A Time-and-Value Centric Provenance Model and Architecture for Medical Event Streams

Marion Blount, John Davis, Archan Misra, Daby Sow, Min Wang[1]
IBM T. J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532, USA
{mlblount ,davisjs,archan,sowdaby,min}@us.ibm.com

ABSTRACT

Provenance becomes a critical requirement for healthcare IT infrastructures, especially when pervasive biomedical sensors act as a source of raw medical streams for large-scale, automated clinical decision support systems. Medical and legal requirements will make it obligatory for such systems to answer queries regarding the underlying data samples from which output alerts are derived, the IDs of the processing components used and the privileges of the individuals and software components accessing the medical data. Unfortunately, existing models of either annotation or process based provenance are designed for transaction-oriented systems and do not satisfy the unique requirements for systems processing high-volume, continuous medical streams. This paper proposes a simple, but useful, hybrid provenance model called Time-Value Centric (TVC) provenance. In this model, each entry in the output data stream (e.g., an output alert) is linked to some specific time windows of incoming data samples that contribute to the generation of the particular output entry, with the time-dependence potentially varying with the data values. An initial design of the provenance storage and querying architecture for this TVC model is also presented.

Categories and Subject Descriptors

J.3 [**Life and Medical Sciences**]: Medical Information Systems

General Terms

Algorithms, Management, Security

Keywords

Provenance, data streams, medical sensors.

1. INTRODUCTION

Faced with a rising elderly population, governments and healthcare providers in many developed countries are investigating the development of a large-scale, distributed infrastructure for remote medical monitoring. A particularly compelling vision for IT-enhanced healthcare delivery focuses on the use of body-worn pervasive sensors that provide multiple streams of medical data (such as heart rate, SpO2 saturation, ECG signals and muscular activity levels). The availability of such rich, personal medical history can provide compelling benefits, such as proactive anomaly detection, drug side-effect monitoring and trend analysis of lifestyle activity patterns, especially for patients with chronic diseases such as congestive heart failure, high blood pressure and Alzheimer's. Prior research on remote monitoring has largely focused on the *client* side, typically developing various components for a pervasive relay device that acts as a personal hub [1] in a 3-tier remote monitoring framework [1,2]. Research on the *server-side* has been largely absent, with research prototypes treating the server merely as a large data-store.

As part of our recently-launched Century project, we are investigating the data management components and capabilities required at the backend to make this vision of pervasive healthcare a reality. Century is intended as a unified repository of biomedical data streams from a large patient population—however, our effort is driven by the observation that the backend must be *much more than a passive repository; it must be capable of* **analyzing**, **processing** *and* **transforming** *the raw medical sensor streams.* In particular, automated stream analysis components will prove critical to the development of sensor-driven diagnostic and clinical support systems, as otherwise medical professionals will simply be swamped by the high volume of raw sensor data generated by a large patient population.

The issue of *data provenance support* is an important one for systems such as Century, as the medical domain often makes it mandatory to support functions such as "dependency analysis" or "data replay" needed to satisfy auditing and other regulatory requirements. In particular, various Century stakeholders (e.g., doctors, family members) should be able to inspect, often at a much later date, Century's internal "information flow", to either manually validate the processing logic within Century or to perform fault-diagnosis in the case of anomalies. Broadly speaking, "provenance" in our context implies that the CENTURY infrastructure must not only filter and process high-volume data streams, but also store sufficient metadata to subsequently answer questions such as *a)* "What low-level sensor data contributed to (or did not) to this automated alert", *b)* "which stream processing components did this alert depend on?" or *c)* "what sensor was used to obtain this raw medical reading?".

A modest amount of research on provenance mechanisms has been conducted, largely for scientific workflow systems (e.g.,

[1]Authors listed in alphabetical order

Karma[3], PreServ[4]), low-level operating systems or file systems (e.g., PASS [5]) or Web-Services oriented workflows (e.g., PASOA [6]). However, all of these provenance approaches apply to transactional systems and do not satisfy the unique requirements and constraints of *stream-based* systems. In particular, we argue that the most widely-used "annotation" model of provenance, where the requisite metadata is associated with each individual data element, is inappropriate for stream-based systems, as it fails to exploit the unique dependency characteristics across data streams and results in prohibitive storage overhead. Based on this observation, we introduce a *Time-Value-Centric* (**TVC**) model for stream-oriented provenance in this paper. The key driver of this model is the observation that, in many instances, the provenance metadata is invariant over multiple successive data elements of a stream. It is thus best to collect the provenance data at the stream segment level (rather than at the data element level).

1.1 Contributions of This Paper

This paper makes the following important contributions:

- It establishes the distinctive features of the provenance problem for stream-oriented middleware systems.
- It introduces the TVC model as a simple, yet widely applicable, and efficient primitive for expressing lineage dependencies among medical events.
- It provides an initial specification of the way in which the provenance relationships between various medical events and streams may be stored and retrieved.

2. CENTURY BACKGROUND AND NEED FOR MEDICAL PROVENANCE

The Century project [8] is building a distributed and shared middleware platform that ingests very large numbers of event streams, makes "sense" of this input data and distributes the resulting knowledge to a diverse set of stakeholders. This set includes patients and their extended families, medical professionals, hospitals and insurance companies, among others. These stakeholders might be interested in different perspectives on the data collected. For example, nurses may monitor the wellbeing of specific patients, while a CDC analyst might mine the data streams, aggregated across a large patient population, to predict disease outbreaks. However, all stakeholders face a common risk: of being inundated with data if the raw, unprocessed, biomedical event streams were directed towards them. To address this central problem of data overload, Century's core component is a stream processing infrastructure capable of accepting and analyzing very large volumes of input data streams.

Figure 1 shows a high level view of Century. At its heart is the Stream Processing Core (SPC) [9] platform, which offers support for the development and deployment of scalable stream analysis applications. Unlike most prior stream processing technologies, SPC supports both relational and user defined stream processing operators. Figure 1 also includes an example of a Congestive Heart Failure (CHF) analysis graph containing a combination of stream operators (that we refer to as Processing Elements or **PEs**). Broadly speaking, the CHF application takes 3 types of sensor inputs (ECG, blood pressure and weight) for an individual over a specific time interval and periodically generates an "alert" (either "normal" or "abnormal") depending on the temporal evolution of the streams. Century adds several additional components to SPC. The first component is an event store where high-rate biomedical event streams are persisted, potentially for very long time periods (e.g., 6 years) to satisfy specific regulatory requirement. The event store allows authorized external applications to query, update or even delete these records. The provenance subsystem is another major new component of Century, as well as the focus of this paper. The provenance subsystem allows allow stakeholders to query for the origins of raw and processed event streams.

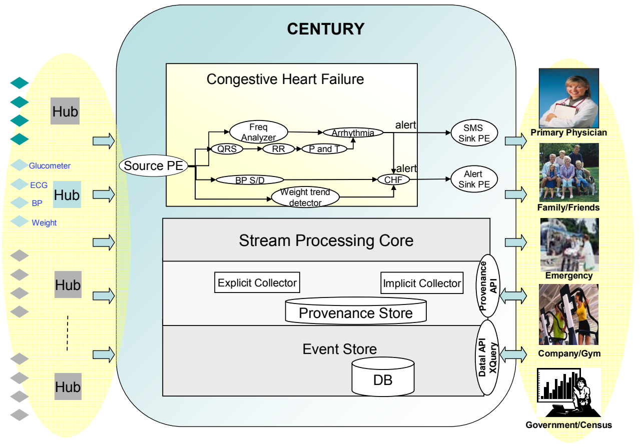


Figure 1. Century Components (The stream processing graph illustrates the CHF monitoring application)

3. PRIOR WORK AND PROVENANCE CHALLENGES FOR MEDICAL STREAMS

The bulk of prior research on provenance focuses on transactional systems, which typically involve a discrete (and relatively low-rate) request-response style of interaction. Examples of such transaction workflows include:

- *Scientific and Web-Services Workflows*: Systems such as Karma and PreServ are designed to capture interactions among various components for data-driven scientific workflows, such as atmospheric sensing and genomic computing. These systems focus purely on process provenance, i.e., they store the history of inter-component interactions (e.g., SOAP invocations), rather than the actual transformation of the datasets.
- *File Systems and Databases*: Approaches such as PASS and LinFS [7] are typically annotation-based—they associate provenance metadata with individual data items, such as files or DB records. As an example, PASS automatically stores the modification history of files, including information on the calling application, the file descriptor state, etc.

In contrast, Century needs to support provenance for high-volume data streams. Here, the term "stream" implies a logically-related set of data events possessing a well-defined temporal sequence. For example, continuous ECG readings from a cardiac monitor or periodic accelerometer readings (for movement monitoring) from a single individual sensor can be considered to be two independent data streams. The limited work in stream-based provenance includes [10], which focuses on identifying and storing the dependency relations among streams (by encoding, as a tree, the IDs of ancestor streams of a derived stream), rather than the data dependencies for various elements of the stream.

Table 1: Data Rates for Representative Medical Sensors

Type of Sensor Device	Bits/ sensor sample	Channels/ device	Typical reporting frequency	Data rate (KB/day)
SpO2	3000	1	3 Hz	94,922
EKG (cardiac)	12	6	256 Hz	194,400
Accelero-meter	64	3	100 Hz	202,500
EEG (brain)	12	12	256 Hz	388,800
EMG (muscle)	12	6	1024 Hz	777,600

We believe that provenance support in Century must consider the following characteristics:

a) *High Data Volume:* Many of the medical applications that we consider involve the transformation and storage of data from sensors with very high data rates. (As an illustration, Table 1 shows the basic data rates associated with a variety of medical sensors.) Given such a high data rate, the “annotation” based model of provenance, where the metadata is appended to the data itself, is not particularly efficient from a storage perspective.

b) *High Throughput Support:* The stream-processing middleware must also be *computationally-efficient* to support the high stream event rate. For high volume data streams, the process of per-data item annotation is likely to result in significant processing overhead (as provenance requires the generation of per-packet information, such as unique identifiers and timestamps), as well as incur high storage latency [11], and could thus impact the system’s throughput. Accordingly, we require a provenance solution that is *reactive* (i.e., the computation of provenance is invoked only when appropriate attributes of interest change).

c) *Statefulness of Stream Operators:* Many transformations of interest on raw medical data occur over “time windows” of raw sensor data. In other words, the operators are *stateful*, with the transformation logic for a particular data element implicitly affected by past input elements or other external “context”. An example of such a stateful operator is an “activity detector that detects if an individual’s pedometer reading (number of steps walked) on a particular day falls below 70% of the average footsteps/day over the past week”. In contrast, the provenance for a user accessing a scientific library via a Web service would consist of parameters such as the requester’s user ID or the library’s version number—this provenance data is independent of past or future transactions.

The limitations of existing provenance models become clear on reviewing these three vital features of Century. Annotation-based approaches are particularly poorly suited to our high-volume, stream-oriented environment due to the high storage and processing overheads. Process-oriented provenance models, on the other hand, are inadequate for many clinical decision-support applications, especially if healthcare professionals desire to visually inspect the data dependencies and derive their own conclusions (and perhaps, override the recommendations of the automated system). The observations above lead to two key new requirements:

a) A *hybrid model* of provenance is required. Such a hybrid model should ideally combine the low storage and processing overhead of stream-based provenance with the higher descriptive capabilities of data-oriented annotation models.

b) The provenance system should be consciously architected to support a query-vs-store tradeoff, in effect trading off higher reconstruction complexity (during the infrequent provenance queries) for more efficient metadata storage and generation.

In the next section, we introduce the TVC model as the first, simple yet useful, instance of such a hybrid provenance model.

4. THE TIME AND VALUE CENTRIC MODEL

The temporal aspect of the TVC model draws on the fact that many of the transformations of sensor data streams implicitly involve the use of input data samples that lie within a finite time-window (or sets of such windows). Mathematically, a simple model could track the value of a data element belonging to a stream S_i (output by a processing element PE_i) as a function of specific samples (from input streams) that occur within designated time intervals in the past. In other words, for any element $e_i(t) \in S_i$, with a creation timestamp of t, the following relationship holds:

$$e_i(t) \Leftarrow \bigcup_{j:S_j \text{ is input to } PE_i} \bigcup_{k=1}^{L_j} \{e : e \in S_j(t - start_{jk}, t - end_{jk})\} \quad (1)$$

where L_j is the number of distinct disjoint ‘time intervals’ which define the values of S_j on which $e_i(t)$ depends, and $start_{jk}$ and end_{jk} define the boundaries of these intervals. For the simple cases where end_{jk}=0 and $L=1$, we may express the time dependency of an input stream S_j in terms of a single interval value Δ_j. Moreover, these dependencies are *time-invariant*—in other words, the dependence of a derived medical event to other medical samples or raw sensor data can be expressed succinctly and completely independently of the specific timestamp or ID of each sample. Moreover, as long as each individual data item contains its timestamp and a stream ID, the dependent set of data elements can always be derived.

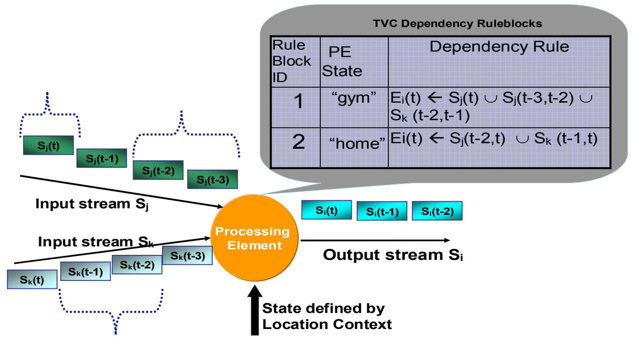
4.1 State-Based Changes in Temporal Dependencies

In several practical cases of interest, the temporal dependency may be time-invariant, but still exhibit a dependency on some data value. In general, the “data” value can refer to an element belonging to one of the input streams or to some internal “state” (that indirectly modifies) the processing logic of a PE. As an example of such state-dependent processing, consider the following rule (where *HR* indicates values from a heart rate sensor and *loc* refers to the user's location context:

IF *loc= "gym", then generate ALERT iff (AVG(HR(t-**10**,t)>140);*

ELSE *generate ALERT iff (AVG(HR(**t-30,t**)> 90).*

In this case, it is easy to see that the temporal dependence term Δ depends on the patient's location, with a more aggressive policy (smaller time window) being used when the patient is exercising in the gym. To capture and reconstruct this sort of provenance dependence, we need to extend the provenance model—as a generalization, we can associate multiple "dependency ruleblocks" (the logic of Equation 1) with each PE, with each block corresponding to a distinct state (or range of specific values). To reconstruct the data provenance, the output data sample must not only possess a timestamp and stream ID, but also an index to the specific ruleblock (from the set of ruleblocks of the creating PE) associated with it. An example of the logical metadata associated with an arbitrary PE is shown in Figure 2.


TVC Dependency Ruleblocks

Rule Block ID	PE State	Dependency Rule
1	"gym"	$E_i(t) \leftarrow S_j(t) \cup S_j(t-3,t-2) \cup S_k(t-2,t-1)$
2	"home"	$Ei(t) \leftarrow S_j(t-2,t) \cup S_k(t-1,t)$

Figure 2. The PE-based model for expressing TVC provenance. The curly braces indicate the input dependencies on $S_i(t)$ for the state "loc=gym".

4.2 The Query for the TVC Model

Although relatively simple, we believe that the TVC model provides adequate expressiveness and can answer the following important provenance query:

Given a specific event (or data sample), show me the set of data samples from the input streams on which this output event may depend.

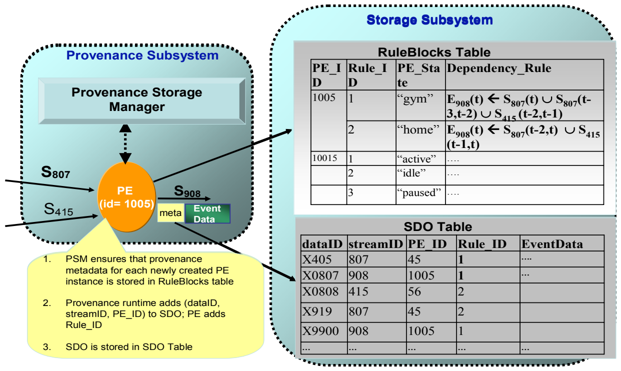
This can clearly be applied recursively, to reconstruct an expanding data-dependency tree). As a CHF-based example, a nurse receiving a CHF alert may inspect the relevant subsets of ECG, weight and pressure data that generated that alert.

5. ARCHITECTURE FOR PROVENANCE STORAGE AND QUERY RESOLUTION

To implement the TVC model in any stream processing system, we require the following features:

a) Each PE must be associated with a set of rules (which we call a *RuleBlock*), where each rule expresses (in some format) a specific temporal dependence on input streams.

b) Each individual data element must be associated with a specific stream, with each stream possessing a unique stream ID, and have a well-defined timestamp (to help establish appropriate temporal relationships).



Provenance Subsystem
Provenance Storage Manager
S_{807}
S_{415}
PE (id= 1005)
S_{908}
meta
Event Data
1. PSM ensures that provenance metadata for each newly created PE instance is stored in RuleBlocks table
2. Provenance runtime adds (dataID, streamID, PE_ID) to SDO; PE adds Rule_ID
3. SDO is stored in SDO Table
Storage Subsystem

RuleBlocks Table

PE_ID	Rule_ID	PE_State	Dependency_Rule
1005	1	"gym"	$E_{908}(t) \leftarrow S_{807}(t) \cup S_{807}(t-3,t-2) \cup S_{415}(t-2,t-1)$
	2	"home"	$E_{908}(t) \leftarrow S_{807}(t-2,t) \cup S_{415}(t-1,t)$
10015	1	"active"	….
	2	"idle"	….
	3	"paused"	….

SDO Table

dataID	streamID	PE_ID	Rule_ID	EventData
X405	807	45	**1**	….
X0807	908	1005	**1**	…
X0808	415	56	2	
X919	807	45	2	
X9900	908	1005	1	
…	…	…	…	…

Figure 3. The Provenance sub-system and associated storage for the TVC model.

Figure 3 shows the functional design of the provenance storage sub-system (modeled as a set of database tables) for the TVC model. In our initial design, we assume that the set of dependency rules is *manually-specified*: in other words, the author of a PE is responsible for explicitly specifying the IDs and logic of each rule block. In addition, the Provenance Storage Manager (PSM) runtime component ensures that every data element output by the PE is "marked" with a system-generated timestamp and the ID of the PE. Figure 3 shows how this resulting metadata is stored in the event database as part of the *Streaming Data Object* (**SDO**) table. In addition, every instance of a PE creates a unique entry in the *RuleBlocks* table, containing the relevant set of rules.

5.1 Logical PEs and Instances: Optimizing Through Rule Templates

A careful inspection of Figure 3 shows that the *RuleBlock* entry associated with a particular PE needs to be unique for each "physical" running instance of a PE. For example, if both "Joe" and "Bob" are being monitored for CHF using the same "logical" QRS detector, they would each have a separate entry in the *RuleBlocks* table, with the *Dependency_Rule* field containing the user-specific stream bindings. This additional overhead can however, be avoided by the use of "template" ruleblocks, which are associated with a logical PE. This is based on the observation that multiple instances of the same PE employ essentially identical logic, differing only in the actual stream bindings. Accordingly, in an optimized TVC architecture, the temporal "ruleblocks" are associated with each unique logical PE. Each instance of a PE creates a an entry in a separate *StreamMapping* table, containing the IDs of the actual input streams and an index into the *RuleBlocks* table for the corresponding physical PE.

5.2 Satisfying Data Dependency Queries

Given the provenance subsystem architecture presented above, we can now answer the query "*Show me all data samples on which* *e(t) depends*" for any arbitrary data event (e.g., CHF alert) stored in the event database (the SDO table). Figure 4 illustrates the pseudocode for the data dependency query. Intuitively speaking, the process consists of first retrieving the "rule dependency" block associated with the event being queried, and then using the *StreamMapping* structure (for the corresponding PE instance) to identify the template in the appropriate logical rule-block. Subsequently, the resolution algorithm will determine the precise input stream IDs and issue a SELECT query to retrieve the data samples possessing the relevant stream ID and having a timestamp within the corresponding time window.

In the example pseudocode in Figure 4, we assume that the ruleblock template is expressed using the algebraic notation illustrated in Figure 3. In reality, there are many options for encoding the dependency rule. To help component developers easily specify such provenance rules, we are investigating the use of specific *ontologies* to help capture the dependency relationships. We believe the development of ontologies to provide support for TVC-based dependencies constitutes a key research issue for building practical healthcare decision-support systems.

```
RetrieveDependentData (Event e){
1. ts= e.Timestamp; streamID= e.streamID;
2. streamMap= lookupStreamMap(e.processID);   //retrieve
        streamMap of PE instance that created event e
3. ruleBlock= getRule(streamMap.logicalPE, e.ruleIndex);   //retrieve
the  logical ruleblock
4. {inputStream[], startInterval[],endInterval[]}= resolveDependency
(ruleblock, streamMap.streamIDs[]);   //this is the method that creates
the specific stream dependencies for this input sample
5. for (j=1; j <  inputStream.size(); j++); {
     5.1 Depend_j= SELECT f FROM SDO WHERE (ts-
startInterval[j] < f.Timestamp < ts-endInterval)  AND
(f.streamID== inputStream[j]);
      5.2 SetofDependentData = Depend_j;
   }
6. return SetofDependentData;
}
```

Figure 4. Algorithm for Resolving a Data Dependency Provenance Query

6. SIMPLE OVERHEAD ANALYSIS OF TVC SCHEMES

In this section, we present a relatively simple, but representative, numerical analysis of the storage overheads of annotation-based provenance vs. our TVC-based provenance approach. Table 2 lists the various mathematical symbols used in our analysis, as well as the typical values we associated with our CHF sample application.

For the annotation approach, each output "alert" must carry as metadata the IDs of all the input SDOs that spawned this alert—in general, this implies a total of $\Delta_i * \rho_i^D * S_{sdo}$ bits from input stream S_j. Accordingly, the additional "fractional overhead" (**FO**) of provenance metadata FO_{annot} in this model (as a function of the total time period of monitoring T) is given by:

$$FO_{annot} = \frac{N\left\{\overbrace{\rho_{alert} T S_{alert}}^{alert\ ....overhead} + \overbrace{\sum_{i=1}^{S} \rho_i^D T S_i^D}^{SDO\ ....storage\ ...overhead}\right\}}{N\left(\sum_{i=1}^{S} \rho_i^D T S_i^D\right)} - 1 \qquad (2)$$

For the basic TVC model (where the dependencies are enumerated on a 'per-instance' basis), there are significant savings in per-alert overhead; however, this comes at the cost of slightly higher (1+κ) overhead to store stream-specific metadata (e.g., the stream IDs) in the raw SDOs, as well as the per-PE metadata needed to store the temporal dependence rules. In this case, the fractional overhead FO_{TVC} as a function of T is given by:

$$FO_{TVC} = \frac{N\left\{\overbrace{N_{pe} C_s R_s}^{RuleBlocks\ ....overhead} + \overbrace{(1+\kappa)}^{TVC\ ..overhead} \sum_{i=1}^{S} \rho_i^D T S_i^D\right\}}{N\left(\sum_{i=1}^{S} \rho_i^D T S_i^D\right)} - 1 \qquad (3)$$

If users are allowed be clustered into groups in which provenance dependency rules are invariant and can be shared (i.e, PEs are instantiated per groups of users), the rule block overhead can be reduced by a factor $0 < \beta < 1$ corresponding to the average group size 1/β. In this case, the fractional overhead FO_{TVCG} becomes:

$$FO_{TVCG} = \frac{N\left\{\overbrace{\beta N_{pe} C_s R_s}^{RuleBlocks\ ....overhead} + \overbrace{(1+\kappa)}^{TVC\ ..overhead} \sum_{i=1}^{S} \rho_i^D T S_i^D\right\}}{N\left(\sum_{i=1}^{S} \rho_i^D T S_i^D\right)} - 1 \qquad (4)$$

Finally, for the optimized TVC model (where the dependency is captured by instance-independent templates), we eliminate the per-user group overhead of metadata storage, but incur a lower overhead of maintaining the stream-to-user group mappings. As $N\beta$ user groups, each with S streams, require, per group, $S*log(SN\beta)$ bits to store the stream mappings, the overhead $FO_{TVC\text{-}op}$ is:

$$FO_{TVC-Opt} = \frac{\overbrace{N_{pe} C_s R_s}^{RuleBlock.Template} + N\left\{\beta S \overbrace{\log(N\beta S)}^{Stream...overhead} + (1+\kappa)\sum_{i=1}^{S} \rho_i^D T S_i^D\right\}}{N\left(\sum_{i=1}^{S} \rho_i^D T S_i^D\right)} - 1 \qquad (5)$$

Table 2: Mathematical Symbols for Provenance Analysis

Symbol	**Meaning**	**Value Used in Analysis**
N, S	No. of users being monitored and number of input sensor streams/user	{50, 50,000}; 3 (ECG, blood pressure (BP) and weight (BP))
β	Average fraction of users in groups	0.1
N_{pe}	No. of PEs in the application "graph"	=11 for the CHF application
r_i	Bits/sec of 'raw' data from sensor i. We assume 256Hz, 12bit, 3 channels for ECG, 3 samples/day@3bytes/sample for BP, and 3 samples/day@2bytes/sample for WT.	9216 bps (ECG), $8.33*10^{-4}$ bps (BP), $5.5*10^{-4}$ bps (WT)
ρ^D_i	Samples (SDOs)/sec for sensor i. We assume that ECG is chunked in 1-sec blocks, while HR and BP associate each raw sample with a new	1 for ECG, $3.44*10^{-4}$ for HR and BP

	"chunk".	
Δ_i	No. of seconds of past values of S_i that contribute to alert	3600 sec for ECG, 6.04.10^5 sec (1 week) for BP and HR.
ρ_{alert}	Avg. No. of alerts/user per sec. (We assume an alert (normal or abnormal) is generated every hour).	1/3600.
S_i^D	Size of an SDO for input stream S_i (in bits). We assume each such SDO stores a "chunk" of raw data, and requires an additional 10 bytes of "overhead".	$r_i * \left(1/\rho_i^D + 80 \right)$
S_{alert}	Size of an "alert" SDO under the annotation model (bits). $\left(80 + \sum_{i=1}^{S} \Delta_i * \rho_i^D * S_{id}^{SDO} \right)$	An 'annotated' alert contain the IDs of all the "input" SDOs on which it depends+ 10 byte overhead
C_s	No. of distinct "states" (values) for individual PE	40
R_s	No. of bits/state to represent the formula for temporal dependence	80000 ("rules" can be expressed by 10KB of XML data)
κ	Overhead/data bit added in TVC model. (For the dominant ECG data, each data SDO is ~1162 bytes, and TVC metadata is ~20 bytes/SDO)	2%.
S_{id}^{SDO}	The no. of bits used to represent a unique SDO.	64 (this is enough to index ~10^{20} distinct "chunks" (SDOs)
T	Time period which a CHF application runs (we assume long-term continuous monitoring).	Varied between 7days-6months

Figure 5 plots the relative storage overhead of these approaches as the time (T) for the CHF monitoring application is varied. From the plots, we see that either of the TVC-based approaches results in a sharp reduction of the provenance-related overhead (which is almost 100% under the annotation-based approach for our application). Moreover, the optimized TVC approach (using templates associated with logical PEs) is particularly efficient when the number of patients is large (e.g., 50,000)—in this case, it is able to support provenance with very low overhead (<5%). Consequently, this model is *especially attractive for long-term monitoring applications over large patient populations.*

7. CONCLUSIONS

In this paper, we have introduced *stream provenance support* as an important new functional requirement for healthcare systems employing sensor-based remote monitoring. The proposed TVC model provides a simple, expressive and very-low overhead construct for capturing process-level dependencies. We are currently working to implement this model within the SPC platform, by building system-level support for automatic collection and storage of stream-level mappings as PE graphs are instantiated or destroyed. We believe that the design of ontologies by which authors of PEs can express the associated TVC dependencies is an important topic of research. Finally, provenance in the medical domain gives rise to important privacy problems—in particular, Century will have to accommodate the fact that privacy preferences for provenance metadata might be distinct from similar policies on the medical data itself.

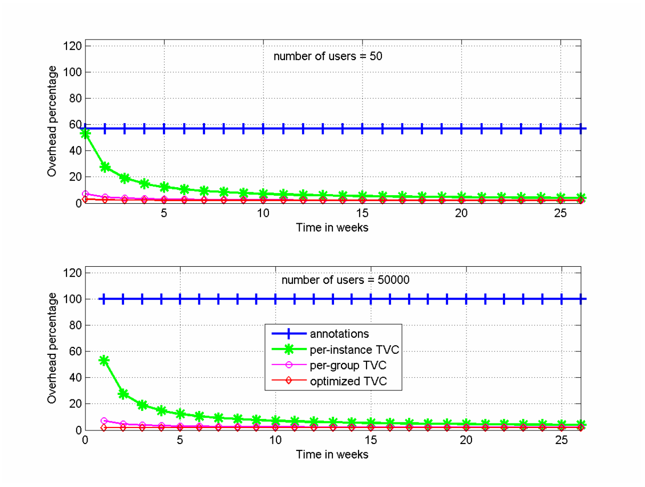


Figure 5: Comparative Provenance-based Overheads

8. REFERENCES

[1] Husemann, D., Narayanaswami, C., and Nidd, M. Personal Mobile Hub. *8th International Symposium on Wearable Computers (ISWC 2004)*, November 2004.

[2] Lubrin, E., Lawrence, E., and Navarro, K.F. MoteCare: An Adaptive Smart BAN Health Monitoring System. In *Proceedings of the 24th IASTED international conference on Biomedical Engineering*, February 2006.

[3] Simmhan, Y.L., Plale, B. and Gannon, D. Performance Evaluation of the Karma Provenance Framework for Scientific Workflows. *International Provenance and Annotation Workshop (IPAW)*, May 2006.

[4] Groth, P., Luck, M., and Moreau, L. A protocol for recording provenance in service-oriented grids. In *Proceedings of of the 8th International Conference on Principles of Distributed Systems (OPODIS'04),* December 2004.

[5] Muniswamy-Reddy, K., Holland, D., Braun U., and Seltzer, M. Provenance-Aware Storage Systems. In *Proceedings of the 2006 USENIX Annual Technical Conference,* June 2006.

[6] Chen, L., et al. A proof of concept: Provenance in a Service Oriented Architecture. In *Proceedings of the Fourth All Hands Meeting (AHM)*, September 2005.

[7] Lineage File System, http://crypto.stanford.edu/~cao/lineage. html

[8] M. Blount, et al. Century: Automated Aspects of Patient Care. *Under submission*.

[9] Amini, L., et al. SPC: A Distributed, Scalable Platform for Data Mining. *SIGKDD 2006 Workshop on Data Mining Standards, Services, and Platforms*, August 2006.

[10] Vijayakumar, N., and Plale, B. Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering. *International Provenance and Annotation Workshop (IPAW),* May 2006.

[11] Nicola M., and John. J. XML parsing: A threat to database performance. *12th International Conference on Information and Knowledge Managment*, 2003.