A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems

Mahima Suresh, Usha Manohar†, Anjana G R†, Radu Stoleru, Mohan Kumar M S†
Department of Computer Science and Engineering, Texas A&M University
†Department of Civil Engineering, Indian Institute of Science

***Abstract*—Water Distribution Systems (WDSs) are prone to events such as leaks, breaks, and chemical contamination. Continuous monitoring of WDSs 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 [9] [18]. Recently mobile sensors for WDS monitoring has attracted research interest to overcome the shortcomings of static sensors [21] [14] [11]. However, most existing solutions are either unrealistic, or focus on on-demand methods (i.e., deploying sensors when presence of an event is suspected). In this paper, we propose a Cyber-Physical system (CPS) – CPWDS, for continuous monitoring of a WDS. Mobile sensors reside 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 prevent sensors from getting stuck and to ensure the sensors cover the main pipes of the WDS. We evaluate the proposed algorithms/protocols for the communication, computation and control of the CPWDS and demonstrate their performance through extensive simulations.**

I. Introduction

Water distribution systems (WDSs) are critical infrastructures of national importance, vulnerable to a variety of attacks that can potentially have severe health and economic impacts [7] [18]. The basic purpose of a WDS is to supply water of required quantity and quality to consumers at a required pressure. Any event that can impact WDS functioning, therefore impacts the population directly. Hence, WDSs need to be monitored continuously using methods that are long term.

The problem of WDS monitoring has been traditionally solved using static sensors placed at strategic locations [9] [18]. Such sensors often fail to cover the entire WDS since they are sparsely distributed and WDSs are, typically, large scale and complex [29] [23]. With recent technological advancements [21], the use of mobile sensor networks for WDS monitoring has received attention from the research community [14] [11]. Existing solutions that use mobile sensors, however, are unrealistic for deployment in WDSs. E.g., [14] requires that sensors follow a predetermined path, which cannot be achieved without disrupting normal functioning of a WDS. [20] studies the use of mobile, and static sensors, where mobile sensors are assumed to follow a specific path. [11] requires RFID tags to be set up throughout the pipelines, which is impractical. A notably important contribution in this area studies optimal monitoring of an acyclic flow-based system [24]. That research focuses on the computation aspects of WDS monitoring, specifically the minimization of number of mobile sensors and beacons deployed to ensure: a) a certain degree of sensing coverage of an area of interest; and b) localization accuracy of an event. The mechanism proposed in [24] is *on-demand*, wherein sensors and beacons are deployed when the water utility company suspects the presence of an event. In [24] it is also assumed that the flows in pipes do not vary over time, and a practical mechanism to recover sensors from WDSs exists. Importantly, none of these proposals are designed for continuous WDS monitoring.

In this paper, we present a Cyber-Physical System for *continuous monitoring* of a WDS, i.e., a Cyber-Physical Water Distribution System (CPWDS).

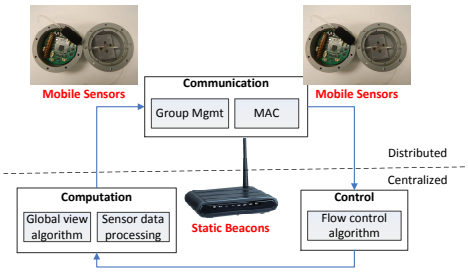

Fig. 1. Proposed Cyber-Physical Water Distribution System (CPWDS)

A representation of the CPWDS we propose is in Fig 1. We envision a CPWDS where *Mobile Sensors* move in the main pipes of the WDS, aided by the flow of water. They exchange sensed data among themselves using underwater acoustic communication and report data to *Static Beacons* placed outside the pipes (Communication component in Fig 1). The beacons predict the paths that sensors will take after communicating with them and transmit this information to the sensors (Computation component in Fig 1). To ensure sensing coverage of the WDS (i.e., the sensors move in the main pipes without getting stuck), a *control system* operates valves and pumps to change the direction of flows in certain pipes (Control component in Fig 1). The CPWDS is therefore described by the three key pieces - Communication, Computation, and Control.

A CPWDS poses several interesting research challenges in communication, computation, and control. The mobile sensors are equipped with acoustic modems to communicate among themselves. Underwater acoustic communication is challenging [25] [12] [15] due to the high propagation delays. A MAC protocol that is able to work in such environments is challenging to design. Using a MAC layer protocol such as T-Lohi [25] (a MAC protocol designed for low range underwater acoustic networks) with a decoupled routing/application layer protocol is ineffective, as we will demonstrate in Section IV. On the other hand, beacons are placed outside the pipes and are

978-1-4799-5041-6/14/$31.00 ©2014 IEEE

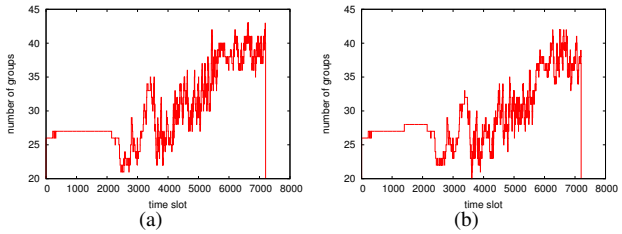

Fig. 2. The number of groups varying over time for two runs.

aware of the entire WDS topology. This information could be very useful to sensors in pipes, unaware of their locations. It is challenging to design an algorithm to be executed by beacons to provide information about the WDS in an efficient way (i.e., avoiding redundant and unnecessary information) to sensors. Owing to the varying demands of the consumers and valve actions, the flows in the pipes of a WDS change in magnitude and direction over time. The main pipes of a WDS are usually of larger diameter than the pipes leading to consumers. We remark here that if sensors are designed to have a large form factor, they move around without getting stuck in small pipes. However, they may move to parts of the WDS where flows do not change directions over time. In order to effectively monitor a WDS with mobile sensors, we need sensors to traverse the main pipes continuously.

The contributions of this paper are:

- It present the design of a Cyber-Physical System for continuous WDS monitoring.
- It presents a MAC/group communication protocol for CPWDS (CPWDS Communication Problem).
- It presents an algorithm that uses global network knowledge to provide information to sensors that aids in communication (CPWDS Computation - Global View - Problem).
- It presents a control mechanism to change WDS flows (CPWDS Flow Control Problem).
- It evaluates proposed solutions through extensive simulations and compares them with state of art.

II. Motivation, State of the Art and Problems Formulations

In this section, we motivate the research, review the state of art, and provide insights on how to tackle the research challenges in all CPWDS components.

A. Communication issues in CPWDS

In a CPWDS, the topology of the communication network for mobile sensors changes over time. Moreover, the movement of sensors is not deterministic. Consider a protocol in which sensors are assigned 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 vary over time), we ran simulations in CPWDSim, a simulator specifically designed for sensors moving in a WDS (details on CPWDSim are given in Section IV). Figures 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 [3]. 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.

MAC protocols for underwater acoustic sensor networks are mainly based on CDMA or TDMA [19]. CDMA is a preferred protocol in mobile sensor networks, because the uncertainty in propagation delay does not affect the performance of the algorithm [28] [8]. TDMA is also efficient in scenarios where delays of a few seconds can be tolerated. Techniques for time synchronization are also prevalent [26].

MAC protocols for underwater acoustic networks is a widely explored area, especially in oceans and seas. However, very few protocols consider in-pipe systems where the network topology is dynamic and has a relatively low communication range ($\sim$10-100m) [25] [12] [4]. T-Lohi [25] 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. Hybrid MAC [12] 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 paper 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 [4] where channel access occurs in a predetermined order. [15] 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.

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 [6] [5]. Distributed algorithms for group management operations, such as leader election are considered for networks with variable message delays and for mobile ad-hoc networks have been studied [27] [22]. An interesting approach in leader election is link reversal [10]. Here, directions are assigned to bidirectional links to represent leadership. These algorithms, however, are effective when the links in the network are known, and there are very few topology changes, since a number of messages need to be exchanged before the algorithm terminates. Several distributed algorithms for self-organization [30], for distributed event detection, localization, and tracking [1] have also been studied in the past. However, these protocols work best in the terrestrial networks with bounded, predictable delays without spatial uncertainty.

In this paper, we combine group management with MAC, thereby determining when channel access is required. Such a mechanism reduces collisions and avoids the overhead of contention-based and TDMA protocols.

B. Computation issues in CPWDS

In a CPWDS, mobile sensors traversing the pipelines are unaware of their position, uncertain about sensors in commu-

nication 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 Fig 3. After a sensor node n_1 comes in 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 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.

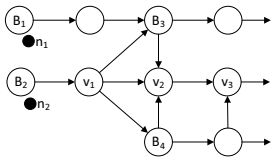

Fig. 3. Example of the data that beacons can provide.

We propose a global view algorithm to predict the movement of sensors. The research challenge here is to account for the time-varying nature of the WDS in making predictions.

C. 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 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 vertices in the network have more than one set of pipelines connected to them, ensuring water supply to all users even with a break or leak in one of the pipelines.

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 [17]. DI controllers are well known methods for nonlinear controls, carrying out feedback linearization on nonlinear 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 [13]. Nonlinear DI controller with PID features was used to achieve the equitable distribution of water in Bangalore water inflow systems [16]. Controllers can also be applied for qualitative control on water networks [7]. In [7], flow control was achieved by valve throttling. Several studies were performed for different controller tunings [13].

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.

D. Formulations of Problems

Based on the above discussion, we broadly define the three problems that are addressed in this paper:

MAC/Group Communication Problem: Design a MAC/Group communication protocol that is suitable for acoustic underwater communication for time varying network topology with nodes moving in groups.

Global View Computation Problem: Design an algorithm that uses predicted flows to compute global information such as potential group splits and group merges to broadcast to sensors.

Flow Control Problem: Control the direction of flow in any pipe that will ensure movement of sensors without disrupting normal functioning of a WDS.

III. A Cyber-Physical System for Continuous Monitoring of Water Distribution Systems

To address the challenges mentioned in the previous section, this paper makes contributions in all the three areas of a CPWDS, namely Communication (solving the MAC/Group Communication Problem), Computation (solving the Global View Computation Problem), and Control (solving the Flow Control Problem). Before presenting our solutions to the three problems, we clarify our assumptions and definitions.

A. 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 [26]. 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 Fig 4, 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 [25]. The slot length Δ is selected to be large enough so that a message

Algorithm 1 Leader node n_i

Input: 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: $timeElapsed$ = 0
8: **if** $BEACON - HELLO$ received **then**
9: Broadcast (RF) $allData$
10: **if** $dataToReport$ received **then**
11: Append $nodeId(dataToReport)$ to ACK
12: **if** ACK is not empty at beginning of time slot **then**
13: Broadcast (Acoustic) ACK
14: Clear ACK
15: **if** $beaconData$ received from Beacon **then**
16: Add $beaconData$ to $HELLO$
17: **if** $HELLO$ received from n_k **then**
18: $mergeDetected$ = true;
19: Relinquish leadership and follow n_k
20: **if** $MERGE$ received from n_k **then**
21: $mergeDetected$ = true;
22: Increment $timeElapsed$

Algorithm 2 Follower node n_j

Input: 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: **if** Contention for leader won **then**
5: $g_j = n_j$; p_j = new prime number
6: Perform leader operations
7: **if** ACK received and contains n_j **then**
8: $report$ = false
9: **if** $HELLO$ received **then**
10: $timeElapsed = 0$
11: **if** Data needs to be sent **then**
12: $report$ = true
13: **if** $HELLO$ contains $beaconData$ **then**
14: Update $beaconData$
15: **if** $report$ is true **then**
16: Broadcast (Acoustic) $allData$ summary
17: Increment $timeElapsed$

from a transmitter can reach all the nodes in its communication range [12]. Since the length of the time slot is on the order of milliseconds, the protocol works well with a low accuracy of time synchronization.

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.

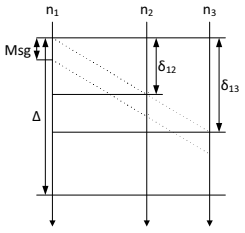

Fig. 4. Spatial uncertainty in the acoustic medium.

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 [2].

We assume that beacons are aware of the network topology, and can predict flow in pipes with high accuracy. This assumption is reasonable, since utility managers collect monthly water usage of consumers. We define $\mathcal{F}$ as the time varying graph of the WDS. At any time instant, $\mathcal{F}$ is a directed acyclic graph.

B. CPWDS Communication

In a WDS, several sensors are inserted at select vertices to add redundancy to ensure sensing coverage. However, sensors spread to different edges in the network owing to the structure of the network and the variations in the flow. Hence, at the time of insertion, the groups are large, then their sizes vary over time, and eventually becomes small and sparse.

Based on the presented state of the art in Section II, we observe that most of the TDMA-based protocols are inflexible to topology changes. On the contrary, contention based protocols, such as CSMA, adapt to changing topologies, but offer no delay or reliability guarantees and suffer from inaccuracies due to the high propagation delay. Hybrid protocols and protocols designed for acoustic environments perform fairly well.

We use the concept of a leader that is responsible for broadcasting messages to beacons and other sensors, and a group that consists of nodes following a particular leader. The invariants desired are: (i) every node is in communication range of at least one leader; (ii) every node follows at most one leader; and (iii) no two leaders are in range of each other. The leaders periodically broadcast their data. If a follower sensed data that is different from the leader (typically happens when sensing range is smaller than communication range), then it reports that data to the leader. It is important to note that if a node traverses the same path as the leader, the sensed data of the node is at an offset from the leader's data, and over sufficient number of periods, the node observes the offset and ceases to report data. The leaders exchange data when they come in range of each other, and all leaders, except one of them, relinquish leadership. Group splits are also considered, when a node ceases to hear the periodic broadcast from its leader. We use a TDMA-like slotted mechanism at the MAC layer. To avoid unnecessary reporting, the leaders transmit a $HELLO$ message periodically. Followers send their data only if it is determined that the data they sensed differs from the leader's data. We make use of the high propagation delay to approximate the distance between the leader and the followers.

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 is as described in Algorithm 1, and the actions taken by a follower is as described in Algorithm 2. 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 1, 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 2, 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 1, lines 12-14). When two groups merge, the leaders of the groups come in range of each other and can hear the $HELLO$ messages from each other. In such an event, the leader that transmitted the first $HELLO$ retains leadership, and the leaders exchange their data in a $MERGE$ message (Algorithm 1, lines 4, 20-21). A $MERGE$ message contains n_i, p_i, and $allData$. 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

Algorithm 3 Beacon B at vertex v_i

```
Input: vertex, F, reportedData
1: while true do
2:     if time % period = true then
3:         Broadcast BEACON − HELLO
4:     if message received from leader then
5:         G(V, E) = F(time)
6:         Q = (v_i, 0)
7:         while Q not empty do
8:             (v, t) ← Q.dequeue()
9:             if In-degree of v > 1 then
10:                Add (v, t) to groupMerges
11:            if Out-degree of v > 1 then
12:                Add (v, t) to groupSplits
13:            if Beacon at v then
14:                Add (v, t) to nextBeacons
15:            for each c ← child of v in G do
16:                t' ← t
17:                e(t') ← (c, v) ∈ E, d_e(t') = length(e)/2
18:                while d_e(t') > 0 or < length(e) do
19:                    t'_prev ← t'
20:                    t' ← time after t' when flows change
21:                    d_e(t') ← distance covered on e(t') from t'_prev to t'
22:                if d_e(t') < 0∧ no beacon at v then
23:                    Q.enqueue(v, t')
24:                if d_e(t') > length(e)∧ no beacon at c then
25:                    Q.enqueue(c, t')
26:        Transmit (groupSplits, groupMerges, nextBeacons)
```

smallest id among all its neighbors, it becomes the new leader (Algorithm 2, lines 2-6).

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. A time slot is long enough so that the node farthest away from the sender receives a message within the time slot (e.g., if the communication range is 10m, the maximum propagation delay of a sound wave is 6.67 ms. Accounting for the time to send a message, the time slot size is set to 10 ms).

C. CPWDS Computation - Global View

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 3 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.

At each vertex visited, if the in-degree is greater than 1, there is a possible group merge; if the out-degree is greater that 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.

D. CPWDS 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: $h_{loss-fp}(t) = \left(\frac{f_p l_p}{2gD_p A_p^2}\right) Q_p^2(t)$. 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: $h_p(N, Q_p) = A_0 N^2 + \frac{B_0}{n} N Q_p - \frac{C_0}{n^2} Q_p^2$. 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 [7]. The PID Controller is generally represented as $u = K_p e + K_i \int e dt + K_d \frac{de}{dt}$, 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 are 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: $X(t) = [Q_1\ Q_2\ \ldots Q_n]$, where n is the number of pipes. The control variables are the valve settings, $U(t) = [u_1\ u_2\ \ldots u_m] = [K_{v_1}\ K_{v_2}\ \ldots K_{v_m}]$, 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 $E = [e_1\ e_2\ \ldots e_n]$, i.e., $= [Q_1 - Q_1^*\ Q_2 - Q_2^*\ \ldots Q_n - Q_n^*]$. 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 $K_{v_1} = K_{p_1} e_1 + K_{i_1} \int e_1 dt + K_{d_1} \frac{de}{dt}$.

IV. Performance Evaluation

We evaluate the performance of our algorithms/protocols in CPWDSim, a simulator we implemented specifically for CPWDS. The WDS map of real cities is a security sensitive issue. Hence, research in WDSs uses Micropolis, a vitrual city model [3] integrated with EPANET (which accounts for hydraulics in WDSs). CPWDSim obtains reports of flows, velocity, and demands at end points from EPANET and simulates the movement of sensors in WDSs. CPWDSim also simulates the communication among sensors and with beacons.

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. 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 [25].

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.

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 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.

A. 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 [25] 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 Fig 5. 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 Fig 6, 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.

B. 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

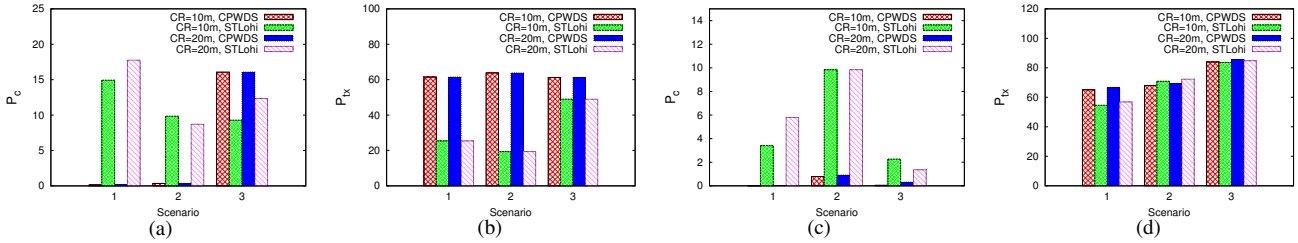

Fig. 5. Impact of CR, Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

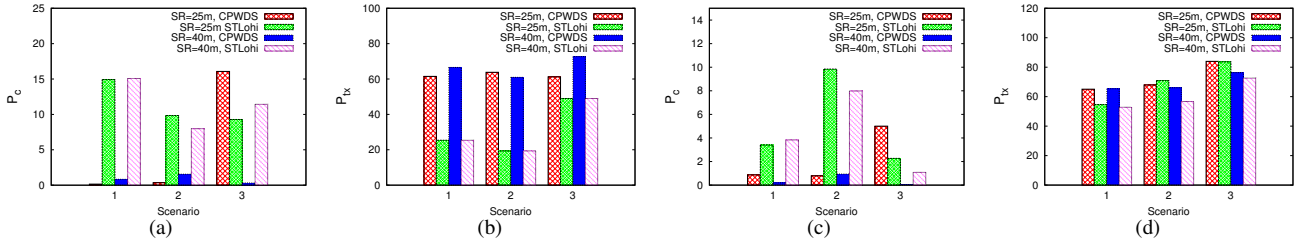

Fig. 6. Impact of SR, Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

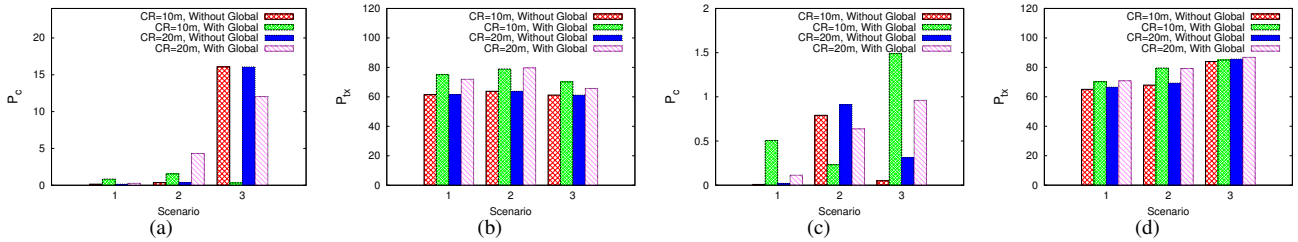

Fig. 7. Impact of CPWDS Computation (i.e., Global View information), CR, and Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

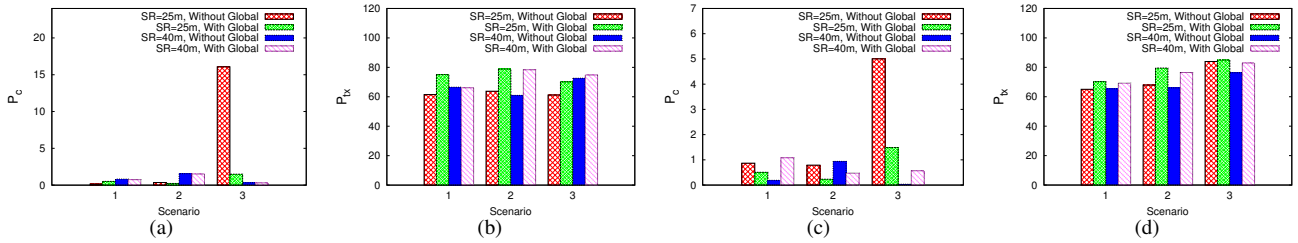

Fig. 8. Impact of CPWDS Computation (i.e., Global View information), SR, and Scenario on P_c and P_{tx} during Morning (a)-(b) and Afternoon (c)-(d).

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 3 consistently helps increase P_{tx}, although it does not always decrease P_c as shown in Figures 7 and 8. In Fig 7, SR = 10m. Fig 8 has CR = 10m. 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 Fig 8, Scenario 3, SR = 25m 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}.

C. Performance of CPWDS Flow Control

For evaluating our control component, we used a modified version of Micropolis WDS with two valves added, as shown in Fig 9. 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.

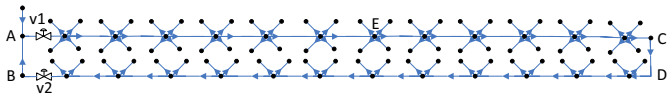

Fig. 9. Part of the Micropolis virtual city used to demonstrate flow reversal

As presented in Section III-D, 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.

Fig 10 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 III-D. We observe that valve setting for $v1$ converged to $\sim$66 and $v2$ converged to $\sim$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

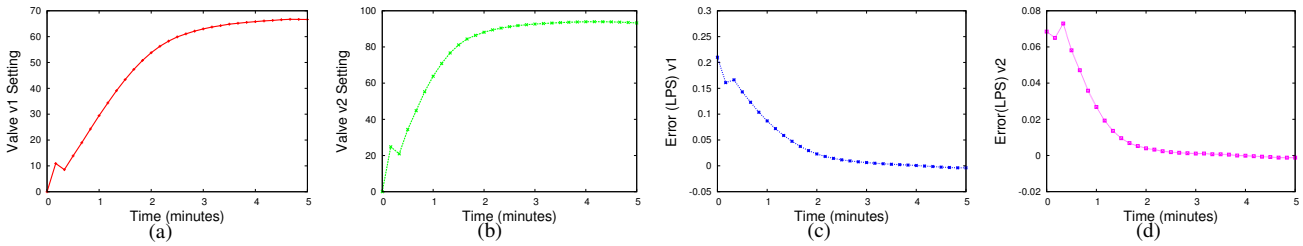

Fig. 10. Valve settings for (a) $v1$, (b) $v2$, and error in valve setting for (c) $v1$, (d) $v2$ over time

faster with $v2$ than with $v1$. We observe that in both cases, the error approaches zero within 5 minutes (convergence time $\sim$ 5 minutes), indicating that the controller action is satisfactory.

V. Conclusions

This paper presents the first design of a CPWDS, consisting of mobile sensors that move through the pipes aided by the flow of water, static beacons that are placed outside the pipes and collect data from mobile sensors, and a control system to change the direction of flows in the main pipes of the WDS. We present: i) a MAC/group communication protocol for communication among sensors; ii) a global view algorithm executed by beacons; iii) a control system designed to enable flow reversal in the pipes. We implement the algorithms in our simulator, CPWDSim and demonstrate improvement over state of the art. We also demonstrate flow reversal in several pipes of a WDS. We leave for future work the design of algorithms that optimize cost, and implementation on hardware [21] of proposed algorithms and protocols.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No: 1145858, 1253968, 1127449 and in part by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

References

[1] T. Abdelzaher et al. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *ICDCS*, 2004.

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

[3] K. Brumbelow, J. Torres, S. Guikema, E. Bristow, and L. Kanta. Virtual cities for water distribution and infrastructure system research. In *WEWRC*, 2007.

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

[5] R. de Araujo MacEdo, A. Freitas, and A. de Sa. A self-manageable group communication protocol for partially synchronous distributed systems. In *LADC*, 2011.

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

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

[8] G. Fan, H. Chen, L. Xie, and K. Wang. An improved CDMA-based MAC protocol for underwater acoustic wireless sensor networks. In *WiCOM*, 2011.

[9] W. Hart and R. Murray. Review of sensor placement strategies for contamination warning systems in drinking water distribution systems. *Journal of Water Resources Planning and Management*, 2010.

[10] R. Ingram, P. Shields, J. Walter, and J. Welch. An asynchronous leader election algorithm for dynamic networks. In *IPDPS*, 2009.

[11] J. Kim, G. Sharma, N. Boudriga, and S. Iyengar. SPAMMS: a sensor-based pipeline autonomous monitoring and maintenance system. In *COMSNETS*, 2010.

[12] K. B. Kredo, II and P. Mohapatra. A hybrid medium access control protocol for underwater wireless networks. In *WuWNet*, 2007.

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

[14] T. Lai, W. Chen, K. Li, P. Huang, and H. Chu. TriopusNet: automating wireless sensor network deployment and replacement in pipeline monitoring. In *IPSN*, 2012.

[15] J. Ma and W. Lou. Interference-aware spatio-temporal link scheduling for long delay underwater sensor networks. In *SECON*, 2011.

[16] U. Manohar and M. M. Kumar. Modeling equitable distribution of water: Dynamic inversion based controller approach. *Journal of Water Resources Planning and Management*, 2013.

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

[18] 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*, 2006.

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

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

[21] L. Perelman, W. Salim, R. Rouxi Wu, J. Park, A. Ostfeld, K. Banks, and D. Porterfield. Enhancing water distribution system security through water quality mobile sensor operation. In *WEWRC*, 2013.

[22] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. Self-stabilizing leader election for single-hop wireless networks despite jamming. In *MobiHoc*, 2011.

[23] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline. PIPENET: A wireless sensor network for pipeline monitoring. In *IPSN*, 2007.

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

[25] A. Syed, W. Ye, and J. Heidemann. T-lohi: A new class of MAC protocols for underwater acoustic sensor networks. In *INFOCOM*, 2008.

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

[27] S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *ICNP*, 2004.

[28] M. K. Watfa, S. Selman, and H. Denkilkian. UW-MAC: An underwater sensor network MAC protocol. *International Journal of Communication Systems*, 2010.

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

[30] J. Zhang, K. Premaratne, and P. Bauer. A distributed self-organization algorithm for ad-hoc sensor networks. In *WCNC*, 2003.