

Collecting Data for One Hundred Years

Jeffrey Shneidman, Bryan Choi, Margo Seltzer
Harvard University
Division of Engineering and Applied Sciences
{jeffsh, bchoi, margo}@eecs.harvard.edu

Abstract

Sensor clusters (aka sensor networks) are interesting because of the role they play in distributed data collection. Conventionally, a sensor network designer had to worry about getting data from the sensors to a processing application. This paper argues that dedicated data collection networks are better suited to this data collection task. Like the sensor clusters they support, data collection networks are designed to run for years. Together, the sensor clusters and data collection network form a system that must overcome temporal obstacles that are not considered in today's networking infrastructure. These temporal issues are now revealed only because sensor networks are being designed to run embedded in buildings, pavement, and other "permanent" structures.

This paper focuses on the issues involved in building and maintaining a data collection network. The goal is for a sensor designer to simply activate a new sensor cluster and have data collection "just work"; data generated from this cluster should be automatically relayed and distributed to data processing applications.

This paper presents the temporal issues facing data collection networks and design suggestions to overcome these problems. We present the initial design of a prototype data collection network that we are building.

1.0 Introduction

A recent photograph in the New York Times Magazine is a stunning example of network archaeology. The picture of twenty vertical feet of a torn up New York city street (post World Trade Center cleanup) shows the detritus of almost one hundred years of networking. The strata show pipes dating back

to 1913 at several foot intervals. Five feet below the Qwest fiberoptic cable, one can spy the old pneumatic tubes used by Western Union and the Postal Service to route mail around a busy early-20th-century Manhattan.

Even in our industry where a technology's half-life can be measured in days, physical elements of a network frequently exist for years. In the past, this infrastructure has been discarded. However, there have been recent efforts (both research [13] and commercial [2]) to create useful long-term networks. Sensor networks of nodes embedded in buildings or painted on a wall have been put into action [6]. "Smart buildings" with sensors measuring stress, light, heat, motion, audio, and even the presence of particular individuals are expected to perform for the lifetime of the structure [13]. It is conceivable that sensor networks, designed to collect data on the scale of years with little or no post-installation user modification, will soon be commonplace.

Deployed in such circumstances, sensor networks are intended to be the "front line" in distributed data collection. A common model is for sensor nodes to detect data and send this information, via relay nodes, to applications that run on more powerful machines [9]. This approach requires sensor network creators to build not only the sensors, but also the infrastructure to move collected data to interested applications. Each time a sensor network is deployed, researchers must re-invent the wheel when considering how to get data from collection nodes to the applications requesting information.

In an alternate universe with widely deployed data collection networks, sensor designers can simply distribute sensor nodes in buildings, paint, or consumer products [13, 8, 1] and know that once activated, the collected data will propagate to the appropriate application.

This is possible because in the alternate universe, computer science researchers have figured out how to build general-purpose data collection networks that deliver data from sensors to applications.

This paper's premise is that the tasks of (1) building a sensor and (2) pushing the collected data to user applications should be separated. This is justified since networking technology will no doubt improve over the lifetime of the sensor. It is probable that the mechanism used to transmit data to applications now (TCP over IP, for example) will be a relic years from now. It is equally conceivable, however, that the data collected by these sensors will retain its relevance. Applications wishing to use sensor data should be free to use whatever network and system paradigm exists in the future, while sensors should not be torn down and replaced simply because their communication mechanisms are obsolete.

Another advantage of this build/push task separation is that once a data collection network is in place, it can be used by new sensor systems and applications at little additional hardware cost. The only possible infrastructure needed to support a new sensor cluster is a bridge from the network of the sensor cluster to the "outside world" data collection network.

The right approach when building embedded sensor systems is to build a corresponding data collection network that addresses the problems of aging. This network acts as a sensor's only customer and is responsible for relaying gathered information to data stores and applications.

What do these data collection networks look like? The idea proposed in this paper is to create relay nodes that send data packaged in a well-defined OSI-application-level protocol. Sensor nodes send collected data to a "sensor entry point," which is a bridge that converts data from the sensor cluster's network/payload format to the standardized format of the OSI-application-level protocol. This entry point then "magically" relays the data to the interested application. In other cases, the sensor itself may be able to "speak" the OSI-application-level protocol and interact directly with the data collection network. (See Figure 1.)

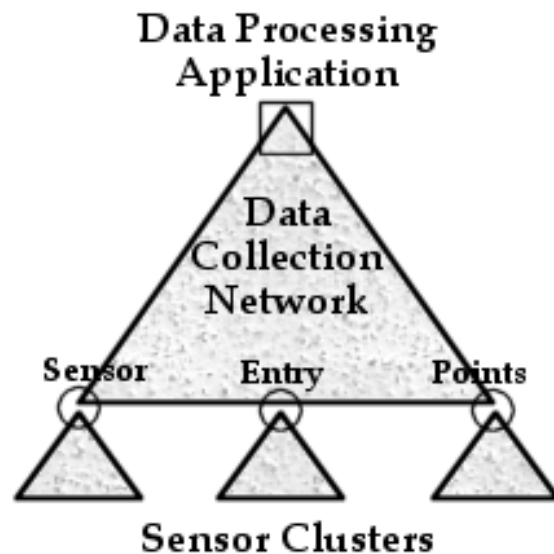


Figure 1: Sensor clusters (which may each have very different designs) interact with the Sensor Entry Point, which is a bridge to the data collection network. This network is made up of relay nodes which transmit data to the application level (a data store or a processing application.)

This paper has three goals: The first goal is to define and motivate the long-term data-collection problem. The second goal is to present a general approach to thinking about data collection networks that overcomes some of the temporal issues we have presented. Finally, we present a high-level description of the prototype data collection network that we are constructing.

This paper is organized as follows. In the next section, we discuss in more detail the challenges that network longevity introduces. In section three, we present general approaches to the problem and in section four, we present our prototype system, Hourglass, that addresses the issues we have laid out.

2.0 The Problems of Time

The challenges facing long-running data collection networks are a superset of those faced by regular networks. The existing problems created by security concerns, privacy issues, and limited resources (including power restrictions)

are applicable to our domain. Data collection networks are special, however, in that they introduce new time-related problems.

Temporal issues can be broken down into micro-temporal and macro-temporal with respect to the lifetime of the network. Most existing network research has focused on the micro-temporal problems. Typical micro-temporal problems considered in the networking community include routing and connectivity. Node transience is also a micro-temporal problem in mobile networks as nodes move in and out of range of neighbors.

Macro-temporal issues deal with the problems of operating on time scales that span generations of technologies. New problems include dealing with the possibility that:

1. network implementations change.

Network protocols (not to mention the models) are not constant. We assume that since sensor clusters might be physically embedded in long-lasting structures (e.g., concrete), their own communication infrastructure will be fixed. These clusters will rely on bridges to the outside world to share results. However, networks in the outside world might change. As relay nodes are added to a data collection network over the years, they might speak different protocols or employ different network models.

2. node implementations change.

As nodes are added to a data collection network, one cannot assume that they will run the same operating system, have the same programming model, or be homogeneous in any way.

3. node capabilities and usages change.

Over time, nodes added to a data collection network might have different abilities (to be described further in section 3.3), and existing nodes are subject to failure and obsolescence. As new types of sensor clusters are disseminated in an existing network, the data format, bandwidth, and frequency of data sent to applications will change.

If one asks, “How can I collect data from a non-static network for the next hundred years?” it quickly becomes apparent that the

fewer fixed assumptions a system relies on, the more time-scalable the system becomes.

3.0 General Solutions

In this section, we outline general approaches to the macro-temporal issues of change.

3.1 Network implementation details change? **Use application level protocols.**

It seems like common sense that time-scalable models should remain agnostic to the details of the underlying node/network implementation. A suggested tenet of mobile network design is to eliminate network inter-layer dependencies [11]. The OSI networking stack is intended to be a logical reinforcement of the idea that different problems should be solved at different layers, but papers describing the woes of mobile computing over the Internet demonstrate how poorly the isolation between layers has actually been achieved [11]. Data collection networks present a new chance to “get isolation right.” At what level does it make sense to place our data collection protocol? The most time-scalable way to think about network design is to consider building an application level protocol. They are a useful abstraction because they allow us to distance ourselves from the physical network implementation. The whole point of an application-level data collection protocol is to avoid embedding data in networking layers that will change.

3.2 Node implementation details change? **Keep programming models out of the network.**

If we assume that individual node architectures can change, it is important to keep the application programming model out of the network. The oft-repeated end-to-end argument suggests “dumb networks” where application smarts are placed at the endpoints of a network [10]. Other forms of networking involve placing application intelligence (sometimes called filters or agents) at various points in a network. Examples of these “active networking”

approaches include D'Agents, and the TinyOS/Continuous Streams projects [5, 7].

When it comes to a data collection network, there are several problems with allowing applications to inject their own logic into the underlying network.

First, the complexity of nodes in such a network increases. If applications are permitted to inject logic into network nodes, problems arise. How do applications negotiate to run their logic in the network? How much space can application logic consume on a node? How many simultaneous application logics can a network node support? How is faulty application logic retracted? There are a number of questions that must be considered. Protocols necessarily become more complicated to deal with this added complexity.

Second, the maintenance of such a network increases. In a network designed to last several generations, as little as possible should be assumed about the underlying network nodes. If virtual machines are employed to hide the underlying details, nodes become restricted to the limitations of a particular virtual machine. If processor-native code is to be run on individual nodes in the network, either the application binary interface (ABI) must be well-specified and legacy code automatically supported as new network nodes are introduced, or many versions of application logic need to be created to support different processors as they are released.

Finally, embedding logic in the network requires programmers to adopt a new programming model that is not always intuitive or conducive to debugging. Distributed debugging of a data collection network that supports application-integrated logic is non-trivial.

3.3 Node capabilities and usages change? **Use self-adaptation and management.**

Nodes in the data collection network will have different capabilities that make them more or less suited to relaying different kinds of information. Although we have argued that the programming model should be left out of the data collection network, we are exploring use of active networking to provide common tasks like compression. Others have proposed that

Data Processing Applications

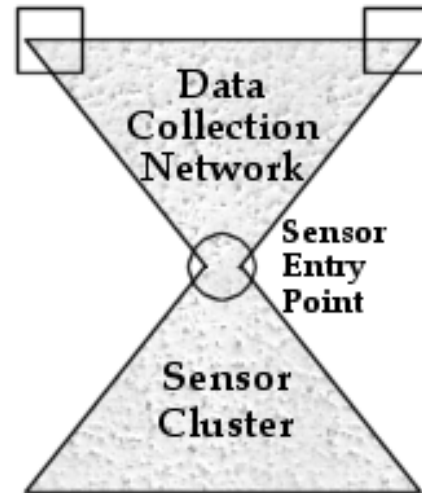


Figure 2: This alternate architecture (that inspired the name of our prototype, Hourglass) shows one sensor cluster sending data to multiple applications via the Sensor Entry Point. As in Fig. 1, the network is made up of relay nodes. We're exploring issues of self-management within the data collection network but treat the sensor cluster as a "black box."

compromises in end-to-end networking and active networking designs may be appropriate in some situations [10], and the data collection network falls into this category. Many sensor nodes probably lack the computational power to do significant "end" tasks such as compression. If the sensor entry point can offer features like this to fit a data stream, the data collection network would be providing a valuable service. We are exploring ways to allow a data collection network to add "features", such as compression, cost accounting, etc., to data streams.

As a data collection network matures, certain relay nodes may become "data hot spots" requiring more bandwidth than is available. If nodes in a data collection network can send feedback to the end-level application about where additional relay nodes are needed, or perhaps even automatically recruit nearby inactive relay nodes to help share the load, the network will be better able to handle different sensor cluster workloads.

Such self-adaptation and management, studied in previous sensor cluster research [3, 4], can also be used in the data collection network.

4.0 An Introduction to Hourglass

We are in the initial phases of building Hourglass, an XML-based data collection network. This network is designed to address the temporal issues outlined in section two. In our setup, we are building the data collection network with off-the-shelf desktop machines and handheld devices to communicate with small sensor devices using standard physical infrastructure including Ethernet, Bluetooth and 802.11 for communication.

Our implementation is designed to organize itself into a tree structure, with different layers corresponding to different functions within the network. At the top lies the main data store where all the collected data is gathered. The lower layers consist of intermediary computation and data nodes which may or may not have specialized functionalities. The primary task of these nodes is to discover sensor entry points and pass up any data packets to parent nodes higher in the network. These intermediary nodes may also feature additional abilities such as compression, encryption, or other tasks neglected by the low-level sensors.

Figures 1 and 2 depict alternate logical architectures using our data collection system. The lower part of the diagrams correspond to sensor clusters. The points at the middle represent the special "sensor entry point" which must be able to translate the data passed up by the sensor node into an XML packet that is intelligible to our system. Thus if a sensor node can collect its data and package it with the required XML metadata, the sensor node can act as its own sensor entry point. On the other hand, if a sensor designer does not wish to build in this complexity, a bridge must be built so that the sensor can be recognized by our network. In our implementation, the metadata will be used to perform self-management and adaptation tasks.

Nodes publicize "resource signatures" which describe the set of resources that the node can offer to other nodes. Resource signatures are useful if the network is self-managing and a node wants to choose between several parent

nodes. A signature consists of several core properties including communication abilities such as latency and bandwidth, and computational abilities such as compression and encryption. The signatures are extendable and can be used to represent other properties such as battery power. Extendable signatures allow the data compression network to support unanticipated node abilities that could be useful in self-management decisions.

In addition to pure data packets being sent upstream, parent nodes can send configuration messages downstream toward the sensor entry point. Configuration messages will be limited to simple pre-defined tasks such as "turn on all sensor nodes in room A."

We are in the early phase of building this network.

5.0 Conclusion

In this paper, we introduced the concept of the "data collection network," a dedicated relay network designed to propagate data collected by a sensor cluster. We argued that temporal problems should play a role in designing these networks, and we identified three classes of problems. We proposed general approaches to use when designing data collection networks. Finally, we introduced Hourglass, which is our initial implementation of a distributed data collection network.

References

- [1] Auto-ID Center, <http://www.autoidcenter.org>
- [2] Commercial structural health monitoring companies: Acellent Corporation, <http://www.acellent.com>; Smartec Corporation, <http://www.smartec.ch>
- [3] Cerpa, A., Estrin, D., "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies," *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, June 2002.

[4] Embedded Everywhere: A Research Agenda for Networked Systems of Embedded Computers, National Research Council, National Academy Press, Washington, D.C., 2001.

[5] Gray, R., Cybenko, G., Kotz, D., Rus, D., "Mobile agents: Motivations and State of the Art," Technical Report TR2000-365, Dept. of Computer Science, Dartmouth College, 2000.

[6] Lynch, J., Law, K., Kiremidjian, A., Carryer, E., Kenny, T., Partridge, A., Sundararajan, A, "Validation of a Wireless Modular Monitoring System for Structures," *Proceedings of the 9th International SPIE Symposium on Smart Structures and Materials*, San Diego, CA, USA, March, 2002.

[7] Madden, S., Shah, M., Hellerstein, J., Raman, Vijayshankar, R., "Continuously Adaptive Continuous Queries over Streams," *Proceedings of ACM Special Interest Group on Management of Data (SIGMOD 2002)*, Madison, WI, June, 2002.

[8] Pushpin Computing at the MIT Media Lab, <http://www.media.mit.edu/resenv/Pushpin/>

[9] Ratnasamy, S., Estrin, D., Govindan, R., Karp, B., Shenker, S., Yin, L., Yu, F., "Data-Centric Storage in Sensornets", Submitted Manuscript, <http://lecs.cs.ucla.edu/~estrin/papers/dht.pdf>, 2002

[10] Reed, D., Saltzer, J., Clark, D., "Comment on Active Networking and End-to-End Arguments", *IEEE Network*, 12(3), May/June 1998, 69-71.

[11] Snoeren, A., Balakrishnan, H., Kaashoek, M.F., "Reconsidering Internet Mobility," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, Elmau, Germany, May 2001.

[12] Sullivan, R., "Dig It: Rebuilding Lower Manhattan, from the Ground Down," *New York Times Magazine*, May 12, 2002, 46-49.

[13] *Proceedings of the Third International Workshop on Structural Health Monitoring*, September 12, 2001, <http://structure.stanford.edu/workshop/>