# Provenance in Sensornet Republishing

Unkyu Park and John Heidemann

Information Sciences Institute
University of Southern California
{ukpark,johnh}@isi.edu

**Abstract.** Sensornets are being deployed and increasingly brought on-line to share data as it is collected. Sensornet *republishing* is the process of transforming on-line sensor data and sharing the filtered, aggregated, or improved data with others. We explore the need for data provenance in this system to allow users to understand how processed results are derived and detect and correct anomalies. We describe our sensornet provenance system, exploring design alternatives and quantifying storage trade-offs in the context of a city-sized temperature monitoring application. In that application, our link approach outperforms other alternatives on saving storage requirement and our *incremental compression* scheme save the storage further up to 83%.

## 1 Introduction

Sensor networks have been proposed and deployed for study of scientific phenomena with levels of detail that was previously impossible [23, 12, 25, 10]. Research groups are using sensornets to study microclimates, animal habitats, or geology. To date, these deployments are undertaken by different research groups each to accomplish a specific objective. While many make their data available, reuse of data remains rare, and collaboration across multiple sensornets rarer still.

As sensornets become easier and more widely deployed, sharing data *across* sensors becomes increasingly important [20]. Several groups have recently begun exploring the role of the Internet in sharing sensor data [20, 21, 17], both to interconnect isolated sensornet patches, and to lower the barrier to sharing sensor data. In the limit, we see a world of *slogging* (sensor logging), where thousands of individual sensors each connect to the Internet to share data, analogous to how blogs share discourse. For example, WeatherUnderground.com allows "citizen scientists" to publish local weather conditions [13], while Sensorbase [7] and SensorWeb [21] provide frameworks for sharing sensor data, and for visualizing sensors and aggregates.

While individual sensors are sometimes of interest, the data becomes much more compelling when it is aggregated and processed. More than just visualizing individual sensors, we see a rich world where sensor values can be checked against each other, filtered, corrected, combined and divided, and indexed, not just by the sensor owners but potentially by anyone with access to the data.

*Republishing* is this process of transforming sensor data, and it can involve multiple steps and different users. As sensor sharing grows and republishing becomes more complex, tracking data back to its source is increasingly important. Understanding data flow is important to track the evolution of data, to discover duplicate or supplementary data sources, to give credit and confidence to data sources for indexing, to interpret data properly and reproduce results, to uncover the causes of anomalies, and to troubleshoot and improve the transformation process.

Tracking data transformation is well established in scientific workflow and databases; in this paper, we propose *data provenance for sensornet republishing* which allows users to locate where the sensor data come from and to further identify how they are processed. We propose a novel, tuple-level linking scheme that tracks sensornet data as it is processed and republished (Section 3). To support fine-grain, tuple-level tracking, we compare several compression schemes in Section 4, showing that our *incremental compression* scheme saves the storage up to 83%. We also describe how our approach can support user-centric access control. We have implemented our approach and are evaluating it in the context of weather monitoring in West Los Angeles (Section 5).

## 2 Related Work

***Sensornet/Internet Interaction:*** Several research efforts are exploring how sensornets and the Internet can interact [11, 18, 26, 3, 20]. We previously proposed an architecture to support sensornet sharing over the Internet [20]; this prior work is distinguished by the concept of *slogging*, the loose collaboration of many individually managed sensors; and by *republishing*, the idea that data will be processed and reprocessed in the Internet by different parties. In this paper we extend this prior work to explore how data provenance functions in the context of republishing.

***Scientific Workflow:*** Data provenance is important in the field of scientific workflow. Close to our work is Kepler [1], supporting access and analysis of distributed, heterogeneous scientific data [14]. Many techniques have been proposed to support data provenance in scientific workflow, although details vary depending on the scientific domains [22].

Data provenance in sensornets differs from scientific workflow in several ways. First, data is often static in scientific workflow, or treated as static snapshots. Sensornets and our republishing instead focus on live data feeds and streaming processing; so our system for provenance must explicitly record the status of the changing stream. Second, ownership and access are often handled out-of-band in scientific workflow. We instead assume many users and so must integrate easy-to-use disclosure management with our provenance system. Similarly, scientific workflow can often make assumptions about storage of data in a local or shared file system; we instead assume data is located anywhere on the Internet with a web services-like protocol. Lastly, computations in scientific workflow are often quite heavyweight, often involving large supercomputers and taking hours or even weeks per job. We see sensornet republishing as usually very lightweight, so the cost of providing data provenance must scale accordingly.

***Databases:*** Several researchers have considered provenance in the context of relational databases [9, 5, 6]. While we draw inspiration from their prior work, sensornets place several additional requirements on provenance. First, research efforts in databases mainly focus on capturing SQL-based transformations [24]; we instead wish to support transformations in republishing that include arbitrary, external programs not strictly described by SQL. We therefore capture the version of source code (or executable program) used in the transformation as part of our provenance scheme (Section 3.3). Second, database work in the area typically concentrates on provenance for a single database with centralized administrative control, while we assume a distributed environment with many data providers. In addition, the need to support distributed republishing motivates our plans for data disclosure management (Section 3.4). Lastly, databases support addition, deletion and update of information, but data provenance can be computed only for the current state of tables. Sensornets, on the other hand, rarely delete or update collected data, but constantly add new sensor data from live sensors and corresponding republishers. Assuming that data are inserted only, our provenance system can reconstruct old snapshots of the data by maintaining an explicit timestamp on provenance information. Therefore, we can trace the provenance of old results (Section 3.2).

***Provenance Model:*** Recently community of data provenance researchers has begun to standardize provenance models. The Open Provenance Model (OPM) is proposed for provenance information to be interoperable via a compatibility layer [16]. It represents a provenance information with three entities and five types of dependencies among them. Our provenance approach records two basic provenance relationships; the derivation between republished data to source data and transformation of republished data (Section 3.2). These are a subset of OPM relationships, so our model can easily be converted to OPM if desired.

Our work goes beyond OPM, though, by considering the access control and disclosure of data and provenance information. We support user authorization for data sources and transformations through provenance tracking (Section 3.4). Separate steps of provenance chain can therefore have different privacy and permissions controls, and the global view of a provenance chain may vary according to the user's permissions in different data repositories.

Currently we provide basic provenance queries to locate sources of data and transformations applied to the data. These are relatively primitive compared to ones used in the Provenance Challenges [15]. We are planning to provide more sophisticated provenance queries through a sensornet search engine.

## 3 Data Provenance in Sensornet Republishing

We describe the definition and goals of data provenance in sensornets and how we can achieve those goals.

Our work builds on our model of sensornet sharing [20]. We assume many users independently maintain sensors, each attached to the Internet (perhaps indirectly over a wireless edge network). Analogous to blog hosting sites, these sensors *slog*, publishing data to one of many centralized *sensor stores*.

Users can also schedule computation to run on other Internet-attached computers; these *republishers* read data from a sensor store, compute some result (such as aggregation, statistics, interpretation, etc.), and then publish the data back to some other sensor store. As a special case of republishers, sensor search engines index data. We show the *publishing* and *republishing* examples in Section 5.1.

### 3.1 Definition and Goals of Sensornet Provenance

Data provenance is well established in many scientific domains; however the definition of provenance varies depending on the scientific domain [22]. In sensornet republishing, we define data provenance as information of the source and the transformation applied to the source. We explore our approaches to data provenance below (Section 3.2). The core of our sensornet provenance is a linkage between republished data and its source. In this paper we use this link to locate the sensor data stored in our sensor stores, but in principle it can also track any resource provided as a web service (Section 4.1).

The ultimate goal of sensornet republishing is to allow users to process and share transformed data. Data provenance should allow any end-user to follow back to the original source data, observing each step of processing. As in scientific workflow, provenance is useful for validation, both to assist in debugging a republisher and to confirm faulty source data.

Sensornet republishing and slogging also need to be able to assign “credit” for data generation and processing [8]; we expect provenance to help with this process. Collaborative processing systems from SETI@home [4] to Wikipedia and blogging all benefit because data generators can observe who uses their data; we are seeking to recreate this ecosystem for sensor data [20].

More than just encouragement, we seek to reproduce a link structure in sensornet data that mirrors the link structure in the web, with the intent that we can harvest this link structure to identify high quality sensor data, much as PageRank exploits links in the web [19].

Finally, data provenance provides attribution information that is useful in slogging to inform data disclosure. We expand on access control in Section 3.4.

### 3.2 Approaches to Provenance for Sensornet Republishing

We describe our approaches for sensornet provenance: what and how to record for provenance (annotation vs. inversion; content vs. link), provenance granularity (tuple- vs. table-level), and timestamping to handle changing streams of data.

**Representation** There are two approaches to represent data provenance: annotation and inversion [22]. Annotation keeps the provenance information explicitly as metadata; on the other hand, inversion keeps the property of inverted transformation to find the source of derived data. The inversion is attractive if processing can be inverted to find the source of republished data, because it needs to keep only a single inversion function for the provenance. However our processing for sensor-data is arbitrary and cannot, in general, be inverted. We therefore choose *annotation* for sensornet provenance.

Given annotations, the annotation can either consist of a copy of source data, or a link to it and the transformation function. For small data items, copying source data to the republisher may be efficient. However, in some cases source data may be large, particularly for images, video, or audio. Thus a link to the source data is a good choice, because it is independent to the size of source data. In addition, over several steps, copying will accumulate many layers of data while linking is fixed in cost.

An additional advantages of linking is that a user following the provenance can discover not only the source data, but subsequent data generated later by the same source. It is also easy to trace back through multiple levels of republishings. This advantage is of particularly importance in streaming sensornet data where there is often new data, and where we wish to encourage repeated republishing.

**Granularity** How much detail of data provenance should be provided for sensor republishing? Coarse-grained provenance keeps one record per transformation or republishing. It is useful to figure out the overview of the processing, but is not enough for tracking data tuples. Instead we provide fine-grained provenance – each tuple has its provenance – which can pin down the source data used for each republished data. However, a problem of fine-grained provenance is storage. The storage of fine-grained provenance increases according to the number of data while that of coarse-grained provenance does not. We provide fine-grained provenance while its provenance storage is managed to be small with our compression scheme. The details of compression scheme is described in Section 4.2.

**Consistency** Sensornet data is often streaming, with new data arriving periodically. To truly reproduce a data transformation, data provenance must not only connect to a particular sensor, but also to a particular period of source data at that sensor.

Transformations are often expressed via user computations that are relative (for example, return the most recent five sensor readings). Provenance using this exact information would track a changing result as “most recent” changes when the sensor generates new data.

To manage changing data streams with potentially relative user queries, we embedded a timestamp with each data provenance record. This timestamp ties a query to a specific set of data at the source sensor store, regardless of when the link is later followed. Moreover, this timestamp approach supports data deletion. We soft-delete tuples by recording time of detection, allowing resolution of post-deletion references. The more details about the link are described in Section 4.1.

### 3.3 Tracking the Transformation

As we described in previous section, sensornet provenance allows users of the republished data to locate the source data for a transformation. Input data alone, however, does not fully define provenance. Data in our system is modified arbitrarily by some republisher—an arbitrary program running on some computer in the Internet.

To capture the republisher, we store transformation resource which includes a general description of republishing, source codes, and executable programs. We define transformation identifier to locate these transformation resources on the Internet (Section 4.1).

Our approach to tracking transformations has following benefits. First, it provides details transformations on every republished data. We store a simple identifier on every republished data as we do with the source data location; the specific transformation resource can be located by looking the identifier. Second, it is easy to distinguish data that are processed by different transformations. Because each transformation uses a unique identifier, the republished data can be grouped or selected according to transformations without looking the actual transformation resources.

### 3.4 Data Disclosure for Provenance

Our security model for sensor data allows the data generator to control data access. Data may be made publicly available, or access may be granted to individuals on a case-by-case basis [7]. This security model interacts with link-based data provenance because links may refer to data that a link-follower may not be able to access. To ease data disclosure, we integrate support for adjusting data disclosure into our data provenance system. When a user resolving a provenance encounters an access limitation, we generate a "letter of reference" about that user to pass to the data owner. This letter includes context about that user's activities, collaboration with other projects, other sharing activities, and how the user encountered the provider's data. He or she may then annotate or edit this information before sending it to the data owner who is responsible for controlling direct access to the source data. Our hope is that this information provides context to inform the owner of the data source, while the mechanism allows the requester to control what information they disclose.

## 4 Implementation

We have a prototype implementation of data provenance for sensornets. We use sensorbase.org [7] as our sensor store, and extend it to provide predecessor links. When a user creates a new table, we automatically create an additional column to store data provenance. We also have extended the sensorbase user interface to display data provenance; clicking on a predecessor link takes the user to the source data. APIs exist to extract this information and the transformation program. We use the existing sensorbase privacy model, and are in the process of automating support for data disclosure (Section 3.4).

We provide a PHP-based library that encapsulates this functionality and makes it easy for users to write republishers. We expect to provide bindings in other languages as well.

### 4.1 Predecessor Link

Our approach to data provenance in sensorbase provides exactly the information needed to track from derived data to its source data, potentially in another sensor store. As described in Section 3.1, we need the location of the source repository and table at that repository, the search used to retrieve the data from that table, and a timestamp to fix any temporarily relative portions of the query.

**Table 1.** Predecessor Link Template

| sb://<location of wsdl>?s=<service name>&a1=<arg 1>...an=<arg n>&t=<timestamp>&x=<xid> | |
|---|---|
| ***<location of wsdl>*** | This is the url of wsdl file which has the web service description.<br>(message format, available service and etc)<br>The actual url of wsdl is "http://<location of wsdl>" |
| ***<service name>*** | This indicates the service name to get the data. Currently we have a "getData" service to retrieve the data. |
| ***<arg 1>...<arg n>*** | These are arguments for the service. the "getData" service takes five arguments which are "attributes", "tables","condition", "from" and "delta". |
| ***<timestamp>*** | The timestamp of the link is created.('YYYY-MM-DD HH:MM:SS' UTC) |
| ***<xid>*** | The identifier of program doing transformation (a url)<br>on-site identifier format : `http://⟨sensorbase⟩/transformation_view.php?project=⟨no⟩&program=⟨name⟩&version=⟨version⟩` |

We encode this information into a URI-compatible link, the *predecessor link*, and use Web Services to access sensorbase [2]. The template of predecessor link is shown in Table 1. In a link, we directly encode the SQL-based search query, and any search parameters as arguments. We add a UTC-based timestamp corresponding to the query time, allowing us to replay a relative query later while producing the same result (Section 3.2). We add the user's ID and password at link resolution time, allowing the data provider to control access by requiring each user to authenticate separately (Section 3.4). Finally, in addition to the information locating the source data, we identify the transformation program (Section 3.3).

A sample predecessor link is: `sb://sensorbase.org/soap/sensorbase2.wsdl?s=getData&a1="datetime,temperature"&a2=p_97_temperature&a3=‘sensor id="sum-in"’&a4=0&a5=1&t="2008-02-24 12:00:00"&x="http://www.isi.edu/ilense/siss/tempread.html"` which locates temperature data used in a republishing. In this link, the user retrieves the *datetime* and *temperature* fields from the "sum-in" sensor. To deference this link, a user's system will retrieve the WSDL file (`http://sensorbase.org/soap/sensorbase2.wsdl`), and invoke the *getData* service with the five arguments (a1 through a5). The link also indicates when it was created and which program used the source data.

It is worth to note that transformation identifier is a URL which can represent the location of program, source code, or webpage describing the transformation. It is completely possible that identifier points off-site resource located shown in above sample link. However, we provide a on-site resource management for accessing the transformation resources on the sensorbase more efficiently. For example, an on-site identifier such as `http://sum.isi.edu/sb/transformation_view.php?project=97&transformation=tempread&version=0.4` indicates a program called *tempread* and its version is *0.4* which is used in project no *97*. The web interface shows not only the specific program used in the transformation but also other versions of that.

### 4.2 Incremental Compression

While self contained and easy to manage with existing tools, the links we described above are quite verbose and redundant. If used directly, link size would quickly overwhelm small sensor data and dominate storage consumption. We therefore employ *periodic incremental link compression* to provide simple link definition with reasonable storage cost. We quantify storage costs in Section 5.3 and consider compression approaches here.

Our goal in link compression is to take advantage of redundancy in repeated links. Often only a few parameters will vary, perhaps just query time. We considered at several alternatives: per-link compression, complete compression and periodic incremental compression. We chose periodic incremental compression to balance read and write cost.

A naive approach would be the *per-link compression*, where each link is passed through a conventional compression algorithm independently. While very simple to manage, this approach does not take advantage of the redundancy across links since that requires a compression dictionary that spans multiple links.

The *complete compression*, to exploit the redundancy across links, we maintain the compression dictionary over many links. An easy way of maintaining the pattern history is keeping it as an external file, although current dictionaries are optimized for run-time and not storage efficiency, so overall this approach consumes considerable fixed storage. Alternatively, we can rebuild the dictionary on-the-fly when it is needed. This approach requires additional run-time each link update, but it requires neither an additional storage nor maintenance of explicit history. The disadvantage with complete compression are the dictionary run-time cost is proportional to the number of saved data items, and loss of any item will invalidate the dictionary, requiring recomputation to rebuild all subsequent compressed links.

The system we adopt is *periodic incremental compression*—we avoid complete history by periodically checkpointing and restarting compression. This approach is robust to tuple loss and limits the computational cost of updates. We implemented periodic incremental compression with the widely-used LZW compression algorithm. Although subsequent compression algorithms (such as those in gzip and bzip2) improve performance somewhat, LZW provided good tradeoff between compression and ease of implementation. We evaluate our periodic incremental compression compared to other alternatives in Section 5.3

## 5 Evaluation

We next consider several ways to evaluate our provenance system. Ultimately, we would like to show that sensornet provenance is useful to users. Such demonstration requires an extended period of use; at this point we can only summarize our use of it in one application with three stages of republishing (Section 5.1). We then focus on two design questions: first we compare the storage costs of different provenance approaches (Section 5.2), then we look at tradeoffs in our compression algorithms (Section 5.3).

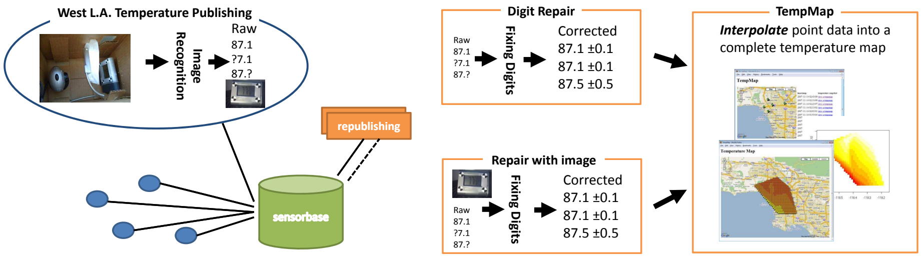


**Fig. 1.** West Los Angles Temperature Monitoring

### 5.1 Provenance Benefit

We explore sensornet provenance in the context of one application: collecting temperature data from a city-wide region. This application has several steps (Figure 1): first, we collect temperature from low-cost, off-the-shelf wireless sensors via computer-attached web cameras and publish both the image and the interpreted digits of temperature to a sensor store. Two different republishers can then examine this data and recover from common image interpretation errors, passing along either just the temperature digits (*digit repair*), or the digits and image (*repair with image*). Finally, we collect temperatures from a city neighborhood and interpolate a uniform grid of temperature with *TempMap*. We have been running this application with different numbers of sensors since March 2007, and currently have ten operational sensors.

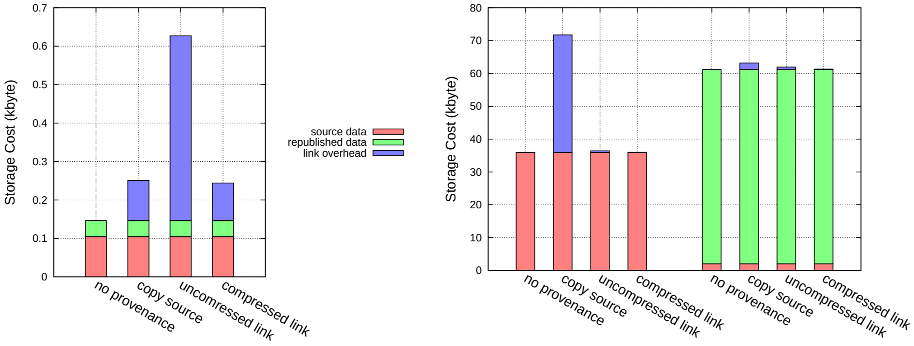
Full evaluation of the benefits of provenance will require long-term experience with this application. However our initial experience is promising; we have found provenance important for helping evaluate and debug problems with both forms of digit repair. We also have occasional problems with sensors going off-line; drilling down to the raw data is essential to debug these problems. Finally, we expect it to be useful when peering through the TempMap data. If an abnormality is found on the map, provenance helps follow through to the sensor that is mis-reporting.

### 5.2 Provenance Design for Sensornet Republishing

In Section 3.2 we discussed alternative implementations of provenance, choosing annotation with incrementally compressed links. Here we compare our choice against two alternatives: copying the source data and using uncompressed links, and without preserving provenance. Our goal is to understand how these alternatives affect storage overhead.

Figures 2(a) and 2(b) show the amount of storage consumed in the three republishing examples. We break the storage down in three categories: source data, republished data, and provenance overhead. We show the data on two separate graphs because the storage cost of digit repair is much less than that with images or TempMap.

These graphs show differences in the three stages of the application. Simple digit repair has small source and republished data, just the temperature value. Repair with image has much larger source data because it includes a digital picture of the sensor in



(a) Digit Repair

(b) Repair with image (left) and TempMap (right)

**Fig. 2.** Comparison of required storage in the republishing examples

addition to the interpreted value. Finally, TempMap generates a large, uniform array of interpolated temperatures from a sparser source set.

First, we observe that copying the source works well when source data is small (digit repair and TempMap), but it becomes quite expensive when the source is large (repair with image). Uncompressed links, on the other hand, do quite well when sensor data is large (repair with image and TempMap), but the provenance overhead is quite large compared to small source and republished data (digit repair)—making storage four times more than the basic data. Finally, we observe that compressed links do quite well when the data is large. When the data is small, the storage of compression link becomes smaller that that of copying the source even for small source data. The compressed link takes the smallest storage in all three examples.

As a final point, we selected tuple-level, fine-grained provenance. While we did not implement table-level, coarse-grain provenance, the overhead of a per-table link would be nothing with large tables of data. Approximating that cost with the “no provenance” bar, we can see that the cost of tuple-level provenance is dwarfed by the cost of data in the large-data cases, but roughly doubles the cost of storage with small data (digit repair). In that case, incremental compression (explored next) is essential.

### 5.3 Redundancy across Predecessor Links

As we have just shown, links storage can dominate storage costs when sensor data is small. We therefore compare several compression alternatives, including independent, per-link compression and incremental compression with different levels of checkpointing.

Figure 3 shows per-link storage for a series of 0 to 150 predecessor links with these cases and without compression. First, we observe that per-link compression halves storage because each link must build its own dictionary table. In this case we do not take advantage of redundancy across links.

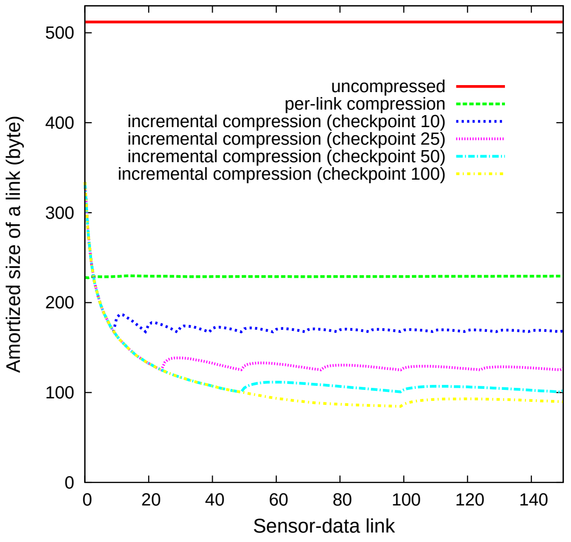


**Fig. 3.** Provenance storage with incremental compression

With incremental compression, we exploit compression dictionary across multiple links. For reasons described in Section 4.2, our incremental compression algorithm (LZW) is less efficient than per-link compression (gzip), so the first incremental link is less efficient. But the benefits of a shared dictionary quickly take over, making average links is the best with longer checkpoint periods. All incremental algorithms converge on different asymptotic efficiencies from only 90B/link with 100 links/checkpoint to 170B/link with 10 links/checkpoint. With less frequent checkpointing, read and write cost grows, therefore we need to balance efficiency with update speed. We selected 50 links/checkpoint as a reasonable trade-off, showing 80% savings in space which is slightly less than 100 links/checkpoint (83%).

## 6 Conclusion and Future Work

As data from sensornets are increasingly shared over the Internet, we expect that sensornet republishing will become an important means to share these abundant sensor data. In this paper, we showed how the principles of data provenance from scientific workflow and databases also apply to sensornets. We described our prototype system for data provenance in sensornet republishing and showed how it can assist debugging and serve as a source for sensornet search engines. Then, we evaluated our provenance system with republishing examples, showing that our link scheme with *incremental compression* save the storage up to 83%.

There are several areas of immediate future work, including implementation of provenance-aware data disclosure, improving user interface for provenance data and republishing APIs. Sensornet republishing APIs will make easy for users to write republishers with automated provenance management. We also plan to explore link structures among republished sensor-data to build a sensor search engine.

Data provenance already plays an important role in many scientific domains and data-oriented applications. We expect that our provenance system will also contribute to sharing and reuse in future sensor-network sharing.

## 7 Acknowledgment

This work is supported by National Science Foundation (NSF) grants CNS-0626702, Sensor-Internet Sharing and Search. Thanks to Sung Jin Kim and Junghoo Cho for helpful comments on our preliminary version.

## References

1. Kepler project. `http://kepler-project.org/`.
2. Sensorbase web service. `http://sensorbase.org/help/web_services.php`.
3. Karl Aberer, Manfred Hauswirth, and Ali Salehi. A middleware for fast and flexible sensor network deployment. In *VLDB*, pages 1199–1202, 2006.
4. David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 2002.
5. Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. An annotation management system for relational databases. In *vldb'2004: Proceedings of the Thirtieth international conference on very large data bases*, pages 900–911. VLDB Endowment, 2004.
6. Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 539–550, New York, NY, USA, 2006. ACM.
7. Kevin Chang, Nathan Yau, Mark Hansen, and Deborah Estrin. Sensorbase.org - a centralized repository to slog sensor network data, May 2006.
8. Dana Cuff, Mark Hansen, and Jerry Kang. Urban sensing: out of the woods. *Commun. ACM*, 51(3):24–33, 2008.
9. Yingwei Cui and Jennifer Widom. Lineage tracing for general data warehouse transformations. In *The VLDB Journal*, pages 471–480, 2001.
10. S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G-S. Ahn, and A. T. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 87–101, New York, NY, USA, 2007. ACM.
11. Phillip B. Gibbons, Brad Karp, Yan Ke, Suman Nath, and Srinivasan Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 02(4):22–33, 2003.
12. Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen K. Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. CarTel: A Distributed Mobile Sensor Computing System. In *4th ACM SenSys*, Boulder, CO, November 2006.
13. The Weather Underground Inc. Weather Underground. http://wunderground.com, 2006.
14. Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

15. Luc Moreau and etc. Special issue: The first provenance challenge. *Concurr. Comput. : Pract. Exper.*, 20(5):409–418, 2008.
16. Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. McGrath, Jim Myers, and Patrick Paulson. The open provenance model. 2007.
17. Suman Nath, Amol Deshpande, Yan Ke, Phillip B. Gibbons, Brad Karp, and Srinivasan Seshan. Irisnet: An architecture for internet-scale sensing services.
18. Suman Nath, Jie Liu, and Feng Zhao. Challenges in building a portal for sensors world-wide. In *First Workshop on World-Sensor-Web*, Boulder,CO, October 2006. ACM.
19. Lawrence Page, Sergy Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Unpublished manuscript, January 1998.
20. Sasank Reddy, Gong Chen, Brian Fulkerson, Sung Jin Kim, Unkyu Park, Nathan Yau, Junghoo Cho, and John Heidemann Mark Hansen. Sensor-internet share and search—enabling collaboration of citizen scientists. In *Proceedings of the ACM Workshop on Data Sharing and Interoperability on the World-wide Sensor Web*, pages 11–16, Cambridge, Mass., USA, April 2007. ACM.
21. A. Santanche, S. Nath, J. Liu, B. Priyantha, and F. Zhao. Senseweb: Browsing the physical world in real time. http://research.microsoft.com/nec/senseweb, 2006.
22. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, 2005.
23. Robert Szewczyk, Alan Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM.
24. Wang Chiew Tan. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.
25. Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.
26. Alec Woo. Demo abstract: A new embedded web services approach to wireless sensor networks. In *Proceedings of the Fourth ACM SenSys Conference*, page 347, Boulder, Colorado, USA, November 2006. ACM.