A CYBER-PHYSICAL SYSTEMS APPROACH TO WATER DISTRIBUTION SYSTEM MONITORING

A Dissertation

by

AGUMBE SURESH MAHIMA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2015

Major Subject: Computer Science

ABSTRACT

Water Distribution Systems (WDS) are critical infrastructures of national importance that supply water of desired quality and quantity to consumers. They are prone to damages and attacks such as leaks, breaks, and chemical contamination. Monitoring of WDS for prompt response to such events is of paramount importance. WDS monitoring has been typically performed using static sensors that are strategically placed. These solutions are costly and imprecise. Recently mobile sensors for WDS monitoring has attracted research interest to overcome the shortcomings of static sensors. However, most existing solutions are unrealistic, or disrupt the normal functioning of a WDS. They are also designed to be deployed on-demand, i.e., when the utility manager receives complaints or suspects the presence of a threat.

We propose to solve the problem of WDS monitoring through a Cyber-Physical system (CPS) approach. We envision a Cyber-Physical Water Distribution System (CPWDS) with mobile sensors that are deployed in the CPWDS and move with the flow of water in pipes; mobile sensors communicate with static beacons placed outside the pipes and report sensed data; the flows in the pipes are controlled to ensure that the sensors continuously cover the main pipes of the WDS. We propose algorithms to efficiently monitor the WDS with limited number of devices, protocols to efficiently communicate among the devices, and mechanisms to control the flows in the WDS such that consumer demands are met while sensors continuously move around. We evaluate our algorithms, protocols, and design of communication, computation and control components of the CPWDS through a simulator developed specifically to model the movement of sensors through the pipes of the WDS. Our simulations indicate that investing on improving the sensing range of mobile sensors reduces the cost of monitoring significantly. Additionally, the placement of beacons, and the communication range impact the accuracy of localization and estimation of sensor locations. Our flow control system is observed to converge and improve the coverage over time.

DEDICATION

To My Beloved and Supportive Parents

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Radu Stoleru, for his support, tremendous encouragement, and guidance. He has been a constant source of inspiration, and has provided valuable advice in my research and career. His enthusiasm towards research is contagious. Even through times of doubt and confusion, his valuable support has been constant. His motivation was instrumental in completing this dissertation.

I would like to thank my collaborators Dr. Amin Rasekh, Dr. Avi Ostfeld, Dr. Basem Shihada, Dr. Emily Zechman Berglund, Dr. Mohan Kumar, Dr. Prabir Barooah, Dr. Lidia Smith, Anjana G.R., Usha Manohar, Wei Zhang, and Weijiao Gong. I also thank my committee members, Dr. Jennifer Welch, Dr. Anxiao (Andrew) Jiang, and Dr. Kelly Brumbelow for their valuable comments and advice throughout. Their feedback and inputs to the research has shaped this dissertation.

My thanks also goes to my colleagues, past and present members of the LENSS lab, especially Dr. Harsha Chenji, Dr. Amin Hassanzadeh, Dr. Myounggyu Won, Wei Zhang, Weijiao Gong, Jay Chen, Chen Yang, for providing valuable inputs through discussions and collaborations.

My special thanks to my parents for all their sacrifices, support, and encouragement. I am truly grateful to all my friends for their support and all the fun filled memories. My remarkable experience at College Station will always remain with me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Water distribution systems (WDSs) are identified as a critical infrastructure system in the Public Health, Security, and Bioterrorism Preparedness and Response Act [74]. In the U.S.A, 90% of the population receives drinking water from nearly 170,000 public WDSs [18]. These buried WDS pipelines cover close to a million miles [90] [6], and many of them date back to the early 20^{th} century. Repairing and upgrading these pipeline systems is not a trivial task. American Water Works Association's report [7] has estimated the repair to incur $250 billion over the next 30 years.

WDSs are vulnerable to a variety of physical and chemical attacks, such as leaks, breaks, backflow events, and chemical contamination. Ensuring the supply of water of a desired quality and quantity is essential to industrial growth and public health. Chemical contamination of water is a great public health risk [91], and may be introduced into the WDS at the sources or in any of the pipes of the WDS, by malicious agents or accidentally. Also, structural damages such as breaks in water pipes cause threats due to leakages (leading to water loss and service disruption), and backflows events (leading to public health risks). There have been approximately 237,600 water main breaks per year in the U.S.A., causing nearly $2.8 billion worth of loss [92]. To protect the consumers from health risks, and to prevent losses, there is a need to install sensor systems to identify the *presence* and *location* of these threats, referred to as *events* henceforth.

Following the 9/11 attacks in U.S.A., contaminant monitoring in WDSs was considered to be of prime importance due to impact it has on the population. As a result, a design competition called the Battle to Water Sensor Networks (BWSN) was de-

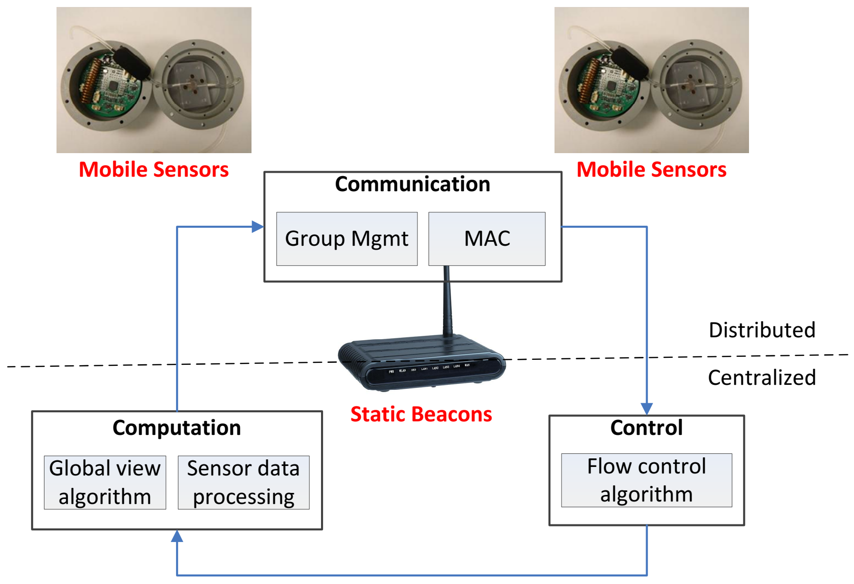


Figure 1.1: Pictorial representation of the CPWDS

signed to encourage research in this area. The competition resulted in a project [50] that used static sensors to detect the presence of contamination in WDSs. Static sensor systems were then tested through real world deployments in cities and municipalities, as well as in laboratory settings [75] [99], not only for contamination detection, but also for leak/break detection. Efficient sensor placement to collect the location and time at which contaminants are introduced into the WDS were also studied [104]. Further research studied the strategies to flush out contaminants [103] by opening hydrants.

Recent advancements in wireless sensor networks (WSN) pave way for the development of mobile sensors traversing water pipelines [64](depicted in Figure 1.2), [47] [42] [50] [75] [42] [46]. Also recently, Pure Technologies has developed a leak detection equipment called SmartBall [67] that is currently in use in municipalities. The SmartBall is inserted in water pipes, and is collected at a fixed location by ensuring that it follows a predetermined path. Leaks are detected using acoustic signals.

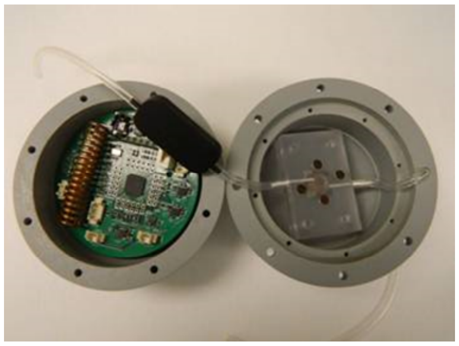
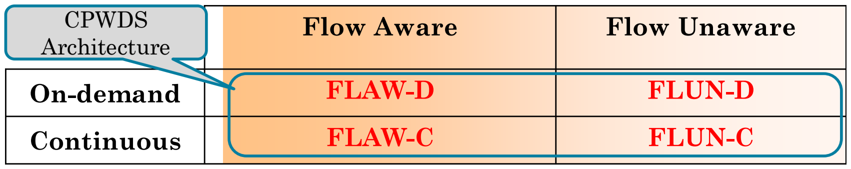


Figure 1.2: TAMU/Purdue University sensor for chemical monitoring in WDS [8]

Such systems anchor our research into reality. However, most mobile sensor systems are not suitable or efficient for deployment in WDS. For example, [47] and SmartBall are predicated on the requirement that sensors follow a predetermined path during the deployment, which is achieved by blocking some regions of the WDS, thereby disrupting the normal function of the WDS. [42] uses RFID tags set up at regular intervals inside the pipelines, which is impractical. [62] studies the use of mobile sensors and static sensors, where all mobile sensors are assumed to follow the path of the highest flow, which the authors admit, is inaccurate. To sum up, existing solutions make assumptions that disrupt the normal functioning of a WDS, are impractical, or are inefficient.



CPWDS Architecture	**Flow Aware**	**Flow Unaware**
On-demand	**FLAW-D**	**FLUN-D**
Continuous	**FLAW-C**	**FLUN-C**

Figure 1.3: Nomenclature and classification of CPWDS

1.1 Motivation

To overcome the limitations of current solutions and to provide a cost effective way to monitor a WDS, we present the design of a Cyber Physical System approach to monitor a WDS, i.e., a Cyber-Physical Water Distribution System (CPWDS), as depicted in Figure 1.3. Our CPWDS consists of two kinds of devices, namely, mobile sensors and static beacons. Mobile sensors traverse through the pipelines aided by the flow of water, sense the environment, communicate among each other, and report data to beacons. Beacons on the other hand are static, aid in localization, and are responsible for data collection and providing a global view to the sensors.

We envision two modes of WDS monitoring - on-demand and continuous. Both these modes are essential and fulfill different objectives. On-demand monitoring is required when the utility manager or a static sensor network suspects the presence of a leak/backflow or contamination in the WDS, whereas continuous monitoring ensures long term monitoring of the WDS. In our CPWDS, since the sensors move freely, their movement is dependent on the flow of water in the pipes, the knowledge of flows in a WDS determines the difficulty in monitoring the WDS. As summarized in Figure 1.3, there are four different approaches to the CPWDS problem - FLAW-D (Flow Aware On-demand), FLUN-D (Flow Unware On-demand), FLAW-C (Flow Aware Continuous), FLUN-C (Flow Unaware Continuous). Additionally, a CPS is composed of three components: Communication, Computation, and Control. A CPWDS poses several interesting questions that are not answered by any related work to the best of our knowledge. For each component, we present the motivation for all four approaches to WDS monitoring through CPWDS approach.

- **Communication:** The sensors in our CPWDS are underwater and mobile. As they move through the pipes, they sense and record data. This data needs to reach devices outside the pipeline at the earliest possible opportunity. Therefore, there needs to be wireless communication from sensors to beacons. Some sensors may not traverse paths containing beacons. Retrieving information from such sensors without sensor-sensor communication is time consuming both in flow-aware and flow-unaware monitoring constructs. In a WDS, the medium of communication among sensors is chosen to be acoustic due to the short communication range of RF underwater. Underwater acoustic communication is challenging due to the high propagation delays of the acoustic medium [84] [44] [52]. A MAC protocol that is able to work in such environments is challenging to design. In case of on-demand monitoring, sensors are usually deployed at once to add redundancy (since the movement of sensors in pipes is random). The movement of sensors in high density groups with changing communication topologies makes sensor-sensor communication more challenging.

- **Computation:** A key advantage of the CPWDS is that the devices are much cheaper than static sensors. There is a need to reduce ineffective deployments so as to improve monitoring. Optimal (i.e., low cost) deployment of sensors and placement of beacons is therefore an important problem to WDS monitoring, especially in on-demand monitoring. In FLAW-D approach, ensuring that there is a significant amount of sensor-sensor communication among sensors traversing different paths improves the event localization. Therefore, we need to also decide when to insert the sensors is also important. Sensors traversing the pipes of a WDS are unaware of their locations. On the other hand, beacons placed outside the WDS have a global view of the WDS. When sensors come in contact with a beacon, the information from the beacons is useful to a sensor to tune its communication parameters. It is, however, challenging to provide useful information while reducing redundancy. Especially in continuous monitoring, such information may prevent a delay in obtaining data from sensors.

- **Control:** During continuous monitoring of a WDS, owing to the varying demands of the consumers and valve actions, the flows in the pipes change in magnitude and direction over time. The main pipes of a WDS are usually of larger diameter than the pipes leading to consumers. Therefore, if the sensors are designed to have a large form factor, they move around in the WDS without getting stuck in small pipes. However, they may travel to parts of the WDS where flows do not change directions over time. In order to effectively monitor a WDS with mobile sensors, sensors need to traverse the main pipes continuously. It is challenging to determine the pipes on which flows need to be reversed, and actually reversing the flows by controlling valves, pumps, and demand points.

1.2 Dissertation Statement

We propose that a Cyber-Physical System composed of mobile sensors that move through the pipes (aided by the inherent flow of water), static beacons (that collect data from sensors and aid in locating the events) and water flow/sensor control subsystems, is an efficient and practical solution to the problem of WDS monitoring. Contrary to the state of the art in WDS monitoring, a CPS approach provides several advantages, as follows. First, due to the use of mobile sensors, accuracy of event detection and identifying the location of the event improves over the use static sensors. Next, communication between sensors and among sensors and beacons improves the delay in obtaining information about the WDS. Further, continuous monitoring voids the need to collect and redeploy sensors.

1.3 Main Contributions

Our contributions are classified into three categories based on the three components of a CPS: Communication, Computation, and Control. We also have contributions in terms of system design and evaluation. Our contributions are elaborated as follows:

- **Communication** [82] [78]: We propose a communication architecture that enables the design of physical, MAC, routing, and higher layers independent of each other. We present a MAC/group communication protocol that accounts for high propagation delays in the acoustic medium in which sensors communicate, and adapts to changing topologies.

- **Computation** [80] [81] [82] [79] [78]: We present algorithms for optimal deployment of sensors, optimal placement of beacons, and algorithms run on beacons to provide information to sensors. For on-demand monitoring, optimal deployment of sensors involves either minimizing the number of sensors to achieve a desired level of coverage, or to maximize the coverage when a fixed number of sensors are available. Similarly, optimal beacon placement involves either minimizing the number of beacons to achieve a certain level of accuracy, or to get the best accuracy with a given number of beacons. We also solve the problem of maximizing the accuracy of event detection and localization when we have a fixed budget that may be allocated to either sensors or beacons. In the context of continuous monitoring, we develop a global view algorithm that predicts the way groups of sensors move, and provides this information to sensors in a small packet of information.

- **Control** [78]: Computational algorithms assume that we have knowledge of the flows in the WDS, i.e., the case of flow-aware monitoring. For flow-unaware monitoring, we propose a flow learning algorithm that uses feedback from sensors to learn the directions, and magnitude of flows in both on-demand and continuous monitoring. For continuous monitoring, we propose a control mechanism that ensures a desired magnitude and direction of flows in each pipe of the WDS so as to ensure coverage of a region of interest. We propose to derive the desired flows such that the probability of sensors reaching a region of interest is maximized. We propose an algorithm to estimate the positions of sensors using information collected at beacons.

- **System Design** [82] [78]: We design the interaction and behavior of all components of the CPWDS including sensing, communication, control, and computation components through a software architecture, a control system design, and a high level model of the entire system.

- **Evaluation** [80] [81] [82] [79] [78]: To evaluate our design, we developed a simulator called CPWDSim to simulate the movement of sensors through pipes of the WDS, communication among sensors, and communication between sensors and beacons. CPWDSim simulator accepts input about a WDS from a well known WDS hydraulics simulator, EPANET [27]. EPANET generates the flows in the pipes of a WDS. Using these, CPWDSim simulates the movements of the sensors. We also simulate the communication among the sensors using our proposed group management/MAC protocol. The simulator is also equipped with all the algorithms, including the control system design. CPWDSim is a first step in CPWDS simulations, and further research is required in improving its fidelity. The implementation of CPWDSim revealed the complexity of the WDS environment for mobile sensor movement and communication. It is not trivial to extend the mobility model of the sensors, and the harsh acoustic environment for sensor-sensor communication in existing simulators. Hence, there is currently no other simulator that may be used for the purposes of this research.

1.4 Organization

This dissertation is organized into eight sections. This section, i.e., Section 1 motivates the problem of WDS monitoring and the importance of a CPS approach to WDS monitoring. The dissertation statement and our contributions are also presented in this section. Section 2 contains the state of the art related to CPWDS, i.e., mobile wireless sensor networks for WDS monitoring, underwater acoustic MAC protocols, group communication protocols, and control systems for flow control in WDS. Section 3 provides the preliminary background and some initial results that motivate our CPWDS. Contributions in each of the components of the CPWDS are then presented in Sections 4-6. Section 4 discusses the design of communication, section 5 tackles the computational aspects, and section 6 explores the control aspects of the CPWDS. Our evaluation of the proposed design in real world implementation and simulation are covered in Section 7. Finally, Section 8 concludes this research and explores future directions.

2. STATE OF THE ART

In this section, we present the state of the art and related work in all the components of the CPWDS. We classify this section into seven sections. First, we present the history of WDS monitoring, including specific related work in contamination and leak detection. Next, background in communication relevant to our study, i.e., MAC protocols for underwater communication, and group communication protocols, are discussed. We then elaborate on existing research in the computation questions tackled in this work. Finally, we present existing control mechanisms in WDS flow control.

2.1 History of WDS Monitoring

Following the 9/11 event in the U.S.A, a competition called the Battle of the Water Sensor Networks (BWSN) design competition was undertaken [60] so as to develop infrastructures to identify chemical attacks on the WDS. As a result of the BWSN competition, a project [50] proposes the placement of static sensors for contaminant monitoring. Consequently, Pipenet [75] implemented and tested the static sensor solution in Boston. This led to several studies to optimize the placement of static sensors due to their high cost [104] [103]. WaterWise [99] also studies the use of sensors equipped with GPS devices for the monitoring of water pipelines in Singapore.

Since static sensor systems were costly and imprecise, and with the advent of wireless sensor network technologies [32] [35] [85], mobile sensor systems for WDS monitoring gathered research interest [42] [47] [46]. TriopusNet [47] attempts to provide a solution for autonomous continuous monitoring of pipelines. The sensors move through the pipes in predetermined paths, and attach themselves to the pipes of the WDS at the appropriate locations. When sensors need to be replaced, they detach from the pipe and get flushed out through the faucets. However, this solution is impractical, since flows in pipes cannot be restricted to follow one path while supplying water to consumers. SPAMMS [42] also proposes to use mobile sensors for pipeline monitoring. The pipelines are set up with RFID tags at regular distances for localization of the mobile sensors. The mobile sensors are retrieved at the sinks to determine where the event is present. A mobile robot is then sent in to counter the event. However, the placement of RFID tags at regular intervals is impractical. MISE-PIPE [77] is another similar system that uses magnetic induction to communicate with sensors moving through underground pipes. This paper demonstrated that it is feasible to have sensors in buried pipelines communicate with access points on the ground.

Specific to leak detection, there are several methods based on various operating principles for detection, localization, and pinpointing of leakages in municipal water distribution systems. Water audits based on metering and water balance calculations can be performed to quantify water losses and provide an extremely crude approximation of the location of losses. A better estimation is achieved through step-testing method whereby valves are systematically closed to subdivide the area and localize the leakage.

A comparatively more recent leak localization method is acoustic logging, which is performed using hydrophones or vibration sensors [38]. Ground penetrating radar is employed to localize the leaks by virtue of detecting underground voids caused by leakage water flow in the immediate vicinity of pipes. More accurate leakage localization, which is also referred to as leakage pinpointing, may be achieved using leak noise correlation, tracer gas, and pig-mounted acoustic techniques. Detailed description and comparison of these well-known methods for detection and pinpointing of

Table 2.1: Existing technologies that use mobile sensors for WDS leak/backflow detection

Authors / Company	**Name**	**Capabilities**	**Sensing technology**	**Free-flowing or line tethered**
Pure Technologies	Sahara [66]	Detecting leaks, pockets of trapped gas, and visual inspection	Hydrophone; camera	Line tethered
Pure Technologies	SmartBall [67]	Detecting leaks, pockets of trapped gas, and structural defects	Acoustic emitter and receiver	Free-flowing
Pure Technologies	PipeDiver [65]	Detecting leaks, pockets of trapped gas, and structural defects	Acoustic emitter and receiver	Free-flowing
Lai et al.	PipeProbe [46]	Mapping hidden pipeline	Metering pressure and angular velocity	Free-flowing
Trinchero et al.	[87]	Detecting leakage. Includes wireless transmission system	Hydrophone	Free-flowing
Chatzigeorgiou	[15]	Detecting leakage	Hydrophone	Free-flowing
Purdue-TAMU sensor	[8]	Measuring water quality parameters; can be used for backflow detection. Includes energy harvest and wireless transmission systems	Ion-selective electrode-based biochip	Free-flowing
MIT MRL Lab	PipeGuard [14]	Detecting leakage. Can be potentially used for backflow detection	Measuring pressure	Free-flowing
Perelman et. al.	[63]	Monitoring water quality. Uses both mobile and static sensors	Water quality sensors	Free-flowing

leaks may be found in [92] [68] [15].

The application of inline, mobile sensors technology for leakage pinpointing has attracted a lot of attention by both researchers and practitioners during the recent years [67] [15] [46] [8]. These are small monitoring devices that traverse freely with the water flow inside the pipes. The sensors can collect measurements from the environment and also process them and transmit the processed data to access points placed outside the pipelines. They have achieved popularity recently thanks to their unique abilities in collecting spatially high resolution data. Simple acoustic sensors are already being employed in real WDSs for leak detection [66] [12] and new sensors are under test and evaluation for monitoring water quality parameters in near-realtime scales [8]. Once contaminants are identified, the response to such events to minimize public health risks are recently being studied [69]. Such studies shed more light on contaminant propagation and sensing modeling.

Deployment of mobile sensors may happen on-demand by a utility manager as part of a periodic system monitoring, or it may be triggered when the presence of a leak/backflow is suspected with the help of a static sensor system or by complaints from consumers. They have been already applied to water utilities of several cities around the world, including Dallas, Montreal, and Manila [66]. Their increasing popularity is presumably due to their ability to pinpoint the leaks more accurately than other existing methods without causing any disruption to regular water utility service. Table 2.1 contains most of the existing practical solutions that use mobile sensors for leak/backflow detection in WDSs.

Although inline, mobile sensors for locating leakages have been already designed and fabricated, decision support models to facilitate and enhance their operation through simulation of their movement in the pipelines network and optimization of their application is still underdeveloped. Development of such computational models is a major focus of this study.

A recent work [63] examines the deployment of mobile water quality sensors along with static water quality sensors for contamination detection. The paper demonstrates the improvement in detection rate provided by mobile sensors in conjunction with static sensors. However, the movement of the sensor is modeled to move through a deterministic path, provided the flow velocities in the pipes are known. The model's use is also limited to contamination event detection and is not developed to enable source localization as well.

In this dissertation, we study varying sensing ranges, including leak/backflow detection, which requires a more stringent sensing model where the mobile sensors are required to move close to the leak points to enable detection and localization. In reality, mobile sensors move randomly through the pipes with a probability distribution at each junction which has not been yet considered in existing solutions in the literature. The sensor mobility model in this work is more general to accommodate the random movement of sensors at junctions. This way, the sensors mobility may be modeled more accurately and thus be deployed more efficiently and effectively.

2.2 Related Work in Communication

The area of in-pipe acoustic systems including the digital communication aspects has been explored by [43]. However, the dynamics of multi-path communication in complex pipe topologies are not considered. In-pipe wave propagation models are discussed in [89]. Underwater RF communication research is done only in open seas and oceans [16] [39] [56]. [88] discusses a methodology wherein sensors that are moving around in pipe conduits can communicate with a surface receiver using microwave communication.

For open sea applications, several simulators are developed and used, such as

Aqua-Sim, MIRACLE, etc. [72], but the models used are insufficient for in-pipe acoustics.

MAC protocols for underwater acoustic networks are widely explored, especially in oceans and seas. MAC protocols for underwater acoustic sensor networks are mainly based on CDMA or TDMA [61]. CDMA is a preferred protocol in mobile sensor networks, because the uncertainty in propagation delay does not affect the performance of the algorithm [98] [28]. TDMA is also efficient in scenarios where delays of a few seconds can be tolerated. Techniques for time synchronization are also prevalent [83]. However, very few protocols consider in-pipe systems where the network topology is dynamic and has a relatively low communication range ($\sim$10-100m) [84] [44] [17].

T-Lohi [84] is a contention based MAC protocol that uses low power tones for reservation. It is designed to be energy-efficient, and is suitable for randomly changing topologies and for sparse short range networks with relatively low data rates. When node density is high, T-Lohi spends a lot of time in contention, which adds delay. Applying T-Lohi directly to a CPWDS is inefficient since the data rate and network size are expected to be high. We have demonstrated this to be true in our evaluations.

Hybrid MAC [44] is designed for fixed topologies, where a period of random access is preceded by a slotted TDMA access. An idea from Hybrid MAC that we adopt in this work is that during the slotted access time period, each time slot is as long as the time for a packet to reach the farthest node. Another protocol that uses contention based methods is Ordered-CSMA [17] where channel access occurs in a predetermined order. [52] discusses the uncertainty at receiving the message when two transmitters are separated in space and time using a conflict graph with respect to space and time.

Table 2.2: MAC protocols for underwater acoustic communication

	Protocol	High PD	Changing topology	Objective achieved
UW-MAC [98]	CDMA to form clusters, TDMA among clusters	✓		Energy efficiency
T Lohi [84]	Contention based, random backoff	✓	✓	Energy efficiency and channel utilization
Secon [52]	Generation of conflict graph	✓		Handling spatio-temporal uncertainty
Hybrid MAC [44]	Time slots to exchange control packets. Unslotted for data packets.	✓		Energy efficiency and low latency
Z-MAC [70]	CSMA and TDMA. Slot used based on contention and allocation.		✓	Reduces collisions

Due to complex nature of underwater acoustic communication, traditional communication models are usually insufficient, and real implementations are difficult. Only recently, there have been efforts to understand the underwater acoustic communication characteristics through testbed implementations [3] [22] [34] [1].

Table 2.2 provides a description of the protocols reviewed here and indicates whether the protocols can be extended to, or already support, mobility, changing topology, and high propagation delay (PD) environments like acoustic.

In the area of wireless adhoc networks, group communication refers to assigning each node to a particular group such that members of the same group can communicate reliably [24] [21]. Distributed algorithms for group management operations, such as leader election for networks with variable message delays (asynchronous) using link reversal have been studied [5] [41] [40]. Here, directions are assigned to bidirectional links to represent leadership. These algorithms, however, are effective when there are very little topology changes, since a number of messages need to be exchanged before the algorithm terminates. Also, in [5], the nodes are required to know all the communication links in the network.

For mobile ad-hoc networks, [94] [73] provide leader election algorithms. [94] is based on growing and shrinking spanning trees based on the Dijkstra-Scholten termination detection algorithm. However, collisions are not considered. The method proposed in [73] is to use heartbeat messages to check if a leader is alive, similar to the idea that we propose. The protocol also uses the concept of a vice-coordinator to take over when the leader fails. This is very similar to the idea we propose of group splitting. The paper however fails to consider merging groups. A recent study on leader election [71] accounts for collision. The nodes in this network send packets with some probability. The nodes progressively tune this value. There are also time slots reserved for leaders to transmit. Leaders and followers follow different algorithms. This paper looks at only single-hop networks. Several distributed algorithms for self-organization [105], for distributed event detection, localization, and tracking [96] [2] have also been studied in the past. However, all these protocols work best in the terrestrial networks with low, bounded, predictable delays without spatial uncertainty.

2.3 Related Work in Computation

Sensor mobility in a WDS resembles a delay tolerant networking (DTN) scenario. The problem of event localization in DTN, however, has received little attention, primarily because it is done using GPS. DTN typically involves vehicles, for which energy is not an issue. Consequently, solutions to problems similar to the event localization problem addressed in this work have not been proposed. For completeness, we review a set of representative DTN research. Data Dolphin [53] uses DTN with fixed sinks and mobile sensors in 2D area. A set of mobile sinks move around in the area. Whenever a sink is close to a sensor, it exchanges information over one hop, thereby reducing the overhead of communicating multi hop and saving energy on the static sensors. A survey done by Lee et al. encompasses the state of art in vehicular networks using DTN [48]. Sensing coverage problems in DTN are handled by CarTel [37], MobEyes [49] etc. These systems use vehicles that can communicate with each other and localization is based on GPS. [97] uses the idea of heterogeneous system with mobile sensors that are less capable than stationary sensors to develop data delivery schemes.

Coverage problems in sensor networks have been considered before [86] [55]. These papers consider coverage problems in 2D or 3D area, unlike coverage on graphs, as in this dissertation. Sensing coverage, in general, has been studied under different assumptions. [13] uses both a greedy approach, and linear programming to approximate the set covering problem. These problems consider only minimizing the number of vertices to cover edges in a graph. Alireza et al. [86] discusses the detection a hole that is not covered by the sensors. The sensors have no location information. Laplacian method is used here to obtain the area in a plane that is not covered. Stephen et al. [55] propose a method to solve the coverage problem using Voronoi diagrams. The sensors are free to move in a 2D space. [106] uses mobility prediction to solve the problem of localization. Vieira et al. [95] propose locating mobile sinks underwater.

[102] performs a mathematical analysis of complex flow-based systems using graph-theoretic concepts. The paper presents several metrics and mechanisms to study the vulnerabilities of flow-based systems. [51] uses robots to monitor pipes. The solution is based on the gallery guarding problem that ensures that every point in a pipe is monitored. The robots attach themselves to the pipes and move about when required. Usually sensing coverage in sensor networks is done so as to ensure

k-coverage. A similar problem for flow-based systems is presented in [101], while ignoring the mobility of sensors. The coverage requirement of our CPWDS is to ensure that mobile sensors cover the WDS over time. Therefore, none of these solutions are suitable.

2.4 Related Work in Control

The concept of control of hydro-systems was initially used on irrigation canals and later on WDS. Controllers can be used for water level control, flow control, pump speed control, etc. A control system basically compares the measured value of the flow with the target value to obtain an error. There are mainly two types of controllers: i) linear controllers, such as PID, PI, PD etc.; and ii) nonlinear controllers such as Dynamic Inversion controller (DI).

In process industries, PID controllers are used for process control like pressure control and temperature control since the late 1940's [59]. DI controllers are well known methods for nonlinear controls, carrying out feedback linearization on non-linear systems. A comparative study was carried out for the different controllers in water distribution network and it was found out that nonlinear controllers perform better than linear controllers in achieving the target flows [45].

Nonlinear DI controller with PID features was used to achieve the equitable distribution of water in Bangalore water inflow systems [54]. Simultaneous proportional integrated derivative (SIM-PID) and normalized proportional integrated derivative (NOM-PID) were proposed to regulate flows and maintain water quality in water distribution systems and two case studies were carried out for the same. Several studies were performed for different controller tunings [45]. Controllers can also be applied for qualitative control on water networks [25]. In [25], flow control was achieved by valve throttling. The flow control systems in [54] and [25] are suitable for our

CPWDS model, but need to be integrated appropriately. We need to determine the reference points for the flow controller appropriately, and also implement it in our simulator to evaluate the control system.

Control of Cyber-Physical Systems and Water Distribution Systems have being studied using Model Predictive Control (MPC) [57] [58]. In [57], the authors study a MPC method to decide taxi dispatch such that supply-demand ratio is maintained, and idle driving times are reduced. The MPC makes a decision based on a prediction of the future. Applying model predictive control to our CPWDS is difficult because the movement of sensors is a complex, non-linear, discontinuous function.

3. PROPOSED ARCHITECTURE AND PRELIMINARIES*

In this section, we present the preliminaries for the dissertation. First, we present a software architecture of the proposed CPWDS. This architecture encompasses the functionalities of the mobile sensors and static beacons. We also define some terms that are used throughout this thesis. More terms specific to each component are presented in the later sections. Next, we describe a general sensing model and the mobility model.

There are three components in a CPWDS, as shown in Figure 1.1 - computation, communication, and control. The communication component describes the communication among sensors and between sensors and beacons. The result of communication acts as input to the control system design. The control system then provides input to the computation aspects, i.e., the algorithms for sensor/beacon placement, global view etc. The beacons act as a bridge between the distributed and centralized components of the CPWDS. The objective of this research is to design and analyze all three aspects of the CPWDS, i.e., communication, computation, and control for both on-demand and continuous WDS monitoring, and when flows are known and

*Parts of this section are reprinted with permission from "Towards Optimal Monitoring of Flow-based Systems using Mobile Wireless Sensor Networks" by Suresh, M.A. and Zhang, W. and Gong, W. and Rasekh, A. and Stoleru, R. and Banks, M.K., ACM Transactions on Sensor Networks, doi: http://dx.doi.org/10.1145/2700256, 2015 Copyright © 2015 Associated Computing Machinery, Inc., "A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems" by Suresh, M.; Manohar, U.; R, Anjana. G.; Stoleru, R.; Mohan Kumar, M.S., IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 Copyright © 2014 IEEE, "Mobile Sensor Networks for Leak and Backflow Detection in Water Distribution Systems" by Suresh, M.A.; Smith, L.; Rasekh, A.; Stoleru, R.; Banks, M.K.; Shihada, B., IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014 Copyright © 2014 IEEE, "On Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Zechman, E.M.; Shihada, B., IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013 Copyright © 2013 IEEE, "Towards Optimal Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Denton, R.; Zechman, E.; Shihada, B., IEEE International Conference on Distributed Computing and Networking (ICDCN), 2012 Copyright © 2012 Springer

unknown.

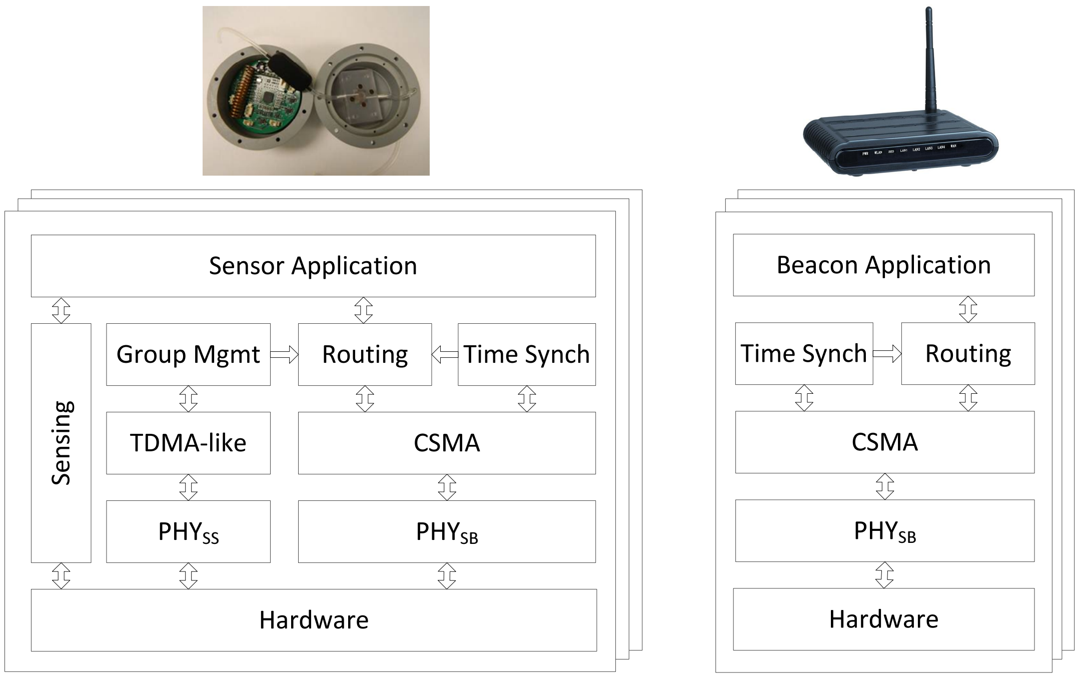


Figure 3.1: Software architecture of the CPWDS

Figure 3.1 is a pictorial representation of the software architecture of the components of the CPWDS. This software architecture may be extended to any system with an inherent flow and sensors flowing through the system, such as the human circulatory system, oil & gas exploration. Hence, we present this model for any general *flow-based* system (FBS). Mobile sensor nodes (executing the shown *Sensor Application*) sense the environment and communicate among themselves using PHY$_{SS}$ (i.e., acoustic modems in a WDS). Sensor nodes are able to communicate through PHY$_{SB}$ (i.e., RF in a WDS) with static beacons, placed outside the pipes strategically. They aid mobile sensors in time synchronization, collect data, and provide information to the sensors (*Beacon Application*).

3.1 Research Thrusts and Preliminary Results

Before presenting the specific problems concerning a CPWDS, we highlight the three key research thrusts, i.e., Communication, Computation, and Control and provide some motivating examples to demonstrate the challenges in these research thrusts.

3.1.1 Communication Issues in CPWDS

In a CPWDS, the topology of the communication network for mobile sensors changes over time. Also, the movement of sensors is not deterministic. Consider a protocol in which sensors are assigned the roles of leaders or followers. At any time, every follower is in communication range of a leader and no two leaders are in communication range of each other. A group can then be defined as a set of sensors containing exactly one leader and its followers. To understand the group dynamics in a WDS (i.e., how the number of groups varies over time), we ran simulations in CPWDSim, a simulator specifically designed for sensors moving in a WDS (details on CPWDSim are given in Section 7.2). Figures 3.2(a)-(b) show how the number of groups varies with time for two different simulation runs on a deployment scenario in Micropolis, a model city [10]. It is clear from these results that the topology of the communication network in a CPWDS changes frequently, and the changes are not deterministic. Therefore, the dynamics of the network, albeit predictable, is challenging to handle.

3.1.2 Computation Issues in CPWDS

Mobile sensors provide improved coverage of pipes at a lower cost to municipalities. The key advantage of using in-pipe sensors, when compared to static sensors, is that events will be detected at closer range. Moreover, these sensors are transported

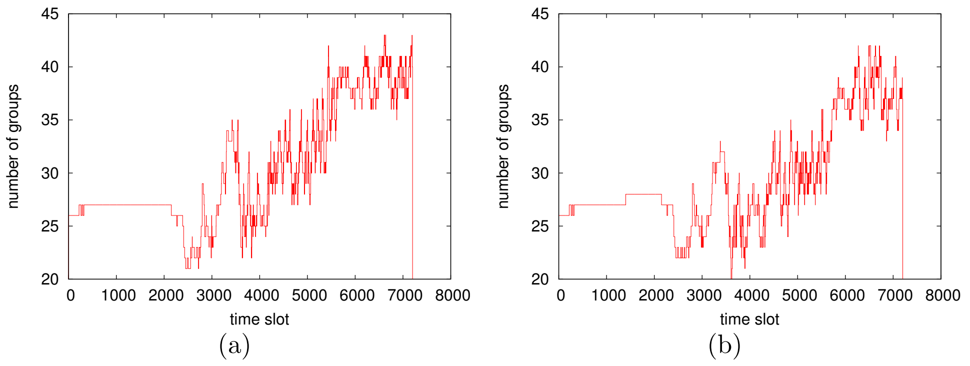


Figure 3.2: The number of groups varying over time for two runs.

by the flow in the network. Consequently, the sensed data is collected more effectively than static sensors with static beacons. When few static sensors are used, the sensors will have to be placed at points where the contaminant has highest concentration to obtain similar results. Additionally, mobile sensors are inexpensive compared to static sensors for WDS monitoring. However, their number still needs to be optimal, especially for on-demand monitoring. To understand the reason to optimize the sensor placement, consider a flow-aware on-demand monitoring example where sensors may be deployed at any junction in the virtual city, Micropolis [10]. Sensing coverage is defined as the fraction of pipes in a zone of interest (i.e., given subset of the pipes) traversed by sensors.

The sensing coverage results obtained from our simulator, CPWDSim are depicted in Figure 3.3. We observe that when the optimal number of sensors (50 sensors in total: 20 at IN1534, 10 at IN1090 and 20 at VN826) are placed at the three insertion points, the coverage achieved is highest. The achieved sensing coverage is higher than the scenario when we insert 100 sensors at the pumpstation, and higher than the scenarios the same number of sensors (i.e., 50 sensors) are all inserted at a single insertion point. Therefore, optimally deploying sensors is necessary for

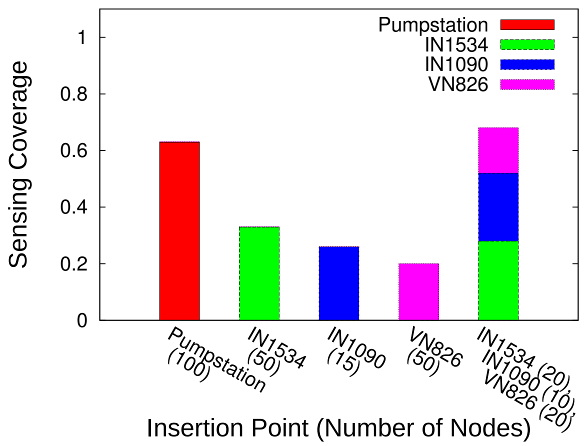


Figure 3.3: Impact of number of sensors and placement on coverage.

cost-effective monitoring of WDS.

In a CPWDS, mobile sensors traversing the pipelines are unaware of their position, uncertain about sensors in communication range, and incapable of predicting encounters with other groups of sensors. In contrast, beacons have knowledge of the WDS network, are aware of their position, and can make predictions about future group splits, group merges, and beacon encounters of sensors. Based on monthly water bills and usage statistics of consumers, which is an indicator of the average demand at the consumer end points of the WDS, beacons *estimate* the flows in the system at any time. Under the assumption that in a WDS, flows in the pipes at all time instants are known, or predicted with high accuracy, the beacons can provide useful information to the sensors.

Consider the scenario in Figure 3.4. After a sensor node n_1 comes within communication range of beacon B_1, the only path the sensor can follow is to beacon B_3. The beacon B_1 provides this information to n_1. Similarly, the sensor n_2 obtains information from B_2 that B_3 and B_4 are the next beacons that the nodes can encounter. The beacon B_2 also knows that a group split at vertex v_1, a group merge

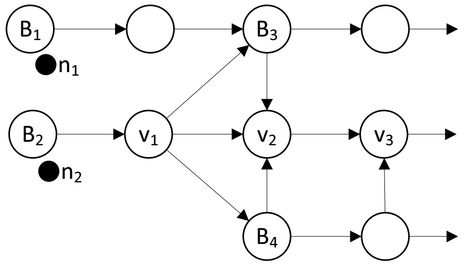


Figure 3.4: Example of the data that beacons can provide.

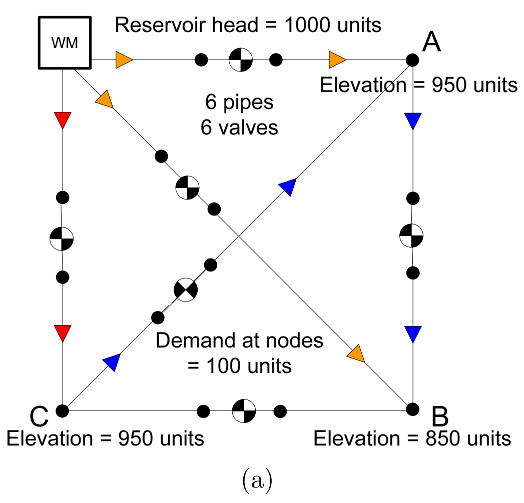
at vertex v_2 or v_3 are possible. The beacons inform the node of the time taken to reach these vertices and beacons (e.g., B_3 at t_1; B_4 at t_2; GM at t_3 or t_4; GS at t_5). The sensors can then tune their protocol parameters to maximize data transfer to beacons or other sensors.

3.1.3 Control Issues in CPWDS

In a WDS, it is possible to restrict mobile sensors from flowing into pipes with low diameter, so as to prevent them from getting stuck at the end points of the network. However, it is possible that sensors move to regions in the network that we are not interested in monitoring or pipes from which they cannot return. In such a case, a flow control mechanism that does not disrupt normal functioning of a WDS (i.e., continues supplying water of required quality to consumers) becomes necessary to ensure availability of sensors in the main pipes.

WDS are either tree-type networks or looped networks. Most of the real world networks are a combination of the two. In looped networks, flow reversal is less costly since the redundancy in the network is high. Most consumers in the WDS have more than one set of pipelines connected to them, ensuring water supply to all users even with a blockage in one of the pipelines.

The reversal of flow in pipes, specific to every network, is achieved by various



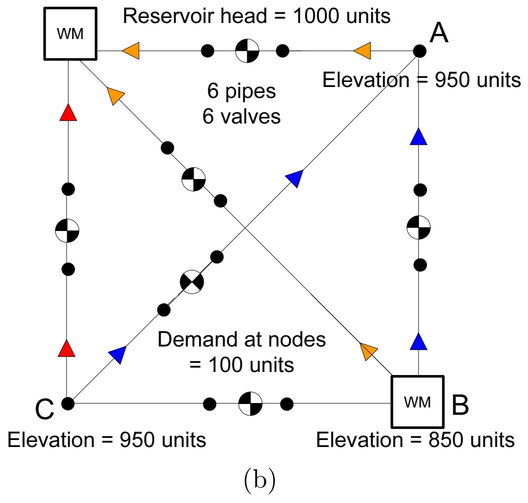


Figure 3.5: Flow reversal using an additional source "WM" at B. Black dots are junctions. Quartered circles are valves.

techniques such as: controllers applied to control the flow in pipes; sinks deployed at certain nodes; additional sources of energy (like pumps) and sources installed according to the hydraulic gradient along the system.

To demonstrate a simple case, consider the WDS in Figure 3.5. Here, either of these approaches or a combination of all the approaches may be implemented for achieving flow reversal. Additionally, care is taken to ensure that the required quantity of water reaches the consumers at required pressure. We performed various iterations by closing valves, running a controller, and by providing sink at different nodes. The most suitable solution is to provide an additional source (as shown in Figure 3.5(b)). Now, the flow from the reservoir has to be stopped (using a valve) and water has to be pumped from the additional source located near node B. In this case, flows in all pipes except pipe AC were reversed. The flow in pipe AC is reversed by establishing a sink at node C. So in this example, flow reversal is done by providing an additional source and sink at nodes.

Flow reversal in pipes without disrupting the water supply to consumers is possible, although it might require installation of additional infrastructure. In an efficient WDS, there is sufficient redundancy to avoid the extra cost. In other cases, such additions make the WDS more fault tolerant.

3.2 Preliminaries

3.2.1 Definitions

A WDS is modeled as a graph $G(V, E)$ in which every edge $(u, v) \in E$ has a non-negative, real-valued capacity denoted by $c(u, v)$, and two sets of vertices: $S = \{s_1, s_2, ..., s_k\}$ a set of *sources*, and $D = \{d_1, d_2, ..., d_k\}$ a set of *sinks*, where $S, D \subset V$.

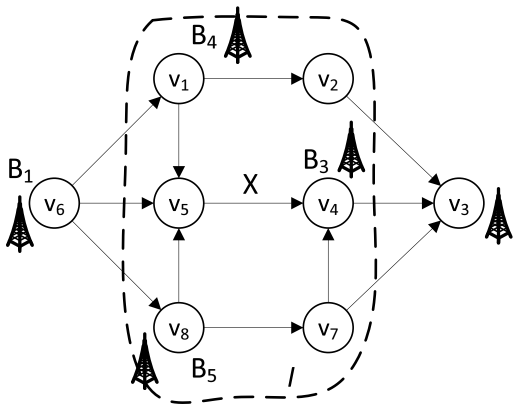


Figure 3.6: Graphical representation of a WDS involving junctions and pipes for the event detection problem: a zone of interest I, beacons B_i and an event X along edge (v_5, v_4).

Definition 1 *A* Sensor *(s_i) is a component which moves through the water pipes guided solely by the flow in the WDS. It follows a mobility model, described in the next section.*

Table 3.1: List of symbols used throughout the dissertation

Symbol	**Definition**
$\mathcal{F}$	Flow network representing the WDS
V	The set of vertices in $\mathcal{F}$
E	The set of edges in $\mathcal{F}$
B_i	Beacon at vertex v_i
SP_i	*Sensed Path* of sensor n_i
PS_i	*Path synopsis* of sensor n_i
I	*Zone of Interest*

Definition 2 *A* Beacon *(B_i) is a component which periodically broadcasts its location. A beacon is placed at a vertex $v_j \in V$.*

Definition 3 *A* Zone of Interest *(I) is a subset of edges in graph $G(V, E)$, i.e., $I \subseteqq E$, which we are interested in monitoring. A given WDS may have multiple Zones of Interest. A typical zone of interest in a WDS may be an area from which complaints are received. All edges in graph $G(V, E)$ can also be considered a zone of interest.*

3.3 Sensing Models

The design of the sensors is specific to the event that needs to be sensed, and the physical properties of the WDS. However, there is a need to model the sensing in a mathematical model to be used in our algorithms for optimal WDS monitoring. In this section, we design a sensing model for any general event in a WDS, or any FBS in general.

To model Sensing in a WDS (as shown in Figure 3.1) we consider two different kinds of events - *non-diffusive*, and *diffusive*. Non-diffusive events propagate mainly based on the distance [4] (e.g., leaks), and diffusive events propagate based on the flow distribution and, typically, time (e.g., chemical contamination of WDS).

Typically, a sensor for physical phenomenon, e.g., pressure sensor for leaks and blocks, and chemical sensors for chemical attacks have specific sensitivities. A sensor can detect an event when the intensity of the event at the point of detection is higher than the sensor sensitivity. This depends on the intensity of the event at the point of the attack, the distance from the event through the flow network, the structure of the WDS and the flows in each pipe. Modeling such a sensor is highly complex. Therefore, in a first attempt, we consider the distance between the event and the sensor, along the network edges, and the availability of any tool that determines the propagation of an event, e.g., contaminant propagation in a WDS.

The intensity of the event as sensed around the event up to a reference distance r_{ref} is said to be $S(0)$. We define $S(r)$ as the intensity of the event at distance r, through the pipes, from the event. Each sensor has a sensitivity Θ_S, e.g., the Purdue/TAMU chemical sensor shown in Figure 1.2 can be tuned to have a sensitivity in the range of 0.1-10 mg/l [8] [100]. If $S(r) > \Theta_S$, the event is detected.

3.3.1 Non-diffusive Events Model

This model describes the intensity of the event at a distance from the source, such as noise, pressure, etc. $S(r)$ in this model is governed by:

$$S(r) = \begin{cases} S(0)(\frac{r_{ref}}{r})^{\alpha} + \mathcal{N}(\mu, \sigma^2) & \text{if event is upstream} \\ \beta\ S(0)(\frac{r_{ref}}{r})^{\alpha} + \mathcal{N}(\mu, \sigma^2) & \text{if event is downstream} \end{cases}$$

where α is the attenuation factor of the event, and $\mathcal{N}$ is noise having a normal distribution with mean μ and variance σ^2. As mentioned earlier, r_{ref} is a reference distance upto which the intensity of the event is $S(0)$. If the event propagates upstream, β is 1. Otherwise, β is 0. We remark here that $r_{ref} = 0$ corresponds to binary sensing, whereby a sensor detects the event only by traversing the edge where it is present.

3.3.2 Diffusive Events Model

This model characterizes the diffusion property of events, such as contamination by non-reactive substances. An example of such a model is the water quality model used in EPANET [27]. The intensity of the event is mainly determined by the flow diversions and mixing at the vertices of the WDS. The intensity of the event through a pipe is assumed to be uniform. The mixing/diversion of an event at a vertex in $\mathcal{F}$ is the flow weighted distribution of intensity of the event from the incoming edges at the vertex.

For a WDS, EPANET [27] simulates chemical contamination propagated through the pipes of the WDS over time. Using EPANET, we obtain reports about the concentration of a chemical contaminant in the edges of the WDS. The concentration of the chemical changes over time. The difference in time (in hours) between the diffusive event entering the system and the sensors being deployed is denoted as τ. More specifically, in this dissertation, the sensors are assumed to be deployed approximately τ hours after the presence of the event is suspected. The concentration of the chemical contaminant in the pipes of the WDS after τ hours is obtained from EPANET, which computes the concentration based on the dimensions of the pipes, the flows in them, and other physical characteristics, such as rate of the reaction (depends on volume, temperature, and mass).

3.3.3 Sensing Range Matrix

We use our proposed sensing models to define a matrix for sensing range as:

$$\mathbf{SR}(i,j) = \begin{cases} 1 & \text{if } S(r) > \Theta_S \\ 0 & \text{otherwise} \end{cases}$$

where r is the minimum edge traversal distance from the mid points of edges e_i and e_j. For any edge e_i, the row i of $\mathbf{SR}$ indicates the edges that can be sensed by a sensor in e_i, and column i indicate the edges whose sensing range include e_i. We note here that the matrix $\mathbf{SR}$ is computed based on a predetermined $S(0)$, which is the minimum intensity of the event that our model is designed to detect.

We remark here that the sensing models presented here may be used to fit the behavior of any sensing modality, by setting the parameters appropriately. Any uncertainty or deviation from the attenuation model or dilution model (as modeled by in EPANET in this case) may be modeled as noise. However, such a fit can only be an approximation.

Example: In the example shown in Figure 5.1, given hardware capabilities of sensor and beacons nodes, let us assume $\alpha = 1$, $\beta = 0$, $\Theta_S = 0.0001$ mg/l, $S(0) = 0.5$mg/l, $r_{ref} = 0.01$m. Therefore, the distance r up to which the event can be detected is $(\frac{S(0)}{\Theta_S})^{\frac{1}{\alpha}}\ r_{ref} = 50$m. Now, $\mathbf{SR}(5,0)$, $\mathbf{SR}(5,2)$ and $\mathbf{SR}(5,4)$ are 1 (i.e., a sensor in e_5 can sense events in e_0, e_2 and e_4) since the distance through the pipes between e_5 and e_0, e_2, and e_4 is less than 50m. Although since the distance through the pipes between e_5 and e_7 is 50m, $\mathbf{SR}(7,5) = 1$ and $\mathbf{SR}(5,7) = 0$ since $\beta = 0$.

3.4 Mobility Model

The mobile sensors in our CPWDS move freely with water flow in pipes and their movement cannot be directly controlled. Assume that flow in pipes is known or estimated. The movement of sensors at junctions is probabilistic owing to the fluid dynamics at the junctions of the WDS. We assume that the probability of a sensor moving into a new pipe section is dependent on the distribution of outgoing flows. The probability of moving from one vertex to another by traversing a single edge is represented in matrix $\mathbf{M}$, where an element m_{ij} of matrix $\mathbf{M}$ is defined as

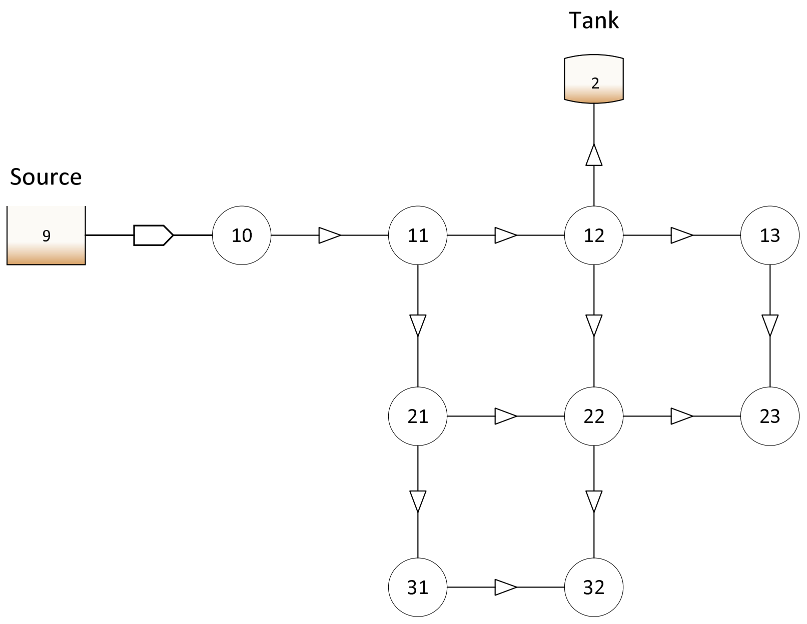


Figure 3.7: Example WDS in EPANET, called Net1

the probability of taking edge e_{ij} at vertex v_i (in one transition step).

$$\mathbf{M} = \begin{pmatrix} 0 & m_{12} & \dots & m_{1n} \\ m_{21} & 0 & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

For example, in Figure 3.7, $m_{11,12} = 0.4360$, $m_{11,21} = 0.5640$, $m_{21,22} = 0.4631$, $m_{21,31} = 0.5369$, $m_{12,2} = 0.7067$, $m_{12,22} = 0.1740$, $m_{12,13} = 0.1193$, $m_{22,23} = 0.3291$, $m_{22,32} = 0.6709$. We note here that if the WDS is a directed acyclic graph (this is mostly the case), then the matrix $\mathbf{M}$ is an upper triangular matrix. We then use a "traversal probability matrix" to represent the probability of a sensor reaching another vertex traversing any path as:

$$\mathbf{T} = \sum \mathbf{M}^k = \mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \ldots$$

An element of matrix $\mathbf{T}$, t_{ij}, is the probability of reaching vertex v_j with a sensor starting from v_i. Typically, we are interesting in monitoring pipes, rather than the junctions, especially for leak detection. Therefore, we determine the probability with which sensors reach the edges of the graph. The probability that a sensor starting from vertex v_r will visit the edge e_{ij} is:

$$t^e_{r,ij} = t_{r,i} \times m_{ij}$$

The above models the probability of a single sensor reaching an edge. We are more interested in the probability that a sensor at a vertex does not reach an edge. The probability of the complementary event is denoted by $\beta_{r,ij} = 1 - t^e_{r,ij}$. For example, in the example illustrated in Figure 3.7, $p^e_{11,(22,32)} = 0.6398 \times 0.6709 = 0.4292$.

Probability of covering an edge

We model the movement of sensors through vertices as a binomial distribution. Each sensor represents a trial in the binomial experiment and the probability that a sensors travels through a certain edge is the probability of "success" for the trials. We assume that the movement of a sensor is independent of the movement of any other sensor.

Consider a scenario where s_r sensors are inserted at a single vertex v_r. The probability that none of the s_r sensors reach edge e_{ij} is $\beta^{s_r}_{r,ij}$ (we assume that the movement of each sensor is independent of the other). Then the probability that none of the sensors reach edge e_{ij} with the configuration $\mathbf{S}$ is:

$$u_{ij}^{e} = \prod_{r=1}^{n} \beta_{r,ij}^{s_r}$$

The sensor deployment is defined in a vector $\mathbf{S}$, of length V, containing the number of sensors inserted at each vertex v_i. Hence, the probability that at least one sensor in a configuration $\mathbf{S}$ reaches e_{ij} is denoted as:

$$PV(\mathbf{S}, e_{ij}) = 1 - u_{ij}^{e}$$

This section laid the foundation of the dissertation. We defined the terms used throughout this dissertation, and described the sensing and mobility models. In the following sections, we will elaborate on each of the components of the CPWDS.

4. COMMUNICATION IN CPWDS*

It is important to encompass the communication mechanisms in a general communication model so as to apply it in our algorithms for optimal WDS monitoring. In this section, we present communication models for sensor-sensor and sensor-beacon communication.

4.1 Assumptions and Definitions

The first assumption we make is that sensors are time synchronized. Unlike other underwater acoustic systems, we use static beacons to aid with time synchronization. Once deployed, there may be time intervals when no beacon is encountered. During such intervals, well known time synchronization techniques may be used [83]. With this assumption, the slotted access mechanism becomes simpler, where the beginning of a slot on each node is synchronized. It will soon be clear that the accuracy of time synchronization required here is low.

As shown in Figure 4.1, depending on the distance of a node n_j from a source n_i, a message Msg sent by n_i at the beginning of a time slot reaches n_j offset from the beginning of the time slot by δ_{ij} due to the inherent spatial uncertainty of the acoustic medium. This helps predict the distance between the transmitter and receiver and to identify contenders [84]. The slot length Δ is selected to be large enough so that a message from a transmitter can reach all the nodes in its communication range [44].

*Parts of this section are reprinted with permission from "Towards Optimal Monitoring of Flow-based Systems using Mobile Wireless Sensor Networks" by Suresh, M.A. and Zhang, W. and Gong, W. and Rasekh, A. and Stoleru, R. and Banks, M.K., ACM Transactions on Sensor Networks, doi: http://dx.doi.org/10.1145/2700256, 2015 Copyright © 2015 Associated Computing Machinery, Inc., "A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems" by Suresh, M.; Manohar, U.; R, Anjana. G.; Stoleru, R.; Mohan Kumar, M.S., IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 Copyright © 2014 IEEE

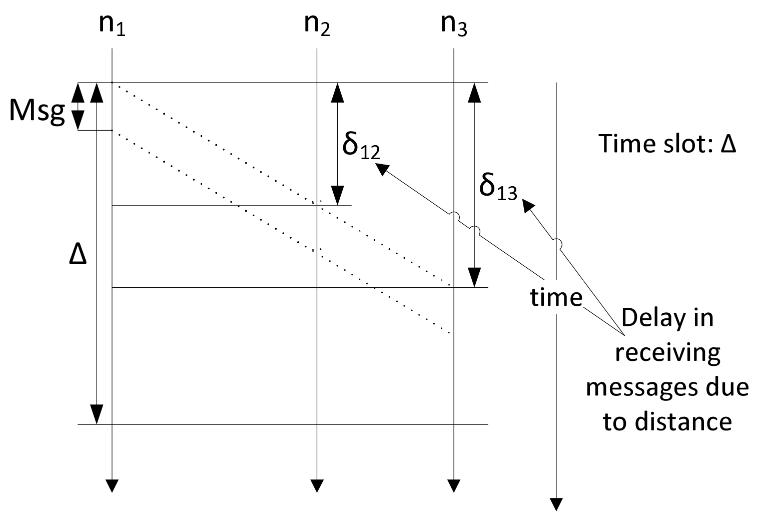


Figure 4.1: Spatial uncertainty in the acoustic medium.

Since the length of the time slot is on the order of milliseconds, the protocol works well with a low accuracy of time synchronization.

Table 4.1: List of symbols used in this section

Symbol	**Definition**
$\mathrm{PHY}_{\mathrm{SS}}$	The medium of communication among sensors
$\mathrm{PHY}_{\mathrm{SB}}$	The medium of communication between sensors and beacons
CR_s	The maximum distance between two sensors such that a message sent by a sensor through $\mathrm{PHY}_{\mathrm{SS}}$ can be received by the other sensor
CR_bs	The maximum distance between a sensor and a beacon such that a message sent by one through $\mathrm{PHY}_{\mathrm{SB}}$ can be received by the other
g_i	group ID of node n_i
p_i	Period of transmission of group g_i

In our communication model, the communication between sensors and beacons is RF. Since it involves the communication between sensors that are buried underground in pipelines and beacons that are outside the pipes, the range is assumed to be much smaller than the acoustic communication among sensors.

We assume that the only reason for failure of communication among nodes that are in range of each other is collisions. All nodes in the network listen to acoustic messages all the time, since receiving consumes lower power than transmitting [9].

Table 4.1 contains all symbols used in this section.

4.2 Physical Layer Models (PHY_{SS} and PHY_{SB})

We have designed the physical layer models for a generalized case of CPWDS, i.e., any system with an inherent flow in it, called flow-based systems (FBS).

As illustrated in Figure 3.1, mobile sensors can communicate with other mobile sensors and with beacons (when they are in range of each other). However, the design of PHY_{SS} and PHY_{SB} physical layers is dependent on the physical conditions in which the FBS monitoring infrastructure is deployed. E.g., in a WDS, mobile sensors communicate among themselves using acoustic modems, because sensors can communicate using pipes as waveguides. Communication between sensors and beacons is RF due to limitations in acoustic transmission between water (i.e., sensors inside pipes) and ground (i.e., beacons outside pipes) [77]. In this dissertation, we model the specific case of a WDS where PHY_{SS} is underwater acoustic and PHY_{SB} is RF.

We model the pathloss for RF communication between underground underwater sensor and a beacon placed outside the pipe as a log-distance pathloss model:

$$H_{RF} = PL_{RF}(r) - PL_{RF}(r_{ref}) = 10nlog\left[\frac{r}{r_{ref}}\right] + \mathcal{N}(\mu, \sigma^2)$$

where r is the range, r_{ref} is the reference distance, set to 1m and n is the path loss exponent that depends on obstacles between the source and the receiver. $\mathcal{N}$ is noise having normal distribution with mean μ and variance σ^2. We evaluate the RF communication between sensors in water pipes and static beacons outside the water and present the results in Section 7.1.

We model the acoustic signal transmission using the path loss model:

$$H_A = PL_A(r) - PL_A(r_{ref}) = 10klog\left[\frac{r}{r_{ref}}\right] + ar + \mathcal{N}(\mu, \sigma^2)$$

where r is the range, r_{ref} is a reference distance, set to 1m, k is a factor that depends on the structure of the pipes and a is a factor determined by the fluid carried - i.e., water, and $\mathcal{N}$ is noise normally distributed with mean μ and variance σ^2.

We model acoustic communication in a straight pipe using a cylindrical path loss model [89], where k=1 and $a = A + 1$, where A is the loss due to scattering. For pipes with bends, we use a spherical spreading model [43] as an approximation where k=2 and a is the absorption factor for water that depends on the frequency of sound waves [89].

4.2.1 Communication Range for Sensors (CR_s)

CR_s is defined as the maximum distance between an acoustic sender and receiver (i.e., sensors) such that the receiver can decode the data transmitted by the sender at a known transmit power in the absence of noise and interference.

4.2.2 Communication Range for Beacons (CR_b)

CR_b is defined as the maximum distance between an RF sender and receiver (i.e., sensors) such that the receiver can decode the data transmitted by the sender at a known transmit power in the absence of noise and interference. We use a communication range matrix for beacons, $\mathbf{CR}_b$ as:

$$\mathbf{CR}_b(i,j) = \begin{cases} 1 & \text{if } r < CR_b \\ 0 & \text{otherwise} \end{cases}$$

where $v_i \in V$ and $e_j \in E$, and r is the physical distance from v_i to the midpoint of e_j. For every vertex v_i, row i of $\mathbf{CR}_b$ represents edges on which a sensor needs to be present to hear from a beacon at v_i. For every edge e_j, the column j represents vertices that are in communication range with a sensor at e_j.

4.3 Sensor-Sensor Communication Protocols

Communication among sensors serves two important purposes: i) it serves as indicators of positions (as in range-free localization [76]), thereby reducing the number of beacons required to be deployed solely for that purpose; ii) they are required to collect information from sensors that may move through paths where there may not be any beacons to report data to. In our design, sensor nodes use Group Management (as shown in Figure 3.1) to reduce communication traffic. Unlike terrestrial RF networks, underwater acoustic networks suffer from high propagation delays, making it challenging to reduce delays, while ensuring a low collision rate. In this section, we design the MAC and Group Management protocols to be used in the communication among sensors that has a low collision rate, but is capable of ensuring that groups of sensors can exchange data with bounded delay. The protocol is described and its delay bound is proved in this section.

In our Sensor-Sensor communication protocol, we assume that nodes have unique IDs. When a set of nodes is deployed into the network, a leader node is pre-assigned. In our group management protocol, every node has a leader in its range, and no two leaders are in range of each other. The group that a node belongs to is determined by its leader. The leader chooses a random group ID g_i. It periodically broadcasts a heart-beat message called GROUP-HELLO containing the node ID n_i and group ID g_i.

We consider two modalities for the group communication protocols. The first is when the followers do not communicate, and the other when followers communicate. If the sensing range is larger than the communication range, the followers need not

Algorithm 1 Group Management on node n_i without reporting

```
Require: node list, $g_i$, $l_i$
 1: merged $\leftarrow$ false
 2: for each time slot $T_i$ do
 3:   if $n_i = l_i$ then
 4:     if $T_i \% Prime(g_i) = 0$ then
 5:       if merged = true then
 6:         Transmit GROUP-MERGE
 7:       else
 8:         Transmit GROUP-HELLO
 9:       end if
10:     end if
11:     if Received GROUP-HELLO from $n_j$ then
12:       merged $\leftarrow$ true; $l_i \leftarrow n_j$; $g_i \leftarrow g_j$
13:     end if
14:   else
15:     merged $\leftarrow$ false
16:     if $T_i \% Prime(g_i) = 0 \wedge$ No hello received then
17:       time waited = 1
18:     end if
19:     if time waited = rank($n_i$) in node list then
20:       $l_i \leftarrow n_i$; $g_i \leftarrow$ new group id
21:     else if time waited > 0 then
22:       increment time waited
23:     end if
24:     if GROUP-HELLO received from $n_j \wedge n_j \neq l_i$ then
25:       $l_i \leftarrow n_j$; $g_i \leftarrow$ group id of $n_j$
26:       time waited = 0
27:     end if
28:   end if
29:   if Received GROUP-MERGE from $n_j$ then
30:     merged $\leftarrow$ true; Update node list
31:   end if
32: end for
```

sense, or transmit. Otherwise, an event might be missed by followers not sensing and reporting to the leader.

First, consider the case when the follower nodes do not communicate among themselves or with the leader (i.e., in the acoustic channel). All nodes in the group (including the leader) store the last beacon encountered. Our distributed algorithm for group management without reporting is shown in Algorithm 1. Acoustic communication from leaders is slot-based (TDMA-like). A hash function maps the group ID g_i to a prime number p_i. The leader broadcasts the GROUP-HELLO packet with a period p_i (Lines 4-8).

When followers also need to communicate, the protocol requires the leaders and followers to perform different functions. Every node is given a unique node id. The actions taken by a leader are described in Algorithm 2, and the actions taken by a follower are described in Algorithm 3. A leader sends a $HELLO$ message every p_i+2 slots where p_i is a prime number associated with a group led by node n_i (Algorithm 2, lines 2-7), selected from a hash table. The $HELLO$ message includes p_i, n_i, and $beaconData$ that includes global view data as described in the next subsection. When a follower hears this message, and it is determined that data needs to be reported, the follower sends its report to the leader (Algorithm 3, line 11-12, 15-16). Once the leader receives the reports from its followers, it sends an ACK message containing the node ids of the followers it received reports from (Algorithm 2, lines 12-14).

Complications arise when: a) group members do not hear the GROUP-HELLO or HELLO (i.e., a subgroup split from the main group); b) a leader hears another leader's GROUP-HELLO or HELLO (i.e., two or more groups merge). These are described below.

4.3.1 Group Splitting and Leader Election

When a group g_i split occurs, the subgroup containing the leader l_i will continue as group g_i.

First, we consider the case where followers do not sense and report. Among the

Algorithm 2 Leader node n_i with reporting

```
Require: $g_i$, $p_i$, $timeElapsed$, $beaconData$, $ACK$, $allData$, $mergeDetected$
 1: for each time slot do
 2:   if $timeElapsed \% p_i + 2 = 0$ then
 3:     if $mergeDetected$ is true then
 4:       Broadcast (Acoustic) $MERGE$
 5:     else
 6:       Broadcast (Acoustic) $HELLO$
 7:     end if
 8:     $timeElapsed = 0$
 9:   end if
10:   if $BEACON - HELLO$ received then
11:     Broadcast (RF) $allData$
12:   end if
13:   if $dataToReport$ received then
14:     Append $nodeId(dataToReport)$ to $ACK$
15:   end if
16:   if $ACK$ is not empty at beginning of time slot then
17:     Broadcast (Acoustic) $ACK$
18:     Clear $ACK$
19:   end if
20:   if $beaconData$ received from Beacon then
21:     Add $beaconData$ to $HELLO$
22:   end if
23:   if $HELLO$ received from $n_k$ then
24:     $mergeDetected$ = true;
25:     Relinquish leadership and follow $n_k$
26:   end if
27:   if $MERGE$ received from $n_k$ then
28:     $mergeDetected$ = true;
29:   end if
30:   Increment $timeElapsed$
31: end for
```

nodes in a group without a leader, a node will decide to be new leader, based on the following rule: after detecting the absence of a leader, each node waits for a GROUP-HELLO based on its rank in list of nodes that were in the group before the split is detected (Lines 16-23). E.g., if a node's rank was 8 before the group split,

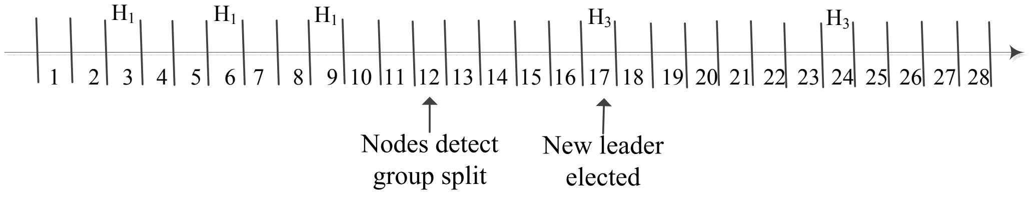
Algorithm 3 Follower node n_j with reporting

```
Require: $g_j$, $p_j$, $timeElapsed$, $beaconData$, $ACK$, $allData$, $report$
 1: for each time slot do
 2:     if $timeElapsed > p_i + 2$ then
 3:         Content to be leader
 4:     end if
 5:     if Contention for leader won then
 6:         $g_j = n_j$; $p_j$ = new prime number
 7:         Perform leader operations
 8:     end if
 9:     if $ACK$ received and contains $n_j$ then
10:         $report$ = false
11:     end if
12:     if $HELLO$ received then
13:         $timeElapsed = 0$
14:         if Data needs to be sent then
15:             $report$ = true
16:         end if
17:         if $HELLO$ contains $beaconData$ then
18:             Update $beaconData$
19:         end if
20:     end if
21:     if $report$ is true then
22:         Broadcast (Acoustic) $allData$ summary
23:     end if
24:     Increment $timeElapsed$
25: end for
```

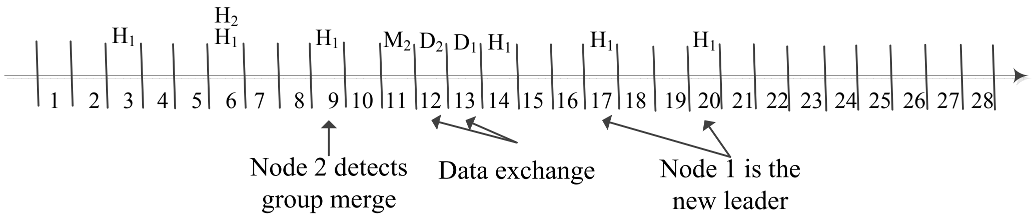
the node would wait 7 slots. If no GROUP-HELLO packets are received, the node assumes the leader role and broadcasts its ID together with a randomly chosen group ID (Line 20). All nodes in the split group accept the leader and the group ID, and do not contend for the leader role. The hash function, based on the newly chosen group ID, will produce a new prime number p_j. The new leader will start advertising GROUP-HELLO packet with a period p_j.

Next, when the followers sense and report, when a follower does not hear from its leader after $p_i + 2$ slots, it is determined that the follower is no longer in range of the leader, i.e., a group split has occurred. The followers then contend to be a leader by sending out messages containing their id. If a node has the smallest id among all its neighbors, it becomes the new leader (Algorithm 3, lines 2-6).

Figure 4.2(a) shows an example of a group split. Leader of group 1 sends hello message H_1 every 3 time slots. Let's assume the nodes in group 1 are {1, 2, 5, 19, 503, 840}. When nodes 1, 2, 5, 19 go out of range, nodes 503 and 840 detect a group split after 3 time slots. Their ranks in the list of nodes are 5 and 6, respectively. From slot 17, node 503 sends hello message H_3 every 7 slots with group ID 3. Nodes 530 and 840 update the list of nodes to {503, 840}.



(a)



(b)

Figure 4.2: Examples of group management split (a) and merge (b) operations.

4.3.2 Group Merging and Leader Selection

When a leader l_i hears the GROUP-HELLO packet of another leader l_j, the two groups need to be merged.

Without followers sensing and reporting, the leader l_i relinquishes its leader role, accepts the new group ID g_j and the ID of the new leader l_j (Line 11). After p_i slots, l_i sends a GROUP-MERGE packet, instead of a GROUP-HELLO, containing new leader's ID l_j, the new group g_j, the IDs of nodes in group g_i and the data maintained by the group g_i (Line 6). l_i will not communicate after this, unless it becomes a leader after a group split. When leader of group g_j hears the GROUP-MERGE sent by leader from group g_i, it responds with the IDs of nodes in its group (i.e., group g_j) and the data maintained by group g_j (Lines 6 and 30).

With followers sensing and reporting, when two groups merge, the leader that transmitted the first $HELLO$ retains leadership, and the leaders exchange their data in a $MERGE$ message (Algorithm 2, lines 4, 20-21). A $MERGE$ message contains n_i, p_i, and $allData$.

Figure 4.2(b) shows an example of groups 1 and 2 merging. Leader nodes send hello messages (H_1 and H_2). Once leader node 2 detects a group merge, it sends merge message M_2 followed by data D_2. Nodes update the list of nodes in their group. Node 1 then sends its data and remains the leader.

4.3.3 Delay Analysis for Group Management

At the MAC layer, we set the time slot size such that a message sent by a node reaches all nodes in range, within one time slot (e.g., if the range of communication is 10m, the time slot chosen will be 7ms). This is done to avoid spatio-temporal uncertainty associated with underwater acoustic communication [52].

Without followers sensing and reporting, since nodes transmit only in time slots that are multiples of prime numbers, the worst delay in identifying a group merge is $kp + 1$ where $k + 1$ is the number of groups merging, and p is the minimum p_i among all merging groups ($min(p_i) \forall$ merging g_i). Since nodes in group g_i that are not leaders expect to hear GROUP-HELLO from their leader every p_i slots, the time to detect a group split is p_i.

With followers sensing and reporting, since leaders transmit at regular intervals with a period, the worst case delay in identifying a group merge is $kp+1$ where $k+1$ is the number of groups merging, and $p = min(p_i + 2) \forall$ merging g_i. The time to detect a group split is $p_i + 2$ since nodes in a group g_i that are not leaders expect to hear $HELLO$ from their leader every $p_i + 2$ slots.

4.4 Sensor-Beacon Communication Protocols

Communication between sensors and beacons is necessary to collect information from the sensors without collecting the sensors physically. In this section, we design the protocols to be used in the communication between sensors and beacons.

Nodes synchronize with the beacons when they come in range of each other, using Cristian's Algorithm [19] for time synchronization. This step is necessary, since the nodes report time-stamped data to beacons. The accuracy of time synchronization required is in the order of tens of milliseconds since sensor-sensor acoustic communication and timing of event detection can tolerate these errors. The sensors use a simple epidemic routing algorithm. Since the communication is wireless, all devices in a given range can listen to a broadcast message.

Beacons act as sinks and are critical for data collection. They also periodically broadcast their identifiers for localization in a RF-HELLO packet. Only specific sensor nodes, called leaders, transmit data to beacons. The concept of a leader and its group is presented in Section 4.3. RF communication between the group leader and beacons is based on CSMA. When a mobile sensor receives the RF-HELLO packet sent by a beacon, it first performs time-synchronization and then sends its data to the beacon. At the end of the RF communication with the beacon node, the nodes erase the transmitted data, and start recording new sensed data.

To save power, nodes may use a duty cycling mechanism. The total capacity of data that can be transferred to the beacons at each contact depends on the beacon interval, duty cycle and the communication ranges. The amount of processing that is performed onboard and the rate of sampling of sensor nodes have to be chosen appropriately to avoid losing data. Even with a duty cycling mechanism, the time of contact is expected to be sufficient for the sensor to hear the beacon, since the movement of sensors in the pipes is slow (e.g., a few meters per second).

5. COMPUTATION IN CPWDS*

In this section, we present the computation aspects of the CPWDS, i.e., sensor deployment, beacon placement, event localization, and global view algorithm. Some definitions of terms used in this section are as follows:

Definition 4 *A* Sensed Path *(SP_i) is a set $\{e_j \mid e_j \in E\}$ of edges through which a sensor n_i traveled and sensed events and proximity to beacons.*

Definition 5 *An* Insertion Point *(or* Source*) is a vertex $s_i \in V$ at which sensors are introduced into the WDS.*

Definition 6 *A* Path Synopsis *(PS_i) for a sensor n_i is an ordered list of events and beacons encountered by the sensor along its Sensed Path SP_i.*

Table 5.1 contains all symbols used in this section.

5.1 Sensor Deployment Problem for On-demand Monitoring

In flow-aware on-demand monitoring, flows in the WDS can be predicted with high accuracy for any time period based on monthly average demands at the consumer ends. We can make an intelligent decision on the placement of sensors. Event

*Parts of this section are reprinted with permission from "Towards Optimal Monitoring of Flow-based Systems using Mobile Wireless Sensor Networks" by Suresh, M.A. and Zhang, W. and Gong, W. and Rasekh, A. and Stoleru, R. and Banks, M.K., ACM Transactions on Sensor Networks, doi: http://dx.doi.org/10.1145/2700256, 2015 Copyright © 2015 Associated Computing Machinery, Inc., "A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems" by Suresh, M.; Manohar, U.; R, Anjana. G.; Stoleru, R.; Mohan Kumar, M.S., IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 Copyright © 2014 IEEE, "Mobile Sensor Networks for Leak and Backflow Detection in Water Distribution Systems" by Suresh, M.A.; Smith, L.; Rasekh, A.; Stoleru, R.; Banks, M.K.; Shihada, B., IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014 Copyright © 2014 IEEE, "On Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Zechman, E.M.; Shihada, B., IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013 Copyright © 2013 IEEE, "Towards Optimal Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Denton, R.; Zechman, E.; Shihada, B., IEEE International Conference on Distributed Computing and Networking (ICDCN), 2012 Copyright © 2012 Springer

Detection happens if at least one of the sensors deployed traverses an edge where the event can be detected. Formally, we decide *Sensor Deployment (Location and Time)* $\mathcal{S} = \{(s_i, q_i, t_i) \mid s_i \in V, q_i \in \mathbb{N}, t_i \in \mathbb{N}\}$, where q_i denotes the number of sensors inserted at time t_i at vertex s_i.

We note here that optimal sensor deployment is necessary for on-demand monitoring so as to improve the probability that the events are detected. We solve this problem from two different, yet equally relevant, perspectives. For the first problem, we assume that we have access to any number of sensors, and the utility manager is interested in obtaining a certain degree of coverage of a zone of interest in the WDS. For the other problem, we assume that we have a fixed number of sensors. We then decide their deployment to maximize the degree of coverage.

5.1.1 Minimization of Number of Sensors

The optimal sensor deployment problem is the problem of determining the number of sensors such that the probability that any edge is not covered is above a false negative rate, FN. Covering an edge simply means that at least one sensor traverses that edge. Hence, we can formally define the problem of minimizing the number of sensors deployed as:

$$\text{minimize} \sum_{i=1\ldots n} q_i$$

$$\text{subject to}$$

$$\left(\sum_{s_i \in V} q_i \ \ln(1 - ts_{ij})\right) \leq \ln \ FN \ \forall \ v_j \in V$$

$$q_i \geq \ 0 \ \ \forall i = 1 \ldots n$$

where $ts_{ij} = max(t_{ij}, t_{ik} \times \mathbf{SR}(j,k)) \ \forall \ e_k$. t_{ij} for every pair of vertices is the probability that a sensor in vertex v_i reaches vertex v_j. The variables and constraints are

Table 5.1: List of symbols used in this section

Symbol	**Definition**
D_c	*Degree of Coverage*
P_d	*Event Localization Accuracy*
FN	False negative rate, i.e., the probability that an event goes undetected
LE	Localization error, i.e., the radius of the region where an event is suspected to be present
Suspects List	List of suspected edges
p_i	*Edge chances*: the probability that a sensor flows through edge e_i.
$\mathbf{M}$	*Transition Matrix*: the matrix composed of the probability p_i of each edge.
$\mathbf{T}$	*Traversal Probability Matrix*: the matrix composed of transition probabilities given by $\sum_{k=1}^{l} \mathbf{M}^k$.
l	$l \in \mathbb{N} \mid \mathbf{M}^l = 0$ and $\mathbf{M}^{l-1} \neq 0$
ν_i	Number of sensors that traverse edge e_i.
$\mathbf{N}$	*Sensor Requirement Matrix*: the matrix containing number of sensors to insert in each vertex to reach the vertices of the graph.
$\mathcal{G}$	*Goodness Matrix*: the matrix used to select insertion points.
S	Set $S = \{(s_i, q_i) \mid s_i \in V \wedge q_i \in \mathbb{N}\}$ where s_i is an insertion point and q_i is the number of sensors inserted there.
$v_i.\phi$	*Potential*: the maximum of the number of edges in the set of paths leading from a beacon to v_i.
BT	*Beacon Table*: the table that contains information about edges between any two beacons.
SR	The sensing range matrix
SR	The sensing range, i.e., the distance through the edges up to which an event can be sensed
B_I	*Badness*: Badness metric of zone of interest I
ale	*Average Localization Error*
sle	*System Localization Error*

explained in this section.

We note here that the above problem has been proven to be NP Hard (Appendix A). We therefore present a greedy heuristic in this dissertation. We not only determine *where* to insert the sensors, and *how many* need to be inserted. We also determine the *time of deployment* for the sensors. We need the time of deployment for on-demand monitoring to ensure that groups of sensors merge or split at certain vertices, thereby eliminating the need to place beacons at those vertices. The sensor placement algorithm ultimately decides: $\{s_i, q_i, t_i\}$, where q_i sensors are inserted at vertex s_i at time t_i.

Algorithm 4 Sensor Deployment

```
Require: $FN$, $\mathcal{F}$, $\mathbf{SR}$
1: $\{s_i, q_i, 0\}$ = Location Of Deployment($\mathcal{F}$, $\mathbf{SR}$, $FN$)
2: $\{s_i, q_i, t_i\}$ = Time Of Deployment($\mathcal{F}$, $\{s_i, q_i, 0\}$)
```

Algorithm 4 presents the Sensor Deployment (with its two steps) while ensuring a bound false negative. Step 1 of the algorithm ensures a bounded false negative rate FN for sensor deployment, and is presented in Algorithm 5. The false negative bound is defined as the probability of an event being present but not detected, is less than FN. Step 2 of the algorithm determines the time of deployment of sensors, and is presented in Algorithm 6.

To determine $\mathcal{S}$ in Algorithm 5, we first use the sensor mobility model described in Section 3.

From the mobility model, we derive the number of sensors to be inserted at vertex v_i to ensure that at least one sensor reaches a vertex v_j as:

$$\mathbf{n}_{i'j'} = \frac{\ln(FN)}{\ln(1 - p_{ij})}$$

A matrix $\mathbf{N}$, representing the number of sensors to be inserted at each vertex such that the probability of at least one sensor reaches another vertex (using transition probabilities from $\mathbf{T}$) is at most FN is then computed (Lines 9-14) by computing $\mathbf{n}_{i'j'}$ for all vertices. If we ensure that the probability that an edge in I is not covered by any sensor, is at most FN, then the expected false negative rate of I is at most FN, thereby meeting our requirement.

Algorithm 5 Location Of Deployment

```
Require: $\mathcal{F}$, $\mathbf{SR}$, $FN$
 1: for each $v_i \in V$ do
 2:   for each $v_j \in V$ do
 3:     $\mathbf{M}_{ij} = P(v_j \mid v_i)$
 4:   end for
 5: end for
 6: while $\mathbf{M}^k \neq 0$ do
 7:   $\mathbf{T}+ = \mathbf{M}^k; k++$
 8: end while
 9: for each $v_i \in E$ do
10:   for each $v_j \in E$ do
11:     $\mathbf{n}_{i'j'} = \frac{\ln(FN)}{\ln(1-\mathbf{T}_{ij})} + 1$
12:   end for
13: end for
14: for each $v_i \in V$ do
15:   for each $v_j \in V$ do
16:     $E_k$ = set of edges between $v_i$ and $v_j$ that are covered by $n_{ij}$ sensors
17:     $\mathcal{G}_{ij} = \frac{n_{ij}}{\sum_{e_k} \sum_{\forall e_l} (\mathbf{SR}(e_k, e_l))}$, where $e_k \in E_k$
18:   end for
19: end for
20: while Edges in $I$ are not covered do
21:   $\mathcal{G}_{ij} = \min(\mathcal{G})$
22:   Append $\{v_i, n_{ij}, 0\}$ to $\mathcal{S}$
23:   Update $\mathcal{G}$ for covered edges
24: end while
```

Next, we derive a goodness matrix $\mathcal{G}$ from $\mathbf{N}$ to help choose the best vertices of insertion and the number of sensors to be inserted, i.e., $\mathcal{S}$. The problem of choosing the best vertices is NP Hard. Therefore, we use a greedy heuristic to solve this problem (approximation ratio of $\ln |V|$). The goodness matrix $\mathcal{G}$ is defined as $\mathcal{G}_{ij} = \frac{n_{ij}}{\sum_{e_k} \sum_{\forall e_l} (\mathbf{SR}(e_k, e_l))}$, where $e_k \in$ set of edges between v_i and v_j that are covered by the n_{ij} sensors, and n_{ij} is an element of the matrix $\mathbf{N}$. The denominator of the expression indicates the number of edges that can be sensed by the n_{ij} sensors inserted at v_i. The set E_k (Line 16) is computed using breadth first search starting at v_1 and searching only through branches that require less than n_{ij} sensors.

Algorithm 6 Time Of Deployment

```
Require: $\mathcal{F}$, $\{s_i, q_i, 0\}$
 1: Initialize $J \leftarrow$ Empty array, $G_{rev}$, $\Delta \leftarrow$ Empty array of arrays.
 2: $IP \leftarrow \{s_i\} \forall s_i \in$ insertion points
 3: for each $v_i \in V$ do do
 4:    if $inDegree(v_i) > 1 \vee outDegree(v_i) > 1$ then
 5:       Insert $v_i$ into $J$
 6:       $\Delta(i) \leftarrow$ Longest path from all sources $s_j$ to $v_i$
 7:    end if
 8: end for
 9: Queue $Q \leftarrow J$
10: while $Q$ is not empty do
11:    $v_i \leftarrow tail(Q)$
12:    for each $v_j \in$ out-vertices of $v_i$ in $G_{rev}$ do
13:       $\Delta(j).insert(\Delta(i)$- Time in $(v_i, v_j))$
14:    end for
15: end while
16: for each $ins_i \in \{s_i, q_i, 0\}$ do
17:    $q_j \leftarrow q_i / size(\Delta(i))$
18:    for each $t_j \in \Delta(i)$ do
19:       $ins_i \leftarrow \{s_i, q_j, t_j\}$
20:    end for
21: end for
```

The smallest $\mathcal{G}_{ij}$ corresponds to the best insertion point, and we add $(v_i, n_{ij}, 0)$ to $\mathcal{S}$ (Lines 14-19). Once a selection is made, $\mathcal{G}_{ij}$ is updated by removing all edges covered at each step, i.e., edges e_l for which $\mathbf{SR}(e_k, e_l) = 1$, where $e_k \in E_k$ (Lines 20-24). If SR is high, more edges are removed at each step.

We define *Sensing Coverage* as the fraction of edges in I that are covered by at least one sensor. The expected value of 1-FN is then equal to the expected sensing coverage. Therefore, if the sensing coverage for a particular sensor deployment is at least 1-FN, the sensor deployment also meets the false negative bound requirement. We use Sensing Coverage as a metric in the evaluation of Sensor Deployment. We refer to 1-FN as the degree of coverage (D_c).

The next edge a sensor traverses after reaching a vertex is determined by the out-flow rates at the vertex. Owing to irregularities in the pipe dimensions and varying demand at the sinks, the difficulty of covering a zone of interest with sensors can vary. Hence, we define the term *Badness for Zone of Interest*, defined as $B_I = \sqrt{\sum (f_x^i - f_x)^2} \times 1000$, where f_x is the flow in edge x of the zone of interest and f_x^i is the ideal flow in that edge (i.e., all out-flows are equally divided at all vertices. This is ideal since it requires the least number of sensors for a required FN).

A relevant concern with the algorithm is that the insertion points may be restricted to a few nodes in I (e.g., manholes are only available at a few vertices in a WDS). This challenge can be overcome by reconstructing the graph: remove vertices that are inaccessible and add edges to maintain connectivity.

The Time of Deployment of sensors in $\mathcal{S}$ is presented in Algorithm 6. We define vertices of I with more than one incoming/outgoing edge as *intersections* J (Lines 3-8). These are vertices where sensors traversing different paths can communicate. We clarify here that sensors traversing through the intersections may move through different paths. We assign each edge of I a weight equal to the time a sensor would spend in the edge, and determine the longest paths between insertion points and intersections.

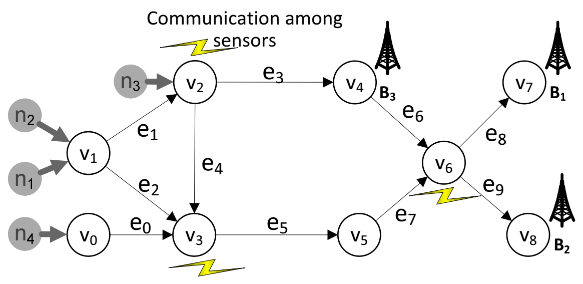


Figure 5.1: WDS monitoring example: empty circles are vertices of $\mathcal{F}$, filled circles are sensors, small towers are beacons.

Let $G(V, E)$ and $G_{rev}(V, E')$ be two DAGs where the only difference between G and G_{rev} is the direction of their edges. More formally, G and G_{rev} have the same set of vertices and $(u, v) \in E$ if and only if $(v, u) \in E'$ $\forall$ $u, v \in V$. Traversing G_{rev} in a breadth first manner from the intersections will lead us to the insertion points from various paths. If a sensor needs to reach a intersection at time t, it needs to be inserted at $t - t'$ at insertion point s_i, where t' is the time taken to reach the intersection from s_i. We obtain a list of possible insertion times for each insertion point corresponding to different intersections and the paths leading to them, as shown in lines 10-15. For example, in Figure 5.1, if sensors inserted at v_0 and v_1 have to reach v_6 at time t, and the time to traverse any edge e_i is t_i, the possible times of insertion at v_0 is $\{t\text{-}t_0\text{-}t_5\text{-}t_7\}$, and at v_1 are $\{t\text{-}t_1\text{-}t_3\text{-}t_6,\ t\text{-}t_1\text{-}t_4\text{-}t_5\text{-}t_7\}$. Finally, we equally distribute sensors among insertion times (Lines 16-21).

5.1.2 Maximization of Degree of Coverage

Given a fixed number of sensors, we solve the problem of maximizing the coverage of the sensors. We formulate two problems, namely, the problem of maximizing the least probability of covering any edge, and the problem of maximizing the average probability of covering any edge. The two formulations impress on the tradeoffs involved, because the first problem ensures a better coverage, but is computationally intensive. For this problem, we consider SR to be 0m, i.e., covering an edge is synonymous to detecting a event present on that edge.

5.1.2.1 Maximize Lower Bound Sensing Coverage (MLBSC)

We define Lower Bound Sensing Coverage, $LBSC$ as the minimum probability of covering an edge, i.e., Lower Bound Sensing Coverage is the largest number such that $\forall\ e_{jk} \in E$, $[PV(s, e_{jk})] \geq LBSC$, i.e.,

$$LBSC = \min_{e_{jk}} [PV(s, e_{jk})] \tag{5.1}$$

The problem of maximizing $LBSC$ is formulated below.

$$\text{maximize LBSC i.e.,}$$

$$\begin{aligned} &\text{maximize } \min_{e_{jk}} \left[\left(1 - \prod_i \left(1 - t^e_{i,jk}\right)^{s_i} \right) \right] \\ &\text{such that} \\ &\quad \sum_{i=1\ldots n} s_i = c \\ &\quad s_i \geq\ 0\ \ \forall i = 1 \ldots n \end{aligned} \tag{5.2}$$

The problem of maximizing lower bound sensing coverage $MLBSC$ is a min-max problem that can be reduced to an integer linear programming problem as:

$$\max_s\ \min_{e_{jk}} \left[1 - \prod_i \left(1 - t^e_{i,jk}\right)^{s_i} \right]$$

which reduces to

$$\min_{s} \max_{e_{jk}} \prod_{i} \left(1 - t^{e}_{i,jk}\right)^{s_i}$$

Since logarithm is a monotone increasing function,

$$\ln\left[\prod_{i} \left(1 - t^{e}_{i,jk}\right)^{s_i}\right] = \sum_{i} \ln\left(1 - t^{e}_{i,jk}\right) \cdot s_i$$

where $\log\left(1 - t^{e}_{i,jk}\right)$ are constants. The problem therefore reduces to:

$$\text{minimize } x$$

$$\text{such that}$$

$$\sum_{i=1\ldots n} s_i = c$$

$$\sum_{i} \ln\left(1 - t^{e}_{i,jk}\right) \cdot s_i \leq x \;\forall i = 1 \ldots n$$

$$s_i \geq \; 0 \;\; \forall i = 1 \ldots n$$

where x is a new variable introduced to convert a min-max problem to a linear program. The above problem is solved using the CPLEX mixed integer linear programming function.

5.1.2.2 Maximize Average Sensing Coverage (MASC)

We define Average Sensing Coverage, ASC for edges as the expected number of edges to be visited by at least one sensor of the configuration divided by total number of edges. For every edge e_{jk}, we introduce the indicator random variable, $\chi_{s,e_{jk}}$, that takes the value 1 if the edge e_{jk} is visited by the configuration of sensors, and the value 0 otherwise.

$$\chi_{s,e_{jk}} = \begin{cases} 1 & \text{with probability } p = PV(s, e_{jk}) \\ 0 & \text{with probability} 1 - p \end{cases}$$

Sensing coverage is therefore formally defined as $\mathbb{E}\left[\sum_{\forall e_{jk}} \chi_{s,e_{jk}}\right]$, the expected number of edges visited by sensors in the configuration s. Due to the linearity of expected value

$$ASC = \mathbb{E}\left[\sum_{\forall e_{jk}} \chi_{s,e_{jk}}\right] = \sum_{\forall e_{jk}} \mathbb{E}\left[\chi_{s,e_{jk}}\right]$$

$$ASC = \sum_{\forall e_{jk}} PV(s, e_{jk}) \tag{5.3}$$

The problem of maximizing ASC is formulated below.

maximize ASC i.e.,

$$\begin{aligned} &\text{maximize } \sum_j \left[1 - \prod_i (1 - t^e_{i,jk})^{s_i}\right] \\ &\text{such that} \\ &\quad \sum_{i=1\ldots n} s_i = c \\ &\quad s_i \geq \ 0 \ \ \forall i = 1 \ldots n \end{aligned} \tag{5.4}$$

5.1.3 Algorithms/Heuristics to Solve the MASC Problem

The objective in the problem of maximizing average sensing coverage ASC is written as

Algorithm 7 Greedy for MASC problem

```
Require: $n$, $c$, $t^e_{i,jk} \forall e_{jk}$
 1: initialize $s_i$ as 0
 2: for $k = 1$ to $c$ do
 3:     $max = 0$, $insertAt = 0$
 4:     for all $v_i \in V$ do
 5:         increment $s_i$
 6:         if $\sum_{e_{jk}} \prod_i \left[1 - (1 - t^e_{i,jk})^{s_i}\right] > max$ then
 7:             $max = \sum_{e_{jk}} \prod_i \left[1 - (1 - t^e_{i,jk})^{s_i}\right]$
 8:             $insertAt = v_i$
 9:         end if
10:         decrement $s_i$
11:     end for
12:     increment $insertAt$ element in $s$
13: end for
```

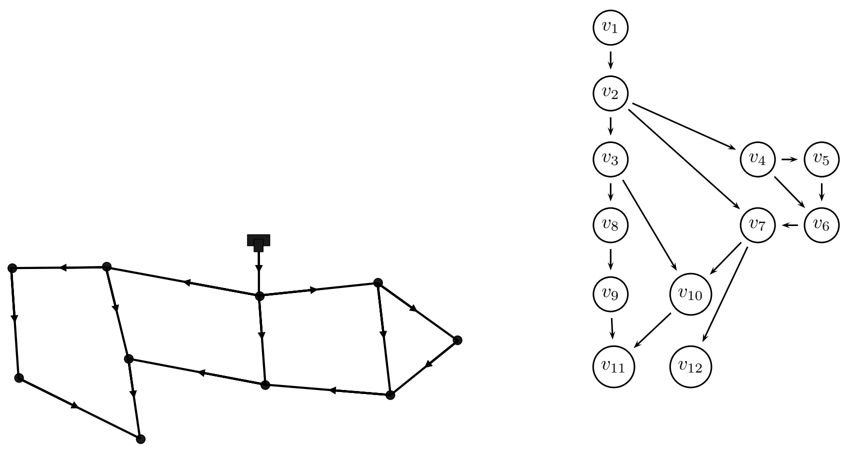


Figure 5.2: (a) Example network from EPANET; (b) Graph representation of the EPANET example network from (a)

$$\text{minimize} \sum_j \left(\prod_i \beta_{ij}^{s_i} \right)$$

which is a nonlinear convex programming problem. The $MASC$ problem is also NP-Hard by reduction from the Weighted Maximum Coverage Problem. The construction and proof of NP-Hardness is analogous to that of the proof for the MNS problem in Appendix A. We solve this problem using a heuristic as described in Algorithm 7 for the integer optimization problems. The algorithm starts with an initial configuration in which no sensors are inserted (line 1) and insert one sensor at a time (line 12), with insertion done at the node that would generate the best value of the objective function given the configuration of sensors already in place (lines 4-11).

Example: To understand the optimization problems and their solutions, we present a sample 12 node network generated from EPANET [27], as shown in Figure 5.1.2.2. This network can be simplified and represented as a graph as shown in Figure 5.2(b). At each junction, the flows are equally distributed in all the out-going edges, i.e., $p_{23} = p_{24} = p_{27} = \frac{1}{3}$, $p_{45} = p_{46} = \frac{1}{2}$, etc. Here, the zone of interest includes all the edges.

The problems are solved with the constraint $s_1+s_2+\ldots s_{12} = 10$. The solution to the MLBSC problem, solved using CPLEX is $\{6, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0\}$, and the lower bound sensing coverage achieved is 0.9095. The solution to the MASC problem, solved using the greedy heuristic in Algorithm 7 is $\{5, 1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0\}$ and the average sensing coverage is 0.9405.

5.2 Beacon Placement Problem for On-demand and Continuous Monitoring

Beacons are important to collect information from sensors, as well as to aid in localization. However, placing beacons everywhere in the WDS is costly, and inefficient. Similar to sensor deployment, we solve beacon placement problem from two perspectives. The first is where we minimize the number of beacons to achieve a certain localization measure. The second is where we are given a certain number of beacons, and we wish to achieve the best localization measure with the given number of beacons. We also consider two different objectives for beacon placement, one based on number of edges, and the other based on the radius of localization. Also, we present the problems of sensor deployment and beacon placement in a single problem formulation, where we assume that we have a budget constraint and we allot the budget to purchase sensors and beacons.

5.2.1 Minimization of Number of Beacons to Ensure Localization Error

We define *Localization Error*, LE as the radius of the region where an event is *suspected* to be present. When the presence of an event in the FBS is determined, we will need to determine the location of the event so as to attend to it in a timely fashion. We remark here that localization error does not refer to the range of the actual event. It only refers to the resolution of the localization.

Algorithm 9 determines the beacon placement such that bounded localization error LE is ensured in a zone of interest I, i.e., a subgraph of $\mathcal{F}$. First, we define a variable $v_i.\phi$ for a vertex v_i that determines the localization error of a sensor that reaches v_i from the sources. We place beacons to reduce ϕ at all vertices, thereby ensuring that LE is bounded. $v_i.\phi$ is updated through Algorithm 9. The algorithm initializes the required data structures in lines 1-4. *Sources* refers to the source vertices in I, i.e., have an in-degree 1. We perform a breadth first search of the graph, adding vertices to the queue in a topological order, so that we can determine all paths from sources to sinks. While we traverse these paths, we place beacons when ϕ exceeds LE. In this algorithm, we keep track of every path in I from insertion points to sinks and edges leading out of the zone of interest. Each path is assigned an id. The list of paths that a vertex v_i is part of is maintained in a map *pathsOn*. The list of edges in every path is maintained in a map *pathSummary*, called the

Algorithm 8 Beacon Placement with Localization Error requirement

```
Require: $LE$, $I$, $\mathbf{CR}_b$, $\mathbf{SR}$, $J$
 1: Initialize $pid \leftarrow 0$, $pathsOn$, $pathSummary$, $Q \leftarrow Sources \in I$
 2: for all $s \in Sources$ do
 3:     $pathsOn(s) \leftarrow pid$; $pid++$; $s.\phi = 0$
 4: end for
 5: while $Q$ is not empty do
 6:     $v_i \leftarrow deque(Q)$
 7:     for each $path \in pathsOn(v_i)$ do
 8:         $pathSummary(path).append(v_i)$
 9:     end for
10:     if $v_i.children$ is empty then
11:         Place beacon at $v_i$
12:     end if
13:     for each $v_j \in v_i.children$ do
14:         if $v_j$ is first child of $v_i$ then
15:             $pathsOn(v_j).append(pathsOn(v_i))$
16:         else
17:             for $path \in pathsOn(v_i)$ do
18:                 $pathsOn(v_j).append(pid)$
19:                 $pathSummary(pid) \leftarrow pathSummary(path)$; $pid++$
20:             end for
21:         end if
22:         $v_j.\phi \leftarrow v_j.\phi + v_i.\phi + \sum_k \sum_l \mathbf{SR}(e_k, e_l)$, where $e_k \in$ edges with $v_j$ as terminal
            vertex, and $e_l$ such that $\mathbf{CR}_b(v_j, e_l) = 1$
23:         if $\neg v_j.queued$ then
24:             $Q.enque(v_j)$
25:         else if $v_j.\phi \geq LE \wedge v_j \notin J$ then
26:             Place beacon at $v_j$
27:         end if
28:     end for
29: end while
```

path summary.

A breadth first search of the graph is done in lines 5-29. At each iteration in the BFS, the path summaries of each path containing the vertex is updated (Lines 7-9). Since beacons are not only identifiers, but also collection points for information from sensors, we need to place them at all sinks. If the vertex has no children (i.e., sink vertex), we place a beacon at this point for collection (Line 11). If a vertex has no out-edges that lead into I, it is considered a pseudo-sink. Since the input to this algorithm is I, such vertices have no children. If a vertex has only one child, the child vertex is added to the path summary *pathSummary* of every path containing the parent vertex (Line 15). No new paths are added. If the vertex has more than one child, each child vertex is in a different path. The path summaries for multiple paths are updated in lines 17-20. ϕ of each child vertex is updated in line 22. If a child vertex was not already queued, it is added to the BFS queue in line 24. If ϕ of the vertex exceeds LE, i.e., $v_i.\phi \geq LE$, a beacon is placed at the child vertex (Line 26). This step also updates the potential of the vertex for the beacon.

We place beacons at vertices only when $\phi > LE$. Since vertices are considered in a topological order, optimal beacon placement (excluding sinks) in the subgraph at any iteration is optimal with a new vertex added, as long as the topological order still holds. Hence, when the algorithm terminates, optimal beacon placement is complete.

5.2.2 Minimization of Number of Beacons to Ensure Localization Accuracy

The *Probability of Detection / Event Localization Accuracy* (P_d) is the probability of finding an event (or the accuracy of event localization) in zone of interest I. Formally, $P_d = \frac{TP+TN}{TP+TN+FP+FN}$, where TP, TN, FP and FN are true positives (i.e., an event existed and the algorithm detected it), true negatives (i.e., an event did not exist and the algorithm correctly indicated a non-existence), false positives (i.e., an event did not exist, but the algorithm detected one) and false negatives (i.e., an event existed and the algorithm failed to detect it), respectively.

Our solution to the problem of optimal placement of beacons (i.e., reduce the number of beacons), so that probability of event detection P_d is met is as follows.

The Beacon Placement algorithm is presented in Algorithm 9. This algorithm

Algorithm 9 Beacon Placement with Localization Accuracy requirement

```
Require: $P_d$, $G(V, E)$,
 1: $Q \leftarrow V.sources$
 2: $V.sources.place\_bcn$
 3: $\tau \leftarrow (1 - P_d)|E|$
 4: while $Q \neq \Phi$ do
 5:     $v_i \leftarrow deque(Q); no\_bcns \leftarrow \Phi$
 6:     for each $p_j \in v_i.parents$ do
 7:         if $\neg\ p_j.completed$ then
 8:             $Q.insert(p_j)$
 9:         end if
10:         if $\neg\ p_j.has\_bcn$ then
11:             $no\_bcns.add(p_j)$
12:         end if
13:     end for
14:     while $v_i.\phi < \tau \wedge no\_bcns \neq \Phi$ do
15:         $p_j \leftarrow no\_bcns.GET\_MAX$
16:         $p_j.place\_bcn$
17:         $v_i.\phi \leftarrow v_i.\phi - p_j.\phi - 1$
18:     end while
19:     if $v_i.\phi > \tau$ then
20:         $v_i.has\_bcn \leftarrow true$
21:     end if
22:     for each $v_j \in v_i.children$ do
23:         $v_j.\phi \leftarrow v_j.\phi + v_i.\phi + 1$
24:         if $\neg v_j.queued$ then
25:             $Q.enque(v_j)$
26:         end if
27:     end for
28:     $v_i.completed \leftarrow true$
29: end while
```

optimizes the placement of beacons in the network, so that the event localization algorithm can achieve a P_d accuracy. Consider the directed acyclic graphs $G(V, E)$ and $G_{rev}(V, E')$ where the only difference between G and G_{rev} is the direction of their edges. More formally, G and G_{rev} have the same set of vertices and $\forall\ (u, v) \in E$, $(v, u) \in E'$ and $\forall\ (u, v) \in E'$, $(v, u) \in E$ where $u, v \in V$. The algorithm uses an approach similar to Breadth First Search on a directed acyclic graph $G(V, E)$. In Line 1, a vertex queue Q is initialized with sources of $\mathcal{F}$. In Line 2, a beacon is placed in all sources of $\mathcal{F}$.

Every vertex has a potential ϕ, where ϕ for a vertex v_i is $max(|Path_{ij}|)$, where $|Path_{ij}|$ is the number of edges in the set of paths from v_i to a beacon B_j in G_{rev}, such that each path contains at most one beacon. For example, in Figure 3.6, $v_3.\phi = 0$, since beacon B_2 is placed at v_3. If there was no beacon at v_3, then $v_3.\phi$ would be $max(|\{(v_3, v_4)\}|, |\{(v_3, v_7), (v_7, v_8)\}|, |\{(v_3, v_2), (v_2, v_1)\}|) = 2$. If v_j is a parent of a vertex v_i in G, then intuitively $v_i.\phi > v_j.\phi$ unless v_i has a beacon. Hence, we can iteratively obtain ϕ for a vertex, using ϕ of its parents. When a beacon is placed at v_j, v_i's potential will decrease. At the end of beacon placement, every vertex should have a potential less than a threshold to ensure accuracy of event localization. A threshold τ for ϕ is derived from P_d in Line 3.

Lines 4-29 iterate over the vertices of a graph using Breadth First Search (BFS). If the parent of a vertex v_i was not iterated over, the vertex is added back to the queue, with a priority, in Line 8. This is because we cannot make an informed decision about beacon placement in v_i without knowing the potential value $v_i.\phi$. We maintain a heap for the parents of v_i that do not have beacons, with key as $p_j.\phi$, in Line 11. Once we check all parents of v_i, we are sure that the potential of v_i is correctly computed. Now we start placing beacons at the parent vertices of v_i until the potential of v_i decreases below τ. Parents are selected using a greedy approach, so that as few parents as possible have beacons, as shown in Lines 14-18. If $v_i.\phi$ is still greater than τ, a beacon is placed at v_i. Lines 22-27 add the children of v_i to the queue, similar to Breadth First Search. Once a vertex is iterated over, it is marked as completed in Line 28. Since we consider directed acyclic graphs, Line 8 will not introduce an infinite loop.

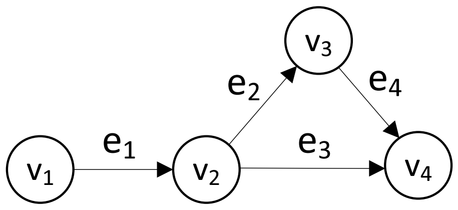


Figure 5.3: Small example to demonstrate beacon placement and event localization

Example: Consider the graph shown in Figure 5.3. At each iteration, the ϕ for one edge is updated in a breadth first search order. For $P_d = 0.5$, the threshold τ is 2 edges. ϕ will be updated and beacons will be chosen as shown below. At first a beacon is placed in v_1 and $Q = \{v_1\}$.

1. $v_1.\phi = 0$ $v_2.\phi = v_3.\phi = v_4.\phi = \infty$. Now $Q = \{v_2\}$
2. $v_1.\phi = 0$ $v_2.\phi = 1$ $v_3.\phi = v_4.\phi = \infty$. Now $Q = \{v_4, v_3\}$ and v_4 is dequeued. In line 7, it becomes evident that $v_3 \in v_4.parents$ is not completed. So, now $Q = \{v_3, v_4\}$ and v_3 is dequeued.
3. $v_1.\phi = 0$ $v_2.\phi = 1$ $v_3.\phi = 2$ $v_4.\phi = \infty$. Now v_4 is dequeued.
4. $v_1.\phi = 0$ $v_2.\phi = 1$ $v_3.\phi = 2$ $v_4.\phi = 4$. $v_4.\phi$ is greater than τ. So, parents of v_4 are selected greedily. A beacon is placed in v_3. Now, ϕ for each vertex becomes:
5. $v_1.\phi = 0$ $v_2.\phi = 1$ $v_3.\phi = 0$ $v_4.\phi = 2$. Now, $Q = \Phi$. Algorithm terminates. We can now see that no vertex has ϕ greater than τ.

This algorithm provides an optimal solution to the Beacon Placement problem for directed acyclic graphs, since we ensure optimal result for each subgraph of G. The time complexity of this algorithm depends on the number of times a vertex is added back in the queue and the number of parents a vertex has. Adding and removing parents from heap takes $O(\lg n)$ time, where n is the number of parents. A vertex can be added back to the queue at most $O(V)$ times. There is no cyclic dependency because the graph is directed and acyclic. The number of parents of a vertex is also $O(V)$. Consequently, the Beacon Placement problem is in P, and our algorithm has $O(V^3(\lg V))$ time complexity.

5.2.3 Jointly Determining Sensor and Beacon Placement with Budget Constraint for Leak Detection

Probability of covering an edge: A major goal in WDS monitoring for leak/backflow events is to narrow down the suspected area, as it is closely related to the scale of the problem and the time and labor it takes to pinpoint and eliminate the events. The design objective of our monitoring system is to minimize the scope of suspected area quantified as system localization error.

Imagine a mobile sensor inserted in vertex v_{11} in Example 3.7 which travels to v_{22} through vertex v_{12}. If there are beacons at vertices v_{11} and v_{22}, and a leak/backflow in edge $e_{11,12}$, it is reported by the sensor through communication with beacons placed at vertices v_{11} and v_{22}. Any of the four pipes between v_{11} and v_{22} are now suspected to contain the reported leak/backflow. But, if there was an additional beacon present at v_{12}, we would be able to narrow down the location of the leak/backflow to one pipe.

Consider the problem of leak/backflow detection as an example where $SR = 0\text{m}$. Beacons placed at junctions can localize the event by collecting information from the sensors passing by. The probability that a pipe contains leaks/backflows depends on its age and operation conditions. Leak probability of a pipe is denoted by l_{ij}.

Edge localization error: We define *edge localization error*, or localization error of an edge, denoted by $le(\mathbf{S}, e_{ij})$, as the product of its leak probability and uncovered probability, i.e. $l_{ij} \cdot u^e_{ij}$. The upper bound of all edge localization error is denoted by $ubele$.

For clarity, we first introduce two concepts, *inner vertices* and *inner edges*, before giving the definition of *beacon localization error*. *inner vertices* is defined between two vertices v_i and v_j in the network, and it consists of all vertices which a sensor may pass by if it travels from v_i to v_j, excluding v_i and v_j. Similarly, *inner edges* includes those edges which the sensor may go through during its travel.

Beacon localization error: The *beacon localization error*, or localization error of a leak/backflow between two adjacent beacons, say b_i and b_j, denoted by le_{ij}, is defined as the sum of localization errors of all *inner edges* between v_i to v_j.

The average beacon localization error, denoted by ale, is calculated by:

$$ale = \sum_{(b_i,b_j)} \frac{le_{ij}}{bp\_num} \tag{5.5}$$

where (b_i, b_j) is a pair of beacons and bp_num is the number of beacon pairs.

System localization error: The metric system localization error, denoted by sle, is calculated as:

$$sle = AchSC \cdot ale + (1 - AchSC) \sum_{e_{kh}} l_{kh} \tag{5.6}$$

where $AchSC$ is the achieved average sensing coverage, i.e., $\sum_{\forall\ e_{ij}} PV(\mathbf{S}, e_{ij})$.

For each edge, say e_{ij}, we introduce the indicator random variable, $\chi_{S,e_{ij}}$, that takes value 1 if leak/backflow present on edge e_{ij} cannot be detected with a specific configuration of sensors $\mathbf{S}$, and value 0 otherwise.

$$\chi_{S,e_{ij}} = \begin{cases} 1 & \text{with probability } p = le(\mathbf{S}, e_{ij}) \\ 0 & \text{with probability} 1-p \end{cases}$$

The expected system localization error, denoted by *esle*, is calculated by:

$$else = \mathbb{E}\left[\sum_{\forall e_{ij}} \chi_{S,e_{ij}}\right]$$

which is equal to the system localization error when no beacons are placed. Due to the linearity of expected value, we have $\mathbb{E}\left[\sum_{\forall e_{ij}} \chi_{S,e_{ij}}\right] = \sum_{\forall e_{ij}} \mathbb{E}\left[\chi_{S,e_{ij}}\right]$, then $esle = \sum_{\forall e_{ij}} le(S, e_{ij})$.

Optimization Problem

The optimization objective function is defined as the minimization of system localization error represented by Equation 5.6. The decision variables include the number of mobile sensors and stationary beacons used, and their deployment locations. The respective number of sensors and beacons is subject to the available budget. The constrained optimization problems is thus mathematically formulated as:

$$\text{minimize } sle \tag{5.7}$$

subject to:

$$sp \cdot \sum_r s_r + bp \cdot \sum_r b_r \leq cost \tag{5.8}$$

$$s_r \in \{0, 1, \ldots, \lfloor \frac{cost}{sp} \rfloor\} \tag{5.9}$$

$$b_r \in \{0, 1\} \tag{5.10}$$

where *cost* is the available budget for purchasing sensor and beacon devices, *sp* is the price of a mobile sensor, *bp* is the price of a beacon and $r = 1, \ldots, n$. In all the equations in this section, the indices range from 1 to n, unless specified otherwise. The first constraint represents the cost budget constraint and the last two constraints define variables s_r to be integer and b_r to be binary.

The joint sensor and beacon placement optimization presented in Section 5.2.3 is a computationally intensive problem. Hence, we simplify the formulation to solve the sensor and beacon placement jointly. However, the problem continues to be nonlinear and computationally intensive. To this end, we will define two alternative formulations that separately optimize sensor and beacon placement for a given cost.

Joint Formulation

Minimization of *sle* is a complex problem, e.g., in our experiments, finding the optimal solution for a WDS with over 50 junctions in not possible even in weeks. An alternative approach used here is minimizing *ale* instead, which is a less complex problem and can be solved in a reasonable time.

We note here that the *sle* is $AchSC \cdot ale + (1 - AchSC) \sum_{e_{kh}} l_{kh}$. The term $AchSC$ in the *sle* formulation ensures that we place emphasis on false negatives in leak/backflow detection as well. Another notable point is that *ale* is also dependent both on the sensor and beacon configurations. The benefit of minimizing *ale* as opposed to the minimization of *sle* is not as harmful, as we see in the Performance Evaluation section.

The set $\{(i,j)|p_{ij} > 0\}$ includes all non-independent vertex pairs in graph G, denoted by CN. The set $\{(i,j)'|p_{ij} > 0, \text{or } i == j\}$ is denoted by CN'. In order to express *ale*, we introduce a binary variable c_{ij} to represent the presence ($c_{ij} = 1$) and absence ($c_{ij} = 0$) of beacons between vertices v_i and v_j including vertex v_j, defined as:

$$\begin{cases} c_{ii} = 1 \\ c_{ij} = 0, (i,j) \notin CN \\ c_{ij} = 1 - \prod_{e_{kj}} (1 - c_{ik} \cdot (1 - b_j)), (i,j) \in CN \end{cases} \tag{5.11}$$

Then, the localization error le_{ij} can be expressed as:

$$\begin{aligned} le_{ij} = &\sum_{e_{kh}} (c_{ik} \cdot c_{kh} \cdot conn_{hj} \cdot l_{kh} \cdot u^e_{kh}) \\ &+ \sum_{e_{gj}} c_{i,g} \cdot l_{gj} \cdot u^e_{gj} \end{aligned} \tag{5.12}$$

where $conn_{hj} = (1 - \prod_{e_{gj}} (1 - c_{hg}))$ which is 1 only if there exists a neighbor v_g of vertex v_j such that c_{hg} is 1.

The joint formulation is a Mixed Integer Non-Linear Programming ($MINLP$) formulation and is given as:

$$\text{minimize } ale \text{ i.e.,}$$

$$\text{minimize } \frac{\sum_{(i,j) \in CN} [le_{ij} \cdot b_i \cdot b_j]}{(\sum_r b_r)^2 - \sum_r b_r} \tag{5.13}$$

subject to the same constraints represented by Equations 5.8, 5.9, and 5.10.

Disjoint Formulation

Although many MINLP solvers [30] [11] have been developed and can be used for solving the joint formulation, their time complexity is exponential to the problem core size (it takes several days to solve the problem for 100 vertices). A possible idea to reduce time complexity of the problem is to split and solve the sensor and beacon placement problems separately.

Considering the computational intensity of *ale* minimization problem, we define two alternative formulations that separately solve the sensor and beacon placement optimization problems.

The function le_{ij} in the original formulation is a monotonically decreasing function (i.e., if $i' > i$ and $j' > j$, then $le_{i'j'} \leq le_{ij}$). A monotonic optimization problem achieve optimum solution at boundaries when constraints set is convex. The only constraint $sp \cdot \sum_{r=1}^{n} s_r + bp \cdot \sum_{r=1}^{n} b_r \leq cost$ in our formulation is convex. Therefore a separate solution will not sacrifice the optimality of the results.

The number of cases satisfying $sp \cdot \sum_{r=1}^{n} s_r + bp \cdot \sum_{r=1}^{n} b_r = cost$ is linearly related to the network size because the number of beacons is the upper bounded by the number of vertices in the network. The outline of the searching process of the disjoint method is shown in Algorithm 10. The algorithm iteratively splits the total cost among sensors and beacons starting with no beacons and adds one beacon at a time (Lines 2 and 5). For each s and b, we solve the sensor and beacon problems separately by methods described later in this section (Lines 6 and 7). During the iterations, we record $\mathbf{S}$ and $\mathbf{B}$ that achieve the least *sle* (Lines 10 - 14).

The alternative disjoint formulations and algorithms for sensor deployment problem and beacon placement problem are presented in the following subsections.

Sensor Deployment Problem

Two objectives of the sensor deployment problem are to minimize *esle* and to minimize *ubele*. We present formulations and solutions for the two objectives this subsection.

Greedy Heuristic for Minimizing the Expected System Localization Error (MESLE)

The problem of minimizing *esle* is formulated as follows:

Algorithm 10 Exhaustive Searching Algorithm

```
Require: $n$, $cost$, $sp$, $bp$
1: initialize both $bmax$, and $b$ as 0
2: initialize $local\mathbf{S}$ and $local\mathbf{B}$ with all $s_i$ and $b_i$ as 0
3: $min = INF, localSle = INF$
4: for $bmax = 1$ to $n$ do
5:     $smax = (cost - bmax \cdot bp)/sp$
6:     solve sensor deployment problem
7:     solve beacon placement problem
8:     update $localSle$ using the Equation 5.6
9:     if $localSle < min$ then
10:        $min = localSle$
11:        $s = smax$
12:        $b = bmax$
13:        $\mathbf{S} = local\mathbf{S}$
14:        $\mathbf{B} = local\mathbf{B}$
15:    end if
16:    increment $bmax$
17: end for
```

minimize *esle* i.e.,

$$\text{minimize} \sum_{e_{ij}} \left[le_{ij} \cdot u^{e}_{ij} \right] \tag{5.14}$$

subject to:

$$\sum_{r=1\ldots n} s_r = s$$

$$s_r \geq 0 \quad \forall r = 1 \ldots n$$

The objective in the problem of minimizing *esle* is written as:

$$\text{minimize} \sum_{e_{ij}} \left(le_{ij} \cdot \prod_{r} \beta_{r,ij}^{s_r} \right)$$

This is a convex nonlinear programming problem and we solve it using a greedy heuristic similar to the MASC problem, by Algorithm 7.

When a utility manager is only interested in detecting the presence of the leak/backflow, and not in localizing it, only the sensor deployment problem needs to be solved. In this case, we define the Average Sensing Coverage (ASC) as the average probability of covering any edge, i.e., $\sum_{e_{ij}} \left[1 - u_{ij}^e\right]$. The MESLE problem is then reduced to maximizing the average sensing coverage problem MASC. If the leakage/backflow probability is not considered, we solve the MASC problem using the greedy algorithm with all the leak probabilities set to to 1. The approximation ratio of greedy algorithm for MASC is $(1 + \frac{1}{e-1})$ (See Appendix B).

Integer Linear Programming for Minimizing the Upper Bound Edge Localization Error (MUBELE)

As defined before, $ubele$ is the smallest number such that $\forall\ e_{ij} \in E$, $[le(S, e_{ij})]$ $\leq ubele$. The formulation for minimizing $ubele$ is as follows:

$$\begin{gathered}
\text{minimize } ubele\text{, i.e.,}\\
\text{minimize } \max_{e_{ij}} \left[\left(le_{ij} \cdot (1 - u_{ij}^e)\right)\right]\\
\text{subject to:}\\
\sum_{r=1\ldots n} s_i = s\\
s_r \geq\ 0\ \ \forall r = 1 \ldots n
\end{gathered} \tag{5.15}$$

The MUBELE problem is a min-max problem that can be reduced to an integer linear programming problem (IP) as follows:

$$\min_{s}\ \max_{e_{ij}} \left[le_{ij} \cdot u_{ij}^e\right]$$

Taking the logarithm of the objective function, we get $\ln(le_{ij} \cdot u_{ij}^e) = ln(le_{ij}) +$

$\sum_r s_r \cdot \ln(\beta_{ij})$, Since logarithm is a monotone increasing function and $ln(le_{ij})$, $\log(\beta_{ij})$ are constants, the problem can be reduced to:

$$\begin{aligned} &\text{minimize } x \\ &\text{subject to} \\ &\sum_r ln(le_{ij}) + s_r \cdot \ln(\beta_{ij}) \leq x \\ &\sum_r s_r = s \\ &s_r \geq \ 0 \end{aligned}$$

where x is a new variable introduced to convert a min-max problem into a linear program.

Similar to MLBSC, when only sensor deployment is considered, the coverage of a pipe becomes synonymous to detection of a leak/backflow in that pipe. The Lower Bound Sensing Coverage ($LBSC$) is defined as the minimum probability of covering any edge, i.e., $LBSC$ is the largest number such that $\forall\ e_{ij} \in E$, $\left[1 - u^e_{ij}\right] \geq LBSC$. The problem of MUBELE is then transformed into maximizing the lower bound of covering any edge (MLBSC).

Beacon Placement Problem

Once the sensor deployment problem is solved, the joint optimization problem is reduced to the beacon placement problem. We can further reduce the problem to a linear programming problem by linearizing the equations for c and le_{ij} in Equation 5.11 and 5.12, respectively, as follows:

$$\begin{cases} c_{ii} = 1 \\ c_{ij} = 0, (i,j) \notin CN \\ c_{ij} \geq c_{ik} - b_j, (i,j) \in CN \\ c_{ij} \leq 1 - b_j, (i,j) \in CN \end{cases} \tag{5.16}$$

The linearized c_{rh} is equal to 1 when there is no beacon along the path from vertex r to vertex h. Its value is uncertain otherwise.

The localization error le_{ij} is linearized based on the AMGM inequality:

$$\begin{aligned} le_{ij} = &\sum_{e_{kh}}^{n} \left(\frac{c_{ik} + c_{ih} + conn_{kj}}{3}\right) \cdot le_{kh} \\ &+ \sum_{g} c_{ig} \cdot le^{e}_{gj} \end{aligned} \tag{5.17}$$

where $conn_{kj} = (\sum_{e_{gj}} c_{kg})/n$.

We define the linearized *ale* minimization, denoted by MLALE, as:

$$\text{minimize} \sum_{(i,j)\in CN} \left[\frac{le_{ij}}{bmax^2 - bmax}\right] \tag{5.18}$$

and define the linearized *uble* minimization, denoted by MLUBLE, as:

$$\text{minimize} \max_{(i,j)\in CN} \left[\frac{le_{ij}}{bmax^2 - bmax}\right] \tag{5.19}$$

The two separate solutions we propose are two different combinations of sen-

Table 5.2: Optimization problems used in this dissertation

Method	**Objective**	**Constraints**
joint	minimizing *ale*	Budget
MESLE	minimizing *esle*	Budget
MUBELE	minimizing *ubele*	Budget
MLALE	minimizing *ale*	Budget
MLUBLE	minimizing linearized *ubele*	Budget
greedy	Algorithm 10 with MESLE and MLALE	Budget
linear	Algorithm 10 with MUBELE and MLUBLE	Budget
MASC	Maximizing *ASC*	Sensor budget
MLBSC	Maximizing *LBSC*	Sensor budget
MNS	Minimizing number of sensors	*LBSC*

sor placement and beacon placement formulations. The former, *greedy*, combines MESLE and MLALE and the latter, *linear*, integrates MUBELE and MLUBLE.

All the optimization problems used in this dissertation are summarized in Table 5.2.

5.2.4 Beacon Placement for Continuous Monitoring

The above algorithms for beacon placement assume that the WDS is monitored on-demand. Beacon placement algorithm to ensure a certain localization error/localization accuracy is harder for a continuous monitoring case. In on-demand monitoring, the graph of the WDS is acyclic. Hence, computing all the paths in the WDS is simple.

With continuous monitoring, we cannot assume any directions for the edges in the graph. Therefore, to extend the above algorithms for continuous monitoring, we need to find all the paths between any pair of vertices in an undirected graph. Enumerating all the paths in an undirected graph is a sharp-P problem [93].

We therefore consider both a directed acyclic graph generated using the average used demands, $G(V, E)$, and $G_{rev}(V, E')$, where $\forall\ (u, v) \in E$, $(v, u) \in E'$ and $\forall\ (u, v) \in E'$, $(v, u) \in E$, where $u, v \in V$ (i.e., the only difference between G and G_{rev} is the direction of their edges). Since Algorithms 9 and 13 rely on the input graph G being directed, we run the algorithm with G and G_{rev} as inputs and the union of the beacons generated is used to place beacons in G.

Algorithm 11 Shortest paths to Zone of Interest

```
Require: $P_d$, $G(V, E)$
 1: $Q \leftarrow$ all $V \in I$
 2: $depth(v_i) = 0\ \forall\ V \in I$
 3: while $Q \neq \Phi$ do
 4:     $v_i \leftarrow deque(Q)$;
 5:     for each $v_j \in v_i.children$ do
 6:         if $\neg v_j.queued \wedge v_j \notin I$ then
 7:             $Q.enque(v_j)$
 8:         end if
 9:         if $depth(v_j) > depth(v_i) + 1$ then
10:             $depth(v_j) = depth(v_i) + 1$
11:         end if
12:     end for
13: end while
```

Ensuring the same event localization error and accuracy throughout the WDS may not be necessary, since we are usually interested in monitoring a particular zone of interest. Therefore, once the P_d requirement for the zone of the interest is set, we set P_d for the other set of edges to be $\frac{P_d}{\gamma_i}$, where γ_i is the ratio of the number of edges in the shortest path from edge e_i to any edge in the zone of interest and the number of edges in the zone of interest. The farther away a edge is from the zone of interest, the lesser is the P_d requirement of the edge. A similar approach is used for LE.

We note here that the beacon placement is computed only once for every zone of interest. Also, the number of edges in the shortest path between any pair of edges is not dependent on the zone of interest. Hence, we pre-compute the number of edges in the shortest path between every pair of edges (as described in Algorithm 11), thereby reducing the computation time when the zone of interest is changed. The algorithm performs a breadth first search (Lines 3-13) starting from the vertices in the zone of interest (Line 1). For each vertex dequeued in the BFS, the depth of its children is updated to reflect the shortest path (Line 10).

5.3 Event Localization Algorithm

Once the data from the sensors is collected at beacons, there is a need to parse that data to collect information about the events. In this section, we design an algorithm to deduce the location of the events using the knowledge of the flows in the WDS, beacon locations, and data collected from sensors. Based on the two beacon placement objectives, event localization algorithms are also of two types.

The localization of an event (presented in Algorithm 12) is based on data collected at beacons with the objective of minimizing localization error. A summary of events and beacons encountered, (i.e., *path synopsis* PS) is collected by beacons from sensors. The Event Localization Algorithm deduces the list of edges that possibly contain the event (list labeled EE) to determine the localization error. The achieved localization error is the radius of the circle covering the midpoints of edges EE.

We classify path synopsis collected by a beacon based on time stamps in a list, EL (Lines 1-2). EL is the set of data recorded with the same timestamp. If EL contains and event and a beacon (Lines 3-4), we add edges from columns of $\mathbf{SR}$ corresponding to all edges in CR_b of the beacon to the EE. We also mark those edges as *suspect* (Lines 5-12). If EL contained an event, but did not contain a beacon, the edges in the path between the previously seen beacon P and the beacon seen next N, are

Algorithm 12 Event Localization

```
Require: $d$, $\mathcal{F}$, $\mathbf{CR}_b$, $\mathbf{SR}$, $B$, $PS$
 1: for each $PS_i$ in $PS$ do
 2:   for each $EL$ in $PS_i$ do
 3:     if $X \in EL$ then
 4:       if $B_i \in EL$, where $B_i \in B$ then
 5:         for all $e_k$ such that $(\mathbf{CR}_b(B_i, e_k) = 1)$ do
 6:           for all $e_j$ do
 7:             if $\mathbf{SR}(e_j, e_k) = 1$ then
 8:               $EE$.append($e_k$)
 9:               $e_k$ marked as *suspect*
10:             end if
11:           end for
12:         end for
13:       else
14:         for all $e_k \in$ edges $e_i$ between $P$ & $N$ do
15:           for all $e_j$ do
16:             if $\mathbf{SR}(e_j, e_k) = 1$ then
17:               $EE$.append($e_k$)
18:               $e_k$ marked as *suspect*
19:             end if
20:           end for
21:         end for
22:       end if
23:     else
24:       for all $e_k$ such that $(\mathbf{CR}_b(B_i, e_k) = 1)$ do
25:         for all $e_j$ do
26:           if $\mathbf{SR}(e_j, e_k) = 1$ then
27:             $e_k$ marked as *sensed*
28:           end if
29:         end for
30:       end for
31:     end if
32:   end for
33: end for
34: $EE$.remove(Edges $\in$ *sensed*; $\notin$ *suspect* )
```

suspected to contain the event (Line 14-21). If no event was present in EL, the edges in **SR** are marked as *sensed* (Line 24-30). After iterating over data from all sensors, we remove the edges from EE that were not identified as *suspect*, but were *sensed* (Line 34).

Example: In the example in Figure 5.1, the paths taken by the sensors are: n_1 - v_1 v_2 v_4 v_6 v_7; n_2 - v_1 v_3 v_5 v_6 v_8; n_3 - v_2 v_3 v_5 v_6 v_7 and n_4 - v_0 v_3 v_5 v_6 v_7. If there is an event X present on edge e_2 and sensors n_1, n_2 and n_4 reach v_6 at the same time, n_1, n_2, n_3 and n_4 report the following information: $(n_3, t(n_1)_1)$ $(B_3, t(n_1)_2)$ $(n_2, t(n_1)_3)$ $(n_4, t(n_1)_3)$ $(B_1, t(n_1)_4)$, $(X, t(n_2)_1)$ $(n_4, t(n_2)_2)$ $(X, t(n_2)_2)$ $(n_1, t(n_2)_2)$ $(B_2, t(n_2)_3)$, $(n_1, t(n_3)_1)$ $(X, t(n_3)_2)$ $(B_1, t(n_3)_3)$ and $(n_2, t(n_4)_1)$ $(X, t(n_4)_1)$ $(n_2, t(n_4)_2)$ $(n_1, t(n_4)_2)$ $(B_1, t(n_4)_3)$, respectively where $t(n_i)_j$ refers to the time at which node n_i recorded the j^{th} report. Based on the knowledge of expected time to traverse each path, we can locate the event on edge e_2. The LE is $|e|/2 = 50$m.

The localization of an event (presented in Algorithm 13) is based on data collected at beacons with the objective of minimizing localization accuracy.

The algorithm for Event Localization is presented in Algorithm 13. In Line 1, we initialize *Suspects List* (i.e., edges where an event might be present) to contain all edges in the network. We follow an elimination method to localize events to as few edges as possible. In Line 2, we initialize a Beacons Table (BT). Each entry in the BT contains, for each pair of beacons, the number of paths and the list of edges between them, and an indication of whether an event is present or not between them. The number of paths and list of edges between each pair of beacons is obtained from the graph. The event indicator is initialized to $false$. Next, in Lines 3-15, we iterate over all sensors to analyze their path synopses. For each entry in the path synopsis p of a sensor n_i, Line 5 checks if no event was detected. If no event is detected between two beacons, and there is only one path between them, then the edges in that path definitely do not have an event. Hence, Line 8 eliminates such edges from *Suspects List*. If an event is found in the path synopsis, we mark an event in the

Algorithm 13 Event Localization

```
Require: $PS$, $N$, $G(V, E)$
 1: $Suspects\ List \leftarrow E.$
 2: $BT \leftarrow$ initialize Beacon Table
 3: for each $n_i \in N$ do
 4:    for each $p \in PS_i$ do
 5:       if $p \neq X$ then
 6:          if $BT[p][p.next].path = 1$ then
 7:             for each $e_j \in BT[p][p.next]$ do
 8:                $Suspects\ List.remove(e_j)$;
 9:             end for
10:          end if
11:       else
12:          $BT[p][p.next].event$
13:       end if
14:    end for
15: end for
16: for each $b_i \in BT$ do
17:    if $b_i.event = false$ then
18:       for each $e_j \in b_i$ do
19:          $Suspects\ List.remove(e_j)$;
20:       end for
21:    else
22:       $b_i.event$
23:    end if
24: end for
```

corresponding BT entry, in Line 12.

Upon iterating over all path synopses obtained from all sensors, the BT entries will reflect whether or not an event was detected on a path between pairs of beacons. Consequently, in Lines 16-24 we iterate over the entries in BT. An entry in the BT will be marked for an event only if one of the sensors detected an event between the beacons for that entry. If the entry in BT is not marked with an event, Line 19 removes edges between those beacons from *Suspects List*. At the end of the iteration, we will be left with the smallest possible *Suspects List*, i.e., the highest event localization accuracy.

The time complexity of this algorithm depends on the number of sensors, the number of beacons in each path synopsis and the number of edges between any two beacons. The number of edges between any two beacons is $O(E)$. Number of sensors is $O(V)$ and number of Beacons in the Path Synopsis is also $O(V)$. The worst case time of the algorithm is $O(V^2E)$.

Example: Consider again the flow network in Figure 3.6. Between source v_6 and sink v_3, there are 6 possible paths. When sensing coverage is ensured, sensors are inserted in such a way that all these paths are covered. Without loss of generality (since we solve here the event localization problem), we can assume that all sensors were inserted in the source. Let there be an event in edge (v_5, v_4). The sensed paths SP_i of the sensors (and their path synopsis PS_i) are:

$SP_1 = \{v_6, v_1, v_2, v_3\}$ with $PS_1 = \{B_1, B_4, B_2\}$;

$SP_2 = \{v_6, v_1, v_5, v_4, v_3\}$ with $PS_2 = \{B_1, B_4, X, B_3, B_2\}$;

$SP_3 = \{v_6, v_5, v_4, v_3\}$ with $PS_3 = \{B_1, X, B_3, B_2\}$;

$SP_4 = \{v_6, v_8, v_5, v_4, v_3\}$ with $PS_4 = \{B_1, B_5, X, B_3, B_2\}$;

$SP_5 = \{v_6, v_8, v_7, v_4, v_3\}$ with $PS_5 = \{B_1, B_5, B_3, B_2\}$;

$SP_6 = \{v_6, v_8, v_7, v_3\}$ with $PS_6 = \{B_1, B_5, B_2\}$.

In the first part of the algorithm, the following edges are removed: (v_6, v_1), (v_1, v_2), (v_2, v_3), (v_6, v_8), (v_8, v_7), (v_4, v_3), (v_7, v_4), (v_7, v_3). Next, we use *BT* entries, but we cannot remove more edges. So finally, in the *Suspects List*, we have (v_1, v_5), (v_8, v_5), (v_5, v_4), (v_6, v_5).

We remark here that if we know that there was only one event in the network, we can localize the event more precisely by taking only the common edges from the *BT* entries that have events. In the above example, we can reduce *Suspects List* to

(v_5, v_4), thereby achieving 100% success.

5.4 Global View Algorithm

Static beacons in the network provide an external perspective to the network dynamics. When a leader comes in range of a beacon, it transfers all its data to the beacon. The beacons have a global view of the network. At a given point of time, the directions of flows in a network can be approximated using monthly bills and usage patterns. Based on this knowledge, the topology of the WDS, and nodes encountered by other beacons, the beacons provide the set of possible group splits, group merges, and beacon encounters over time.

The beacons periodically broadcast a $BEACON - HELLO$ message to indicate their presence to the leader nodes. When a leader hears a $BEACON - HELLO$ from a beacon, it transmits *allData* packet containing all the data collected by its sensor and from the followers, including data from group merges. $\mathcal{F}$ is a time varying graph of the network. At any instant of time, the beacons are aware of the snapshot of $\mathcal{F}$. At each step of the breadth first search, while adding nodes to the queue, the next edge is determined based on the time varying graph, rather than a snapshot.

Algorithm 14 describes the algorithm used by the beacons. The beacons periodically send $BEACON - HELLO$ to enable leaders to identify their presence (lines 2-3). Upon receiving a message from leaders, they follow the procedure as shown in lines 4-26. The algorithm is an adaptation of the breadth first search (BFS) for a time varying graph. Starting at the beacon vertex, the algorithm performs a BFS on the time-varying graph until beacons are reached. New vertices are added to BFS queue based on how long it takes for the nodes to traverse the edges (lines 15-25). Unlike traditional BFS, the same vertex may be visited repeatedly due to the varying flows. Infinite loops are avoided by using a time limit.

Algorithm 14 Beacon B at vertex v_i

```
Require: $vertex$, $\mathcal{F}$, $reportedData$
while true do
  if time % period = true then
    Broadcast $BEACON - HELLO$
  end if
  if message received from leader then
    $G(V, E) = \mathcal{F}(time)$
    $Q = (v_i, 0)$
    while $Q$ not empty do
      $(v, t) \leftarrow Q.dequeue()$
      if In-degree of $v\ >\ 1$ then
        Add $(v, t)$ to $groupMerges$
      end if
      if Out-degree of $v\ >\ 1$ then
        Add $(v, t)$ to $groupSplits$
      end if
      if Beacon at $v$ then
        Add $(v, t)$ to $nextBeacons$
      end if
      for each $c \leftarrow$ child of $v$ in $G$ do
        $t' \leftarrow t$
        $e(t') \leftarrow (c, v) \in E$, $d_e(t') = length(e)/2$
        while $d_e(t') > 0$ or $< length(e)$ do
          $t'_{prev} \leftarrow t'$
          $t' \leftarrow$ time after $t'$ when flows change
          $d_e(t') \leftarrow$ distance covered on $e(t')$ from $t'_{prev}$ to $t'$
        end while
        if $d_e(t') < 0 \wedge$ no beacon at $v$ then
          $Q.enqueue(v, t')$
        end if
        if $d_e(t') > length(e) \wedge$ no beacon at $c$ then
          $Q.enqueue(c, t')$
        end if
      end for
    end while
    Transmit ($groupSplits$, $groupMerges$, $nextBeacons$)
  end if
end while
```

At each vertex visited, if the in-degree is greater than 1, there is a possible group merge; if the out-degree is greater than 1, there is a possible group split; finally, if there is a beacon, the leader will get to communicate with it. This information is stored in three data structures, namely, $groupMerges$, $groupSplits$, and $nextBeacons$. This information is then sent to the leader node that broadcasts it to the group. Based on this data, the leaders choose their communication schedules, which are also broadcast.

6. CONTROL IN CPWDS*

In this section, we present the control aspects of our CPWDS. It includes a flow estimation algorithm required for flow-unaware monitoring, a sensor position estimation algorithm, and a control system design to modify the flows in the WDS.

Table 6.1: List of symbols used in this section

Symbol	**Definition**
G_{est}	The flow network with estimated flows in the WDS
$\mathcal{S}$	Sampling function to sample from a discrete probability distribution
$X(t)$	The two dimensional state of the sensors at time step t
$X_{i0}(k)$	The edge that sensor i is present on at iteration k
$X_{i1}(k)$	The fraction of the edge covered at iteration k
Q_i^*	Desired flows in edge e_i

Table 6.1 contains all symbols used in this section.

6.1 Flow Estimation

Up until this point, we assumed that flows in the network edges are known. In the real world (i.e., a real water distribution system), due to the usage/flow dynamics, the flows (i.e., directions and magnitude) are not known precisely. In this section we present a solution which relaxes our assumption about known flows in WDS.

We propose a solution in which an estimate of flow is initially derived, based on knowledge about the network flow topology and average usage patterns (e.g., utility

*Parts of this section are reprinted with permission from "A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems" by Suresh, M.; Manohar, U.; R, Anjana. G.; Stoleru, R.; Mohan Kumar, M.S., IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 Copyright © 2014 IEEE providers have access to household average water usage). Then, the actual flows in the network are learned, based on events and encountered beacons, reported by sensors.

Based on the monthly bills of users, an average demand at the consumers is available. Additionally, the pattern of the demands throughout the day may also be estimated. Using these values, estimates of flows for any part of the day are generated.

Mathematically, the problem flow learning is defined as: given $G(V, E)$, $D = \{d_i \mid \forall v_i \in V\}$ (i.e., the demand at each vertex of the network), and $c(u, v) \forall u \in V$ and $v \in V$ (i.e., the capacities of all edges), derive $\mathcal{F}(u, v) \forall u \in V$ and $v \in V$, the flows in all edges. This problem is precisely the computation of the maximum flow in a network. To estimate these flows, each sink is replaced by an edge. The demands at the end points set the flows in those edges. We know the capacities of the network edges. Hence, a max-flow algorithm can be used to compute the flows in all edges. To approximate the flows in all edges based on the maximum flow in the network, we use the Edmonds-Karp algorithm [26]. Considering the graph example in Figure 5.3, let's set the capacities of all edges be 5 units. If the demand in v_4 fixes the total incoming flow to 8 units, the flow will be divided by the Edmonds-Karp algorithm in v_2 and v_3 as 3 units and 5 units, respectively.

If the levels of water in reservoirs and tanks are also known, EPANET may be used to derive the estimated flows in greater accuracy, since it also accounts for energy losses as the water flows through the pipes.

In flow-unaware on-demand monitoring, the beacon placement based on estimated flows may not be optimal if the expected flows are different from the actual flows in the WDS. Hence, we will use the beacon placement for continuous monitoring here. Since Algorithm 12 and 13 for Event Localization does not depend on direction or

Algorithm 15 Flow Learning

```
Require: $G_{est}(V, E)$
 1: $PS \leftarrow$ Collected path synopses.
 2: clear $unseen$
 3: $unseen \leftarrow BT$
 4: for each $n_i \in N$ do
 5:   for each $p \in PS_i$ do
 6:     Remove $BT[p][p.next]$ from $unseen$
 7:     if $edges(BT[p][p.next]) = 0$ then
 8:       $flows\_to\_change \mathrel{+}= BT[p][p.next]$
 9:     end if
10:   end for
11: end for
12: for each $bp$ in $BP$ do
13:   if $bp \in flows\_to\_change$ then
14:     Reverse flows on $bp.edges$
15:   end if
16:   if $bp \in unseen$ then
17:     Reduce edge chances on $bp.edges$
18:   end if
19: end for
20: Update $G_{est}$ with the new flows.
21: Repeat from Line 1.
```

magnitude of flows, it is not affected by knowledge about flows.

6.1.1 Flow Learning

So far, we have approximated the flow in each edge of the graph. Now, we use information collected by sensors to learn flows in the network. This step can be repeated several times, i.e., through multiple deployments in on-demand monitoring.

The key intuition for how we derive flows is as follows. Consider an undirected graph. Between any two beacons, there is a fixed number of paths/edges that do not include another beacon. Consequently, the direction of some edges between the two beacons is inferred by the order in which the beacons are sensed by sensors.

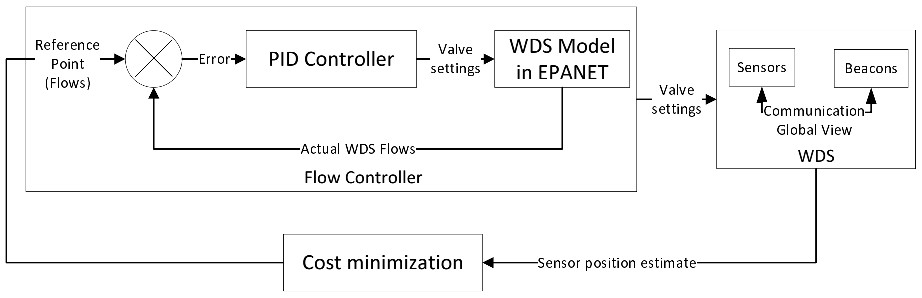


Figure 6.1: Block Diagram of the CPWDS

For example a path synopsis B_1B_2 suggests that B_2 was sensed after B_1. Thus, the directions of edges between B_1 and B_2 are inferred. When flows in the system do not change, after several deployments in on-demand monitoring, or over several hours in continuous monitoring, all flow directions are inferred. While flows are inferred, the insertion points, the number of deployed sensors, and the placement of beacons are decided using the new inferred flows after each on-demand deployment.

The steps for learning the flows, are described in Algorithm 15. Lines 1-3 initialize the algorithm. We collect the Path Synopses from beacons. Lines 4-11 iterates over the information collected on all sensors and records unseen beacon pairs and flows that are reversed. Lines 12-19 then iterate over all possible beacon pairs. Based on the information collected by Path Synopses, the flows in the estimated graph are altered in line 20. The same procedure is repeated as given by Line 21. The flow learning algorithm will reduce the difference between estimated flows and the actual flows in the pipes of the network.

6.2 Sensor Position Estimation

To estimate the position of sensors, we will use the reports collected at beacons. The sensors send neighborhood information to beacons and the neighborhood information of sensors is updated whenever there are group splits and group merges. At a group split, when the leader election happens, the nodes update the nodes that were no longer in range of them.

Algorithm 16 TimeToExit(e)

```
Require: $e$, $f$
1: $f$ is the fraction of the edge $e_i$ covered
2: $d_e$ - Length of edge $e$
3: $v_e$ - Velocity of flow in edge $e$
4: $l = d_e/v_e$
5: Return $l \times (1 - f)$
```

Algorithm 17 Location update for a sensor s_i

```
Require: G, $X_i(k)$
1: Let $t = T$
2: $X_{i0}(k+1) = X_{i0}(k)$
3: while $t \neq 0$ do
4:   if $X_{i1}(k) = 1$ then
5:     if Sensor reaches beacon $B_x$ at vertex $v_y$ then
6:       $X_{i0}(k) = \mathcal{S}(v_y)$
7:     else
8:       $X_{i0}(k+1) = \mathcal{S}(\text{terminal vertex of} X_{i0}(k))$
9:     end if
10:    $X_{i1}(k+1) = 0$
11:  else
12:    if TimeToExit($X_{i0}(k+1)$) $< T$ (Algorithm 16) then
13:      $t = T$ - TimeToExit($X_{i0}(k+1)$)
14:      $X_{i1}(k+1) = 1$
15:    else
16:      $X_{i1}(k+1) = X_{i1}(k+1) + \frac{T}{TimeToExit(X_{i0}(k+1))}$
17:      $t = 0$
18:    end if
19:  end if
20: end while
```

For continuous monitoring of WDS, the insertion point of the sensors is inconsequential. Every sensor is moving through one of the *edges* of the WDS. If the sensor has reached the terminal vertex of an edge, it moves from that edge to another adjoining edge. Therefore, the transition probabilities of the sensors from an edge to another edge adjoining it, and from a vertex to the edges incident on it are important here. The terms used in the sensor position estimation algorithm are as follows.

The flows in the WDS are obtained from EPANET, a simulator designed to model WDS. The flows at a vertex are used to model the transition probability of a sensor from vertex to an adjoining edge. The probability distribution of the movement of a sensor at a junction is proportional to the flow distribution, as described in the mobility model. We define a function $\mathcal{S}$ that samples from the discrete probability distribution of movement of a sensor at a vertex. E.g., $\mathcal{S}(v_i)$ provides an edge that is sampled from the probability distribution at that vertex.

Example: Consider the graph shown in Figure 6.2. Suppose the single step transition probability of sensor movement are: $P[v_2|v_1] = 0.75$, $P[v_3|v_1] = 0.25$, $P[v_5|v_2] = 0.3$, $P[v_4|v_2] = 0.2$, $P[v_6|v_2] = 0.5$ $P[v_4|v_3] = 1$, $P[v_6|v_4] = 1$, $P[v_6|v_5] = 1$. $\mathcal{S}(v_1)$ picks e_1 with probability 0.75 and e_2 with probability 0.25.

The state of the system $X(t)$ contains the position of the sensors at time t. For every sensor, s_i, the state $X_i(k)$ consists of two variables $X_{i0}(k)$ that indicates the edge that the sensor is present on, and $X_{i0}(k)$ represents the fraction of the edge covered at iteration k. The system is considered to be discrete, with a sampling interval of T. The state of the sensors is updated as shown in Algorithm 17. Whenever a sensor reaches a beacon, as shown in Lines 5-6, the location is updated. Otherwise, the movement of the sensor for the time duration t is updated (Lines 11-19). Using the data about the WDS, we obtain the time spent in each pipe at any time as described in Algorithm 16.

For any time duration $[t, t + \delta]$, we may determine the edges covered during this interval as $\bigcup_{\forall s_i, k=t}^{k=t+\delta} X_{i0}(k)$. Our objective for continuous monitoring is to cover all the edges in a zone of interest in the interval δ.

6.3 Flow Controller

For continuous monitoring of a WDS, the reference point of the control system changes over time and needs to be computed for every *iteration*. An iteration in the continuous monitoring of a WDS contains a flow control, sensor movement, and sensor position estimation. Figure 6.1 shows the model of our CPWDS. As shown in the figure, the Flow Controller sets the valve settings in the WDS. Sensors are beacons in the CPWDS communicate among themselves. This communication, aided by the global view provided by the beacons, helps in estimating the position of the sensors. Given that we know the positions of the sensors, we obtain the reference point for the flow controller by observing the positions of the sensors at the beacons. The first step in this work is in obtaining a optimal target value Q_i^* for the flow controller, so as to minimize the control effort and control error. The obtained target flows only act as a reference point to a flow controller. The actual flow control by means of valve throttling, or source and sink control is achieved by the PID controller. Setting the reference point by observing the sensors ties the communication, computation, and control aspects of the CPWDS together.

Flow control is considered in two distinct ways: sensor-agnostic, and sensor-aware. In the sensor-agnostic flow control mechanism, we control the flows such the sensors do not leave the zone of interest, i.e., we do not need to track the positions of the sensors, since they will always stay in the zone of interest. In the sensor-aware flow control mechanism, we track the position of the sensors and use them to determine the desired flows in the WDS. We then use a single flow controlling mechanism based

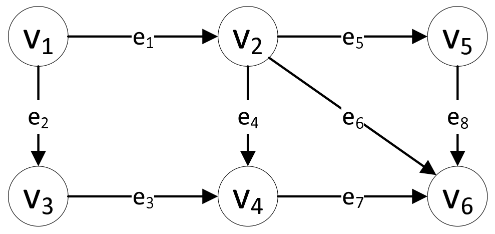


Figure 6.2: Sample graph

on the reference points set by either of the two flow control mechanisms.

6.3.1 Sensor-agnostic Flow Controller

The key assumption for sensor-agnostic flow control is that it is possible to maintain cycles in the WDS with the addition of pumps. The idea is that if there is a cycle that prevents sensors from leaving the zone of interest, we are ensured coverage of the zone of interest. Such a mechanism must also account for the varying demands of the WDS and tune the flow controller accordingly.

The objective for the Sensor-agnostic Flow Controller is the minimization of total pumping cost. The constraints for the controller are the flow and capacity constraints, and the constraint that defines the desired direction of flows. Desired direction of flow is defined for each edge, and can be +1, -1, or 0 (+1 indicates a default direction of flow, -1 indicates flow in the opposite direction, and 0 indicates indifference to the preferred direction of flow in the edge).

The WDS is observed at periodic intervals (e.g., hourly) to measure the demands and the flow rates at sources. The flow controller uses this information to maintain flow conservation and ensure that consumer demands are met. The desired direction of flow ensure that there is a cycle in the zone of interest, and the pipes that lead directly into the zone of interest have flows toward the zone of interest. This can be formulated as follows:

$$\text{minimize } pumpingCost \tag{6.1}$$

subject to:

$$A(t) \times f(t) = j(t) \tag{6.2}$$

$$B(t) \times f_d(t) \geq 0 \tag{6.3}$$

$$f(t) \leq c \tag{6.4}$$

where:

A is the incidence matrix of the zone of interest (dimension $V \times E$). a_ij is 1 if the flow on edge e_j is inbound from vertex v_i, and -1 otherwise.

f is a flow magnitude vector (dimension $E \times 1$). f_i is the magnitude of flow on edge e_i.

j is the demand vector (dimension $V \times 1$). j_i is positive if the vertex v_i is a sink, negative if vertex v_i is a source, and 0 otherwise.

B is a diagonal matrix with desired direction of flows (dimension $E \times E$). b_{ii} is the $+1$ if the direction of desired flow in edge e_i is the same as its predetermined direction, and -1 otherwise.

f_d is a flow vector (dimension $E \times 1$). f_i is the flow on edge e_i and f_i is positive or negative based on predetermined direction of flows.

t indicates the time step t.

The *pumpingCost* in the objective function refers to pumping cost, which may be assumed to be quadratic in f, i.e., $\sum_{\forall e_i} \frac{1}{2c_i} f_i^2$. Equation 6.2 is the flow conservation equation. For any vertex v_i, the sum of incoming flows (positive) and outgoing flows (negative) is the total demand at the vertex. Equation 6.3 ensures flows in the desired directions. Equation 6.4 ensures that the flows in the pipes do not exceed the maximum capacity on the pipes. The decision variables in the objective function is f. Solving the optimization problem provides the optimal desired flows in the zone of interest and neighboring edges of the WDS.

For example, in Figure 6.2, the edge e_4 is defined to be from v_2 to v_4. At vertex v_2, the sum of incoming flow should be equal to the sum of outgoing flow, i.e., $a_{21} * f_1 + a_{24} * f_4 + a_{26} * f_6 + a_{25} * f_5 = j_2$. Here, $j_2 = 0$. If we desire the flow to be from v_2 to v_4, $b_{44} = 1$, if we desire the flow to be from v_4 to v_2, then $b_{44} = -1$.

When the flows cannot be retained in a certain region of the WDS, and the position of the sensors are not tracked, the objective of a flow controller is to maximize the probability that any sensor in any part of the WDS reaches the zone of interest in T time.

Let the probability that a sensor moves from a edge e_i to edge e_j, M_{ij} be consolidated in a matrix $\mathbf{M}$. The probability that a sensor reaches a edge e_j starting at edge e_i at any time in the future is thus obtained as $\mathbf{T} = \sum_{k=1\ldots K} \mathbf{M}^k$, where K is the smallest number such that $\mathbf{P}^K = 0$. We note here that we may also set K to be the maximum number of transitions in T time (i.e., the time period for which we attempt to control the flow). We also define a function $outij$ as:

$$out_{ij} \begin{cases} 1 & \text{if end vertex of } e_i = \text{ begin vertex of } e_j \\ 0 & \text{otherwise} \end{cases}$$

Formally, the problem of maximizing the probability that sensors reach the zone of interest is represented as:

Algorithm 18 Sensor-aware Flow Controller

```
Require: $G$, $I$, $X$, $t$, $\delta$
 1: $\mathbf{C} = \bigcup_{\forall s_i} \bigcup_{k=t}^{t+\delta} X_{i0}(k)$
 2: $\mathbf{S} = \bigcup_{\forall s_i} X_{i0}(t+\delta)$
 3: Enqueue in $Q$, $\mathbf{S}$
 4: while $Q$ is not empty do
 5:     $q \leftarrow$ Dequeue from $Q$
 6:     $P \leftarrow$ shortest path from $q$ to $I - \mathbf{S}$
 7:     for all $e$ in $P$ do
 8:         Assign direction to $e$
 9:     end for
10: end while
```

$$max. \sum_{\forall j \in I} \sum_{\forall i \in E} \mathbf{T}_{ij}$$

$$s.t. \sum_{\forall j \in E} \mathbf{M}_{ij} = 1 \ \ \forall e_i \in E$$

$$\sum_{\forall j \in \text{edges adjoining to vertex } v_i} f_j = 0 \ \ \forall v_i \in V$$

$$f_i \leq capacity_i \quad \text{, for every edge } e_i$$

$$\sum_{\forall j} \mathbf{M}_{ij} = 1 \quad \text{, for every edge } e_i$$

$$where \ \mathbf{T}_{ij} = \sum_{k=1toK} \mathbf{M}_{ij}^k$$

and $\mathbf{M}_{ij}$ = flow f_j in edge e_j / sum of all out-flows from edge e_i, if e_j is adjacent to e_i, and the flow from e_i drains into e_j, and 0 otherwise. Formally,

$$\mathbf{M}_{ij} = out_{ij} \times \frac{f_j}{\sum_{\forall e_k \in E} f_k \times out_{ik}}$$

The decision variable is f_i. Although, deciding $\mathbf{M}_{ij}$ may also suffice. Both deciding f_i and $\mathbf{M}_{ij}$ are complex problem that are time consuming even for small networks.

6.3.2 Sensor-aware Flow Controller

Contrary to the sensor-agnostic flow controller, the sensor-aware flow controller allows the sensors to leave the zone of interest briefly. The flow controller then reverses the flows in some edges to bring the sensors back to the zone of interest. This method does not require additional pumps and may require merely throttling valves, and adding few tanks that act as sources or sinks interchangeably. As mentioned earlier, sensor positions are estimated by Algorithm 17. We observe the WDS periodically, similar to sensor-agnostic flow controller. But, in this case, we also estimate the sensor positions and update their locations periodically. For a period δ, we make the following measurement:

$\mathbf{C} = \bigcup_{\forall s_i} \bigcup_{k=t}^{t+\delta} X_{i0}(k)$ is the set of edges covered by all sensors in the interval $[t,\, t+\delta]$

Our objective for the next δ interval is to cover the remaining edges in zone of interest I (I is the set of edges in the zone of interest), i.e., $I - \mathbf{C}$. The set of edges on which the sensors are present at time $t+\delta$ is given by $\mathbf{S} = \bigcup_{\forall s_i} X_{i0}(t+\delta)$. We may now consider these edges as "insertion points" of the sensors for the next interval.

Let s_i be the number of sensors at vertex v_i at the time at which we start the flow control. We use the definitions of $\mathbf{T}$ and $\mathbf{M}$ from the previous section. Our objective is:

$$\text{maximize } \textstyle\sum_{\forall i}(1 - \prod_{j \in I}(1 - \mathbf{T}_{ij}))^{s_i} \tag{6.5}$$

subject to

$$\sum_{\forall j} \mathbf{M}_{ij} = 1 \text{ , for every edge } e_i$$

The above objective ensures that the probability that the sensors reach the zone of interest in the time horizon is maximized. The decision variables are $\mathbf{M}_{ij}$, and the objective is a non-linear function. However, similar to the sensor-agnostic controller, this problem takes too long to obtain results.

Therefore, a simple mechanism to ensure that the sensors cover the remaining area of the zone of interest is by setting the flow directions towards the zone of interest from edges in $\mathbf{S}$ towards $I - \mathbf{C}$ edges by following the shortest path (weights of the edges given by the cost function c_r), as described in Algorithm 18.

This algorithm attempts to assign direction to every edge in the path to reach the zone of interest. However, it may not be possible to ensure these flows. Also, since the movement of sensors is probabilistic, it is not sufficient to ensure that there is at least one path leading back to the zone of interest. We also need to ensure that the probability of traversing those edges is high.

As mentioned earlier, the sensor position estimation algorithm serves as an input to the flow controller. The flow controller tracks a reference input that is provided by solving an optimization problem to maximize the probability that sensors reach the zone of interest.

6.3.3 Flow Control

Pipes, pumps, reservoirs, valves etc. are the main components of a WDS. Pipes convey water from the source to users. As water moves along the pipe, its energy gets dissipated. Consider a pipe section with length l_p (m), diameter D_p (m), and area A_p (m^2). If the difference in head between two ends of a pipe section is considered as $\Delta(h)$, the nonlinear differential equation, describing the fluid flow behavior is: $\frac{dQ_p(t)}{dt} = gA_p(\Delta h - h_{loss}(t))$.

The total head loss in a pipe section is given as $h_{loss}(t) = h_{loss-fp}(t) + h_{loss-l}(t)$, where $h_{loss-fp}$ is the friction loss in pipe section and h_{loss-l} is the local friction loss in sections like bends, valves etc. Friction loss in pipe sections are usually calculated using Hazen-William equation or Darcy-Weisbach equation. According to Darcy-Weisbach equation: $h_{loss-fp}(t) = \left(\frac{f_p l_p}{2gD_pA_p^2}\right)Q_p^2(t)$. According to Hazen Williams equation: $h_{loss-fp}(t) = \left(\frac{10.71 l_p}{CHW^{1.852}D_p^{4.87}}\right)Q_p^{1.852}(t)$. Here, Q_p (m^3/s) is the flow rate in a pipe. In Darcy-Weisbach equation, f_p is the friction loss coefficient and in Hazen William equation, CHW is the Hazen-Williams roughness coefficient. The local friction losses mainly constitute valve loss, expressed as: $h_{valveloss}(t) = \left(\frac{K_v}{2gA_p^2}\right)Q_p^2(t)$, where K_v is the valve loss coefficient.

Pumps supply energy to water, balancing loss due to friction and elevation and are generally described by the head versus flow characteristics. The head characteristic of a variable speed pump is: $h_p(N, Q_p) = A_0N^2 + \frac{B_0}{n}NQ_p - \frac{C_0}{n^2}Q_p^2$. Here, $h_p(m)$ is the head, A_0, B_0 and C_0 are constants of a pump. N (rpm) is pump speed and n is number of pumps.

Reservoir storage enhances system flexibility, providing supplies for random fluctuations in demand. When a reservoir discharges under its own head without external pressure, the continuity equation is $\frac{d(V(t))}{dt} = Q_i(t) - Q_0(t)$, where Q_i(m^3/s) and Q_0(m^3/s) denotes the reservoir input and output water flow rates respectively and V (m^3) is the volume of a particular reservoir.

For any WDS, the above equations provide the system dynamics. All the above equations act as constraints in the controller. The controller must provide a solution that satisfies all of the above equations.

We propose a standard PID (Proportional-Integral-Derivative) controller, which is a form of lead-lag compensator with one pole at the origin and other at infinity. The objective of the PID controller is to determine an output based on the error between the desired set point and the actual value. The error is then used to adjust some input to the process so that the error gets minimized [25]. The PID Controller is generally represented as $u = K_p e + K_i \int edt + K_d \frac{de}{dt}$, where e is the error; u is the input to the system; K_p is the proportional gain; K_i is the integral gain; and K_d is the derivative gain. The gains of a PID controller are tuned using Zeigler-Nicholas, a commonly used method, to achieve the desired values.

A single valve or a combination of valves are throttled in order to reverse the flow in a particular pipe. That particular combination of valves is identified using trial and error method, and for valve throttling, a PID controller is implemented. For the PID controller, the pipe flows are considered as the state variable: $X(t) = [Q_1\ Q_2\ \dots Q_n]$, where n is the number of pipes. The control variables are the valve settings, $U(t) = [u_1\ u_2\ \dots u_m] = [K_{v_1}\ K_{v_2}\ \dots K_{v_m}]$, where m is the number of valves.

The goal of the controller is to change the direction flow in a specific pipe or a combination of pipes while satisfying the aforementioned constraints. The error is defined as the difference between actual (simulated) value Q_1 and the target value Q_1^* of the flows. The error vector E is represented as $E = [e_1\ e_2\ \dots e_n]$, i.e., $= [Q_1 - Q_1^*\ Q_2 - Q_2^*\ \dots Q_n - Q_n^*]$. Please note that when x valves are throttled, $x - n$ error values will be zero. The valve loss coefficient is then estimated using the PID equation $K_{v_i} = K_{p_i} e_i + K_{i_i} \int e_i dt + K_{d_i} \frac{de}{dt}$.

6.4 Sensor Controller

In this dissertation, we propose a sensor control mechanism to minimally steer the sensors in a on-demand flow aware (FLAW-D) monitoring mode. The assumptions we make for sensor control are:

1. The flows in the WDS are known and do not vary for a certain duration

2. We assume that sensors are inserted into edges, rather than vertices. At

sources, we add a virtual edge for insertion

3. We assume that there are beacons/devices that can provide control inputs to sensors at each junction

We note here that this is the first work to steer sensors in the WDS, and since the problem is not trivial, these assumptions need to be relaxed in further detailed studies.

The WDS is a complex non-linear system and the movement of sensors through this system is stochastic. To model the dynamics of sensor movement in the pipes of a WDS and to generate control inputs to steer them is a complex task in general. Therefore, we consider here a very simple model where it is assumed that all pipes of the WDS are traversed in the same time T (i.e., for an edge e_j of length l_j and velocity of the sensor in the pipe v_j, $\frac{l_j}{v_j} = T$).

Given the above condition, we design the state of the system at time k as $x(k)$, given by:

$$x(k) = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ x_{2,1} & 0 & \dots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{T,1} & 0 & \dots & x_{T,m} \end{pmatrix}$$

where $x_{i,j} \in \{0,1\}$ indicates the presence ($x_{i,j} = 1$) or absence ($x_{i,j} = 0$) of a sensor at distance $\frac{i \times l_j}{T}$ on edge e_j.

The sensor density function is defined as $f_{sum}(x) = 1_m * x$, where $1_m = (1_1, 1_2, \dots, 1_T)$ is 1*T vector, the coverage function is defined as $f_{cover}(u)$ where u is a 1*m vector and the $j-th$ element of u indicates the coverage status of edge e_j, given by:

$$u_j = \begin{cases} 1 & f_{sum}(x)_j > 0 \\ 0 & otherwise \end{cases}$$

We define the current system state as the measured output $y(k) = x(k)$, the feedback gain function of the transducer is defined as $h = f_{cover}(f_{sum}(y))$.

6.4.1 Sensor Controller Architecture

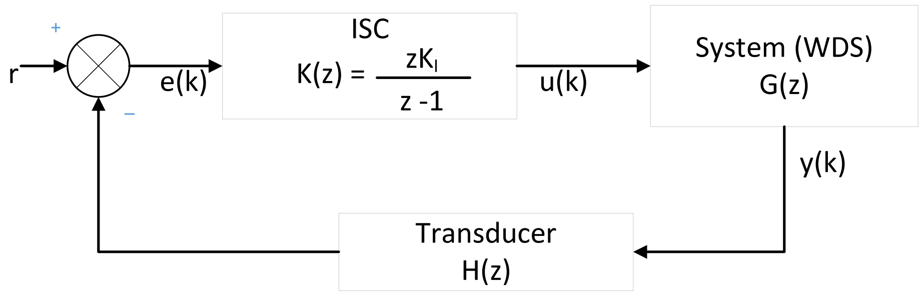


Figure 6.3: Design of the control system

As a baseline design we present the Integral Sensor Controller(ISC) which employs the linear quadratic regulator(LQR) control algorithm, as shown in figure 6.3.

The feedback control system is defined as follows:

$$y(k+1) = x(k+1) \tag{6.6}$$

$$e(k) = 1_m - h(y(k)) \tag{6.7}$$

$$e_I(k+1) = e_I(k) + e(k) \tag{6.8}$$

$$u(k) = u(k-1) + K_I e_I(k) \tag{6.9}$$

The cost function to be minimized in ISC is defined in Equation 6.10 below, including the cost of sensors $u_i + f_{sum}(y)$ and cost of control force u_i and u_r, where y is the state vector, u_i is the insertion control force and u_r is the rotation control force, Q and R are weighting matrices. The first part of the cost function is related to the cost of sensor density imbalances and the second part accounts for the power consumption for steering sensors.

$$J = (u_i + f_{sum}(y))^T * Q * (u_i + f_{sum}(y)) + u_r^T * R * u_r \tag{6.10}$$

where u_i is given by $u_i = (u_{i1}, u_{i2}, \ldots, u_{im})$ where $u_{ij} = 0, 1, 2, \ldots$ indicates the number of sensors passing through edge e_j, u_r is given by $u_r = (u_{r1}, u_{r2}, \ldots, u_{rm})$ where $u_{rj} = 0, 1, 2, \ldots$ indicates the power consumption for sensor at edge e_j steering at the end point of the edge according to the control input.

With Q and R defined, the control parameters K_I is determined by the LQR. Q and R are constructed based on cost models for control and resource imbalances which can be obtained through experiments. To enable real-time control, distributed control architecture where each beacon takes role of controller locally and traditional control method may be used to generate the control input. The connection between these local controllers is the control reference signal which indicates those edges need to be covered.

7. PERFORMANCE EVALUATION*

To evaluate our proposed models and algorithms, we present real world proof of concept implementations and results of our simulations in this section.

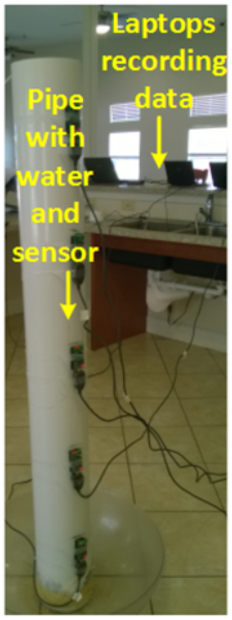
7.1 Proof-of-Concept System Implementation and Validation

We anchor our proposal to use mobile WSN for WDS monitoring into reality, through proof-of-concept system implementations comprising mobile sensor nodes in pipes and static beacons outside the pipes. We evaluate sensor-beacon communication in WDS using sensors placed in a water pipe and beacons placed outside as shown in Figure 7.1(a). We also present a testbed that uses air flow to propel sensors, as shown in Figure 7.1(b) to demonstrate sensing coverage and event localization.

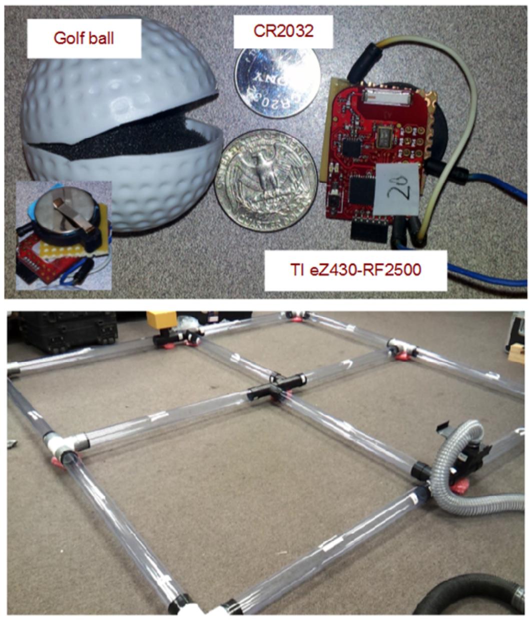
7.1.1 Experimental Results to Validate Communication Model

We validate our communication model through a testbed as shown in Figure 7.1(a). We tested RSSI on four pipes with diameters of $4'', 6'', 8'', 10''$ respectively. We use six sensors with CC2420 RF transceivers powered by Rayovac - AAA batteries for sensors and beacons. Five sensors are fixed outside the pipe at intervals of 0.3 meter,

*Parts of this section are reprinted with permission from "Towards Optimal Monitoring of Flow-based Systems using Mobile Wireless Sensor Networks" by Suresh, M.A. and Zhang, W. and Gong, W. and Rasekh, A. and Stoleru, R. and Banks, M.K., ACM Transactions on Sensor Networks, doi: http://dx.doi.org/10.1145/2700256, 2015 Copyright © 2015 Associated Computing Machinery, Inc., "A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems" by Suresh, M.; Manohar, U.; R, Anjana. G.; Stoleru, R.; Mohan Kumar, M.S., IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 Copyright © 2014 IEEE, "Mobile Sensor Networks for Leak and Backflow Detection in Water Distribution Systems" by Suresh, M.A.; Smith, L.; Rasekh, A.; Stoleru, R.; Banks, M.K.; Shihada, B., IEEE 28th International Conference on Advanced Information Networking and Applications (AINA), 2014 Copyright © 2014 IEEE, "On Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Zechman, E.M.; Shihada, B., IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013 Copyright © 2013 IEEE, "Towards Optimal Event Detection and Localization in Acyclic Flow Networks" by Suresh, M.A.; Stoleru, R.; Denton, R.; Zechman, E.; Shihada, B., IEEE International Conference on Distributed Computing and Networking (ICDCN), 2012 Copyright © 2012 Springer



(a)



(b)

Figure 7.1: (a) Testbed to validate communication range between sensors and beacons; (b) Testbed for system evaluation and a Sensor node.

and the sensor immersed under water is at the same level with the lowest sensor outside. The sensor in water broadcasts 3000 packets in 5 minutes for each experiment.

The average, maximum, minimum RSSI for each pipe are shown in Figure 7.2 (a)(b)(c). RSSI shows a logarithmic decrease with distance for all pipes. Although the value of RSSI changes greatly, it is in the range of -50dbm to 10dbm for all experiments, which can guarantee reliable communication.

From Figure 7.2 (d), we see that our log-distance path loss model can fit the experimental data well. $f(x)$ line in the figure fits the RSSI for $4''$ and $6''$ pipes, and $g(x)$ for $8''$ and $10''$ pipes. As we can notice, the pathloss exponents are 2.9 and 3.5. We remark here that, since we do not use soil around the pipes, the path loss in real WDS deployments may be higher. Therefore, in our theoretical models as mentioned earlier, we consider the path loss to be 4. Any inaccuracies will only lead to a low

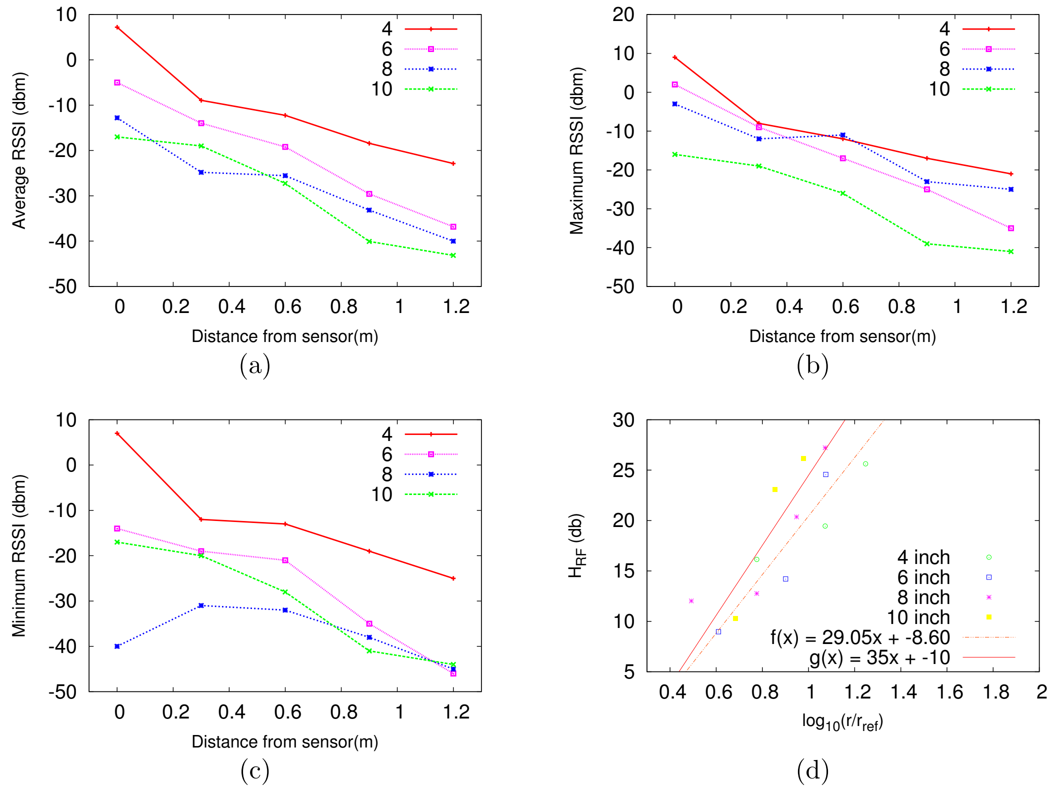
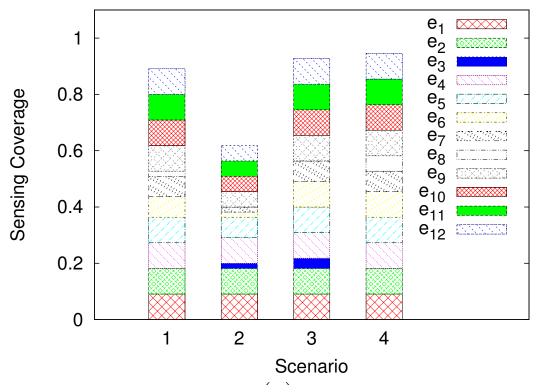


Figure 7.2: (a) Average, (b) Maximum, and (c) Minimum RSSI as the distance of the sensor from the beacon is varied. (d) The path loss for different pipe lengths and r. The r_{ref}s for 4″ pipe is 0.051m, 6″ pipe is 0.076m, 8″ pipe is 0.102m, 10″ pipe is 0.127m

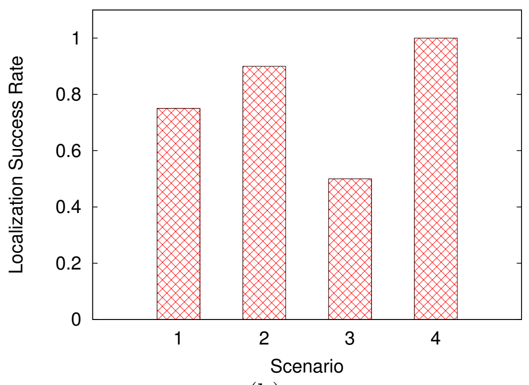
CR_b.

7.1.2 Proof-of-concept Testbed

For mobile sensors and beacons we use TI eZ430-RF2500 motes powered by CR2320 cell batteries. The components of a sensor are shown in Figure 7.1(b). The TI eZ430 and the battery can easily fit in a golf ball with 3.8cm diameter. A golf ball containing the mote and the battery, padded with sponge, can roll well inside the pipe. To generate network flow, we use an industrial vacuum cleaner. A beacon is a TI eZ430-RF2500 mote connected to a laptop.



(a)



(b)

Figure 7.3: (a) Sensing Coverage for 4 scenarios. (b) Localization success rate.

The sensor nodes and beacons are programmed (using the C language) with functionality as described in Sections 4-5 and presented in Figure 3.1. Due to the design of our testbed (i.e., the absence of acoustic communication), we employ only communication between sensors and beacons, which is RF in the 2.4GHz range (note here that testbeds in acoustic communications are complex to design and evaluate [3] [1] [22]). As described in Section 4, a beacon periodically broadcasts an RF-HELLO packet, containing its ID and the time, for time synchronization with sensor nodes. When a sensor node receives an RF-HELLO packet from a beacon, it broadcasts the collected data. Even if there is nothing to report, the sensor sends a HELLO message to the beacon. Events are emulated as RF signals by using motes that broadcast periodically. In our sensing model, r_{ref}, as described in Section 3.3 is 0m.

At the MAC layer, we use the IEEE standard 802.15.4 protocol. The mote backs off randomly and at the end of the backoff, if two consecutive clear channel assessments indicate a clear channel, the mote transmits. In our experiments we noticed a relatively short battery lifetime (∼10 minutes, due to reduced capacity of the CR2320 battery). We therefore use Low Power Listening (LPL) with a duty cycle

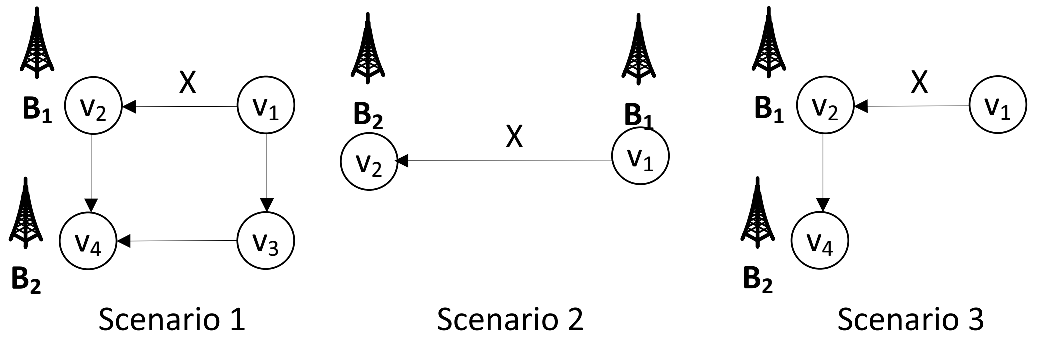


Figure 7.4: Scenarios for event localization. Scenario 4 is the same as Scenario 3, but with a longer CR_b. X indicates an event.

of 10% to prolong the battery life to ~30 minutes. The TI ez430-rf2500 platform has 32KB flash memory and 1KB RAM. Our implementation for the beacon nodes (~2,600 lines of code) uses 301B of RAM and 4,954B of program memory. The sensor node implementation (~3,000 lines of code) uses 610B of RAM and 5,530B of program memory.

To validate our Sensor Deployment Algorithm we ran four sets of experiments (i.e., scenarios) on the testbed shown in Figure 7.1(b). Each scenario had different insertion and collection points and were repeated 5 times with events placed on all edges. We used a total of 14 sensors for each experiment.

The results for the average sensing coverage for each scenario over 5 runs are shown in Figure 7.3 (a). The $e_1 \dots e_{12}$ correspond to the 12 edges in the WDS. Covering a single edge corresponds to a sensing coverage of $\frac{1}{12}$. The figure also shows the sensing coverage for each edge, averaged for the 5 runs. The expected FN in scenarios 1, 2, 3, and 4 with 14 sensors are 14.5%, 23.9%, 8.4% and 7.5%, respectively, obtained by estimating the flows on all edges in all scenarios. Edge e_3 in scenarios 1 and 4 and edge e_8 in scenarios 2 and 3 had very low flow. They were therefore not included in the zone of interest while calculating the sensing coverage and FN in 1, 4 and 2, 3 respectively.

To validate our Event Localization Algorithm, we ran 4 sets of experiments (i.e., scenarios) on a subset of the testbed shown in Figure 7.1(b). The four scenarios are shown in Figure 7.4. For each scenario, our results depicted in Figure 7.3 (b) represent averages of 10 experiments. The actual localization error achieved obtained by time stamps of reported data was 0.2m for scenarios 1 and 2, 0.64m for scenario 3, and 0.5m for scenario 4 on an average. We observe that the localization success rate (1 - FN, for false negatives due to failure in communication) was lower than 100% in Scenarios 1-3, when CR_b, the communication range of beacons, was small. Several factors contributed to this, such as sensor speed (the sensor crossed the beacon without waking up) and the structure of the pipes (bends, and irregular communication ranges). In Scenario 4 all events were successfully localized, due to the higher CR_b. *We therefore conclude that mobile sensors need to have a higher transmit power, especially if LPL is used, to ensure that communication failures do not lead to false negatives.*

7.2 Large Scale Simulation Platform: Cyber-Physical Water Distribution System Simulator (CPWDSim))

Since the evaluation of our algorithms and protocols in a real world WDS, or in a testbed is tedious and time consuming, we developed a simulator called CPWDSim. CPWDSim is a simulator specifically designed for WDS, as depicted in Figure 7.2. CPWDSim includes a *Network Model*, implementation of *Algorithms* and a *Simulator*. CPWDSim simulates the movement of sensors in pipes, given a Network Model, which contains the structure of the network and the flows. The Network Model is generated using: a) the model of a WDS in a city (e.g., for this we use Micropolis [10]); and b) flow rates, flow velocity and demands at end points in the network,

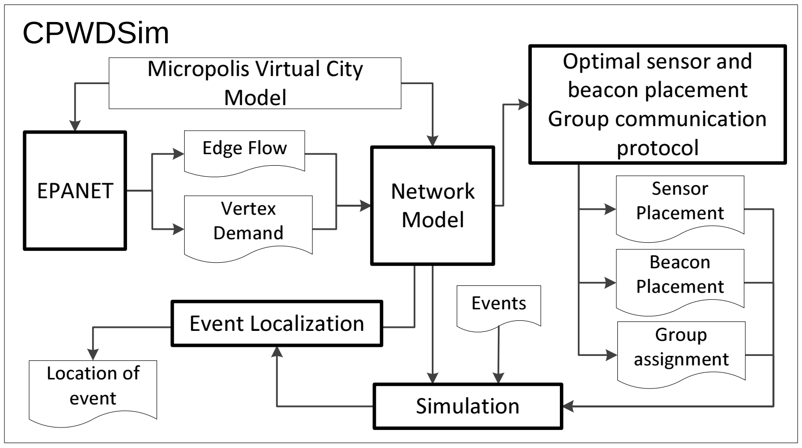


Figure 7.5: CPWDSim design

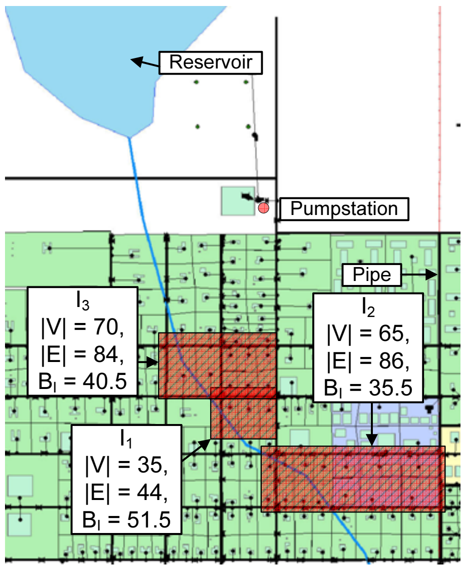
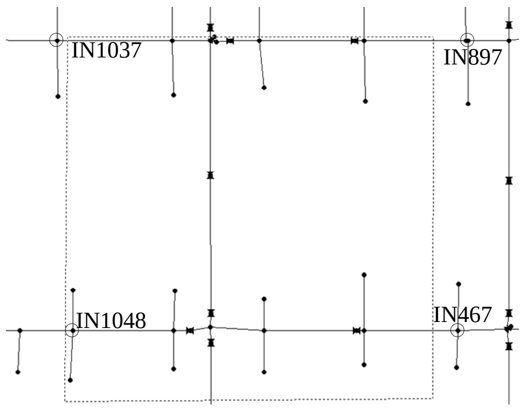


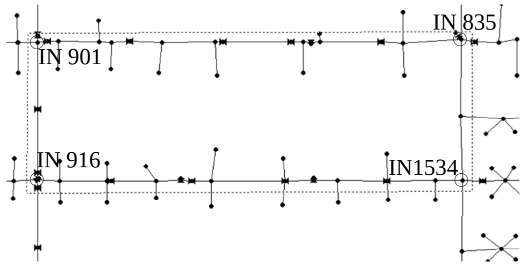
Figure 7.6: Micropolis [10] with zones of interests I_1, I_2, and I_3.

for a given time of the day (e.g., report files from EPANET [27]).

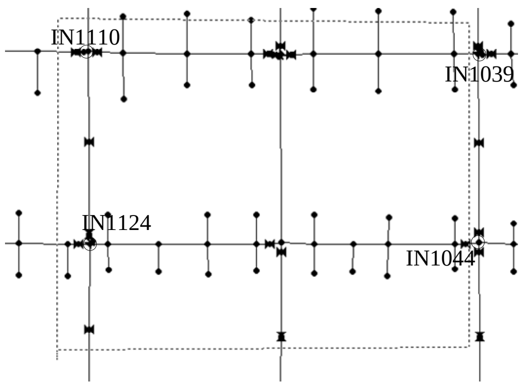
When the CPWDSim is started, it accepts the network model, flows, and demands as inputs from EPANET. These inputs initialize the *Network Model*. This network model is used to determine the placement of sensors and beacons, and the



(a)



(b)

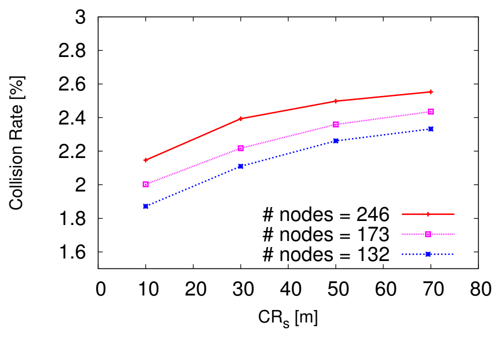


(c)

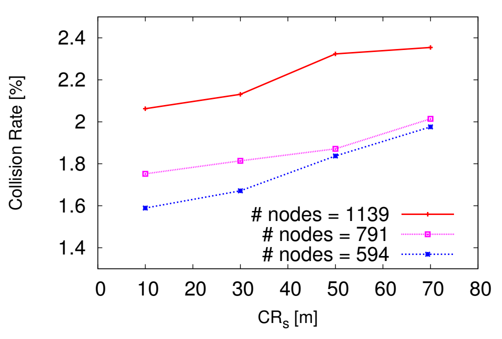
Figure 7.7: Zoomed in views of (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

communication parameters for the sensors (i.e., the p_i for each leader). During the simulation, the movement and communication of each sensor for one hour is computed. After each hour, if the user has chosen to control the flows, the algorithms relating to flow control are run. The flows for the next hour are either computed using valve settings from the flow controller, or read from a file. At the end of the simulation duration, the results are written into an output file.

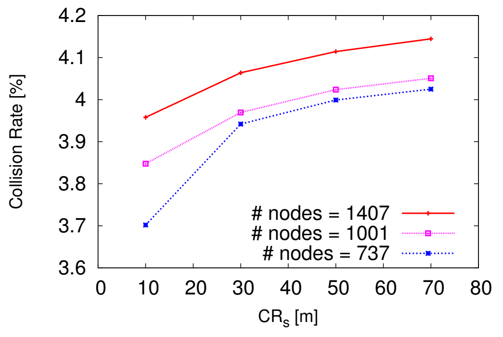
We evaluate our algorithms on three zones of interest, I_1, I_2, and I_3, as depicted in Figure 7.2 (zoomed in structures of the zones are in Figure 7.7).



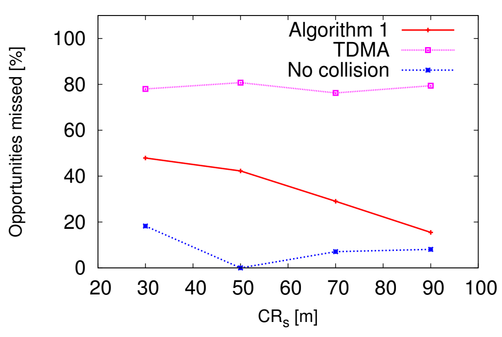
(a)



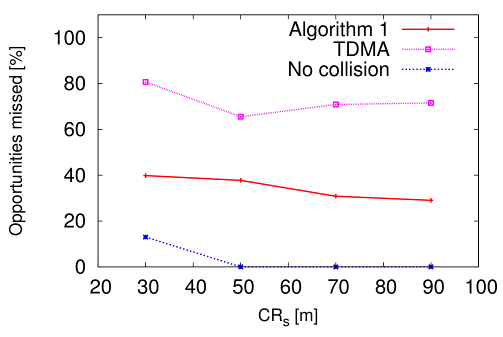
(b)



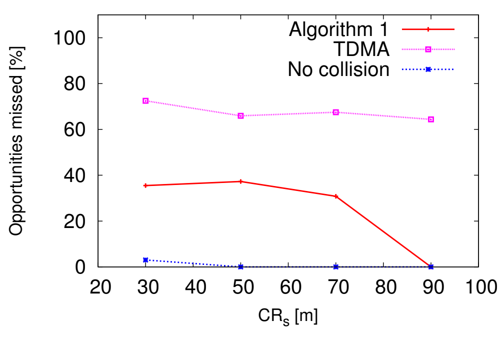
(c)



(d)



(e)



(f)

Figure 7.8: Percentage of MAC layer collisions, by varying CR_s and FN: (a) I_1; (b) I_2; (c) I_3. Percentage of communication opportunities missed, by varying CR_s and FN on I_3: (d) $FN = 30\%$; (e) $FN = 20\%$; and (f) $FN = 10\%$.

Metrics: (i) rate of collisions during group communication and the percentage of communication opportunities missed (Algorithm 1); (iii) average CG (to evaluate Algorithm 6, we define CG as the ratio of group merges among sensors, and the total number of group merges possible);

7.3.1 Group Communication

To evaluate Algorithm 1, we consider the rate of collisions and percentage of communication opportunities missed. Rate of collisions is important to evaluate, since collisions will cost energy and cause information to be potentially lost. To deduce if a communication opportunity was missed, we keep track of group merges, group splits, and encounters with beacons. This is important because missed communication opportunities may translate to false negatives. We consider the sensor deployment as derived for a non-diffusive event scenario. We note here that the initial sensor deployment only determines the initial network topology.

To evaluate the rate of collision, we vary the communication range CR_s and the number of nodes (obtained for FN = 10%, 20%, and 30% and SR=10m). The results are depicted in Figure 7.8(a,b,c). Interestingly, we observe that increasing the number of sensors and CR_s increases collisions, but it is always between 1.5% and 4.2%. Different zones of interests (as shown in Figure 7.7) have different collision rates, because of the number of sensors inserted and the graph structure of I.

The evaluation of percentage of communication opportunities missed is presented in Figure 7.8(d,e,f). We compare Algorithm 1 against a TDMA protocol, where sensors broadcast during predetermined time slots, and against a hypothetical scenario where sensors can decode the packets even if there was a collision. We notice that increasing CR_s reduces the percentage of communication opportunities missed, because the sensors have more time per contact. Another important observation is that the gap in the communication opportunities missed between Algorithm 1 and the hypothetical scenario arises solely because of collisions.

The above evaluation highlights the importance of a MAC and group management algorithm. It also emphasizes that having a sufficiently large CR_s is important to ensure reliability of data transfer among sensors.

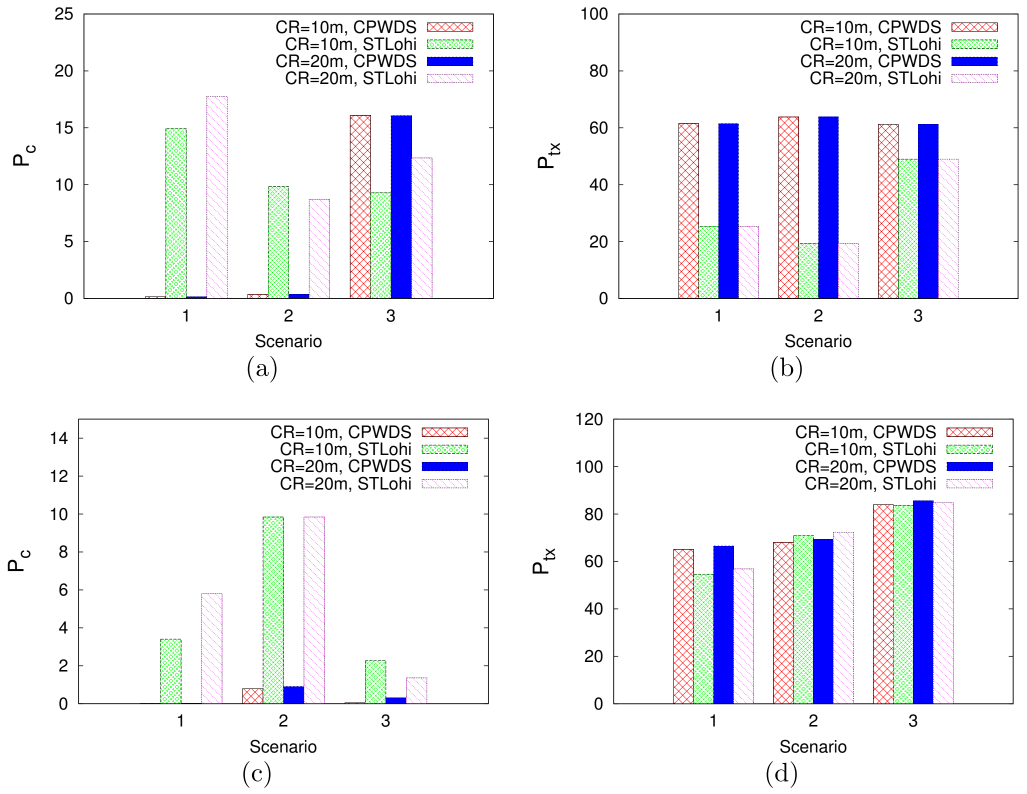


Figure 7.9: Impact of CR, Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

We evaluate the performance of continuous monitoring of WDS with our algorithms/protocols in CPWDSim on Micropolis [10] virtual city.

The flows in the main pipes of Micropolis change significantly every hour. Therefore, we use reports of the flows in the edges for each hour in the simulated 12 hours.

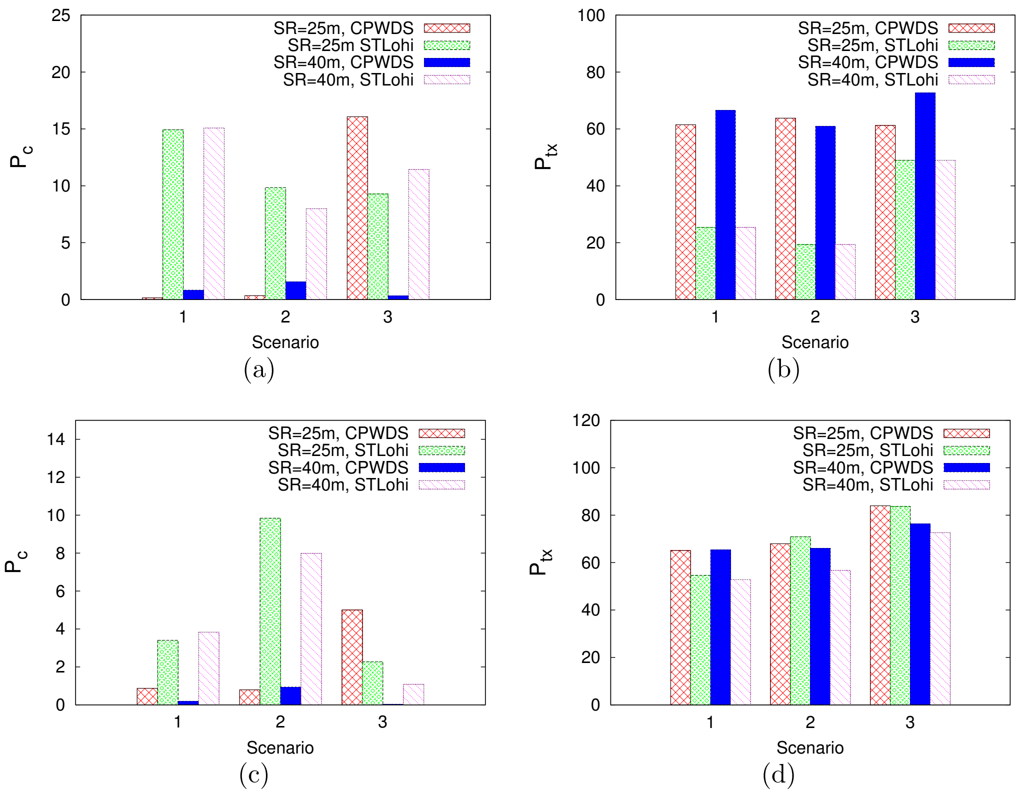


Figure 7.10: Impact of SR, Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

We simulated 96 different set of parameters, 10 times each, with different random seed values and a simulation duration of 6 hours, where the flows change every 1 hour. Our simulation results plot mean values. Due to the fact that, to the best of our knowledge no comparable system exists, we chose to evaluate the individual components (communication, computation and control) of our proposed CPWDS. For comparison with state of the art in communication we chose T-Lohi [84].

The *metrics* we use for evaluating our algorithms are: (i) percentage of collisions (P_c) - the percentage of transmissions that result in collisions; (ii) percentage of data transferred successfully (P_{tx}) - the percentage of nodes that have transferred their data to a beacon, or another sensor node; and (iii) valve settings and error for our controller.

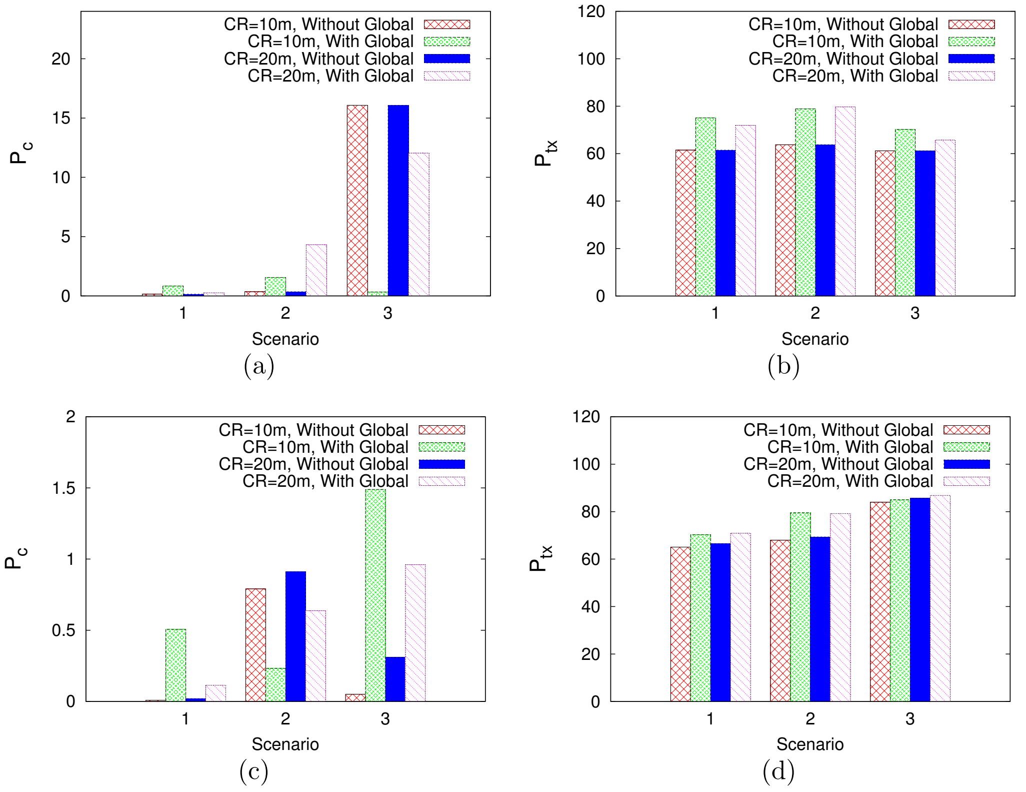


Figure 7.11: Impact of CPWDS Computation (i.e., Global View information), CR, and Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

The parameters we vary are: (i) *Scenario*, i.e., number of sensors deployed and their deployment locations; (ii) *Time Period*, i.e., time of the day, with varying WDS flows; (iii) SR - Sensing ranges for sensors (i.e., distance through the pipes up to which an event, such as a leak or chemical contamination, can be sensed); and (iv) CR - Communication range among sensors (the distance through the pipes up to which a receiver can decode a message transmitted by a sensor). We simulate 3 *Scenarios*. Scenario 1 has 67 sensors initially deployed in 11 locations, scenario 2 has 124 nodes initially deployed in 23 locations, and scenario 3 has 98 nodes initially

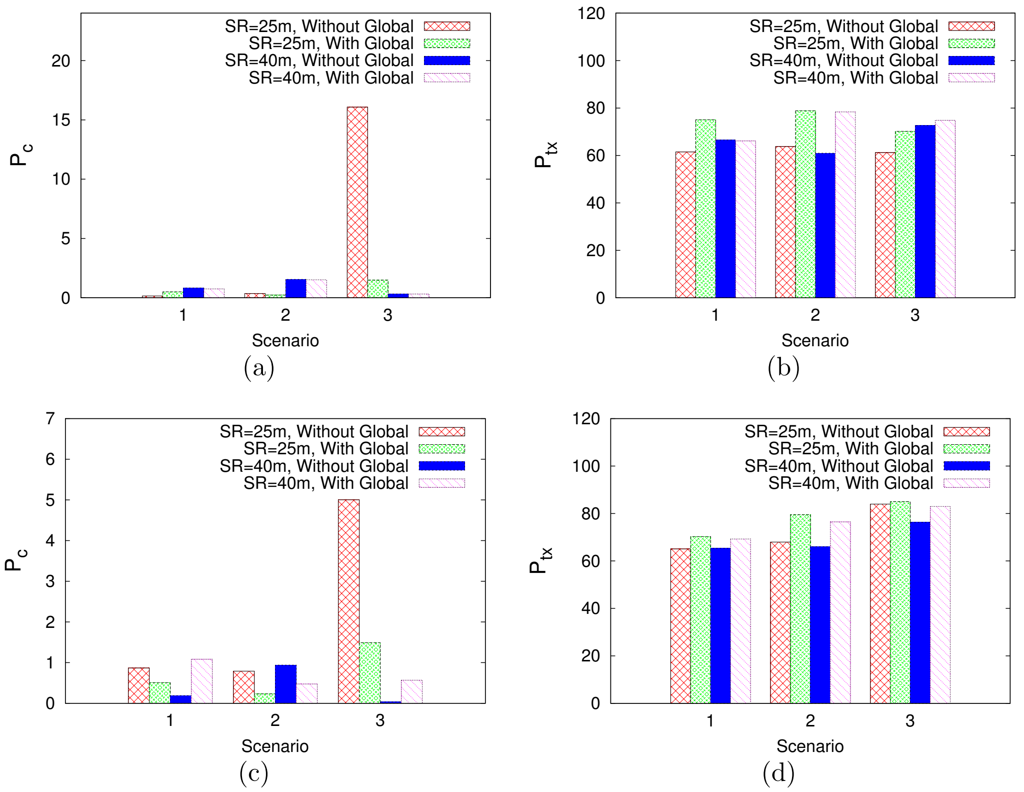


Figure 7.12: Impact of CPWDS Computation (i.e., Global View information), SR, and Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

deployed in 24 locations. For *Time Period*, we have Morning and Afternoon, 6 hours each. The flows in pipes corresponding to these 12 hours are obtained from Micropolis. The average length of an edge in Micropolis is 10m. We placed beacons at vertices, with an average separation of 30m. The time slot, Δ is chosen according to the communication range among sensors.

7.3.2 Impact of SR, CR, Scenario *and* Time Period *on CPWDS Communication*

We consider the percentage of collisions (P_c) and percentage of data successfully transferred (P_{tx}) for the three Scenarios, two Time Periods, two sensing ranges and two communication ranges. We compare our MAC/group communication mechanism with the state of the art MAC protocol for underwater short range communication, T-Lohi [84] in which nodes contend to report their data to each other every 10 time slots (e.g., 100ms when communication range is 10m). In a network of 1,000 nodes the T-Lohi protocol simulation takes several minutes to run and is very inefficient, since the nodes contend for the channel simultaneously. We therefore evaluate scenarios with at most 124 nodes.

The evaluation of our group communication protocol for communication ranges CR of 10m and 20m and sensing range SR=25m is shown in Figure 7.9. As shown, increasing the CR increases the node density, thereby increasing P_c. However, both T-Lohi and our protocol overcome this problem. Interestingly, P_{tx} is higher for our algorithm than T-Lohi in most cases. Whenever T-Lohi has higher P_{tx}, it also has a very high P_c. We observe that the Time Period, which directly affects the topology of the network, impacts P_c and P_{tx}. In Figure 7.10, the CR=10m. When SR is increased, the number of followers that have data to report increases. However, the results do not reflect a direct increase in P_c. It is interesting to note that P_{tx} is higher for our algorithm than for T-Lohi in most cases. In the case where SR = 25m in the Afternoon Time Period of Scenario 3, the sensors move such that the number of group splits and merges are high. That is the reason for the high P_c. However, P_{tx} is still high.

7.3.3 Impact of CPWDS Computation (Global View) on CPWDS Communication

For evaluating the computational aspect (i.e., the global view algorithm), we compare P_c and P_{tx} with and without using global information for the three Scenarios, by varying CR, SR, and the Time Period of day. Upon receiving global view information, the leader modifies its broadcast interval for the $HELLO$ message to the least allowable if it predicts group splits or beacon encounters. Otherwise it chooses a random broadcast interval to prepare for group merges.

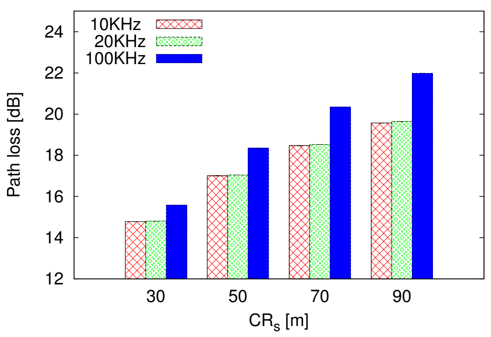
Algorithm 14 consistently helps increase P_{tx}, although it does not always decrease P_c as shown in Figures 7.11 and 7.12. In Figure 7.11, $SR = 10$m. Figure 7.12 has $CR = 10$m. Interestingly, P_c is always lower than 15% when global view information is used. These results clearly demonstrate that the frequency with which the leader sends the $HELLO$ message affects P_c and P_{tx} and show that using global information from beacons increases the efficiency of communication. As we can see in Figure 7.12, Scenario 3, $SR = 25$m has a very high P_c without global view information. This is because there are many group splits and merges in this scenario. The global view data in this case significantly reduces the P_c while maintaining a high P_{tx}.

7.4 Performance Evaluation of Computation

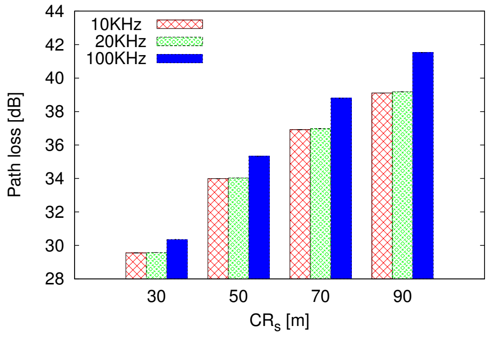
The metrics we used for evaluating our proposed algorithms were: (i) the number of sensors and sensing coverage (Algorithm 5); (ii) the number of beacons and the achieved localization error (Algorithms 9 and 12). The *parameters* we were varied were: (i) CR_s; (ii) CR_b; (iii) SR and the sensing model; (iv) FN; (v) LE; and (vi) the zone of interest I. The sizes and badness of the zones of interest are also depicted in the figure. The length of the diagonal of I_1 is 701m, I_2 is 1,349m, and I_3 is 1,100m.

To evaluate the impact of sensing range for non-diffusive events on our algorithms, we use a parameter SR, which is the distance r up to which $S(r) > \Theta_S$ in the absence of noise (i.e., $r = (\frac{S(0)}{\Theta_S})^{\frac{1}{\alpha}}\ r_{ref}$). In our simulations, we set $\Theta_S = 10^{-4}$, $S(0) = 1$, $\alpha = 1$, $\beta = 0$ and r_{ref} to 0m, 0.001m, and 0.0025m, giving us SR of 0m, 10m, and 25m respectively. Evaluation of the group communication algorithms are done on one of these scenarios.

To evaluate the diffusive sensing model, we assume that the sensors are inserted after a certain time period after which the event occurred, τ. In this dissertation, we use data from EPANET to provide the range of the event in terms of the con-



(a)



(b)

Figure 7.13: H_A at varying distances for different frequencies of communication in (a) Straight pipes; (b) Pipes with bends

centration of the contaminant in the pipes of a WDS. When τ is provided as an input to EPANET, it provides the concentration of the contaminant in each edge after τ hours. We vary τ as 2, 4, and 6 hours, as well as the sensing threshold, Θ_S of the sensors as 1mg/l, 5mg/l, 10mg/l, and 20mg/l. Both these parameters vary the sensing range of the sensors.

To choose the CR_s for our simulations, we derive the results of using the path loss model, H_A. [29] derives the absorption factor for sound waves in sea water, where the expression for absorption due to pure water is defined as: $A_3 P_3 f^2$, where $A_3 = 3.964 \times 10^{-4} - 1.146 \times 10^{-5} T + 1.45 \times 10^{-7} T^2 - 6.5 \times 10^{-10} T^{-10} dBkm^{-1} KHz^{-1}$; $P_3 = 1 - 3.83 \times 10^{-5} D + 4.9 \times 10^{-10} D^2$, D being the depth of water; f is the frequency of the sound wave. From a testbed that has been developed specifically for underwater acoustic communication experiments [3], we consider the frequency of communication of the order of 20KHz. Using the communication model from Section 4, for 10 KHz, 20 KHz, and 100KHz frequency of communication, the path loss H_A for distances 30m and 60m, for pipe with and without bends are as shown in Figure 7.13. As the figure suggests, at 100 KHz, the path loss is the highest. Based on a desired data rate and the range of communication, researchers may use this analysis to determine the appropriate frequency of communication. This also provides us suitable values of CR_s to use in our evaluation. In our experiments, we use CR_s between 10m and 90m.

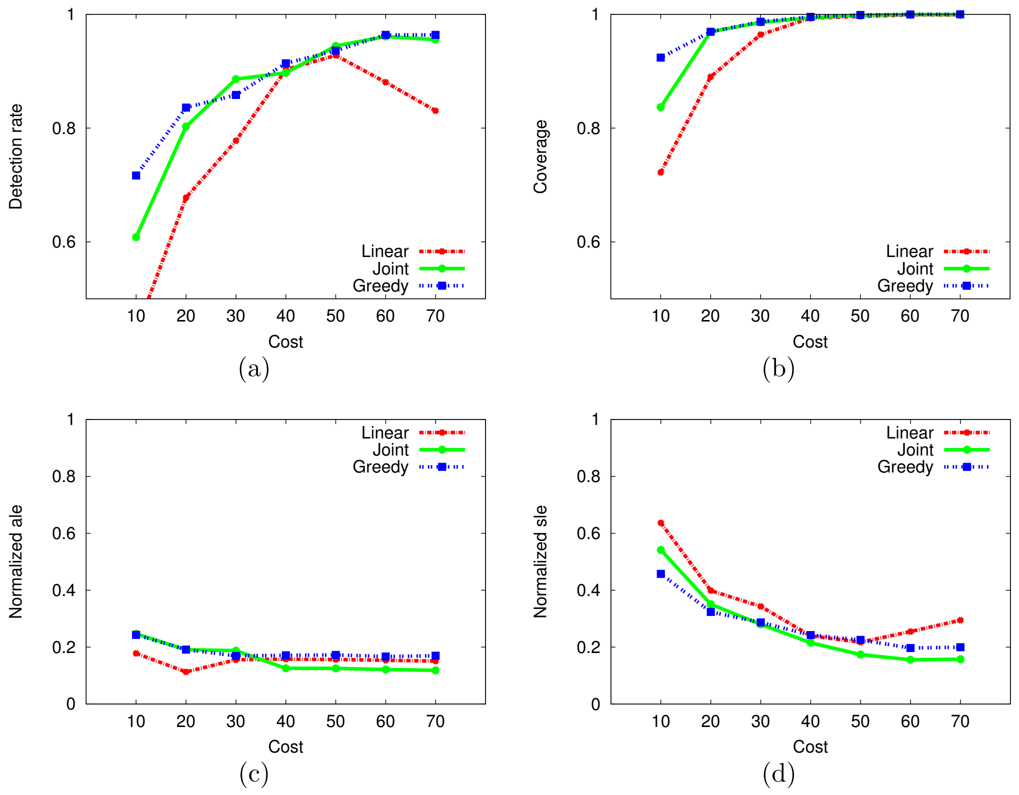


Figure 7.14: Model results for the Net1 example shown in Figure 3.7, with a uniform distribution of the leaks for (a) Detection rate, (b) Coverage, and (c) Average Localization Error, (d) System Localization Error

We divide the evaluation into three parts: (i) evaluation on the Net1 example; (ii) evaluation on the Micropolis example; (iii) evaluation of sensing coverage solutions for cases where localization is not required. We perform evaluations on the Net1 example to verify the accuracy and robustness of our models and algorithms. Micropolis

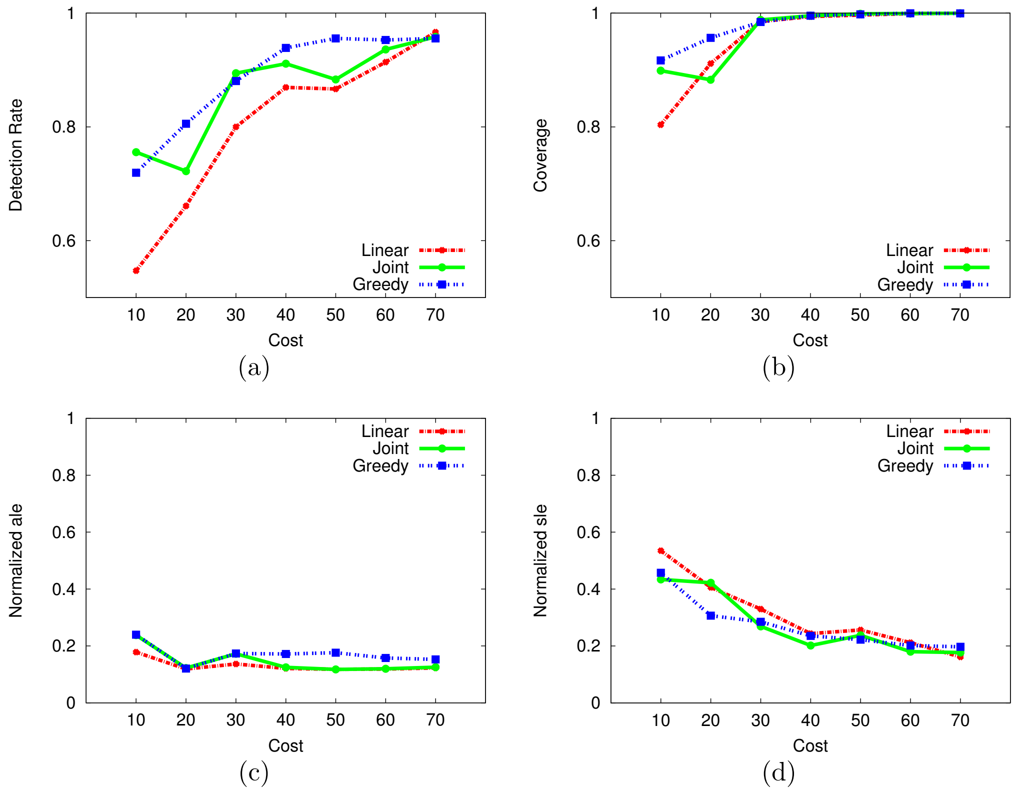


Figure 7.15: Model results for the Net1 example shown in Figure 3.7, with a random distribution of the leaks for (a) Detection rate, (b) Coverage, and (c) Average Localization Error, (d) System Localization Error

evaluations are done to ensure the scalability of our algorithms to bigger networks.

We first present our evaluations on a simple EPANET example, Net1, depicted in Figures 7.14 - 7.18. All the results are averaged over 30 runs each, with leaks being present on each of the pipes of the WDS. We run sensitivity analysis by varying the leak distribution, sensor-beacon cost ratio, and number of simulation runs. We also check robustness of the formulations by adding restrictions on sensor and beacon placement locations.

In Figure 7.14, the leak distribution is uniform (i.e., the leak probability is the same in all edges), whereas in Figure 7.15, the leak distribution is random (i.e.,

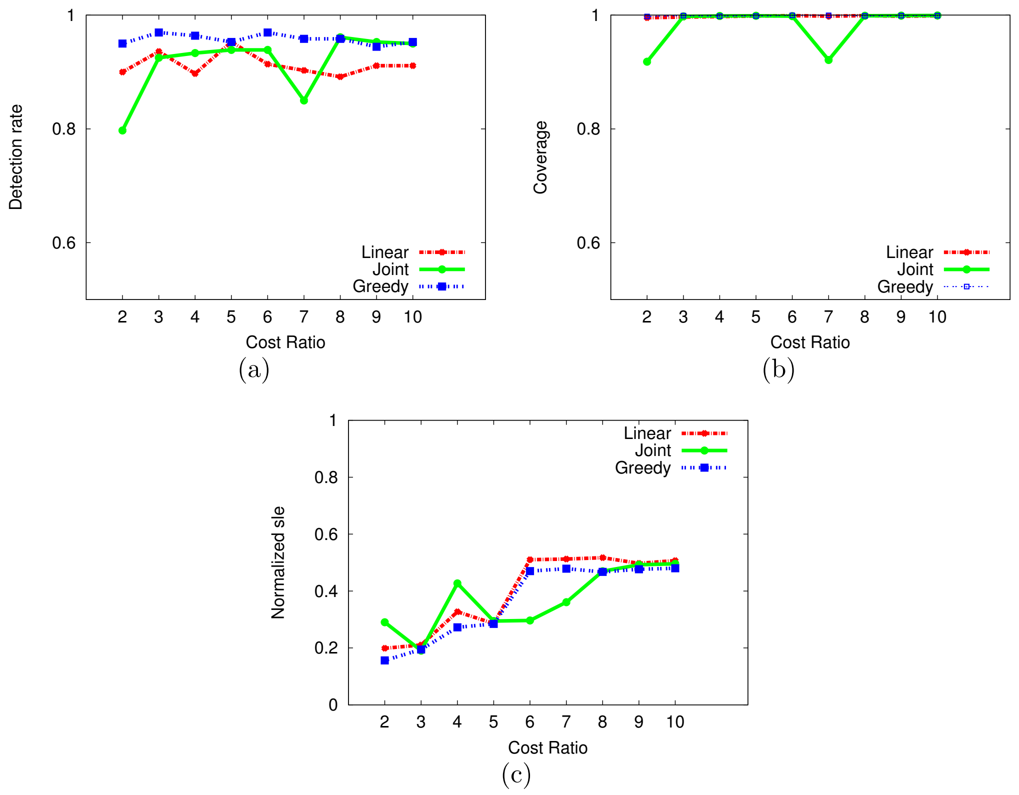
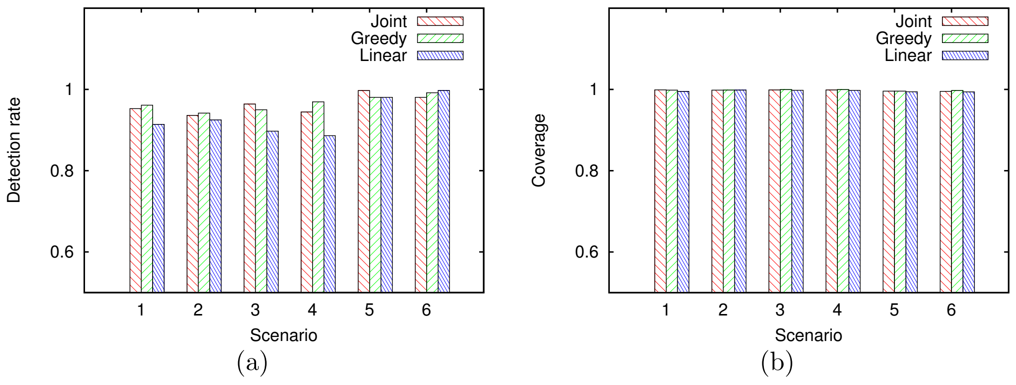


Figure 7.16: Model results for the Net1 example shown in Figure 3.7, while varying cost ratio between sensors and beacons for (a) Detection rate, (b) Coverage, and (c) System Localization Error

sampled from a uniform random distribution where each pipe has different leak probabilities). In both cases, the cost ratio of a beacon to a sensor is 3. As the figures suggest, the general tendency is for the coverage and detection rate to increase, whereas the *sle* and *ale* reduce. This is expected, since when more budget is available, it is possible to afford more devices to improve *sle* and *ale*, and that in turn improves the coverage and detection rate.

In Figure 7.14, we notice that the normalized *sle* for the linear case does not decline smoothly. This is because the objective function in the linear problem formulation is to maximize the minimum sensing coverage with optimizing sensor placement



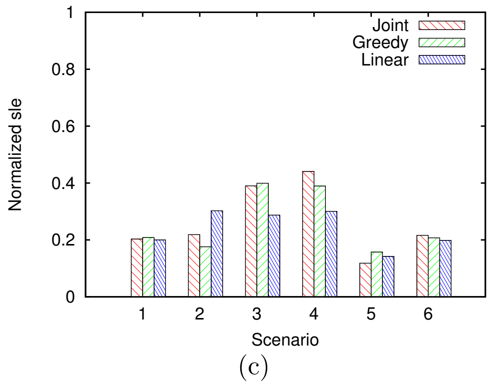


Figure 7.17: Model results for the Net1 example shown in Figure 3.7, when location of beacons and sensors may be restricted for (a) Detection rate, (b) Coverage, and (c) System Localization Error

first, and to minimize the maximum *ale* with optimizing beacon placement next. This places a lot of emphasis on sensor placement thus deviating from the real optimum for the original joint problem. This indicates that although the linear formulation is simple and easy to solve, it does not provide the best result in practice.

Figure 7.16 illustrates the sensitivity of performance metrics to the beacon-sensor cost ratio. The cost budget is set to be 50. Normalized *sle* increases overall with increasing the cost ratio. This is because fewer beacons are placed. Fewer beacons lead to an increased *ale*, because there are more pipes between each of the beacon pairs. In case of a cost ratio of 2, the *joint* formulation places beacons at all vertices,

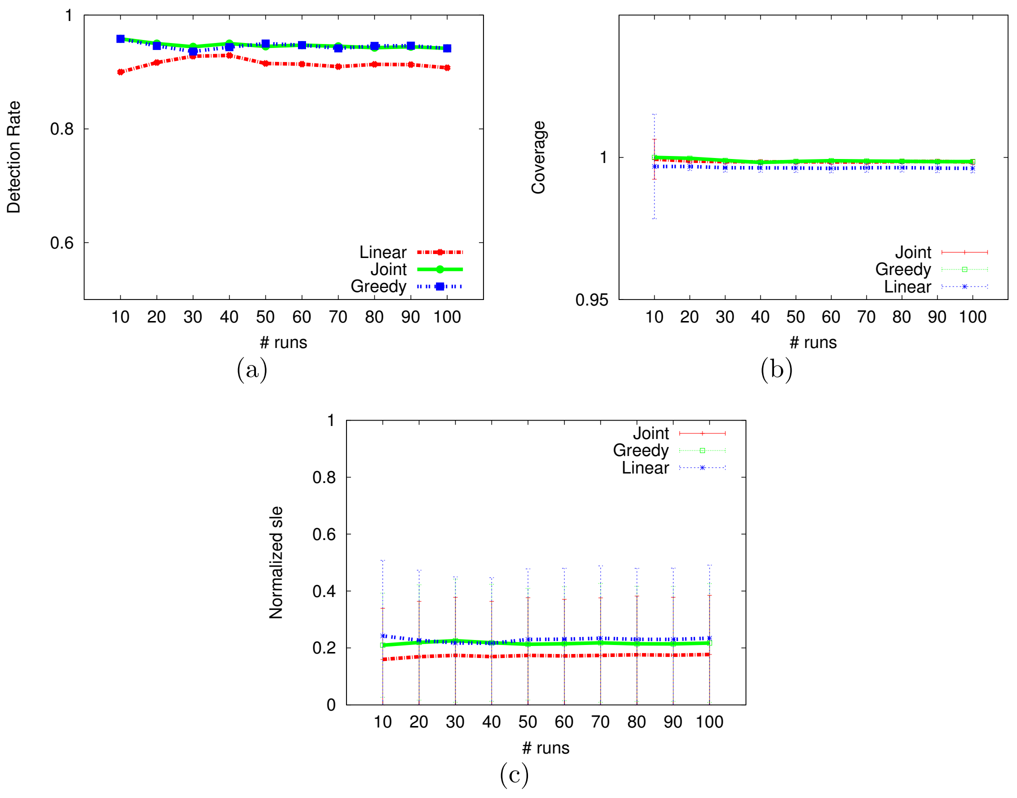


Figure 7.18: Model results for the Net1 example shown in Figure 3.7, for different number of simulation runs for (a) Detection rate, (b) Coverage, and (c) System Localization Error

thereby spending less on sensors, which is determined to be the optimal for that cost ratio. This leads to a higher coverage, and a lower *sle* for the cost ratio of 3. However, the *greedy* formulation is able to maintain an increasing trend in *sle* because by nature, it optimizes the average coverage and localization errors. Above a cost ratio of 5, the trend is more uniform over all formulations.

We test robustness of our solution by adding constraints to the placement of beacons and sensors, as shown in Figure 7.17. The nodes that are restricted are as described in Table 7.1. The cost budget is set to be 50.

Restricting the placement of beacons and sensors harms the coverage and de-

Table 7.1: Scenarios for which sensor and beacon placements are restricted

Scenario	**Beacon restriction**	**Sensor restriction**
1	12, 21	12, 22
2	11, 12, 21	11, 12, 22
3	11, 12, 21, 31	11, 12, 21, 31
4	9, 11, 12, 21, 31	
5		9, 11, 12, 21, 31
6	9, 11, 12, 21, 31	9, 11, 12, 21, 31

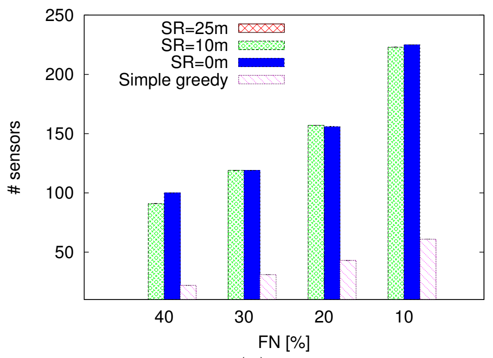
tection rate as expected. But mainly, with a higher cost, the beacon placement restriction affects sle more than the sensor placement restriction. It is observed that when both sensor and beacon placement are restricted severely (Scenario 6), the beacons are deployed at all possible vertices, even though it reduces the number of sensors slightly as compared to a case when there are fewer restrictions (Scenario 3).

The obtained coverage, detection rate, and sle values for varying number of runs, are shown in Figure 7.18. The cost budget is set to 50. However, the error bars for coverage gradually reduce. In case of the normalized sle, the error bars are wide because the edge localization errors are different for each edge. Moreover, if the leak is not detected, the normalized sle is 1, thereby increasing the variance in the results.

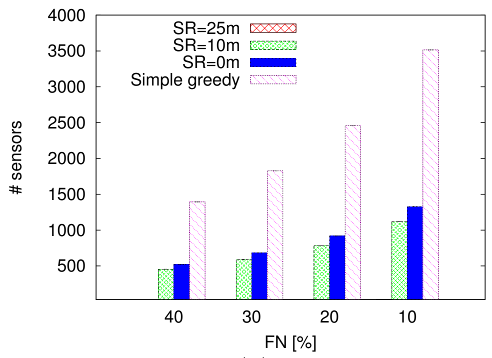
The median length of edges in all three zones of interest is ~10m. We add noise (mean $\mu = 0$m and a variance $\sigma^2 = 1$m) to sensing and communication ranges, SR, CR_s, and CR_b.

7.4.1 Sensor Placement

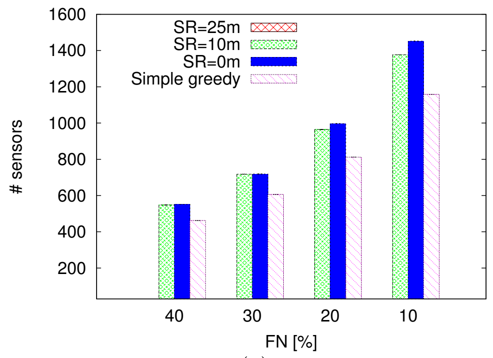
First, we evaluate the Sensor Deployment Algorithm (Algorithm 5) by varying parameters that influence sensor deployment, namely the false negative rate FN, SR and the Zone of Interest. We then evaluate the Time of Deployment Algorithm (Algorithm 6) by comparing against other algorithms. The evaluations are performed



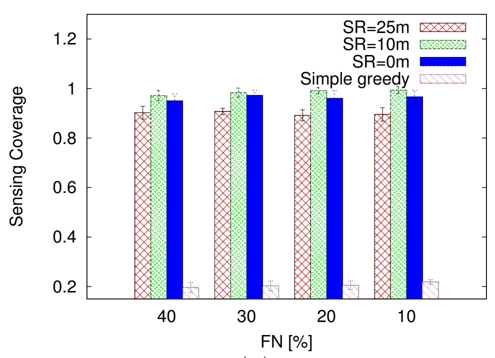
(a)



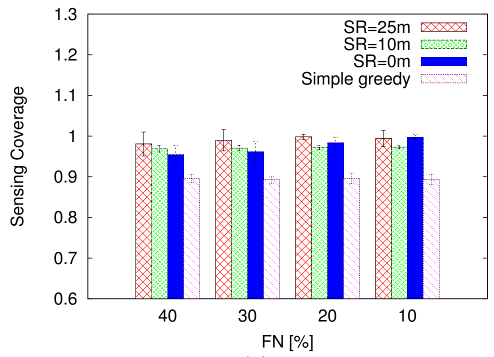
(b)



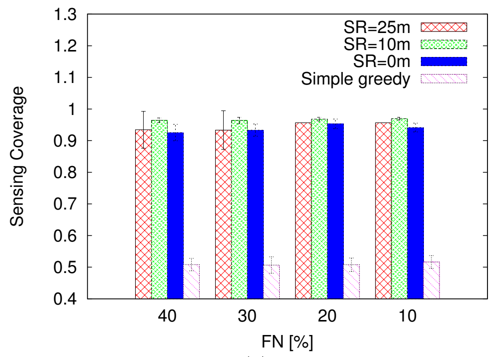
(c)



(d)



(e)

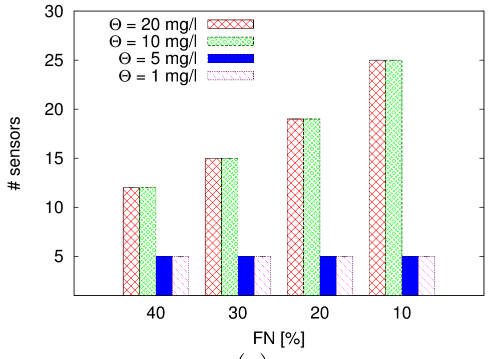


(f)

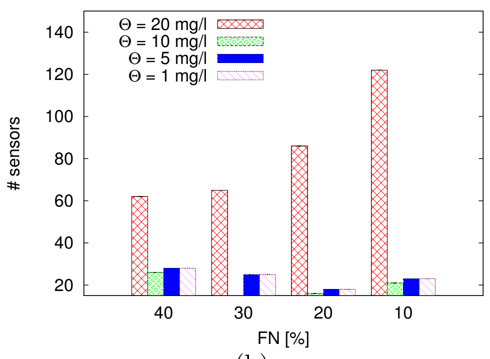
Figure 7.19: # sensors required to achieve FN for non-diffusive events, given SR and: (a) I_1; (b) I_2; (c) I_3. Achieved Sensing Coverage for different FN, SR in: (d) I_1; (e) I_2; (f) I_3.

for both diffusive and non-diffusive events. In case of diffusive events, τ is varied too.

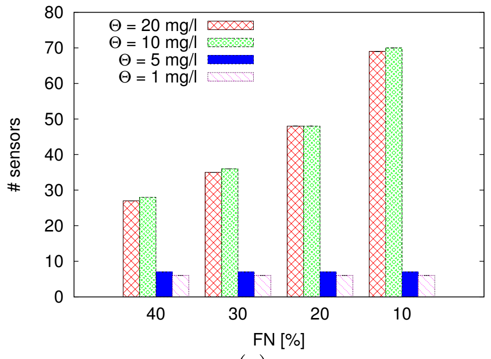
For a given Zone of Interest, FN and SR, the number of sensors inserted is deterministic, and is depicted in Figures 7.19(a,b,c) (i.e., without error bars). Results



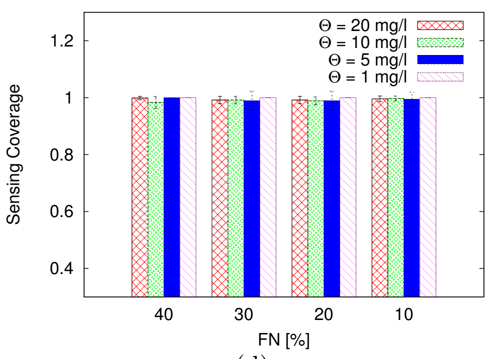
(a)



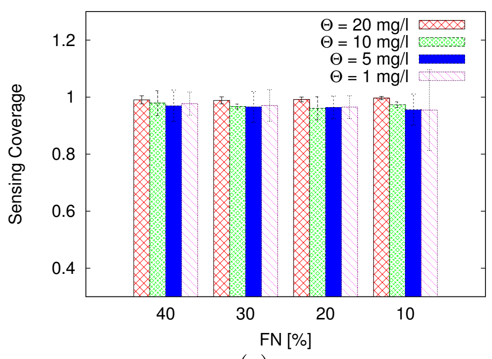
(b)



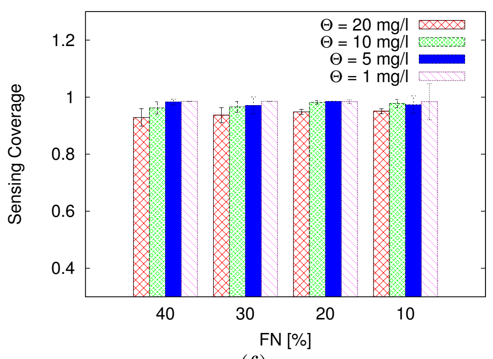
(c)



(d)



(e)



(f)

Figure 7.20: # sensors required to achieve FN, given SR and: (a) I_1; (b) I_2; (c) I_3. Achieved Sensing Coverage for different FN, SR in: (d) I_1; (e) I_2; (f) I_3 when for τ = 2 hours.

evaluating sensing coverage are shown in Figures 7.19(d,e,f). From Figures 7.19(a), (b), and (c), as expected, we observe that the required number of sensors increases as we decrease FN. We also notice that when SR is high (i.e., the event can be

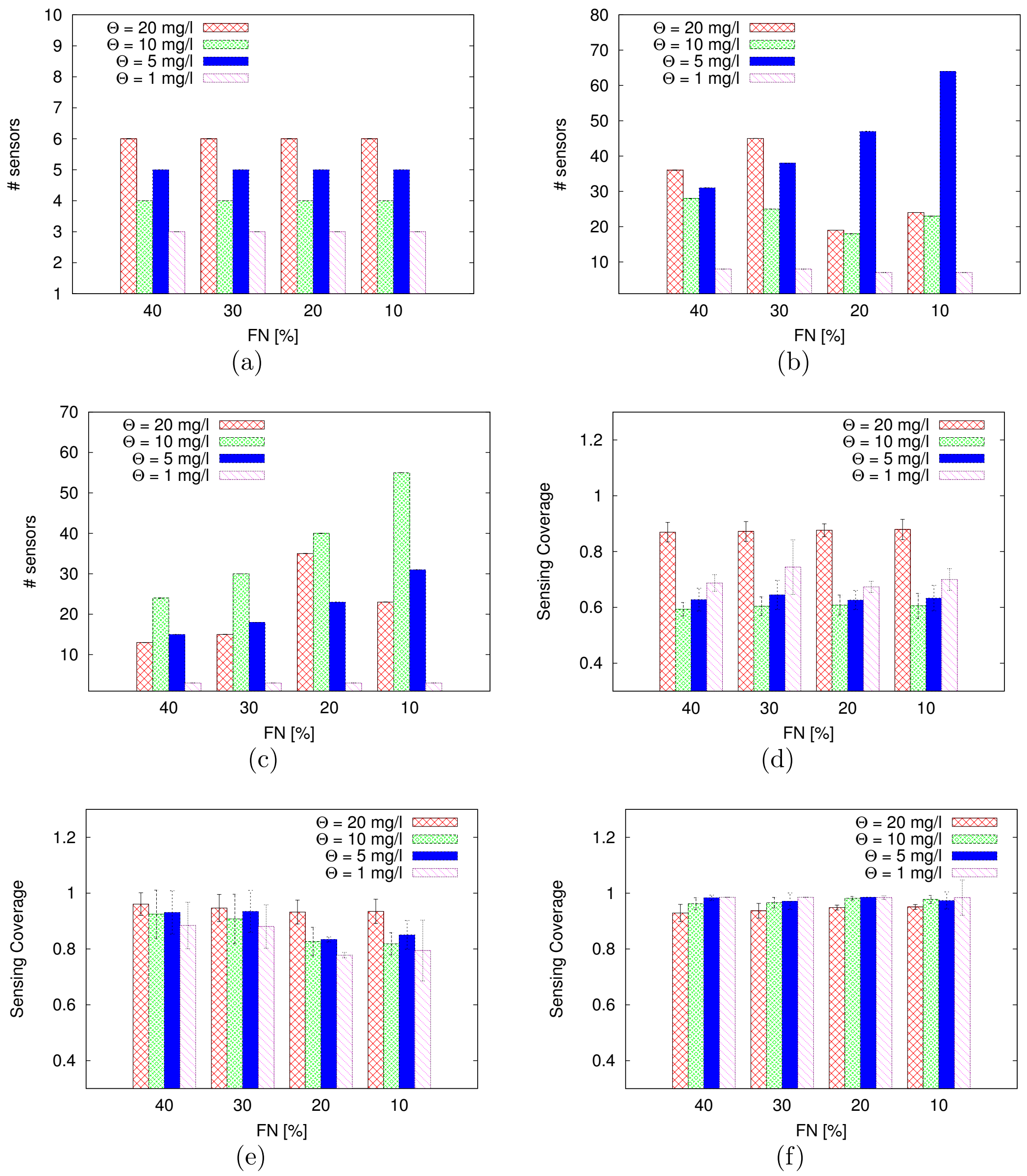


Figure 7.21: # sensors required to achieve FN, given SR and: (a) I_1; (b) I_2; (c) I_3. Achieved Sensing Coverage for different FN, SR in: (d) I_1; (e) I_2; (f) I_3 when for τ = 4 hours.

detected from a longer distance from the source of the contaminant) the number of sensors required to ensure a given FN dramatically reduces. In I_3, increasing SR from 10m to 25m for FN = 10% decreases the number of sensors by 91 times and

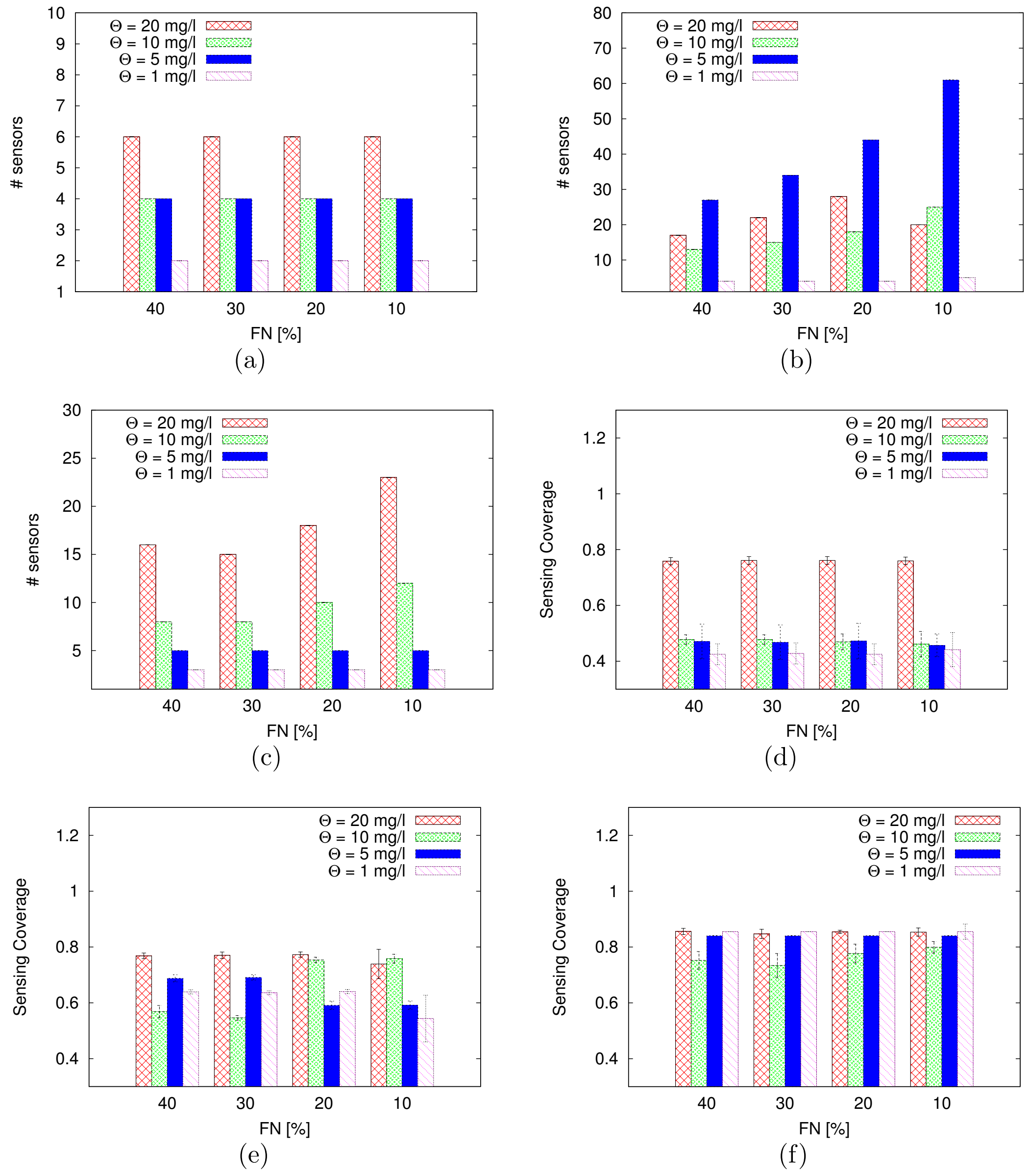


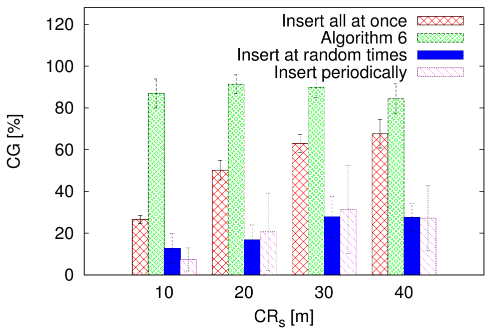
Figure 7.22: # sensors required to achieve FN, given SR and: (a) I_1; (b) I_2; (c) I_3. Achieved Sensing Coverage for different FN, SR in: (d) I_1; (e) I_2; (f) I_3 when for τ = 6 hours.

increases the sensing coverage by 1%. *Although a decrease is expected, it is very interesting to note the factor of decrease. This means that investing a little more on sensors to get a higher sensing range can translate to a dramatic cost reduction.*

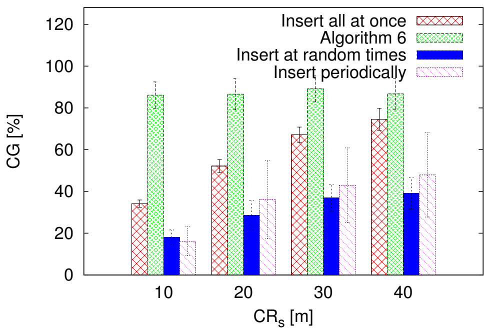
Figures 7.19(d), (e), and (f) show that Sensing Coverage improves for lower values of FN. This is because we ensure that sensors inserted will cover every edge with at least 1-FN probability. An interesting observation is that for the same FN and SR, the number of sensors inserted in I_2 is smaller, and the sensing coverage achieved is higher than the number of sensors inserted and sensing coverage achieved, respectively, in I_3, although the number of edges in I_2 is higher. This is because I_2 has a lower badness than I_3. Since I_1 is significantly smaller, the number of sensors inserted is smaller than I_2 and I_3.

In the older version of the CPWDSim simulator, we used a simpler greedy algorithm, where the weights for the heuristic are represented for every vertex, rather than vertex pairs, and SR is 0m (binary sensing). We call this the "simple greedy" algorithm. Simple greedy has an additional tuning parameter α. We compare the number of sensors deployed and sensing coverage achieved in Simple greedy for $\alpha = 0$ and degree of coverage $1 - FN$ to those obtained for $SR = 0m$ in CPWDSim. The sensing coverage of Simple greedy on I_3, 45.6% lesser than CPWDSim using 17.7% fewer nodes on an average. On the contrary, in I_2, Simple greedy has a sensing coverage of 8.2% lesser than CPWDSim using 166% more nodes on an average. This erratic difference is due to the difference in badness of I_2 and I_3, making Simple greedy unsuitable for large networks with high badness.

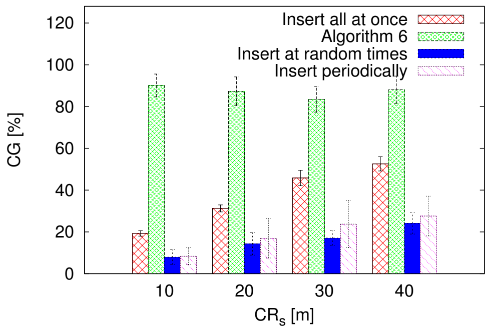
The number of sensors inserted, and the sensing coverage for for $\tau = 2$, 4, and 6 hours for diffusive events are presented in Figures 7.20, 7.21, and 7.22 respectively. For smaller τ, a larger number of sensors are inserted, since the event has not yet propagated through the system. Very interestingly, the trend of reduction in number of sensors with the decrease in FN is not apparent because, unlike the non-diffusive sensing model, the event may not be propagated uniformly in the system. Another very interesting result is that the trend of increasing Sensing Coverage is not appar-



(a)



(b)

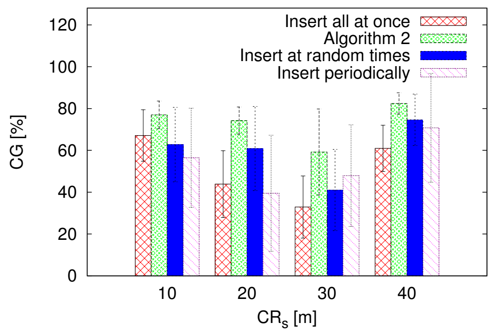


(c)

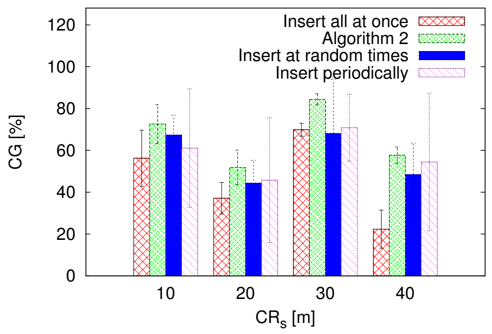
Figure 7.23: Evaluation of number of possible contacts explored by the sensors depending on their times of insertion, with the non-diffusive sensing model, for various CR_s and the algorithm used for: (a) I_1; (b) I_2; (c) I_3

ent, and the results have a high variance because the number of sensors inserted is fewer, and a single sensor failing to cover any edge leads to a larger number of edges to remain uncovered. This is also the reason why the trends of increasing sensing coverage and reduced number of nodes with decrease in FN is more noticeable when the sensing threshold, Θ_S is high.

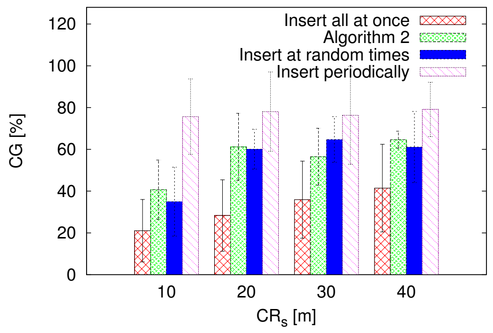
We now compare the Time of Deployment algorithm (Algorithm 6) to three other algorithms: (i) all sensors are inserted at the same time; (ii) the time of deployment of a sensor is randomly chosen; (iii) sensors are grouped and inserted periodically; the period is determined by the number of sensors inserted and the duration of the



(a)



(b)



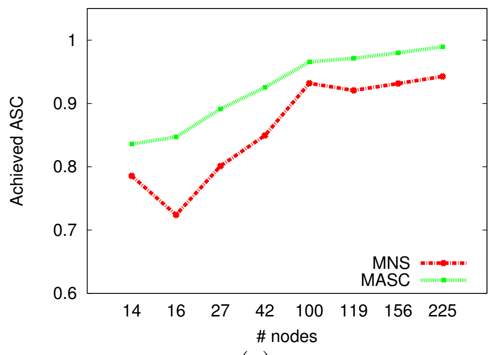
(c)

Figure 7.24: Evaluation of number of possible contacts explored by the sensors depending on their times of insertion, with the diffusive sensing model, for various CR_s and the algorithm for: (a) I_1; (b) I_2; (c) I_3

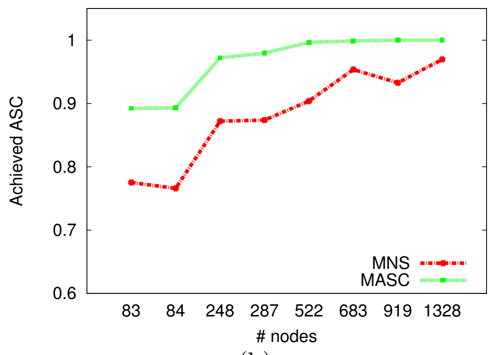
simulation. We investigate how CR_s affects our algorithm. The results are depicted in Figure 7.23 for non-diffusive events, and Figure 7.24 for diffusive. Here, FN and SR are set to 10% and 10m respectively.

From Figure 7.23, it is clear that Algorithm 6 outperforms the other algorithms on I_1, I_2, and I_3. We observe that increasing CR_s increases CG, since sensors do not need to be in close proximity to communicate, given a large communication range. The importance of time of deployment and communication among sensors is clear in this evaluation.

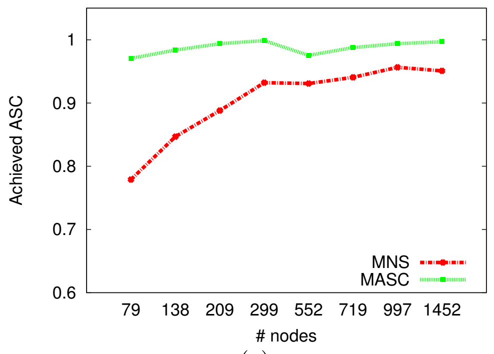
Figure 7.24 shows that except for I_3, Algorithm 6 performs better than the other algorithms. This is because the number of sensors is smaller, and concentrated in a smaller area than the non-diffusive case. Also, we observed that the number of possible contacts were low. Therefore, a single missed communication leads to a large reduction in CG. As observed in the sensor deployment results, the variance on the results are high.



(a)



(b)



(c)

Figure 7.25: Comparison between MNS and MASC for achieved average sensing coverage in MATLAB for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

Now, we evaluate sensor deployment for the $MLBSC$ and $MASC$ problems.

The placement of sensors is determined using the MNS problem, CPLEX in MATLAB for the $MLBSC$ problem, and a greedy heuristic implementation in MATLAB for the $MASC$ problem. The analytical results were obtained by calculating

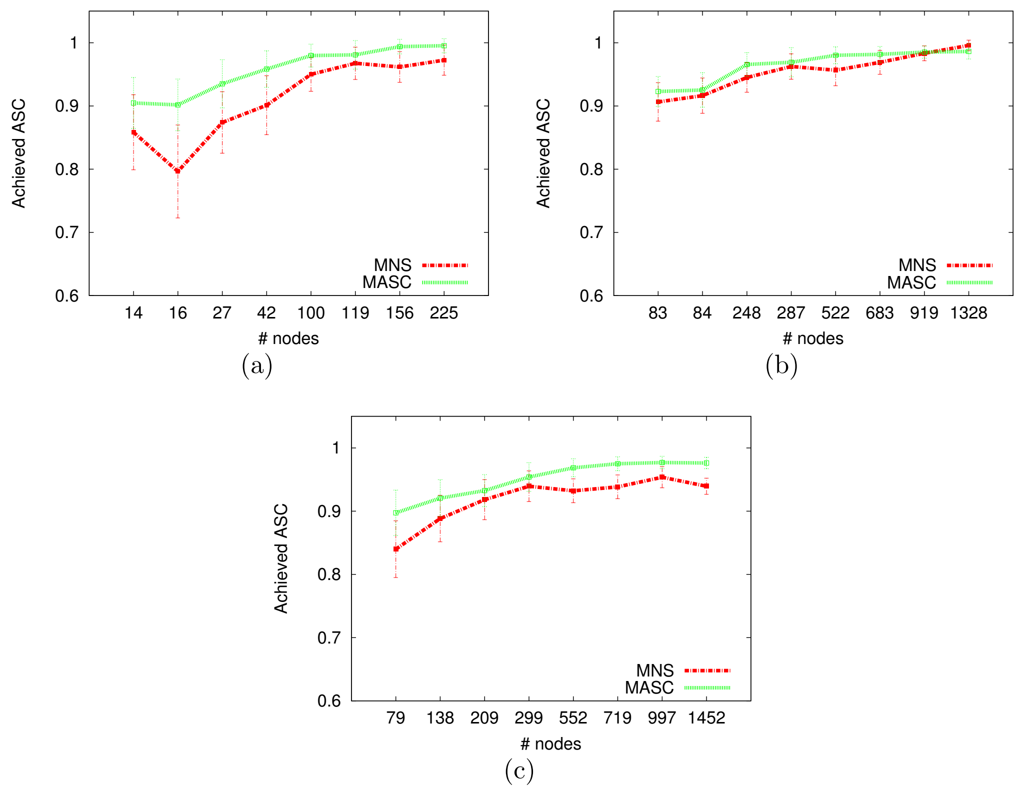


Figure 7.26: Comparison between MNS and MASC for achieved average sensing coverage over 100 runs in CPWDSim for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

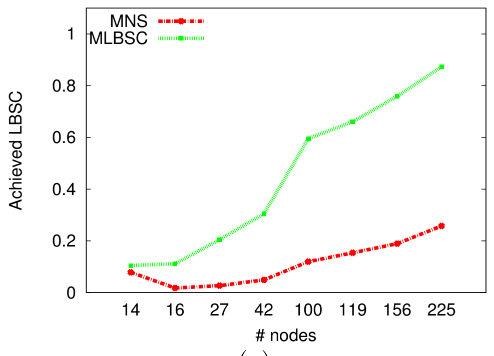
$LBSC$ and ASC for a given sensor configuration using equations 5.1 and 5.3 respectively. The simulations were run on CPWDSim.

Mobile sensors have a clear advantage over static sensors in terms of locating the leaks/backflows. However, to achieve this end, we need to ensure that the mobile sensors traverse through the pipe with the leak/backflows. This work addresses the issue of covering all the edges in a given zone of interest with mobile sensors, since it is an essential prerequisite for locating leaks/backflows. Comparing the solution against static sensor networks will therefore be unfair. On the other hand, other mobile sensor solutions assume that the path followed by the sensors is deterministic. Therefore comparing against these solutions is also unfair.

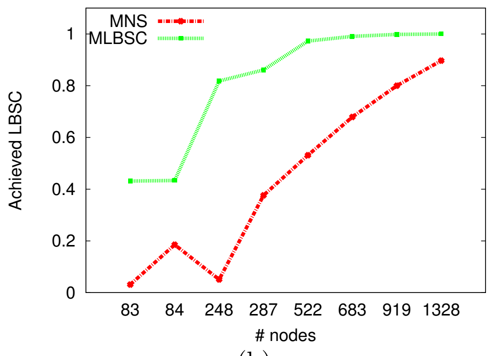
The parameters that we varied are: (i) Number of sensors; (ii) zone of interest - I_1, I_2, I_3. For comparison, the metrics we use are: (i) average sensing coverage (ASC); (ii) lower bound sensing coverage ($LBSC$). We set the total number of sensors to 8 different values based on 8 degrees of coverage inputs to MNS (0.2, 0.3, ... 0.9) for 3 different zones of interest. The three zones of interest are as shown in Figure 7.2. All three zones of interest have different number of vertices, number of edges, and flow distributions at junctions. The ASC and $LBSC$ calculated analytically using MATLAB are plotted in Figure 7.26 and Figure 7.27. The ASC and $LBSC$ achieved in simulation using MNS is presented in Figure 7.25 and Figure 7.28. Here, each scenario is simulated 100 times, and the figures plot the mean and one standard deviation (if applicable).

As the number of sensors is increased, the achieved ASC uniformly increases for both $MASC$ and MNS problems in both simulation and analytically, as shown in Figure 7.25 and Figure 7.26. It is interesting to observe in Figure 7.25 that the achieved ASC with the $MASC$ is usually higher than that of MNS, except in one case (Zone I_2, number of sensors 1328). This is because the algorithm in $MASC$ is a greedy heuristic and does not always achieve the optimal solution. However, in most cases, $MASC$ performs better than MNS.

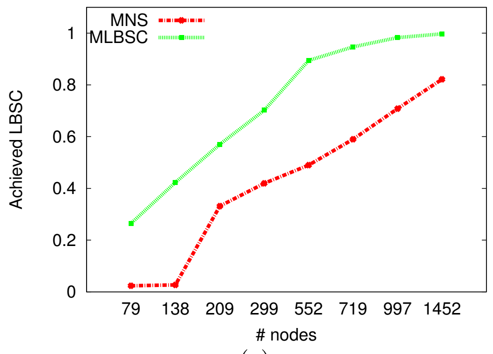
It is interesting to observe Figure 7.28 that as the number of sensors is increased, the achieved $LBSC$ does not show any discernable trend in simulations. However, the $LBSC$ shows an increasing trend in theory, as shown in Figure 7.27. This is because of the random movement of sensors at junctions. The randomness has a lower effect on an average in the case of ASC, but when movement of sensors through individual edges is considered in calculating the $LBSC$, the results become less predictable. In the analytical results in Figure 7.27, it is clear that $LBSC$ from MNS is lower than that of $MLBSC$, since the $MLBSC$ problem is designed to maximize $LBSC$, and



(a)



(b)



(c)

Figure 7.27: Comparison between MNS and MLBSC for achieved lower bound sensing coverage in MATLAB (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

the CPLEX solver is used to obtain the optimal solution.

7.4.2 Beacon Placement and Event Localization

We evaluate the beacon placement algorithm by calculating the number of beacons placed when LE, CR_b and SR are varied. The results are depicted in Figures 7.29, and 7.30. For diffusive events, the τ is 2 hours. Figure 7.29 shows that SR and Θ does not have a large impact on the number of beacons deployed. However, the input LE greatly affects the number of beacons. To calculate the achieved LE, we ignore false negative cases, i.e., if the event was not detected by the sensors, we discard that result. Therefore, evaluating event localization against parameters that affect only the false negatives (I and FN) is unnecessary.

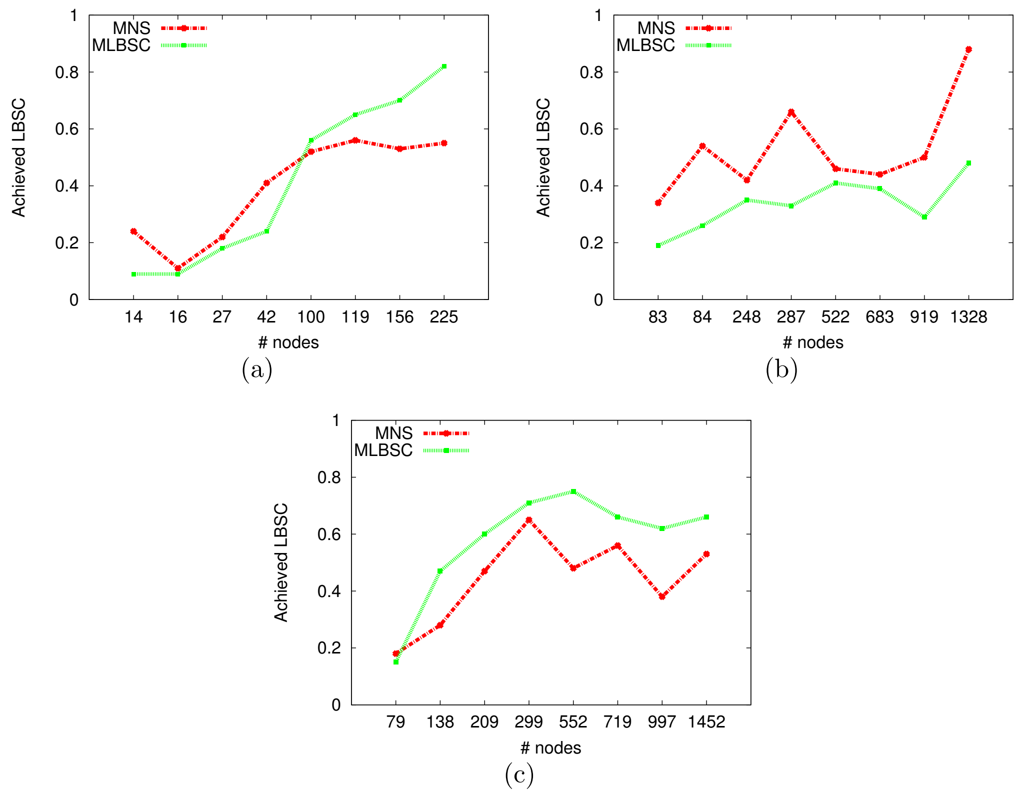
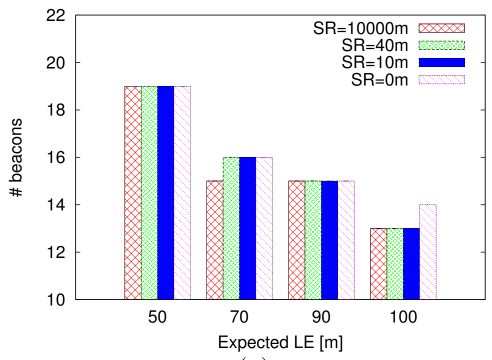
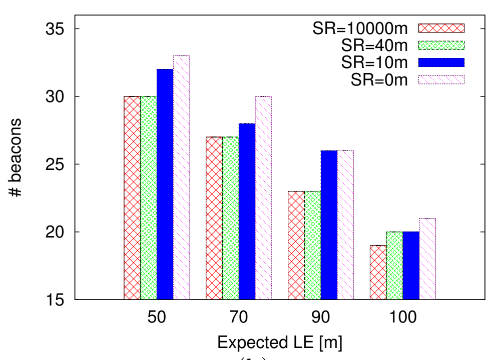


Figure 7.28: Comparison between MNS and MLBSC for achieved lower bound sensing coverage over 100 runs in CPWDSim for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

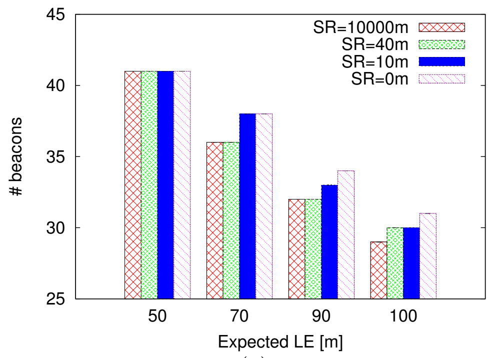
The beacon placement algorithm was run for varying LE and CR_b on I_3 for non-diffusive events. It is important to study the effects of large CR_b (i.e., larger than the length of a pipe). We evaluate our beacon placement algorithm for higher values of CR_b, since CR_b may be higher with a higher transmit power. The results are plotted in Figure 7.30 (a) and (b). In each run, an event was placed in a different edge of I_3. The plotted result is the average localization error achieved, which is lower than expected LE in all cases. Figure 7.30(c) shows that the achieved localization error depends on where the events are present. As expected, higher expected LE yields higher achieved localization error. But in some edges, the localization error can be much smaller.



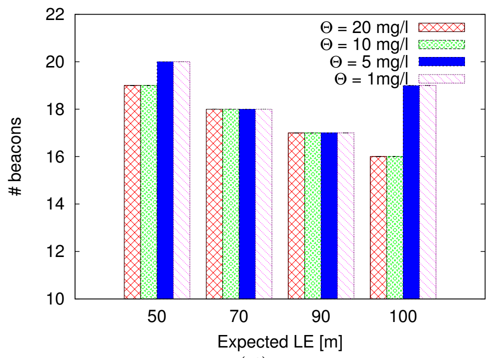
(a)



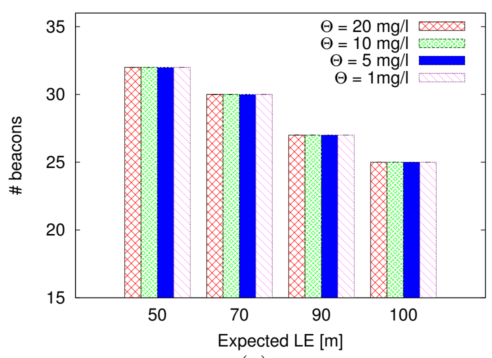
(b)



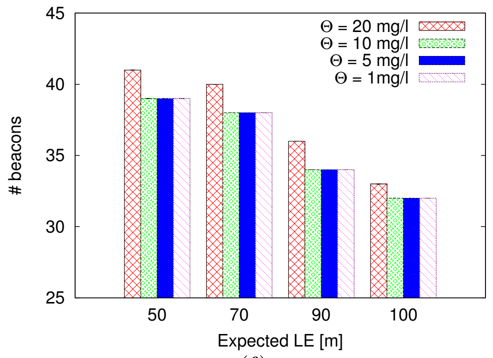
(c)



(d)



(e)



(f)

Figure 7.29: Number of beacons deployed for non-diffusive events, varying LE and SR on (a) I_1; (b) I_2; and (c) I_3. Number of beacons deployed for the diffusive sensing model, varying LE and Θ, for $\tau = 2$ hours on (d) I_1; (e) I_2; and (f) I_3.

In this set of simulations we investigate how P_d and D_c affect the accuracy of event localization. Events to be detected and localized were randomly placed in 15 edges in I_3, for different runs. This was repeated 15 times for different values of D_c

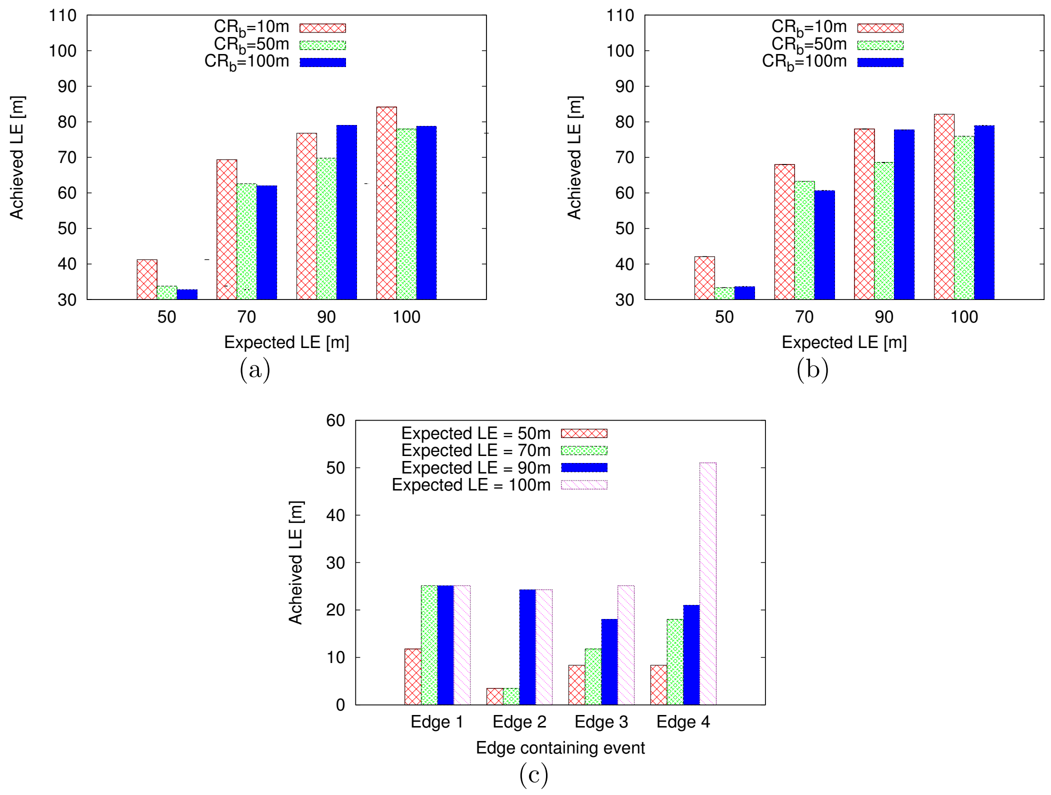


Figure 7.30: The actual localization error for different communication ranges varying with LE on zone of interest I_3 with (a) FN=0.2 and (b) FN=0.1 and the event type is non-diffusive. (c) Localization error LE when events are placed in different edges on I_3.

and P_d and the results were averaged.

Table 7.2: Number of beacons corresponding to a given P_d in I_3

P_d	#beacons
0.75	34
0.85	36
0.95	41
0.99	70

The algorithms evaluated for Event Localization, for different values of D_c, are:

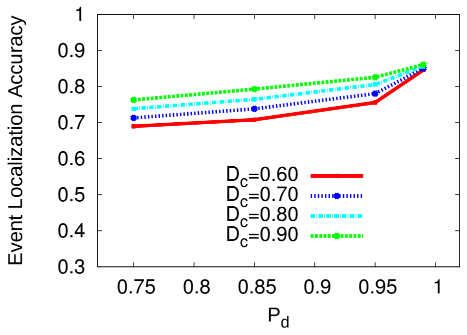


Figure 7.31: Evaluation of event localization as a function of P_d and D_c for estimated flows.

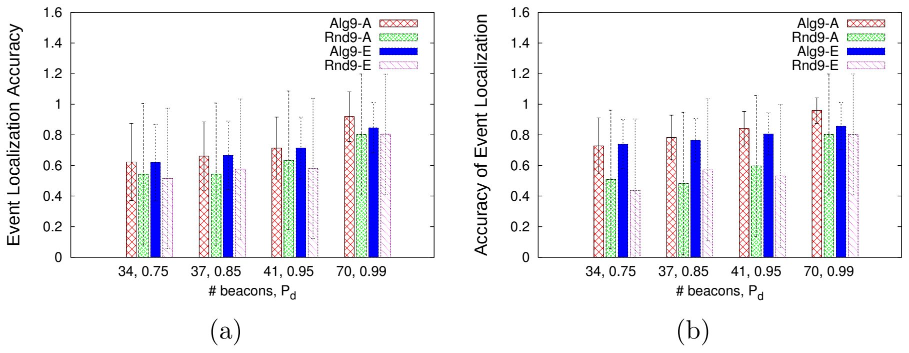
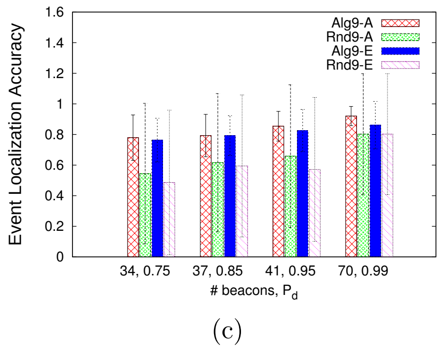



Figure 7.32: (a) Evaluation of event localization as a function of P_d for: (a) D_c=0.6; (b) D_c=0.8; and (c) D_c=0.9

i) Alg9-A, which executes Algorithm 9 if actual flows are known; ii) Alg9-E, which executes Algorithm 9 with flows being estimated; iii) Rnd9-A, which randomly deploys a given number of beacons and sensors, if flow information is known; and iv) Rnd9-E, which randomly deploys a given number of beacons and sensors, with flows being estimated. We note here that D_c influences the coverage of the zone of interest, i.e., if the event is detected or not. When the event is not detected, event localization is not possible.

The results for accuracy of event localization, given different D_c and P_d, are depicted in Figure 7.31. As shown, the achieved event localization accuracy is smaller than the P_d, for lower D_c. Nevertheless, this result is expected, since the sensing coverage is not 100% (for 60% sensing coverage, with 75% detection probability of detection, the expected event localization accuracy is ∼45%). We remark here again that, although D_c does not affect the placement of beacons, the event localization accuracy depends on D_c. The reason for this is the fact that for localization, edges *with events* need to be covered.

As presented in Table 7.2, which shows the number of beacons inserted for a given value of P_d in I_3, a higher P_d requires a higher number of beacons. To achieve a P_d=99% we would require 70 beacons, placed in all 70 vertices of I_3. For a P_d=95%, we would only require 41 beacons.

The results comparing event localization accuracy of Alg9-A, Alg9-E, Rnd9-A and Rnd9-E, for P_d=0.6, 0.8 and 0.9, are presented in Figures 7.32(a), 7.32(b) and 7.32(c), respectively. The number of beacons placed by Rnd2-A and Rnd2-E are those mentioned in Table 7.2, for different P_d.

As expected, Rnd9-E and Rnd9-A have large standard deviations. These random deployments always performed worse than Alg9-E and Alg9-A, for all values of P_d and D_c. As shown in Figure 7.32, for smaller D_c the standard deviations are also large for Alg9-A and Alg9-E, since some of the edges with events are not covered.

The results depicted in Figure 7.32 (a), (b), and (c) consistently show that for higher P_d, the accuracy of event localization is higher for all Alg9-A and Alg9-E. Interestingly, the improvement in event localization accuracy for Alg9-A and Alg9-E is significantly larger when D_c is small (e.g., for D_c=0.6, event localization accuracy improves from 62.2% to 91.8%, whereas, when D_c=0.9, event localization accuracy improves from 77.9% to 92.1%).

When P_d=0.99, beacons are placed on all vertices. The event localization accuracy is still not 100% since the sensing coverage is not 100%. The results validate that increasing P_d increases the event localization accuracy for all values of D_c. We verified (but not depicted here) that when beacons were placed on all vertices, and when sensing coverage was 100%, the accuracy of event localization was 100%.

To compare LE and localization accuracy, we map P_d to LE by calculating the radius of the circle covering the midpoints of these edges. For a given number of beacons, the LE achieved with the P_d requirement is very high compared to our solution (average of 36% lower LE by CPWDSim). E.g., when 34 beacons are placed, we achieve a LE of less than 90m, whereas the best result with the P_d requirement was 202m (55.44% decrease). Mapping LE to the event localization accuracy shows that CPWDSim yields lower accuracy for the same number of beacons. However, we believe that practically, it is more useful to provide a radius of error rather than a set of edges distributed over a large area.

The sensor and beacon placement formulation determined in the same formulation (*joint* formulation) is solved using a mixed integer nonlinear solver MINLP [30], which implements a Branch and Bound(*minlpBB*) algorithm. Two separate solutions, denoted as *greedy*, and *linear*, are implemented in *AMPL* [31]. The sensor deployment algorithm aims to minimize the *esle* in *greedy* and to minimize *ubele* in *linear*. All formulations are solved by a MINLP solver, an online service for solving numerical optimization problems [20] [23] [33]. The MINLP solver is capable of solving a wide range of linear and nonlinear optimization problems.

The evaluation of sensing coverage solutions is done to compare our algorithms with MNS. We note here that to the best of our knowledge, there is no current work in localizing leak/backflow events using static access points in the manner in which it is used in this work. Models are run on a desktop with 3.7GiB Ram and Intel Core 2 Duo CPU $E8400@3.00GHz \times 2$.

The input parameters of our algorithms are the price of devices and cost budget. To simplify the calculations, these parameters are expressed as integer multiples of the price of sensors and beacons. We assume the price of a beacon to be an integer (say, c) times the price of of a sensor. A sensor's price is normalized to be 1, thereby making the price of a beacon c. Throughout this evaluation, unless otherwise mentioned, the price of a beacon is 3. The total cost budget varies from 10 to 70 units with an interval of 10. When generating the data file for each zone in Micropolis example, we use the flow rates generated at hour 7 of the EPANET simulation, and eliminate edges whose flow probability is less than 1%.

In order to compare the performance of an algorithm on different networks, we use the normalized *sle* instead of the absolute *sle* we defined before. Normalized *sle* is calculated by dividing absolute *sle* over the sum of leak probabilities of all edges. We denote normalized *sle* as *normalized sle* in subsequent discussion. The same applies to *ale*. The normalization is done so as to make the metric uniform among all network sizes.

When comparing our separate solutions to the joint solution, the cost budget is an input parameter. The metrics we use are the normalized *sle*, sensing coverage, detection rate, and normalized *ale*. For the Micropolis example, we run simulations on a group of edges (covering more than 75% edges) for each zone. For each run, we set an event on an edge and measure the detection rate, *sle* and other metrics

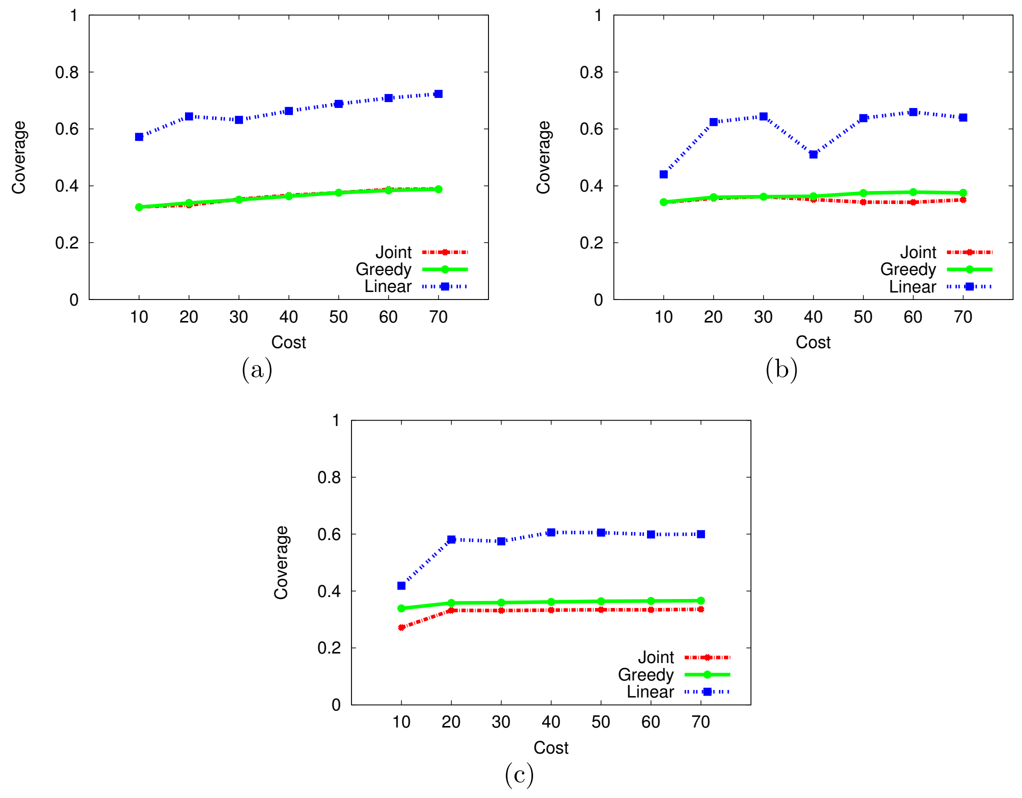
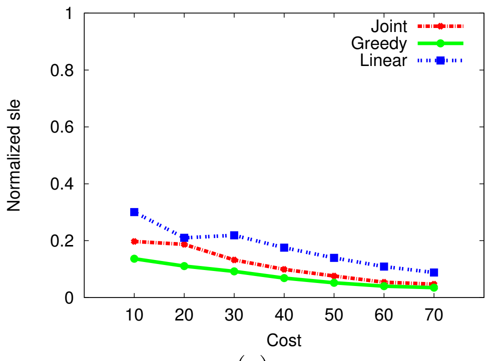


Figure 7.33: Comparison of average sensing coverage determined mathematically for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

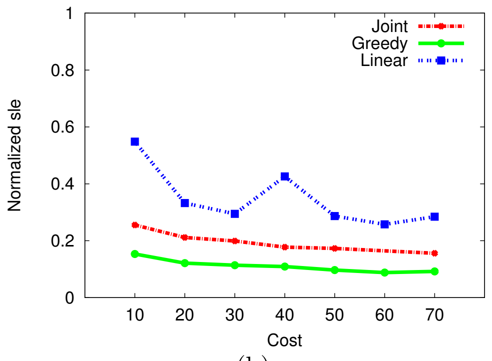
among the group of edges. Each point is the average of 30 runs.

We finally address the issue of maximizing sensing coverage in a given zone of interest with mobile sensors, since it is an essential prerequisite for locating leaks/backflows. It is also important when the utility manager is interested only in determining the presence, not the location of the leak/backflow. We compare our solution of MLBSC with MNS. The placement of sensors concerning the sensing coverage is determined using CPWDSim for the MNS problem, CPLEX in MATLAB for the MLBSC problem, and a greedy heuristic implementation in MATLAB for the MASC problem.

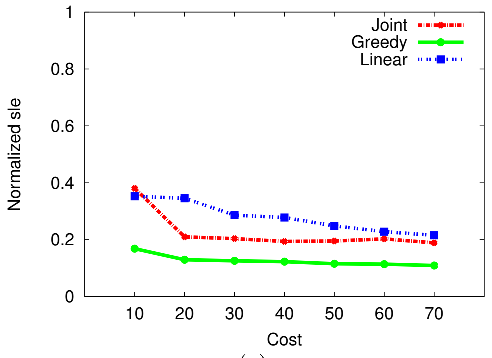
For the three sensor deployment algorithms, the parameter that we varied is



(a)



(b)



(c)

Figure 7.34: Comparison of normalized *sle* determined mathematically for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

the number of sensors. For comparison, the metrics we use are: (i) average sensing coverage (ASC); (ii) lower bound sensing coverage ($LBSC$). We set the total number of sensors to 8 different values based on 8 degrees of coverage inputs to MNS (0.2, 0.3, ... 0.9) for the 3 different zones of interest in the Micropolis example.

Once the sensor and beacon placements (i.e., **S** and **B**) are obtained from the MINLP solver, we plug in the values in the expression for *sle* from Equation 5.6 to obtain a theoretical value of the *sle* using the expression of *esle* from Equation 5.14 to obtain the theoretically expected sensing coverage. Figures 7.33 and 7.34 indicate average sensing coverage and normalized *sle* obtained mathematically for each of the three zones of interest in Micropolis network.

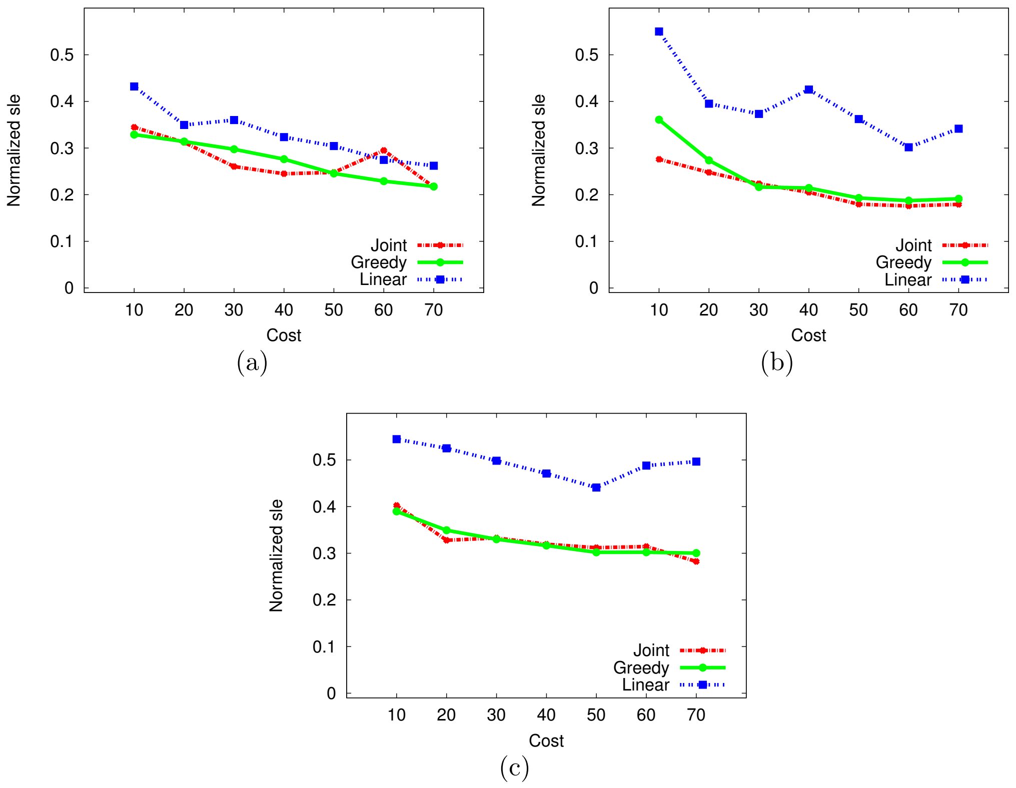


Figure 7.35: Comparison of achieved normalized *sle* in CPWDSim for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

As Figure 7.34 shows, the *sle* reduces when more budget is available. This is expected in all the formulations because the objective in the *joint* formulation is to minimize *ale*, and *ale* is directly related to *sle*. The term *sle* only places more emphasis on coverage as compared to *ale*. Minimizing only *ale* can also have a similar effect on *sle*. In case of the *linear* formulation, we select the best distribution of cost between sensors and beacons at each step of the exhaustive search. However, *sle* decreases with increasing budget except in a very few outlier cases (only 1 in this case). This may be attributed to the difference in *where* the same number of sensors are placed as compared to the case with lower cost. As Figure 7.33 shows, the coverage at a cost of 40 units in Zone 2 is lower than that for cost of 30. This

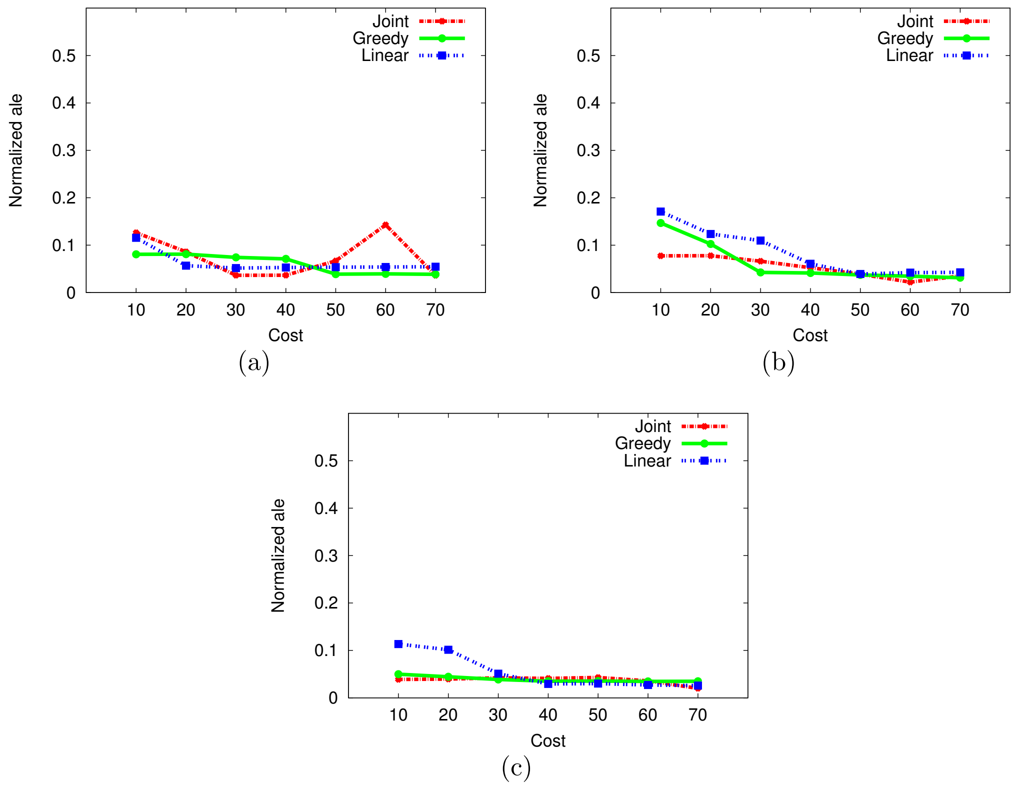
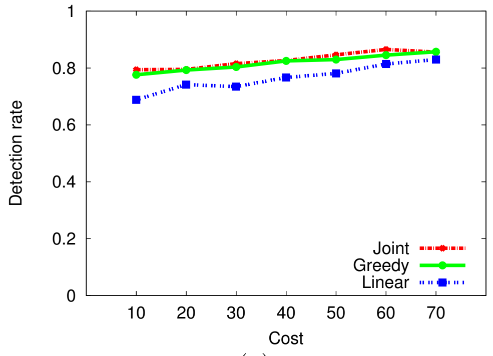


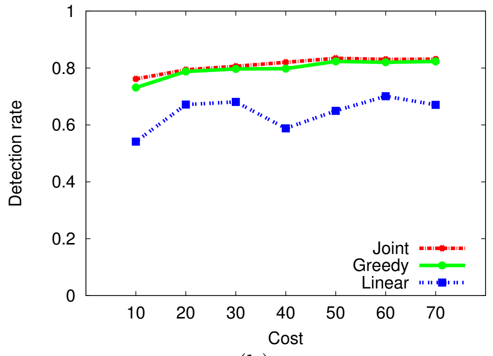
Figure 7.36: Comparison of average normalized *ale* for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

is possible because the *linear* formulation maximizes the minimum edge coverage, thereby placing more emphasis on those edges with smaller flows and leak probability. Also, some of the extra cost is spent on placing more beacons. The lower coverage has a direct impact, therefore, on *sle*. However, as the figures indicate, such variations are more of an outlier than the norm.

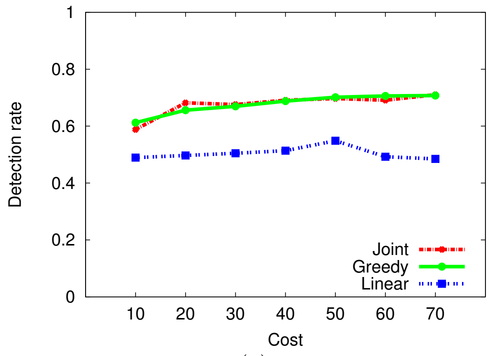
Figures 7.35 - 7.38 indicate that results obtained through simulations match the mathematically determined values. Figures 7.35 and 7.36 indicate that normalized *sle* decreases with increasing the available budget as expected, except for one case of Zone 1 when the cost is 60 units. Zone 1 has a relatively higher uneven distribution of flows at the junctions. For a cost of 60, there are only 11 beacons placed as



(a)



(b)



(c)

Figure 7.37: Comparison of detection ratio in CPWDSim for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

opposed to 16 beacons placed when a 50 unit budget is available. Theoretically, from Figure 7.34, the *sle* has to reduce. However, the budget put into increasing the coverage, so as to improve *sle*, has a smaller impact than the number of beacons might have had, as can be seen in Figures 7.38 and 7.37. This is because the flow distribution at junctions in Zone 1 has more variance than in Zones 2 or 3, and the increase in number of sensors is not sufficient to cover pipes with low flow rates. The increased cost invested in sensor placement is not sufficient to reduce *sle* as much as expected. This is also more of an outlier than the norm.

In general we notice that all three formulations present an increasing trend in coverage and detection rate and a decreasing trend in *sle* and *ale*, which is expected.

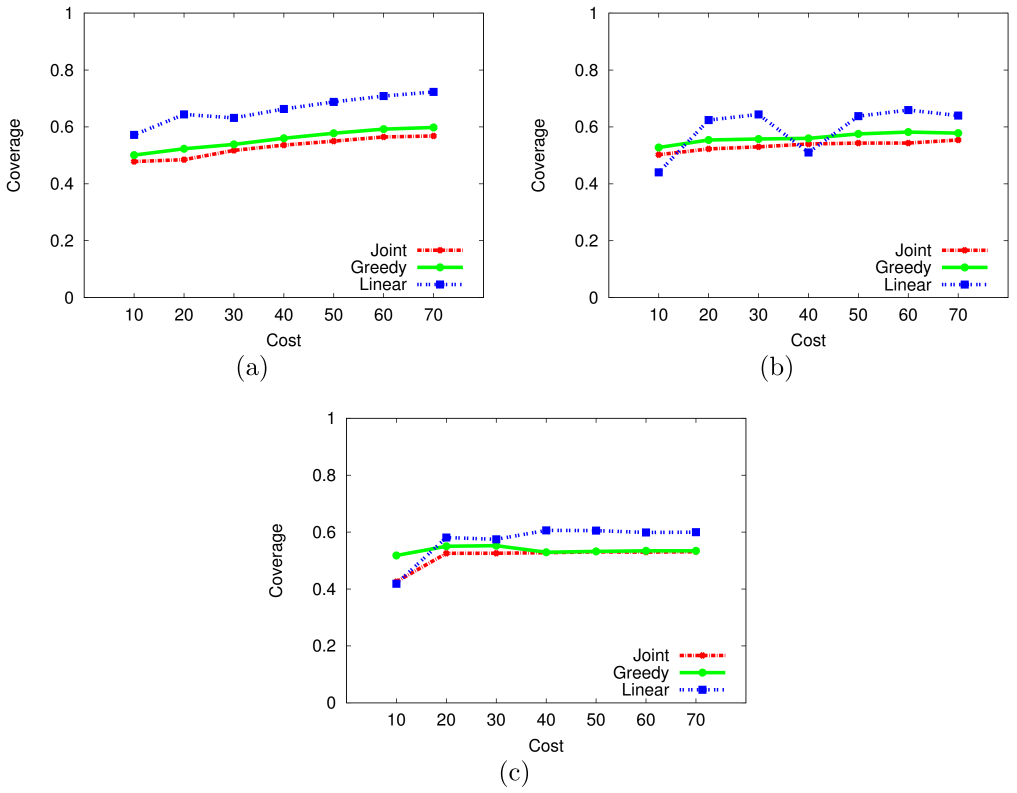


Figure 7.38: Comparison of coverage in CPWDSim for (a) Zone I_1, (b) Zone I_2, (c) Zone I_3

The *greedy* performs nearly as good as and sometimes better than the *joint* formulation although the *joint* formulation is more rigorous. The *linear* formulation is the computationally fastest of the three methods. However, it comes at a cost of not being as accurate or effective as the others. In simulations as well, *greedy* and *joint* solutions perform very well. Additionally, the *greedy* solution has consistent results with no outliers.

7.5 Performance Evaluation of Control

7.5.1 Evaluation of Flow Learning

To evaluate the performance of our flow learning algorithm, we executed 16 simulation runs on I_3 with P_d=0.95 and D_c=0.6, 0.7, 0.8 and 0.9.

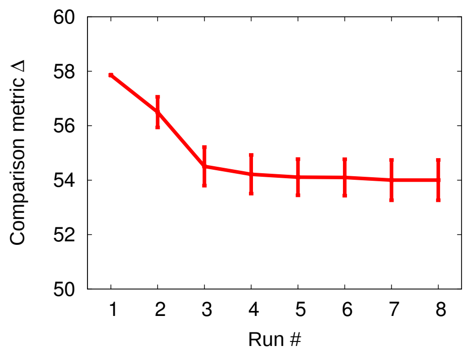
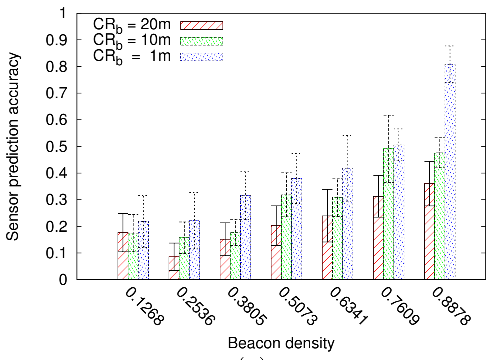
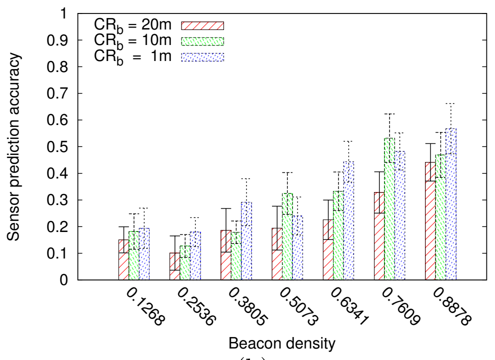


Figure 7.39: Evaluation of flow learning with a P_d=0.95, averaged over D_c=0.6, 0.7, 0.8, 0.9.

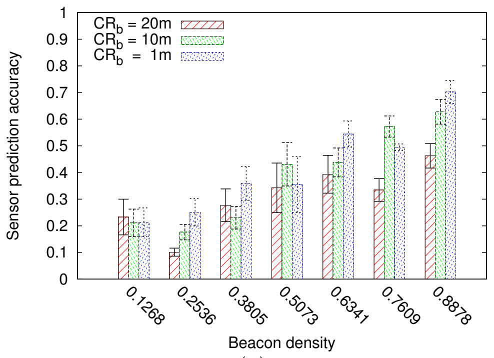
The results are presented in Figure 7.39. As shown, the difference between the actual and estimated flow networks reduces with the run number. For the first run, since the algorithm is executed with no flow learning, the difference Δ is the same, regardless of the D_c used (i.e., error bar is 0). As the algorithm learns actual flows from path synopses, the difference between the two flow networks (i.e., the actual flow network, versus the learned flow network) reduces. We have observed that when D_c is higher, Δ reduces, i.e., we learn the flows more quickly. We explain this result by the fact that more beacon pairs are covered, since more sensors are inserted (i.e., higher D_c). We are also noting an expected behavior of our flow learning algorithms. As the estimated flow gets closer to the actual flow, flow learning ceases to make an impact.



(a)



(b)



(c)

Figure 7.40: Sensor position accuracy for varying number of beacons and communication ranges with (a) Flow scenario 1 with average badness of the WDS 6.03, and of I, 19.484, (b) Flow scenario 2 with average badness of the WDS 5.984, and of I, 16.921, (c) Flow scenario 3 with average badness of the WDS 5.88, and of I, 16.847

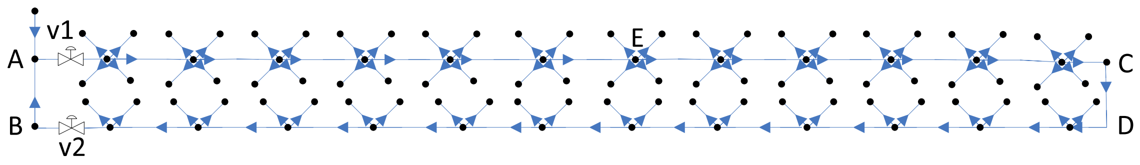


Figure 7.41: Part of the Micropolis virtual city used to demonstrate flow reversal

7.5.2 Evaluation of Sensor Position Estimation Algorithm

We first evaluate the sensor position estimation algorithm described in Algorithm 17. The results are depicted in Figure 7.40. The parameters varied here are

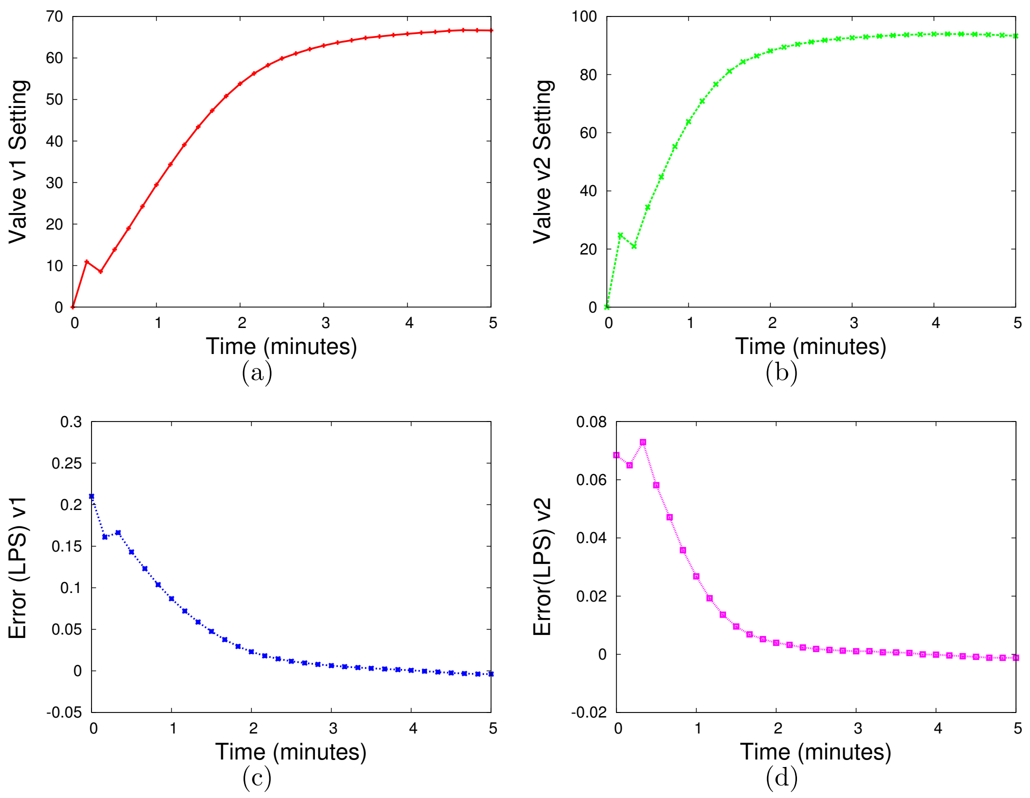


Figure 7.42: Valve settings for (a) $v1$, (b) $v2$, and error in valve setting for (c) $v1$, (d) $v2$ over time

the communication range from sensors to beacons, the number of beacons as represented by the beacon density (i.e., number of beacons divided by number of vertices), and different flow distributions. We quantify the flow distribution by a metric called badness, defined earlier.

We observe that having a higher beacon density improves the accuracy of sensor position estimation. Since the movement of sensors in the WDS is random, we sample the flow distribution at the vertices to predict the sensor's path (Lines 6 and 8 of Algorithm 17). Also, having a higher communication rate decreases sensor prediction accuracy. This is due to the fact that from a given position, a sensor may be able to communicate with multiple beacons. One of them is randomly chosen as the position

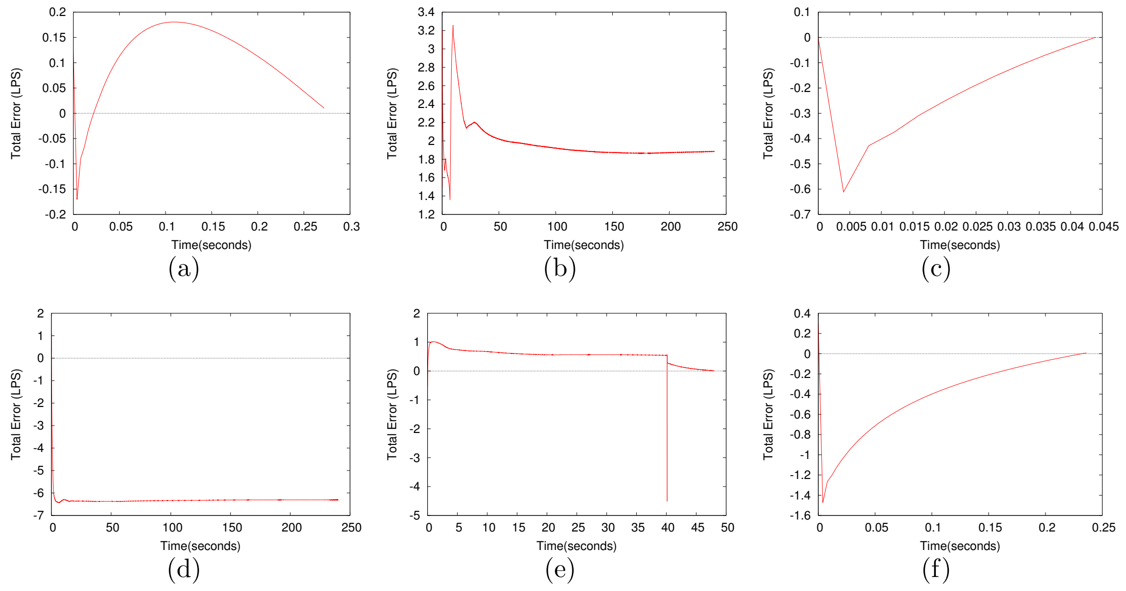


Figure 7.43: Total control error over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 1

of the sensor. We note here that the use of received signal strength to determine the approximate position of the sensor is also inaccurate here because the signal propagation is dependent on a variety of factors such as pipe and soil properties that cannot be modeled accurately.

An interesting observation here is that the accuracy dips for a beacon density of 0.2536. This is because at a density of 0.1268, the beacons are spaced out enough to ensure that sensors are in range of one beacon most of the time. However, increasing the beacon density make it more likely that the sensors see more than one beacon.

From practical experiments, we know that the communication rate is bound to be low. This experiment benefits from having a low communication range from sensors to beacons.

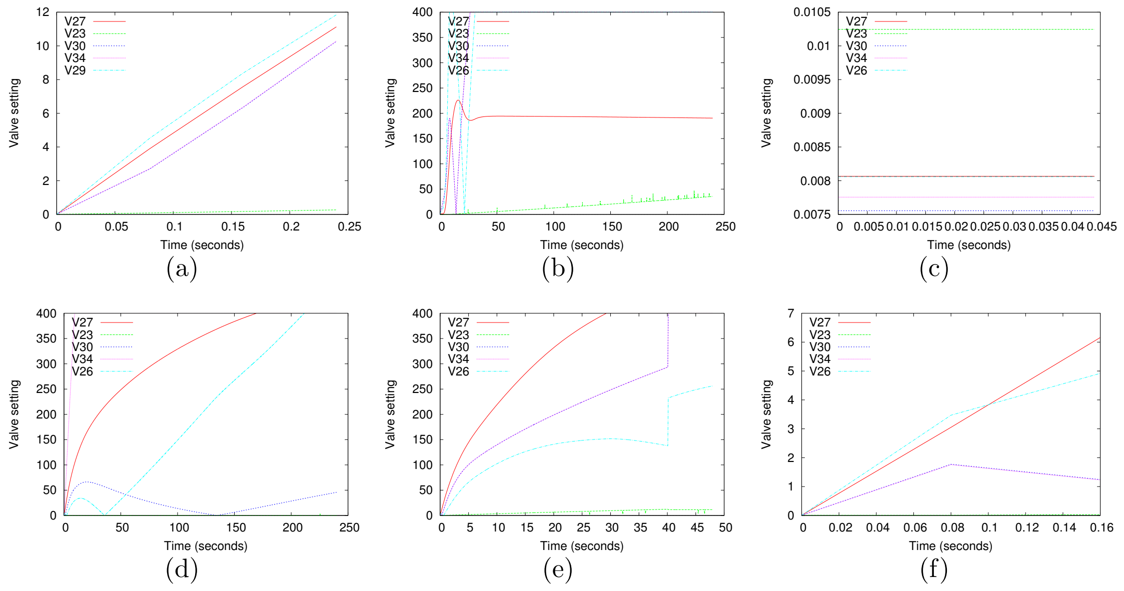


Figure 7.44: Select valve settings over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 1

7.5.3 Evaluation of CPWDS Flow Control

For evaluating our control component, we used a modified version of Micropolis WDS with two valves added, as shown in Figure 7.41. In this network, section ABCDE is the area of interest. We chose this section of Micropolis WDS because the flow in the pipes do not change direction during extended periods of time without throttling any valves.

As presented in Section 6, our solution uses a PID controller to throttle valves and change the flows. Throttling valve $v1$ reverses the flow in the pipe section AE, and throttling valve $v2$, reverses flow in the section BDCE. The metrics used here are valve settings, error, and convergence time.

Figure 7.42 shows the valve setting and their corresponding error plots for valves $v1$ and $v2$. The valve settings are the control variables, as described in Section 6.

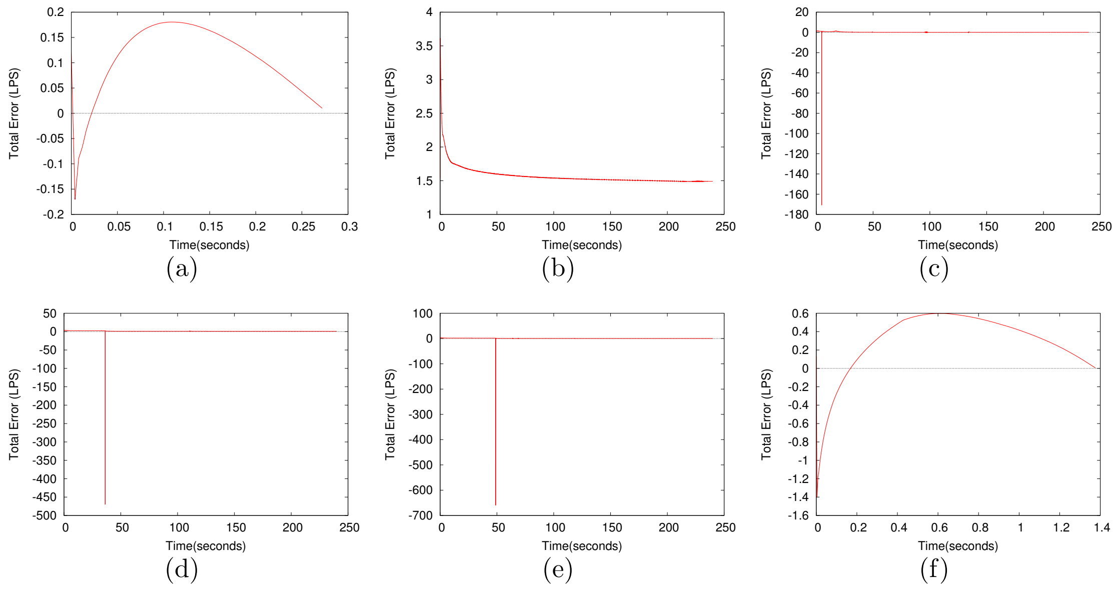


Figure 7.45: Total control error over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 2

We observe that valve setting for $v1$ converged to ∼66 and $v2$ converged to ∼93. The valve setting in $v2$ is higher due to flow reduction, which requires more valve throttling than $v1$. Controlling $v2$ reached a steady value faster than $v1$ due to the change in the hydraulic behavior of the system. The error approached zero faster with $v2$ than with $v1$. We observe that in both cases, the error approaches zero within 5 minutes (convergence time ∼ 5 minutes), indicating that the controller action is satisfactory.

With this preliminary result, we further evaluate the outer feedback loop on the entire Micropolis city model [10]. We simulate the WDS for 24 hours with and without the control system. When we use the flow controller, we run the simulation for 1 hour, and collect the estimated sensor positions to determine the reference flows. Using these reference flows, we run the PID controller until the total error reaches

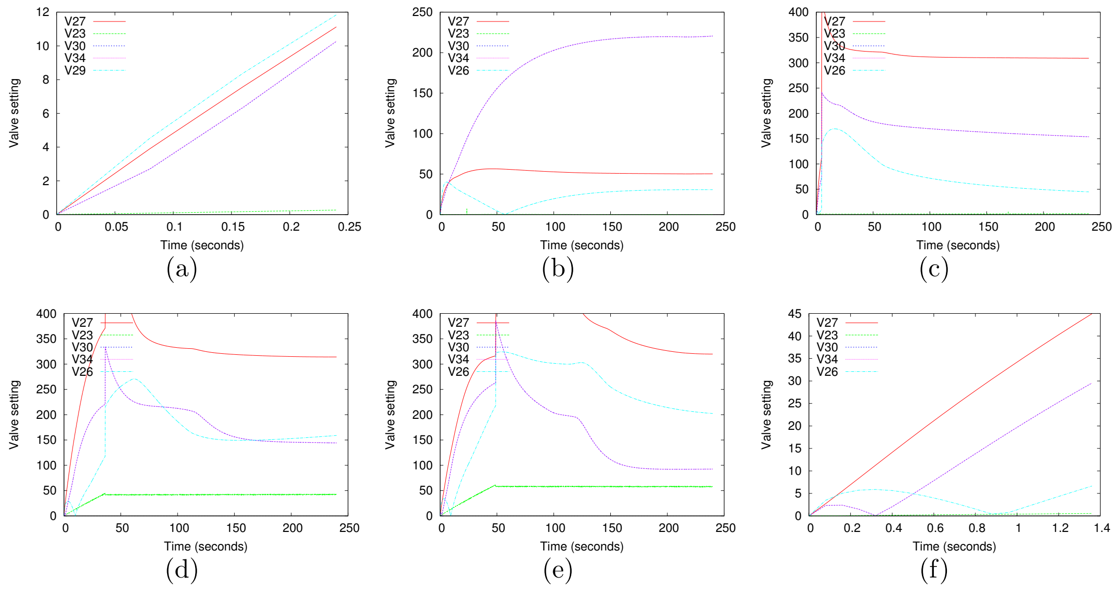


Figure 7.46: Select valve settings over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 2

0 (with an error margin), or until five minutes, whichever is sooner. The reason to put a time limit is because at times, the reference flows cannot be achieved. Also, we foresee using this flow controller in a WDS. The controller should then be able to give us a quick result. Since the reference flows are set without considering the physical limitations of the WDS, we remark that the flow controller may not always be able to achieve the reference flows.

Figures 7.43, 7.45, 7.47, and 7.49 depict the total errors over time for four different runs. During each run, the sensors move differently, and hence, the reference flows differ. For select valves, the valve settings are also shown in Figures 7.44, 7.46, 7.48, and 7.50. From these results, it is observed that the errors do not always reach 0. We note here that the valves in Micropolis are throttle control valves. Their settings are the friction factor. A higher friction factor implies that the valve is still open,

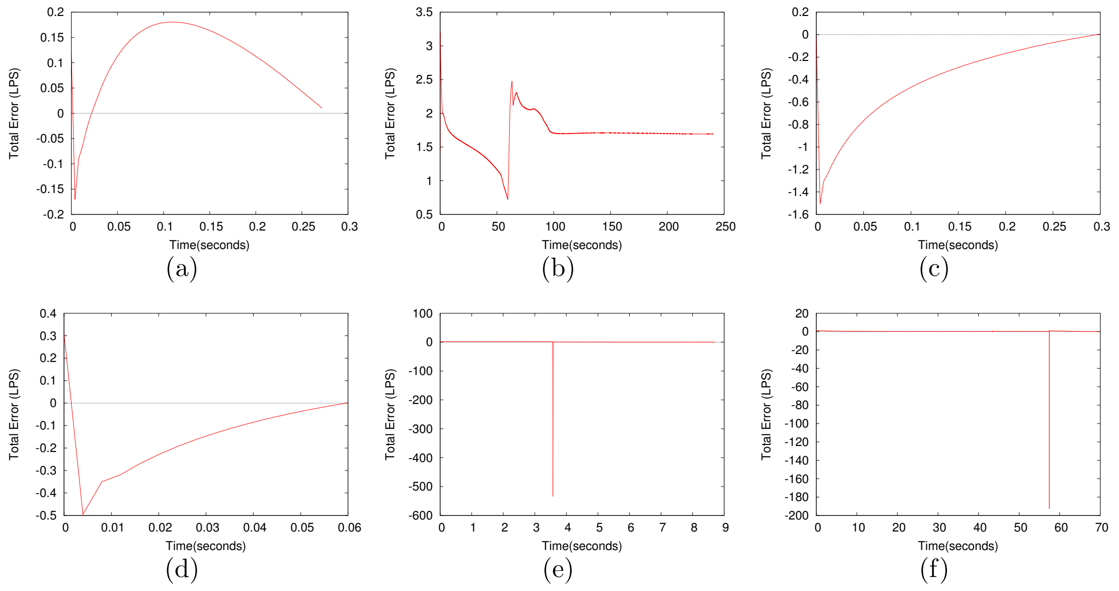


Figure 7.47: Total control error over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 3

but offers high friction, thereby restricting the flow. Hence, in few cases, we see that the valve setting keeps increasing.

The desired result of the flow controller is that the coverage of the zone of interest should be ensured in long term. To demonstrate this, we study the coverage at every hour when the WDS is controlled, and when it is not. The results are depicted in Figure 7.51(a) and 7.51(b). The four runs for the controlled cases correspond to the figures depicting errors and valve settings. As we observe, without control, the coverage of the zone of interest reduces to 0 eventually, since the sensors may move to pipes from which they do not return to the zone of interest without valve throttling. However, when we include the flow controller, we are able to ensure a good coverage by bringing the sensors back into the zone of interest.

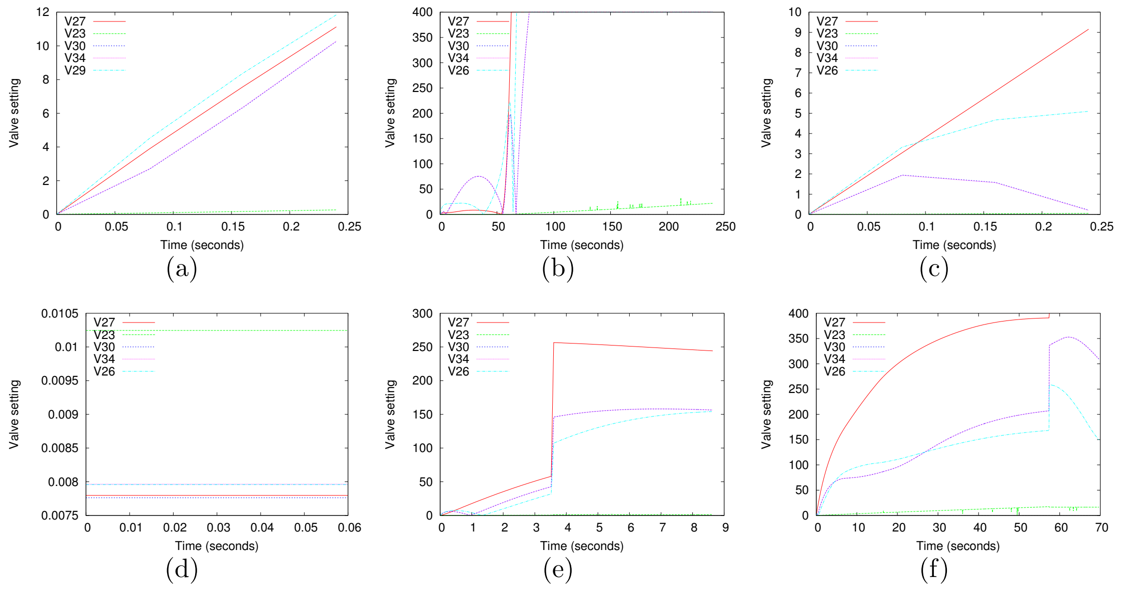


Figure 7.48: Select valve settings over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 3

7.5.4 Evaluation of CPWDS Sensor Control

Sensor control mechanism is evaluated on a small WDS example in EPANET, as shown by Figure 7.52. We simulated the WDS with and without sensor control for 10 runs each, for a period of 1 hour. We observed that with sensor control, the sensing coverage achieved is 84.9% with a standard deviation of 10.86%, and without sensor control, the achieved sensing coverage is 69.1% with a standard deviation of 10.21%. This shows us that the sensor control mechanism is beneficial and there is a need to develop sophisticated models for the same.

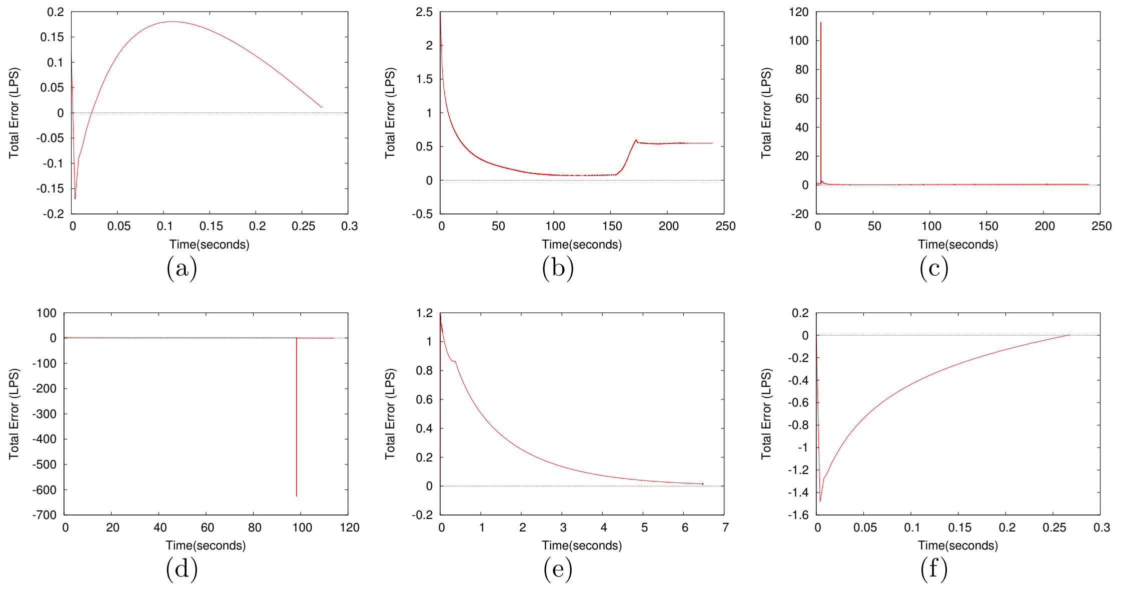


Figure 7.49: Total control error over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 4

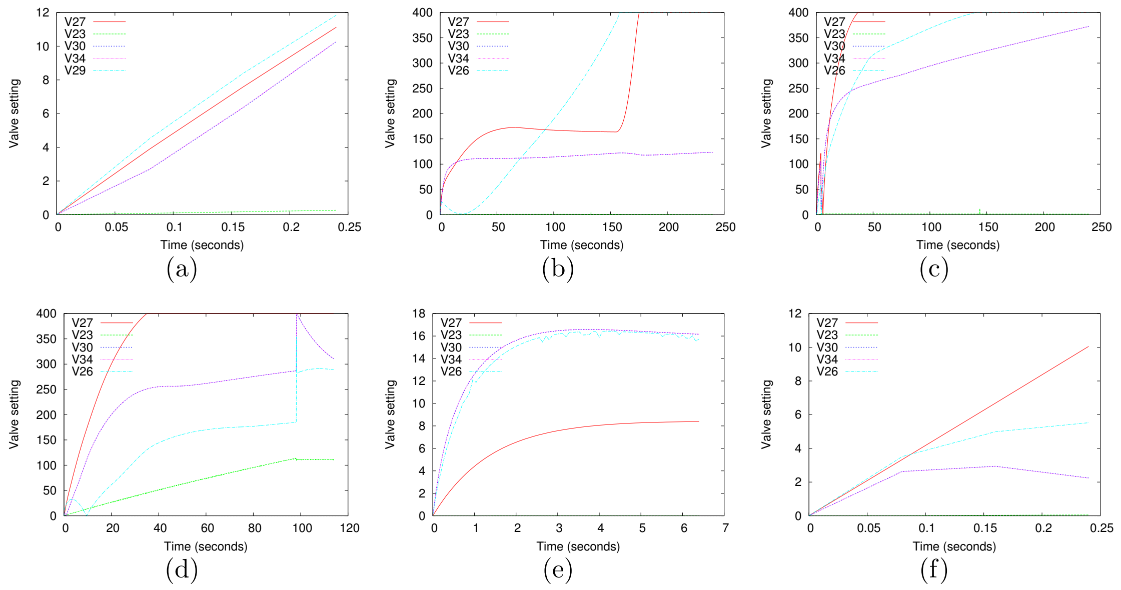
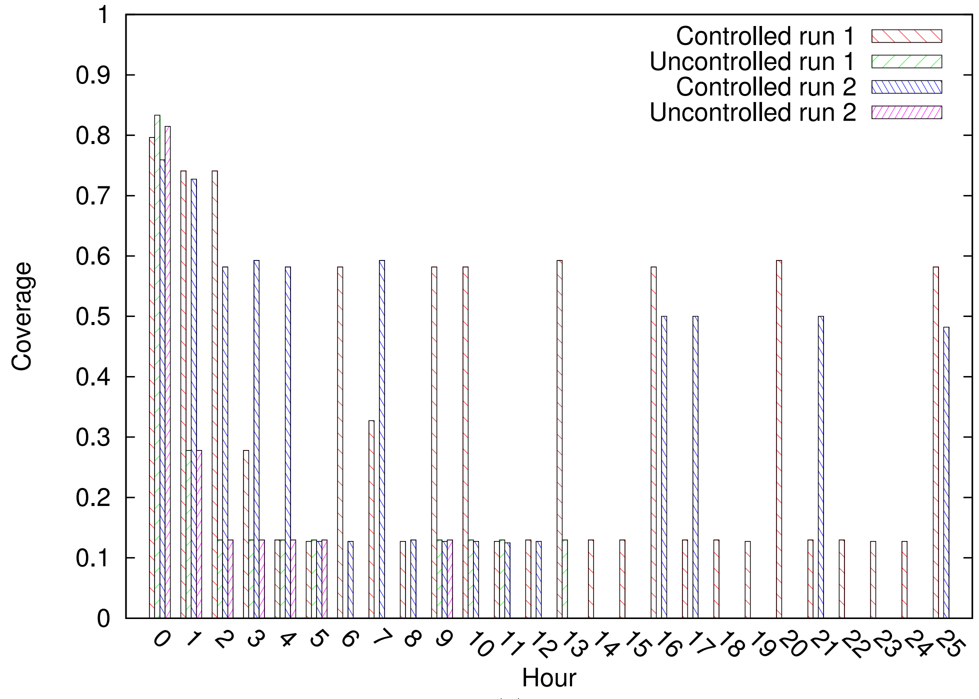
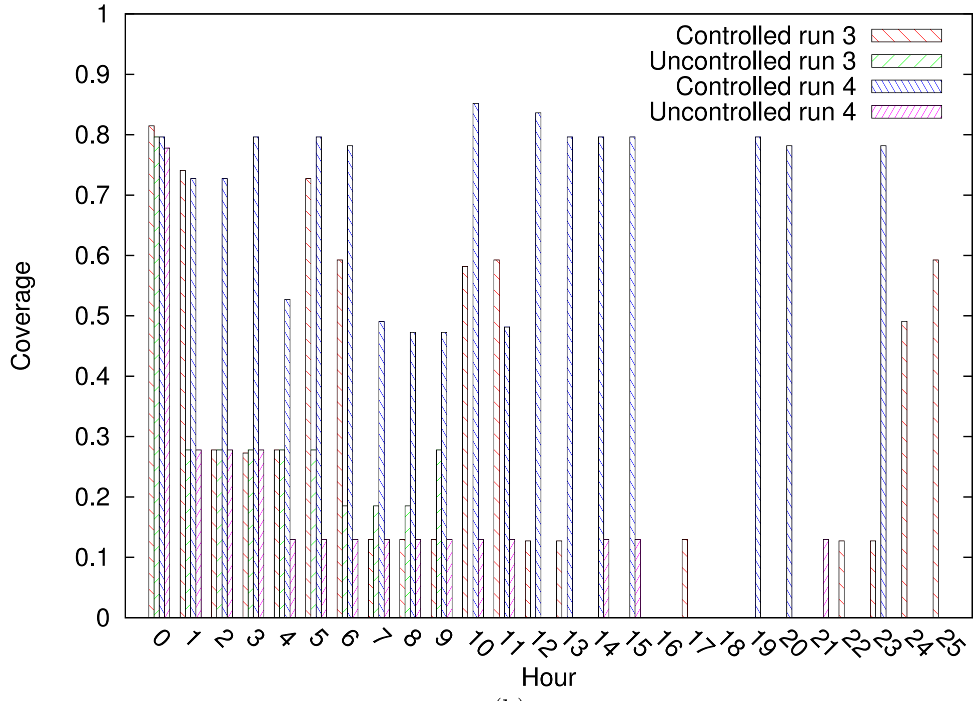


Figure 7.50: Select valve settings over number of iterations after (a) 0 hour, (b) 4 hours, (c) 8 hours, (d) 12 hours, (e) 16 hours, (f) 20 hours through the day during run 4



(a)



(b)

Figure 7.51: Comparison of coverage with and without flow control

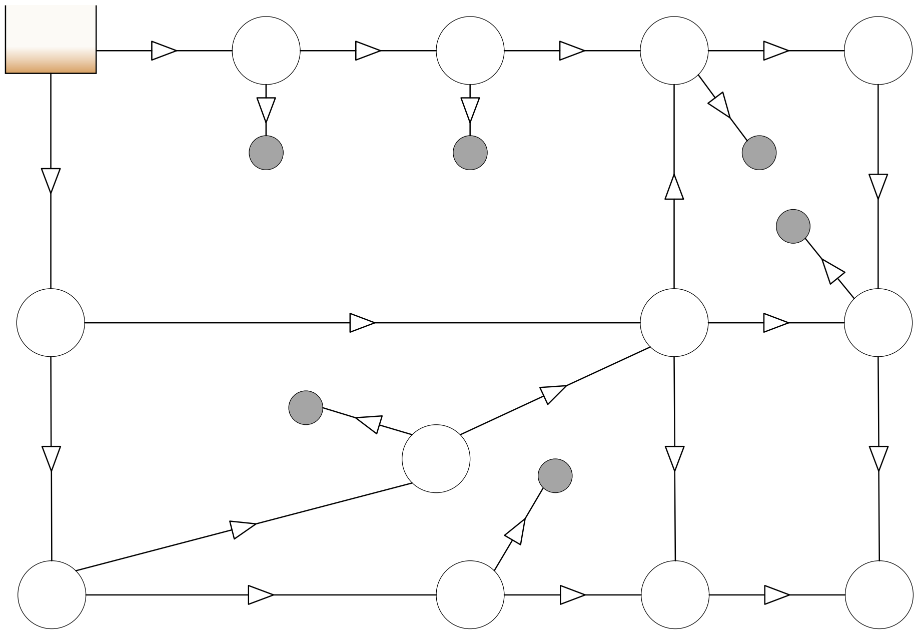


Figure 7.52: Sample WDS on which sensor control is evaluated

8. CONCLUSIONS AND FUTURE WORK

In this section, we conclude the dissertation and present future work in this field.

8.1 Conclusion

Water distribution systems (WDS) are posed with threats such as chemical contamination, leaks, and backflow events. Considering the importance of WDS, monitoring them to identify the presence and location of the threats is essential. In this dissertation, we tackle the problem of WDS monitoring from a cyber-physical systems approach.

We propose a Cyber-Physical Water Distribution System (CPWDS) comprising of free-flowing mobile sensors that move along with the flow of water in the pipes. To obtain information from mobile sensors, and to aid in localization of the threats, we propose the use of static access points, called beacons, at strategic locations outside the pipeline. The sensors communicate among themselves using the underwater acoustic medium, and to beacons placed outside through RF. To ensure that sensors continuously sense a region of interest, we control the flows in the pipes by throttling valves.

Every CPS has three key components, namely, communication, computation, and control. We have contributed to all three components of the CPWDS. From the communications perspective, it is challenging to design protocols for underwater acoustic communication due to the high propagation delays. We have overcome this challenge and presented a cross-layer group management/MAC protocol for communication among sensors. Computation challenges in our CPWDS are mainly due to the complexity of the problems (e.g., the problem of optimally deploying sensors is NP-hard). The computation problems tackled in this dissertation are: deployment of sensors, placement of beacons, and a global view algorithm. We have presented heuristics and optimal algorithms for computational problems of the CPWDS. To the best of our knowledge, there is no state of the art work that solves these problems for our CPWDS. Finally, flow control in WDS is challenging due to the non-linear nature of the WDS, and since most WDS are extremely large and complex. We have designed a flow controller that uses input from the data collected at beacons, and decides the valves to throttle through a PID controller. We have also presented an algorithm to track the position of the sensors based on their communication with beacons, and a heuristic to determine the desired flows to ensure coverage to support the control system.

We have classified WDS monitoring based on the nature of monitoring (on-demand and continuous), and the level of knowledge about the WDS (flow-aware and flow-unaware). We have studied how each of our contributions fits into these classifications.

Finally, we have validated our proposed ideas through proof-of-concept testbeds, and extensive large scale simulations on a simulator called Cyber Physical Water Distribution System Simulator (CPWDSim). CPWDSim is designed to mimic the movement of sensors through the pipes of a WDS. It uses input from EPANET [27], a software that solves the hydraulics for a WDS. We have implemented most of the algorithms in CPWDSim (e.g., some optimization problems were solved using MATLAB and other non-linear solvers), and equipped CPWDSim to accept the result of other solvers. It also simulates the communication among sensors and between sensors and beacons. From our proof of concept testbeds in communication, we validated that the path loss exponent assumed in our design is valid. Through our simulations, we have evaluated the different algorithms for sensor deployment, beacon placement, sensing models, communication models, and the group management protocol using global knowledge. Mainly, we notice that sensing range has a huge impact on the number of sensors. Also, we observe that the number of beacons and communication ranges impact the event localization and estimation of sensor locations. Our control system is observed to converge and improve the coverage over time.

8.1.1 Realism of Our CPWDS

We envision a long lasting CPWDS continuous monitoring the WDS, and hence we require that the batteries of these devices last long. Since the water in the pipes are pressurized, the energy from the flow of water may be harvested, especially when the sensors are temporarily stuck in some part of the WDS. Preliminary designs have studied for the same [15]. The process of harvesting the energy from the water is bound to slow down the sensors. To overcome this challenge, recently the use of magnetic induction [36] is explored. These works support that long-term continuous monitoring of the WDS using our CPS approach is feasible.

Some questions about the realism also arise with respect to sensor control. We assume that sensors spinning in a particular direction modifies their mobility model. This is predicated on the sensor spinning around a fixed axis. In the WDS, the orientation of the sensor may be fixed by reducing the density in a part of the sensor (e.g., by introducing a small air bubble).

A relevant concern is that we may not be able to place sensors and beacons at all points in the WDS. Such restrictions may be introduced into the model by appropriately eliminating the vertices in our algorithms. We have performed some preliminary evaluations of the restriction on a small example network in EPANET, and observed that slightly restricting the placement of sensors and beacons does not harm the performance greatly.

8.2 Future Work

8.2.1 Flow-based Cyber Physical Systems

Flow-based systems are physical systems that contain an inherent flow in them (e.g., human circulatory systems, oil & gas pipelines) that also need to be monitored. A majority of the theoretical foundations of this dissertation is generic to any flow-based system. The communication and sensing models in our CPWDS may be designed for the specific physical system. Also, the challenges and requirements of each flow-based system is different. For example, oil & gas pipelines are long, and beacon placement at junction may not be sufficient to localize the events precisely. Mechanisms need to be designed to improve the monitoring in each flow-based system. Additionally, the design of the mobile sensor in each flow-based system will be different (e.g., recent advancement in nanotechnology is applicable for human circulatory system monitoring).

8.2.2 Implementation of the Cyber-Physical Water Distribution System

In this dissertation, our evaluations are mostly in simulation, with very few proof of concept experiments. To realize a real world CPWDS, it is necessary to evaluate and validate the theory on a testbed. We envision a testbed where mobility of sensors through different structures of junctions, communication among sensors and between sensor and beacons, sensing models, and flow control mechanisms may be evaluated.

Further, there are several engineering questions relating to the design of the sensors that are not fully addressed in this dissertation. The size, structure, sensing modality, sensitivity, form factor, and communication capabilities are important design issues that need to be addressed. E,g., we need to design the sensor such that it moves freely through the main pipes without constricting the flow of water, but does not enter pipes of smaller diameter.

The models presented in this dissertation are approximate, and validated through simulation. Such a testbed also gives way to designing more sophisticated sensing and mobility models.

8.2.3 Making Water Distribution Systems Controllable

Water Distribution Systems are typically complex and large. Assuming that it is possible to reverse the flows on every pipe in any WDS is unrealistic. On the other hand, adding redundancy to WDS by introducing more tanks that act as both sources and sinks is desirable. With several paths of pipes leading to each consumer, the failure of some components will not cause disruption in services. Placing control points at strategic locations is necessary. With the addition of redundant tanks, flow control on all pipes may be done with valve throttling alone. It is worthwhile to investigate where to include sources, controllable sinks, and valves in the WDS to improve the failure tolerance and the controllability of the WDS.

8.2.4 Improving the Accuracy of the CPWDSim

In a complex WDS, any new algorithms need to be evaluated in simulations, since real world implementations and testbeds are tedious. However, our simulator makes assumptions about several physical parameters. E.g., we have assumed that the propagation speed of the acoustic waves in the pipes is uniform. There has been research in studying the propagation delays in pipes of various materials. Given that the WDS data about a city also includes the information about the pipe materials, we may be able to dynamically decide the speed of propagation of the sound waves in our simulation. We leave the analysis and implementation of this simulation feature for future work.

REFERENCES

[1] W. Abbas, N. Ahmed, Usama C., and A. A. Syed. Design and evaluation of a low-cost, diy-inspired, underwater platform to promote experimental research in UWSN. *Ad Hoc Networks*, 2014.

[2] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, et al. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Proceedings of the International Conference on Distributed Computing and Networking (ICDCS)*, 2004.

[3] N. Ahmed, W. bin Abbas, and A. A. Syed. A low-cost and flexible underwater platform to promote experiments in uwsn research. In *Proceedings of the International Conference on Underwater Networks and Systems (WuWNet)*, 2012.

[4] N. Aitsaadi, N. Achir, K. Boussetta, and G. Pujolle. Differentiated underwater sensor network deployment. In *Proceedings of OCEANS Conference*, 2007.

[5] R. Ali, S. S. Lor, R. T. Benouaer, and M. Rio. Cooperative leader election algorithm for master/slave mobile ad hoc networks. In *Proceedings of the 2nd IFIP conference on Wireless days (WD)*, 2009.

[6] American Water Works Association. Buried no longer: confronting america's water infrastructure challenge. `http://www.awwa.org/infrastructure`. [Online; accessed 30-Oct-2013].

[7] American Water Works Association. Reinvesting in drinking water infrastructure. `http://www.win-water.org/reports/infrastructure.pdf`. [Online;

accessed 30-Oct-2013].

[8] M. K. Banks, A. Brovont, S. Pekarek, M. Porterfield, A. Salim, and R. Wu. Development of mobile self-powered sensors for potable water distribution. In *EPA Grant Number: R834868*, 2012.

[9] B. Benson, Y. Li, B. Faunce, K. Domond, D. Kimball, C. Schurgers, and R. Kastner. Design of a low-cost underwater acoustic modem. *IEEE Embedded Systems Letters*, 2(3), 2010.

[10] K. Brumbelow, J. Torres, S. Guikema, E. Bristow, and L. Kanta. Virtual cities for water distribution and infrastructure system research. In *World Environmental and Water Resources Congress*, 2007.

[11] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An Integrated Package for Nonlinear Optimization. In *Large-Scale Nonlinear Optimization*, 2006.

[12] Canadian commercial newswire. Pure technologies awarded multi year leak detection contract in qatar. `http://www.newswire.ca/en/story/1369537/pure-technologies-awarded-multi-year-leak-detection-contract-in-qatar`. [Online; Published 9-June-2014].

[13] M. Cardei, M. T. Thai, Y. Li, and W. Wu. Energy efficient target coverage in wireless sensor networks. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, 2005.

[14] D. Chatzigeorgiou, K. Youcef-Toumi, and R. Ben-Mansour. Design of a novel in-pipe reliable leak detector. *IEEE/ASME Transactions on Mechatronics*, 20(2), 2015.

[15] D. M. Chatzigeorgiou. Analysis and design of an in-pipe system for water leak detection. Master's thesis, Massachusetts Institute of Technology, 2010.

[16] X. Che, I. Wells, G. Dickers, P. Kear, and X. Gong. Re-evaluation of RF electromagnetic communication in underwater sensor networks. *IEEE Communications Magazine*, 48, 2010.

[17] Y-J. Chen and H-L. Wang. Ordered CSMA: a collision-free MAC protocol for underwater acoustic networks. In *Proceedings of OCEANS Conference*, 2007.

[18] C. Copeland and M. Tiemann. Water infrastructure needs and investment: review and analysis of key issues. Technical report, US Congressional Research Service (Report RL31116), 2010.

[19] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3), 1989.

[20] J. Czyzyk, M. P. Mesnier, and J. J. Moré. The neos server. *IEEE Computational Science and Engineering*, 1998.

[21] R.J. de Araujo MacEdo, A.E.S. Freitas, and A.S. de Sa. A self-manageable group communication protocol for partially synchronous distributed systems. In *Proceedings of the Latin-American Symposium on Dependable Computing (LADC)*, 2011.

[22] S.C. Dhongdi, K.R. Anupama, and L.J. Gudino. Review of protocol stack development of underwater acoustic sensor network (uasn). In *Proceedings of Underwater Technology (UT)*, 2015.

[23] E. D Dolan. Neos server 4.0 administrative guide. *arXiv preprint cs/0107034*, 2001.

[24] S. Dolev, E. Schiller, and J. L. Welch. Random walk for self-stabilizing group communication in ad hoc networks. *Transactions on Mobile Computing*, 5(7), 2006.

[25] C. D. D'Souza and M.S. M. Kumar. Integrated approach in the quantitative and qualitative control of water distribution systems through control systems. *Journal of Hazardous, Toxic, and Radioactive Waste*, 16(2), 2012.

[26] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19, 1972.

[27] Environmental Protection Agency. EPANET v2.0. Technical report, Environmental Protection Agency, 2006.

[28] G. Fan, H. Chen, L. Xie, and K. Wang. An improved cdma-based mac protocol for underwater acoustic wireless sensor networks. In *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2011.

[29] F. H. Fisher and V. P. Simmons. Sound absorption in sea water. *The Journal of the Acoustical Society of America*, 62(3), 1977.

[30] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for miqp branch-and-bound. *SIAM Journal on Optimization*, 8(2), 1998.

[31] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A mathematical programming language*. AT&T Bell Laboratories Murray Hill, NJ 07974, 1987.

[32] S.M. George, Wei Zhou, H. Chenji, M. Won, Y. O. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah. DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response. *Communications Magazine*, 48(3), 2010.

[33] W. Gropp and J. Moré. Optimization environments and the neos server. *Approximation Theory and Optimization*, 1997.

[34] X. Han, J. Yin, G. Yu, and P. Du. Experimental demonstration of single carrier underwater acoustic communication using a vector sensor. *Applied Acoustics*, 98, 2015.

[35] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu. An energy-efficient surveillance system using wireless sensor networks. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[36] D. Hoffmann, A. Willmann, R. Göpfert, P. Becker, B. Folkmer, and Y. Manoli. Energy harvesting from fluid flow in water pipelines for smart metering applications. *Journal of Physics: Conference Series*, 476(1), 2013.

[37] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: a distributed mobile sensor computing system. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*, 2006.

[38] O. Hunaidi, A. Wang, M. Bracken, T. Gambino, and C. Fricke. Acoustic methods for locating leaks in municipal water pipe networks. In *International Conference on Water Demand Management*, pages 1–14, 2004.

[39] K.P. Hunt, J.J. Niemeier, and A. Kruger. RF communications in underwater wireless sensor networks. In *Proceedings of the Internation Conference on Electro/Information Technology (EIT)*, 2010.

[40] R. Ingram, T. Radeva, P. Shields, S. Viqar, J. E. Walter, and J. L. Welch. A leader election algorithm for dynamic networks with causal clocks. *Distributed Computing*, 26(2), 2013.

[41] R. Ingram, P. Shields, J.E. Walter, and J.L. Welch. An asynchronous leader election algorithm for dynamic networks. In *Proceedings of the International Symposium on Parallel Distributed Processing (IPDPS)*, 2009.

[42] J.H. Kim, G. Sharma, N. Boudriga, and S.S. Iyengar. Spamms: a sensor-based pipeline autonomous monitoring and maintenance system. In *Proceedings of the International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2010.

[43] G. Kokosalakis, A.M. Gorlov, E. Kausel, and A.J. Whittle. Communications and power harvesting system for in-pipe wireless sensor networks. US Patent App. 20,070/209,865.

[44] K. B. Kredo, II and P. Mohapatra. A hybrid medium access control protocol for underwater wireless networks. In *Proceedings of the 2nd workshop on Underwater networks (WuWNet)*, 2007.

[45] P. M. Kumar and M. Kumar. Comparative study of three types of controllers for water distribution networks. *Journal American Water Works Association*, 101(1), 2009.

[46] T.T. Lai, Y.T. Chen, P. Huang, and H. Chu. Pipeprobe: a mobile sensor droplet for mapping hidden pipeline. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.

[47] T.T.T. Lai, W.J. Chen, K.H. Li, P. Huang, and H.H. Chu. Triopusnet: automating wireless sensor network deployment and replacement in pipeline monitoring. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2012.

[48] U. Lee and M. Gerla. A survey of urban vehicular sensing platforms. *Computer Networks*, 54, 2010.

[49] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi. Mobeyes: smart mobs for urban monitoring with a vehicular sensor network. *Wireless Communications*, 13(5), 2006.

[50] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDDM)*, 2007.

[51] X. Li, W. Yu, X. Lin, and S.S. Iyengar. On optimizing autonomous pipeline inspection. *IEEE Transactions on Robotics*, 28(1), 2012.

[52] J. Ma and W. Lou. Interference-aware spatio-temporal link scheduling for long delay underwater sensor networks. In *Proceedings of the International Conference on Sensing, Communication and Networking (SECON)*, june 2011.

[53] E. Magistretti, J. Kong, U. Lee, M. Gerla, P. Bellavista, and A. Corradi. A mobile delay-tolerant approach to long-term energy-efficient underwater sensor networking. In *Proceedings of the International Conference on Wireless Communications and Networking Conference (WCNC)*, 2007.

[54] U. Manohar and M.S. M. Kumar. Modeling equitable distribution of water: Dynamic inversion based controller approach. *Journal of Water Resources Planning and Management*, 140(5), 2013.

[55] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, 2001.

[56] S. Mekid, A. Khalifa, R. Mansour, S. Ho, and S. Sarma. Water leaks detection: Assessment of wireless communication through water and sand media in buried supply pipes. In *International Conference on Environmental Science and Technology (ICEST)*, 2012.

[57] F. Miao, S. Lin, S. Munir, J. A. Stankovic, H. Huang, D. Zhang, T. He, and G. J. Pappas. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. In *Proceedings of the Internation Coference on Cyber-Physical Systems (ICCPS)*, pages 100–109, 2015.

[58] C. Ocampo-Martinez, V. Puig, G. Cembrano, and J. Quevedo. Application of predictive control strategies to the management of complex networks in the urban water cycle [applications of control]. *IEEE Control Systems*, 33(1), 2013.

[59] K. Ogata and Y. Yang. *Modern control engineering.* Prentice-Hall Englewood Cliffs, 1970.

[60] A. Ostfeld and et al. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6), 2006.

[61] G. Owojaiye and Y. Sun. Focal design issues affecting the deployment of wireless sensor networks for pipeline monitoring. *Ad Hoc Networks*, 11(3), 2012.

[62] L. Perelman and A. Ostfeld. Operation of remote mobile sensors for security of drinking water distribution systems. *Water Research*, 47(13), 2013.

[63] L. Perelman and A. Ostfeld. Operation of remote mobile sensors for security of drinking water distribution systems. *Water Research*, 47(13), 2013.

[64] L. Perelman, W. Salim, R. Rouxi Wu, J. Park, A. Ostfeld, K.M. Banks, and D.M. Porterfield. Enhancing water distribution system security through water quality mobile sensor operation. In *World Environmental & Water Resources Congress (WEWRC)*, 2013.

[65] Pure Technologies. Pipediver. `http://www.puretechltd.com/products/pipediver/pipediver-pccp.shtml`. [Online; accessed 30-Nov-2013].

[66] Pure Technologies. Sahara leak & gas pocket detection. `http://www.puretechltd.com/products/sahara/sahara_leak_gas_pocket.shtml`. [Online; accessed 30-Oct-2013].

[67] Pure Technologies. Smartball for water and wastewater pipelines. `http://www.puretechltd.com/products/smartball/smartball_leak_detection.shtml`. [Online; accessed 30-Oct-2013].

[68] R. Puust, Z. Kapelan, D.A. Savic, and T. Koppel. A review of methods for leakage management in pipe networks. *Urban Water Journal*, 7(1), 2010.

[69] A. Rasekh and K. Brumbelow. A dynamic simulation-optimization model for adaptive management of urban water distribution system contamination threats. *Applied Soft Computing*, 32, 2015.

[70] I. Rhee, A. Warrier, M. Aia, J. Min, and M.L. Sichitiu. Z-MAC: A hybrid MAC for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 16, 2008.

[71] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. Self-stabilizing leader election for single-hop wireless networks despite jamming. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2011.

[72] S. A. Samad, S. K. Shenoy, G. Santhosh Kumar, and P.R.S. Pillai. A survey of modeling and simulation tools for underwater acoustic sensor networks. *International Journal of Research and Reviews in Computer Science*, 2, 2011.

[73] A.K. Singh and S. Sharma. Elite leader finding algorithm for manets. In *Proceedings of the International Symposium on Parallel and Distributed Computing (ISPDC)*, 2011.

[74] J. B. Stephenson. Drinking water: Experts' views on how federal funding can be spent to improve security. *U.S. Government Accountability Office*, Sept. 2004.

[75] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. PIPENET: A wireless sensor network for pipeline monitoring. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.

[76] R. Stoleru, T. He, and J. A. Stankovic. Range-free localization. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30 of *Advances in Information Security*. Springer US, 2007.

[77] Z. Sun, P. Wang, M. C. Vuran, M. A. Al-Rodhaan, A. M. Al-Dhelaan, and I. F. Akyildiz. MISE-PIPE: Magnetic induction-based wireless sensor networks for underground pipeline monitoring. *Ad Hoc Networks*, 9, 2011.

[78] M. Suresh, U. Manohar, A. G R, R. Stoleru, and M. Kumar M S. A cyber-physical system for continuous monitoring of water distribution systems. In *Proceedings of the International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2014.

[79] M.A Suresh, L. Smith, A Rasekh, R. Stoleru, M.K. Banks, and B. Shihada. Mobile sensor networks for leak and backflow detection in water distribution

systems. In *Proceedings of the International Conference on Advanced Information Networking and Applications (AINA)*, 2014.

[80] M.A Suresh, R Stoleru, R Denton, E Zechman, and B Shihada. Towards optimal event detection and localization in acyclic flow networks. In *Proceedings of the International Conference on Distributed Computing and Networking (ICDCN)*, 2012.

[81] M.A. Suresh, R. Stoleru, E.M. Zechman, and B. Shihada. On event detection and localization in acyclic flow networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3), 2013.

[82] M.A. Suresh, W. Zhang, W. Gong, A. Rasekh, R. Stoleru, and M.K. Banks. Towards optimal monitoring of flow-based systems using mobile wireless sensor networks. *ACM Transactions on Sensor Networks*, 11(3), 2015.

[83] A. A. Syed and J. Heidemann. Time synchronization for high latency acoustic networks-extended technical report. Technical Report ISI-TR-2005-602b, ISC, 2005.

[84] A.A. Syed, W. Ye, and J. Heidemann. T-lohi: A new class of mac protocols for underwater acoustic sensor networks. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, 2008.

[85] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[86] A. Tahbaz-Salehi and A. Jadbabaie. Distributed coverage verification in sensor networks without location information. *IEEE Transactions on Automatic Control*, 55(8), 2010.

[87] D. Trinchero and R. Stefanelli. Microwave architectures for wireless mobile monitoring networks inside water distribution conduits. *IEEE Transactions on Microwave Theory and Techniques*, 57(12), 2009.

[88] D. Trinchero and R. Stefanelli. Microwave mobile sensor networks within underground conduits filled of fluids. In *International Symposium on Electromagnetic Theory*, 2010.

[89] R.J. Urick and R.J. Urick. *Principles of underwater sound*, volume 3. McGraw-Hill New York, 1983.

[90] U.S Environmental Protection Agency. Aging water infrastructure research. `http://www.epa.gov/awi/`. [Online; accessed 30-Oct-2013].

[91] U.S. Environmental Protection Agency. Response protocol toolbox: Planning for and responding to contamination threats to drinking water systems. *Water Utilities Planning Guide-Module 1*, 2003.

[92] U.S Environmental Protection Agency. Control and mitigation of drinking water losses in distribution systems. *Publication No. EPA 816-D-09-001*, 2009.

[93] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3), 1979.

[94] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, 2004.

[95] L.F.M. Vieira, U. Lee, and M. Gerla. Phero-trail: a bio-inspired location service for mobile underwater sensor networks. *IEEE Journal on Selected Areas in Communications*, 28(4), 2010.

[96] M. Wälchli, P. Skoczylas, M. Meer, and T. Braun. Distributed event localization and tracking with wireless sensors. In *Wired/Wireless Internet Communications*. Springer, 2007.

[97] Y. Wang and H. Wu. Delay/fault-tolerant mobile sensor network (dft-msn): A new paradigm for pervasive information gathering. *IEEE Transactions on Mobile Computing*, 6(9), 2007.

[98] M. K. Watfa, S. Selman, and H. Denkilkian. Uw-mac: An underwater sensor network mac protocol. *International Journal of Communication Systems*, 23(4), 2010.

[99] A. J. Whittle, L. Girod, A. Preis, M. Allen, H. B. Lim, M. Iqbal, S. Srirangarajan, C. Fu, K. J. Wong, and D. Goldsmith. WATERWISE@SG: A testbed for continuous monitoring of the water distribution system in singapore. In *Proceedings of the Conference on Water Distribution Systems Analysis (WSDA)*, 2010.

[100] R. Wu, W. Salim, A. Rasekh, J. Park, A. Brovont, S. Pekarek, M. Porterfield, and K. Banks. Mobile sensor technology for monitoring water quality in drinking water distribution. In *World Environmental & Water Resources Congress*, 2014.

[101] S. Xiong, L. Yu, H. Shen, C. Wang, and W. Lu. Efficient algorithms for sensor deployment and routing in sensor networks for network-structured environment monitoring. In *Proceedings of the International Conference on Computer Communications (INFOCOM)*, 2012.

[102] A. Yazdani and P. Jeffrey. Complex network analysis of water distribution systems. *Arxiv preprint arXiv:1104.0121*, 21(1), 2011.

[103] E.M. Zechman. Agent-based modeling to simulate contamination events and evaluate threat management strategies in water distribution systems. *Risk Analysis*, 31, 2011.

[104] E.M. Zechman and S. Ranjithan. Evolutionary computation-based methods for characterizing contaminant sources in a water distribution system. *Journal of Water Resources Planning and Management*, 135, 2009.

[105] J. Zhang, K. Premaratne, and P.H. Bauer. A distributed self-organization algorithm for ad-hoc sensor networks. In *Proceedings of the International Conference on Wireless Communications and Networking Conference (WCNC)*, 2003.

[106] Z. Zhou, Z. Peng, J-H Cui, Z. Shi, and A.C. Bagtzoglou. Scalable localization with mobility prediction for underwater sensor networks. *IEEE Transactions on Mobile Computing*, 10(3), 2011.

APPENDIX A

PROOF THAT THE MNS PROBLEM IS NP-HARD

Minimization of Number of Sensors Problem (MNS): Given an acyclic WDS, represented as a flow network $\mathcal{F}$, a zone of interest I in $\mathcal{F}$, and degree of coverage D_c, find the set $S = \{(s_i, q_i) \mid s_i \in V \wedge q_i \in \mathbb{N}\}$ of insertion points s_i (sources) where sensors need to be deployed, and the number q_i of sensors to be deployed at s_i, such that their sensed paths cover each edge of I with a probability $\geq D_c$ and the number of sensors inserted, $\sum_{i=1}^{|S|} q_i$, is minimized.

Theorem 1: MNS is NP-Hard.

Take an instance of the Weighted Set Cover (WSC) problem $(\mathcal{E}, \mathcal{V}, \mathcal{S}, w)$ where:

$$\mathcal{E} = \{e_i \mid i = 1, 2, \ldots, n\}$$

$$\mathcal{V} = \{\mathcal{V}_j \mid j = 1, 2, \ldots, m; \mathcal{V}_j \subseteq \mathcal{E}\}; \bigcup_{j=1}^{m} \mathcal{V}_j = \mathcal{E}$$

$$\mathcal{S} \subset \mathcal{V} \mid \bigcup_{\mathcal{V}_j \in \mathcal{S}} \mathcal{V}_j = \mathcal{E}$$

$$\omega : \mathcal{V} \to \mathbb{R}$$

$$W = \sum_{\mathcal{V}_j \in \mathcal{S}} \omega_j$$

where $\mathcal{E}$ is a set of n elements, $\mathcal{V}$ is a set of m subsets of $\mathcal{E}$ covering all elements of $\mathcal{E}$, $\mathcal{S}$ is a subset of $\mathcal{V}$ that contains all elements of $\mathcal{E}$. Each subset $\mathcal{V}_j$ has a weight ω_j. $\mathcal{S}$ is constructed such that $\mathcal{E}$ can be covered with cost W.

We construct $f : \text{WSC} \rightarrow \text{MNS}$

$$V = \{w_j \mid j = 1, 2, \ldots, m\} \cup \{u_i \mid i = 1, 2, \ldots, n\}$$
$$\cup \{v_i \mid i = 1, 2, \ldots, n\}$$
$$E = E_1 \cup E_2 \text{ where}$$
$$E_1 = \{(w_j, u_i) \mid e_i \in \mathcal{V}_j;\ j = 1, 2, \ldots, m;\ i = 1, 2, \ldots, n\}$$
$$E_2 = \{(u_i, v_i) \mid i = 1, 2, \ldots, n\}$$
$$I = E_2\ ;\ D_c = 1$$

$$\mathcal{F}(u_i, v_i) = 1 \mid i = 1, 2, \ldots, n \text{ if } \exists (w_j, u_i) \mid j = 1, 2, \ldots, m$$
$$\mathcal{F}(u_i, v_i) = 0 \text{ otherwise.}$$
$$\mathcal{F}(w_j, u_i) = 1 \mid i = 1, 2, \ldots, n \text{ if } u_i \neq \text{first element in} \mathcal{V}_i$$
$$\mathcal{F}(w_j, u_i) = \frac{1}{\omega_j} \mid i = 1, 2, \ldots, n \text{ if } u_i = \text{ first element in } \mathcal{V}_i$$

where V, E, I, and D_c represent the vertices, edges, zone of interest and degree of coverage of FSN, respectively. $\mathcal{F}$ defines the flow in any edge. I should be covered with W sensors. Here, the set of edges E can be divided into two sets - E_1 represents the mapping of $\mathcal{E}$ and $\mathcal{V}$ using elements of the sets in $\mathcal{V}$, and E_2 represents the set of elements $\mathcal{E}$. The corresponding vertices are represented by the three subsets of V with labels u, v and w, as shown in the above construction.

The flows in $\mathcal{F}$ are such that if ω_j sensors are inserted in w_j, the edges in E_1 starting at w_j are covered, i.e., $(w_j, u_i) \mid e_i \in \mathcal{V}_j;\ i = 1, 2, \ldots, n$. Any sensor that reaches u_i also covers the edge (u_i, v_i) in E_2.

Note that V is constructed in $O(m)$ time, E and I in $O(n{+}m)$ time, $\mathcal{F}$ in $O(n{+}m)$

time and D_c in constant time. Hence, this construction occurs in polynomial time.

Equivalence: $\mathcal{S}$ covers $\mathcal{E}$ with cost $W \iff \forall e \in I$, e is covered by W sensors with D_c probability in $\mathcal{F}$.

$\Rightarrow$ Given $\mathcal{S}$ covers $\mathcal{E}$ with cost W. Since $D_c = 1$, all edges in I need to be covered with 100% probability. The number of sensors to be inserted in w_j, $j = 1, 2 \ldots m$ such that all edges incident on it are covered is $\omega_j \mid j = 1, 2 \ldots m$. Any sensor that reaches u_i will cover the edge (u_i, v_i). Since $\mathcal{S}$ covers $\mathcal{E}$ with cost W, selecting the corresponding vertices in $\mathcal{F}$ covers all edges in I with 100% probability. So, if $\mathcal{S}$ covers $\mathcal{E}$ with cost W, inserting W sensors in the corresponding vertices in $\mathcal{F}$, $\forall e \in I$, e is covered by the W sensors with D_c probability in $\mathcal{F}$.

$\Leftarrow$ Given all edges of I in $\mathcal{F}$ can be covered by W sensors with D_c probability. By our definition of E, all u_i's are covered by at least one w_j. Hence, any set of u_i in the set of insertion points can be replaced by an existing w_j without increasing the cost. Note that using our definition of E, each vertex w_j can be used to uniquely identify $V_j \in \mathcal{V}$. Further, each w_j covers a set of edges in I and each corresponding V_j covers a set of elements in $\mathcal{E}$. So, the sets corresponding to the insertion points ensure that $\mathcal{V}$ covers $\mathcal{E}$ with cost W.

APPENDIX B

APPROXIMATION RATIO OF THE GREEDY ALGORITHM FOR *MASC* PROBLEM

In this appendix, we prove that the approximation ratio of the greedy algorithm for Maximizing Average Sensing Coverage Problem ($MASC$) is $(1 + \frac{1}{e-1})$. Before the proof, we define four operations "$\cap$", "$\cup$", "$+$", "$-$", and one relation "$\in$" on vectors:

$$\mathbf{A} \cap \mathbf{B} = [min\{a_1, b_1\}, \ldots, min\{a_n, b_n\}]$$

$$\mathbf{A} \cup \mathbf{B} = [max\{a_1, b_1\}, \ldots, max\{a_n, b_n\}]$$

$$\mathbf{A} + \mathbf{B} = [a_1 + b_1, \ldots, a_n + b_n]$$

$$\mathbf{A} - \mathbf{B} = [max\{0, a_1 - b_1\}, \ldots, max\{0, a_n - b_n\}]$$

where $\mathbf{X} \in \mathbf{A}$ for $a_i \geq x_i$ for $0 \leq i \leq n$, and $\mathbf{A}$ and $\mathbf{B}$ are vectors in n-dimensional space, $\mathbf{X}$ is a base vector in n-dimensional space.

The objective function $f : \mathbf{S} \to \mathbf{R}$ is given by:

$$f(\mathbf{S}) = m - \sum_{j=1}^{m} \prod_{i=1}^{n} p_{ij}^{s_i}$$

where $p_{ij} = \beta_{ij} \in [0, 1]$ are constants, $\mathbf{S} = [s_1, s_2, \ldots, s_n]$ is a vector and s_i specifies the number of sensors inserted at vertex v_i. The vectors in the standard basis for n-dimensional space comprise the basic elements in $\mathbf{S}$.

Let $\mathbf{S}_k$ denote the first k elements selected by the greedy algorithm and let $\mathbf{S}^*$ denote the actual optimum, $f(\mathbf{S}^*) = OPT$. The greedy algorithm will select exactly

c elements, i.e. $\mathbf{S}_c$ is the vector returned by the algorithm. We claim by induction that the inequation B.1 holds for $0 \leq k \leq c$, where c is the given number of sensors.

$$f(\mathbf{S}^*) - f(\mathbf{S}_k) \leq (1 - \frac{1}{c})^k f(\mathbf{S}^*) \tag{B.1}$$

The base case $k = 0$: $f(\mathbf{S}^*) - f([0, \ldots, 0]) \leq f(\mathbf{S}^*)$ holds.

Suppose the inequation B.1 holds true for the k^{th} step. We now prove it also holds for the $(k+1)^{th}$ step. First, we prove three basic inequations:

$$f(\mathbf{A} \cup \mathbf{B}) \geq f(\mathbf{A}) \quad or \quad f(\mathbf{B}) \tag{B.2}$$

Proof of Inequation B.2:

$$\begin{aligned} & f(\mathbf{A} \cup \mathbf{B}) \\ & \qquad = m - \sum_{j=1}^{m} \prod_{i=1}^{n} p_{ij}^{max\{b_i, a_i\}} \\ & \qquad \geq m - \sum_{j=1}^{m} \prod_{i=1}^{n} p_{ij}^{a_i} \\ & \qquad = f(\mathbf{A}) \quad or \quad f(\mathbf{B}) \end{aligned}$$

$$(\mathbf{A} \cap \mathbf{B}) \leq f(\mathbf{B}) \quad or \quad f(\mathbf{A}) \tag{B.3}$$

Proof of Inequation B.3: Please refer to the proof of inequation B.2.

$$f(\mathbf{A} + \mathbf{B}) \leq f(\mathbf{A}) + f(\mathbf{B}) \tag{B.4}$$

Proof of Inequation B.4:

$$f(\mathbf{A} + \mathbf{B})$$

$$\begin{aligned}
&= m - \sum_{j=1}^{m}\prod_{i=1}^{n} p_{ij}^{a_i+b_i} \\
&= \sum_{j=1}^{m}(1 - \prod_{i=1}^{n} p_{ij}^{a_i} * p_{ij}^{b_i}) \\
&= \sum_{j=1}^{m}(1 - \prod_{i=1}^{n} p_{ij}^{a_i} * p_{ij}^{b_i} + \prod_{i=1}^{n} p_{ij}^{a_i} - \prod_{i=1}^{n} p_{ij}^{a_i}) \\
&= \sum_{j=1}^{m}(1 - \prod_{i=1}^{n} p_{ij}^{a_i} + \prod_{i=1}^{n} p_{ij}^{a_i} * (1 - \prod_{i=1}^{n} p_{ij}^{b_i}) \\
&\leq \sum_{j=1}^{m}(1 - \prod_{i=1}^{n} p_{ij}^{a_i} + 1 - \prod_{i=1}^{n} p_{ij}^{b_i}) \\
&= 2m - \sum_{j=1}^{m}\prod_{i=1}^{n} p_{ij}^{a_i} - \sum_{j=1}^{m}\prod_{i=1}^{n} p_{ij}^{b_i} \\
&= f(\mathbf{A}) + f(\mathbf{B})
\end{aligned}$$

We define the marginal value of element E with respect to $\mathbf{S}$ as $f_{\mathbf{S}}(\mathbf{X}) = f(\mathbf{S} + \mathbf{X}) - f(\mathbf{S})$. Now we can prove inequation B.5:

$$f(\mathbf{A}) - f(\mathbf{B}) \leq \sum_{\mathbf{X}\in\mathbf{A}-\mathbf{B}} f_{\mathbf{B}}(\mathbf{X}) \tag{B.5}$$

$$\begin{aligned}
&f(\mathbf{A}) - f(\mathbf{B}) \\
&\leq f(\mathbf{A} \cup \mathbf{B}) - f(\mathbf{B}) \\
&\leq f(\mathbf{B}) + f(\mathbf{A} - \mathbf{B}) - f(\mathbf{B}) \\
&\leq \sum_{\mathbf{X}\in\mathbf{A}-\mathbf{B}} f_{\mathbf{B}}(\mathbf{X})
\end{aligned}$$

According to the *greedy* Algorithm 7, the element $\mathbf{X}_{k+1}$ selected in the $(k+1)^{th}$ step maximizes $f_{\mathbf{S}_k}(\mathbf{X}_{k+1})$ among the remaining elements, including all the elements in $\mathbf{S}^* - \mathbf{S}_k$. This implies that the element $\mathbf{X}_{k+1}$ has the marginal value:

$$\begin{aligned} f_{\mathbf{S}_k}(\mathbf{X}_{k+1}) &\geq \frac{1}{|\mathbf{S}^* - \mathbf{S}_k|} \sum_{\mathbf{X} \in \mathbf{S}^* - \mathbf{S}_k} f_{\mathbf{S}_k}(\mathbf{X}) \\ &\geq \frac{1}{c}(f(\mathbf{S}^*) - f(\mathbf{S}_k)) \end{aligned}$$

Finally, we can prove inequation B.1 at the $k+1$ step:

$$\begin{aligned} & f(\mathbf{S}^*) - f(\mathbf{S}_{k+1}) \\ & = f(\mathbf{S}^*) - f(\mathbf{S}_k) - f_{\mathbf{S}_k}(\mathbf{X}_{k+1}) \\ & \leq f(\mathbf{S}^*) - f(\mathbf{S}_k) - \frac{1}{c}(f(\mathbf{S}^*) - f(\mathbf{S}_k)) \\ & = (1 - \frac{1}{c})(f(\mathbf{S}^*) - f(\mathbf{S}_k)) \\ & \leq (1 - \frac{1}{c})^{k+1} f(\mathbf{S}^*) \end{aligned}$$

Using the claim for $k = c$, we get

$$f(\mathbf{S}^*) - f(\mathbf{S}_i) \leq (1 - \frac{1}{c})^c f(\mathbf{S}^*) \leq \frac{1}{e} f(\mathbf{S}^*)$$

then we have approximation ratio:

$$r = \frac{f(\mathbf{S}^*)}{f(\mathbf{S}_i)} \leq 1 + \frac{1}{e-1}$$