Scalable Anomaly Detection for Smart City Infrastructure Networks

Dynamically detecting anomalies can be difficult in very large-scale infrastructure networks. The authors' approach addresses spatiotemporal anomaly detection in a smarter city context with large numbers of sensors deployed. They propose a scalable, hybrid Internet infrastructure for dynamically detecting potential anomalies in real time using stream processing. The infrastructure enables analytically inspecting and comparing anomalies globally using large-scale array processing. Deployed on a real pipe network topology of 1,891 nodes, this approach can effectively detect and characterize anomalies while minimizing the amount of data shared across the network.

Djellel Eddine Difallah and Philippe Cudré-Mauroux
University of Fribourg

Sean A. McKenna
IBM Research Smarter Cities Technology Center

Smarter cities are built on physical systems that provide the foundational elements for life in urban areas. We can lump these physical elements together as critical civil infrastructures – the roads, pipes, rail lines, conduits, treatment, distribution, storage, and disposal systems that enable the movement of traffic, water, sewage, pedestrians, and energy throughout the city. As the world continues to urbanize, cities are trying to extract more value from their existing civil infrastructures by extending the lifespan of aging systems and making smarter decisions about infrastructure use, retrofitting, and replacement. Rapidly emerging sensing technologies and sensor networks hold tremendous promise for enabling cities to better manage their civil infrastructure systems. As real-time sensing becomes ubiquitous, new technologies are needed to absorb the large amounts of data generated and provide analytics that can extract useful information from these data streams.

Today, networked sensing and large-scale data analytics could revolutionize how municipal infrastructures are monitored. Sensor miniaturization is offering new sensing modalities with lower power requirements, greater ease of installation, and improved network communications. The data streams from these sensors could let utilities operate infrastructures more efficiently in real time (particularly during extreme events) to identify maintenance issues early and make informed decisions as regards retrofitting or replacing certain civil infrastructure components.

Here, we propose a scalable, hybrid Internet infrastructure that would let monitoring systems dynamically detect

1089-7801/13/$31.00 © 2013 IEEE

Related Work in Wireless Sensor Network Monitoring

A sensor is a low-cost, standalone, micro-electronic component with limited computational ability, built-in sensing components, and a radio transceiver. When a large number of sensors is deployed over a site for monitoring purposes, they form what is called a wireless sensor network (WSN). Ian Akyildiz and his colleagues have summarized the outlook for WSNs in several monitoring applications.[1] The authors discuss numerous applications, including flood detection, biological and chemical detection, agricultural monitoring, and other areas within the environmental monitoring realm. Although WSNs' initial promise in environmental applications hasn't been fully realized,[2] the technology is progressing, particularly in hydraulic and water quality monitoring within water networks. One small prototype monitoring network has been deployed for sewers in Boston.[3] More recently, WSNs were deployed to monitor pressure and acoustics within the Singapore drinking water distribution network.[4] Within this network, 25 monitoring stations in an urban area transmitted 4 to 8 Kbytes/s to a central processing server using a 3G wireless network with an average distance of 1 km between stations.

References

1. I.F. Akyildiz et al., "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, 2002, pp. 393–422.
2. P. Corke et al., "Environmental Wireless Sensor Networks," *Proc. IEEE*, vol. 98, no. 11, 2010, pp. 1903–1917.
3. I. Stoianov et al., "Pipenet: A Wireless Sensor Network for Pipeline Monitoring," *Proc. 6th Int'l Symp. Information Processing in Sensor Networks*, IEEE, 2007, pp. 264–273.
4. M. Allen et al., "Real-Time In-Network Distribution System Monitoring to Improve Operational Efficiency," *J. Am. Water Works Assoc.*, vol. 103, no. 7, 2011, pp. 63–75.

potential anomalies in real time using stream processing, and analyze and compare them globally via large-scale array processing. (See the "Related Work in Wireless Sensor Network Monitoring" sidebar for more on this topic.)

Water Distribution Networks

The civil infrastructure system we focus on is water distribution networks (WDNs). We can model a WDN as a directed graph, typically with some level of looping. The physical manifestations of the graph edges are pipes, whereas the graph nodes are pipe junctions and network end points where water is extracted for consumption. Unlike the electrical grid or communications networks, water networks contain storage both in the network itself and at nodes (that is, tanks or reservoirs). Additionally, transmission rates for water are on the order of centimeters per second (cm/s), rather than the speed of electrons.

Operational efficiency and regulatory directives require monitoring both hydraulic (pressure and flow) and water quality (chlorine, pH, specific conductance, and so on) parameters in WDNs. The current state of the practice is to record hydraulic parameters continuously at only a fraction of the network elements — roughly 0.01 to 0.001. Most water quality monitoring is still performed with noncontinuous samples taken at discrete times and locations within the network. Continuous monitoring of hydraulic and water quality parameters typically uses physically large sensors that require hard-linked power and communications connections to send raw data to a centralized supervisory control and data acquisition (SCADA) system.

Earlier technology dictated that external power, communications, and, in some cases, wastewater connections be available at any monitoring location. These requirements and the physical footprint of previous-generation sensors significantly constrained the locations within a network where cities could install sensors and added considerable costs to monitoring network installation.[1] In many ways, water utilities had come to rely on citizens to overcome these limitations by playing a large role in infrastructure monitoring — that is, reporting situations such as breaks in water mains or strange odors or tastes indicating degraded water quality.

We are only beginning to realize the full potential deploying new sensors could have within WDNs. Before this vision can become a reality, however, we must solve several key issues.

Ease of Sensor Deployment

Fine-grained WDN monitoring requires installing sensors at every network node. However, installing flocks of smart sensors with wide-area network (WAN) or on-board processing capabilities in underground water pipes represents a significant challenge in terms of both installation and operational management. Hence, the sensing infrastructure used in large-scale WDN deployments should be as simple, robust, and energy efficient as possible.

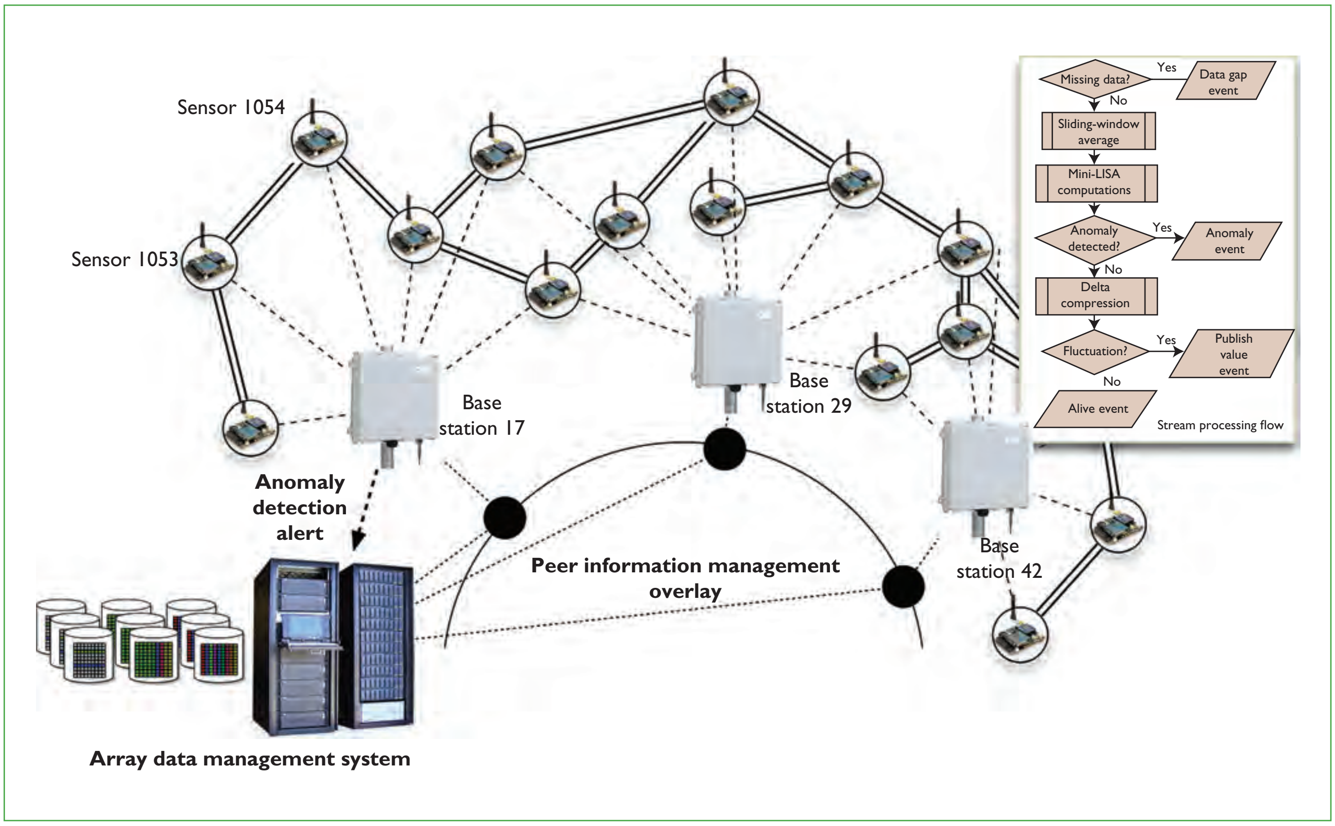

Figure 1. Water distribution network data management architecture. The architecture has three main components: simple water sensors that periodically broadcast their measurements; self-organizing base stations that gather the sensor readings and clean them using a stream-processing flow (on the right) and share them through an overlay network; and an array data management back end that durably stores and analyzes all values.

Real-Time, Network-Scale Monitoring

Resource theft or leakage detection is highly time sensitive. Detecting potential anomalies in real time can be challenging for large-scale WDNs consisting of tens of thousands of nodes. Current sensor deployments for WDNs impose high delays with regard to data acquisition (up to several hours) that must be drastically reduced. In addition, the sensor data collected are often erroneous or noisy, and human operators must curate them manually before they are usable. Also, the main-memory batch-processing software currently used to process data (such as Matlab), though highly efficient for small operations, can't scale to larger networks, thus limiting current solutions' practical applicability.

Big Data Analytics

Finally, analytics and demand forecasts require processing a significant amount of historical WDN data. No solution is readily available for durably storing and efficiently managing such data. Legacy relational databases are ill-suited to handle the enormous quantities of (nonrelational) time series that flocks of sensors produce over time. Processing platforms such as Matlab are even worse because they consider only simple, flat files. Hence an urgent need exists for new WDN storage and data- processing infrastructures that can analyze historical readings at scale.

Architecture Overview

Our hybrid stream/array-processing architecture meets these challenges in the context of WDN monitoring. Specifically, we use our architecture to compute the Local Indicators of Spatial Association (LISA) metric for anomaly detection.[2] In addition, we extend the metric to consider temporal associations.

Figure 1 depicts a simplified view of our distributed Internet infrastructure for handling data from WDNs. We can split the overall architecture into three main components: the water sensors themselves, which monitor and locally broadcast flow, pressure, or water quality values in the WDN; the stream-processing subsystem comprising sensors in the vicinity of base stations that gather, consolidate, and forward sensor readings in real time; and the *array database management system* (ADBMS) back end, which globally analyzes and durably stores all data originating from the WDN.

Sensing Infrastructure

Producing fine-grained analyses of large WDNs requires deploying myriad sensors to cover all nodes (and potentially also edges) in the pipe network. To reach this goal, we deliberately limit sensor functionalities to reduce our sensing infrastructure's cost, energy consumption, and number of potential failures. We propose using easily installable, low-cost, durable sensors whose only duty is to intermittently broadcast their measurements over local, low-powered digital radio channels.

Various technologies are available for such sensors, such as ISA100.11a (www.isa.org/ISA100-11a), IEEE's 802.15.4 media access control layer (www.ieee802.org/15/pub/TG4.html), or full-blown Zigbee (www.zigbee.org) modules. Depending on the exact technology used, the sensors might act as simple wireless transmitters broadcasting to base stations only, or can self-organize into transceiver mesh networks communicating over longer distances (that is, passing data through intermediate devices to reach more distant ones). We can deploy several sensor types in the pipes in this way, monitoring flow, pressure, or water quality, for instance.

Stream-Processing Subsystem

Our stream-processing subsystem consists of a handful of more powerful base stations scattered across the WDN that collect all sensor measurements. Regardless of whether they self-organize into a mesh network, the sensors always communicate to the base stations using redundant, point-to-multipoint (P2MP) broadcast communication to minimize data loss in the advent of sensor or base station failures.

The base stations' first function is to gather live measurements originating from neighboring sensors. Each cell constitutes a substream-processing system centralized at the base station. It applies stream operations on the measurements on-the-fly and intermittently transmits the locally aggregated information together with the raw data to the analytic back end.

The base station first tries to detect gaps in the data stream. It fires a *data gap* event (see the right side of Figure 1) whenever the frequency of the messages it receives from a given sensor falls below a certain threshold. The base station then applies a simple smoothing function to the data by running a sliding window average on the values to level out potential noise in the measurements. It applies a local anomaly-detection algorithm (which we discuss in detail later) on the resulting values and fires an anomaly detection exception that it sends to the data management back end in case it detects any abnormal pattern. Finally, the base station applies delta compression to the current data and pushes the new value to the overlay network when it differs somewhat from the last transmitted value (that is, if $|v_{\text{current}} - v_{\text{last\_transmitted}}| \geq \varepsilon$). If the system is running in steady state and doesn't observe any fluctuation, sensors shares less data, and the base station emits only an occasional *alive* message. This lets us collect values frequently (for instance, several times a minute, thus reducing the time-to-anomaly-detection delay), while minimizing the traffic the base stations generate, because practically no data is shared in the overlay in steady-state mode.

In terms of networking and data-sharing capabilities, we require all base stations to have WAN modules (such as High-Speed Downlink Packet Access or Wi-Fi). They share data with each other and with the analytic back end by maintaining a dynamic peer-to-peer (P2P) overlay network.[3] The base station registers the events and data it must share in the overlay network by applying consistent hashing[4] on the involved sensor's identifier (for example, `publish(hash(sensor123), "sensor123 pressure at 2013-11-05T08:15:30-05:00 : 196000"))`. All data are stored in the overlay using soft states and have an expiration date after which the system deletes them.

The overlay network hence serves two main purposes: First, it consolidates all values originating from a given sensor stream (remember that the sensors and the base stations are loosely coupled, and that several base stations could report values for the same sensor). Second, it serves as a scalable and robust information management system to expose data to our architecture's third tier, discussed next.

Array Data Management: The Renaissance

Array data management has long been a popular topic in computer science. It gained renewed interest recently, motivated by the rapid emergence of extremely large array data in eScience and Web analytics. In the past few years, several new initiatives such as SciDB (www.scidb.org),[1] SciLens (www.scilens.org), Rasdaman (www.rasdaman.com), SciHadoop,[2] or KeplerDB[3] were launched to provide new solutions to this problem.

Contrary to traditional database systems, this new wave of *array database management systems* (ADBMSs) supports only limited transactional functionalities and focuses instead on distributed array processing and analytics. These systems seek to combine several decades of efficient structured data processing from the relational world with the latest advances in distributed batch-processing à la MapReduce. ADBMSs typically support scalable linear algebra operators over massive shared arrays stored natively on large clusters of commodity machines. They are hence much faster than relational databases on array analytics and linear algebra workloads, and scale to much larger datasets than main memory matrix-oriented systems such as Matlab and R. Technically, we can summarize their benefits in three points.

Native Array Storage

ADBMSs store both dense (images or videos, for example) and sparse (adjacency matrices) multidimensional arrays natively, using compact structures to physically collocate adjacent cells on disk. These systems also seek to provide the advanced array management features that many scientific applications request, such as data lineage or array versioning.[4]

Scalable Processing

Extremely large arrays are increasingly common in astronomy (www.lsst.org), bioinformatics, Web analytics, or smart city contexts such as the one we describe in the main text. Because such data typically can't fit on one machine, ADBMSs support horizontal scaling to provide scalable array processing of extremely large arrays using clusters of commodity machines. In this case, the arrays are typically partitioned (or "chunked") across several physical nodes, which then run parallel versions of array operators locally.

Declarative Interfaces

Like relational database systems, ADBMSs increasingly provide declarative interfaces to let data administrators process arrays easily using libraries of linear algebra and array operators that can be combined using languages similar to SQL — for instance, the SciQL (www.scilens.org/Resources/SciQL) or AQL (www.paradigm4.com/technology/aql-afl-query-languages/) query languages.

References

1. P. Cudré-Mauroux et al., "A Demonstration of SciDB: A Science-Oriented DBMS," *Proc. Very Large Databases Endowment*, vol. 2, no. 2, 2009, pp. 1534–1537.
2. J.B. Buck et al., "SciHadoop: Array-Based Query Processing in Hadoop," *Proc. 2011 Int'l Conf. High Performance Computing, Networking, Storage and Analysis*, ACM, 2011, pp 66:1–66:11.
3. S. McCauliff et al., "The Kepler DB: A Database Management System for Arrays, Sparse Arrays, and Binary Data," *Proc. SPIE 7740, Software and Cyberinfrastructure for Astronomy*, SPIE, 2010, p. 77400M; doi:10.1117/12.856792.
4. A. Seering et al., "Efficient Versioning for Scientific Array Databases," *Proc. IEEE 28th Int'l Conf. Data Eng.* (ICDE 12), IEEE, 2012, pp 1013–1024.

Array Database Management System Back End

Our architecture's final component is a back end ADBMS. A new wave of such systems is under development, pushed by the rapid growth of large-scale array data emerging in eScience, Web analytics, and smart city contexts such as ours (see the "Array Data Management: The Renaissance" sidebar for more information).

The ADBMS back end has two main roles in our architecture: First, it durably stores all sensor values and anomalies that the base stations share (the values shared in the P2P overlay being ephemeral, as we described). Second, it provides global analytics capabilities, both to monitor in near-real time the WDN as a whole, and to analytically compare/forecast demand or anomaly patterns over time.

The streaming subsystem directly collects sensor values from the information overlay network that the base stations create. Such overlays support efficient range queries — for example, to retrieve all recent measurements — as well as point queries (typically requiring $\log(O(N))$ messages, where N is the number of nodes in the network) that enable, for instance, dynamically retrieving mini-LISA statistics or specific values when the system detects an anomaly. The database back end, on the other hand, stores all data and anomaly statistics compactly in multidimensional structures. It also stores additional information such as each sensor's type, location, and ID, and time stamps for most information.

Analysts can implement many useful analytic operations on top of the collected data. So far, we have implemented two operations that we think represent the two main classes of query in our context — namely, global monitoring using LISA statistics and historical anomaly comparison using pattern matching over LISA statistics. We describe both in more detail in the remaining sections.

Local Indicators of Spatial Association

LISA statistics were originally developed to detect anomalies in geographic studies,[2] where the observations are associated with physical coordinates and geographically weighted connections to other observations. The Moran's I test serves as an example LISA calculation value:

$$LISA(v_a) = \left(\frac{v_a - m}{S}\right)\left[\sum_{k=1}^{K}\left(\frac{1}{K}\right)\left(\frac{(v_k - m)}{S}\right)\right] \qquad (1)$$

Here, v_a is the observation at the current node a, with its K neighboring nodes (connected to a through the network topology) having observation values v_k. Observations are standardized using the mean of all current nodes' measurements, $mean(v) = m$, and the standard deviation $stdev(v) = S$. LISA value calculation measures local clusters of similar measurements. In practice, the LISA value's sign indicates the presence of high- or low-value clusters when positive, or outliers when negative, while its magnitude indicates how much the local value differs from its neighbors.[5] Here, we use multiple random permutations of the observations across all network nodes to calculate the LISA statistic values under the null hypothesis of complete spatial randomness[2,6] against the alternative spatial-clustering hypothesis.

LISA statistics have only recently been applied to network topologies, and to date these applications don't include WDNs. In our context, we define LISA networks using the WDN topology and identify anomalies within the sensor data. In addition, we propose two extensions.

LISA with Temporal Association

Most published work on LISA statistics has been for time-stationary problems in the geographic domain (for example, identifying hotspots of criminal activity or cancer mortality). We extend the local neighborhood to contain both temporal and spatial neighbors. Thus, we enlarge the set of K measurements around a node a to include its own previous measurements in addition to both the current and past measurements from its neighbors.

Mini-LISA for Real-Time Anomaly Detection

From a performance perspective, computing LISA values and conducting statistical-significance tests implies a global knowledge of the network's state, which can be time consuming in large deployments (owing to missing values, slow connections, computational overhead, and so on). Our solution uses the stream subsystem at each base station, thus limiting the population used for the mean and sigma computations to only those nodes informing a single base station. Given the limited spatial information necessary to detect local anomalies, we compensate by using a larger temporal window on each node to identify anomalies based on previous values in the subnetwork.

The System in Action

We built a prototype of the architecture we present here using Twitter Storm (http://storm-project.net) as a stream-processing engine and SciDB (www.scidb.org) as the ADBMS.

ADBMS Setup

First, a network of N nodes is represented by an adjacency matrix. Initially, our network only holds direct connections, where a cell $Network[i, j] = 1$ represents an edge between nodes i and j. Additionally, and to save subsequent processing time, we store a limited transitive closure of up to three hops for each node — that is, $Network[i, j] = k, \forall ij$, where j is k hops away from node i. We achieve this using SciDB's *multiply*() built-in operator. To support temporal locality, we reshape *Network* into a cube *TimeNetwork*, where the third dimension is time. That is, a cell $TimeNetwork[i, j, t] = k, \forall ij$, where j is k hops away in space, in time, or both from node i. The weights of the temporal links, or *edges*, are predefined in the following to 0.5, which implies that a node's [*current_time* – 1] observation is preferred to its direct neighbor's [*current_time*] observation. As time goes by, a listener process receives incoming vectors of observations from the overlay network or the stream-processing subsystem directly. The observations are appended in an unbounded 2D array.

LISA Operator and Distributed Significance Test

Next, we developed a LISA operator that takes a parameter K indicating how many neighbors

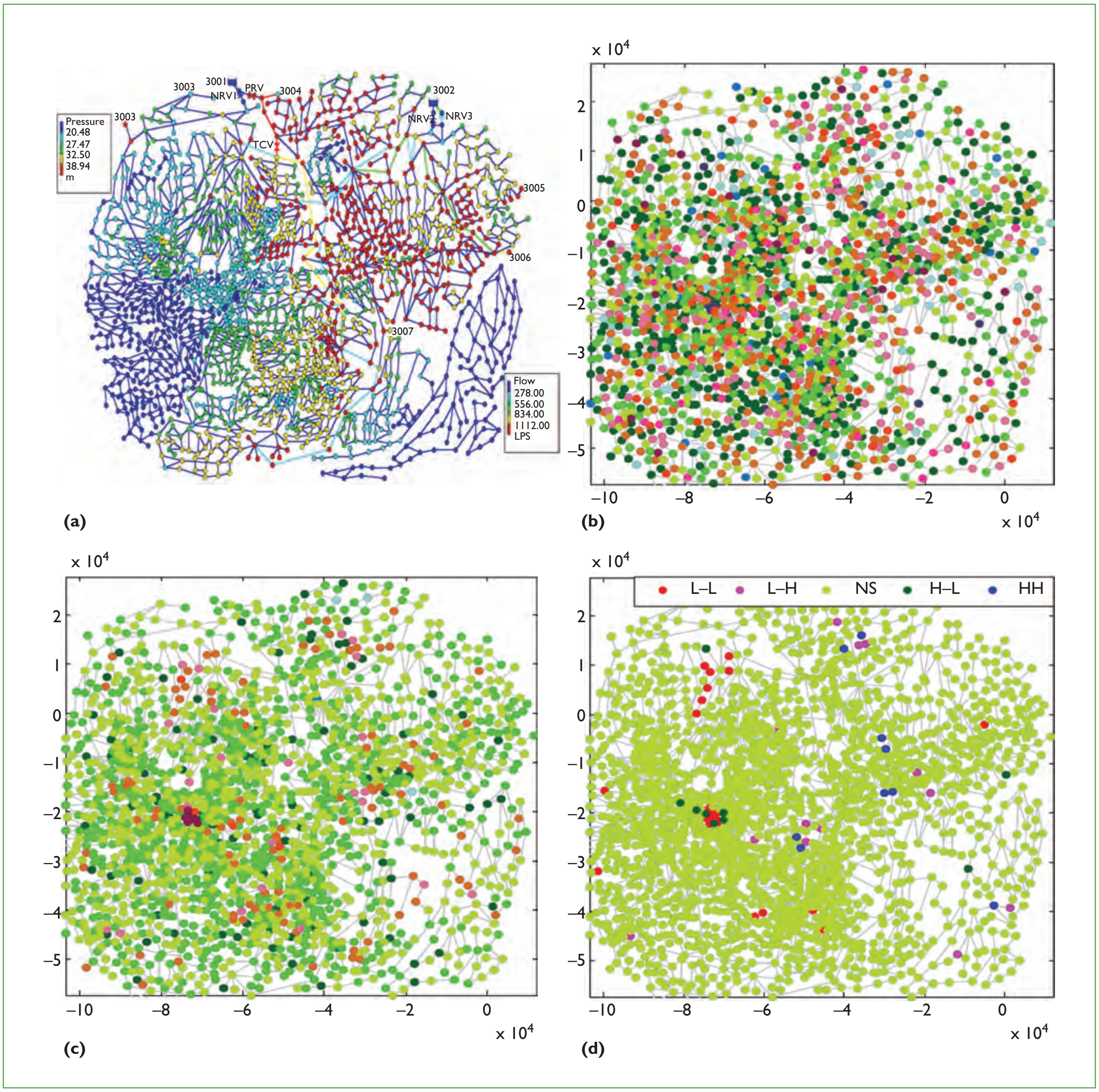

Figure 2. Water distribution network simulation and results. (a) The hydraulic simulation shows pressure (meters) at the junctions and flow (liters per second) within the pipes. We can see (b) the actual observed values in the network, (c) the computed Local Indicators of Spatial Association (LISA) statistics considering eight neighbors with one backtracking time, and (d) the cluster map. Significant high-high and low-low clusters are highlighted, as are high-low and low-high outliers. The remaining points are considered statistically insignificant.

to include, and a parameter T that specifies the maximum time backtracking. The operator constructs the LISA values and then proceeds to a significance test to determine anomalies that reject the null hypothesis where a significance level of α = 0.010 is used. This process is particularly computationally intensive: because we compute a 1,000-value statistic for each node in the network, it greatly benefits from a distributed computing system such as that SciDB offers.

LISA Analytics

In addition to being able to compute global LISA statistics, we have also implemented simple LISA analytics operations. We can thus, given a new anomaly, search for previous anomalies that might share some similarities with the new one, or cluster anomalies using pattern matching on the historical LISA data stored in the ADBMS.

Performance Evaluation

To evaluate our approach's results and performance, we have used a real WDN topology constructed for a medium-sized city in the UK (http://emps.exeter.ac.uk/engineering/research/cws). This network comprises 1,891 junctions and 2,465 pipes, and is designed to supply water to a city with roughly 400,000 people. To meet our scalability promises, we also implemented a scalable network generator that generates topology with generally square connections, a coordination number of 4, and equal edge lengths. We consider the observations to be residuals between a predictive model providing the expected values of a property (such as pressure) across the network and the actual observations. As such, we consider the observed residuals to be independent and identically distributed with $N(0, 1)$.

Effective Anomaly Detection

The system produces three visualization maps (see Figure 2). The cluster map lets users identify at a glance significant anomalies in the network, which we classify as

- *clusters* — HH (high-near-high measurements) and LL (low-near-low measurements); or
- *outliers* — LH (low measurement in a high neighborhood) and HL (high measurement in a low neighborhood).

The remaining points are nonsignificant measurements.

In our experiment, in addition to isolated anomalies that are present at random, we manually introduced a cluster of five low-negative anomalies with different intensity levels to simulate this specific scenario. We can see that the anomaly cluster is correctly highlighted at the middle left of Figure 2c (in red). We also successfully tested our approach's scalability on large topologies of tens of thousands of nodes, and of our LISA analytics capabilities in subsequent tests (we don't provide these additional experiments' numeric results owing to space constraints).

New sensing infrastructures could revolutionize how municipal infrastructures are monitored and handled. Analytics solutions applied to smart water data, for instance, could provide the basis for variable pricing, detection of resource theft or leakage, load or demand forecasts, and incentivizing consumers to conserve resources. Our architecture has several distinct advantages, including

- ease of deployment and management, because the data sensing, gathering, and analytics components are all loosely coupled, self-organizing, and able to be installed independently;
- real-time monitoring, including local anomaly detection at the base stations directly using Mini-LISAs; and
- analytics and global processing capabilities using a horizontally scalable array data management system.

We have shown using an early implementation of our architecture that our solution scales to real, large water topologies and can detect anomalies successfully. Our ultimate goal is the real-time understanding of water systems at scale based on (both spatial and temporal) fine-grained WDN quality monitoring and on the architecture and methods we have described. IC

References

1. T.M. Walski, *Water Quality Sensor Placement in Water Networks with Budget Constraints*, tech. report, Sandia Nat'l Laboratories, 2005.
2. L. Anselin, "Local Indicators of Spatial Association LISA," *Geographical Analysis*, vol. 27, no. 2, 1995, pp. 93–115.
3. P. Cudre-Mauroux, S. Agarwal, and K. Aberer, "Gridvine: An Infrastructure for Peer Information Management," *IEEE Internet Computing*, vol. 11, no. 5, 2007, pp. 36–44.
4. D. Karger et al., "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," *Proc. 29th Ann. ACM Symp. Theory of Computing*, ACM, 1997, pp. 654–663.
5. P. Goovaerts and G.M. Jacquez, "Detection of Temporal Changes in the Spatial Distribution of Cancer Rates

Using Local Moran's I and Geostatistically Simulated Spatial Neutral Models," *J. Geographical Systems*, vol. 7, no. 1, 2005, pp. 137–159.

6. P. Goovaerts and G.M. Jacquez, "Accounting for Regional Background and Population Size in the Detection of Spatial Clusters and Outliers Using Geostatistical Filtering and Spatial Neutral Models," *Int'l J. Health Geographics, vol. 3, no. 1, 2004, p. 14.*

Djellel Eddine Difallah is a research assistant at the eXascale Infolab and a third-year PhD student in computer science at the University of Fribourg, Switzerland. His research interests are in large-scale data management, distributed computing, and crowdsourcing. Difallah is a Fulbright Alumni and a member of ACM. Contact him at ded@exascale.info.

Philippe Cudre-Mauroux is a Swiss-NSF professor and the director of the eXascale Infolab at the University of Fribourg, Switzerland. His research interests are in large-scale data management infrastructures for non-relational data. Cudre-Mauroux has a PhD from the Swiss Federal Institute of Technology (EPFL). Contact him at phil@exascale.info.

Sean McKenna is the senior research manager for water and the environment at the IBM Research Smarter Cities Technology Center in Dublin. His research interests include numerical modeling and parameter estimation in ground water systems, spatial statistics, and development of analytics and optimization solutions for water distribution networks. McKenna has a PhD in geological engineering from Colorado School of Mines. Contact him at seanmcke@ie.ibm.com.