WISDOM: Watering Intelligently at Scale with Distributed Optimization and Modeling

Daniel A. Winkler

University of California, Merced
dwinkler2@andres.ucmerced.edu

Alberto E. Cerpa

University of California, Merced
acerpa@andres.ucmerced.edu

ABSTRACT

As lawn irrigation is estimated to consume 7 billion gallons of scarce fresh water each day in North America alone, lawn irrigation systems are a high priority for improvements in efficiency. To this end, recent work has introduced several key advancements in irrigation control. Distributed actuation systems allow the irrigation system to apply water completely independently across the field allowing flexibility of control, and the use of fluid flow modeling and optimization allows more efficient schedules to be computed automatically, significantly improving the irrigation quality of service as well. However, the proposed systems are designed with centralized architectures that introduce single points of failure, computational bottlenecks in data processing, and significant network energy for data forwarding used by the centralized data-driven modeling strategies. In response to these challenges, we propose and demonstrate WISDOM, whose novel and flexible hardware and processing pipelines enable the use of a *distributed system* for the management of irrigation systems of any scale, with energy independence by way of energy harvesting. Across 4 weeks of live system deployment, we find that the WISDOM system can save up to 32.9% of water in comparison to industry-best, while maintaining a perfect quality of service to the plant. Furthermore, with substantial analysis in simulation we find that in addition to practical system improvements, the use of the proposed distributed system within typical operating conditions will provide *all* of the efficiency and quality-of-service benefits of the globally-modeled, centrally controlled systems, while allowing the robust control of irrigation systems of any size.

ACM Reference Format:

Daniel A. Winkler and Alberto E. Cerpa. 2019. WISDOM: Watering Intelligently at Scale with Distributed Optimization and Modeling. In *The 17th ACM Conference on Embedded Networked Sensor Systems (SenSys '19)*, November 10–13, 2019, New York, NY, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3356250.3360023>

1 INTRODUCTION

The United States Environmental Protection Agency (EPA) estimates that 30% of all fresh water consumed in the residential domain goes to the irrigation of residential landscaping [14], and the

United States Geological Survey estimates that lawn irrigation systems consume 7 billion gallons of fresh water each day in North America alone [33]. A land survey performed using aerial imaging by the National Aeronautics and Space Administration (NASA) concludes that lawns represent the largest irrigated crop by surface area, covering approximately 128,000km² of land in North America [11]. At the same time, it is estimated that only about 1% of Earth's surface water is fresh AND available for our use [4]. With residential lawn irrigation restrictions becoming more common [12], and the University of California's goals to reduce overall water consumption by 20% by 2020 and 36% by 2025 [19], it is clear that there is increasing political, environmental, and economic motivation to reduce the impact of such a large consumer.

Although we wish to minimize the water consumption of lawn irrigation systems, the responsibility of these systems is to apply the correct amount of moisture to the area. Soil moisture levels that are too low will cause the plant to wilt and die, and levels that are too high can cause the plant roots to rot, soil erosion, and in extreme cases transport fertilizer chemicals into nearby drinking water supplies, as has occurred in California's Salinas Valley [27]. To prevent these issues, previous work has introduced systems for distributed actuation and model-based schedule optimization [40, 41], allowing fluid to be placed efficiently where it is required. Although they were found to substantially improve the quality of service provided to the plant and the irrigation system efficiency, the proposed systems require a centralized controller to process all data and make control decisions. This introduces significant computational bottlenecks in optimization due to the growth of the global models used at scale, a single point of failure that can cripple the system, potentially causing massive water waste as experienced in [40], and increases the overall energy consumption of the sensor network as all data must be forwarded across the network for processing. With the additional requirement of periodic battery changes in these systems, it is unlikely these systems could be adopted for control of irrigation systems at scale.

Recognizing that nearly all of the aforementioned system limitations are due to centralized architectures, we develop WISDOM, a *distributed system* for the control of truly large-scale irrigation systems whose primary intellectual contributions are as follows:

(1) Evaluating the mechanics of water movement in the space, we modify existing modeling and optimization techniques for use in *localized neighborhoods*, and introduce a processing pipeline for *distributed* irrigation decision making on the edge devices themselves. By operating within local neighborhoods, we remove the point of failure, computational bottleneck, and energy penalties caused by centralized control, making the system truly scalable.

(2) As no existing system architecture provides the low-power operation of the traditional embedded device *and* the processing

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SenSys '19, November 10–13, 2019, New York, NY, USA

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6950-3/19/11...\$15.00

<https://doi.org/10.1145/3356250.3360023>

capabilities required for model training and optimization on the edge devices, we provide a low-power embedded device with a relatively high-performance co-processor to meet these requirements. With power duty-cycling by the low-power “master” device, the co-processor *vastly* improves storage, memory, and processing power, while incurring a relatively small energy penalty during operation. WISDOM is the first system to utilize a co-processor architecture for distributed computation in data-driven modeling and control.

(3) Finally, recognizing that our devices are collocated with sprinklers throughout the space, we take advantage of the available fluid energy flowing through them with small water turbines¹ installed underground that allow us to harvest energy for battery recharge during irrigation. We show that even with the additional energy requirements of our co-processor, the energy harvesting capabilities allow the WISDOM system to operate indefinitely with true energy independence. In the wireless sensor network domain, WISDOM is the first to utilize sprinkler water source for energy harvesting.

Across 4 weeks of deployment and extensive analysis in simulation, we demonstrate that the WISDOM system is capable of achieving *all* of the quality of service and efficiency benefits of existing systems while maintaining true energy independence while controlling irrigation systems at *any* scale.

2 RELATED WORK

The best irrigation controllers available in the industry make use of sensed weather events to make irrigation decisions. The simplest of these controllers are standard timer-based control equipped with a rain sensor [9], which disables irrigation when rain is sensed. The more advanced systems use the Evapotranspiration metric [21, 29], which takes recent measurements of temperature, humidity, wind, and solar exposure to estimate the amount of water that has been lost to the environment. The controller, which retrieves an ET loss estimate from a local weather station, will then re-apply this amount of lost water to the surface of the lawn space. These systems have no way to take advantage of distributed actuation as they have no understanding of local conditions.

More recently, irrigation controllers have been developed that react to the output of one or more soil moisture sensors installed in the irrigated space to decide valve scheduling [20, 31]. These systems are designed for use with a centralized valve architecture, and are not configured to take advantage of distributed actuation.

Standard irrigation system architectures require a single water valve to control the flow of water to many sprinkler heads located throughout the space. However, in practice the environmental conditions such as soil type and depth, solar exposure, surface slope, and others are heterogeneous across an irrigated space. Under this simple centralized actuation, to maintain healthy levels of moisture everywhere, the system is required to irrigate the whole area for as long as the area that requires the most water. To address this fundamental limitation of all turf irrigation systems, [41] developed a distributed wireless sensor/actuator device, which allows for each sprinkler head to be actuated independently to allow for more flexibility of control. A PDE model of moisture movement was developed and used in optimization to compute valve schedules that minimize system water consumption while maintaining

a satisfactory level of moisture to maintain plant health across the space. Schedules produced were found to improve system efficiency and quality of irrigation service, but slow system performance, manual model configuration, and lack of model correction over time made the system impractical for use at large scale.

To address the practical limitations of [41], [40] introduces a *data-driven* modeling strategy to be used for irrigation schedule optimization. Trained on soil moisture data collected by the distributed sensor/actuator nodes, the use of a data-driven strategy offers several significant advantages. Data driven models do not require manual configuration and will retrain automatically over time to learn changes to the environment, will be more accurate than a mechanistic model based on fixed assumptions of spatial homogeneity, and do not require heavy computational resources allowing the system to control larger irrigation systems. The proposed system further improved both system efficiency and quality of service in addition to the aforementioned practical improvements. Although [40] is capable of controlling larger irrigation systems than [41], scalability of control is limited due to the centralized nature of the control strategy. The data driven model used in optimization maintains *global* information about each sprinkler's effects on soil moisture across the *entire* irrigation system, which is computationally wasteful and limits the scale, as optimization processing time becomes the bottleneck. In addition, this architecture becomes a single point of failure in the system as discussed in [40], where failure of the communication channel has led to data loss and substantial unplanned over-irrigation. Finally, the system still relies on periodic battery changes in long-term operation, which has the potential to affect system adoptability due to the recurring costs and manpower required to perform this maintenance at scale.

As the devices used in wireless sensor networks are generally incapable of performing heavy computation, several approaches have been used to overcome these limitations. Firstly, it has become common for distributed applications to be configured for use on mobile [26, 30, 34] or embedded [10] devices with on the order of GBs of memory and storage, but they are both very energy-hungry and expensive at the scale of our application. Another approach is to reduce the required memory and computation by simplifying the models used, for instance in deep neural nets [24, 42, 43]. However, in cases like ours where mobile phones are too expensive and power hungry and the problem can't be adequately simplified, the use of a co-processor comes to mind. Used in off-the-shelf devices like the iPhone [1] and Samsung Galaxy [5], a co-processor can handle background tasks such as motion sensor querying with low power, or highly specific tasks such as neural net computation for image processing and facial recognition without burdening the primary processor on the device. While co-processor architectures have been used in the WSN domain for chip reprogramming [28] and security-specific applications [32, 37, 39], no standard co-processor architecture has been developed allowing for low-power operation AND computational capabilities for model training and optimization as required in our application, so we design our own.

3 SYSTEM OVERVIEW

WISDOM is a *distributed control system* for lawn irrigation at scale, made up of a wireless network of sensor/actuator devices that sit

¹We use COTS water turbines, we did not develop new water turbine technology.

Figure 1: Comparison of PICS (left) and WISDOM (right) system architectures

below each sprinkler in the irrigation system. Each device is managed by a low-power master device in the form of an SAM R21 [13] SoC, which can perform simple computational tasks, speak wirelessly with sister nodes in the space with its 802.15.4 compliant radio, and maintain device peripherals. These peripherals include a soil moisture sensor to monitor the local soil moisture status near the device, a solenoid allowing the master device to control the flow of water to the sprinkler independently of the rest of the network, and an onboard co-processor (Raspberry Pi Zero [16]), providing significant storage and computational capabilities directly on the edge devices. With the inclusion of the co-processor, the end devices have the ability to make intelligent decisions for irrigation *without* the need for cross-network data communication for centralized decision-making. To ameliorate energy costs associated with co-processor operation, each device is equipped with its own energy harvesting capability using a water turbine that activates when the attached sprinkler is activated, recharging the batteries. In sum, each device in the system has the ability to monitor local conditions, communicate with the network to make sprinkler actuation decisions, and put those actuations into effect.

For a plant to remain healthy, three conditions must be met. There must be adequate solar exposure, the soil must have the correct nutrients in appropriate amounts, and sufficient water in the soil must be available near the roots for uptake. Although an irrigation system has no control over solar exposure and soil nutrient availability, its primary purpose is to distribute water where it is required to maintain adequate moisture levels at all locations in the space. In plant physiology literature, the level of soil moisture below which the plant can no longer extract fluid through its roots is known as the Permanent Wilting Point (θ_{pwp}), and as such serves as the lower bound of acceptable moisture levels in our irrigation system. While more conservative moisture thresholds exist such as the crop fraction for no stress [21] that may be more appropriate for system installation on decorative turfgrass, we choose the more restrictive θ_{pwp} as it serves as the most restrictive and difficult goal for the irrigation system, and thus helps demonstrate the WISDOM system’s ability to meet even the most strict requirements. For different turf plants, the θ_{pwp} remains similar so our experiments are valid for a wide range of turf irrigation applications. Although the θ_{pwp} threshold may be constant across the space, previous work [40, 41] has found across several irrigation system deployments that uniform water application will not result in uniform soil water levels, due to differences in the way water will move in different areas. To learn these local variations, we

learn models of water movement over time, trained by data traces collected at each location, to be used to compute optimal irrigation customized to the space.

As shown in Figure 1a, each device in the PICS [40] system transmits its data to a centralized controller for processing. The controller trains a *global* water movement model, which is then used to optimize valve schedules. These schedules are then re-distributed to the devices for actuation. With a large irrigation system, the global water movement model makes schedule optimization time too slow as later discussed in Section 7, increasing network traffic causes forwarding nodes to consume more energy, and any lapse in communication with the controller can cripple the system.

Although we could break a very large centralized system into smaller centralized systems to help overcome the computational limitation, such a solution does nothing to improve the other fundamental limitations, single point of failure and cross-network data communication. For these reasons, we design the WISDOM system to be *fully distributed* as shown in Figure 1b to overcome all of these issues. As the WISDOM system runs, its installed devices periodically measure and log the state of moisture in the space, as well as the on/off status of the solenoids. Each device shares this information with all devices in its *local neighborhood*, which in practice is defined to include all nearby devices that are within radio range. Over time, using these collected data traces, each device learns *localized models* that describe the effect of actuation by all nodes within the neighborhood on the soil moisture distribution across the neighborhood. Likewise each device learns how its own local soil moisture will degrade over time after irrigation has ended, in a two-model technique inspired by [40]. These models, trained by the co-processor, are then used by the co-processor in optimization to find schedules that minimize system water consumption while maintaining adequate moisture levels everywhere in the system, our two primary goals. By performing these tasks in a distributed way, we enable a system that can robustly scale to control an irrigation system of any size, without sacrificing system efficiency or quality of service, as later discussed in Sections 6 and 7.

To maintain a long system lifetime even during seasons when energy harvesting is unavailable, we must be cautious with our energy budget to prevent unnecessarily burning stored power with our co-processor when it isn’t required. Each day, all master devices in the network wake up and distribute all buffered data to the surrounding nodes within the local neighborhood. Once all data is

Figure 2: Minimum local neighborhood definition (square dotted box) for modeling and control

received, the master device will use the current moisture distribution in the neighborhood to make a decision whether or not irrigation is necessary on this day, returning to low power if it is not necessary. If irrigation is likely to be required, the co-processor is activated and all recently-collected data from the neighborhood is transmitted by wire to the co-processor to be integrated into the data-driven models, and an on-board optimization problem is initiated that computes valve scheduling within the local neighborhood to guide the moisture levels in the soil towards adequate levels. Adequate levels are defined as the daily *Goal State* of moisture as defined in [40], the soil moisture level to be achieved by the end of irrigation that considers the minimum moisture level θ_{pwp} we wish to maintain at all times, the local losses we have learned to expect from historical data, and the forecasted weather. As internet access is required to retrieve weather forecasts, weather prediction is the *single* parameter that must be provided by a central source.

$$\text{Goal State} = \theta_{\text{pwp}} + \text{Expected Losses} + ET_{\text{forecasted}} \quad (1)$$

Once optimization is complete, the schedules are sent from the co-processor to the low-power master device to be actuated by the attached solenoid. As irrigation continues, this optimization may be performed several times using the freshest data from the neighborhood to correct for any deviation due to error in the data-driven models used. When the daily operations are complete, the master devices deactivates the co-processor and returns the device to a low-power state until the next scheduled wakeup.

4 SYSTEM DESIGN

4.1 System Modeling

In an irrigation system, each sprinkler has a relatively limited area of fluid influence on the soil. Standard sprinkler installations use sprinklers that reach 75%-100% of the distance to the next adjacent sprinkler in the grid, to ensure that water can also reach the diagonal region between the elements of the grid installation. Once the fluid is distributed onto the surface of the soil, the fluid will flow through the grass layer due to diffusion or runoff effects, if the system is installed on sloped land. The fluid will flow until it is absorbed by the soil, after which it will move very slowly, on the order of $10^{-2} - 10^{-3}$ cm/s [22]. Due to fluid's very slow movement through the space, the global modeling proposed by [40] is computationally very wasteful, as it tracks each sprinkler's effect on the soil moisture at *every* location throughout the space. In a very large system, a sprinkler will have no effect on soil moisture at a *vast* majority of locations, so modeling these interactions are very wasteful and causes optimization to take more time. To improve on this technique and allow distributed operation of the system, each device in the space instead maintains information of only its surrounding

Variable	Description
t	Temporal index $\in \{0, \dots, T\}$
N_K	Number of nodes in local neighborhood
$\mathbf{s}_t$	Vector of moisture levels at time t , size N_K
$\mathbf{f}_t$	Vector of sprinkler actuation at time t , size N_K

Table 1: Model Variables

Variable	Description
t	Temporal index $\in \{0, \dots, T\}$
k	Sprinkler location index $\in \{1, \dots, N_K\}$
$\mathbf{s}_t$	Vector of moisture levels at time t , size N_K
$\mathbf{f}_t$	Vector of binary sprinkler actuation at time t , size N_K
$f_{k,t}$	Sprinkler k actuation at time t , $\in \{0, 1\}$
$s_{k,t}$	Volumetric water content (VWC) of location k at time t
c_k	Flow rate of sprinkler k (Constant, known beforehand)
θ_k	Measured VWC of sensor k (Constant, known beforehand)

Table 2: Optimization Variables

neighborhood of embedded devices. In practice this neighborhood includes all devices within radio range; as radio range can be on the order of 100m and sprinklers are at most 30m apart, each node will maintain information about its second or even third neighbor, with a neighborhood of tens of devices. However, in this work we posit that even a minimum neighborhood including only directly adjacent neighbors will provide sufficient information for the distributed system to have equivalent performance to the centralized system using global modeling, while making the system truly scalable and improving robustness of operation. To test this hypothesis, the minimum neighborhood containing only direct neighbors as shown as the square box in Figure 2 is used in both our live deployment and evaluation in simulation.

To train the fluid movement model later used to optimize schedules, we collect a training dataset in which each sprinkler is actuated one at a time. As the sprinkler runs, all devices in the neighborhood record their sensor data along with the neighborhood actuations. This data is then shared across all N_K devices in the neighborhood and used to train a linear regressor that takes as input current moisture levels at all N_K locations as vector $\mathbf{s}_t$ and current sprinkler actuations at all N_K locations as vector $\mathbf{f}_t$, with values as defined in Table 1. This regressor then provides as output the predicted moisture levels at all N_K locations at Δt seconds in the future as vector $\mathbf{s}_{t+\Delta t}$, where Δt is the control timestep, 60 seconds in our system. The linear function $\mathbf{g}$ is defined as follows:

$$\mathbf{g}(\mathbf{s}_t, \mathbf{f}_t) = \mathbf{s}_{t+\Delta t} \quad (2)$$

The model, trained on local historical data within the node's local neighborhood, can then be used to predict future moisture changes as a function of sprinkler actuation.

4.2 Optimization Over the Schedule

Each day during irrigation time, each active irrigation device in the system will use its locally-gathered information to find the best sprinkler scheduling to satisfy the needs of the space; in using local models only in a distributed fashion, we are solving relatively small optimization problems in parallel making the system truly scalable. Using our model of neighborhood fluid movement during irrigation as described in Section 4.1 and the goal state of moisture defined in Section 3, we set up an optimization problem that

will minimize water consumption in the system subject to quality of service constraints. We define the binary sprinkler actuation as $f_{k,t}$ for each sprinkler in local neighborhood $k \in \{1, \dots, N_K\}$ and time index $t \in \{0, \dots, T\}$, soil moisture values as $s_{k,t}$ for each sprinkler in local neighborhood $k \in \{1, \dots, N_K\}$ and time index $t \in \{0, \dots, T\}$, and other optimization variables as shown in Table 2. The optimization problem is defined as follows:

$$\min_{\{f_{k,t}, s_{k,t}\}_{k=1, t=0}^{N_K, T}} \sum_{k=1}^{N_K} \sum_{t=0}^T c_k f_{k,t} \quad \text{s.t.} \quad (3a)$$

$$0 \leq f_{k,t} \leq 1 \quad k = 1, \dots, N_K \quad t = 0, \dots, T \quad (3b)$$

$$s_{k,t} \geq \theta_{\text{pwp}} \quad k = 1, \dots, N_K \quad t = 0, \dots, T-1 \quad (3c)$$

$$s_{k,T} \geq \theta_{\text{goal},k} \quad k = 1, \dots, N_K \quad (3d)$$

$$s_{k,t=0} = \theta_k \quad k = 1, \dots, N_K \quad (3e)$$

$$s_t = \mathbf{g}(s_{t-1}, \mathbf{f}_{t-1}) \quad t = 1, \dots, T \quad (3f)$$

As we wish to minimize the amount of water consumed by the irrigation system, the objective function is defined by the sum of all $f_{k,t}$, weighted by the water flow rate of each sprinkler c_k , which represents the total water consumed under schedule $\mathbf{F}$. The quality of service is constrained in two discrete time horizons; fluid levels *during* irrigation are constrained to remain above satisfactory levels, but fluid levels are held in sufficient levels *between* irrigation cycles by constraining that each device reach its *Goal State* of moisture by the end of irrigation, which offsets for predicted losses to maintain adequate soil moisture at all times. The soil moisture state at time $t = 0$ is defined to be the most recently measured state of moisture by the WISDOM system sensors in the neighborhood, defined as θ_k , and future moisture levels will evolve with respect to sprinkler actuations as defined by our linear function trained in Section 4.1. As the sprinkler actuation values $f_{k,t}$ are binary in the physical irrigation system, the defined problem is an Integer Linear Program (ILP), making the problem NP-Complete. Solving this original problem can take as long as several minutes, which can interfere with our control cycle, as we may want to optimize schedules as often as our control timestep, defined to be 1 minute. To instead find an approximate solution, we allow the $f_{k,t}$ values to be real within $[0,1]$ and then round the resulting actuations to be binary, a simplification that has been found to have minimal effect on the resulting schedules [40] while making the optimization a Linear Program (LP), which is much easier to solve.

As written, this LP has $2N_K(T+1)$ variables and $4N_K(T+1)$ constraints. Although solving this optimization results in schedules for each device within our local neighborhood, the device will only extract the schedule optimized for itself to be actuated by the attached solenoid. By considering the initial conditions and solving for optimal scheduling on our neighbors, we ensure our local schedules take into account the likely actions of our neighbors to prepare for or take advantage of any shared conditions such as sprinkler overlap or runoff effects that affect neighbors. We solve this problem using the Julia programming language [23] to interface with the GNU Linear Programming Kit (GLPK) [6] solver, and later discuss the performance and scalability in Section 7.

4.3 Node design

In order to handle the computation required of the modeling and optimization of Sections 4.1 and 4.2, we need significant improvements in system performance. The simplest LP solvers such as Simplex [35] have a storage complexity for the problem matrix that is $O(mn)$, where m is the number of constraints and n is the number of optimization variables. Assuming a small neighborhood $N_K = 9$ and 2 hours of irrigation $T = 120$, even our small LP introduced in Section 4.2 requires a matrix with 9M entries to store the problem, not feasible on embedded devices with 32-48KB of system memory, let alone space to perform matrix operations. Training our linear model from Section 4.1 using ordinary least squares requires the entire dataset be able to fit into system memory, and although gradient descent has significantly less memory requirements, the data points must still be stored and retrieved, and both techniques require significant processing power. In addition to system memory limitations, the standard embedded devices run at relatively slow processor speeds on the order of 48MHz and typically do not have a hardware floating point unit, requiring floating point to be approximated using integer math which is much slower. Finally, these devices have very little support for standard libraries, likely requiring us to port standard libraries ourselves. Rather than trying to reinvent the wheel, we decide to export these tasks to a separate co-processor that is capable of handling these tasks with ease.

To make this happen, we design our own system management chip to meet the requirements of our application. The low-power devices designed for use in embedded systems are unbeatable in their very low power consumption for long system lifetime on a limited energy budget. For this reason, we use the SAM R21 [13] SoC, which combines the powerful cortex m0+ microcontroller with the ATRF233 radio, as our system's low-power master device. While insufficient to perform our heavy computation, its 802.15.4 wireless capabilities and basic computational performance makes it perfect for performing regular background tasks such as data syncing and peripheral management while maintaining a low power profile. Its 256KB of onboard flash storage allows it to buffer data in the short-term until it can be transferred to the co-processor for processing. The master device is wired to our latching solenoid via an H-Bridge, which allows us to send of pulse with positive and negative voltage to open or close the valve according to any schedule we choose. The master device is also wired directly to an attached Decagon EC-5 sensor [3], allowing the device to monitor the local soil moisture conditions. Finally, the device is equipped with a header that allows it to be directly connected to the co-processor, a Raspberry Pi Zero [17], a \$5 device that provides the flexibility of storage and computation that far exceeds the low-power master device. The Zero has a single core 1GHz Arm processor, 512MB of system memory, and microsd external storage. Furthermore, the Zero runs a flavor of Debian Linux with significant support for languages, platforms, and standard tools. The co-processor and master device are connected via a bi-directional UART communication channel, allowing data and commands to be sent between the two devices as required. Finally, the master device has direct control over the power source to the co-processor, allowing the master device to completely disable the co-processor's power to prevent unnecessary power draw when the co-processor is not required.

Use of our co-processor significantly increases peak power consumption. To put its requirements into perspective, analysis later performed in Section 4.5 finds that our co-processor consumes an average of 165mA power to complete its daily tasks, a draw about 12× higher than our master device with the radio in a transmitting state at full power (13.8mA), its most energy-hungry activity. To ameliorate this increased power draw, we recognize that as each device is collocated with a periodically-active water source, there is substantial fluid energy that can be harvested. In researching potential harvesting equipment, we found that small water turbines that are capable of providing a steady 5 volt output at up to 2 amps are available off the shelf. Furthermore, we found that charge circuits such as the MCP73831[2] are readily available that can charge a LiPo battery at a rate up to 500mA. With the use of a 1Ah LiPo battery, this means we can fully charge the battery from empty with 2 hours of irrigation. While 2 hours of irrigation is not typical in a single day of irrigation, we will later show in Section 4.5 that by being cautious with our energy budget, this harvested energy is more than sufficient to keep our system operating indefinitely.

4.4 Daily system operation

Each day, the low-power master devices in each WISDOM node will wake up at a time scheduled by the installer or system administrator, likely to be either early morning or evening. Once awake, the following actions are performed:

(1) At wake-time, each master device goes into a low-power listening state [36] while the neighborhood wakes up. During this time, the basestation will use the DRIP [38] dissemination protocol to distribute the single value that requires internet from a centralized source, future weather prediction. This value is disseminated during network wake-up, giving it plenty of time to cross the largest networks without disrupting the processing pipeline.

(2) Once a device receives a status ping from all listed neighbors indicating the neighborhood is awake, it will turn the radio into an all-on state and broadcast all data single-hop from the previous day to their neighbors. In its current configuration, the WISDOM system samples its sensor at a maximum granularity of 1 minute during irrigation, as this data is used to perform re-training of the data-driven models, and reduces the sampling rate to 10 minutes for the rest of the day. Considering that the maximum irrigation period is set as 3 hours, the maximum period allowed with our campus water pressure requirements and regulation, each device will have at most 306 data samples stored as 16-bit unsigned integers, or 612 bytes each day. We additionally package the node's locally-actuated schedules for model training; 3 hours of irrigation has at most 180 binary actuations at a minute granularity, which can be encoded 8-per-byte in just 23 bytes. With a radio packet payload of 100 bytes, supported by our hardware and software platforms, this daily data is sent to all neighbors in at most 7 data packets.

(3) Once all data is received by the master device on each node, it will be buffered in onboard flash storage. The master device will sample the most current long-term model and use the weather forecast we have received to compute the *Goal State* of moisture at its location, the amount of moisture we must reach by the end of irrigation to maintain adequate moisture across the full 24-hour cycle. If the most recent moisture level measured by the sensor tells us that our current moisture is already higher than our *Goal State* plus

a safety margin that we define to be 5% volumetric water content to account for any water losses that may occur *during* irrigation, then the master device decides that irrigation is not required today, and will put the device back into sleep mode, to wake up on the following day. The ability for the master device to terminate the processing pipeline is crucial for long system lifetime during the winter months, as there will be no energy harvesting to recuperate the energy losses incurred by the co-processor.

(4) If the master device decides that irrigation is required, the co-processor will be powered on by flipping a transistor activating the co-processor's power source. Once the co-processor boots, a wired serial communication channel allows the master device to transfer all buffered soil moisture data and actuations from the neighborhood and weather forecasts to the co-processor. Automatic processes running on the co-processor receive these new data streams to re-build the long-term and short-term models as discussed in Section 4.1, and then the models and current data state are used to optimize schedules for all nodes in the neighborhood. In this way, we will use all knowledge from our local neighborhood to find optimal valve scheduling for *our* device, while respecting the likely actions of our neighbors as well. The optimized valve schedule for our device is then communicated back to the master device along with the updated long-term model for use in Step 3 above, and the schedule is applied using the attached solenoid. If further irrigation is required today, the co-processor will remain active to perform any additional optimization, as later discussed in Section 4.5.

To use as little energy consumption as possible, our co-processor has a discrete set of tasks to perform to minimize on-time. These functions are enumerated in Figure 3 along with their effect on the co-processor's power consumption. This current data is measured by tracking the voltage drop across a high accuracy shunt resistor with an oscilloscope, and post-processing with a moving average filter to reduce signal noise. Once power is provided by the master device, the co-processor boots its "Raspbian" [18] operating system as shown in the figure as (a). Once the operating system finishes its boot cycle, a local Julia HTTP server is launched, which will handle all requests for schedule optimization until the co-processor is powered down. The Julia initialization period is shown in the figure at (b), taking approximately 25 seconds. During this initialization time, all buffered data is transferred from the master device to the co-processor via a wired connection and used to retrain the data driven models, operations that are complete in seconds and do not seem to impact power consumption considerably.

Once Julia initialization is complete, a python process is launched that gathers the freshest models and soil moisture readings from the neighborhood and connects to the Julia server to initialize schedule optimization. Figure 3 shows that schedule optimization, enumerated with (c), takes just over a minute to complete. When finished, the optimized schedules are transferred back to the master device to be actuated by the attached solenoid. At point (d) when the optimization has completed, the co-processor will do one of two things; if the optimized schedules show that no further irrigation is required on this day, the co-processor will perform a graceful shutdown and the master device will cut off its power entirely to return to low-power operation, and schedule wakeup for the following day. If further irrigation is required and optimization is to be repeated, the co-processor will return to an idle power level as

Figure 3: Node energy profile across (a) boot, (b) Julia initialization, (c) first optimization solve, (d) completion and return to idle, and (e) subsequent optimization

shown in the figure, and the Julia server will remain running to be re-used, amortizing its startup costs. In our experience, although the initialization of the Julia server and first optimization take on the order of 100 seconds, subsequent optimization problems are solved in a fraction of a second. The power profile of subsequent optimization can be seen as a small blip in energy at (e) in the figure, 60 seconds after the first problem is solved.

If the optimized schedule activates the attached sprinkler, the water turbine will immediately begin charging the LiPo battery to recuperate the energy consumed in steps (a)-(d), along with all communication, sensing, and actuation costs incurred by the master device since the last battery recharge. As it is difficult to monitor energy harvested by a turbine in our live system deployment, we perform a table-top test of the turbine’s power output, attached to a standard hose spigot providing approximately 40psi of water pressure. The power output of the turbine, nearly 700mA at 5v, is then plotted as the green dashed line in Figure 3. As our battery charge circuit [2] features a maximum charge rate of 500mA, the thin blue line in the figure shows the effective system charge rate. Although any power harvested above this rate is not used, the charge rate far surpasses the consumption of our device, even at peak power load. In a long irrigation cycle, the power consumption across repeated optimizations will be similar to that shown between points (d) and (e) consuming around 100mA for the majority of on-time, allowing a rapid battery recharge during irrigation.

4.5 Co-processor energy profile

In terms of energy, the worst-case scenario is for soil moisture values at irrigation time to be just above the *goal state* so that optimization will still occur to be cautious, but the optimized schedules do not require the attached sprinkler to be activated. This is possible if the small amount of water required can be received by an adjacent sprinkler, or if the learned daily losses are small enough, such that no irrigation is required to stay above the goal state. In this case, the energy costs to start the co-processor and optimize schedules will not be recovered as water will not be flowing through the turbine. The energy that would be unrecovered in this worst-case is shaded in blue in Figure 3, which we compute to be an average current draw of 165mA at 5v for 110 seconds, a total power draw of 25.2mWh per day. As our 1Ah, 3.7v LiPo battery has a capacity of 3700mWh, our system can handle this worst-case scenario *each day* for nearly 5 straight months before the battery will be drained. In practice, this occurrence would be next to impossible, as even a very minor change in weather (temperature, humidity,

or precipitation) can slightly raise soil moisture adequately such that the master device can safely skip co-processor activation for optimization, or slightly lower soil moisture such that irrigation is required, allowing the battery to be recharged. To guarantee energy neutrality, WISDOM is configured to run a short irrigation cycle (10 minutes) to force a battery recharge if the battery is detected to be critically low. This would come at the expense of some water waste, but such a condition is expected to never occur.

5 CASE STUDIES

5.1 Live Deployment

To test the WISDOM control strategy against existing systems, we launch two side-by-side irrigation systems on our campus, covering a total of just under 10,000 ft², purpose-built for irrigation system analysis. As closely identical as possible, the two irrigation systems each cover an area of 60’ by 60’ along a gently-sloping hillside that drops approximately 3 feet from the highest to lowest end, and are each installed with a 3x3 grid of Hunter Pro-Spray [8] sprinklers, providing irrigation coverage to the area. By constructing these systems to be as identical as possible, it allows us to compare two irrigation control strategies simultaneously, such that they are both experiencing weather conditions, runoff effects, sprinkler coverage, and soil conditions as similar as possible to allow a true side-by-side comparison of their operation. In this way, we compare the WISDOM system to the state-of-the-art evapotranspiration control strategy for 2 weeks, and to the PICS control strategy for 2 weeks. The experiments were done on a shoulder season, with both rainy and sunny days to cover a wide range of weather conditions.

As two systems run side-by-side, we monitor the soil moisture levels across the space, as well as the schedules that are being run by the actuators. Using this information, we compute and compare several metrics that indicate how well the two systems are performing relative to one another. The first primary metric is the quality of service provided to the plant, as indicated by the amount of time the sensed locations spend in unhealthy levels of soil moisture. The second is the amount of water consumed by the irrigation system, which can be easily computed by considering the sprinkler flow rates as noted by the manufacturer and the amount of on-time each sprinkler spends, as indicated by the schedules that are run. Finally, a secondary metric we consider is the spatial uniformity of moisture across the irrigation system; a highly uniform moisture coverage will prevent the occurrence of “hotspots”, where local areas of high or low moisture can cause discoloration of the plant or the growth of fungus in the soil. By minimizing water consumption, minimizing exposure to inadequate moisture levels in the soil, and maximizing the spatial uniformity of soil moisture, we will be providing the best irrigation possible.

5.2 Simulation

Although it is best to compare irrigation control strategies side-by-side in a live deployment, our test deployment gives no indication of how well a control strategy will perform on edge-case systems, such as those located on steep hillsides, with unusual sprinkler system architectures, or at very large scale. Due to the configuration of our local models to maintain information about the adjacent devices, any environment or irrigation system architecture in which water is able to move *beyond* the reach of this

Figure 4: Example 14-day simulation of WISDOM control

local neighborhood will be unknown to our system, making these edge-case systems more challenging for the WISDOM system to control. In such systems, WISDOM may find schedules that are less efficient as those produced using *all* information contained in a global model, such as in the PICS system. Specifically, to confirm that the localized techniques of the WISDOM system do not break down due to these effects, we wish to compare WISDOM to the globally-modeled PICS system in conditions with challenging soil conditions, steep surface slope, and sprinkler installations with unusual amounts of overlap, as these have the potential of causing water to move beyond a device's local neighborhood in the WISDOM system.

To allow us the flexibility to test the effect of these changing environmental and architectural features in an irrigated space, we develop an hybrid water movement model using the nonlinear PDE model developed in MAGIC [41] to model water movement during irrigation actuation, and the long-term loss models developed in PICS [40] to model fluid loss between irrigation cycles. In doing so, we can track water movement during irrigation subject to surface runoff, sprinkler overlap, diffusion, and infiltration into the soil at any desired level of spatial and temporal resolution. Then, once water enters the soil where it moves very slowly, we model fluid flow as spatially-independent loss models, represented by exponential decay functions. These loss functions are randomly generated at each measuring location throughout the space to be similar in magnitude to those experienced in reality, and then random noise is added to losses on each day of simulation, providing losses that are noisy but consistent at each location, similar to what we expect in reality. This hybrid simulator allows us to construct sample irrigation systems from scratch under a wide variety of conditions and closely monitor how the water is moving through it with respect schedules chosen by any control strategy. Not only does a simulator's use allow us to compare WISDOM's operation to another system across many environments, but it also allows us to quantify the benefits of the WISDOM system at significantly larger scale than our test irrigation system allows. A sample simulation of the WISDOM system across a 14-day period can be seen in Figure 4, where the control strategy can be seen to react to initially dry soil conditions across the first 4 days to surpass a minimum level of moisture at all locations in the space.

In our side-by-side comparison with PICS in simulation, we consider three primary metrics. The first is the water consumption of the irrigation system, as this will be directly related to the optimized models used by the two control strategies, and the second is the quality of service provided, as computed by the amount of time our sensed points remain below θ_{pwp} , in unhealthy levels of

moisture. If the local models contain sufficient information about the environment, we expect these metrics to be identical between the WISDOM and the PICS systems, with any deviation indicating that the local models of WISDOM lack information present in the global model of PICS. The third metric of comparison is the time required by the two systems to optimize their schedules as the scale of the irrigation system is increased.

6 EXPERIMENTAL RESULTS

To test the operation of the WISDOM system, we perform two side-by-side system deployments for two weeks each against the Evapotranspiration and PICS control strategies. Although it is not necessary for the operation of the WISDOM system at scale, to monitor the system we had all experimental devices forward a copy of its operational data (soil moisture and schedules) to a centralized location for visualization and analysis. The soil moisture data collected during the first deployment comparing Evapotranspiration and WISDOM systems can be seen in Figure 5, and the soil moisture data collected during the second deployment comparing PICS and WISDOM systems can be seen in Figure 6. Also shown in these figures is the horizontal line indicating the soil moisture level θ_{pwp} , which acts as the lower limit the system should allow at all times. As previous results presented by [40] saw the data-driven modeling technique occasionally drop below the threshold due to model error, in this system we included a 5% safety band on the value of θ_{pwp} ; in other words, although we want to maintain a 20% volumetric water content (VWC) as shown, both data-driven systems are actually trying to maintain a 21% VWC due to this safety band.

Each day on the ET controlled system, we can see that irrigation causes a sharp increase in soil moisture at all measuring locations as the irrigated fluid reaches the system. On both the WISDOM and PICS systems, which are utilizing sensed water distribution to optimize schedules, we can see this spike in moisture levels only on the sensor values that are reaching near our minimum allowed moisture level, as the data-driven systems feel that without irrigation moisture levels will drop into dangerously low levels. Interestingly, in the WISDOM and PICS systems, locations that are well above this level and require no irrigation still see a steady rise and fall in sensor values each day. This is an effect of the metric measured by our EC-5 sensors [3]; the soil moisture rise is due to increasing soil temperature, which has influence on the dielectric permittivity [25] which is being measured. With more expensive sensors that also measure temperature, these deviations could be corrected for, but as our installed sensors do not have this feature, we experience a daily soil moisture cycle even when no irrigation is applied. Another interesting artifact was the occurrence of rain on day 1 of the second deployment, leading to a great increase in soil moisture levels. It can be seen that one region within the WISDOM system retained a very large amount of water, with water levels staying high for the remainder of the experiment preventing the requirement of irrigation in that area for the entire deployment. Between days 4 and 7 in the first deployment we can see one sensor exhibiting strange behavior. An animal had chewed a sensor cable, and on these days irrigation was entering the damaged area and causing strange sensor values until day 7, when the problem was diagnosed and corrected.

Figure 5: Collected VWC data across deployment for all Evapotranspiration (left) and WISDOM (right) nodes

Figure 6: Collected VWC data across deployment for all PICS (left) and WISDOM (right) nodes

The primary goal of an irrigation system is to maintain healthy levels of moisture everywhere, and previous work [40, 41] has defined the quality of service metric as the amount of time the system spends in unhealthy levels of moisture. In our experiments, shown in Figures 5 and 6, it is clear that all systems tested maintained a healthy level of moisture (above our minimum level θ_{pwp}) at all times. However, there is a clear tradeoff between quality of service provided and system efficiency. While the evapotranspiration control strategy was able to maintain healthy levels at all time, partially due to the very high initial moisture conditions shown on day 1, we can see in Figure 7 that the ET system consumed a very high amount of water in comparison to the WISDOM system for the entire experiment. Whereas the ET system responds only to recent weather conditions, the WISDOM system also has knowledge of the changing soil moisture levels, irrigating only when more water is required. For this reason, we can see over the data in Figure 5 that the WISDOM system is able to avoid irrigating areas with sufficient moisture already, saving an average of 61.4% of water. Looking at the last 4 days of the first deployment, as all of the WISDOM devices enter the moisture region where irrigation is required, we find that the system is saving an average of 32.9%, which is more indicative of the potential savings of WISDOM in steady state.

As shown in Figure 6, the WISDOM system has considerably higher soil moisture values after the rain on the first day, preventing one device from requiring irrigation for the entire deployment. Figure 8 shows that in the first several days the PICS system consumes considerably more water than WISDOM, with WISDOM saving an average of 46% of water across the two weeks of deployment. However, we can see that as the deployment progresses and WISDOM nodes drop into the range requiring irrigation, the consumption of the two systems become more and more similar, until

the last 7 days of deployment find an average savings of 19.7%. Although the WISDOM systems found savings in this deployment, we are confident that this is a result of the initial conditions of the experiment, with WISDOM nodes having higher initial moisture levels, and expect that a longer deployment would find these two systems approaching equivalent performance in efficiency once all the devices fall into a steady state. Their equivalence is expected as they are both maintaining data-driven modeling techniques, and the sprinkler system architecture and environment do not facilitate enough water movement to make the local models a limiting factor, as later verified in simulation in Section 7.

Finally, as a metric of moisture uniformity, we plot the spatial variance of moisture in all systems as a function of time in Figures 10 and 11. Even though the ET controller is applying water uniformly onto the surface, differences in environment cause the water to settle in a non-uniform way, as seen by a large spread of moisture levels in Figure 5 and high levels of variance in Figure 10. In comparison, as the data-driven PICS and WISDOM systems learn the needs of the space over time, the variance can be seen to quickly decrease to very low values across the system deployment, indicating that the system is learning the needs of the space and distributing water exactly as required.

6.1 Local model disagreement

When local optimization occurs on each device, it computes the optimal valve scheduling for itself *and* all other devices within its local neighborhood. On the boundary of the local neighborhood, however, the local moisture movement information may be missing boundary effects that will influence moisture levels at nodes on the edge of our neighborhood, resulting in a disagreement between the edge node's schedule we've computed and the schedule that edge node computed for itself. To determine the magnitude

Figure 7: Water consumption of ET vs WISDOM

Figure 8: Water consumption of PICS vs WISDOM

Figure 9: Scope of local models within system topology

of this error due to edge effects, we compare the optimized schedule of each node *for itself* to the optimized schedule by the *node's neighbors* for the node, and collect the average error made by the neighbors. If the error is very high, we will know that schedules are highly dependent on the influence of neighbors' irrigation in our test irrigation system, as this could indicate that localized models will introduce significant error in the schedules computed locally. On each day, we compare the on-time of the optimized schedule as computed on a node to those computed by the node's neighbors, and report the average error in sprinkler on-time for each node's neighbors in Figure 12. In this plot, the bar for node 5, for instance, is the average error in sprinkler on-time of all of node 5's neighbors in estimating node 5's schedule.

We see in the figure that at worst, the average schedule deviation is less than 5 seconds, indicating that the localized models make very good estimates of neighboring actions. Interestingly, the resulting errors agree with intuition. The neighborhoods of nodes 2, 5, and 4 as shown in Figure 9 will contain the entire neighborhood of node 1 shown as the red dotted box, so their predictions will have zero error, and similar for all corner nodes 3, 7, and 9. Neighbors of edge nodes 2, 4, 6, and 8 will be missing at most two nodes; for instance nodes 1 and 2 will be missing node 4's neighbors 7 and 8, so these edge nodes will have some error. Central node 5 should have the most error; if node 1 estimates node 5's schedule, it is missing nodes 3, 6, 9, 7, and 8, resulting in high error.

Figure 10: Moving variance of ET vs WISDOM

Figure 11: Moving variance of PICS vs WISDOM

Figure 12: Average schedule error caused by local models

These expectations match the results of Figure 12 with the exception of node 8, which shows zero error in its neighbors' predictions. The cause for this is the fact that node 8 is the node shown in Figure 6 that remains well above irrigation levels. For this reason, regardless of neighbor influence, the optimal schedule is to disable the sprinkler, which all neighbors correctly predict.

7 RESULTS IN SIMULATION

To challenge the local models used in the WISDOM system, we use our simulator introduced in Section 5.2 in 4 environmental and architectural scenarios. Each experiment tracked water movement across a space of 65m × 65m, just over 1 acre, irrigated by a 7 × 7 grid of sprinklers, and run for a continuous period of 100 days. The first was a typical environment across flat land with a typical "loam" soil commonly found in standard irrigated spaces, with each sprinkler radius 75% of the inter-sprinkler distance. The second introduces a soil with a higher clay content, which reduces the water infiltration rate into the soil, increasing the amount of fluid diffusion that occurs on the surface. The third uses the same soil type as the second experiment, but sits on a hillside with a 15° angle, which will lead to a higher amount of runoff, particularly with the lower infiltration rate of the clay soil. A 15° slope was chosen, as it is the maximum slope supported by most ride-on

Figure 13: Resulting system consumption and quality of service penalty for 4 environmental setups: (1) Flat land (2) Flat land with clay soil (3) 15° slope with clay soil (4) Installed sprinklers reaching beyond *second* adjacent neighbor

lawnmowers, making it the steepest slope likely on a large expanse of lawn installation. Finally, to ensure an example where the fluid influence of a sprinkler reaches beyond the neighborhood of the WISDOM local models, we configure a system architecture similar to the first experiment, but with each sprinkler reaching 2.5× the inter-sprinkler distance, which can be visualized as the arching circle centered on the leftmost node in Figure 2. We *strongly* note that standard irrigation system installation practices use sprinklers that reach no more than 1× the inter-sprinkler distance, and that this is purely an exercise to find the breaking point of the localized modeling techniques; such a system would be prohibitively costly, and would likely consume significantly more water than the source could provide. The results of these four independent experiments are shown in Figure 13, with the positive y axis showing the overall system consumption across the period, and the negative y axis showing the total amount of time spent below our desired moisture threshold at our sensing points in the space.

We find that across the 100 days of simulation for the first experiment (bar 1 in the figure), both control strategies consume 520,000 gallons of water, spending a total of 103,000 minutes beneath the desired level of moisture. Averaged across the 49 nodes in the system, each node spends 35 hours beneath this level, but we can see in Figure 4 that this time stems from the first days after system installation, when each node is learning its loss models. In the second experiment (bar 2), we find that the installation of soil with a higher clay content increases the system consumption by 18% across the experiment. This increased consumption is caused by a higher loss to the boundaries of the space, as the fluid will sit on the surface for a longer period where it has the tendency to leave the space more quickly than if it were moving through the soil. The third experiment (bar 3) found an increased water consumption of 23% in comparison to experiment 1, and 7% increase in exposure to unhealthy levels of moisture. The additional consumption is intuitive, as the slope will cause an additional amount of water to flow out of the space on the boundaries over time, exacerbated by the clay soil preventing infiltration of the fluid into the soil. The increased penalty to quality of service is caused during the first days of experimentation, when the uphill region in the space will lie in unhealthy levels of moisture during the first few days to a greater extent, due to the slope carrying fluid away.

Finally, the fourth experiment shows a difference in result between the PICS and WISDOM systems with a sprinkler layout that

Figure 14: Time required to optimize schedules using centralized (PICS) and distributed (WISDOM) systems

reaches beyond the neighborhood of the local modeling technique. This sprinkler coverage allows the system to apply a significant amount of water on the first days to substantially reduce the quality of service penalty that would otherwise occur, while this increased coverage effectively doubles the water usage in comparison to the system in experiment 1. However, we see that the schedules optimized with the global model are able to find schedules that are roughly twice as good as the locally-computed ones with respect to the quality of service metric, but also led to an increase in system water consumption of 9%. The takeaway of these simulated results is that the control systems using global and local models tend to be identical due to the limited rate of water movement in these irrigated spaces. While it is possible for schedules optimized using local models to find inferior solutions to schedules optimized with global ones, it will not occur in even very challenging scenarios (experiments 1-3), but require unrealistic irrigation system architectures to occur (experiment 4).

In addition, to quantify the performance of the PICS and WISDOM systems side-by-side, we use our simulator to train local and global models on irrigation systems of increasing sizes, and optimize schedules from a fixed initial moisture state to a fixed goal moisture state, both interpolated to the size of the irrigated space. The goal is to make the optimization problems as identical as possible, with the only variable being the size of the irrigation system. The resulting optimization time required of the two systems as a function of system size can be seen in Figure 14. The WISDOM system is solving K optimization problems, each using a model of a relatively small local neighborhood, in a distributed way. While this makes the computation required of the WISDOM system linear on K , it is solved in parallel across K devices, making the optimization solve time constant at all system scales. In contrast, the PICS system is solving a centralized optimization problem using a single global model that grows quadratically on K , as shown in the figure. We find that with a system with more than 145 sprinklers, the centralized control strategy is unable to solve the optimization within our control timestep. In practice, as irrigation systems found in the largest sports fields can contain anywhere from 500-5000 sprinklers [7], the scalability of the centralized system is inadequate. While we could increase the control timestep to allow a larger optimization problem to be solved, the quadratic growth of the optimization solve-time will quickly overcome this time cap, and an increased control timestep would cause less efficient schedules to be found by making the temporal granularity of the model

Component	WISDOM	PICS [40]
Mote (includes M & A costs for WISDOM)	\$25.00	\$37.57
Raspberry pi zero	\$5.00	–
Battery and charge circuit	\$21.00	\$4.00
Harvesting turbine	\$12.00	–
Sealed enclosure	\$13.00	\$10.00
Moisture sensor	\$99.00	\$110.00
Latching solenoid	\$14.00	\$15.00
Manufacture and Assembly (only for PICS)	–	\$10.00
	\$189.00	\$186.57

Table 3: Sprinkler Node Manufacture Cost Comparison

more coarse. Regardless, no such sacrifice is required with the WISDOM distributed system, allowing it to operate at any scale.

8 RETURN ON INVESTMENT

To ensure system adoptability, we must be sure it is affordable as a system replacement. We consider the unit cost of each device as shown in Table 3 and compare them with the prototype cost proposed in [40]. We list unit prices of our components but estimate manufacture and assembly at the scale of 1000 units based on a quote from PCBWay [15]. We note that even with the addition of the co-processor and energy harvesting, removing unnecessary hardware through custom design increases the unit cost by just 1.3% in comparison to the prototype proposed in [40]. On our campus, water for irrigation is billed at a price of \$5.60 per 1000 gallons of water. Using the water savings from the last 4 days of our first deployment against the ET control strategy, we extrapolate a 32.9% water savings onto the larger-scale irrigation systems used to irrigate our sports fields at our university, which are equipped with Hunter I-40 rotor sprinkler heads. Figure 15 shows the evolution of the investment over time, and we find that based on water savings alone, the WISDOM devices will pay for themselves in approximately 5-7 months, considering market variation of 10%. We do note, however, that previous work [40] found that optimized scheduling using data-driven modeling saved a more conservative 12%, due to seasonal and environmental differences in the experiments run. Even in such an environment where these more conservative savings occur, our system is still estimated to return its investment in 15-18 months.

9 DISCUSSION AND FUTURE WORK

Distributed solutions are more resilient than centralized ones due to the lack of a single point of failure and performance bottleneck. There are multiple aspects of resilience, including (a) controller, (b) computation, and (c) communication resilience. Of those, the most important in our opinion is (a). In WISDOM each node calculates its schedule. A failure of node only affects its local area. In contrast, in a centralized system like PICS [40], a failure of the centralized controller affects the entire irrigation system. Even if the controller works fine, sometimes failure of the controller communication channel can lead to substantial data loss and unplanned over-irrigation, as it happened in [40]. The failures experienced occur when environmental conditions (humidity/temperature) disrupt the wired connection between the controller and the RF transmitter, causing the channel to fail for hours at a time; in our experience *collocated redundant centralized base stations* experience *correlated failures* (motivating this work).

Figure 15: Device return on investment over time

In WISDOM, every node calculates its schedule using only information from its neighbors. This reduces computational complexity and increases the resilience of the optimization process as we scale since each node uses only local information to perform the optimization, which does not increase with the size of the field, as opposed to PICS [40]. Finally, in WISDOM, all communication during the day (except for the weather forecast once a day) is between neighbors, so it is single-hop only. In PICS, all data collected at the sensing devices must travel across the multi-network to a centralized irrigation controller, and the resulting schedules must travel to the end devices. This cross-network communication has the potential to significantly increase both routing errors and energy consumption of the network. Detailed analysis and experiments exploring communication and computational resilience are left for future work.

10 CONCLUSIONS

As the largest irrigated crop in North America by surface area, lawns are one of the heaviest consumers of freshwater, motivating improvement in irrigation system efficiency. Although previous work have realized substantial efficiency improvements with the use of distributed actuation and model-based schedule optimization, the proposed centralized processing architectures introduce single points of failure, computational bottlenecks in data processing, and increase network energy usage for cross-network data forwarding, which all prevent the technologies from being adoptable for control at large scale. In this work we introduce WISDOM, a *distributed system* for control of irrigation systems at any scale that utilizes energy harvesting to allow a perpetual system lifetime. Across 4 weeks of live system deployment, we find that up to 32.9% water savings is possible in comparison to industry-standard controllers, and demonstrate through extensive comparison in simulation that the proposed distributed system can provide *all* of the efficiency and quality of service benefits of the centralized one, while allowing energy independence and the robust control of irrigation systems of *any* size.

11 ACKNOWLEDGEMENTS

We would like to thank our anonymous referees and our shepherd for their feedback and assistance. This material is based upon work partially supported by the National Science Foundation under grants #1254192 and #1430351 and the Center of Information Technology in the Interest of Society (CITRIS).

REFERENCES

- [1] [n. d.]. A12 Bionic. <https://www.apple.com/iphone-xs/a12-bionic/>.
- [2] [n. d.]. Adjustable LiPo Charger. <https://www.sparkfun.com/products/14380>.
- [3] [n. d.]. Decagon Devices. <http://www.decagon.com/products/soils/>.
- [4] [n. d.]. Freshwater Crisis. <http://environment.nationalgeographic.com/environment/freshwater/freshwater-crisis/>.
- [5] [n. d.]. Galaxy S10 Intelligence. <https://www.samsung.com/global/galaxy/galaxy-s10/intelligence/>.
- [6] [n. d.]. GNU Linear Programming Kit. <https://www.gnu.org/software/glpk/>.
- [7] [n. d.]. Golf Course Irrigation Systems - Water Efficiency. http://www.liaquifercommission.com/images/Golf_Course_Irrigation_Systems_march_april_2015.pdf.
- [8] [n. d.]. Hunter Pro-spray sprinklers. <https://www.hunterindustries.com/irrigation-product/spray-bodies/pro-sprayr>.
- [9] [n. d.]. Hunter Rain-Click Rain Detection. [hunterindustries.com/irrigation-product/sensors/rain-clickr](https://www.hunterindustries.com/irrigation-product/sensors/rain-clickr).
- [10] [n. d.]. Intel Edison Development Platform. <https://software.intel.com/en-us/iot/hardware/discontinued>.
- [11] [n. d.]. Looking for Lawns. <http://earthobservatory.nasa.gov/Features/Lawn/printall.php>.
- [12] [n. d.]. Mercury News - California Drought Restrictions. <https://www.mercurynews.com/2018/02/13/californias-drought-restrictions-on-wasteful-water-habits-could-be-coming-back-this-time-theyll-be-permanent/>.
- [13] [n. d.]. Microcontrollers and Processors - SAM R21. <https://www.microchip.com/wwwproducts/en/ATSAMR21G18A>.
- [14] [n. d.]. Outdoor Water Use Fact Sheet. <https://www.epa.gov/sites/production/files/2017-03/documents/ws-factsheet-outdoor-water-use-in-the-us.pdf>.
- [15] [n. d.]. PCBWay Manufacturing and Assembly. <https://www.pcbway.com/>.
- [16] [n. d.]. Raspberry Pi. <https://www.raspberrypi.org/>.
- [17] [n. d.]. Raspberry Pi Zero. <https://www.raspberrypi.org/products/raspberry-pi-zero/>.
- [18] [n. d.]. Raspbian Operating System. <https://www.raspbian.org/>.
- [19] [n. d.]. UC Office of the President - Sustainability. <https://www.ucop.edu/sustainability/policy-areas/water/index.html>.
- [20] [n. d.]. UGMO Irrigation. <http://www.ugmo.com/>.
- [21] Richard G Allen, Luis S Pereira, Dirk Raes, Martin Smith, et al. 1998. Crop evapotranspiration-Guidelines for computing crop water requirements-FAO Irrigation and drainage paper 56. *FAO, Rome* 300, 9 (1998), D05109.
- [22] J. Bear. 1972. *Dynamics of Fluids in Porous Media*. Dover. <https://books.google.com/books?id=XEaxd3hGzoC>
- [23] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2014. Julia: A Fresh Approach to Numerical Computing. *CoRR* (2014). <http://arxiv.org/abs/1411.1607>
- [24] Sourav Bhattacharya and Nicholas D. Lane. 2016. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys '16)*. ACM, New York, NY, USA, 176–189. <https://doi.org/10.1145/2994551.2994564>
- [25] André Chanzy, Jean-Claude Gaudu, and Olivier Marloie. 2012. Correcting the temperature influence on soil capacitance sensors using diurnal temperature and water content cycles. *Sensors* 12, 7 (2012), 9773–9790.
- [26] Petko Georgiev, Nicholas D Lane, Kiran K Rachuri, and Cecilia Mascolo. 2014. Dsp. ear: Leveraging co-processor support for continuous audio sensing on smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 295–309.
- [27] Thomas Harter and Jay R. Lund. 2012. *Addressing Nitrate in California's Drinking Water*. Technical Report. University of California, Davis.
- [28] Jason L Hill and David E Culler. 2002. Mica: A wireless platform for deeply embedded networks. *IEEE micro* 22, 6 (2002), 12–24.
- [29] Marvin Eli Jensen, Robert D Burman, and Rick G Allen. 1990. Evapotranspiration and irrigation water requirements. ASCE.
- [30] Z. Jia, M. Alaziz, X. Chi, R. E. Howard, Y. Zhang, P. Zhang, W. Trappe, A. Sivasubramanian, and N. An. 2016. HB-Phone: A Bed-Mounted Geophone-Based Heartbeat Monitoring System. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 1–12. <https://doi.org/10.1109/IPSNS.2016.7460676>
- [31] Yunseop Kim, R. G. Evans, and W. M. Iversen. 2008. Remote Sensing and Control of an Irrigation System Using a Distributed Wireless Sensor Network. *IEEE Transactions on Instrumentation and Measurement* 57, 7 (July 2008), 1379–1387. <https://doi.org/10.1109/TIM.2008.917198>
- [32] Jongdeog Lee, Krasimira Kapitanova, and Sang H Son. 2010. The price of security in wireless sensor networks. *Computer Networks* 54, 17 (2010), 2967–2978.
- [33] M. A. Maupin, J. F. Kenny, S. S. Hutson, J. K. Lovelace, N. L. Barber, and K. S. Linsey. 2014. *Estimated use of water in the United States in 2010*. U.S. Geological Survey Circular 1405. 56 pages. <https://doi.org/10.3133/cir1405>
- [34] M. Moazzami, D. E. Phillips, R. Tan, and G. Xing. 2017. ORBIT: A Platform for Smartphone-Based Data-Intensive Sensing Applications. *IEEE Transactions on Mobile Computing* 16, 3 (March 2017), 801–815. <https://doi.org/10.1109/TMC.2016.2573825>
- [35] John A Nelder and Roger Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.
- [36] Joseph Polastre, Jason Hill, and David Culler. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 95–107.
- [37] Jorge Portilla, Andrés Otero, Eduardo de la Torre, Teresa Riesgo, Oliver Stecklina, Steffen Peter, and Peter Langendörfer. 2010. Adaptable security in wireless sensor networks by using reconfigurable ECC hardware coprocessors. *International Journal of Distributed Sensor Networks* 6, 1 (2010), 740823.
- [38] Gilman Tolle and David Culler. 2005. Design of an application-cooperative management system for wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceedings of the Second European Workshop on*. IEEE, 121–132.
- [39] Yi Wang, Hailong Shi, and Li Cui. 2013. EasiSec: a SoC security coprocessor based on fingerprint-based key management for WSN. *International Journal of Sensor Networks* 13, 2 (2013), 85–93.
- [40] Daniel A Winkler, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. 2018. Plug-and-play irrigation control at scale. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE Press, 1–12.
- [41] D. A. Winkler, R. Wang, F. Blanchette, M. Carreira-Perpinan, and A. E. Cerpa. 2016. MAGIC: Model-Based Actuation for Ground Irrigation Control. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. <https://doi.org/10.1109/IPSNS.2016.7460680>
- [42] Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher. 2018. FastDeepIoT: Towards Understanding and Optimizing Neural Network Execution Time on Mobile and Embedded Devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems (SenSys '18)*. ACM, New York, NY, USA, 278–291. <https://doi.org/10.1145/3274783.3274840>
- [43] Shuochao Yao, Yiran Zhao, Aston Zhang, Lu Su, and Tarek Abdelzaher. 2017. DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys '17)*. ACM, New York, NY, USA, Article 4, 14 pages. <https://doi.org/10.1145/3131672.3131675>