Performance analysis of a user-centric crowd-sensing water quality assessment system

Nikolaos Rapousis and Maria Papadopouli
Department of Computer Science, University of Crete
Institute of Computer Science, Foundation for Research and Technology-Hellas
Email: {rapousis, mgp}@ics.forth.gr

***Abstract*—The water is an important commodity for the existence and development of human society. The placing, maintenance, and distribution of water, drinking or not, is laborious. The current physical sampling-based monitoring with periodic collection has been proven ineffective. The need of real-time monitoring motivated the development of the contamination warning systems (CWSs). Various approaches of CWSs that employ sensor technology have been developed to monitor water quality. By placing the humans in the loop, the assessment of the water quality and the provision of feedback can be enhanced. To evaluate this hypothesis, we developed the QoWater, a novel user-centric crowd-sourcing system that relies on measurements collected from sensors and user feedback to detect changes in the water quality. This system enables smartphone users to evaluate the quality of water based on its chemical properties, taste, odor, and color. It also informs users for potential contamination events, that have been detected close to their location. This paper analyses the performance of the QoWater system, in terms of power consumption, response delay, scalability and shows that the power consumption and response delay are relatively low. Furthermore, it presents a Trickle-based algorithm that extends the QoWater sensor battery lifetime by approximately 4 times.**

I. Introduction

Global water consumption is doubling every 20 years, which is more than twice the rate of human population growth. If this trend persists, by 2025 the demand for fresh water is expected to rise to 56% above the current available amount [1]. More than 663 million people on earth already lack access to fresh drinking water [2]. Additionally, the occurrence of global pollution events has increased [3]. Pressure loss or changes during pipe cleaning or power failure may cause back-flow incidents with contaminated soil water entering through pipe breaks or leaking joints. Climate change may increase the soil water level and the risk of introducing polluted liquid to water distribution networks [4]. Therefore, monitoring and maintaining the quality of water is crucial. To this end, various contamination warning systems (CWSs) have been developed [5]. Such systems monitor the water infrastructure without incorporating customer and public health feedback. In general, a CWS consists of various components, e.g., *on-line monitoring*, *public health notifications*, and *consumer complaint notifications*. Moreover, the placement and maintenance of sensors across the water distribution network is costly and time consuming. To resolve these issues, various placing algorithms have been proposed, which select the minimum number of sensor and their corresponding locations. Various systems have implemented monitoring and surveillance components. However, none of them integrate the end user in the loop. This is still an open interdisciplinary problem. Inspired by the u-map paradigm [6], a user-centric recommendation tool based on the crowd-sourcing/sensing, we developed the QoWater. A u-map client enables the smartphone to collect network and service measurements during a service session. Users can also provide their opinion scores about their services. These objective and subjective measurements are uploaded to a geo-database server for analysis.

The QoWater system employs a wireless sensor network (WSN) to monitor water distribution network infrastructure through objective measurements and collects feedback from users about the water quality (subjective measurements) in real-time. When a contamination is detected, the QoWater can inform consumers, providers, and regulators in real-time through appropriate alerts, aiming to minimize the spread of the contamination. The incorporation of subjective measurement takes place at the QoWater client via a Graphical User Interface (GUI). QoWater sensors monitor the water quality and upload the measurements on the server for analysis. The QoWater system can potentially enrich the monitoring process by incorporating user feedback, improve the timeliness and reliability, reduce the maintenance cost, and raise the awareness of citizens.

Our earlier work [7] presented the main concept of QoWater and the overall system architecture. This work focuses on system aspects, namely, the consumption, response delay, and scalability and analyses its performance. It also presents the V-Trickle algorithm that aims to extend the QoWater sensor battery lifetime.

The paper is structured as follows: In Section II, we overview the related work. Section III describes the architecture of QoWater, while Section IV focuses on the evaluation of the QoWater system. Finally, Section V summarises our conclusions and future work.

II. Related Work

A large body of research focuses on automating the detection of contamination based on stationary sensors [8]–[10].

This work is supported by the General Secretariat for Research and Technology in Greece with a Research Excellence, Investigator-driven grant, 2012-2015 (CoRLAB, PI Maria Papadopouli).

978-1-5090-1161-2/16/$31.00 ©2016 IEEE

Ostfeld *et al.* [8] presented an early-warning detection system (EWDS), which estimates the likelihood of contamination based on the minimum hazard level and level of service. Pelerman *et al.* [11] estimates the contamination event probability using artificial neural networks. An important aspect in these systems is the number and the position of the sensor [12]–[15]. Kessler *et al.* [16] described a methodology to identify an optimal set of monitoring stations, for a given maximum volume of consumed contaminated water prior to detection. While Protapo *et al.* [12] proposed a mixed-integer linear algorithm to identify optimal sensor locations. Extensive work has also been done for the detection of contamination based on mobile sensors. Pelerman *et al.* [17], [18] described the development of a theoretical mathematical framework for modeling mobile sensor movement in the WDN, as well as the integration of sensory data (stationary and mobile) to increase system reliability. Suresh *et al.* [19], [20], presented the localisation and detection events on any flow-based system, using mobile sensors that communicate with given position beacon. Lai *et al.* [21] proposed the TriopusNet system, which changes the topology of mobile sensors by injecting new sensor when a sensor runs out of battery. The QoWater system can employ such state-of-the-art event detection and localisation algorithms.

Many studies highlight the importance of the customer complaints surveillance in contamination detection [22]–[25]. They emphasize the need to improve customer feedback data collection by filtering the calls, using electronic storage and a standardized format [22]–[24]. Additionally, Whelton *et al.* [23] highlighted the need of making the recorded complaints accessible to multiple users. Further, Dietrich *et al.* [25] visualised consumer complaints using tag clouds and spider plots.

The closest relevant applications to QoWater client are the following: Jonoski *et al.* [26] presented a system (deployed on lakes and rivers) that allows a user to select a sensor of the Soil Water Assessment Tool (SWAT) for a specific time period, and display the available SWAT server data on smartphone (png format). The sensor measurements are accessible also by a website. DJB [27] is an Android application that enables residents of Delhi to register complaints about dirty water, missing of manhole cover, sewer overflow, and leakage of water of the WDN. There are also several applications [28] [29] that allow residents to report problems on the 311 systems of U.S. cities. Bick [30] developed an application that takes a picture of water exposed to fluorescent light or placed inside a special device mixed with chemicals. The picture to determine the organic or inorganic qualities of the water. A similar approach was deployed by researchers of the McMasters Biointerfaces Institute [31]. However, none of these applications allow users to assess the water quality in a systematic manner and incorporate this feedback for a joint analysis with the measurements collected from sensors in order to detect more accurately and timely the contamination.

III. System Architecture

The QoWater system consists of three components, namely, the QoWater server, clients, and sensors (Fig. 1). The QoWater client includes a *back-end interface*, *data recorder*, *database*, and its *GUI* as illustrated in Fig. 2. The *querying* allows a user to inquiry about the quality of water in a specific area and time period. Users can adopt two different roles, namely, the *customer* role and *scientist* one with different registration and access requirements. Customers indicate their age and gender, and optionally their name, while Scientists in addition provide information about their profession and their e-mail address. Standard technologies (e.g., public-private key pairs, TLS) are employed to ensure registration and integrity for the client-server communication. A non-register QoWater client is not authorized to query and submit feedback. Depending on the role, different questionnaire forms are used via the GUI for the assessment of the water quality (Fig. 2 (d) & (e)). The customer form provides fields for overall evaluation score, color, taste, odor, appearance, and pressure. A photo captured by a *camera* may also be attached to the customer input. On the other hand, the Scientist form prompts users to insert chemical and biological measurements, as well as information about color, appearance, and pressure assessment. The recorded information may be directly uploaded on the QoWater server or saved locally. Moreover, this information is annotated with the current location of the device, provided by the *WiFi* localisation and timestamp. Without the localisation being enabled, the measurements cannot be uploaded. The *SQLite* database maintains a log of the feedback. The *back-end interface* includes a secure Hypertext Transfer Protocol (HTTP) client for connecting to the QoWater server. Furthermore, the *back-end interface* captures broadcast messages from QoWater sensors.

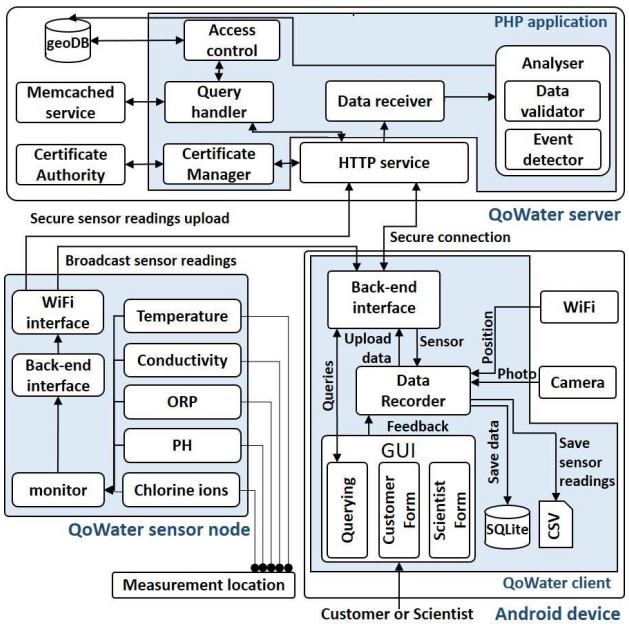


Fig. 1: The QoWater system architecture.

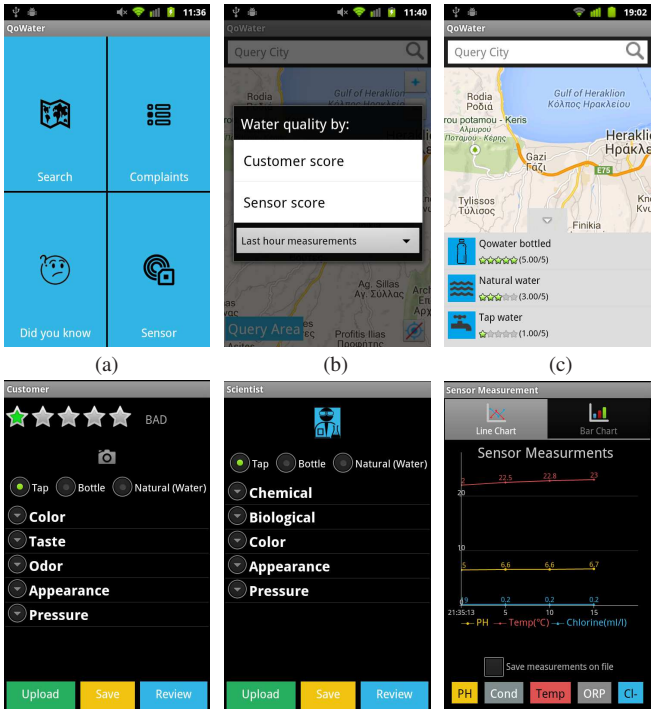
The QoWater sensor consists of a *monitor*, a *back-end interface*, and a number of *sensor probes*. The *monitor* is responsible for sampling and analysing measurements. The *back-end interface* includes an HTTP client responsible for uploading the measurements on the QoWater server. Furthermore, the QoWater sensor broadcasts its measurements to clients.

A QoWater server consists of a *PHP application*, *geo-database*, *memcached* service, *certificate authority and manager*, *access control*, and *analyser*. The *HTTP service* communicates with the QoWater sensors and users. The *certificate manager* determines whether or not the uploaded data have been uploaded by an authenticated QoWater client. The sensitive information (e.g., user position) is protected by the *access control*. The QoWater responses correspond to aggregate feedback collected from more than one user in a relatively large region (e.g., the size of the area specified in a query can not be smaller than a threshold (17,000 m^2)). The *event detector* checks whether or not the sensor measurements comply with the World Health Organization (WHO) standards [32] (e.g., the pH level is in range of 6.5 to 8). In the case of a disagreement, the *event detector* indicates that a contamination event occurred. A client feedback which indicates the presence of "foreign" elements in the water, by default, is considered as a possible contamination event and need to be further analyzed. In this case, the QoWater client requires from the user to upload a photograph. The *query handler* serves the queries sent by clients. The response is returned to the client and stored in the *memcached* service. The stored response expires after a certain amount of time (e.g., request on daily data expires after 2 hours). The *data validator* aims to detect sensor/client malfunctions or miss-calibrations by checking the range of recorded values (e.g., a pH value cannot be greater than 14).

IV. Performance Analysis

This section focuses on the evaluation of the response delay, battery consumption, and scalability, which are critical aspects of the system. The client is an HTC Nexus One smartphone. The QoWater server runs on a virtual machine (VM) with 2 cores at 2.4 GHz, 4 GB RAM.

To measure the power consumption of the QoWater client, we use the AppScope [33] tool, an energy metering framework for Android OS using kernel activity monitoring. The AppScope estimates the energy consumption per hardware component (e.g., CPU, Display, Global Positioning System (GPS), WiFi, 3G), as well as per process. Other approaches, such as PowerScope [34], provide the energy consumption of applications at a fine-grained level but require post-processing using an external device. For example, PowerTutor [35] accesses usage statistics from *profcs and* BatteryStat, built-in Android operations, to provide application-specific energy information, but this approach has several limitations affecting the accuracy. We use the same testbed for the power consumption estimation as the one of the u-map system [36]. The flow and the



(a) (b) (c)

(d) (e) (f)

Fig. 2: GUI screenshots from a QoWater client device: (a) home screen, (b) building of a query, (c) presentation of the query result, (d) customer feedback form, (e) scientist feedback form, and (f) measurements from a sensor node.

events that take place during the AppScope measurements are illustrated in Fig. 3.

We measure the power consumption of the data uploading, querying, and client-sensor communication. The scenarios are described on Table I. The Suspended state takes place when the screen is locked and there is no running application (except system operations processes). Table II reports the mean and in parenthesis the standard deviation of the power consumption during the 100 repetitions of each scenario (mW). The 3G is zero in all scenarios and has been omitted from Table II. The display is the most energy demanding component compared to CPU, GPS, and WiFi across all scenarios (Table II). The QoWater scenario with the greatest total power consumption is the uploading. Even during the most energy demanding scenario of QoWater, its energy consumption is lower compared to Skype and YouTube, since the changing of

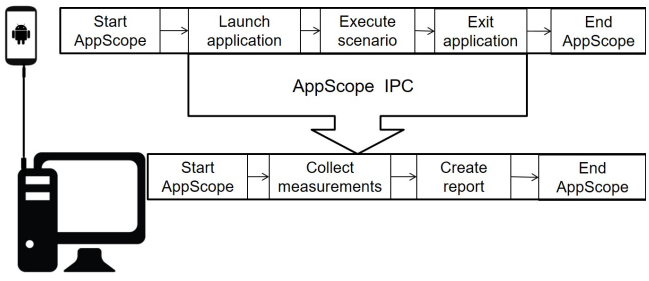


Fig. 3: The QoWater testbed for the power consumption measurement.

TABLE I: Power consumption scenarios

Step	QoWater Uploading	QoWater Querying	QoWater Client-Sensor	u-map Uploading	u-map Querying	Skype Call	YouTube Video
1	Unlock phone	Unlock phone	Unlock phone	Unlock phone	Unlock phone	Call a contact	Unlock phone
2	Launch QoWater	Launch QoWater	Launch QoWater	Launch u-map	Launch u-map	Call duration 20 sec	Launch YouTube
3	Tap Sentiment	Tap Search	Tap Sensor	Tap Upload data	Tap Select Operator	N/A	Display 78 sec video
4	Fill complaint form	Retrieve user location	Display 35 sec sensor measurements	Upload 4.8 MB data	Tap Polygon	N/A	Return to Home Screen
5	Fill 5 star QoE	Tap Query Area	Return to Home Screen	Return to Home Screen	Tap Submit	N/A	N/A
6	Upload 1.5 MB data	Display response	N/A	N/A	Display response	N/A	N/A
7	Return to Home Screen	Return to Home Screen	N/A	N/A	Return to Home Screen	N/A	N/A

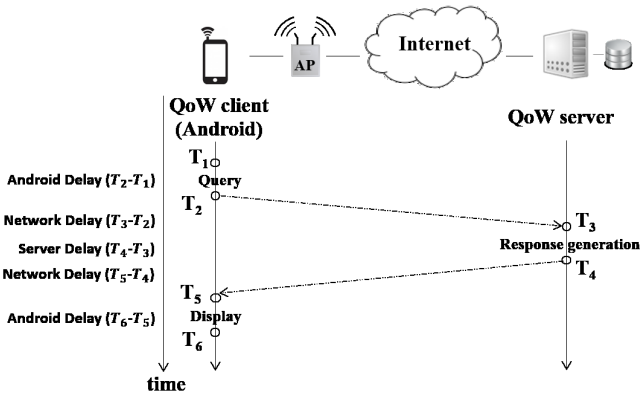


Fig. 4: The QoWater testbed for the delay measurements.

the GPS energy is due to AppScope limitation. Compared to u-map, the uploading of QoWater is more energy demanding, since the number of GUI refreshes is greater and the GUI is colourful. On the other hand, the QoWater querying is less demanding compared to u-map querying, since it relies on WiFi instead of GPS for localisation. To address the limitation of the AppScope in the case of small datasets, short time-state changes (e.g., GPS ON/OFF/SLEEP) and accurately measure the consumption, we employed a larger dataset. The regular dataset size of the QoWater uploading is maximum 5 KB, instead of 1.5 MB. In the experiments, a QoWater client communicates with the QoWater server via an HTTP wireless connection. The uploaded dataset size is 5 KB. The QoWater server contains 100,000 entities. Three types of delay have been estimated, namely, the Android, server, and network delay (Fig. 4). The server delay corresponds to the total time elapsed between the reception of a client request by the server and the transmission of the response (i.e., $T_4 - T_3$). The network delay consists of the time required for the client request to reach the server and the server response to reach the client (i.e., $T_3 - T_2 + T_5 - T_4$). Note that the timestamps T_2, and T_5 are recorded at the android smartphone, while T_3, and T_4 are recorded at the QoWater server. The android delay corresponds to the time it takes to display the server's response to the user ($T_2 - T_1 + T_6 - T_5$). Finally, the total delay is the sum of the Android, network and server delays (i.e., $T_6 - T_1$). Figs. 5 (a) & (d) show the cumulative distribution function (CDF) of the uploading and querying delay, respectively.

TABLE II: Mean and standard deviation of power consumption (*mW*)

Scenario	CPU	Display	GPS	WiFi	Total
QoWater: Uploading	68 (14)	712 (10)	17 (1)	23 (7)	820
QoWater: Querying	56 (3)	548 (15)	17 (1)	8 (3)	629
QoWater: Sensor	4 (1)	767 (10)	0	1 (0)	772
u-map: Uploading	35 (23)	567 (1)	0	88 (14)	690
u-map: Querying	33 (3)	563 (15)	167 (16)	6 (4)	770
YouTube: Video	23 (3)	786 (5)	0	6 (4)	815
Skype: Call	111 (14)	639 (15)	0	57 (20)	807
Suspended	3 (1)	0	0	6 (3)	9

Table III reports the mean delay of each scenario: The Android delay in the QoWater uploading scenario is greater than its corresponding delay in u-map due to the difference in the retrieving process. Moreover, the server delay on QoWater uploading is less than the corresponding of the u-map delay due to the larger processing capabilities of the server. The Android delay of QoWater querying is greater compare to u-map querying, because of the pre-processing of the server response. Additionally, the server delay of QoWater querying is greater than u-map due to different size of stored data. The network delay on u-map querying is smaller compare to QoWater querying because the client is on the same Local Area Network (LAN) with the server, instead of a Wide Area Network (WAN) that QoWater is connected. However, using the memchached service of the QoWater server, the server delay can be reduced to 50 ms, regardless of the size of the queried area or the size of the data that have been stored on the server.

We analyzed the scalability of the system by emulating a number of QoWater clients (running on desktop PC consist of 32 GB RAM, 12 cores, and CPU 2.4 GHz) that execute either the uploading or the querying scenario. The number of clients varies from ten to one thousand. Each scenario was executed 20 times. In these experiments, there is no Android or PC delay. Fig. 5 (b) shows the server and network delay. Each point corresponds to the mean delay of all runs across all emulated clients. The maximum number of concurrent querying (uploading) clients that can be supported by the QoWater server is 250 (150), respectively. Under heavy load, a number of request will not be served in time and a HTTP time-out occurs. To adequately support a large number clients, multiple QoWater server instances should be deployed on a cloud computing environment. Moreover, using the Hadoop Distributed Filesystem (HDFS) and the HBase, a higher aggregate I/O throughput can be achieved.

TABLE III: Mean and standard deviation of the delay (*ms*)

Scenario	Android/Waspmote	Server	Network	Total
QoWater: Uploading	393 (52)	69 (5)	249 (47)	711
QoWater: Querying	102 (19)	1451 (83)	262 (66)	1815
QoWater: Sensor	4000 (223)	32 (1)	4000 (500)	8032
u-map: Uploading	49 (N/A)	159 (N/A)	379 (N/A)	587
u-map: Querying	43 (N/A)	135 (N/A)	8 (N/A)	186

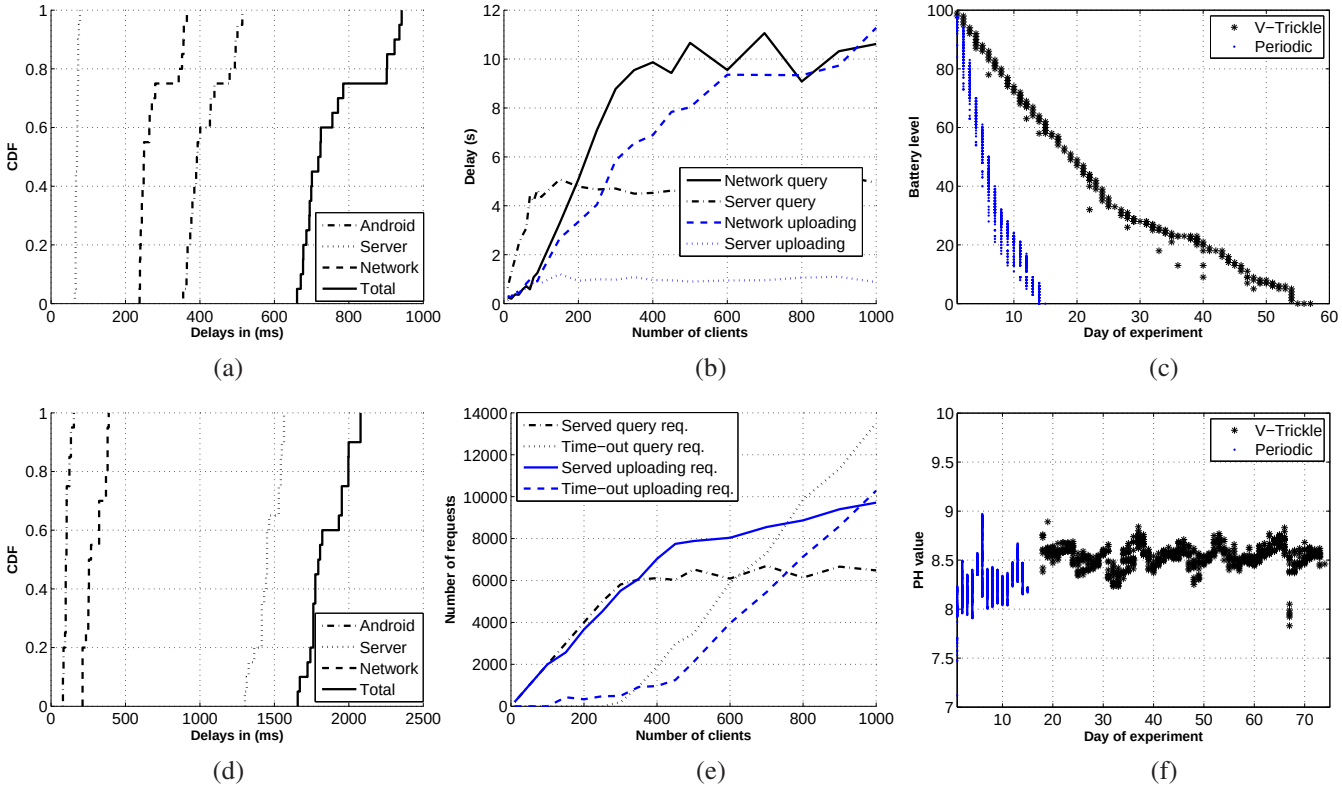


Fig. 5: (a) The uploading delay. (b) The scalability of QoWater system, as a function of concurrent users. (c) The battery level of V-Trickle vs. Periodic. (d) The querying delay. (e) Served vs. time-out request. (f) The pH evolution during the 2 (V-Trickle, Periodic) experiments.

We also examined the possibility of extending the battery lifetime of the QoWater sensor using the V-Trickle algorithm (1) based on the Trickle algorithm, which is used on routing protocol for low-power and lossy networks [37]. We focused on the QoWater sensor uploading operation and compared the V-Trickle with the Periodic approach. The V-Trickle algorithm uploads measurements to the QoWater server in two cases: either when two consecutive measurements differ by 5% or more (irregular uploading), or when no measurements have been uploaded for I counter (regular uploading). When the algorithm starts, it sets the I to I_{min}, and doubles it each time the R becomes equal to the current I. On the other hand, the Periodic approach uploads measurements periodically (with a period of one minute). The Periodic algorithm is a simple loop of a) wake up, b) analyse water, c) send measurement to QoWater server, d) sleep for 35 sec. The sampling duration is of 25 sec. We evaluated the battery life of sensor in these two scenarios (Fig. 5(c)). The V-Trickle approach lasted 55 days, while the periodic approach only 13 days. During each experiment, the sensor uploaded 905 (19,460) measurements in the case of V-Trickle (Periodic) approach, respectively. Note that due to the lack of a second sensor, the V-Trickle experiments followed the period ones, using the same water which has been sensed and analyzed for the periodic experiments. Despite the fact that the V-Trickle approach uploads a smaller number of measurements, it still detects the water quality variation as shown in Fig. 5(f).

Algorithm 1 V-Trickle

$C \leftarrow 1$ /* C: number of samples */
$S \leftarrow T_{min}$ /* S: sleep time (in second), T range [35, 95]*/
$R \leftarrow 0$ /* R: consecutive number of similar samples*/
$I \leftarrow I_{min}$ /*I: current maximum number of similar samples*/
loop
  WakeUp()
  if $C == I$ **then**
    RegularUploading()
    $C \leftarrow 0, R$++
  else
    if $Cur_{Measurement} VS Prev_{Measurement} >= 5\%$ **then**
      IrregularUploading()
      $C \leftarrow 0, S \leftarrow T_{min}, R \leftarrow 0, I \leftarrow I_{min}$
    else
      R++
  if $R == I$ **then**
    $I \leftarrow I + I_{min} * (I/I_{min}), R \leftarrow 0, S \leftarrow T_{max}$
  if $I > I_{max}$ **then**
    $I \leftarrow I_{max}$ /*I_{max}=88*/
  if $I < I_{min}$ **then**
    $I \leftarrow I_{min}$ /*I_{min}=11*/
  C++
  Sleep(S)

V. Conclusion and Future Work

This paper presents the QoWater and its system performance evaluation demonstrating that it has relatively low response delay and power consumption. The most energy demanding

scenario of QoWater system consumes less energy than 20-sec of Skype call. The current server cannot support more than 200 concurrent requests, and thus, it cannot adequately serve large-scale regions. However, it can be easily deployed as a distributed system to satisfy the needs of a larger metropolitan area. The main source of battery drain is the display. Compared to the periodic approach, the V-Trickle algorithm extends the battery life by 4 times, capturing the changes in the water quality. An extended pilot field study with real users is required to evaluate the user satisfaction and system usability (e.g., user friendliness of the GUI, features of the system, questionnaire, privacy issues).

An important challenge is the timely and reliable detection of the contamination events. To enhance the participation of citizens, the offer of regional (or fine-scale) monitoring in specific regions, improved services, and alerts could be important. Moreover, the statistical analysis of measurements to detect false or noisy values provided by malicious users or misconfigured devices is critical. The "user-in-the-loop" requirement in large-scale urban cyber-physical systems triggers several exciting questions that require interdisciplinary research in systems, networks, modeling, environmental engineering, economics, privacy, and social sciences.

REFERENCES

[1] M. Barlow, "The global water crisis and the commodification of the worlds water supply," in *International Forum on Globalization*, 2010.

[2] W. H. Organization and UNICEF, "Progress on sanitation and drinking water," 2015.

[3] C. M. Hogan, "Water pollution," 2014.

[4] I. Kelman, I. Tryland, L. Robertson, A.-G. B. Blankenberg, M. Lindholm, T. Rohrlack, and H. Liltved, "Impact of rainfall on microbial contamination of surface water," *International Journal of Climate Change Strategies and Management*, 2011.

[5] R. Murray, T. Haxton, S. McKenna, D. Hart, K. Klise, M. Koch, E. Vugrin, S. Martin, M. Wilson, V. Cruze *et al.*, "Water quality event detection systems for drinking water contamination warning systems—development, testing, and application of canary," *EPAI600IR-lOI036, US*, 2010.

[6] G. Fortetsanakis, M. Katsarakis, M. Plakia, N. Syntychakis, and M. Papadopouli, "Supporting Wireless Access Markets with a User-centric QoE-based Geo-database," in *ACM MobiArch*, 2012.

[7] N. Rapousis, M. Katsarakis, and M. Papadopouli, "Qowater–a crowdsourcing approach for assessing the water quality," in *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*. ACM, 2015.

[8] A. Ostfeld and E. Salomons, "Optimal layout of early warning detection stations for water distribution systems security," *Journal of Water Resources Planning and Management*, 2004.

[9] D. G. Eliades, D. Stavrou, S. G. Vrachimis, C. G. Panayiotou, and M. M. Polycarpou, "Contamination event detection using multi-level thresholds," *Procedia Engineering*, 2015.

[10] J. A. Roberson, K. M. Morley, A. W. W. Association *et al.*, *Contamination warning systems for water: an approach for providing actionable information to decision-makers*. American Water Works Association, 2005.

[11] L. Perelman, J. Arad, M. Housh, and A. Ostfeld, "Event detection in water distribution systems from multivariate water quality time series," *Environmental science & technology*, 2012.

[12] M. Propato, "Contamination warning in water networks: General mixed-integer linear models for sensor location design," *Journal of water resources planning and management*, 2006.

[13] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan *et al.*, "The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms," *Journal of Water Resources Planning and Management*, 2008.

[14] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos, "Efficient sensor placement optimization for securing large water distribution networks," *Journal of Water Resources Planning and Management*, 2008.

[15] J. W. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J.-P. Watson, "Sensor placement in municipal water networks," *Journal of Water Resources Planning and Management*, 2005.

[16] A. Kessler, A. Ostfeld, and G. Sinai, "Detecting accidental contaminations in municipal water networks," *Journal of Water Resources Planning and Management*, 1998.

[17] L. Perelman, W. Salim, R. Wu, J. Park, A. Ostfeld, M. Banks, and D. Porterfield, "Enhancing water distribution system security through water quality mobile sensor operation," in *World Environmental & Water Resources Congress*, 2013.

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

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

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

[21] T. T.-T. Lai, W.-J. Chen, K.-H. Li, P.-Y. Huang, and H.-H. Chu, "Triopus-net: Automating wireless sensor network deployment and replacement in pipeline monitoring," in *Information Processing in Sensor Networks (IPSN), 2012 ACM/IEEE 11th International Conference on*. IEEE.

[22] N. Mix and B. Pickard, "Customer Complaint Surveillance Updates and Tools—Vendor Integration and Establishing Thresholds," St. Louis, MO, USA, September 10, 2012.

[23] A. J. Whelton, A. M. Dietrich, D. L. Gallagher, and J. A. Roberson, "Using customer feedback for improved water quality and infrastructure monitoring," November 2007.

[24] D. L. Gallagher and A. M. Dietrich, "Statistical approaches for analyzing customer complaint data to assess aesthetic episodes in drinking water," *Journal of Water Supply: Research and Technology-Aqua*, 2014.

[25] A. M. Dietrich, K. Phetxumphou, and D. L. Gallagher, "Systematic tracking, visualizing, and interpreting of consumer feedback for drinking water quality," *Water research*, 2014.

[26] A. Jonoski, A. Almoradie, K. Khan, I. Popescu, and S. Van Andel, "Google android mobile phone applications for water quality information management," *Journal of Hydroinformatics*, 2013.

[27] "Djb," https://play.google.com/store/apps/details?id=com.delhigovernment.djb&hl=en.

[28] "Sf311," http://sf311.org/index.aspx?page=797.

[29] "Minneapolis 311," https://play.google.com/store/apps/details?id=com.seeclickfix.minneapolis311.app.

[30] A. Bick, "Making your water safe, one picture at a time," https://play.google.com/store/search?q=batterydiviner.

[31] C. Sicard, C. Glen, B. Aubie, D. Wallace, S. Jahanshahi-Anbuhi, K. Pennings, G. T. Daigger, R. Pelton, J. D. Brennan, and C. D. Filipe, "Tools for water quality monitoring and mapping using paper-based sensors and cell phones," *Water research*, 2015.

[32] W. H. Organization, "International standards for drinking-water," 1958.

[33] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "Appscope: Application energy metering framework for android smartphone using kernel activity monitoring." in *USENIX Annual Technical Conference*, 2012.

[34] J. Flinn and M. Satyanarayanan, "Powerscope: A tool for profiling the energy usage of mobile applications," in *Mobile Computing Systems and Applications. Proceedings. WMCSA'99.* IEEE.

[35] Z. Yang, "Powertutor-a power monitor for android-based mobile platforms," *EECS, University of Michigan, retrieved September*, 2012.

[36] M. Katsarakis, V. Theodosiadis, and M. Papadopouli, "On the Evaluation of a User-Centric QoE-Based Recommendation Tool for Wireless Access," *ICS-FORTH, Heraklion, Crete, Greece, tech. rep*, 2014.

[37] P. A. Levis, N. Patel, D. Culler, and S. Shenker, *Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks*. Computer Science Division, University of California, 2003.