nodes closer to the sink will drain their energy and use more resources than other nodes in the network, simply because they are in the path of many routes to a single data aggregation point. As a result, the sink's neighboring nodes will suffer from high congestion and packet losses that affect negatively the communication with the sink. Furthermore, these nodes will be the first to run out of energy and the network will quickly become disconnected and in-operational. Researchers are investigating the use of mobile entities that can collect information from the sensor network while moving within the monitored area. A mobile sink approach will not only remove the burden of the nodes closer to a sink, but it will provide a mechanism to reach and collect data from network areas that are disconnected, as well as to increase network lifetime. With the advances of wireless networks and the widespread of thin mobile handhelds such as cellular phones and Personal Digital Assistants (PDAs), a feasible strategy has attracted attention recently [39]. It deals with using handhelds as sensor data collectors, *e.g.*, mobile sinks or cluster-heads. Therefore, the mobile entity is able to collect data from the proximity nodes at very low cost, usually involving one-hop communications. It can decrease energy consumption and traffic load. In addition, the probability of queue buildups and collisions is reduced because the number of hops traversed by a packet in order to reach the sink is diminished significantly.

In this chapter, we propose a low-latency and reliable mobile data gathering solution

for delay-sensitive applications for WSNs. One of the challenges is to alleviate the high traffic load and resulting bottleneck in a sink's vicinity caused by static approaches. Our proposed MDC/PEQ protocol employs mobile data collectors (MDCs) that broadcast beacons periodically. Sensor nodes that receive the beacon will join the MDC's cluster and update their routing information in order to relay data packets to the corresponding MDC. Sensor nodes use the signal strength of the beacon in order to perform a simple but efficient route re-configuration (handoff). All message exchanges are kept locally within the nodes neighbors and the overhead is minimized. Our strategy contributes to reducing packet delivery delay and increasing reliability with little or no overhead by reducing the number of hops a data packet has to traverse. An extensive set of simulation experiments is conducted and results confirm that the introduction of mobile data collectors in wireless sensor networks reduces latency and alleviates the bottleneck at the nodes closer to the sink.

The remainder of this chapter is organized as follows. Section 4.2 presents a review of mobility support approaches for WSNs. Our proposed mobile data gathering is discussed in details in section 4.3. Simulation experiments, results and further discussions are presented in section 4.3.3.

4.2 Related Work

In this section, we investigate the related work on mobile data gathering for WSNs. Examples of research on this subject include [23, 63, 66, 68] among others.

In [66], a simple experimental evaluation is performed in which the authors consider only one mobile sink that moves through a straight line while collecting data from the sensor nodes, as can be seen in Figure 4.1. This approach reduces the number of hops a packet has to travel in order to reach the sink and it also saves energy and increases network lifetime.

The authors employ the Directed Diffusion [60, 61] as the routing protocol. The sink

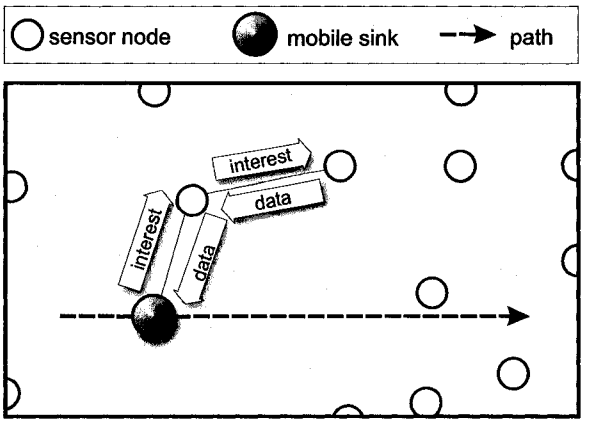


Figure 4.1: A mobile sink moving along a straight line.

broadcasts an interest message to the neighboring sensor nodes while moving through the straight line. The sensor nodes will receive the interest message and possibly forward to their own neighbors. Each sensor node will start transmitting data to the mobile sink when an event matches the sink's interest. The mobile sink may be out of range during the event transmission and packets will be lost. The solution uses acknowledgments to make sure the sink has received the packet successfully, and a sensor node will transmit other packets only after it has received an acknowledgment message from the sink. This work has been extended in [63] to support multiple mobile sinks (also called data mules in the literature). Both approaches focus in non-delay-sensitive application scenarios. Therefore, they show high latency in order to collect data and serve the application because sensor nodes have to wait to transmit data until a data mule is nearby.

In [23], the authors explore a predictable sink mobility in order to save energy in a WSN. They propose a simple sink-driven communication protocol in which the sink is placed on a public transportation vehicle, and they assume that the sink traverses the same path through a wide area sensor network, as depicted in Figure 4.2. Data is pulled by the sink by waking up the sensor nodes based on proximity.

Basically, the sink broadcasts a beacon message continuously while traveling through the network. Sensor nodes listen to the wireless channel periodically in order to check if there is any sink nearby. In a first cycle, the sensor nodes only observe how often

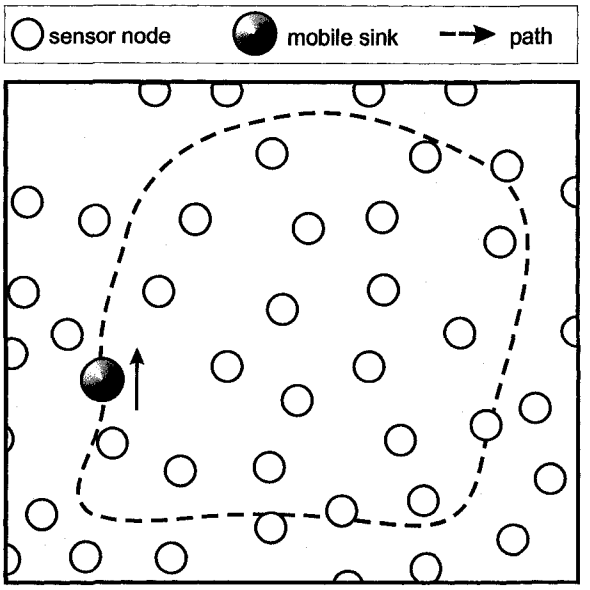


Figure 4.2: Diagram of the sensor network with a mobile sink.

the sink comes within range and for how long. In a second cycle, after the sink have broadcasted its beacon message, each node within range will start a collision-resolution based on 802.11 CSMA/CA, and it will transmit a packet to the sink that contains the position of the node and measurements collected during the first cycle. The sink will have a global picture of the network and it will use this piece of information to set up the communication priority when there are multiple nodes within its range waiting to transmit. Thereafter, the sink sends a wake up signal to the nodes it knows to be within range. Higher priority is given to the sensor nodes that will become out of range first. Furthermore, the sensor nodes predict when the sink is likely to be nearby and start listening to the wireless channel. This mechanism has a positive effect on energy savings because the nodes can be in sleep mode when the sink is not in range. The drawback of this approach is that its application is limited to the assumption that the sink traverses the network through the same path. Therefore, high delays are expected and this scheme cannot be employed in real-time applications such as emergency preparedness that requires fast response time.

The work presented in [28] aims at extending the Directed Diffusion protocol [60]

in order to support sink mobility. The technique performs a handoff scheme that follows a make-before-break strategy similar to the soft-handoff schemes used in cellular networks [50].

The approach presented in [33] consists of the design of a "*Robomote*", a robot platform (hardware and software) that functions as a single mobile node in a mobile sensor network for a number of different applications. For instance, it can be used to recover network connectivity by having a few mobile nodes moving to desired locations and repairing partitioned networks. In addition, it can be used to distribute traffic load and energy consumption by moving mobile sinks to different areas.

The authors in [76] consider the problem of positioning mobile cluster heads in order to perform load balancing in a hybrid wireless sensor network. It is a cluster-based approach in which each cluster head acquires the positions and connection patterns of all sensor nodes in the cluster. In addition, the cluster head tests the possible positions it can move to by estimating the lifetime of the nodes in the case the cluster head moves to that position. The cluster head chooses the position where the maximum lifetime was estimated and moves to that area. According to their simulation results, the authors report that their approach with mobile cluster heads can improve network lifetime by at least 30% after 5 position adjustments.

An interesting research work is discussed in [68], in which the authors propose protocols for mobile data gathering in WSNs. The first protocol is a simple passive data gathering algorithm, *i.e.*, "wait to hear from a nearby mobile sink and then transmit". This technique reduces the energy consumption significantly because only one-hop communications are performed in order to relay data to a mobile sink. However, according to the authors, such solution is not suitable for delay-sensitive applications because of the long periods a sensor node will probably wait until a mobile sink is within its transmission range. The second protocol considers a WSN with mobile and static sinks. Limited propagation trees are constructed and rooted at the sinks in order to collect data from the WSN. Each sink broadcasts a beacon and each sensor node that receives the beacon will

decide if it needs to update its parent nodes according to pre-define rules. Each sensor node uses a propagation tree to deliver data to the corresponding sink. When a sensor node ceases from hearing a beacon message it caches all data and waits for another beacon message. Otherwise, a sensor node will initiate its participation to a new tree after a timer expires. The authors proposed a third scheme that aims at coordinating among the sinks in order to provide a better distribution of the mobile sinks. Basically, each sink tries to detect the presence of other sinks, so that the sink will modify its trajectory to avoid getting closer to other sinks, thereby, contributing to a better distribution of sinks throughout the WSN.

4.3 Our Proposed Mobile Data Gathering Protocol

One of the main challenging issues investigated in this thesis is the design of a protocol for WSNs using multiple mobile data collectors that offers low latency event delivery. The key idea is to always have a valid route from a sensor node to a mobile or static sink, thereby avoiding that sensor nodes have to wait until a mobile sink is nearby. Hence, our scheme does not introduce extra delay, assuming that we have a connected network and the mobile data collectors are the "final" destination of the data. In addition, we focus mainly on delay-sensitive applications like real-time monitoring for emergency preparedness purposes in which time efficiency is preferred to energy savings.

4.3.1 Wireless Sensor Network Model

Consider the deployment of n wireless sensor nodes in a square area A. Nodes can be randomly scattered across region A or deployed as a grid. There is also a base station (sink) S that is responsible for collecting data from all sensor nodes and it acts as a gateway for the sensor network. We consider symmetric communication links. Consider the introduction of M mobile data collectors into the monitored area A, moving at maximum velocity v, and capable of communicating with wireless sensor nodes. A sensor

node utilizes its wireless radio resources to communicate with its one-hop neighbors, and the sensor nodes have no global knowledge of the network. Each node and the sink are assumed to be static and have transmission range R. The sensor nodes relay data to the sink in a multi-hop fashion. Some terms we use to designate the entities employed in this model:

- *Definition 1* (Sensor node): A sensor node is a static node in the network and acts as forwarding nodes.
- *Definition 2* (Source node): A source node is a sensor node that captures an event from the environment. This information must be relayed to the sink.
- *Definition 3* (Sink node): Also known as base station. A sink is a static node that collects data from the sensor nodes in the network. It is the data aggregation point.
- *Definition 4* (Mobile data collector - MDC): Also known as mobile sink. It acts as a mobile entity that collects data from the sensor nodes while wandering through the network field. Usually, an MDC has better power resources than ordinary sensor nodes.

4.3.2 Algorithm Overview

We extend the ideas of CPEQ/PEQ use it as the underlying routing protocol. As discussed earlier, the sensor nodes located near the sink will serve as forwarding nodes for a large number of nodes. This can result in high packet traffic and energy consumption. Most solutions for mobile data gathering in sensor networks use deterministic node mobility models in order to assist the process of predicting the future positions of a sink and, consequently, taking proactive routing actions. The mobility models are usually classified in two categories: entity and group mobility models. The entity mobility model describes the movement of an individual node only with no coordination with the movement of other nodes. On the other hand, the group mobility model coordinates the movement of

the nodes as a group. In our approach, the MDC nodes move randomly according to two different mobility patterns: the Random Walk Mobility Model; and a variation of the Boundless Simulation Area Mobility Model (BSMM), called Bounded Random Mobility Model (BRMM) [88]. In the Random Walk model, each node moves randomly and independently from each other and from previous movements. The BRMM is a group-based random mobility model and it presents smooth movement patterns. We believe that a group-based mobility model is more suitable than a completely random approach to represent the movement behavior of emergency response teams, soldiers in a battlefield, etc. We justify the use of this model by assuming that these entities, e.g., emergency responders, will carry a mobile device and act as data collectors or mobile sinks in our application scenario.

In our mobile approach, each MDC has the role of a cluster-head. It first starts the CPEQ cluster configuration so that all nodes joining the cluster will acquire the proper routing information about how to reach an MDC node. Due to the fact that we aim at delay-sensitive scenarios for emergency preparedness and hostile environment monitoring applications, we use a hybrid strategy with MDCs and a static sink. By this, the sensor nodes do not need to wait until an MDC is nearby in order to start relaying their sensed data. Each node keeps a hop level h_m, in addition to h_s, that represents the number of hops to reach the closest MDC. A node will decide which data collector (sink or MDC) it should forward a DATA message to according to both parameters h_m and h_s. An overview of the MDC/CPEQ protocol can be seen in Figures 4.3 and 4.4The algorithms are discussed in the next subsections.

Cluster Configuration

An MDC starts a cluster configuration phase by sending a BEACON message. Similarly to the initial setup phase, a hop level h_m is assigned to each sensor node in the cluster, which can be seen as a tree rooted at the MDC. The node that receives a beacon will first check its internal hop level h_m and the hop level h of the received message. Thereafter,

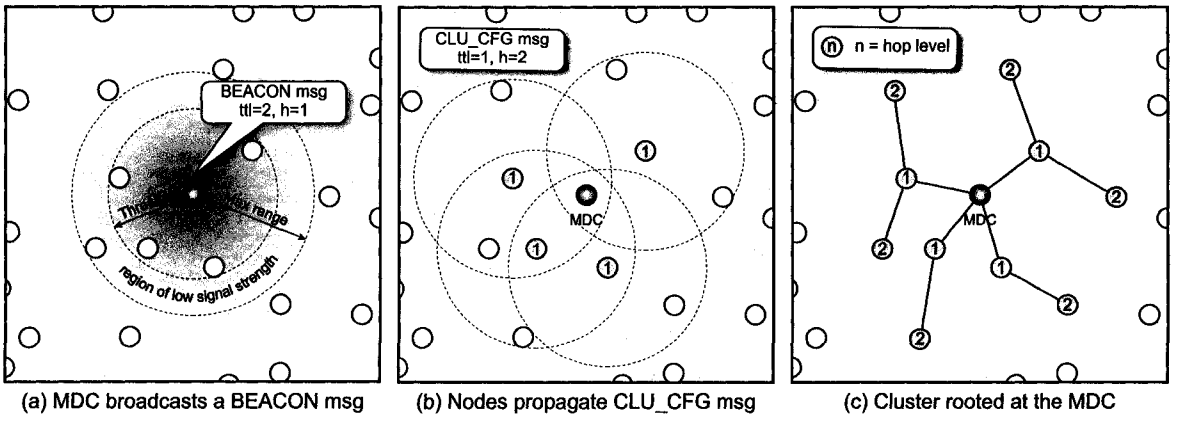


Figure 4.3: Overview of the MDC/CPEQ cluster configuration algorithm.

it forwards the message when appropriate. It is exactly the same algorithm as the initial network setup or the CPEQ's cluster configuration discussed in section 3.13. However, when a sensor node, not a SINK or MDC, receives a BEACON message, it first checks the MDC's signal strength SS before performing any routing updates. An example of the cluster configuration step is depicted in Figure 4.3.

Route Maintenance and Handoff

Each mobile data collector broadcasts a BEACON message periodically at every T_b seconds. The BEACON packet contains the MDC ID, a time to live TTL, and a hop level h. Figure 4.5 depicts the flow chart for a node that receives a BEACON message. The entire process of routing updates is described in Figure 4.4.

As shown in Figure 4.3(a), we specify a reception threshold Rx_{thresh} based on the communication range R of the node. Based on the signal strength, the node will react according to the following rules:

(1) if $(SS \geq Rx_{thresh})$: the node is receiving strong signal from the MDC. After that, the node detects if the sender of the BEACON is its current MDC (based on MDC ID and MDC ID_{curr}):

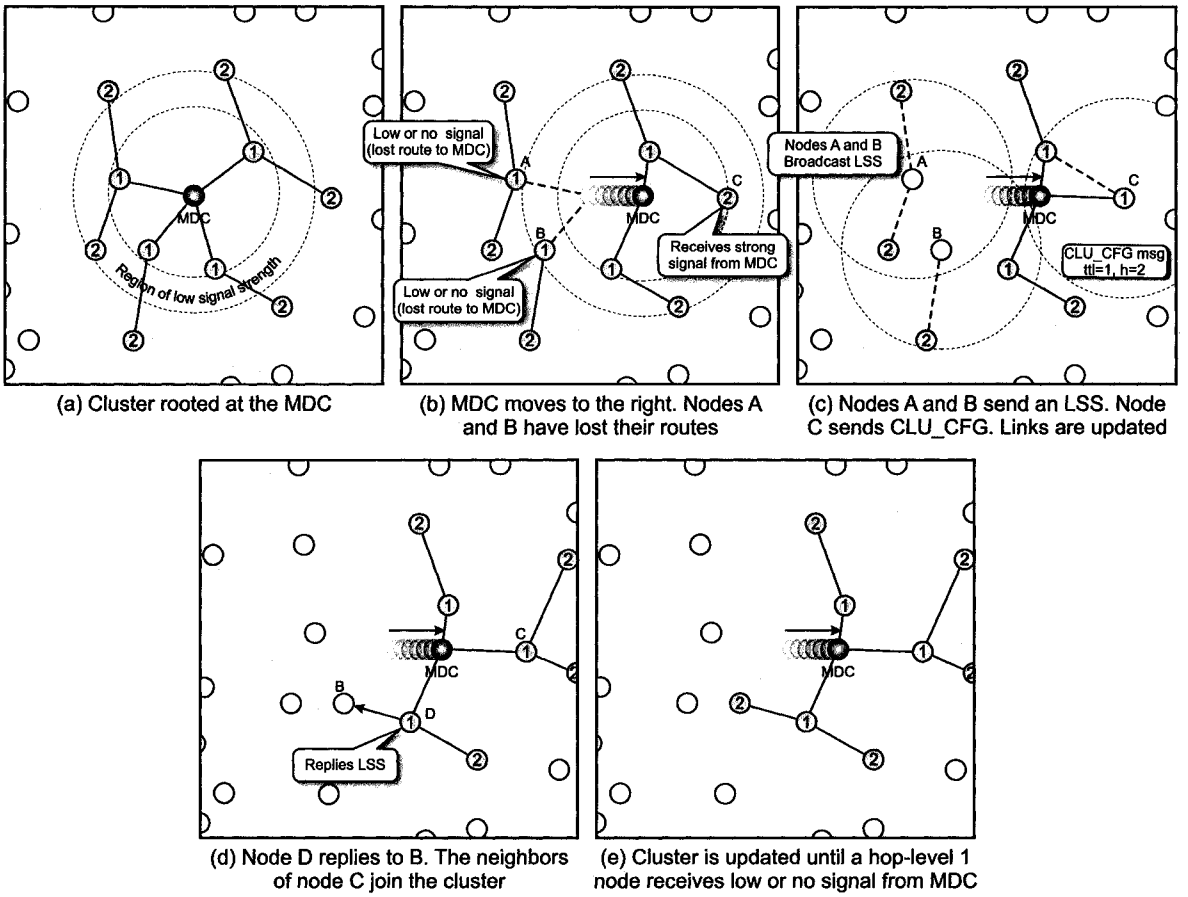


Figure 4.4: An example of the MDC/CPEQ protocol in action.

(1.a) (MDC ID == MDC ID_{curr}): the BEACON was sent from the node's current MDC, thereby the node does not need to perform any routing changes. It will only update ($SS_{curr} = SS$), in which SS_{curr} represents the most up-to-date signal strength received from its current MDC, and SS is the signal strength measured from the received BEACON message. The SS_{curr} value is kept by the node and it will be used when replying to nodes that might request a route to an MDC;

(1.b) (MDC $ID \neq$ MDC ID_{curr}): if the beacon was sent from a different MDC, the node silently drops the packet because it is already in a cluster. In case the

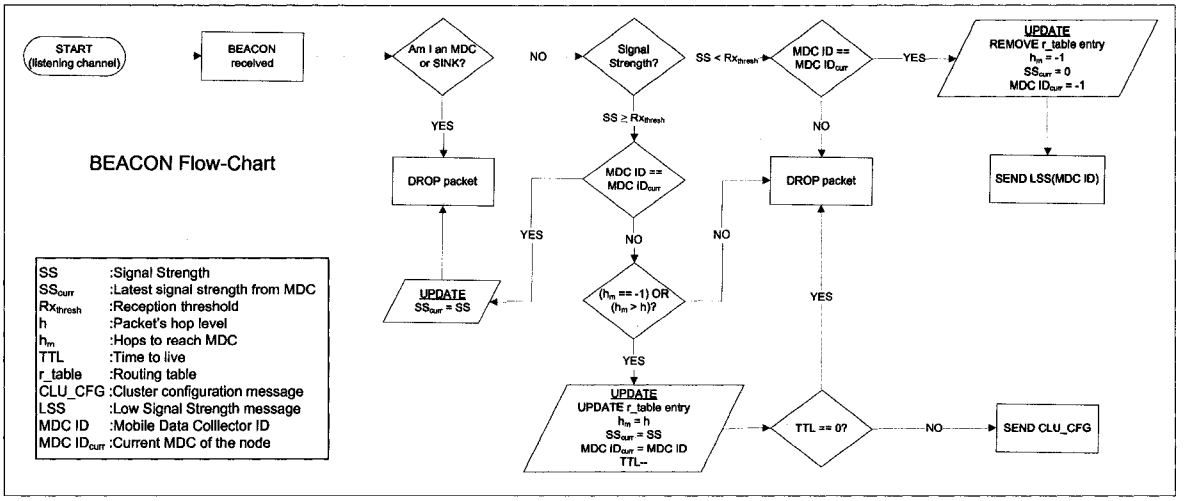


Figure 4.5: Flow chart of the action taken after a BEACON is received

node does not belong to any cluster, *i.e.*, by checking if (h_m == -1)), or in the case the node already belongs to a cluster but ($h_m > h$), which means that there is a shorter path to an MDC, the node joins the MDC's cluster by updating its routing table and state variables as describe in Figure 4.5. Thereafter, the node broadcasts a CLU_CFG message to its neighbors if TTL has not reached 0. Figure 4.4(b) and 4.4(c) show an example of this case. Note that node C has a route to the MDC, but it receives a beacon from the MDC. Node C will update its routing information.

(2) if ($SS < Rx_{thresh}$): the node is within the communication range of an MDC, but the signal strength is below the allowed threshold. This can have two meanings: the MDC is moving away from the node; or the MDC has just entered the node's communication range. Therefore, the node must find out the context of this BEACON message:

(2.a) (MDC ID == MDC ID_{curr}): the originator of the beacon was the node's cur-

rent MDC, but the node is receiving low signal strength. Therefore, the node must update its routing information and start looking for another MDC. The node removes the routing entry for that MDC and updates its parameters. Usually, if the MDC is moving out of the node's range, the node will probably find out that a neighbor has a route to that same MDC. The process of finding another route to an MDC is initiated by the node broadcasting a Low Signal Strength message LSS. An LSS message also informs other nodes that its originator node does not have a route to that specific MDC anymore. The LSS message is sent only after there is a change in the node's routing information. For an illustration of this process, check Figure 4.4(b) and 4.4(c), in which nodes A and B receives a beacon with low signal strength. Thus, nodes A and B broadcast an LSS message to search for another MDC and also to inform their neighbors that they have lost their routes to the MDC;

(2.b) (MDC $ID \neq$ MDC ID_{curr}): a beacon with low signal strength was sent from a different MDC than the node's current one, thereby no routing maintenance is necessary. The node silently drops the packet.

After a sensor node receives a BEACON message, it might send a CLU_CFG message (see rule (1.b)). A sensor node that receives a CLU_CFG message decides whether to join the cluster, as depicted in Figure 4.6. For that, the node will check if it already belongs to a cluster or if it only needs to update its routing information. The sensor node is able to determine what to do next by checking its hop level h_m according to the following rules:

(3) if (h_m == -1): the node does not have any MDC in its routing table, thereby it joins the cluster and creates an entry by setting up the address of the CLU_CFG message originator as the destination to the MDC in its routing table. This case is depicted in Figure 4.4(c) and 4.4(d). Node C broadcasts a CLU_CFG message

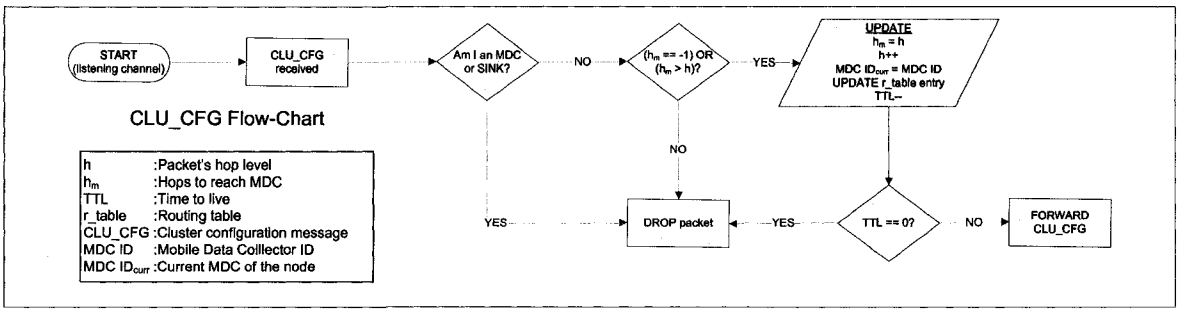


Figure 4.6: Flow chart of the action taken after a CLU_CFG message is received

and its neighbors join the cluster.;

(4) if $(h_m > h)$: the node has a route to an MDC, but there is a shorter path to reach the same or another MDC. The sensor node updates its routing information and its parameters. Thereafter, the node forwards CLU_CFG if TTL has not reached 0;

(5) Otherwise: the node has already the shortest path to an MDC. It silently drops the packet.

When a BEACON is received and the measured signal strength (SS) is below the reception threshold Rx_{thresh}, the sensor node might broadcast a *Low Signal Strength* (LSS) notification (see rule (2.a)). When a sensor node receives a BEACON message, it sets a timeout timer T_{beacon} for receiving the next BEACON. If this timeout expires, the sensor node need to find another route by sending an LSS message. Upon receiving an LSS message, the node verifies if the source node is in its routing table as a destination to reach an MDC, so as to decide if it should update its routing information, as shown in Figure 4.7, according to the following rules:

(6) if $(Next\_hop == LSS\_src)$: the node checks its routing table and realizes that the LSS source is a route to the MDC, i.e., the route the node was using to reach

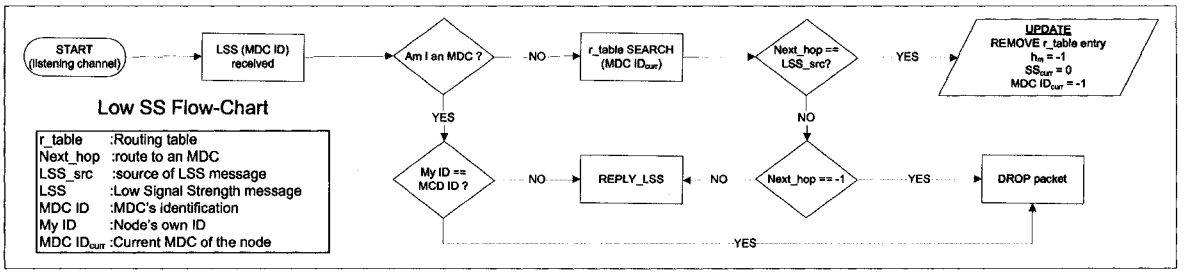


Figure 4.7: Flow chart of the action taken after an LSS message is received

an MDC in no longer valid. In this case, the node must update its routing table by removing the entry and resetting its parameters according to Figure 4.7. For instance, the nodes A and B broadcasted an LSS message in Figure 4.4(c), and the nodes that were using A and B to forward data to the MDC must update their routing information.

(7) if (*Next_hop == -1*): the node does not have an entry in its routing table that leads to an MDC. Therefore, it silently drops the packet;

(8) Otherwise: the node does have an entry in its routing table that leads to an MDC. In this case, the node replies with an LSS_REPLY message to the LSS source node. When the node receives an LSS_REPLY, it will retrieve and add the source address as a route to a new MDC in its routing table. This situation is depicted in Figure 4.4(d) and 4.4(e), in which node D replies to node C and a new link is built.

An LSS packet carries the *ID* of the MDC that originated the BEACON, thereby if the same MDC receives an LSS it will not reply in order to avoid message loops.

Data Transmission

As discussed earlier, we use a hybrid approach in which data can be relayed to a mobile data collector as well as to a static sink. We focus in delay-sensitive applications that require fast response time and reliability. If only MDCs were used, a sensor node would have to buffer its sensed data until an MDC was nearby, thereby introducing high packet delivery delay. Another solution would be the creation and maintenance of routes to an MDC, even if it goes far away from the source node. This unscalable approach would generate high message overhead because several nodes would have to update their routing information frequently. In our solution, if a sensor node turns to be out of the "cluster's range", i.e., if its hop level $h_m > TTL$ after an MDCs move, it must remove the route to the MDC from its routing table. This way, our approach does not leave a "*bread crumb trail*" while an MDC moves along its path. Hence, future route changes are not disseminated through a large number of nodes, but kept locally to a cluster.

Each sensor node holds two hop levels: h_m, that represents the hop level regarding an MDC; and h_s, the hop level with respect to the static sink. When a node has a DATA packet ready to be transmitted, either received from another node or generated by the node itself, it decides what route should be used. This decision is based on the hop level. In the current version of the MDC/PEQ protocol, a sensor node chooses the route with the smallest hop level, or it chooses the static sink as its destination if $h_m = h_s$. For instance, suppose that the sensor node A has DATA to transmit and does not belong to any cluster. The node A checks its hop levels and finds out that it does not have a hop level entry to an MDC. Hence, the node uses the route to the sink in order to relay its DATA. If an intermediate sensor node has a route to an MDC, it will forward DATA through the route leading to its MDC. By this, DATA coming from distant nodes that do not belong to any cluster might be intercepted by a node that belongs to a cluster in its way to the sink. The intercepted data will be forwarded to the nearest MDC. This approach contributes to the main objectives of the protocol: to reduce delay and relief the burden of the nodes closer to the sink. In addition, after receiving DATA packets,

Table 4.1: Simulation parameters.

Parameter	Value
Number of nodes (N)	100
Simulation area (A)	150x150m
Wireless radio range (R)	25m
Number of source nodes	50
Source node data rate	2pkts/s
Beacon expiration timer (T_{beacon})	700ms
Number of MDCs	5
MDCs velocity	5m/s
BEACON rate	2 beacon/s
Time-to-live (TTL)	2
Max network delay (jitter)	50ms
Bandwidth	34.8Kbits
Packet size	32Bytes
DATA packet size	64Bytes
TX power dissipation	0.01488W
RX power dissipation	0.01250W
Idle power dissipation	0.0W
Reception threshold (Rx_{thresh})	2.53789e-08W
ns-2 RXThresh_ (for R=25m)	1.62425e-08W

mobile data collectors may collaborate among them, in an ad-hoc fashion, in order to relay data to another entity that can be the sink itself. For instance, in our application scenario, MDCs are carried by emergency responders who visualize the sensed data in an augmented reality system. Therefore, data is immediately processed at the MDCs, and do not need to be relayed to a higher tier. However, some information must be relayed such as position of emergency responders, thereby members of the team can be aware of teammate locations. We plan to study the impact of adding another tier in our protocol, such as forwarding data from MDCs to the sink using another network besides the WSN.

4.3.3 Performance Evaluation

Simulation Scenario and Metrics

In order to evaluate and validate the performance of the proposed solution for mobile data gathering in wireless sensor networks, we have implemented our MDC/PEQ protocol and carried out an extensive set of NS-2 [87] simulations. We have scattered 100 sensor nodes across an area of 150x150m. We consider a dense and connected network, i.e., all sensor nodes are able to deliver data to a static sink. Future work will include the evaluation of our schemes under a partitioned network and topologies with different

node densities. The sink node is positioned at the border of the simulation area. All nodes have radio range of R=25m. We have set the reception threshold Rx_{thresh} in order to achieve a maximum reception distance of 20m without any obstruction. The power dissipation of idle periods was set to zero to evaluate the energy consumption from the different approaches because idle time dissipation dominates the simulation and it would be difficult to compare the results. The number of data source nodes is set to 50, and they are chosen randomly from the sensor field. Each source node generates 2 packets per second during the entire simulation. The packet size is 32 bytes for all types of messages except DATA packets, which are set to 64 bytes. When not specified in the graph, the number of MDCs is set to 5 and TTL=2. In our future research, we will investigate the impact of different values for the TTL parameter and the percentage of MDCs. Each MDC sends 2 BEACON packets per second and moves at a speed of 5m/s (when not specified). The sensor node parameters are reset in the beginning of the simulation such as h_m = -1, h = -1, MDC ID_{curr} = -1, and the nodes start the simulation with empty routing tables. A summary of our simulation parameters is shown in Table 4.3.3. The performance evaluation of MDC/CPEQ considers different aspects, but focusing packet delay, energy consumption, and traffic load near the sink. The metrics used were:

- Average packet delivery delay: the end-to-end packet delay measured from the source node to the destination (sink or MDC);
- Average packet delivery ratio: the ratio of packets received successfully at the sink or MDCs;
- Average energy consumption: the energy dissipated per node;
- Average transmissions per node: to evaluate the transmission overhead of our protocol;
- Number of hops traversed: it indicates the percentage of DATA packets that traversed a specific number of hops.

Table 4.2: Summary of simulation results.

Mobility Model	Delay (ms)	Ratio	Energy (mW)	Route changes	Msg Overhead	TX per node
Random Walk	119.54	0.97	5.22	1.07	1.81	3.69
BRMM	128.54	0.93	6.72	2.49	2.72	3.91
No mob. PEQ only	220.70	0.86	7.06	—	—	6.20

Simulation Results

The first part our experiments shows the performance of our scheme over the simulation time. The results were extracted for a specific time period over the number of generated packets during the same period. For instance, we measure the metrics every 5 seconds and compute the mean over the packets generated during these 5 seconds. We believe that evaluating the performance over time can demonstrate the dynamics of our proposed protocol and detect drastic fluctuations in the observed metric effectively. We focus in real-time monitoring applications and such fluctuations in delay, for instance, can compromise the validity of an event if it is delivered too late. A simple mean over the entire simulation time would hide such fluctuations. A summary of the experimental results is shown in Table 2. 100 sensor nodes, 5 MDCs, an 1 sink are used. The MDCs move at 5m/s. As can be seen in Figure 4.8, the introduction of MDCs almost halved packet delivery latency. The reason can be clarified in Figure 4.9(a) that shows that the number of packets that traverses only a few hops is significantly increased when compared to the PEQ protocol. In PEQ, the distant source nodes have to relay data through several hops, thereby contributing to an increase of packet collisions and losses. The percentage of packets delivered to the MDCs and to the sink is depicted in Figure 4.9(b), in which MDCs are responsible for almost 80% of the received packets and the sink receives only 12-15% of the transmitted packets (considering packet losses), thereby confirming our expectations about reducing the problem of static data aggregation points. There is little difference in MDC/PEQ's performance between the two motion patterns considered here. When simulated with the BRMM model, the MDCs move out of the sensor nodes' range more frequently than with the Random Walk model. This induces more route

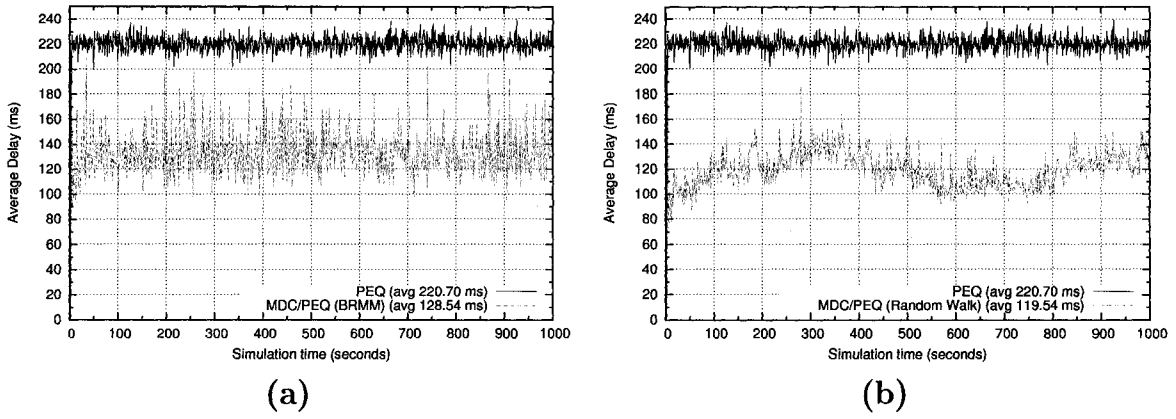


Figure 4.8: Average delay as a function of time

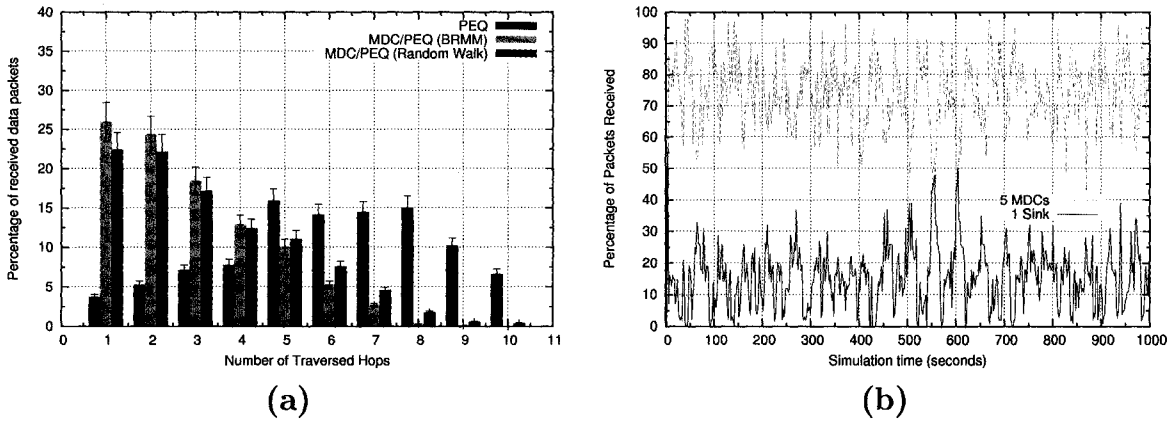


Figure 4.9: (a) Hops traversed. (b) Percentage of packets received

changes and message overhead. In the Random Walk model, we observed that MDCs tend to perform several movement changes but without going too far away from their initial position during a certain time period.

The performance of the MDC/PEQ protocol with respect to packet delivery success is depicted in Figure 4.10. The MDC/PEQ approach performed better than the static solution (PEQ) mainly because it reduces the number of hops traversed by data packets. This implies less traffic and, consequently, it reduces congestion and contention periods. In the Random Walk mobility model, the MDC/PEQ protocol showed even better results due to the fact that, in average, the MDCs stay longer within a node's range, thereby

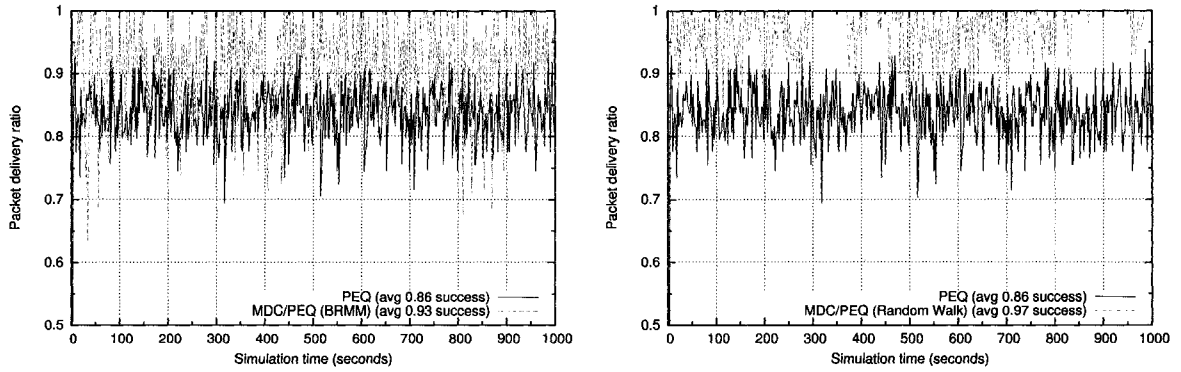


Figure 4.10: Average packet delivery ratio as a function of time

requiring less route changes. The reader can observe in Figure 4.10 that the delivery ratio performance of the MDC/PEQ approach has contrasting results. As can be seen, performance sometimes dropped below 70% of success ratio. This is mainly due to the nodes reacting to the BEACON messages in order to update their routes. High contention periods and collisions are observed when nodes have DATA, CLU_CFG, and LSS messages to transmit at the same time and in the same area. However, the overall performance of packet delivery ratio is improved from 86% to above 90%.

The energy consumption is one of the crucial concerns regarding wireless sensor networks because some application scenarios make it impossible or unfeasible to replace sensor node's batteries periodically. Figure 4.11 shows the power consumption per node over time. It can be seen that MDC/PEQ introduces little or no overhead. When the MDCs send BEACON messages and move thought the sensor network area, the nodes will generate a significant number of configuration messages in order to update their routing tables. However, the reduction in the number of hops a data packet must traverse in order to reach its destination (sink or MDC) brought by the MDC/PEQ protocol helps decreasing energy consumption drastically. At the end we have a slightly better performance in energy savings.

In this second set of simulation experiments, we aim at evaluating the scalability of

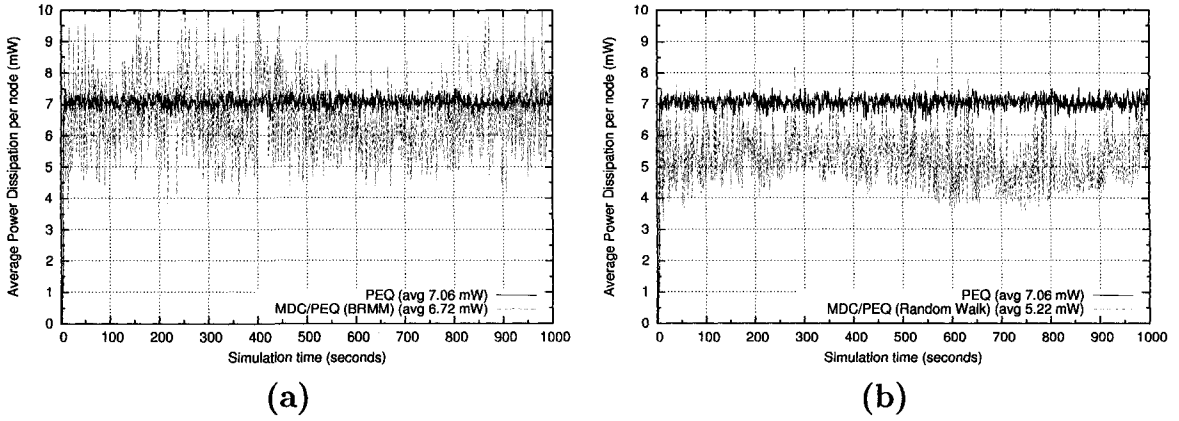


Figure 4.11: Average energy dissipation per node as a function of time

MDC/PEQ considering the same simulation parameters discussed earlier, but varying the number of MDCs and their speed. Hence, the MDCs send BEACON packets at a rate of 2 per second. 50 sensor nodes are randomly selected to produce data at a rate of 2 packets per second, and the BRMM mobility model was used. As can be seen in Figure 4.12(a), the introduction of only a single MDC in the sensor field has helped reducing the average packet delivery delay. When more MDCs are deployed, lower delays are achieved. This is due basically to the impact of reducing the number of hops a data packet traverses to reach the destination. In addition, it also increases the probability of a data packet being intercepted by a cluster and not forwarded to the sink. This also helps alleviating the problem of high traffic load in the sink's neighborhood. The increase in the speed of the MDCs has caused a noticeable effect on the packet delay. A packet that was in route to an MDC is redirected to the sink if the node has lost its routing entry to an MDC. When it is moving faster, an MDC induces more route changes. Therefore, more data packets are redirected to the sink. But it does not mean they will reach the sink, because another cluster might intercept the packet on its way.

The increase in the number of MDCs shows a positive impact on the delivery ratio, as depicted in Figure 4.12(b). However, the delivery ratio is affected by the speed of MDCs. For instance, when the MDCs move at 10m/s, the delivery ratio drops to below 80%.

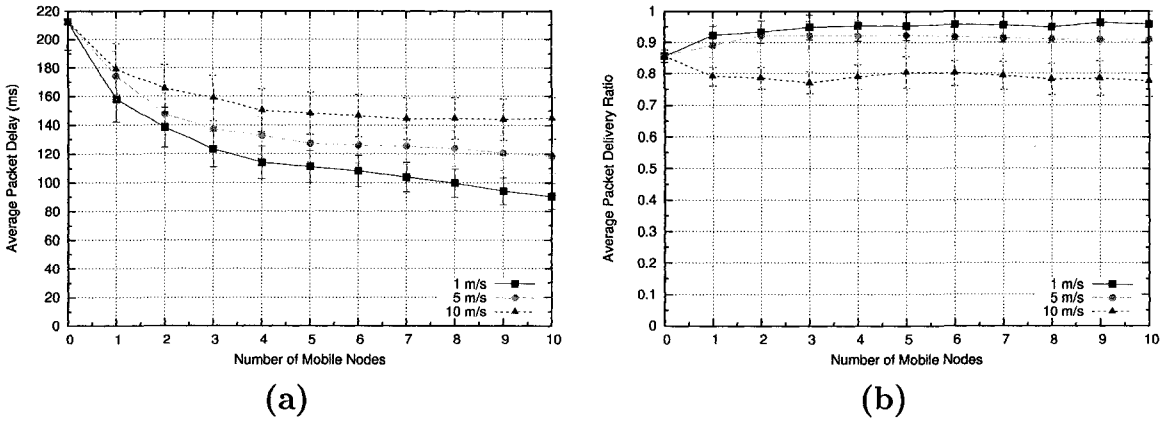


Figure 4.12: (a) Average packet delay; (b) Average packet delivery ration

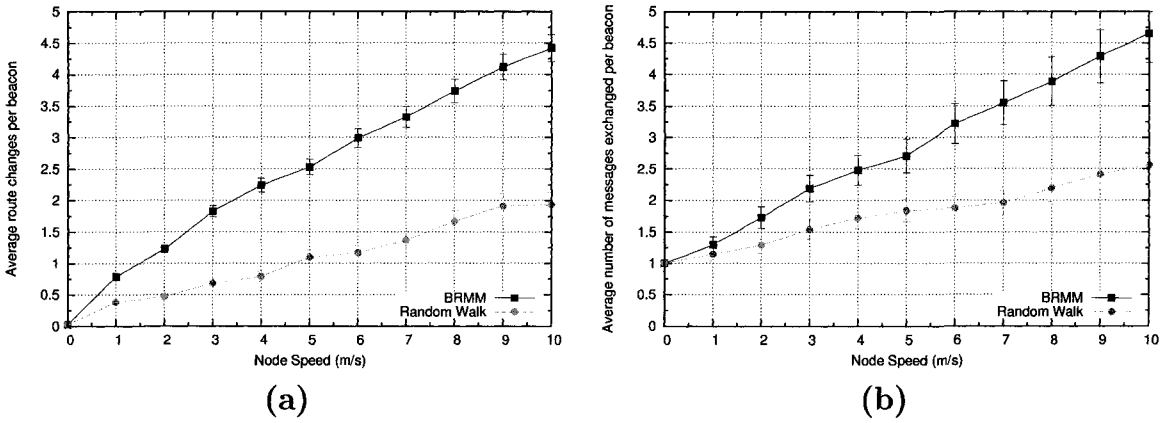


Figure 4.13: Average route change overhead per BEACON.

This is due to the fact that route updates happen more frequently. In addition, packet losses between route changes are more frequent when nodes move faster. Basically, a node sets a beacon timeout timer after it receives a BEACON message. If the MDC moved out of range and the node could not hear the BEACON message, the node will wait until the timer expires. We have noticed some packet losses during this period. A possible solution would be increasing the MDC's BEACON rate according to its speed.

Another important metric that assists us clarifying the performance of the MDC/PEQ protocol is the number of route changes per BEACON, as shown in Figure 4.13. We have fixed the number of MDCs to 5 with varying speeds from 1 to 10m/s. Figure 4.13(a)

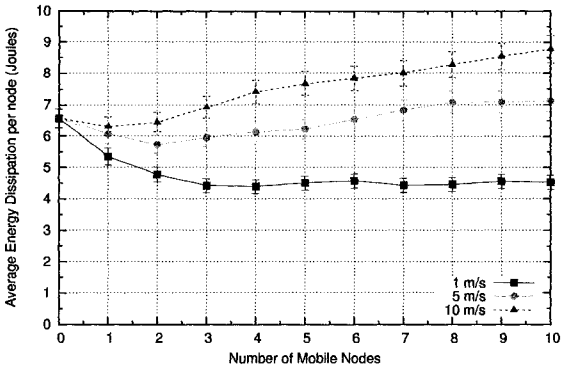


Figure 4.14: Energy dissipation per node as a function of number of MDCs

confirms that increasing the speed will induce more route changes per BEACON because an MDC visits more nodes when moving faster, and the nodes will update their routing information more frequently. Similar results can be seen in Figure 4.13(b), but this time with respect to message overhead. The same applies here: as we increase the speed, the visiting period of the MDCs will be shorter, thereby inducing more messages in order to reconfigure the routes. Finally, Figure 4.14 confirms that higher speeds induce more route changes that affect negatively the energy consumption because more messages are exchanged in order to find alternate routes and to update routing tables.

4.4 Summary

In recent years, wireless sensor networks have been showing their important role in a variety of applications that require fast response such as battlefield and hostile environment surveillance, emergency preparedness, etc. However, most solutions are focused in static sensor nodes and sinks. It has already been shown in the literature that static sinks perform poorly in terms of network lifetime and delivery delay. Furthermore, most techniques that employ mobile sinks deal with the problem of collecting data in non-delay-sensitive scenarios. In this chapter, we have considered the problem of deploying mobile data collectors in order to alleviate the high traffic load and resulting bottle-

neck in a sink's vicinity caused by static approaches. We have discussed our proposed MDC/PEQ protocol and demonstrated its contributions to reducing packet delivery delay and increasing reliability with little or no overhead. Our mobile data collector strategy is responsible for reducing the number of hops that a data packet must traverse. This has a significant impact on all metrics evaluated. The experimental results confirm that the introduction of mobile data collectors in wireless sensor networks reduces the bottleneck at the nodes closer to the sink and almost halves packet delivery delay. We believe that emergency preparedness applications or other delay-sensitive systems can benefit from the contributions of the MDC/PEQ scheme.

Chapter 5

Interactive Multimedia Streaming over Wireless Networks

5.1 Introduction

Audio and video communication over the Internet has boosted a number of interesting applications ranging from entertainment to robust video conferences and video surveillance over the Internet. The provision of such services over wired networks poses significant challenges. These challenges are intensified when wireless networks are considered due to the volatile characteristics of wireless channels [103]. Wireless networks show limited and time varying bandwidth, usually suffer from higher delays due to the multi-hop communication pattern and handoff mechanisms, high burst error rates and interference that cause packet losses.

Multimedia can be classified into two categories: non-interactive and interactive. Non-interactive multimedia is usually linear, *i.e.*, the playback of this type of multimedia follows a predefined sequence of frames, and continuous. Examples of non-interactive multimedia includes video presentations, music, etc. Interactive multimedia, on the other hand, allows the user to control the progress of media or the navigation. There are two types of interactive multimedia: one that allows the user to control the playback,

i.e., VCR functions, on linear and continuous media; and the other one in which the user can change the view or navigate within a non-linear media in real-time, *e.g.*, 3D virtual environments, virtual reality, augmented reality, etc.

This chapter is focused in presenting the design, analysis, and simulation experiments of a non-linear interactive multimedia streaming strategy for wireless multimedia and sensor networks. To this end, an a streaming protocol is proposed as well as a buffering mechanism to support the requirements of bandwidth-demanding multimedia systems. A rate control scheme that uses an existing loss differentiation algorithm is devised. The main goal of the rate control is to achieve high end-to-end throughput and low frame rate fluctuation in order to adapt the data rate to the frequent changes of the bandwidth and error rate present in wireless networks. Lastly, the performance and feasibility of the protocols are evaluated through a round of simulation experiments.

5.2 Background Information

5.2.1 Quality of Service Issues for Multimedia Streaming

The past years have seen a growing interest in continuous media streaming research. Voice over IP (VoIP) and video streaming have been dominating research efforts in this field. Most of the work found in the literature focus on providing guaranteed network delay bounds for multimedia traffic. The objective of video streaming research is to come up with reliable delivery of high-quality video over the Internet and wireless networks while guaranteeing network delay bounds when dealing with unknown and dynamic loss rate, available bandwidth, and delay jitter.

Recovering Packet Losses - Error Control

Packet losses in traditional wired networks are affected mainly by congestion. On the other hand, wireless channels are typically affected by bit errors due to interferences.

The streaming of compressed video is highly sensitive to packet losses in the sense that a single packet loss can affect the video decoding negatively. The challenge lies in how to protect the video from packet losses. There are several error control for image and video frame transmission in the literature. There are basically three forms of error control: forward error correction (FEC), retransmission, and error-resilient video coding.

- *Forward Error Correction - FEC*: The key idea is to add redundant data that can be used to recover from errors. For example, in order to protect the transmission of a JPEG2000 image, the server can duplicate the parts of the image, *e.g.*, the headers, that can affect the quality of the decoding at the client side. The drawback of the FEC approach is the overhead introduced, since redundant data will be transmitted even when there are no losses.

- *Retransmission*: In this category, the receiver notifies the sender about which packets were lost, and this the sender retransmit the lost packets. No extra overhead is introduced, but the drawback is additional delay, thereby limiting its use to non-delay-sensitive applications. Audio and video conference, for instance, requires low latency in order to provide a quality service.

- *Error-resilient video encoding*: This approach consists mainly in recovering from loss of bitstream synchronization to avoid error spreading. Markers are added to the code-stream to allow the decoder to continue decoding the video when packet are lost.

Dynamic Bandwidth Availability

Another important issue to be solved is the problem of available bandwidth. Video streaming requires high-bandwidth. Thus, the video streaming solutions should maximize the bandwidth utilization, otherwise the quality of the stream will be affected. Therefore, there has been several research efforts to solve the problem of rate control, *e.g.*, RAP [96], VTP [122], TFRC [45, 46], among others [34, 72, 97]. For instance, the

solution presented in [1] discusses a streaming media congestion protocol that uses a bandwidth estimation scheme to adapt the rate properly.

Delay and Jitter

Multimedia streaming are also affected by latency fluctuations (jitter). The receiver has meet certain video frame decoding deadlines. For instance, late video frames can produce interruptions or flickering in video playback. Video data is continuous and the server must stream a certain number of video frames in sequence to the remote client, which will in turn play back these frames in sequence. This characteristic makes it possible to use buffers to avoid jitter during playback. The use of video buffers allows the client to absorb more jitter in order to provide a smooth playback experience. However, this solution introduces an initial delay, *e.g.*, filling the buffer, before starting the video playback.

5.2.2 Rate Control over Wireless Networks

The Transport Control Protocol (TCP) [99] is the most employed transport protocol in the Internet. TCP has been developed for reliable data transfer over wired environments. However, it is not suitable for real-time multimedia traffic because of its retransmission and congestion control mechanisms. The rate control of TCP is based on the Additive Increase Multiplicative Decrease (AIMD) algorithm [62], in which the flows adjust their rate to a value that is calculated from the current rate plus a fixed amount, whenever a feedback mechanism indicates network congestion. In addition, TCP performs a fast recovery from congestion situations by halving the transmission rate. This halving characteristic affects the TCP flow's throughput stability, delivering unstable traffic, which is unsuitable for continuous media streaming. Furthermore, the retransmission mechanism employed by TCP introduces unacceptable delay and jitter, thereby affecting real-time streaming flows negatively.

The User Datagram Protocol (UDP) is mainly used to support multimedia streaming

flows. UDP does not use any rate control mechanism. Thus, the rate control must be implemented at higher layers of the protocol stack. The Real-time Transport Protocol (RTP) [98] and its control protocol (RTCP) provide a framework to end-to-end network transport services suitable for continuous multimedia applications, such as VoIP and video streaming. RTP does not guarantee any QoS requirement and assume that the network is unreliable and packets may be delayed. RTCP monitors and provides for minimal feedback about the quality of service about the participating sessions. However, RTCP is based on periodic transmission of control packets to all parties involved in the communication. In wireless networks, such periodic feedback may not perform efficiently because wireless channel services are highly dynamic regarding bandwidth, packet losses, and delay. Therefore, RTCP may deliver delayed feedback and the application will be unable to adapt the rate on time.

One of the most popular rate control schemes over wired networks is the TCP Friendly Rate Control (TFRC) [45, 46]. TFRC does not cause network instability; rather, it avoids congestion collapse and is fair to TCP flows, which are the dominant traffic on the Internet. TFRC shows low rate fluctuation, which is important for ensuring that video quality remains consistent in streaming applications. However, rate control over wireless links is still an issue that is open to debate. The main goal of a rate control protocol is to reduce the sending rate in the presence of packet losses in the network. However, in a wireless channel, packets can be lost due to congestion or due to wireless link problems. Wireless losses can be caused by channel fading, signal collision or interferences. Since the wireless losses are not related to the network congestion, the solutions for wired networks cannot be directly applied to the wireless scenario since they are unable to distinguish between dropped packets due to link congestion and dropped packets at the physical layer, which are due to channel errors. When packets are dropped, the sender reduces the data rate, even if the drop is due to wireless error, thereby impacting streaming performance. If a packet loss is classified as a wireless loss, the congestion control mechanism does not need to reduce the data rate. Therefore, the main challenge

in providing an effective rate control over wireless links lies in how to differentiate the cause of a packet loss.

5.2.3 Loss Differentiation Algorithms (LDAs)

Several LDA schemes have been proposed in the literature recently. The Biaz [10] and mBiaz [20] algorithms use packet inter-arrival time at the receiver to classify losses. The Statistical Packet Loss Discrimination (SPLD) algorithm [90] is based on the statistical value of inter-arrival times of received packets. The Spike mechanism [20] and the Packet Loss Classification (PLC) [58] use predefined thresholds on relative one-way trip time (ROTT) to differentiate losses. The ZigZag scheme [20] classifies losses based on the mean and deviation of ROTT. The ZBS algorithm [20] is a hybrid approach that switches among the Biaz, mBiaz, ZigZag and Spike schemes. The Trend-and-Loss-Density-based (TD) approach [29] uses the trend of the ROTT curve and the loss density to discriminate losses.

The Biaz Algorithm

The Biaz scheme [10] utilizes the packet inter-arrival time in order to classify packet losses. Considering that the bottleneck is the wireless segment of a wireless-last-hop topology, the packet inter-arrival time at the receiver will be constant T when no packet is lost. If:

$$(n+1)T_{min} < T_i < (n+2)T_{min} \tag{5.1}$$

is satisfied, where n is the number of consecutive packet losses, T_i is the instantaneous packet inter-arrival time of the first packet received after the a packet is lost, and T_{min} is the minimum packet inter-arrival time observed so far, then these n packet losses are classified as losses due to wireless problems. Otherwise, they are classified as losses due to congestion.

A drawback of the Biaz scheme is that it only works well in a wireless-last-hop scenario in which the wireless segment is the bottleneck link. Another drawback is that it can classify losses properly only when a single flow exists on the network segment considered.

The mBiaz Scheme

The mBiaz scheme [20] is a variant of the Biaz algorithm. The difference is that it makes a slight change in the upper boundary. Similar to Biaz, if:

$$(n+1)T_{min} < T_i < (n+1.25)T_{min} \tag{5.2}$$

is satisfied, where n is the number of consecutive packet losses, T_i is the instantaneous packet inter-arrival time of the first packet received after the a packet is lost, and T_{min} is the minimum packet inter-arrival time observed so far, then n packets are classified as losses due to wireless link errors. Otherwise, they are classified as losses due to congestion.

Statistical Packet Loss Discrimination (SPLD)

The SPLD mechanism [90] is shown in Similar to Biaz and mBiaz, it uses the packet inter-arrival time to discriminate the losses. In addition, the SPLD scheme fills the gap left by Biaz and mBiaz regarding multiple flows on the same link. It shows lower misclassification rates than Biaz and mBiaz. The SPLD scheme uses the stable average inter-arrival time T_{stable} to classify the loss type, whereas the Biaz and mBiaz schemes are based on the minimum packet inter-arrival time observed so far T_{min}.

Basically, upon the reception of a packet, the packet monitoring module will check its sequence number to detect any gaps. If no loss is found, the statistic manager module updates the stable average inter-arrival time T_{stable}, which is updated according to:

$$T_{stable} = (1-\alpha) * T_{stable} + \alpha * T_i \tag{5.3}$$

where $\alpha = 1/32$ and T_i is the instantaneous packet inter-arrival time. On the other hand, if there was a packet loss, the loss discriminator module will classify it according to the current average inter-arrival time T_{cur}, where $T_{cur} = \frac{T_i}{n}$, and n is the number of dropped packets. If $T_{cur} \geq T_{stable}$, the packet losses are classified and due to wireless problems. Otherwise, they are assumed to be losses due to congestion.

The Spike Algorithm

The Spike algorithm [20] classifies the losses according to the relative one-way trip time (ROTT). Basically, if the link is not in the spike state and the current *ROTT* is larger than *spike_start*, where $spike\_start = rott_{min} + \alpha * (rott_{max} - rott_{min})$, the scheme enters the spike state. Otherwise, if the connection is currently in the spike state, and the current ROTT is less than *spike_end*, where $spike\_end = rott_{min} + \beta * (rott_{max} - rott_{min})$, then the link leaves the spike state.

Furthermore, the receiver classifies the losses based on the current state of the link. If the packet loss happens during a the spike state period, it will be classified as congestion loss; otherwise, it will be classified as wireless loss.

Packet Loss Classification (PLC)

The PLC scheme [58], similarly to Spike, differentiates the wireless losses from congestion losses according to the relative one-way trip time (ROTT). When the receiver detects gaps in the sequence numbers, it will check the *ROTT* of the received packet. If $ROTT <$ *spike_end*, the losses are assumed to be wireless losses. If $ROTT >$ *spike_start*, the losses are classified as congestion losses.

For the *ROTT* between *spike_end* and *spike_start*, a trend detection algorithm to classify the loss type is based on:

$$S_f = (1 - \gamma)S_f + \gamma I(ROTT_i > ROTT_{i-1}) \tag{5.4}$$

where $ROTT_i$ is the $ROTT$ of the i^{th} packet. $I(X) = 1$ if X is valid, otherwise it is 0. The constant γ is the smoothing factor. If there is a strong decreasing trend, S_f will approach 0. If there is a strong increasing trend, S_f will approach 1. A threshold is defined as S_{th}. If $S_f > S_{th}$, the packet loss is assumed to be congestion loss. Otherwise, the packet loss is classified as wireless loss.

The ZigZag Mechanism

The ZigZag mechanism [20] uses the number of lost packets n, $ROTT$, and the mean and deviation of $ROTT$ in order to classify losses according to:

$$rott_{mean} = (1-\alpha) * rott_{mean} + \alpha * rott \tag{5.5}$$

$$rott_{dev} = (1-2\alpha) * rott_{dev} + 2\alpha * |rott - rott_{mean}| \tag{5.6}$$

Therefore, a packet loss is assumed to be a wireless loss if:

$$(n = 1 \bigcap rott_i < rott_{mean} - rott_{dev}) \tag{5.7}$$

$$OR(n = 2 \bigcap rott_i < rott_{mean} - rott_{dev}/2) \tag{5.8}$$

$$OR(n = 3 \bigcap rott_i < rott_{mean}) \tag{5.9}$$

$$OR(n > 3 \bigcap rott_i < rott_{mean} + rott_{dev}/2) \tag{5.10}$$

Otherwise, it is assumed as congestion loss.

The ZBS Approach

The ZBS algorithm [20] is a hybrid approach in which different LDA schemes are chosen based on whether the bottleneck wireless link is shared or not. When n flows share the bottleneck link, the average packet inter-arrival time T_{avg} will be almost $n \times T_{min}$, where T_{min} is the minimum inter-arrival time, according to:

$$T_{avg} = 0.875 * T_{avg} + 0.125 * T_i/n \tag{5.11}$$

Let $T_{narr} = T_{avg}/T_{min}$. If $rott_i < rott_{min} + 0.05 * T_{min}$, the Spike scheme is used. Otherwise, it selects one of the base algorithms according to the value of T_{narr} instead.

5.3 Problem Statement

5.3.1 Rate Control Over Wireless Links

Rate control is a challenging issue to address in wired and wireless streaming application scenarios. One of the most popular rate control schemes over wired networks is TCP Friendly Rate Control (TFRC)[45, 46]. TFRC does not cause network instability; rather, it avoids congestion collapse and is fair to TCP flows, which are the dominant traffic on the Internet. TFRC has low rate fluctuation, which is important for ensuring that video quality remains consistent in streaming applications. However, rate control over wireless links is still an issue that is open to debate. The solutions for wired networks cannot be directly applied to the wireless scenario as they are unable to distinguish between dropped packets due to link congestion and dropped packets at the physical layer, which are due to channel errors. When packets are dropped, the sender reduces the packet rate, even if the drop is due to wireless error, thereby impacting streaming performance. A number of approaches have been developed to improve TCP and TFRC over wireless networks [8]. Streaming applications can use Explicit Loss Notification (ELN) together with a Loss Differentiation Algorithm (LDA) in order to notify the sender the reason for which the packet drop is wireless error.

5.3.2 Link Status Feedback

The Real-time Transport Protocol (RTP) [98] and its control protocol (RTCP) provide a framework to end-to-end network transport services suitable for continuous multimedia applications, such as VoIP and video streaming. RTP does not guarantee any QoS requirement and it assumes that the network is unreliable and packets may be delayed.

RTCP monitors the sessions and provides a packet loss assessment of the channel for minimal feedback about the quality of service. However, RTCP is based on periodic transmission of control packets to all parties involved in the communication. In wireless networks, such periodic feedback may not perform efficiently because wireless channel services are highly dynamic regarding bandwidth, packet losses, and delay. Therefore, RTCP may deliver delayed feedback and the application will be unable to adapt the rate on time. Under such high mobility conditions, in which the topology may change more frequently, it would be necessary a higher rate of reports from RTCP. This would yield a significant increase in network traffic.

5.3.3 Unsuitability of Video Streaming Approaches

Continuous media streaming has received much attention in the last years, and several solutions can be found and exciting applications over the Internet are available to end-users. However, audio and video scheduling and buffering solutions cannot be directly applied to remote interactive virtual environment walkthrough systems. A video streaming protocol benefits from the fact that video data is continuous and the server must stream a certain number of video frames in sequence to the remote client, which will in turn play back these frames in sequence. This characteristic makes it possible to use buffers to avoid jitter during playback on the client end since the server "knows" beforehand what data it must stream. The client device is also aware of this information and therefore waits for the frames in sequence. On the other hand, predicting where the virtual user will next move in an interactive multimedia system is not a trivial task. A solution to motion prediction would yield many benefits such as the ability to focus the rendering of a scene to a specific area more quickly and stream the relevant images to the client-end rather than rendering all of the other scenes, which the user might not walk into. Another incompatibility between streaming video and interactive walkthrough scheduling algorithms is that the deadlines of video frames in the scheduler's queue will always be their real deadlines, whereas the deadline for an image for remote interactive

walkthrough may change as the user moves to another position. Therefore, the more distant an image is from the user's position, the lower the priority to send the image because the probability is higher for a change in the user's movement, thereby the probability of certain images being requested may change as the image falls out of the user's motion path. There is only one situation where the same scheduling scheme will show the same efficiency for both video frames and interactive image-based environment frames: when the user moves along a predefined path. This allows the system to perfectly predict the user's future positions, but affects negatively or limits the user's interactive experience. The perfect prediction in a video streaming system can be seen as guaranteeing the frame requests to be served before their deadlines. There are several scheduling strategies such as earliest-deadline-first (EDF), static priorities (SP), and first-come-first-served (FCFS), that can be used for video streaming.

5.3.4 Resource-Constrained Thin Mobile Devices

PDA-based wireless multimedia sensor nodes, such as the Stargate mote [111], have low processing power, small storage resources, and limited and inefficient 3D graphics rendering hardware. Therefore, demanding 3D applications are limited to low-detail virtual environments, thereby keeping potential interactive multimedia applications from being developed. In addition, downloading complex and large-scale virtual environments requires both high bandwidth and storage capacity. The high volume of 3D data and the dynamic nature of bandwidth pose significant challenges to providing smooth interactive navigation on PDA-based mobile devices over wireless networks: a mobile device can hold only a fraction of the entire scene, the 3D rendering engine is not able to process complex scenes in real-time, the bandwidth changes more frequently than it does on wired networks, and the wireless communication channel is highly susceptible to error. Most of the work concerning 3D rendering on mobile devices has been conducted over the OpenGL ES API [48] and is limited to the mobile device graphics hardware performance. Thus, the rendering quality is still at a very low level of detail. The Mobile 3D Graphics

API (M3G), defined in the Java Specification Request (JSR 184) [65], is another industry effort to create a standard 3D API for Java-enabled thin devices. A possible approach to overcoming these challenges is to render the complex 3D geometry on a graphics workstation server and transmit only images to the remote client, depending on the user's position in the virtual environment.

5.4 Our Proposed Interactive Streaming Approach

In this section, we present our proposed interactive streaming protocols. The ideas presented here aim at the streaming of interactive image-based scenes. Images will be streamed on demand based on the navigation inputs from the user. The challenges lie in: firstly, how to provide a rich and detailed image-based virtual experience on PDA-based wireless multimedia sensor nodes that are known for their lack of proper resources to process large scale 3D geometric data; and secondly, how to cope with the low resources of wireless sensor networks. We consider that the demanding geometry-rendering task is performed on a dedicated remote rendering graphics station that streams the rendering outputs, *e.g.*, images, to a client, thereby leaving only the displaying and certain minor image-based rendering tasks to the local, less powerful mobile hardware. Unlike video streaming, in which the sequence of video frames is known in advance, non-linear image streaming is not sequential and it depends on the user inputs, *e.g.*, position updates as the user wanders within a virtual environment. Therefore, we concentrate our investigations on non-linear streaming of images.

5.4.1 A Multi-Tier Architecture for Multimedia Sensor Networks

Although the advances in wireless sensor hardware have demonstrated a significant evolution in the last few years, it is still challenging to deploy wireless multimedia sensor networks, mostly due to the high bandwidth requirements of multimedia content. For in-

stance, the state-of-the-art sensor motes offer a maximum nominal data rate of 250kbps, which is low depending on the multimedia quality requirements of the application. In addition, the bandwidth problem is intensified when multi-hop communications are taken into consideration, *e.g.*, wireless sensor networks.

Therefore, we propose a three-tier architecture for multimedia sensor networks that focus the streaming of multimedia content "from outside in". The main idea is the integration of wireless sensor networks and multimedia streaming to enhance one's perception of the environment. For instance, the events detected by a sensor network, *e.g.*, tracking of people or objects and emergency situations like fire or gas leaks, could be integrated into a three-dimension virtual representation of the environment and feeded back to a wearable wireless multimedia device, *e.g.*, a head-mounted display or a PDA-based handheld, in order to provide an augmented reality tool for first responders.

The proposed architecture is depicted in Figure 5.1. The wireless sensor network tier comprises the scalar sensor nodes, *e.g.*, Mica motes [80][83], and is responsible for detecting the events and relaying the data to the mobile data collectors or sinks. Due to the constrained resources, this tier performs only scalar event detection and transmission. The second tier represents a hybrid tier that comprises the mobile data collectors and possibly the multimedia-end nodes. The third tier is represented by the sink or multimedia servers that will collect the information from the lower tiers, process the data, and stream the multimedia content to the interested parties.

5.4.2 The Layers of our Multi-tier Architecture

The development of a virtual environment streaming system for wireless multimedia sensor networks requires efforts on multiple layers of the architecture and several challenging issues must be overcome. The following list highlights our solutions for each layer of the architecture.

- **Scalar sensor node tier**: The deployment of a large sensor network involves low-

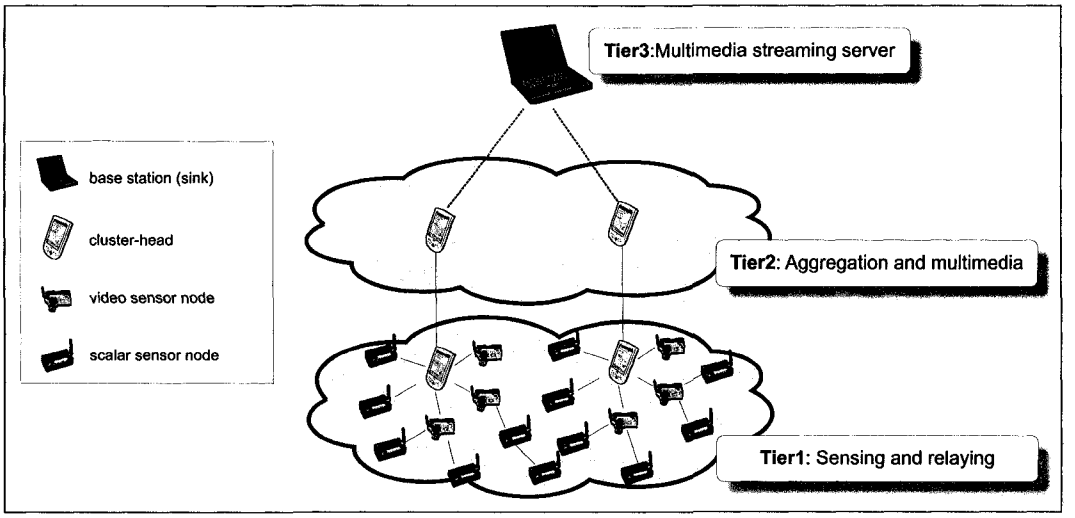


Figure 5.1: Wireless multimedia sensor network architecture.

cost and low-power sensor nodes. Due to their resource constraints, a simple but efficient routing protocol must be developed. If one considers real-time applications, *e.g.*, surveillance or emergency response, the data gathering mechanism should be fault-tolerant and present low latency for data delivery. To this end, we have proposed a set of routing protocols (CPEQ/PEQ) discussed in chapter 3 and a low-latency mobile data gathering scheme (MDC/CPEQ) in chapter 4 designed for delay-sensitive class of applications. Hence, the sensor nodes use the proposed protocols in order to deliver events from the sensor network to mobile collectors or sinks efficiently.

- **Mobile data collector and multimedia tier**: This tier comprises mobile devices capable of communicating with the sensor network tier as well as with the higher tiers. These mobile devices may have two roles in this architecture: they will collect data from the sensor network, process, and relay the data to upper tiers; and they will be the client-end multimedia receivers for the upper tier. The devices can be PDA-class Stargates [111] nodes or other thin mobile devices with two radio interfaces. For the data gathering, the nodes use the MDC/CPEQ approach. For the multimedia part, due to the 3D graphics limitations of mobile

devices, we propose a lightweight streaming solution that basically streams remote 3D rendering outputs to feed an image-based renderer at the client-side that will produce the novel views according to the user position within the environment. Therefore, for the streaming of remote rendering outputs, we propose a complete suite of protocols. From session initiation to buffering, we cover all the necessary components to provide a high quality virtual environment navigation experience over wireless networks. All these mechanisms are discussed along this chapter.

- **Base station - multimedia streaming server tier**: This tier is responsible for subscribing/registering interest on the data that the sensor node tier produces. It is performed through the MDC/CPEQ/PEQ protocols, as discussed earlier. The multimedia role of this tier will be the integration of the events relayed from the sensors into a virtual representation of the environment. This multimedia processing center will render the virtual environment and the results will be streamed to the multimedia clients (that can be the second tier in the architecture or any outside multimedia client that is registered to receive the streams). Thus, a real-time virtual representation of the environment can be explored and any events in the environment will be reflected in the virtual model. To this end, we propose a virtual environment streaming protocol, discussed in this chapter.

5.4.3 Our Proposed Interactive Streaming Protocol (ISP)

The Interactive Streaming Protocol (ISP) is a signaling protocol designed specially for exchanging interactive requests and responses between streaming servers and clients by controlling multiple streaming sessions of image-based virtual environment scenes. Both client and server use the interactive streaming protocol to request an action, reply, or report a link state. The ISP is responsible for the following tasks: locating the streaming server, establishing one or several streaming sessions, requesting the fetching of images from the server, and the termination of a streaming session. The ISP protocol does not deliver the stream of virtual environment scenes itself, but it is rather a component of a multimedia architecture that can be used with other IETF protocols, such as the Real-time Transport Protocol (RTP) to perform the data delivery.

Basically, a client device initiates a streaming session through an ISP request for SETUP that is sent to the streaming server. The server determines if it can provide the services to a new session through an admission control mechanism. Thereafter, it reserves the necessary resources to the streaming session such as streaming buffers and CPU processing time slots, and delivers the initial information about the 3D environment to the client. Upon the confirmation of the session establishment, the server sends the corresponding shared coordinate system and the initial images to the client. The client can start its navigation in the virtual environment. During the virtual exploration, the client-end sends ISP POS_UPDATE messages that contain the user's viewpoint coordinates and orientation in order to request new images. The server replies with the corresponding images. There is also a path prediction in which the server attempts to predict the future viewpoints of the user, based on recent POS_UPDATE messages received from the client, and it pre-fetches the corresponding images. If enough bandwidth is available, the server transmits the pre-fetched images in advance, without the client actually requesting them. Finally, at the end of a streaming session, the client-end informs the server to terminate its streaming session (TERMINATE message) and the server releases all the session resources.

The ISP protocol employs the UDP, which has been proved to be very efficient for real-time communication, together with feedback and rate control mechanisms. In order to obtain the reliable signaling, the ISP protocol uses timers and a retransmission mechanism.

Description of ISP's Packet Header

In this subsection, the accompanying ISP's header fields are discussed. The header fields specify the integrated information of a protocol message. They keep the ongoing streaming session updated and both client and server synchronized in terms of user's viewpoint and orientation.

- **ISP-ADDR**: This field defines the address or identification of the streaming server. It is mostly used by a client when searching for a server. Intermediate entities such as proxies or firewalls should be able to forward the ISP messages by means of an ISP-ADDR. If the ISP-ADDR is a host name and a domain, the originator or the proxy can obtain the destination IP address by looking up a DNS database.

- **SRC-ADDR**: It specifies the source of the message.

- **Session-ID**: The Session-ID is enclosed in every message exchanged by the ISP in order to identify the unique streaming session. The Session-ID is local and unique.

- **Command-Seq**: This field specifies a sequence number for each type of ISP message in order to identify and correlate the messages and their respective acknowledgements. This is necessary in the case of a packet is delayed or lost so that the the entities can distinguish the relationship between them.

- **Viewpoint**: This field stands for the position of the camera from which the image was rendered. It can be seen as the representation of the user's position within the virtual environment. It comprises the x and y coordinates of the viewpoint.

- **Direction**: Represents the orientation of the viewpoint specified by x and y. The direction is represented by an angle α.

Types of Messages in ISP

The ISP protocol exchanges three types of messages: requests, replies and status. A request message is used to request an action, for instance, a client can request to initiate a streaming session with the server. A reply message is employed to respond to a request, for instance, the server can respond to the clients request with an acknowledgment message.

- **SETUP**: This message is used by a client to establish a streaming session. If the server accepts the SETUP request, an acknowledgment message is sent. If the server rejects the SETUP request, an error message is sent back to the client, *e.g.*, "400 client error" or "500 server error".

- **POS_UPDATE**: This message is used to keep both client and server synchronized about the user's viewpoint such that the server can fetch or pre-fetch the images that will be necessary for the rendering of new viewpoints as the user moves within the virtual environment. The client sends this message whenever the user moves to a new position. The POS_UPDATE messages are used as input to a path prediction mechanism to pre-fetch images before being requested by the client in order to absorb jitter and provide a smooth navigation in the virtual environment.

- **SCENE_INIT**: Just after establishing a session, the client needs the first parameters of the virtual environment. Therefore, the server sends a SCENE_INIT message to initialize the navigation parameters, such as the initial position and orientation, the shared coordinate system the session will use. The SCENE_INIT message is used whenever the user enters a new region of the virtual environment that might have different scene or coordinate parameters, *i.e.*, when the coordinate system needs to be reset.

- **TERMINATE**: This message is used by a client or server whenever a streaming session should be terminated. Thereafter, the server can release all the resources allocated for the session.

- **ACK**: This message is used to acknowledge all request messages. A timer is set when a request is sent and an ACK is expected. If an ACK message does not arrive within the specified period, the request will be retransmitted. The ACK message has the same Command-Seq header as the corresponding request message so that a correlation between them can be made.

The Dynamics of the ISP Protocol

This subsection illustrates the dynamics of the ISP protocol by presenting some scenarios that represent the typical actions taken in a session initiation protocol.

The establishment of a streaming session, for instance, starts with a client sending a SETUP message, as depicted in Figure 5.2(a). The client sets the ISP header fields for the SETUP message by assigning the address of the server to ISP-ADDR and it sets the source address field SRC-ADDR with its own address. A unique Session-ID is created at the client and it will be valid until the session terminates. The Command-Seq is set in order to represent the request and its sequence number. After the session is established, the client sends a "200 OK" to notify the server that the initiation process was successful. Thereafter, the server must send the initial parameters about the current virtual environment by sending a SCENE_INIT message. The SCENE_INIT contains the necessary information to start the virtual exploration at the client side. The client acknowledges the SCENE_INIT message and starts the interaction with the virtual scene.

After establishing the session, the user will start exploring the virtual environment. As introduced earlier, the virtual environment is rendered remotely and only the images that corresponds to the position and orientation of the user will be streamed to the client on demand. The demand is driven by the current viewpoint of the user. Therefore, as the user moves, additional images must be streamed so that the client's image-based

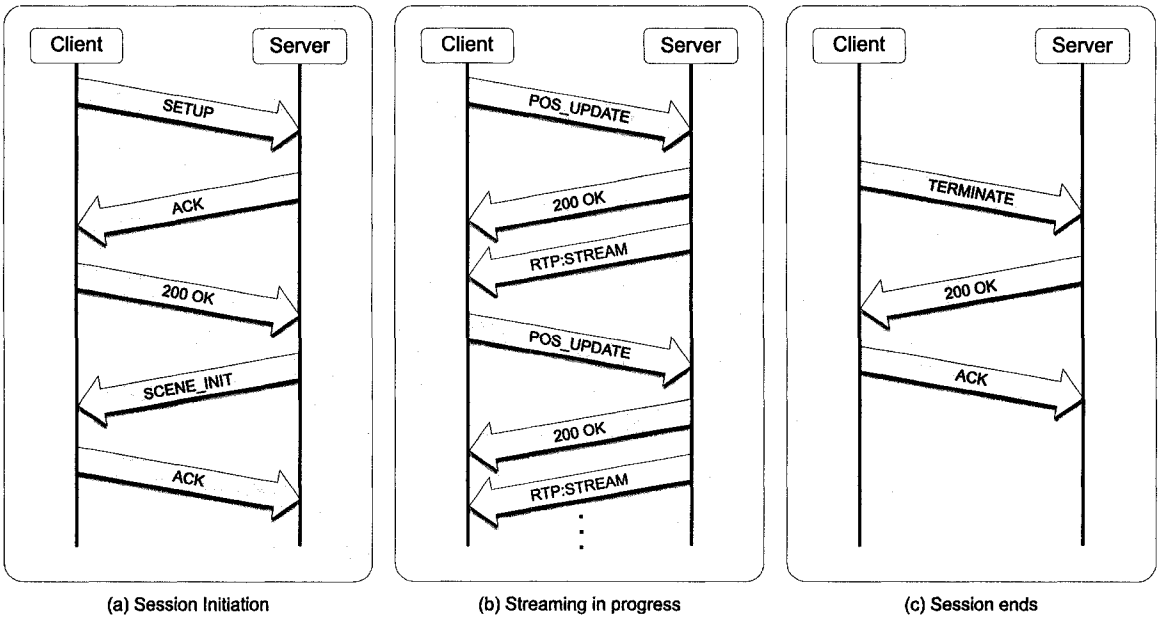


Figure 5.2: Typical scenarios of ISP's session dynamics.

rendering mechanism is able to render new views based on the received images. As shown in Figure 5.2(b), the client updates the server with the most recent viewpoint of the user by sending POS_UPDATE messages that carry the actual position coordinates and the orientation angle. Upon receiving a POS_UPDATE message, the server replies with a "200 OK" status message and starts the transmission of the corresponding images. The "200 OK" message will instruct the client that a streaming flow is coming so that the client can prepare the resources for the incoming images, *e.g.*, image buffers, memory, CPU allocation for the image-based rendering, etc. The POS_UPDATE messages can also be used as input to a path prediction scheme such that the server will pre-fetch images to match future positions of the user within the virtual environment. A path predition mechanism will be discussed later.

Finally, a TERMINATE message is sent when the client or the server needs to close the session, as depicted in Figure 5.2(c). The client sends a TERMINATE message and waits for a reply from the server. The server responds with a "200 OK" that indicates the server will release its resources for that session. The client acknowledges the reply

and the session is terminated.

5.4.4 An Interactive Image-based Transport Protocol (IITP)

The Image-based Transport Protocol is a combination of some existing protocols for rate adaptation and loss differentiation in wireless networks. The ISP protocol will use the IITP to deliver the data to the requesting clients. In this section, the header format and the rate adaptation schemes adopted by IITP are discussed.

The IITP's Packet Formats

The data packet is shown in Figure 5.3. Its fields are described bellow:

- **PT:** This field specifies the IITP's packet type. The possible values are: 0 if it is an image packet; 1 if it is an ACK packet; 2 if it is a viewpoint update.
- **Direction:** Specifies the direction of the viewpoint, in degrees, based on Cartesian Plane for the image acquired by the rendering process. The direction of a viewpoint is captured from the virtual camera parameters. Both client and server must share a common coordinate system.
- **X and Y coordinates:** These fields specify the position of the camera from which the image was rendered. It is the representation of the user's position on the virtual environment. Along with the direction field, the X and Y coordinates have to be transmitted to the client device in order to allow the application to match the received image to the user's viewpoint.
- **Timestamp:** The timestamp indicates when the packet is sent. It is used to calculate round trip times, useful for the rate control mechanism that is described in section 5.4.4.

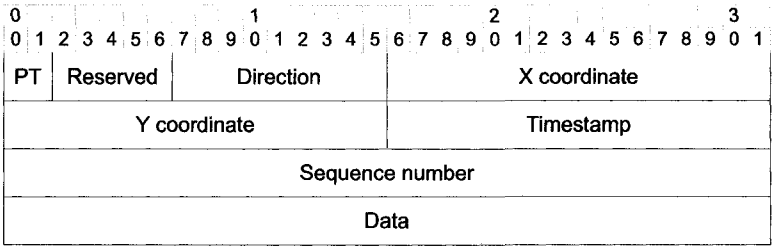


Figure 5.3: ISP's image header

- **Sequence number:** A sequence number is used in order to keep track of dropped packets by checking gaps in the sequence and in order to create a relation between packet and acknowledgement. It is also useful for defragmentation.

- **Data:** Carries the actual image data.

We must also specify an acknowledgement packet header format to our interactive streaming protocol because UDP does not provide any kind of reliable communication. The ACK packet will be used to inform the sender about dropped packets and the rate at which the client is receiving data. The ACK header format is depicted in Figure 5.4. Its fields are described below.

- **PT:** This field must be set to 1 in order to conform with the proposed protocol.

- **CLN:** This bit flag means Congestion Loss Notification. This field will be used by the rate control algorithm to notify the sender about potential dropped packets due to congestion. The sender will reduce the transmission rate if congestion has been detected.

- **Reserved:** This field is reserved for future updates to the protocol.

- **GAPr:** It is the time from the moment the receiver received the first packet to the moment it received the last packet of an image. This information will be used by the sender to adjust the rate of transmission. The rate control algorithm and its parameters are described in the next section.

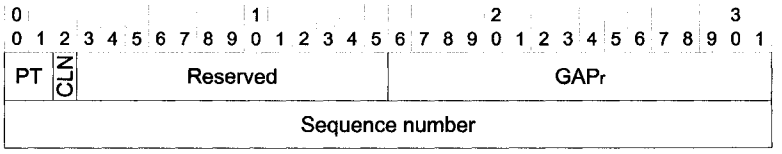


Figure 5.4: Acknowledgment header format

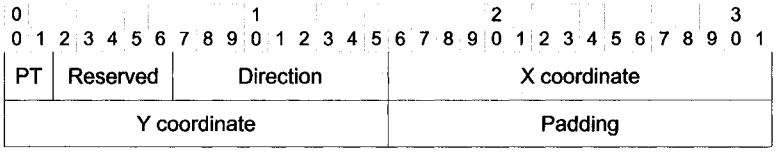


Figure 5.5: Viewpoint update header format

- **Sequence number:** The value of the sequence number of the last image fragment received is assigned to the sequence number of the ACK packet in order for the sender be able to relate the image sent to the ACK.

In order to keep the sender updated about the user's position and direction in the virtual environment, a user's viewpoint update header format must be specified. This header contains information that is vital for the operation of ISP's algorithms. As the user moves in the virtual environment, the server needs to update the virtual camera's position and direction in order to render the new viewpoint and stream the image back to the client. The ISP's viewpoint update packet is shown in Figure 5.5. A brief description of its fields is provided below.

- **PT:** This packet's type must be set to 2 in order to comply with ISP's protocol.
- **Direction:** This field carries the direction of the user's viewpoint.
- **X and Y coordinates:** These are the coordinates of the user's position.
- **Timestamp:** Is used to avoid late updates, destroying the consistency between the user's virtual environment and the remote server's virtual environment until the next update is received.

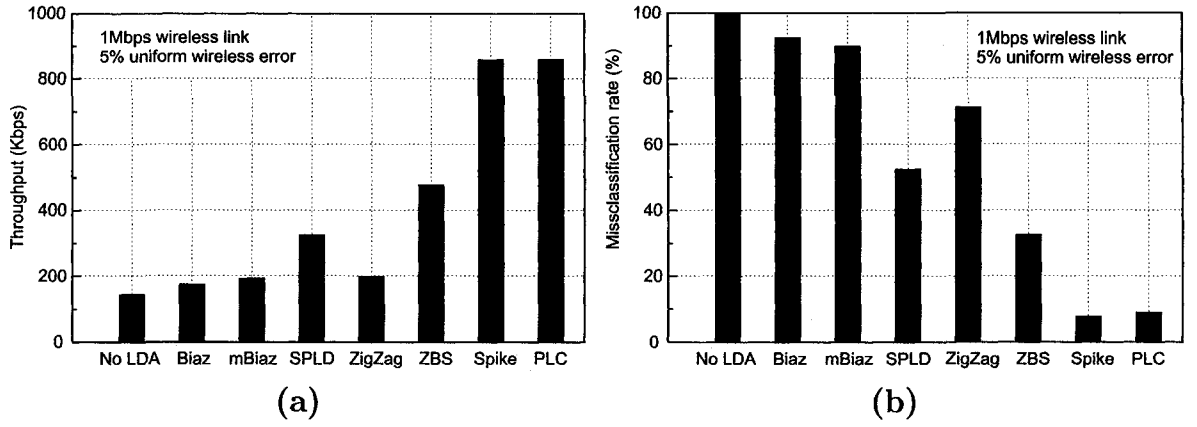


Figure 5.6: Max throughput and misclassification rate of the LDAs

IITP's Congestion Control

The UDP protocol does not provide any form of reliable transmission. Therefore, a congestion control and a feedback mechanism must be used in IITP. Thus, we propose the Gap-based Adaptation Protocol (GAP), which is a rate adaptation that works together with a loss differentiation algorithm called Spike [20], that monitors the relative one way trip time (ROTT) constantly and is able to distinguish wireless errors from congestion losses. Hence, wireless losses will not be counted when computing the GAP's rate adaptation, thereby increasing the bandwidth utilization rate by avoiding that data rate is decreased due to wireless losses. We have chosen Spike based on its performance results that indicate the highest bandwidth utilization and lowest loss misclassification rates among the LDAs studied, as depicted in Figure 5.6. Please refer to section 5.2.3 for more information on Spike and other related loss differentiation algorithms.

The sender keeps track of the time it took to deliver consecutive images. This time will be the sent gap (GAP_s). Upon receiving an image, the receiver calculates the time between the arrival of consecutive images. This is the received gap (GAP_r); it is illustrated in Figure 5.7. The receiver then sends an acknowledgement message to the sender in order to inform the GAP_r.

The sender receives the ACK packet and compares the received GAP_r to its own

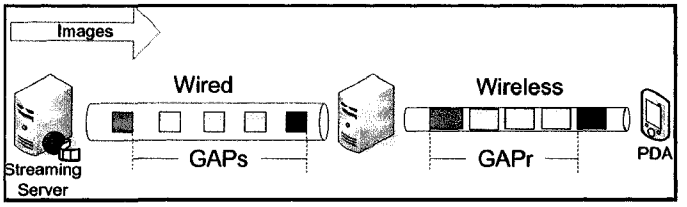


Figure 5.7: Sent gap and received gap.

GAP_s. If GAP_r is greater than GAP_s, the sender is probably transmitting more data than the receiver can handle; this is most likely due to congestion. The sender therefore reduces its transmission rate to the received rate according to:

$$I = I \times \frac{GAP_r}{GAP_s} \times \beta, \beta >= 1.0 \tag{5.12}$$

where I is the time interval for sending consecutive packets and β is a constant used to fine-tune the rate control mechanism. If GAP_r is less than or equal to GAP_s, the sender continues increasing the transmission according to:

$$I = I \times \delta, \delta < 1.0 \tag{5.13}$$

until GAP_r is greater then GAP_s. δ is a constant to fine-tune the increasing step. In our experiments, δ is set to 0.95. The receiver also notifies the sender about any dropped packets that were due to congestion by setting the CLN bit flag in the ACK header. The decision to set this bit is based on the spike state of the link. The Spike algorithm detects if the link is in the spike state, and if so, the CLN bit is set, forcing the sender to reduce the rate and avoiding queue buildups and congestions. Upon receiving an ACK packet with the CLN bit on, the sender must reduce its transmission rate using equation 5.12 but with $\beta > 1$. We used $\beta = 1.10$ in our simulation experiments in order to decrease the sending rate by additional 10% of the usual rate adaptation $\frac{GAP_r}{GAP_s}$. The Spike algorithm will also detect when the link has left the spike state and the CLN bit is not set for the next acknowledgment packet.

5.5 Background Information on Image-Based Rendering

5.5.1 Image-based Rendering Model

Generally, realistic 3D models are complex data structures resulting in large file sizes, restricting low bandwidth network use. In addition, complex models are unsuitable for rendering on low-end thin mobile devices. In image-based model, rendering is performed by the sender, and the resulting images are streamed over the network to the client side, which just performs the displaying task. Also called as Remote Rendering, in which the rendering is performed by the a remote entity, the results are streamed over the network, and the client utilizes those reference images to render novel views according to an image-based rendering algorithm. This model aims at avoiding data starvation at the client by reconstructing novel viewpoints based on reference images available on its cache, while waiting for the remote entity to stream new reference images.

For decades, research on traditional 3D geometry rendering have dominated the literature on graphics rendering [4]. Recently, a new rendering mechanism based on images has been drawing a great deal of attention. Image-Based Rendering (IBR) can render photo-realistic views of the environment and does not depend on scene complexity. IBR techniques combine computer vision and computer graphics algorithms to generate photo-realistic viewpoints.

Traditional computer graphics focus on projections and calculations using geometric 3D data of the virtual environment and objects. Therefore, the cost to render a geometry-based scene is highly dependent on scene complexity (i.e. number of objects, number of polygons, resolution of texture maps, lighting effects, etc). Thus, rendering complex and photo-realistic virtual environments is limited to expensive graphics workstations. On the other hand, image-based techniques depend only on the resolution and quality of reference images. IBR can easily be performed by software-based render-

ers. This allows, for instance, that complex virtual environments be rendered on thin handheld devices.

Image-based rendering approaches are based on the famous Plenoptic Function [2]. The appearance of the world can be thought of as a dense array of light rays filling the space that can be observed by eyes or cameras. These rays are represented by the plenoptic function $P(\theta,\phi,\lambda,t,V_x,V_y,V_z)$, which is a 7D function with an eye or camera at every possible (V_x,V_y,V_z) position that records the intensity of the light rays passing through the center of the camera at every possible angle (θ,ϕ), for every wavelength λ, at every time t. For instance, lightfield and lumigraph [74, 49] are 4D subfunctions of the Plenoptic function by assuming that the scene is static. Such representations have a large amount of data and high computational complexity. The image acquisition systems for these representations are generally complex camera arrays, such as the Carnegie Mellon mobile camera array [19] and the Stanford multi-camera array [110].

Panoramic Image-Based Scene Representations

By constraining the viewing space, the dimension of the plenoptic function is reduced as well as the amount of images required for reproducing the scene. For instance, assuming a static camera aiming at a certain direction, an image or a video sequence captured at that position and direction is good enough for representing the scene.

If we constrain the viewer to be on a plane, the plenoptic function can then be reduced by one dimension, as the viewers space location becomes 2D. This assumption is feasible so as the eyes of person are usually at a certain height level for walkthrough applications. Furthermore, if we assume the viewer is static, panoramic images can be acquired by rotating a camera horizontally on a fixed tripod. Figure 5.8 shows an unwrapped cylindrical panorama to a flat image.

Panoramic images are now popular 3D IBR representations due to QuickTime VR [26] and others [47, 85]. Two of the most used representations are the cubic and the cylindrical panoramas.

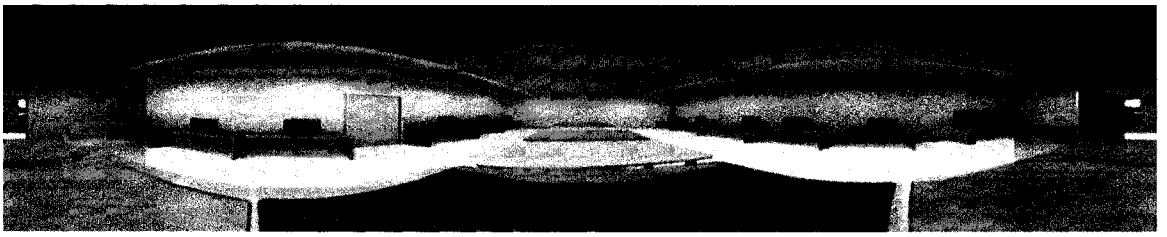


Figure 5.8: Unwrapped cylindrical panorama.

According to [114], if we denote the image coordinates as *(x, y)* and the cylinder coordinates as *(α, h)*, we can transform from image coordinates to cylinder coordinates according to:

$$\alpha = \arctan \frac{x}{f} \tag{5.14}$$

$$h = \frac{y}{\sqrt{f^2 + x^2}} \tag{5.15}$$

where f is the focal length and α is the panning angle.

Cubic panorama is another IBR representation where a panorama captures 360^o field of view horizontally and vertically, and stores the result in six faces of a cube. This is a type of projection for mapping a portion of the surface of a sphere (or the whole sphere) to flat images. Four cube faces cover front, right, back and left, one the top and one the base, each of them having 90^ox90^o field of view. Panoramic images in the cubic projection are commonly used as the image source by several spherical panorama viewers such as Quicktime VR [26].

A panorama does not have necessarily to be composed of static images. Panoramic videos can be captured by multi-camera systems such as LadybugTM from Point Grey [94], among others.

5.5.2 Remote Rendering Frameworks

There are several existing remote rendering systems. The simplest case of remote rendering uses remote desktop methods such as Virtual Network Computing (VNC) [100] or

Microsoft Remote Desktop. In VNC, a remote system running a VNC server sends screen captured images to client machines. Updates are triggered by the client-end when the user presses a key or pointer button, or moves the pointing device - the update protocol is demand-driven by the client.

A well known framework for remote rendering is Chromium [59]. Chromium is a special library implementation of OpenGL that provides applications with functions for parallel and cluster graphics rendering. Basically, Chromium intercepts the application's OpenGL calls and sends them through a graph of processing nodes in order to process and render. Finally, the resulting frame-buffer (images) are extracted and transmitted to a remote client for display.

The system described in [14] is another example of remote rendering. The server renders the virtual environment as a movie that is transmitted to the mobile clients. This is a simple solution and is limited to non-interactive virtual walkthroughs. The system presented in [124] uses a web browser as client application for displaying the remote 3D environment. Authors show that they can provide better frame rates using image-based techniques. In [35], the authors employ the combined processing power of clients and server. The main idea is to just render 2D line primitives on the client, that have previously been generated on the server derived from arbitrary 3D scenes. The authors in [40] developed a remote control interface for SGI's Open Inventor [89] applications. The system transfers compressed images from the server to a Java-based client and returns events generated at the client via CORBA requests

5.5.3 Existing Interactive Image-Based Streaming Systems

This section presents recent research on image-based streaming of interactive virtual environments.

The Rail-Track Viewer [121] focuses on using Image-Based Rendering (IBR) techniques to examine and walkthrough large and complex virtual scenes on a remote high resolution display across the Internet. It can be seen as an extension to QuickTime VR

or other panoramic representations. The authors developed a system that allows the user to move forward and backward by restricting the movement along pre-selected "tracks", in which the user can interact with the environment only at some points as a panoramic viewer. Basically, data sets are pre-rendered using appropriate rendering engines and the images and their geometric information are stored on the server. The reference images are sent to the client on request. The client will use the reference images to reconstruct novel views by using a layered panoramic IBR model. The images are pre-fetched to a cache, waiting to be transmitted to the client. The pre-fetching algorithm is based on the track points in which the client needs a panoramic representation. The points closer to the viewpoint in the virtual environment will have more panoramic information cached in order to speed up the transmission. The client will maintain at least two points with complete panoramic imagery, which requires large bandwidth and storage capacities. This pre-fetching mechanism can be seen in Figure 5.9, where: **(a)** the viewer is located between viewpoints V_1 and V_2. The entire panorama for viewpoints V_1 and V_2 are loaded into cache, while only parts of the panoramas (tiles) are loaded in for the other viewpoints V_3, V_4 and V_5; and **(b)** the viewer moves into the direction of V_3. The pre-fetching completed loading the panoramic information for V_3 and updated the other viewpoints based on the distance of the viewer to the viewpoint. According to the authors, experiment results have shown that their system is able to render at 10 to 15 frames per second (FPS) for 1k x 1k image resolution on a Sun Blade 1000 workstation with dual Ultra-Sparc III 750MHZ, 1 GB of memory and Elite 3D graphics card. Drawbacks of this work are: it cannot be directly applied to thin mobile devices due to high bandwidth and storage requirements; user navigation is limited to pre-selected tracks and viewpoints.

In [116], a client-server approach to image-based rendering on mobile terminals is presented, where the objective is to make it possible to render 3D scenes at interactive frame rates on mobile devices through an IBR method and a client/server architecture. The author's main contribution is how to place the cameras in the virtual environment in

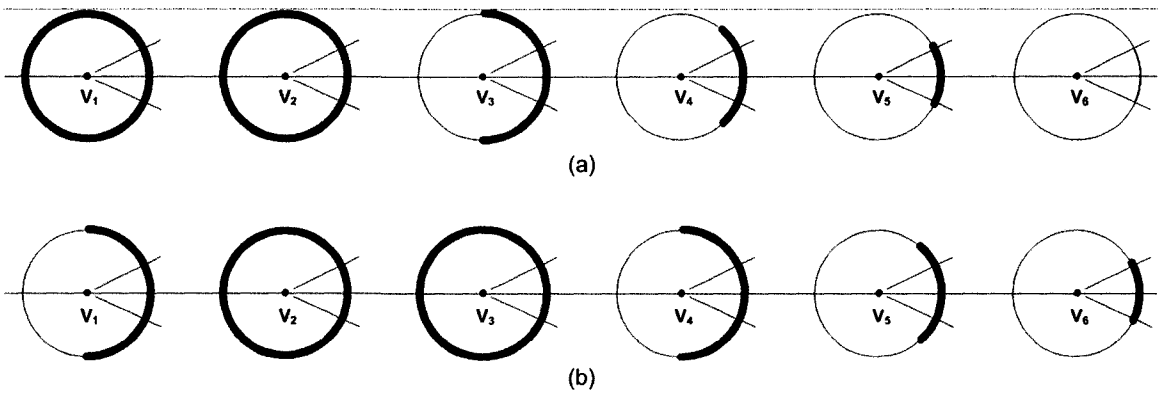


Figure 5.9: Pre-fetching mechanism presented in [121] .

order to avoid exposure and occlusion problems when using IBR. They use the footprint of an urban environment (streets and buildings) to properly select reference images to transmit to the mobile device. Therefore, the camera placement solution works only for urban scenes, limiting its applications. Basically, the 3D virtual environment is rendered on the server, which uses the camera placement algorithm to capture pertinent reference images, and then sends the images to the client that renders new views using IBR. The server sends the initial reference images and the client starts its IBR process. As the user moves within the environment, the client device sends the user position and orientation to the server, which will determine if the client needs new reference images. Experiment results have shown that their system is able of rendering less than 4 FPS on a PDA and between 8 to 20 FPS on a PC.

Using remote rendering, the authors in [69] aim at the protection of copyrighted 3D models manipulated by remote users. The server holds the entire 3D virtual environment and sends certain images to remote clients on demand . The client renders a low-polygon model of the scene while the user interacts with it. When the user stops, the client sends a request to the server that sends back a high-resolution image of the virtual scene.

The authors in [11] present another work on remote interactive virtual environments where the server renders a 3D virtual world and sends images to a client device. The client renders novel views using image-based rendering. The main idea of this work relies

on sending only the difference between consecutive frames, instead of sending the entire frames. The client sends image requests and wait for the server to reply. Each client uses previous views of the environment to predict the next view, using the known camera motion and image-based rendering techniques. The server performs the same prediction, and sends only the difference between the predicted and actual view. The drawback of this approach is that the client may experience long delays waiting for the next image, because it is a client-based request mechanism. Another disadvantage is that it uses lossless compression. The authors used 256x256 images (192KB) for their experiments. It would be relevant to compare this approach with a lossy compression scheme, where entire 256x256 images (20KB) are transmitted in order to evaluate the efficiency of the proposed solution.

In the work presented in [24], the authors explore an alternative approach to achieve 3D graphics capability on mobile devices using image-based rendering. Unlike the geometry-based 3D graphics pipeline, the rendering time of image-based rendering depends on the screen resolution of the output images rather than the complexity of the input models. This offers a potential advantage for mobile devices that typically have small display areas. The authors present a client-server framework for mobile devices equipped with IEEE 802.11b based wireless interface. They have shown a framework in which a 3D graphics programs running on a desktop computer may be used to interact with users on mobile devices. This can simplify the process of developing 3D graphics software for thin mobile devices and offer a way to offload part of the 3D rendering task to the server. Basically, a client device requests images from a server, which renders the 3D environment and send back reference images and depth maps. The client uses the reference images to render intermediate views using McMillan's image warping algorithm [79]. Due to the high capacity server, their scheme can render high quality 3D scenes, which cannot be done on the client through traditional geometry-based rendering. In their experiments, the system is able to render novel views at the speed of 5.9 to 6.2 frames per second on a 206MHz StrongArm processor PDA.

5.6 Proposed Remote Rendering Approach

The proposed interactive streaming system mainly relies on a client/server architecture. The server is the virtual environment host. It holds all of the 3D geometry data that needs to be rendered. Client devices send connection requests to the server. During the connection phase, the initial position and orientation of the user are relayed to the server, as described in the ISP protocol, section 5.4.3. When a session is accepted, a virtual camera is created on the OpenGL-based virtual environment according to the parameters received from the client-end during the session establishment step. The sender sends the initial rendering outputs to the receiver. The receiver will calculate the time it took to receive the image and notify the sender, which will adapt its sending rate according to this information. If no congestion losses are observed, the sender can increase its transmission rate; otherwise, it reduces the rate according to the IITP's rate control mechanism. Therefore, this approach can achieve low rates of fluctuation.

5.6.1 Remote Rendering

In order to overcome the graphics limitations of mobile devices that were discussed earlier, we employ a remote rendering mechanism. A rendering engine based on OpenGL was implemented at the server side. Depending on the graphics hardware capabilities and geometry complexity, the server can render hundreds of frames per second. Our streaming system defines a buffering scheme to improve the quality of navigation on the client device. The idea is to have all possible viewpoints to which the user can go available during the next step. The buffer will be filled with images according to the user's position and direction on the virtual environment. A simple movement prediction scheme is used and priorities are assigned to each position in the buffer, as shown in Figure 5.10.

The movement of the virtual camera is limited to a plane; the camera cannot look up or down, and it can rotate only in steps of 30 degrees. This minimizes bandwidth usage.

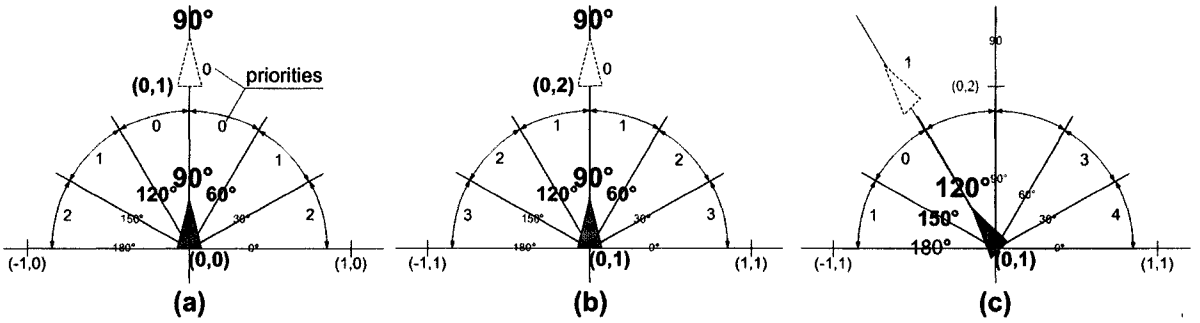


Figure 5.10: The assignment of priorities based on the user's viewpoint.

Suppose that at time t_0 the user's viewpoint and direction are 0.0 and 90 degrees respectively. The next possible viewpoint positions are shown in Figure 5.10(a). Viewpoints are represented by (*x, y, dir*) where x and y are the coordinates of the position and *dir* is the direction in which the virtual camera is pointing in the scene. After every change in position or direction, the new viewpoint coordinates are updated to the server, which keeps track of the user's movements. Thus, at time t_0 the user can move to position (0, 1, 90) or rotate to the left (0, 0, 120) or right (0, 0, 60). Therefore, these three positions have higher priority in terms of being rendered and streamed by the sender. Now, let us suppose that the client has received the three high priority images and the user has decided to move forward to position (0, 1, 90). Because the image representing that position is already in the buffer, the transition between the viewpoints is smooth with no lag. As depicted in Figure 5.10(b), at time t_1 the user's next possible movements will be forward (0, 2, 90), left (0, 1, 120), or right (0, 1, 60). The prediction scheme also considers previous movements of a user when assigning viewpoint priorities. If at a previous step t_0 the user moved forward to (0, 1, 90), for instance, we consider that he or she will continue in the same movement. Therefore, at time t_1 the user will most likely move forward to (0, 2, 90). The new priorities for time t_1 are shown in Figure 5.10(b). The same idea applies in the case of rotation. Suppose that at step t_{n-1} the user was at (0, 1, 90) and at step t_n she or he rotated to the left at (0, 1, 120). The highest priority will therefore be assigned to viewpoint (0, 1, 150) for the next step t_{n+1}, as can be seen in

Figure 5.10(c). The sender must first stream the images with priority 0, then priority 1 and 2, successively if there is available bandwidth. With these prediction and buffering mechanisms, the sender can stream images in advance to the receiver when there is available bandwidth, thereby improving performance at the client side, as the client device does not need to wait for the next image when the user changes the viewpoint because the image will already be available locally.

5.6.2 Buffering for Interactive Image-based Streaming

In order to overcome the graphics limitations of mobile devices, we employ a remote rendering mechanism. A rendering engine based on OpenGL was implemented in our system. Our streaming system makes use of a buffering scheme to improve the quality of navigation on the client device. The idea is to have all possible viewpoints that the user can go to during the next step available on the client side. At the server side, the application decides what viewpoints will be requested to the rendering engine based on the position updates received. The server side will also handle the priority assignment of the viewpoints. The selected viewpoints will then be rendered and streamed to the mobile host according to the scheduling algorithm. The mobile client holds a buffer to cache the received images and serve the image-based rendering algorithm when requested. Both client and server share a common coordinate system in order to provide user interaction and consistency between the client and server. Priorities are assigned and a buffer is constructed according to Figure 5.11. Suppose that at time t_0 the user's viewpoint and direction are depicted in Figure 5.11(a). The rectangles represent the images that compose the panoramas, and the numbers represent how many steps are needed to reach the position that requires the image, as well as the priority assigned to that image. For instance, the camera in Figure 5.11(a), or the user's viewpoint, is facing forward. The image immediately in front of the camera is assigned priority 0 because it is the image that is immediately required by the image-based rendering in order for it to compose a view for the user. From this point, priority 1 is assigned to the side images as they

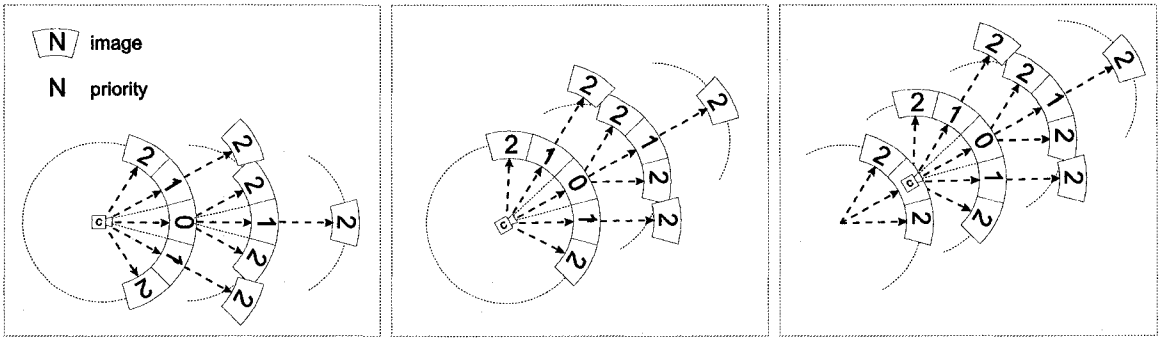


Figure 5.11: Image buffer structure. Priority scheme.

are the most likely movement the user will perform. Therefore, these images should be available at the client's buffer before they are requested. Priority 1 is assigned to the image required to render the viewpoint that is one step ahead of the current position. Unlike video streaming and buffering, the walkthrough system cannot perfectly predict the next frames because the user can move to any of these three viewpoints (left, right, or front). If we adopt a lookahead of 2 steps in the virtual environment, priority 2 must be assigned to the images that can be reached within 2 steps of the user's current position. Figures 5.11(a) to 5.11(c) show the user turning one step left and going one step forward. Notice how priorities are changed for the same image as the user moves through the environment. In Figure 5.11(b), when the user rotates the view to the left, the left image of that panorama, which had priority 2 is now assigned priority 1 as it is one step further from the user's current position. This change must be reflected in the scheduler queue because that image now has a higher probability of being used by the client. The scheduler will have to modify the deadline of that image. Since deadlines will be updated constantly, their entry positions in the scheduler queue may change and some of them may even be removed from the queue as their deadlines recede beyond the lookahead currently in use.

Description of the Scheduling Mechanism

As explained before, the client application keeps track of the user's position and direction and sends this information to the server. The application at the server side is responsible for handling multiple concurrent sessions and computes the possible images that may become visible to the user. Afterward, requests are delivered to the scheduler with their respective deadlines and are sent to the network resource. In order to provide quality of service, the images must be delivered to the client before they are needed by the image-based rendering module, i.e. before the image comes into the user's view. Therefore, a scheduling mechanism is necessary in order to guarantee that all of the sessions are served on time and to better utilize available bandwidth by accepting the number of sessions that the server can handle.

Like video streaming buffering solutions, we use a fixed lookahead LA where a session looks ahead into the future for LA steps and requests images that may be needed in LA steps in advance. A step is the time it takes to move from one viewpoint to another, i.e. between two consecutive images. This can be expressed as the user's speed. In our system, viewpoint transitions are performed smoothly by the panoramic renderer. For example, when the user is rotating, the renderer projects the intermediate views in order to provide smooth navigation. In the case of the user moving forward, a zoom is performed up to a level at which another image will be required, i.e. one step further, in order to guarantee image quality. Considering Figure 5.12, the darker image is one step further than the camera's current position. The application at the server side will apply a deadline D to the image according to:

$$D = \frac{St}{S} \tag{5.16}$$

where St is the number of steps and S is the speed. In our simulations, we used $S = 1$ second, which represents the time the transition between views take after the user input.

The time the 3D renderer takes to render one frame is denoted by T_r, and the time to

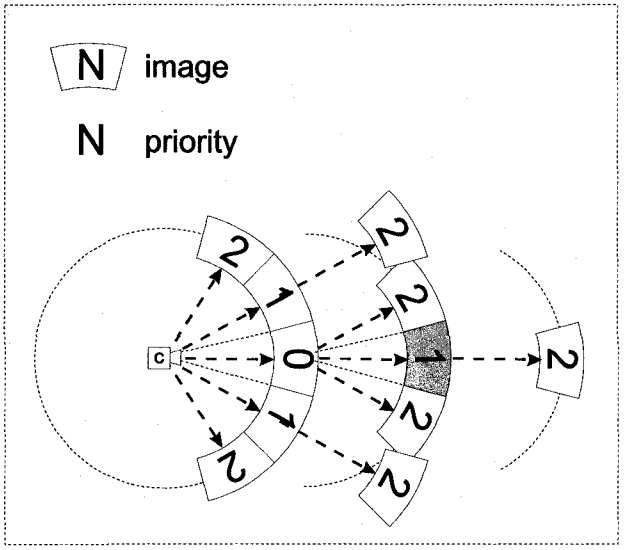


Figure 5.12: The darker image is one step away from the viewer.

perform the cylindrical warping and image compression is expressed by T_c. Considering that a requested image can arrive at the mobile host in T_i seconds (Time to transmit an image plus the network path delay), in order to guarantee that an image will arrive at the mobile host before its deadline, we must guarantee a maximum time during which the request will stay in the scheduler queue. This time is denoted by:

$$T_s \leq D - (T_r + T_c + T_i) \tag{5.17}$$

where T_s expresses the time in the scheduler queue. A diagram of the time spent in each step of the walkthrough system can be seen in Figure 5.13.

The scheduler uses a most-up-to-date earliest deadline first request release policy. However, unlike video streaming, which does not allow a request to be removed or updated once it has entered the scheduler, non-linear interactive multimedia requires dynamic deadlines because the user may change his or her path in the virtual environment, therefore requiring an update in the queued deadlines. In video streaming, the scheduler knows the real deadlines of the requests; thus it is possible to calculate the maximum number of sessions the server is able to accept. However, in our remote interactive mul-

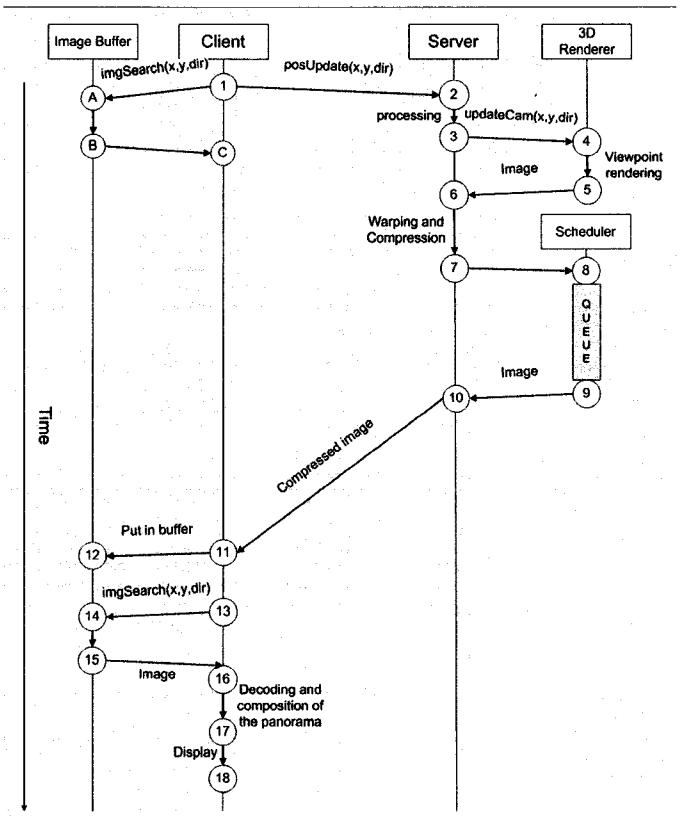


Figure 5.13: Phases of the entire streaming system.

timedia streaming, the deadlines may change on the fly, making it difficult to compute an optimal number of admitted client sessions. When the server receives a position update from the client, an image that had priority 2 in a previous position update may be assigned priority 1. Thus, the scheduler must update its entry in the queue in order to reflect the change. The scheduler re-calculates the deadline D and the maximum time in the queue T_s for that specific request entry.

In order to accept a new client session request, the scheduler has a different approach. To compute whether a new session should be accepted or not, requests of the images with priority 0 and 1 (or steps) will be considered to be carrying their real deadlines and will be served accordingly. Images that are 2 or more steps away from the camera's current position will be kept and only be served if the scheduler is unloaded (when bandwidth permits). This is necessary because the more distant the image is from the user's position,

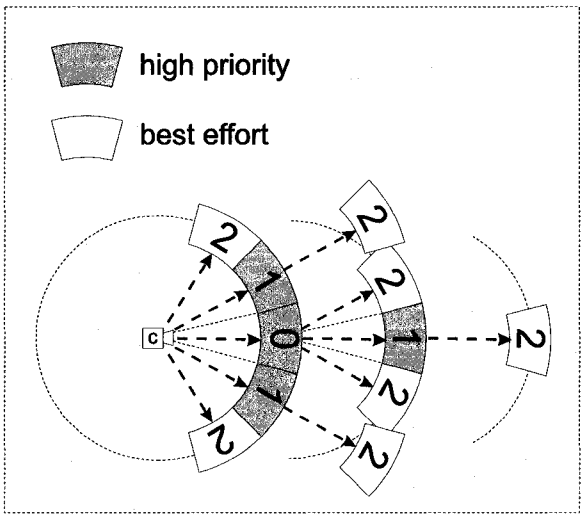


Figure 5.14: Images with priority 0 and 1 must be delivered.

the higher the probability of that image's deadline being modified, as the user may take a different path, which would mean that the image would no longer be required. Figure 5.14 depicts the images that will have their real deadlines in the scheduler queue. It is worth noting that increasing lookahead and interpreting requests' deadlines as real and final deadlines implies that more requests would have to be processed by the scheduler. Furthermore, such an interpretation would require more space at the client's buffer and a lower hit ratio performance would be achieved (the images that will really be used by the client versus the images in the buffer). This could lead to performance degradation; the system would probably perform better if no lookahead and scheduling were used at all. Rather, the client should request the images, the server should process the image and send it back to the client, which would immediately display the image to the user. When a new session is requested, the system will perform an admission test that estimates the scheduler load with one more session. This test takes into consideration the time it took to process the last image requests and whether they were served on-time. We employ a conservative worst-case test in our first attempt to perform admission control. Admission control is not the focus of this research work and it is assumed as our future work.

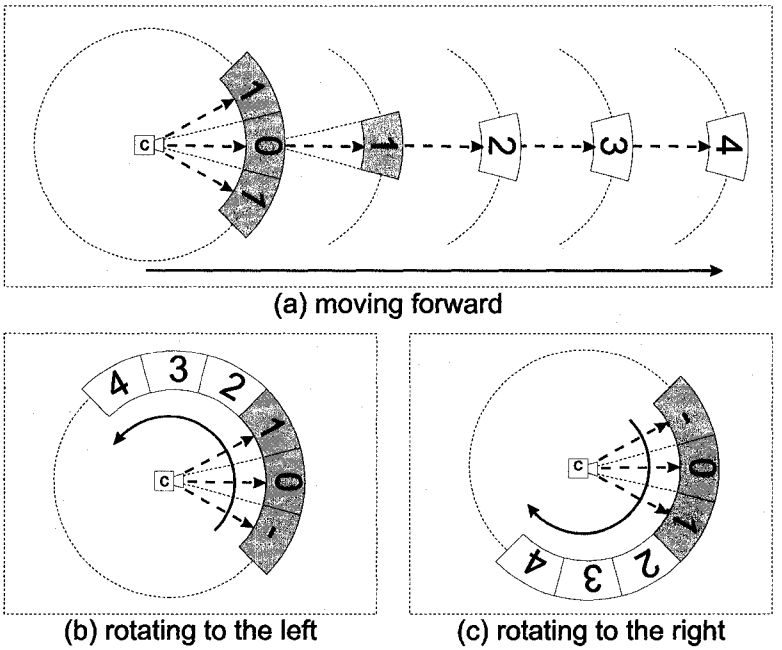


Figure 5.15: Path prediction mechanism.

Path Prediction

A simple path prediction was developed that takes into consideration that the movement of a virtual user tends to follows a sequence of repetitive movements. Sudden changes in this pattern occurs when the user stops briefly to initiate another movement pattern. Figure 5.15 shows the prediction of the future images that will be required by the client, assuming lookahead $LA = 4$. A path prediction can enable the use of higher values for lookahead so that our interactive system can absorb more jitter and provide a smooth navigation within the scenes.

5.7 Performance Evaluation

5.7.1 Simulation Experiments and Results

In order to evaluate and validate the performance of the interactive streaming protocol presented in this section, we measure our schemes through an extensive set of NS-2 [87]

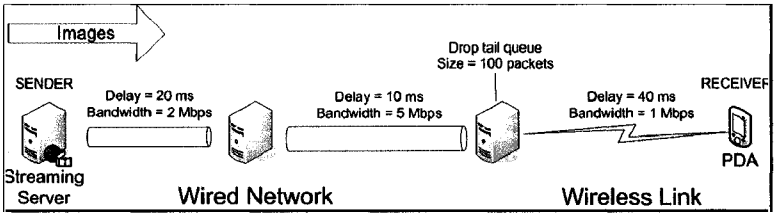


Figure 5.16: Simulation topology

Table 5.1: ISP/IITP simulation settings.

Parameter	**Value**
Simulation time	900 sec
RTTmin	167
Error rate	0.0 - 0.16
Packet Size	1000 bytes
ACK Packet Size	40 bytes
Queue type	Drop tail
Queue limit	100 packets
Image Size	10000 bytes
Initial sending rate	160 Kbps

simulations. The network topology used in the simulations represents a mixed wired-cum-wireless scenario, as depicted in Figure 5.16, with the settings shown in Table 5.1. The wireless link is simulated with an exponential random packet loss model. We stream images of 10KBytes each for 900 seconds in all simulations. The wireless segment is the bottleneck link of the system with a drop-tail queue and a buffer size of 100 packets. We also assume a full 1Mbps wireless bandwidth link capacity. UDP packet size is 1000 bytes and ACK packet is 40 bytes.

We concentrate our simulation experiments on the network aspects of our solution. Thus, simulation results indicating end-to-end throughput for the different protocols as a function of the wireless channel error rate is shown in Figure 5.17(a). We simulate the protocols for 900 seconds and calculate the average end-to-end throughput for error rates varying from 0 to 16 percent. As expected, ISP/IITP maintains the rate of successfully received packets close to the optimal rate because it does not consider all dropped packets to be signs of congestion. As the error rate increases from 0% to 16%, the ISP/IITP

is still capable of keeping bandwidth utilization high. End-to-end results as a function of time with an error rate of 4% and 10% are presented in Figures 5.18(a) and 5.18(b), respectively.

In order to examine the dynamics of our proposed ISP/IITP, we show the frame rate, or the number of images received by the receiver per second, as a function of time and for different error rates in Figure 5.17(b). Frames per second (FPS) is one of the most important metrics for smooth navigation through virtual environments and for movie playback. When simulated with no errors, ISP/IITP achieves optimal frame rate using full bandwidth. When simulating ISP/IITP on a hostile wireless link, the frame rate variation is minimal, floating around 8 frames per second with an error rate of 4%. The frame rate fluctuation is an important metric as it represents the consistency of scene updates. The ISP/IITP protocol shows low rate fluctuation, which is useful for multimedia streaming. It is worth noting the impact that errors have on the achieved frame rate, as depicted in Figure 5.19(a) and 5.19(b). When simulated with an error rate of 8%, the frame rate dropped from 12 to around 5 fps, which represents more than a 50% degradation. Image files are larger than the MTU and must be fragmented into a number of UDP packets. When a UDP packet is lost, the entire image is compromised because the receiver cannot decode it. We plan to use an error-resilience scheme together with forward error protection in our compressed images so as to minimize the impact of channel error on images, but this is the subject of future work.

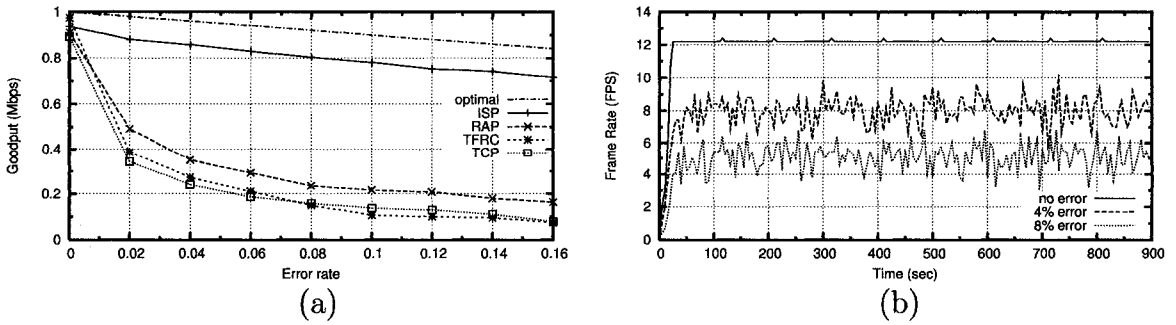


Figure 5.17: (a) End-to-end throughput. (b) Frames per second as a function of time.

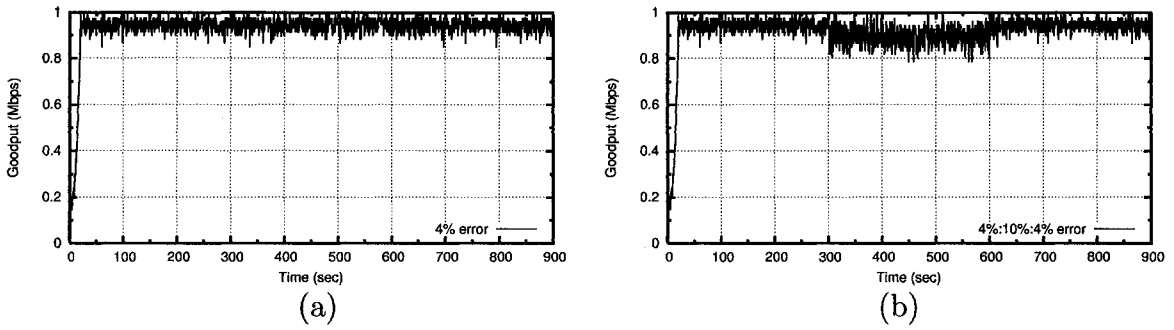


Figure 5.18: End-to-end throughput.

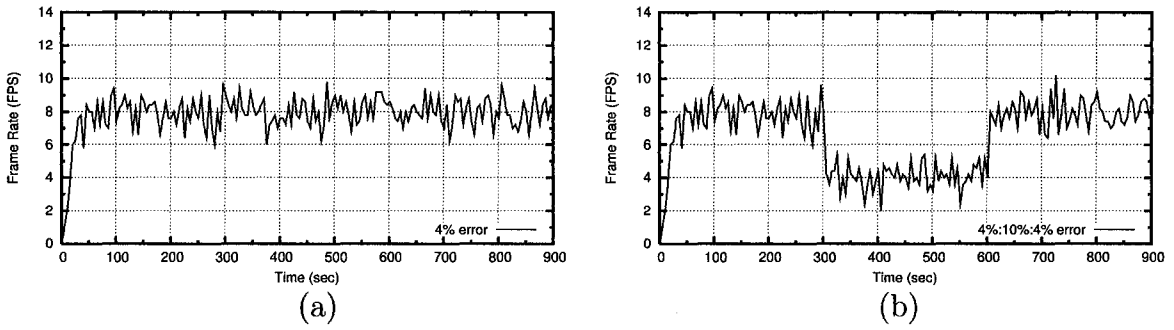


Figure 5.19: Frame rate as a function of time.

As can be seen in Figure 5.20(a), the round trip time is close to the minimum RTT (RTTmin), which is a constant representing the minimum round trip time for a given route, *i.e.*, a route with no queuing delay. For example, on a wireless link with no cross traffic, RTTmin simply corresponds to physical propagation delay. The ISP/IITP's rate control algorithm always checks the RTT to detect any increase in delay and to adapt the rate and avoid congestion. Figure 5.20(b) shows that our solution is fast enough for multimedia applications even in high link error rates.

In order to better understand the behavior of our proposed solution, we conducted a series of simulations that demonstrate an ISP/IITP flow competing with a TCP flow for the bottleneck wireless link. Figures 5.21(a) and 5.21(b) show the ISP/IITP's inter-protocol fairness. First, we run the simulation with TCP flow. At 300 seconds of simulation, an ISP/IITP flow is started. It lasts for 300 seconds. As shown in Figure 5.21(a),

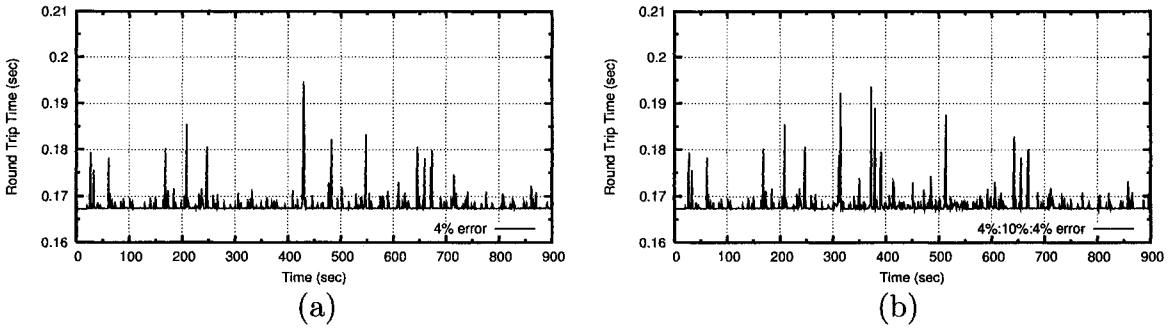


Figure 5.20: Round trip time as a function of time.

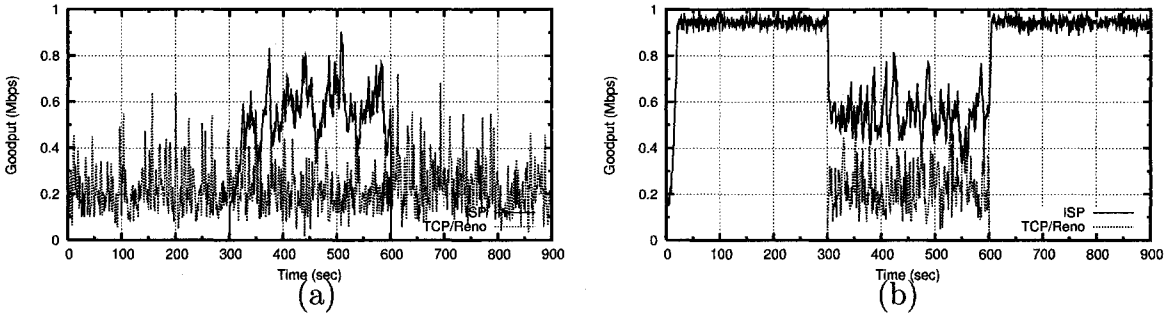


Figure 5.21: (a) ISP and TCP inter-protocol fairness. (b) TCP and ISP fairness.

the ISP/IITP flow does not starve the TCP flow. The same can be concluded from Figure 5.21(b), where an ISP/IITP flow was running when a TCP flow started. Our intention with these two simulations was to show that even when ISP/IITP was utilizing the full bandwidth at the time when TCP began, it shared bandwidth in a friendly manner.

Figure 5.22 highlights the importance of a path prediction mechanism. As we increase the lookahead and guarantee that all images are delivered on time, the number of images that are streamed increases exponentially and the buffer hit ratio drops considerably, as shown in Figure 5.22. However, the path prediction enables the system to employ higher lookahead values without impacting on traffic and hit ratio.

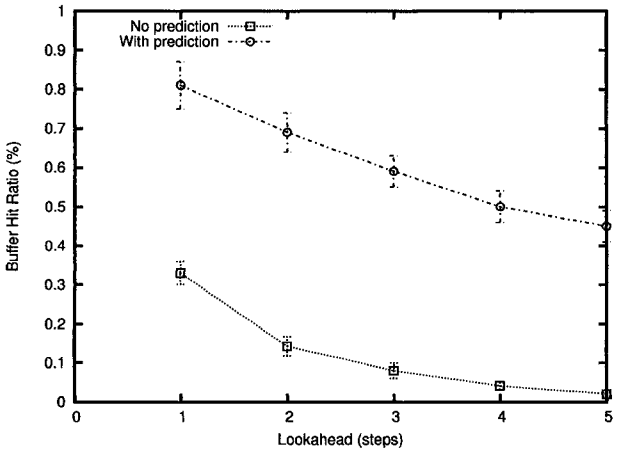


Figure 5.22: Buffer hit ratio.

5.8 Summary

Due to real-time and bandwidth-demanding characteristics, it is extremely challenging to design protocols and systems for streaming multimedia over wireless networks for mobile devices. In this chapter, we focus on the implementation, measurement, and analysis of an interactive image-based streaming strategy. We have proposed a remote interactive multimedia streaming system that can be used for a number of applications. We have proposed end-to-end streaming and rate control mechanisms to support the requirements of bandwidth-demanding multimedia systems. Our rate control scheme achieves high end-to-end throughput and low frame rate fluctuation, which is a critical feature for multimedia systems. Our strategy is based on the adaptation of the sender's transmission rate to the observed received rate. Our buffering solution involves buffering the images closest to the user's position within the image-based scene, therefore providing the client device with the necessary imagery data for rendering future user's viewpoints, removing network jitter, and providing smooth virtual interaction. It is also worth noting that the ISP/IITP is an end-to-end solution that does not require any modification to the network infrastructure. NS-2 experiments were conducted so as to demonstrate the effectiveness of our solutions. Our ISP/IITP protocol shows satisfactory results when

competing for the wireless bandwidth with TCP, and is fair enough not to starve the TCP flow. Our results indicate that our ISP/IITP protocol performed well even when the available bandwidth changed drastically. However, it is clear that streaming multimedia over wireless links must have an error-resilient mechanism in order to protect the most sensitive parts of the images because of the high error rate of the wireless channels.

Chapter 6

Conclusions

6.1 Final Remarks

In recent years, wireless sensor networks have been attracting attention due to their potential applications in several areas. We have observed that a flat sensor network architecture poses serious issues under high network traffic conditions. An extremely high data flow is observed at the nodes surrounding the base station (sink), thereby creating a bottleneck. For unattended low-power sensor networks, this can mean a short lifetime, since the nodes closer to the sink will deplete their energy faster than the other nodes and the network can become disconnected quickly.

Although the cluster-based protocols reduce energy consumption by forming clusters and by aggregating data at the cluster-heads, the bottleneck issue of a single data aggregation point is only postponed. Most data gathering protocols for WSNs use static cluster-heads and sinks. A feasible solution involves the use of mobile entities that collect the data while wandering in the sensor network field. Nonetheless, most techniques that employ mobile sinks found on the literature deal with the problem of collecting data in non-delay-sensitive scenarios. The discoveries and results of this research show that designing a mix of cluster-based and mobile data collector mechanism for wireless sensor networks can meet the challenging requirements of delay-sensitive applications.

While a fully mobile strategy suffices in providing a decentralized data gathering solution and the proper operation in disconnected networks, it does not guarantee the expected performance in terms of latency. Rather than providing a mobile solution in which sensor nodes wait to relay their data until they hear from a nearby mobile data collector, thereby impacting the data delivery latency negatively, this thesis proposes a hybrid routing mechanism that employs both static and mobile data sinks. Furthermore, protocols for wireless sensor networks are in essence application-specific. It is of great importance to define the goals and functionalities of the target application, such that the tradeoffs can be determined and the design of WSNs protocols can be tailored to provide the maximum performance and benefits to the application, while leaving room for generalization that would facilitate the adaptation of the schemes for the different application requirements.

The experimental results confirm that the introduction of mobile data collectors in wireless sensor networks reduces the bottleneck at the nodes closer to the sink and almost halves packet delivery delay. We believe that emergency response applications or other delay-sensitive systems will benefit from the contributions of the proposed MDC/CPEQ scheme.

The feasibility of multimedia sensor networks and we have highlighted that the existing wireless sensor hardware is still far from being able to support high quality multimedia transmission. The multi-tier architecture approach solves this problem partially by defining clearly the role of each tier and assigning more complex tasks to resource-rich devices. We have followed this trend and proposed an interactive image-based streaming solution for multi-tier WMSNs. As far as we are concerned, this is one of the first approaches that consider the streaming of interactive image-based scenes in a wireless sensor network environment.

6.2 Summary of Contributions

We have explored two main issues in wireless multimedia and sensor networks in this thesis: routing and mobile data gathering in wireless sensor networks for delay-sensitive class of applications, and interactive image-based streaming in wireless multimedia networks for thin mobile devices. Hence, we leave our initial contributions and the foundations of a system architecture and its protocols for delay-sensitive applications. Our contributions include: a suite of routing and mobile data gathering protocols for wireless sensor networks; a multi-tier wireless sensor network architecture; and a set of interactive multimedia streaming protocols for WMSNs and wireless networks in general. A list of contributions to the field of wireless multimedia and sensor networks is given below:

(1) We have designed the PEQ protocol: an efficient routing mechanism for flat topologies in wireless sensor networks. The algorithm considers that each sensor node has only local knowledge of the network, *i.e.*, a sensor node knows only its own hop level an the addresses of its immediate neighbors at a lower hop level. This is all a sensor node needs for its routing services. The algorithm is simple enough to be implemented in resource-constrained sensor motes, and fast and reliable to be used for delay-sensitive applications.

(2) A fast path recovery mechanism was proposed for routing in sensor networks. Instead of using a sink-based flooding mechanism, our solution exploits the overhearing technique to determine if a route is available. When a path disruption is detected by a sensor node, it will use another route option learned during the network setup phase. If there is no route available, the node initiates a path recovery by searching for neighboring nodes.

(3) The development of a cluster-based routing solution for wireless sensor networks (CPEQ). The key idea is the use of multi-hop communications for data delivery, instead of direct communications from cluster-heads to base stations, in order to

provide scalability.

(4) A cluster-head selection algorithm that considers the energy remaining at the sensor nodes and that introduces little overhead to the network. The mechanism improves the distribution of energy consumption among the nodes in the network by selecting the nodes with higher residual energy in each round.

(5) We have developed a data gathering mechanism that supports mobility in wireless sensor networks. In the MDC/CPEQ protocol, we consider a hybrid strategy in which mobile and static data collectors gather data from the sensor network. The algorithms involved in this protocol introduce little or no overhead to the network and alleviate the bottleneck problem of a single and static data aggregation point. We have adopted a hybrid approach that reduces latency, unlike other existing solutions for mobile data gathering in sensor networks that use passive mechanisms and are not suitable for delay-sensitive applications.

(6) An interactive streaming protocol (ISP) was devised, in which interactive multimedia is streamed to thin mobile devices, *e.g.*, PDA-class sensor motes like the Stargate platform [111].

(7) We have investigated and developed the Interactive Image-based Transport Protocol (IITP). It works with ISP in order to deliver images to client devices. The IITP implements a simple rate control that uses a loss differentiation algorithm called Spike [20] to discriminate the type of packet loss in order to enhance the bandwidth utilization of the ISP/IITP protocol. The ISP/ITTP protocol is an end-to-end solution that does not require any modifications to the network infrastructure.

(8) We have proposed scheduling and buffering mechanisms for interactive image-based streaming in WMSNs. Since the interactive multimedia considered in this work is non-linear, the ISP/ITTP protocol requires a different scheduling/buffering than the ones for video streaming. The key idea is to assign priorities to the images

according to the position of the viewpoint at the client-end. Thus, the client device will have stored in its buffer the imminent images that will be necessary to render new scenes before requesting them. This solution can absorb jitter and provide a constant and smooth frame rate at the client-end.

(9) We also leave the foundations of a system architecture that can be used for delay-sensitive type of applications over wireless sensor networks. We expect that this can contribute to future applications of augmented reality in the field of surveillance and emergency response.

6.3 Future Research Directions

During the progress of this thesis work, we have identified several future research directions that can add to or enhance the proposed solutions:

(1) The provision of QoS support for the PEQ protocol. A sensor node in PEQ should be able to select the "best" forwarding neighbor node according to some QoS metrics gathered during data delivery. In its current version, PEQ chooses the forwarding neighbor randomly to distribute the load among different paths, which provides low latency, but it is still best effort. To this end, we are investigating the use of different packet or subscription priorities in PEQ. The PEQ protocol uses publish/subscribe and it could provide a certain degree of QoS by, for instance, defining the priority of the interest data to be the criteria in the subscription message. For instance, the update messages of the tracking of a person or object in real-time should have higher priority than the notification that a light is on or a door is open (application-specific).

(2) The design of an ad hoc communication architecture for the mobile data collectors (MDCs) in the MDC/CPEQ protocol. MDCs could communicate among them

in an ad hoc fashion to deliver data in a large-scale sensor network in order to guarantee scalability and low latency simultaneously.

(3) A packet priority scheme for MDC/CPEQ would also increase the performance of our techniques. For instance, when a sensor node has data to deliver to the sink, it could check the priority of its packet and determine whether it should wait until a mobile data collector is nearby, or deliver it immediately to the static route. This way, we could avoid the delivery of low-priority packets through static routes, which is considered costly when compared to the data delivery through a nearby MDC. We plan to extend the MDC/CPEQ protocol in order to support disconnected or partitioned wireless sensor networks. Sensor nodes should be able to cache their data and wait until an MDC is nearby when their are in a partitioned section of the network.

(4) A challenging future work is the study and design of a rate control over multiple wireless hops (ad hoc). If we could make the second tier of the proposed architecture (the tier of mobile data collectors and multimedia client-end) communicate as an ad hoc network, we could come up with a peer-to-peer, decentralized, fault-tolerant interactive multimedia streaming solution for wireless multimedia sensor network. An end-to-end ad hoc rate control would also be used in the lower tier of the architecture, in which the sensor nodes capture images from the environment from different positions and transmit them to the multimedia receivers that would perform the image-based rendering locally, without the need of a rendering server. This solution would leverage our remote image-based navigation system to the exploration of scenes based on real-world images, besides the synthetic ones.

(5) The investigation and design of an error-resilience scheme together with forward error protection for compressed images so as to minimize the impact of channel error on images.

Glossary of Terms

ACK Acknowledgement message

AODV Ad Hoc On-Demand Vector Routing

APTEEN Adaptive PeriodicThreshold-sensitive Energy Efficient sensor Network

BRMM Bounded Random Mobility Model

BSMM Boundless Simulation Area Mobility Model

CENS Center for Embedded and Networked Sensing

CLN Congestion Loss Notification

CPEQ Cluster-based, Periodic, Event-driven, and Query-based Routing Protocol

CSMA Carrier Sense Multiple Access

CSMA/CA Carrier Sense Multiple Access/Collision Avoidance

CSMA/CD Carrier Sense Multiple Access/Collision Detection

DD Directed Diffusion

DSR Dynamic Source Routing

FEC Forward Error Correction

FPS Frames Per Second

GAF Geographic Adaptive Fidelity

GAP Gap-based Adaptation Protocol

GEAR Geographical Energy Aware Routing

GPS Global Position System

GPSR Greedy Perimeter Stateless Routing

IBR Image Based Rendering

ISP Interactive Streaming Protocol

IITP Interactive Image-based Transport Protocol

LDA Loss Differentiation Algorithm

LEACH Low-Energy Adaptive Clustering Hierarchy

MDC Mobile Data Collector

NFL Neighbor Feedback Loop

ns-2 The Network Simulator

OpenGL Open Graphics Library

PDA Personal Digital Assistant

PEQ Periodic, Event-driven, and Query-based Routing Protocol

QoS Quality of Service

RAP Rate Adaptation Protocol

ROTT Relative One-Way Trip Time

RR Rumor Routing

RTP Real-time Transport Protocol

RTCP Real-time Transport Control Protocol

SAR Sequential Assignment Routing

SNGF Stateless Non-deterministic Geographic Forwarding

SPEED A Stateless Protocol for Real-Time Communication

SPIN Sensor Protocols for Information via Negotiation

TCP Transport Control Protocol

TEEN Threshold-sensitive Energy Efficient sensor Network

TDMA Time Division Multiple Access

TFRC TCP-Friendly Rate Control

TTL Time-to-Live

UDP User Datagram Protocol

VANET Vehicular Ad-hoc Network

VNC Virtual Network Computing

WEBS Berkeley's Wireless Embedded Systems

WMSN Wireless Multimedia Sensor Network

WSN Wireless Sensor Network

Bibliography

[1] Aboobaker, N., Chanady, D., Gerla, M., and Sanadidi, M. Y. Streaming Media Congestion Control Using Bandwidth Estimation. In Proceedings of the 5th IFIP/IEEE international Conference on Management of Multimedia Networks and Services: Management of Multimedia on the internet (October 06 - 09, 2002). K. C. Almeroth and M. Z. Hasan, Eds. Lecture Notes In Computer Science, vol. 2496. Springer-Verlag, London, pp. 89-100, 2002

[2] Adelson, E. H. and Bergen, J. R. The plenoptic function and the elements of early vision, Computational Models of Visual Processing, Edited by Michael Landy and J. Anthony Movshon. ISBN 0-262-12155-7. The MIT Press, Cambridge, Mass., Chapter 1, pp. 3-20., 1991.

[3] Agre, J. and Clare, L. "An Integrated Architecture for Cooperative Sensing Networks". IEEE Computer Networks, pp 106-108, May 2000.

[4] Akenine-Moller, T. and Haines, E. "Real-Time Rendering", 2nd edition, A K Peters Ltd, 2002.

[5] Akkaya, K. and Younis, M., A survey on routing protocols for wireless sensor networks. Elsevier Journal of Ad Hoc Networks. Vol. 3, issue 3. 325-349. May 2005.

[6] Akyildiz, I. F. W., Sankarasubramaniam, Su, Y. and E. Cayirci. Wireless sensor networks: a survey. Comput. Networks, vol 38(4), pp 393-422, March 2002.

[7] Akyildiz, I. F., Melodia, T. and Chowdhury, K. R. "A Survey on Wireless Multimedia Sensor Networks," Computer Networks (Elsevier), Vol. 51, no. 4, pp.921-960, March 2007.

[8] Balakrishnan, H., Padmanabhan, V., Seshan, S., Katz, R. A comparison of mechanisms for improving TCP performance over wireless links, ACM SIGCOMM96, pp.256 - 269, 2006.

[9] Bandyopadhyay, S. and Coyle, E. J. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), San Francisco, CA, pp. 1713–1723, March 2003.

[10] Biaz, S. and Nitin H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. Proceedings of the 1999 IEEE Symposium on Application - Specific Systems and Software Engineering and Technology, pages 1017, March 24-27, 1999.

[11] Biermann, H., Hertzmann, A., Meyer, J., Perlin, K., Stateless Remote Environment Navigation with View Compression, NYU Technical Report 1999-784. April 22, 1999.

[12] Boukerche, A. and Nikoletseas, S. Protocols for data propagation in wireless sensor networks. In Wireless Communications Systems and Networks, M. Guizani, Ed. Plenum Press, New York, NY, 23-51, 2004.

[13] Boukerche, A., Chatzigiannakis, I., and Nikoletseas, S. 2005. Power-Efficient Data Propagation Protocols for Wireless Sensor Networks. Simulation 81, 6, pp. 399-411, June 2005.

[14] Brachtl, M., Slajs, J. and Slavk, P. Pda based navigation system for a 3d environment. Computers and Graphics, vol. 25, no. 4, pp. 627634, August 2001.

[15] Braginsky, D. and Estrin, D. Rumor routing algorithm for sensor networks, in: Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, pp. 22-31. September 2002.

[16] Bulusu, N., Estrin, D., Girod, L. and Heidemann, J. Scalable coordination for wireless sensor networks: self-configuring localization systems, International Symposium on Communication Theory and Applications (ISCTA 2001), Ambleside, UK, July 2001.

[17] Campbell, J., Gibbons, P. B., Nath, S., Pillai, P., Seshan, S., and Sukthankar, R. 2005. IrisNet: an internet-scale architecture for multimedia sensors. In Proceedings of the 13th Annual ACM international Conference on Multimedia. Hilton, Singapore,. MULTIMEDIA '05, pp. 81-88. November 06 - 11, 2005.

[18] Cao, Z.-Y., Ji, Z.-Z. and Hu, M.-Z. "An image sensor node for wireless sensor networks". In Proceeding of the International Conference on Information Technology: Coding and Computing (ITCC 2005), vol. 2, pp. 740745, April 2005.

[19] Carnegie Mellon mobile camera array,
2004. http://amp.ece.cmu.edu/projects/mobilecamarray/.

[20] Cen, S., Cosman, P. C. and Voelker, G. M. End-to-end differentiation of congestion and wireless losses. IEEE/ACM Transactions on Networking (TON), 11(5):703-717, October 2003.

[21] UCLA's Center for Embbeded Networked Sensing. http://research.cens.ucla.edu/

[22] Cerpa, A., Elson, J., Hamilton, M. and Zhao J. Habitat monitoring: application driver for wireless communications technology, ACM SIGCOMM'2000, Costa Rica, April 2001.

[23] Chakrabarti, A., Sabharwal, A. and Aazhang, B. Using predictable observer mobility for power efficient design of sensor networks. In The second International Workshop

on Information Processing in Sensor Networks, IPSN 2003, Palo Alto, CA, USA, pp. 129-145, April 22-23, 2003.

[24] Chatzigiannakis, I., Nikoletseas, S. and Spirakis, P. A Comparative Study of Protocols for Efficient Data Propagation in Smart Dust Networks. In Proc. 2nd ACM Workshop on Principles of Mobile Computing - POMC2002 (2002).

[25] Chatzigiannakis, P. Spirakis and S. Nikoletseas, "Efficient and Robust Protocols for Local Detection and Propagation in Smart Dust Networks", in the ACM/Baltzer Mobile Networks and Applications (MONET) Journal, Special Issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks, in MONET 10 (1), pp. 133-149, 2005.

[26] Chen, S. E. QuickTime VR An Image-Based Approach to Virtual Environment Navigation, Computer Graphics (SIGGRAPH95), pp. 29-38, August 1995.

[27] Chen, L., Chen, Z., and Tu, S. 2005. A Realtime Dynamic Traffic Control System Based on Wireless Sensor Network. In Proceedings of the 2005 international Conference on Parallel Processing Workshops (Icppw'05) - Volume 00 (June 14 - 17, 2005). ICPPW. IEEE Computer Society, Washington, DC, pages 258-264.

[28] Choksi, A., Martin, R. P., Nath, B. and Pupala, R. Mobility Support for Diffusion-based Ad-Hoc Sensor Networks. Rutgers University, Department of Computer Science, Technical Report DCS-TR-463, April 2002.

[29] Chou, C., Hsu, M., and Lin, C. 2006. A trend-loss-density-based differential scheme in wired-cum-wireless networks. In Proceedings of the 2006 international Conference on Wireless Communications and Mobile Computing (Vancouver, British Columbia, Canada, July 03 - 06, 2006). IWCMC '06. pp. 239-244.

[30] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," International Journal of High Performance Computing Applications, vol. 6, no. 3, pp. 90–110, 2002.

[31] CMUcam3 Projetc, 2007. Carnegie Mellon University. http://cmucam.org/

[32] CodeBlue Project: Wireless Sensor Networks for Medical Care, Harvard University. Available: http://www.eecs.harvard.edu/ mdw/proj/codeblue/

[33] Dantu, K. et. al. "Robomote: enabling mobility in sensor networks". In Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium, pages: 404- 409, 15 April 2005.

[34] Demircin, M. U. and Beek, P. van. "Bandwidth Estimation and Robust Video Streaming over 802.11e Wireless LANs," in Proc. IEEE Int. Conf. Multimedia and Expo (ICME), pp. 1250-1253, Amsterdam, The Netherlands, July 2005.

[35] Diepstraten, J., Gorke, M., and Ertl, T. 2004. Remote Line Rendering for Mobile Devices. In Proceedings of the Computer Graphics international (Cgi'04) - Volume 00 (June 16 - 19, 2004). CGI. pp 454-461

[36] Dot mote. Berkeley University http://www.jlhlabs.com/LowPowerWireless.htm

[37] Downes, I., Baghaei Rad, L. and Aghajan, H. "Development of a Mote for Wireless Image Sensor Networks". In Proceeding of the COGnitive systems with Interactive Sensors (COGIS 2006), Mar. 2006.

[38] Dust Networks' SmartMesh-XT mote, 2007. http://www.dustnetworks.com/products/m2135.shtml

[39] Dyo, V. Middleware design for integration of sensor network and mobile devices. In Proceedings of the 2nd international Doctoral Symposium on Middleware. Grenoble, France. DSM '05, vol. 114, pp. 1-5, November 28 - December 02, 2005.

[40] Engel, K., Sommer, O. and Ertl, T. A Framework for Interactive Hardware Accelerated Remote 3D-Visualization. In Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym 00, pages 167177, May 2000.

[41] Eugster, P. T., Guerraoui R. and Sventek, J. Distributed Asynchronous Collections: Abstractions for Publish/Subscribe Interaction. Elisa Bertino (Ed.): ECOOP 2000, LNCS 1850, pp. 252-276. Springer-Verlag Berlin Heidelberg (2000).

[42] Eugster, P. T., Felber, P., Guerraoui, R.and Kermarrec, A. "The many faces of publish/subscribe". ACM Comput. Surv. 35(2): 114-131 (2003).

[43] Feng, W., Walpole, J., Feng, W. and Pu, C. Moving towards massively scalable video-based sensor networks. In Proceedings of the Workshop on New Visions for Large-Scale Networks: Research and Applications, pages 12-14, Washington, DC, USA, March 2001.

[44] Feng, W., Code, B., Kaiser, E., Shea, M., Feng, W., and Bavoil, L. 2003. Panoptes: scalable low-power video sensor networking technologies. In Proceedings of the Eleventh ACM international Conference on Multimedia, pages 562-571. Berkeley, CA, USA, November 02 - 08, 2003.

[45] Floyd, S. and Fall, K. "Promoting the use of end-to-end congestion control in the Internet". IEEE/ACM Transactions on Networking, vol. 7(4), pp. 458–472, August 1999.

[46] Floyd, S., Handley, M., Padhye, J., Widmer, J. "Equation-Based Congestion Control for Unicast Applications", Proc. ACM SIGCOMM 2000, pp. 43 - 56, Aug. 2000.

[47] Foote, J. and Kimber, D. FlyCam: practical panoramic video. In Proceedings of the Eighth ACM international Conference on Multimedia (Marina del Rey, California, United States). MULTIMEDIA '00. Pages: 487-488, 2000.

[48] OpenGL Embeded System. http://www.khronos.org/opengles/ 2007

[49] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In Computer Graphics Proceedings, Annual Conference Series, , Proc. SIGGRAPH96 (New Orleans), pages 4354, 2006.

[50] Harte, L., Hoeing, M., Mclaughlin, D. and Kta, R. K. CDMA IS-95 for Cellar and PCS, Mcgraw-Hill, 1999.

[51] He, T., Stankovic, J. A., Lu, C. and Abdelzaher, T. "SPEED: A stateless protocol for real-time communication in sensor networks". In Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS '03), pages 46–55, Washington, DC, USA, 2003.

[52] Heinzelman, W. R., Kulik, J. and Balakrishnan, H. "Adaptive protocols for information dissemination in wireless sensor networks," in Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking, Seattle, WA USA, 1999, pp. 174–185.

[53] Heinzelman, W., Chandrakasan, A. and Balakrishnan., H. Energy-efficient communication protocols for wireless sensor networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS), pages 3005–3014, Big Island, Hawaii, USA, January 4-7 2000.

[54] Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D. and Ganesan, D. Building efficient wireless sensor networks with lowlevel naming. In Proceedings of the Symposium on Operating Systems Principles, pp. 146-159, Banff, Alberta, Canada, Oct. 2001.

[55] Hengstler, S., Prashanth, D., Fong, S. and Aghajan, H. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In IPSN 07: Proceedings of the 6th inter- national conference on Information processing in sensor networks, pages 360369, 2007.

[56] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. "System architecture directions for networked sensors". In Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, pp. 93-104, November 2000, Cambridge, Massachusetts, USA.

[57] Hollar, S. COTS Dust, Master of Science, Thesis, UC Berkeley, December 2000.

[58] Hsiao, H; Chindapol, A.; Ritcey, J.A.; Yaw-Chung Chen; Jenq-Neng Hwang. A new multimedia packet loss classification algorithm for congestion control over wired/wireless channels. Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP apos;05). IEEE International Conference on Volume 2, Issue , 18-23 March 2005 Page(s): ii/1105 - ii/1108 Vol. 2

[59] Humphreys, G., Houston, M., Ng, R., Ahern, S., Frank, R., Kirchner, P. and Klosowski, J. T. Chromium: A Stream Processing Framework for Interactive Graphics on Clusters of Workstations. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002), 21(3):693–702, July 2002.

[60] Intanagonwiwat, C., Govindan, R. and Estrin, D. "Directed diffusion: A scalable and robust communication paradigm for sensor networks, " in Proceedings, Sixth Annual Int. Conf. on Mobile Computing and Networking (MobiCOM '00), Boston, Massachussetts, USA, 2000, pp. 56–67.

[61] Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. 2003. Directed diffusion for wireless sensor networking. IEEE/ACM Trans. Netw. 11, 1, 2-16, Feb. 2003.

[62] Jacobson, V. "Congestion Avoidance Control". In Proceedings of the SIGCOMM'88 Conference on Communications Arquitectures and Protocols, volume 18, pages 314-320, August 1988.

[63] Jea, D., Somasundara, A. A. and Srivastava, M. B. Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks. In DCOSS, pages 244257, 2005.

[64] Johnson, D. B. and Maltz. D. A. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[65] Java Specification Request 184 (2005) - Mobile 3D Graphics API for J2ME. http://www.jcp.org/en/jsr/detail?id=184

[66] Kansal, A., Somasundara, A. A., Jea, D. D., Srivastava, M. B., and Estrin, D. 2004. Intelligent fluid infrastructure for embedded networks. In Proceedings of the 2nd international Conference on Mobile Systems, Applications, and Services - MobiSys '04, pp. 111-124, Boston, MA, USA, June 06 - 09, 2004.

[67] Karp, B. and Kung, H. T. 2000. GPSR: greedy perimeter stateless routing for wireless networks. In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking (Boston, Massachusetts, United States, August 06 - 11, 2000). MobiCom '00, pp. 243-254.

[68] Kinalis, A. and Nikoletseas, S. Scalable Data Collection Protocols for Wireless Sensor Networks with Multiple Mobile Sinks. In Proceedings of the 40th Annual Simulation Symposium . Annual Simulation Symposium. IEEE Computer Society, Washington, DC, pp. 60-72. March 26 - 28, 2007.

[69] Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P., and Scopigno, R. 2004. Protected interactive 3D graphics via remote rendering. ACM Trans. Graph. 23, 3, pp. 695-703, Aug. 2004.

[70] Kulkarni, P., Ganesan, D., and Shenoy, P. 2005. The case for multi–tier camera sensor networks. In Proceedings of the international Workshop on Network and Oper-

ating Systems Support For Digital Audio and Video, NOSSDAV '05. pages 141-146. Stevenson, Washington, USA, June 13 - 14, 2005.

[71] Kulkarni, P., Ganesan, D., Shenoy, P., and Lu, Q. 2005. SensEye: a multi-tier camera sensor network. In Proceedings of the 13th Annual ACM international Conference on Multimedia, MULTIMEDIA '05, pages 229-238. Hilton, Singapore, November 06 - 11, 2005.

[72] Kusmierek, E. and Du, D. H. 2005. Streaming video delivery over internet with adaptive end-to-end QoS. J. Syst. Softw. 75, 3, pp. 237-252, 2005.

[73] Lee, U., Magistretti, E., Zhou, B., Gerla, M., Bellavista, P. and Corradi, A. MobEyes: Smart Mobs for Urban Monitoring with Vehicular Sensor Networks. IEEE Wireless Communications, Vol. 13, No. 5, pages 51-57, Sep. 2006.

[74] Levoy, M. and Hanrahan, P. Light field rendering. In Computer Graphics Proceedings, Annual Conference Series, pages 3142, Proc. SIGGRAPH96 (New Orleans). ACM SIGGRAPH. August 1996.

[75] Li, J., Blake, C., Couto D. S. J., Lee, H. I. and Morris, R. "Capacity of Ad Hoc Wireless Networks". In Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, MOBICOM'01, Rome, Italy, pages 61-69. July, 2001.

[76] Ma, M. and Yang, Y. 2006. Clustering and load balancing in hybrid sensor networks with mobile cluster heads. In Proceedings of the 3rd international Conference on Quality of Service in Heterogeneous Wired/Wireless Networks. Waterloo, Ontario, Canada. QShine '06, vol. 191, paper 16, August 07 - 09, 2006.

[77] Manjeshwar, A. and Agrawal, D.P. "TEEN: A protocol for Enhanced Efficiency in Wireless Sensor Networks," Proceedings of the 1 st Int. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2001.

[78] Manjeshwar, A. and Agrawal, D.P. "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks," Proceedings of the 2nd Int. Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2002.

[79] McMillan, L. An Image-Based Approach to Three-Dimensional Computer Graphics. Ph.D. Dissertation. Technical Report 97-013, University of North Carolina at Chapel Hill, Department of Computer Science, 1997.

[80] MICA mote, Berkeley's http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MIC

[81] MICA2 mote, Berkeley's. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/M

[82] MICA2Dot mote. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2D(

[83] MICAz mote. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datas

[84] Min, R., Bhardwaj, M., Cho, S., Shih, E., Sinha, A., Wang, A. and Chandrakasan, A. P. "Low-Power Wireless Sensor Networks," 14th Interconational Conference on VLSI Design 2001, pp. 205-210, Bangalore, India, January 2001.

[85] Neumann, U., Pintaric, T., and Rizzo, A. Immersive panoramic video. In Proceedings of the Eighth ACM international Conference on Multimedia (Marina del Rey, California, United States). MULTIMEDIA'00. Pages: 493-494, 2000.

[86] Noury, N., Herve, T., Rialle, V., Virone, G., Mercier, E., Morey, G., Moro, A. and Porcheron, T. "Monitoring behavior in home using a smart fall sensor", IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, pp. 607610, October 2000.

[87] ns-2 - The Network Simulator version 2.30. http://www.isi.edu/nsnam/ns/

[88] Nuevo J. and Grgoire. J. Analysis of the Effects of Entity Mobility Models on Ad Hoc Network Communication. International Symposium on Performance Evaluation

of Computer and Telecommunication Systems (SPECTS), Montreal Qc, Canada, July 20-24, 2003.

[89] Open Inventor Toolkit. http://oss.sgi.com/projects/inventor/

[90] Park, M.K.; Sihn, K.-H.; Jun Ho Jeong. A statistical method of packet loss type discrimination in wired-wireless networks. Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE Volume 1, Issue , 8-10 Jan. 2006 Page(s): 458 - 462

[91] Park, J., Lee, U., Oh, S. Y., Gerla, M., and Lun, D. S. 2006. Emergency related video streaming in VANET using network coding. In Proceedings of the 3rd international Workshop on Vehicular Ad Hoc Networks (Los Angeles, CA, USA, September 29 - 29, 2006). VANET '06. ACM, New York, NY, 102-103.

[92] Perkins, C. E. and Royer, E. M. Ad Hoc On-Demand Distance Vector Routing. In Proc. of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), pages 90-100, New Orleans, LA, February 25-26, 1999.

[93] Perkins, C. Ed., Ad Hoc Networking, Addison-Wesley Publishers, Reading, MA, 2000.

[94] PointGrey's Ladybug2^{TM}. http://www.ptgrey.com/products/ladybug2/index.asp

[95] Rahimi, M., Baer, R., Iroezi, O. I., Garcia, J. C., Warrior, J., Estrin, D., Srivastava, M. "Cyclops: In Situ Image Sensing and Interpretation in Wireless Sensor Networks". In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys 2005), Nov. 2005, pp. 192204.

[96] Rejaie, R., Handley, M. and Estrin, D. "RAP: an end-to-end rate-based congestion control mechanism for real time streams in the Internet", in Proc. IEEE INFOCOM, New York, USA, Mar. 1999, pp. 1337–1345.

[97] R. Rejaie, M. Handley, and D. Estrin. Layered Quality Adaptation for Internet Video Streaming. IEEE Journal on Selected Areas of Communications, 18(12), pages 2530-2543, 2000.

[98] Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V. RTP: A transport protocol for real-time applications, 1996. IETF RFC1889.

[99] IETF RFC 2001. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms.

[100] Richardson, T., Stafford-Fraser, Q., Wood, K. R. and Hopper. A. "Virtual Network Computing". IEEE Internet Computing, 2(1):33– 38, January/February 1998.

[101] Rossi, D. "Sensors as Hardware: Motes Evolution", 2001. http://www.telematica.polito.it/wsn/ppt/WSN1_Hardware.pdf

[102] N. Sadagopan, B. Krishnamachari, A. Helmy, "The ACQUIRE Mechanism for Efficient Querying in Sensor Networks", First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA), in conjunction with IEEE ICC, pp. 149-155, Anchorage, 2003.

[103] Salkintzis, A. K. and Passas, N. "Emerging Wireless Multimedia Services and Technologies". John Wiley & Sons Ltd. ISBN 0-470-02149-7, 2005.

[104] Savvides, A., Han, C.-C., Srivastava, M., "Dynamic fine-grained localization in ad-hoc networks of sensors". In the Proceedings of the MobiCom01, Rome, Italy, pp. 166 179, July 2001.

[105] Schwiebert, L., Gupta, S. K., and Weinmann, J. Research challenges in wireless networks of biomedical sensors. In Proceedings of the 7th Annual international Conference on Mobile Computing and Networking (Rome, Italy). MobiCom '01. ACM Press, New York, NY, pp. 151-165, 2001.

[106] SensEye Project. http://sensors.cs.umass.edu/projects/senseye/

[107] Shah, R. C., and Rabaey, J. M. Energy aware routing for low energy ad-hoc sensor networks. In Proceedings of the 3rd IEEE Wireless Communications & Networking Conference, pp. 151-165, 2002.

[108] SmartDust Project, University of California, Berkeley. http://robotics.eecs.berkeley.edu/ pister/SmartDust/

[109] Sohrabi, K., Gao, J., Ailawadhi, V. and Pottie, G. "A Self-organizing Wireless Sensor Network". In Proceeding of the 39^th Annual Allerton Conference on Communication, Control, and Computing, Urbana, Illinois, October 1999.

[110] Stanford multi-camera array, 2004. http://graphics.stanford.edu/projects/array/

[111] Crossbow's Stargate Platform. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Stargate_Datasheet.pdf

[112] Subramanian, L. and Katz, R. H. An architecture for building self-configurable systems. In Proceedings of the 1st ACM international Symposium on Mobile Ad Hoc Networking & Computing. Boston, Massachusetts. International Symposium on Mobile Ad Hoc Networking and Computing, pp. 63-73, 2000.

[113] Sun, T., Chen, L.-J., Han, C.-C., Yang, G. and Gerla, M. Measuring effective capacity of IEEE 802.15.4 beaconless mode, in Proceedings of IEEE Wireless Communications and Networking Conference (WCNC '06), vol. 1, pp. 493498, Las Vegas, Nev, USA, April 2006.

[114] Szeliski, R. and Shum, H.-Y. Creating full view panoramic image mosaics and environment maps. In SIGGRAPH, pages 251258. ACM Press/Addison- Wesley Publishing Co., 1997.

[115] The Telos sensor mote.
http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[116] Thomas, G., Point, G., Bouatouch, K. A client-server approach to image-based rendering on mobile terminals. Technical Report, ISSN 0249-6399, France, January 2005.

[117] Berkeley's TinyOS Operating System. http://webs.cs.berkeley.edu/tos

[118] RFM TR 1000. http://www.rfm.com/products/data/tr1000.pdf, Jan. 2008.

[119] Berkeley's WEBS Project. http://webs.cs.berkeley.edu/

[120] Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for Ad Hoc routing. In Proceedings of the 7th Annual international Conference on Mobile Computing and Networking (Rome, Italy). MobiCom, pp 70-84, 2001.

[121] Yang, L. and Crawfis, R. Rail-track viewer: an image-based virtual walkthrough system. In Proceedings of the Workshop on Virtual Environments 2002. W. Strzlinger and S. Mller, Eds. ACM International Conference Proceeding Series, vol. 23, pages 37-ff. Barcelona, Spain, May 30 - 31, 2002

[122] Yang, G., Sun, T., Gerla, M., Sanadidi, M. Y., and Chen, L. 2006. Smooth and efficient real-time video transport in the presence of wireless errors. ACM Trans. Multimedia Comput. Commun. Appl. 2, 2 (May. 2006), pages 109-126.

[123] Yao, Y. and Gehrke, J. The cougar approach to in-network query processing in sensor networks. SIGMOD Rec., 31(3): pages 9-18, 2002.

[124] Yoon, I. and Neumann, U. Web-based remote rendering with IBRAC (image-based rendering acceleration and compression). Computer Graphics Forum, vol. 19, no. 3, 2000.

[125] Younis, M., Youssef, M. and Arisha, K. "Energy-Aware Routing in Cluster-Based Sensor Networks", in the Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002), Fort Worth, TX, October 2002.

[126] Yu, Y., Govindan, R. and Estrin, D. Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. Technical Report UCLA/CSD-TR-01-0023, Computer Science Department, UCLA, May 2001.