Open Problems in Data Collection Networks

Jonathan Ledlie, Jeff Shneidman, Matt Welsh, Mema Roussopoulos, Margo Seltzer
Harvard University
{jonathan,jeffsh,mdw,mema,margo}@eecs.harvard.edu

Abstract

Research in sensor networks, continuous queries (CQ), and other domains has been motivated by powerful applications that aim to aggregate, assimilate, and interact with scores of sensor networks *in parallel*. Numerous system ingredients are necessary to make these applications possible. Sensor network research is building some of these components from the bottom up, dealing with issues such as wireless connectivity and battery life. CQ, peer-to-peer (P2P), and other research areas are building top down, examining in-network services, naming, decentralized queries, and scale. While many research groups use the same types of applications to motivate their work, many of these applications cannot be built today because of missing bridge research. These challenges include: uniting vastly differing devices and services, managing intermittent connectivity, placing in-network services with QoS and other constraints, developing unified security models, and correlating between sensor networks. This paper distills these new problems and outlines one proposed system that explores solutions to these concerns.

1 Introduction

Visionary application scenarios inspire and motivate computer science research. Descriptions of these systems are compelling even to a non-technical audience. For instance, medical first responders wish to track patients' vital signs and treatments wirelessly at the scene of an accident and immediately make this information available to remote doctors. A more pedestrian scenario finds a driver who wishes to navigate the roads of a busy city to find the "best" parking spot to her destination, taking into account cost, weather, current traffic, and preferred walking distance. These applications appear within reach, and yet currently neither of them can be built.

These scenarios, and the many like them, rely on sensor networks, middleware, distributed query processing, and the work of many other self-contained research disciplines. However, the union of these disciplines leaves several significant research questions unanswered, because each of these fields makes a set of assumptions that do not hold throughout the entire system. Salient open problems emerge when one takes a more holistic cross-systems viewpoint.

At its most basic level, research in sensor networks examines how to efficiently push sensor data through a wireless infrastructure to one or more base stations. Efficiency is gained by intelligently inferring events from the sensed data using in-network processing [14]. Although there has been much research in message-passing algorithms [15, 17], on-the-fly sensor reprogramming [21] and query languages [23, 34], most of the proposed solutions essentially end at the base station.

Likewise, Internet-based data processing research has taken many forms over the years, including work in classic distributed systems, agents [20], publish/subscribe [2, 5, 27, 29], Grid [11] and peer-to-peer scenarios [16, 31]. These systems have focused on other applications, such as federating databases [26], harnessing compute cycles [30], and Web-based content distribution [3]. Most recently, the Continuous Queries work (CQ) from the database community offers in-network processing of streaming data in stable, homogeneous networks [6, 7, 8, 25]. Much Grid work has examined naming and creating common interfaces (*e.g.,* WSDL) [10], a piece in the puzzle needed to link disjoint sensor networks. P2P focuses on scale and disconnection, frequently at the expense of complex queries and a good naming system. All of these groups make different assumptions about their data model and in the connectivity, stability, and consistency, of their networked participants.

Applications that will aggregate, assimilate, and interact with geographically diverse sensor networks share some of the requirements for, and can use some of the existing solutions from, sensor- and Internet-based data processing systems. However, a brief analysis of a few application scenarios reveals a set of research topics that must be addressed before these applications can be realized: uniting vastly differing devices and services, managing intermittent connectivity, placing in-network services with QoS and other constraints, developing unified security models, and correlating data across sensor networks. We outline each of these research challenges in turn and describe the initial design of Hourglass, a scalable data collection network, that is intended to address them.

2 Motivating Applications

This section delineates two reasonable vision applications that are both representative and currently unattainable. These applications highlight the open technology problems that future research should address.

2.1 Medical Monitoring

One application driving research at groups at Harvard, Sun, HP, and MIT is medical monitoring [33]. Consider a mass casualty triage scenario, beginning at an accident and ending with care at a hospital. At the accident, medics first attach sensors to patients to quickly determine their status, just as they now evaluate patients manually to determine who needs help first. A typical sensor might be a wireless, finger-mounted pulse oximeter that describes several patient characteristics at once, facilitating rapid discovery of the patient's condition. As medics get to work, information about treatment and patient status flows to the medics' PDAs and local ambulances. Even as medics move around, they are notified when a patient's condition suddenly changes.

Medics, or a dispatcher also monitoring this information, would then choose an appropriate hospital for each patient, based on each hospital's location and current staffing, bed availability, and road conditions. En route to the hospital, additional data about each patient is transmitted from the ambulance to prepare the emergency room team before the patient arrives.

Later, properly authorized medical researchers might gain limited access to the data in order to study how patients respond to treatment and how medical training might be improved.

This scenario is an example of a data collection and analysis system that must run securely over intermittent connections on a set of disparate devices.

2.2 Infrastructure and Environment Monitoring

Prototypes for monitoring the structural integrity of buildings and bridges [18, 19], the availability of parking spots [9], and the health of natural environments [4, 24] have been proposed and developed. Previous work in infrastructure and environmental monitoring has generated efficient protocols for wireless communication and examined the use of heterogeneous devices for longevity. Separate research has touched on in-network data processing. IrisNet, for example, includes video cameras attached to computers which perform data processing computation and pass the data to a database [9]. This work, as with all CQ work of which we are aware, has sensor streams flowing directly into computers that are connected to a stable core of homogeneous computers.

By combining data from several existing sensor networks, one can build a richer application. Sensor data could come from wireless sources, as it does in the infrastructure and environment monitoring research. More complex processing of wide-spread streams could occur at wired and more stable nodes as it does in CQ. Gluing together these two domains will allow us to answer more interesting application questions that tie together data from multiple sensor clusters.

3 Open Problems

In this section, we review the significant open problems that need to be addressed before we can construct these interesting applications.

Uniting vastly different devices and services. Devices in vision applications will vary widely in storage capacity, availability, network bandwidth, energy usage, processor speeds, and component characteristics; there is not a simple differentiating line between sensors and PCs. Sensor networks are already being built with heterogeneous nodes, where a well-provisioned node may act as an anchor for battery-powered sensors [19]. The intermediate network nodes, responsible for processing and relaying data between sensors and applications, may also be heterogeneous and thereby limit their interchangeability. For instance, the medical monitoring scenario ties together multiple wireless sensor types (*e.g.*, oximeters, ECG), PDAs, laptops, a necessarily reliable and secure routing network, and legacy hospital systems. This diversity presents problems of naming and interfaces, since establishing a *lingua franca* across systems may not be possible.

An extensive body of work on naming, lookup, and interfaces has come from Grid and ubiquitous computing research [1, 10, 13]. Because applications will need to reach below the base station, existing protocols that require point-to-point communication, significant translation processing, and large payloads may prove too heavyweight. Once mechanisms for handling this broad heterogeneity are developed, holistic application development may be facilitated with an expansion of macroprogramming [32].

Functioning continuously despite intermittent connectivity. Medical monitoring and other scenarios may involve mobile, wireless, battery-powered devices. For instance, as medics move around, patient data is not merely forwarded to the ambulance via PDAs. Instead, sufficient information must be available to at least one PDA for medics to be able to react to changes in patients' conditions. Systems like this one must seamlessly transform themselves when reconnected with a larger network, when the movement has ceased.

Partitions in the network appear to occur at well-defined locations, *e.g.*, at the ambulance, and when medics walk out of range of patients. Similar deterministic partition points exist in building and environmental monitoring and,

if explicitly planned for, could allow significant optimizations and changes in operating mode.

The existence of both critical data and partitions differentiates this work from existing work: CQ typically ignores partitions, while most sensor networks drop data until they can be repaired. In the scenarios described here, partitioned nodes need to be particularly intelligent about what data is dropped; some data may be needed as soon as reconnection occurs. Much work in P2P applies to these disconnect/reconnect scenarios, but the well-defined partition points and the differences between wired and wireless links makes it difficult to draw directly from this work.

Placement of in-network services constrained by Device and Network. The process of transmitting data between sensors and applications will be enhanced through in-network processing. For example, if many cars all need the same traffic data from several highways simultaneously, we should not route each tuple to each car; instead, we should aggregate data. CQ has two current solutions to this problem: (1) use a centralized database or (2) decentralize the operators (but with manual control, *e.g.,* the boxes and arrows of Aurora [35]). A centralized mechanism becomes unscalable in the general case. In light of the vastly different devices these systems will contain (and their scale), asking humans to solve this problem seems untenable. Operator placement and online optimization, constrained by device capability, is a hard problem, particularly when operators must be carefully placed within the sensor network according to some constraints (e.g., efficiency, access control rights, etc).

Placement of in-network services is a multi-step process. First, an application demand is translated into a series of operators. In-network operators receive inputs from live sensor data from one or more networks, from stored data, and from other operators. Some operators logically and physically exist within sensor networks. Then, path placement determines the order in which data is to pass through service instantiations; it maps operators onto services that the network can actually provide. This mapping is constrained by application Quality of Service (QoS) requirements, and communication between services may be limited by their heterogeneity. We could try to leverage existing algorithmic techniques, particularly those from the database community. However, because sensor data is expected to be lossy and noisy and because applications will have significant timing constraints, it does not appear that using a distributed database for inter-sensor network communication is the right solution.

System-wide security: integrity, privacy, and authentication. Several of the applications driving this work have stringent security restrictions. *Authentication, access control,* and *encryption* are important in medical applications because of federal privacy regulations, and, in systems where ownership of data is an issue, access control is important. It must not be possible for an eavesdropper to acquire patient data and for researchers to obtain data to which they have not explicitly been given access.

Solving this problem requires a unified threat model: if we design for adversaries of type x (*e.g.,* complete parameter knowledge), at what points can the system be attacked? While encrypting transmissions between sensors and between nodes in the larger network is a start, it does not provide a true analysis of the breaking points. In particular, many of the weak points are likely to be the ones created at the new transition points between sensors and a larger network. Addressing these issues will, built upon existing techniques for security analysis. However, this type of threat model analysis can only be performed well when applications are considered in their entirety, something that has yet to be done.

Power to draw inferences both within and between sensor networks. These macrospatial systems introduce the ability to draw new types of inferences. Sensor network researchers are examining how to use several sensors in a room to cross-correlate to prevent false positives. Frequently this *sensor fusion* is done because sensors with "identical" hardware have sensitivities that differ by orders of magnitude. We can extend the notion of inference logic to reach across base station barriers. Instead of cross-correlation within a single network, applications can say: "There has been a detection of event x in another sensor network near you, be more watchful for it until told to stop." This issue reaches below where CQ systems process data, and by definition, it functions above sensor networks. Sensor-oriented databases like TinyDB [23] may assist in this effort with triggers and syntax, but higher-lever constructs will most likely be necessary.

4 Hourglass: A Data Collection Network

We have outlined the five major problems that must be addressed in order to build certain compelling sensor applications. In this section, we present a system that our research group is building to address these five topics.

We propose to address the challenges outlined above by developing a *data collection network* (DCN), a robust infrastructure for discovery, querying, and delivery of sensor network data. We are currently developing *Hourglass*, a DCN that is intended to scale to a large number of concurrent applications pulling data from a vast number of geographically-diverse sensor networks. Hourglass is based on an Internet-based overlay network in which nodes act as both routers and stream processing engines. Unlike traditional CQ systems, Hourglass specifically addresses the challenges of intermittent connectivity, security, and node heterogeneity.

The Hourglass architecture is depicted in Figure 1. Data

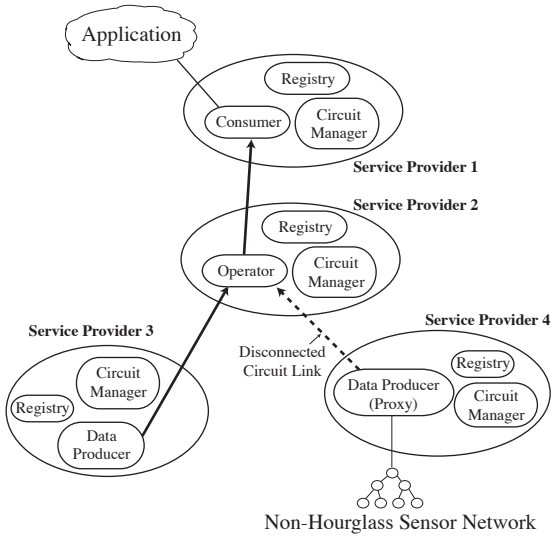

Figure 1: Hourglass Data Collection Network. Data producer proxies manage the interaction with sensor networks and Consumer proxies engage with applications. In-network services compress, filter, aggregate, and temporarily store sensor data. Service continues under disconnected operation, where only some services will function and where devices are heavily constrained (*e.g.*, in terms of power, bandwidth, and latency).

flow in Hourglass is based on a *circuit*, which is a data path through the system that ensures that an application receives the data in which it is interested. A circuit includes intermediate *services* that perform operations on the data. Services are organized into distinct *service providers* that capture a single administrative domain. Each service provider includes a *circuit manger*, which is responsible for the setup and management of circuits, and a *registry*, which aids service discovery.

We are designing Hourglass with five overarching points of design: use of connected circuits, leveraging distinction between core and periphery nodes, lightweight service composition, remaining agnostic to user data types, and scalable resource discovery.

Connected Circuits A *circuit* is a fundamental abstraction that links a set of data producers, a data consumer, and in-network services into a data flow. A circuit enables applications to express their data needs at a high level and pass the responsibility for creating data flows to the DCN, thus simplifying the implementation of sensor data applications. Data injected into the circuit by data producers is processed by intermediate services and then delivered to data consumers.

As illustrated in Figure 1, a circuit in Hourglass is a tree with a data consumer as the root, and data producers as leaves. Data flows towards the consumer of the circuit and is processed at intermediate nodes. Nodes in the circuit can refer to Hourglass services in the system by including a *service endpoint* that binds a given circuit node to an actual instance of a service. A service endpoint could be implemented as an IP address and port number. Multiple circuits can share particular physical realizations of the circuit links within a circuit, avoiding duplicate transmission of data that is used by more than one circuit. A circuit also has a globally unique *circuit identifier* that is used to refer to it throughout the system.

Circuits are established by the *circuit manager* according to requests from applications. An established circuit is associated with a lease and needs to be refreshed periodically, otherwise it is removed from the system. Such a soft-state approach prevents the build-up of stale circuit information after application failures.

Core *vs.* **Periphery** The DCN will be heterogeneous but components will tend to exhibit relatively stable availability, link, and storage properties over time. We plan to leverage these characteristics to make services aware whether they exist on core or peripheral nodes and act accordingly. Core nodes are more stable and would advertise stable storage. More transient nodes would instead provide routing and buffering services. Learning about and using the broad but slow-changing characteristics of the DCNs constituent nodes will let us make good choices about service placement and migration.

Service Composition Hourglass supports a range of *in-network services* that can be dynamically instantiated on nodes along a circuit. Hourglass supplies a small set of stream processing services. Third parties can supply application-specific or resource-intensive services that run on specific hosting centers and may be included in a circuit. One example is a stream storage service, which requires significant (persistent) disk resources; unlike other services, the storage service will not migrate across physical servers (although it may perform internal replication and fail-over that is transparent to the circuit).

Hourglass services provide a suite of stream-processing operations. Two representative services are the buffer service and filter service. A *buffer service* is responsible for buffering data during disconnection and delivering it to the rest of the circuit after reconnection. When a new circuit is created whose semantics require that it be resilient in the face of disconnection, buffer services are inserted at wireless circuit links that are prone to disconnection by the circuit manager. A *filter service* restricts the data that flows through a circuit according to a filter expression. This service depends on the data model used in the circuit. For example, a filter service can reduce the bandwidth consumption of a circuit. The circuit manager can relocate filter services in order to optimize the efficiency of a circuit.

Remaining agnostic to data types Due to the heterogeneity of the environment, Hourglass does not enforce a global

data model for all circuits. Instead, a single circuit can combine different data models, such as partially-structured or relational data, with a range of data schemas, as long as the services involved are able to understand each other, for example, by translating between data representations.

Scalable Resource Discovery To provide a system-wide mechanism for discovery of sensor sources, existing circuits, and instantiated and potential services, we are designing and building a scalable resource discovery layer into Hourglass. This layer is fault-tolerant and exhibits good locality due to its use of a distributed hash table. The mechanism expands topic-based publish-subscribe with strong support for predicates. It builds trees of topics which interested parties and either subscribe or anycast to. Several important areas of research into the Registry layer into the dynamic awareness of core and transient nodes and allowing some topics to be very broad (*e.g.*, storage) while others contain extremely specific predicates.

5 Related Work

The Continuous Queries (CQ) work from the database community offers in-network processing of streaming data in stable, homogeneous networks [6, 7, 8, 12, 22, 25]. This community has addressed issues of operator placement, which is also important in a DCN. TelegraphCQ aims to work in "unpredictable" situations: nodes can fail or query optimization information can be incorrect; users can restate their queries on the fly. However, the criteria by which CQ systems place operators do not include the possibility of intermittent connectivity. Whereas TelegraphCQ and NiagraCQ move the data to a central processing point, Hourglass can act on data either at or close to the publishing node. Additionally, these systems do not address the scalability challenge that we face, nor do they offer the wide range of data flow semantics that are necessary in our target applications.

Most closely related to the notion of a data collection network are systems such as IrisNet [9], PIER [16], Astrolabe [28] and Medusa/Aurora [8], which are intended to support distributed queries over many disparate, real-time data sources using techniques such as overlay networks and dynamic query operator placement. In particular, Aurora [35] is a system designed to support applications that monitor continuous streams of data. While Aurora centralizes stream processing, Hourglass provides a distributed framework for circuit construction. PIER uses a DHT for tuple storage, spreading data around the network based on the namespace and primary key. In contrast, Hourglass creates circuits, yielding a scalable infrastructure like PIER but without the high latencies induced by PIER's DHT architecture. Astrolabe uses hierarchical attribute aggregation for distributed stream management and gossiping for faster attribute propagation.

Hourglass is also closely related to Grid initiatives on resource discovery and computation on streaming data. This work has focused on naming and creating common interfaces for data and computation [10], as well as harnessing computational resources [30] and federating databases [26]. A data collection network faces similar problems; however, the Grid approach generally assumes a stable and relatively high-performance network infrastructure. In contrast, DCNs must gracefully handle temporary disconnection as well as a range of connection bandwidths to sensor networks and the application endpoints receiving sensor data. A DCN encompasses a broader and more diverse set of participants than a traditional Grid system. While common interfaces are important, the critical issue for DCNs is providing interfaces for which minimal functionality can be implemented on resource-constrained devices, which may require interfaces to be backed by sophisticated services on more capable nodes.

More broadly, the Hourglass approach differs from these systems in several key respects. First, we envision an extremely rich set of services that can collect, filter, aggregate, and process sensor data as it flows through a network; the DCN should not constrain the set of services to a small set of operators for a specific query interface. Such an approach allows the system to evolve to support a wide range of as-yet-unforeseen applications. Second, Hourglass is designed to cope with mobility of sensor and application endpoints and the resulting temporary disconnections from the rest of the network. Third, Hourglass dynamically incorporates heterogeneous devices into the system. CQ systems currently do not allow for this dynamic behavior, yet it will occur in long-lived applications that Hourglass aims to address.

6 Conclusions

Motivated by several vision applications, this paper identified five open areas for future systems research. We believe that the problems of handling heterogeneity, handling intermittent connectivity, performing constrained in-network service placement, addressing systemwide security, and drawing inferences between sensor networks, are major enough to warrant additional research attention. We introduced Hourglass, a data collection network, that is being used to explore these problems.

7 Acknowledgements

This material is based upon work support by the National Science Foundation under grant number 0330244. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] L. Arnstein, R. Grimm, C. Hung, J. Kang, A. LaMarca, G. Look, S. Sigurdsson, J. Su, and G. Borriello. Systems support for ubiquitous computing: A case study of two implementations of Labscape. In *International Conference on Pervasive Computing*, Zurich, Switzerland, August 2002.

[2] S. Bhola, R. Strom, S. Bagchi, Y. Zhao, and J. Auerbach. Exactly-once Delivery in a Content-based Publish-Subscribe System. In *2002 International Conference on Dependable Systems and Networks (DSN 2002)*, Bethesda, MD, June 2002.

[3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proceedings of the 19th ACM SOSP*, Bolton Landing, NY, October 2003.

[4] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, Costa Rica, April 2001.

[5] R. Chand and P. Felber. A scalable protocol for content-based routing in overlay networks. Technical Report RR-03-074, Institut EURECOM, February 2003.

[6] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[7] J. Chen, D. DeWitt, F. Tian, and Y. Wang. NiagraCQ: A Scalable Continuous Query System for Internet Databases. In *SIGMOD / PODS 2000*, Dallas, TX, May 2000.

[8] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik. Scalable Distributed Stream Processing. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[9] A. Deshpande, S. Nath, P. Gibbons, and S. Seshan. Cache-and-Query for Wide Area Sensor Databases. In *SIGMOD 2003*, San Diego, CA, June 2003.

[10] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. Grid services for distributed systems integration. *IEEE Computer*, 35(6), 2002.

[11] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. `http://www.globus.org/research/papers/ogsa.pdf`, June 2002.

[12] D. Goodman, J. Borras, N. Mandayam, and R. Yates. INFOSTATIONS : A New System Model for Data and Messaging Services. In *IEEE VTC*, Phoenix, AZ, May 1997.

[13] J. Gray, D. Slutz, A. Szalay, A. Thakar, J. vandenBerg, P. Kunszt, and C. Stoughton. Data Mining the SDSS SkyServer Database. Research Report MSR-TR-2002-01, Microsoft Research, January 2002.

[14] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, October 2001.

[15] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient Communication Protocols for Wireless Microsensor Networks. In *Hawaii International Conference on System Sciences (HICSS)*, Maui, Hawaii, January 2000.

[16] R. Huebsch, J. Hellerstein, N. Lanham, B. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *VLDB'03*, Berlin, September 2003.

[17] D. Johnson, D. Maltz, and J. Broch. *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*. Addison-Wesley, 2001.

[18] S. Kim, T. Oberheim, S. Pakzad, D. Culler, J. Demmel, and G. Fenves. Structural Health Monitoring of the Golden Gate Bridge. `http://www.eecs.berkeley.edu/~binetude/ggb/`, 2003.

[19] V. Kottapalli, A. Kiremidjian, J. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei. Two-Tiered Wireless Sensor Network Architecture for Structural Health Monitoring. In *SPIE'03*, San Diego, CA, May 2003.

[20] D. Kotz, R. Gray, and D. Rus. Transportable agents support worldwide applications. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, Connemara, Ireland, September 1996.

[21] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *NSDI 2004*, San Francisco, CA, March 2004.

[22] S. Madden and M. Franklin. Fjording the Stream: An Architecture for Queries over Streaming Sensor Data. In *ICDE Conference*, San Jose, CA, February 2002.

[23] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *SIGMOD 2003*, San Diego, CA, June 2003.

[24] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, Sept. 2002.

[25] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query Processing, Resource Management, and Approximation in a Data Stream Management System. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[26] Petascale virtual-data grids. `http://www.griphyn.org/projinfo/intro/petascale.php`.

[27] P. Pietzuch and S. Bhola. Congestion Control in a Reliable Scalable Message-Oriented Middleware. In *Middleware 2003*, Rio de Janeiro, Brazil, June 2003.

[28] R. Renesse, K. Birman, D. Dumitriu, and W. Vogel. Scalable Management and Data Mining Using Astrolabe. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, March 2002.

[29] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *NGC 2001*, UCL, London, November 2001.

[30] Seti@home. `http://setiathome.ssl.berkeley.edu`.

[31] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *SIGCOMM '02*, Pittsburg, PA, August 2002.

[32] M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *NSDI 2004*, San Francisco, CA, March 2004.

[33] M. Welsh, D. Myung, M. Gaynor, and S. Moulton. Resuscitation Monitoring With a Wireless Sensor Network. In *Circulation: Journal of the American Heart Association*, October 2003.

[34] Y. Yao and J. E. Gehrke. Query Processing in Sensor Networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, CA, January 2003.

[35] S. Zdonik, M. Stonebraker, M. Cherniack, U. Cetintemel, M. Balazinska, and H. Balakrishnan. The Aurora and Medusa Projects. In *Bulletin of the Technical Committe on Data Engineering*, March 2003.