MediAlly: A Provenance-Aware Remote Health Monitoring Middleware

Atanu Roy Chowdhury*, Ben Falchuk†, Archan Misra†
*School of Engineering and Applied Sciences,Harvard University
†Advanced Technology Solutions, Telcordia Technologies
atanurc@post.harvard.edu, bfalchuk@research.telcordia.com, archan@research.telcordia.com

Abstract—**This paper presents MediAlly, a middleware for supporting energy-efficient, long-term remote health monitoring. Data is collected using physiological sensors and transported back to the middleware using a smart phone. The key to MediAlly's energy efficient operations lies in the adoption of an Activity Triggered Deep Monitoring (ATDM) paradigm, where data collection episodes are triggered only when the subject is determined to possess a specified context. MediAlly supports the on-demand collection of contextual provenance using a novel low-overhead provenance collection sub-system. The behaviour of this sub-system is configured using an application-defined context composition graph. The resulting provenance stream provides valuable insight while interpreting the 'episodic' sensor data streams. The paper also describes our prototype implementation of MediAlly using commercially available devices.**

I. Introduction

Remote health monitoring services promise significant improvements in healthcare delivery and chronic disease management by providing new and detailed insights into an individual's biomedical data or activity patterns. Such remote monitoring and automated medical analytics are becoming increasingly plausible, thanks to recent developments in miniaturized physiological sensors and low-power radios, powerful handheld computing devices and almost-ubiquitous wireless connectivity. A logical three-tier architecture [2], comprising a server for backend integration, a cellular phone/handheld device based personal gateway and a body-worn set of sensors, seems most suited to support such a remote health monitoring service.

The act of monitoring, processing and transporting the medical sensor streams is associated with a non-trivial energy cost, that manifests itself as a resource bottleneck when we employ battery powered devices [5]. In untethered deployments, there is thus a clear tradeoff between the system lifetime and the rate of data generation. For example, [12] supports longer deployment periods for low data rate sensors, whereas [3] and [4] support higher data rates for shorter durations. To address this tradeoff, we suggest the *Activity Triggered Deep Monitoring* (ATDM) paradigm (introduced in [6]), whereby high fidelity sensor data streams are collected and transported only when the subject satisfies a set of contextual constraints (e.g., we collect ECG data only when the subject is exercising, where the *exercise* context is defined by the subject's medical practitioner.). By employing a context-triggered monitoring approach, ATDM produces streams of health sensor data that are episodic and have varying granularity and duration.

This paper describes **MediAlly**, a remote health monitoring service that conforms to the ATDM paradigm and supports a low overhead sub-system for collecting, storing and replaying the *contextual provenance* associated with the monitored sensor data streams. Here, provenance refers to MediAlly's ability to collect, store and (at a future time) reconstruct the subject's contextual states that acted as ATDM triggers. Such reconstructed context provides invaluable insight for the interpretation of the episodically-captured data streams. For example, a doctor would find it useful to know if a data stream corresponding to "30 minutes of elevated heart rate" , recorded a month ago, occurred while the subject was actually exercising at a healthclub or seated at his home. For practical considerations, the MediAlly service architecture is designed to potentially interwork with third party personal health repositories (PHR), such as Google Health™or Microsoft HealthVault™, by *a)* logically separating the 'health' data streams from the 'context' metadata stream, and *b)* providing programmatic APIs to combine these streams as and when necessary (as illustrated by the overlaid explanatory context in Figure 1). Thus the key contributions of the paper are:

- We present a functional architecture for explicit collection and reconstruction of the contextual metadata that supplies the necessary provenance to the captured health data and describe a simple programming model that allows application developers to control the granularity of the captured contextual metadata.
- We develop a new graph-based model for efficiently representing, capturing and reconstructing the time-varying contextual metadata, at different levels of granularity and at diferent levels of "context composition". The model employs a novel *lazy capture* principle to significantly reduce overheads, while preserving the accuracy of provenance reconstruction.

The rest of the paper is organized as follows. Section II elucidates the ATDM paradigm and the role of contextual metadata in providing provenance for the collected health data. Section III explains the MediAlly functional architecture and develops the programming model. Section IV elaborates on our low overhead provenance model, that provides capture and replay functionalities. Section V describes our prototype implementation using a Nokia N95 cellphone and discusses some of the open issues that need further investigation. We present the related work in section VI before concluding in section VII.

II. Harnessing the ATDM paradigm

Our analysis reveals that continuous transmission of high data rate medical data streams can often impose impractical traffic loads on existing wireless PAN technologies likely to be used between the sensors and the mobile gateway. For example, low power IEEE 802.15.4 radios have maximum bandwidth of 250 kbps; while Bluetooth may provide greater data rates ($<$ 1Mbps), it has significantly higher higher energy costs. Moreover, the *continuous* transmission of moderately high-data rate sensor streams back to the server will quickly deplete a phone's battery, leading to unnacceptably frequent recharge cycles. As an illustrative example, the manufacturers claim for a Nokia N95 8GB cellphone with the 1200 mAH BL-6F battery (with an operating voltage of 3.7V) is 280 hours of standby time, which drops to 6 hours while talking or streaming data over a 3G connection. Accordingly, we proposed the Activity Triggered Deep Monitoring (ATDM) paradigm in [6] as an energy saving tradeoff, where the sensor devices are activated and data streams collected and relayed by the mobile gateway only when the monitored individual's context is determined to satisfy certain predicates. This is analogous to the use of a low-power wireless surrogate radio [8] to trigger the primary radio only when a transmission is imminent.

Clearly, determining the correct context itself requires the use of sensor-generated inputs and will itself be subject to some degree of estimation uncertainty and error. We claim that the user context can be determined, to an acceptable accuracy level, by using a combination of both on-board sensors (located on the mobile device) and remote data sources, with significantly lower power consumption.

Thus, under the ATDM paradigm, the mobile gateway takes on the additional role of a *personal activity coordinator* that uses information available from its internal sensors to infer the subject's context. For example, on-board GPS sensors can provide location information, the microphone can provide ambient noise levels and the onboard accelerometer can double as a pedometer. The mobile device also has access to personal information, e.g., the subject's calender. More importantly, there is an increasing amount of generic and personalized context that is stored in the network *cloud*. For example, www.weather.com can be be used to obtain enviromental parameters (such as temperature and air quality measurements) in the subject's current location. Additionally, appropriate mining and reasoning over personal data and activities expressed via channels such as blogs (e.g., Google Blogger^{TM}), microblogs (e.g., Twitter^{TM}) and social networking sites (e.g., Facebook^{TM}) can provide context of sufficient accuracy to control the duty cycle of the external physiological sensors and, more generally, alter the data stream processing logic on the mobile gateway.

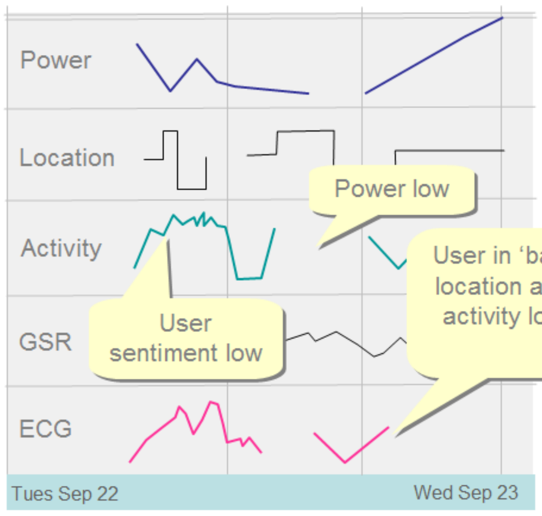


Figure 1. Illustrating the sporadic nature of data collection under the ATDM paradigm. Replaying the contextual provenance helps a medical professional understand the non-medical aspects of the medical data stream.

A. The Utility of Context as Provenance

By adopting the ATDM paradigm, MediAlly can realize significant energy savings because of reduced sampling and transmission costs. However, the health data streams are neither continuous nor always at the same granularity (see Figure 1); rather, the contextual information dictates *when* and *what* data streams are to be generated and collected. The prevalent contextual state that triggered a change in the data generation/ processing logic constitutes the *"contextual provenance"* of the system and helps the medical practitioner to later interpret the sensor data with greater awareness. For example, the doctor can classify a period of elevated heartbeat as normal if she has the information that the subject was at a healthclub. The ability to store and replay such ancillary contextual data is thus a key requirement of any ATDM-based remote monitoring service.

In our design we distinguish between the medical sensor data and the explanatory contextual data for two practical reasons. Firstly, current generation PHR systems are tuned to store time-indexed medical data as numerical values. They do not readily lend themselves to storing the contextual state, specially with different levels of granularity, or the cross-indexing needed (that we shall describe in Section IV) to support appropriate reconstruction. Secondly, the contextual data stream has an innate uncertainty and is therefore subject to lesser oversight (e.g., FDA certification is not required).

Thus, MediAlly infers the subject's context, using a combination of on-board and 'cloud' sensors, and adapts

Table I
CONTEXTUAL TRIGGERS AND MONITORING ACTIVITIES FOR A CANDIDATE STRESS MONITORING APPLICATION

Medical Rationale for Monitoring	**Triggering Contextual State**	**Sensors For Infering Context**	**Recommended Action**	**Sensor Streams Monitored**
Terminate monitoring due to low battery	Phone power $< 5\%$ and charger disconnected	Phone battery sensor	None	None
User may be indulging in depressive behavior	(In Forbidden Area for $\geq$ 5 mins) || (Sentiment=low && Avg. Activity (5 mins)=low)	GPS, Sentiment, Accelerometer	Collect and trx. noise and stress	GSR, microphone
User disturbed at night	(@Home)&& (Sociability=low) && (Avg HR high) && ($10pm < t < 6am$)	GPS, ECG, Calendar Activity	Collect and trx. noise, ECG and stress	GSR, ECG, microphone
User may be isolated or depressed	(@Home)&& (Sentiment=low) && (Avg. Activity (30 mins)=low) && $7am < t < 9pm$	GPS, Sentiment, Accelerometer	Collect and trx. noise and stress	GSR, microphone

the processing logic on the mobile gateway in response to changes in context. This metadata is explicitly collected as an auxiliary data stream by the system and replayed on demand.

B. Sample ATDM-based Applications

The ATDM-based monitoring paradigm, as embodied by the MediAlly system, can be used for both physiological *and* psychological monitoring. It is well known that cardiac failure, a chronic disease, is triggered by a combination of physiological (e.g. high-intensity running), psychological (e.g. depression or anxiety) and environmental (e.g. pollution and ozone levels) factors. Accordingly a subject can be equipped with a set of sensors (e.g. ECG, GSR and pedometer) and a mobile device, so that MediAlly can collect medical data whenever the subject's context indicates physical or emotional stress.

For our implementation, we focused on an application where *activity and physiological data are used as an indicator of mental or emotional well-being*, an area that has been largely ignored in prior remote healthcare monitoring systems. This is motivated by our interactions with clinical psychologists, who have *a)* confirmed that current psychological treatments rely purely on periodic clinic-based interactions, *b)* traditionally placed a lot of faith on user reported data and *c)* expressed excitement about the prospect of using longitudinal context-triggered remotely monitored data to improve treatment of psychological ailments (e.g., post-traumatic stress disorder or depression) [7]. Table I illustrates a subset of context rules that are used by the system in an "activity-driven stress monitoring" application. The high level condition (col1) is decomposed into a set of contextual predicates (col 2), which are evaluated using the sensors in col 3. Columns 4 and 5 respectively indicate the resulting action associated with a successful evaluation and the sensors that get invoked.

III. THE MEDIALLY FUNCTIONAL ARCHITECTURE AND PROGRAMMING MODEL

MediAlly is a client-server based pervasive middleware designed to support ATDM-based monitoring with the following overall goals:

- *Low-overhead Context Provenance:* It must allow the contextual history to be captured, with low communication energy overheads, at appropriately granularity and stored in a way amenable to future reconstruction.
- *Extensibility and Portability:* It should be easy to *a)* extend the middleware to evaluate context corresponding to additional sensors and *b)* operate on a large set of mobile devices.
- *Context Reusability:* It should allow developers of new monitoring applications to reuse (and, if needed, customize) the context inferencing logic already developed for prior applications.

The MediAlly server component is responsible for *a)* derivation of appropriate cloud context and *b)* collection, storage and replay of contextual provenance data. The client component of MediAlly is deployed on the personal mobile gateway (e.g., an individual's cellphone) and performs the following functions:

a) Coordinates communication between the personal gateway and external sensors.
b) Determines local context and retrieves appropriate global context.
c) Collects appropriate contextual provenance metadata and transfers this data to the backend.
d) Performs on-board processing of medical data streams and transmits them to the backend PHR repository.

The component-level functional architecture of the system is shown in Figure 2. This architecture supports context-dependent event monitoring, with contextual triggers dynamically altering the set of monitored sensors and the local stream analytics.

At the heart of the client-side infrastructure is a *Context-Dependent Event Processing Engine* (CEPE), responsible for applying the event processing logic to the incoming medical data streams. Note that these streams are modeled as a sequence of time-value tuples. To improve the energy efficiency of event processing, the CEPE supports both push and pull based data streams and optimizes the data transfer between the sensors and the phone, based on the operational cost of a particular sensor. (These optimizations are not a focus of this paper and are thus not discussed any further.) A *Data Transmission* sub-component pushes relevant data streams to the PHR repository.

The *Activity and Context Trigger* (ACT) component is responsible for inferring valid contextual states. It notifies the *Provenance Manager* (PM) about the temporal evolution of such states. Details about this interaction is provided in Section IV. The PM collects these notifications and subsequently uploads the contextual provenance to the *Personal Context Provenance* (PCP) repository. The PCP is managed by the *Contextual Provenance Server* (CPS) at the server end.

The *Dynamic Sensor Control* (DSC) component implements the 'on-demand' data collection logic. It is responsible for duty-cycling individual sensors and for adjusting appropriate collection and transmission parameters like sampling rates, transmission power, schedules etc.

The *Sensor Adaptation* (SA) component consists of a collection of device/schema-specific adapters, that transform an individual sensor's device-specific data formats into a uniform event-tuple representation. To accommodate complex sensor data types but also allow quick retrieval of canonical data properties, we use a combination of object-oriented (name, value) and XML-based event representation schema.

The *Virtual Sensor* (VS) component serves to shield the CEPE from device specific features of individual sensors by providing an uniform abstraction across local sensors on the phone, external physiological sensors and context sensors in the Internet cloud. It also serves as a means for reusing existing context composition logic by exposing the derived contextual states as additional sensor objects that other applications can leverage upon via a uniform interface. The VS also enforces additional access control policies to

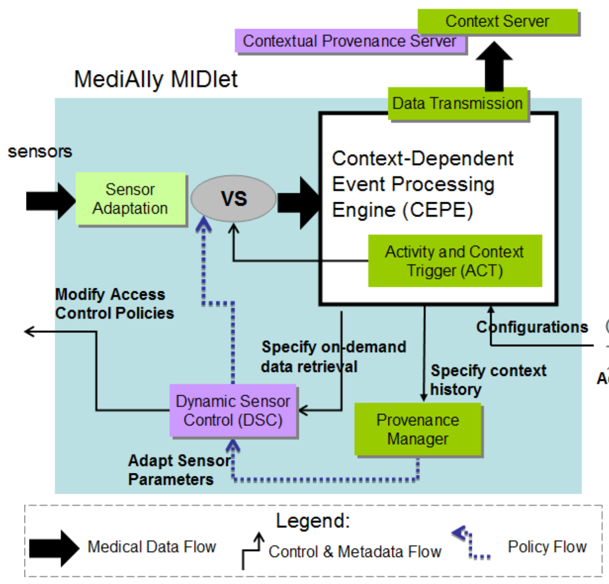


Figure 2. The MediAlly component-level architecture, illustrating the separation of context (provenance) and medical stream monitoring

the logical sensor streams by arbitrating across multiple applications.

The *Context Server* (CS) is responsible for implementing the context sensor connectors. For example the CS can periodically retrieve textual content from the subject's Twitter^{TM} posts, run a sentiment analysis algorithm on the text and return a score to the appropriate client-side VS.

A. The MediAlly Programming and Distribution Model

MediAlly's programming model is meant to ease the development of a runtime infrastructure capable of supporting the ATDM paradigm, which requires applications to adapt their processing in response to appropriate context triggers, as well as specify what contextual data needs to be monitored and stored. To support extensibility (though we have not focussed on this in the first version of our software prototype), we envision the model being supported by both a context composition capability and by an accessible catalog of components. The compositional capability implies that sophisticated runtime behaviour of applications can be built up from more primitive behaviours. This kind of object-oriented abstraction allows application developers to specialize their context inferencing logic by reusing and modifying (as needed) existing context inferencing components. An example is as follows. A developer wants to build a new application that is a superset of VS event streams X and Y. He builds and implements a new component Z with new rules and triggers, fusing the streams supplied by other components specifying X and Y. Note that this brings us to the component catalog. We also envision that rule-trigger components will be described and stored in some way that is easily accessible to the developer (much like Java classes and documentation can be explored and reused using Eclipse), allowing the developer to browse and choose components X and Y (as above) while constructing his new component. In other words, we shall employ a *unified process* for composition.

We structure each application as a set of <*ContextualTrigger, Action*> tuples. Whenever the predicate specified by *ContextualTrigger* is satisfied, the data collection and processing logic in the corresponding *Action* element is invoked. Note that, for example, in Table I, each *ContextualTrigger* itself is likely to involve a process of context composition over the underlying data streams. This process of context composition is modeled as a stream operator graph, with individual nodes representing different contextual states. (Different nodes in this context composition graph can also be encapsulated as Virtual Sensors, enabling other monitoring applications to directly utilize the corresponding inferred context in their context composition process.) After specifying the operator graph, the application programmer is responsible for implementing it within the CEPE, as well as making sure that appropriate changes in the contextual state are reported to the Provenance Manager. The *Action* element of each tuple is also implemented as an operator graph over a set of streams from underlying Virtual Sensors. The output of the *Action* element is a set of "event streams".

IV. Low-Overhead Contextual Provenance Capture and Replay

MediAlly's low overhead contextual provenance capture mechanism relies on the application-specific definition of a **Context Composition Graph** (*CCG*). The CCG is a construct *defined by the developer/programmer of each remote monitoring application* that represents the hierarchical process by which high-level contextual inferences are made by composing low-level sensor-generated data samples. Formally, a CCG is defined as a graph $< V, E, F >$ (V being a set of nodes, connected by a set of edges E and associated with a set of dependency functions F), where a node $v_i \in V$ corresponds to either a specific contextual state or a simple 'logic operator' (either AND, OR or NOT) that expresses how higher level contextual states may be obtained from the values of underlying contextual state nodes. The nodes are connected by directed edges $e_{ij} \in E$, such that an edge from a contextual state v_i to a contextual state v_j implies that v_i is a higher level context state, whose computation involves the composition of context represented by v_j. In this case, $v_i = PARENT(v_j)$ and $v_j = CHILD(v_i)$. Similarly, edges from state nodes to logic operator nodes imply that the state is computed by applying the corresponding operator to the 'child' states, while edges emanating from logic operator nodes point to the sources of underlying context to which the operator is applied. Furthermore, each edge is associated with a *causative function* f_{ij}, that specifies a *causative relationship* between an value of the contextual node v_i at time t and the present or past values of the contextual node v_j. In general, this functional relationship may be viewed as conforming to the Time-Value Centric (TVC) model [14] previously proposed for representing dependencies in medical stream analytics.

Figure 3 illustrates this model for a subset of the contextual states defined by the wellness monitoring application presented in Table I. The nodes on the right represent progressively 'higher' levels of context inferred by the composition of 'lower-level' context nodes to the left. In this simplified figure, each node is assigned a unique *ContextID*. For example, node "015" represents the contextual state of the 'user being at home within the hours of 10pm-6am'. While most of the causative functions have an 'identity mapping' $I(.)$, implying that $v_i(t)$ depends only on the value $v_j(t)$, observe, for example, that the 'average activity' value in $node_{014}$ depends on the 'per-minute step count' values of the past 5 minutes ($node_{014}(t) \leftarrow node_{201}(t-5, t)$). The CCG is thus a representation of the logical process of context computation, with the computed state of $CHILD(v_i)$ acting as an input to the computation of the state represented by v_i.

The *static* CCG specification for an application is defined a priori by the application designer and it is programmatically communicated to the CPS (server-side provenance component) during application registration. The MediAlly runtime tracks the temporal history of each of the node in the CCG and pushes out any change in the value of a node to the PM client. For example, $node_{201}$ represents the contextual value *step count per minute*; whenever its value changes, the latest value is reported to the client-side Provenance Manager. If *step count per minute* is *low* for five minutes the state *average activity over 5 mins becomes low* in $node_{016}$ becomes 'true' and is reported to the Provenance Manager. We note two key aspects of our CCG-based model of contextual provenance collection:

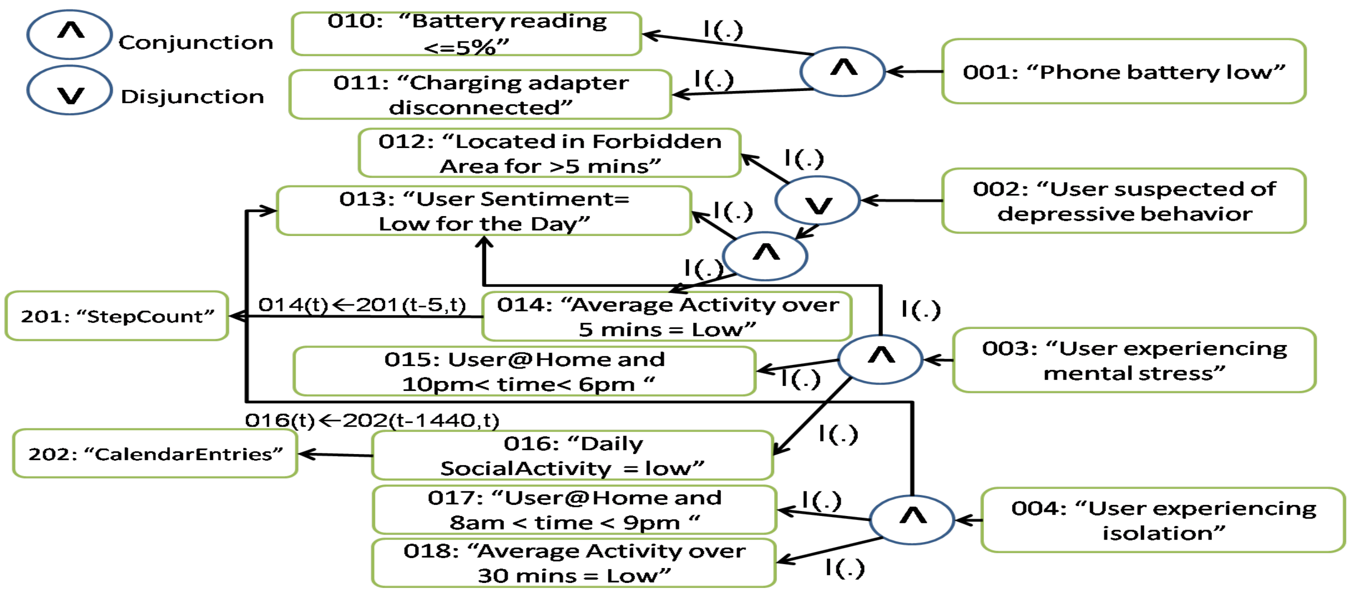
- We view the process of context composition as one of applying a processing graph over a set of either raw or derived streams. Thus changes in an intermediate contextual state correspond to a new element from the corresponding 'logical' data stream. Unlike conventional stream processing, where raw stream values are proactively pushed into the leaf nodes of the processing graph, MediAlly uses a combination of bottom-up and *top-down* processing, with some lower-level contextual states being computed *on-demand*. We discuss this in the following subsection.
- Each application defines its own CCG, which in turn controls the granularity of the contexual provenance that is stored. Even though the application may have computed additional finer grained context internally, it is not tracked by the MediAlly CPS component unless it is explicitly modeled in the CCG. This allows for a tradeoff between the context granularity and the energy burden imposed by the provenance collection mechanism.

A. Collection of Dynamic Provenance State and The Principle of Lazy Capture

To support the easy capture and replay of contextual provenance data, each new value of a node in the CCG as formally represented by the tuple τ:

$$< ContextID, timestamp, value, childMaxOrder > \quad (1)$$

Here $ContextID$ refers to the ID of the context state (node) in the CCG for which this new value is generated, $timestamp$ refers to the local time at which this new value is computed and $value$ represents the newly computed value for that contextual state. The use of the $childMaxOrder$ element is explained shortly. The $value$ element is one of several primitive data types, such as boolean ($node_{002}$), integer ($node_{201}$) or double. The temporal evolution of a context state is a sequence of such tuples. For example,



Conjunction
Disjunction
010: "Battery reading <=5%"
011: "Charging adapter disconnected"
001: "Phone battery low"
012: "Located in Forbidden Area for >5 mins"
013: "User Sentiment= Low for the Day"
002: "User suspected of depressive behavior"
201: "StepCount"
014(t)←201(t-5,t)
014: "Average Activity over 5 mins = Low"
015: User@Home and 10pm< time< 6pm "
003: "User experiencing mental stress"
016(t)←202(t-1440,t)
202: "CalendarEntries"
016: "Daily SocialActivity = low"
017: "User@Home and 8am < time < 9pm "
018: "Average Activity over 30 mins = Low"
004: "User experiencing isolation"
I(.)

Figure 3. Representing the compositional logic of context as a directed graph.

our prototype Wellness Application may generate successive tuples of $< 201, 100, 50, 0 >$ and $< 201, 101, 40, 0 >$ if the user takes 50 and 40 steps in the 100th and 101st minute respectively. The PM client receives such tuples from the ACT and transmits them to the backend, for storage in $ContextID$-specific tables.

We employ two techniques to reduce the overhead of contextual data capturing process. The first technique is straightforward and based on delta transmissions. The CPS caches the last tuple for every unique $ContextID$ and propagates a tuple only if the $value$ or $childMaxOrder$ elements change. The second technique involves the use of logical operators in the CCG, whereby the sub-trees of a contextual predicate are evaluated only when they are necessary to help resolve an ambiguity in the resultant contextual state. Thus the CEPE evaluates contextual states in a top-down fashion and computes the values for child nodes only if these values materially affect the value of the parent node. This technique is similar in spirit to approaches for database query optimizations. To illustrate the point, consider the CEPE trying to determine if the $value$ for $node_{002}$ is *true*. It will initially evaluate $node_{012}$ and if its $value$ is *true* then does not evaluate $node_{013}$ and $node_{014}$. We call this the *principle of lazy capture*.

With lazy capture, the contextual states of sub-graphs in the CCG will not be evaluated at all times. A subtle fallout is that the backend repository might have an incorrect view of the temporal evolution of certain context states. For example, let us say that the $value$ for $node_{014}$ is *high* at time T_{100}. At time T_{110} , $node_{012}$ becomes true and therefore $node_{014}$ is no longer evaluated. In the absence of any further updates from $node_{014}$, we are uncertain about its $value$ after T_{110}.

The key insight is that such inconsistencies in the lower-level contextual state are irrelevant to the accuracy of the provenance reconstruction process if we also store the set of child sub-graphs that were explicitly evaluated. Thus we implicitly define a specific order (e.g., 'left-to-right') for evaluating the child nodes of any logical operator and use the $childMaxOrder$ value to indicate the index of the highest child node that was actually evaluated. In the example above, if the $value$ of $node_{002}$ is *true* on account of $node_{012}$ then we get a tuple of the form $< 002, timestamp, true, 1 >$. On the other hand, if $node_{013}$ and $node_{014}$ are responsible for the $value$ of $node_{002}$ being *true*, then the corresponding tuple is $< 002, timestamp, true, 3 >$. By using a combination of a pre-defined evaluation order for child nodes and the information in the $childMaxOrder$ element, *we can reconstruct the provenance using known good values.*

B. Reconstructing Contextual Provenance

The CPS stores the static CCG during application registration and receives (from the PM) and stores the context state tuples generated during application runtime. By combining such static and dynamic data, one can recreate the relevant contextual conditions at any past instant, using the basic pseudo-code shown in Figure 4. Assuming a time-based causative relationship, it is used to recover the provenance for children of a given contextual state v_i at time t. This reconstruction logic can be used recursively to reconstruct the relevant historical values of contextual states at arbitrary depths.

```
ResolveContextDependency(ParentID, Timestampt t,
Value v, int childMaxOrder)
1. ChildContextSet=$\emptyset$;
2. childNodeIDs= getChildrenFromCCG(ParentID);
// retrieves child node IDs from CCG
3. for ($int j = 0; j <= childMaxOrder; j++$)
4.     $f = getDependency(ParentID, childNodeID[j])$;
       //lookup dependency function
5.     $startTime = t - f.start; endTime = t - f.end$;
       //assumes a time-based TVC-rule
6.     $Causative_j$= SELECT tuple from TABLE(childID)
       WHERE $tuple.startTime > startTime$ &&
       $tuple.endTime < endTime$;
7.     $ChildContextSet = ChildContextSet \bigcup Causative_j$
8. endfor
}
```

Figure 4. Reconstruction of Contextual Provenance

V. Implementation and Results

A. Hardware Details

We chose to implement our external sensors on Realtime System's Shimmer platform, details of which can be found in [1]. This is a lightweight and self-powered platform that can be integrated with a number of physiological sensors using its extension pins. We use the onboard class 2 Bluetooth module to provide connectivity between the sensors and the mobile gateway and keep the emissions below FDA approved thresholds. In our setup we used the onboard Freescale MMA7260Q accelerometer to develop a pedometer, that provides a step count every minute.

For the mobile gateway we used a Nokia N95 8GB cellphone, a Symbian S60 v3 device with 64 MB SDRAM, GSM, HSDPA, Wi-Fi, and multi-profile Bluetooth capabilities. The N95 implements support for MIDP 2.0 (JSR 117) as well as JSR 179 (location), and JSR 82 (Bluetooth).

B. Software Details

Our Medially prototype comprises an Internet-connected mobile phone (including all its internal programmatic capabilities), a platform-independent MIDlet suite implementing the device-resident logic, and programmable Shimmers providing data streams. Server side logic resides on an Internet addressable desktop. In addition, we setup a set of series of cloud (i.e., calendar, blog and social networking sites) context sources and seeded them with values from our 'demo' user, corresponding to the various cloud context states that we wished to investigate.

For development and testing, we used the NetBeans IDE, the Nokia S60 SDK, plugins, and Emulator for NetBeans, and Nokia PC Suite for PC-to-phone syncs. With the appropriate bands or connectors, Shimmers can easily be attached to a demo user's ankles or chest to provide pedometer or ECG data streams. Because it is difficult to test rules whose triggers are biomedical readings (wihout conducting medical trials with real patient populations), we also created emulated sensor readers for all of our sensor types- but especially for the ECG and GSR data that is harder to 'control'- to make testing and demonstration of MediAlly feasible.

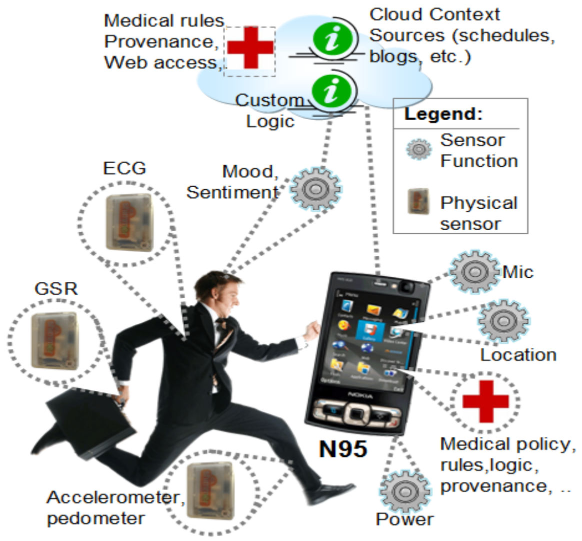


Figure 5. Artifacts comprising the use cases

Figure 5 illustrates the artifacts supporting our use cases. Electrocardiogram (ECG), Galvanic Skin Response (GSR) and pedometer biometric sensor data was gathered from physical sensors. The MIDlet suite embodies our software that interworks with device-resident sensing capabilities: Battery level/charging status, location (via JSR 179 and the GPS receiver), and microphone (noise) levels. The user's mood/sentiment is derived by a custom logic component on a server, as described further in Section V-C. To run our tests, we instrumented the MIDlet to be able to toggle between real Shimmers and emulated streamed data sources (which we could control to recreate appropriate contextual conditions).

Figure 6 illustrates, in greater detail, the relationship between MediAlly components on the mobile device, the server, the cloud, and the database tier. The MIDlet suite includes the CEPE module that interprets the rules that comprise the medical use case. The Provenance component ensured that the appropriate provenance messages were stored and transmitted to the Provenance service on the server. The Virtual Sensor (VS) objects act as gateways to individual sensors (e.g., VS objects may be commanded by the CEPE to change streaming rates or "turn off", which in turn causes an appropriate change on the Shimmer). The MIDlet is designed to communicate back to the Services we installed on our Tomcat instance. These include a Provenance Service (for storing provenance data), a Cloud Service (embodying the interworking and rules of cloud data

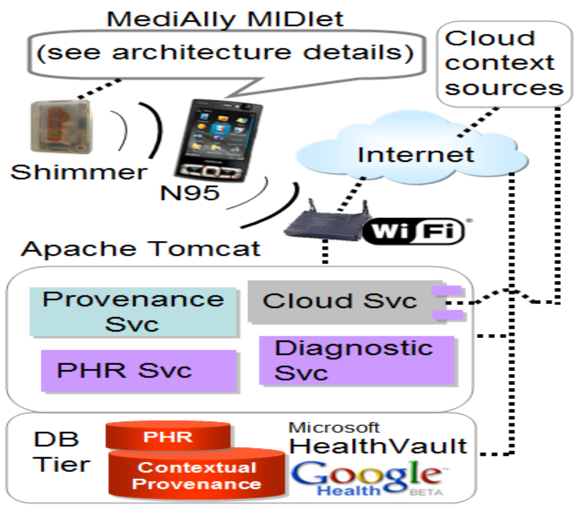


Figure 6. High level implementation architecture

integration), a Personal Health Record (PHR) service (to which our MIDlet posted data streams), and a Diagnostic Service.

C. *Experimental Cloud Service for Opinion Mining*

To obtain mood or sentiment context, we implement a real-time opinion mining (OM) component on the backend server. We achieved this by applying OM analysis to an aggregated set of the user's recent blog and microblog posts (we used Blogger.com and Twitter.com respectively). Our OM implementation utilizes an experimental usage of the SENTIWORDNET software development kit (SDK), which allowed us to build custom OM heuristics around the basic functionality of the SDK, which is to provide Objectivity, Positivity, and Negativity rankings of English words and synsets from WordNet [9].

D. *Database and PHR Tier*

We utilized HTTP-based clients to post both the sensor data streams and the provenance data from the cellphone to the server-side Web Services components (named PHR service and Provenance Service, respectively, in the figures). A combination of Tomcat's in-memory database and a MySQL database was used to store the provenance and medical sensor data. Finally, to allow us to view the streamed and saved provenance and health data, we built a Web-based front end (serving as a mock-up of commercial PHR systems) to the database. The front end uses, among others, the Google Visualization API to implement an annotated timeline for plotting sensor readings over time, much like one sees on commercial PHR sites.

E. *Expected Energy Savings*

The true energy savings in the MediAlly system, arising from reduced sampling and transmission overheads, will be revealed only in a medical trial and will vary based on the application and the sensors in use. Given that we are yet to conduct the necessary clinical experiments, Table II lists the volume of data generated in an illustrative, highly-simplified, cardiac monitoring application, where a medical practitioner monitors the patient's ECG pattern during periods of elevated heartbeat, specially when the subject is active. A ECG sensor is sampled at 250Hz, with 16 bits/sample. In the default (non-ATDM)case, the ECG data stream is transmitted from the sensor to the phone via the Body Area Network (BAN) and relayed to the PHR repository over the Wide Area Network (WAN). Now let us assume that a pedometer provides the activity context (step count/minute) to the mobile gateway. If the application considers high activity as being represented by a step count of more than 10 steps/minute, then the ECG data would be collected only when this predicate is satisfied. Let us assume that a subject crosses the high activity threshold 10% of the time throughout the day. With ATDM (but without lazy capture), this automatically reduces our traffic load to 4Mb, including the additional 3600B of pedometer data stream (1 byte/min) that must now be stored. Now let us assume that the pedometer reports 50 crossings of the '10 step/min' threshold during the day–this will generate only 50 elements in the provenance data stream. Our lazy capture technique will then generate only 50 provenance tuples (8 bytes/tuple) daily from the phone to the backend (instead of the 3600B of provenance data).

Table II
DATA GENERATION IN A CARDIAC MONITORING APPLICATION.

Activity(10%)	Network	Non-ATDM	Context Rules	Provenance capture
ECG stream	BAN	42Mb	4.2Mb	4Mb
	WAN	42Mb	4.2Mb	4Mb
Pedometer stream	BAN	0	3.6Kb	3.6Kb
	WAN	0	3.6Kb	0
Provenance stream	BAN	0	0	0
	WAN	0	0	400b

F. *Open Questions and Issues*

MediAlly continues to be a work-in-progress, with several open questions whose resolution requires the conduct of user trials or experiments. Two important unknowns are the (application-dependent) practical accuracy of the context inferencing logic and the actual observed savings in energy overheads.

In MediAlly, incorrect context inference not only provides misleading provenance data but can also cause the activation of the wrong event processing logic over the medical sensor streams. One common problem in context inferencing is the issue of hysteresis, caused by the fact that an instantaneous change of context (e.g., walking to sitting) will not be accompanied by an instantaneous change in biomedical values (i.e., the heart rate will not instantaneously drop below 70).

This can be countered by embedding a sufficiently long window of evaluation in the event processing logic. Additionally, to accommodate the reality that multiple contextual states may be determined to be simultaneously valid (with different confidence levels), we plan to extend the CCG formalism and lazy capture to apply to the simultaneous storage of *multiple sub-trees* of the CCG.

To establish the practical savings in energy expenditure observed with the adoption of an ATDM paradigm, we are working towards the collection of real activity data for a limited set of individuals. However, based on past experimental data (e.g., observed with the Harmoni prototype in [15]), we expect the system to provide significant savings in communication overhead, especially if the data is stored and then transmitted opportunistically, in bulk, via WLAN hotspots rather than 3G connections.

VI. Related Work

Many pervasive system prototypes have explored the idea of using a cellphone or PDA as a gateway for collecting health data from a variety of medical sensors. While initial prototypes focused on using the cellphone merely as a relay for very infrequently collected medical data (e.g., daily glucose readings [12]), subsequent prototypes have explored the use of localized processing on the mobile device to enable continuous monitoring of health sensor data *streams*. These include the AMON system [10] for multi-parameter (Sp02, pulse and temperature) monitoring, the MoteCare system [13] for personalized health monitoring and the COSMOS middleware [11] for ubiquitous monitoring using ZigBee-based sensors. More recently, the Harmoni prototype [15] used activity context as a trigger for dynamically altering the stream-processing logic on the cellphone, with a view to reducing the transmission overheads of medically unimportant data. The Micro-blog middleware [16] was one of the first to comprehensively demonstrate how the onboard sensors of a collection of commodity mobile phones (e.g., camera, GPS, accelerometers) could be used to develop useful insight into real-world context state. The Mercury project [17] is a more recent and innovative middleware that improves the lifetime of monitoring by dynamically altering the sensing and data transmision profiles (from the sensors to a base station), based on designated lifetime objectives and fluctuating channel conditions. While Mercury does share the notion (with MediAlly) of using context to modulate the transmission of sensor data streams, its focus is more on extending the *sensor lifetimes* (as opposed to our focus on extending the lifetime of the gateway device). Moreover, Mercury does not consider the use of both local and global context to modulate the data collection process. In all of this prior work, there has been no notion of tracking and *storing* the individual's personal and global context with the aim of providing *explanatory provenance* on the underlying historical health data.

The idea of using a combination of locally-generated and cloud context to provide a better situational awareness of an individual's activity has been explored recently–e.g., [20] explored the use of an individual's calendar context and cloud traffic context in building an enhanced, personalized navigation system. Our use of cloud-based sentiment context is based on a variety of machine-learning and classification based techniques that have been recently explored for automatic inference of sentiment, including the classification of product reviews [18] and the detection of an individual's mood changes through blog analysis [19].

Our work on a contextual provenance system architecture borrows from several recent advances in the field of process and data provenance, investigated primarily for scientific worklows [21], file systems [22] and databases [23]. Our model of representing the logic of context composition as a graph is similar to [24], which uses a graph-like representation to support low-overhead *process* provenance tracking in stream-based applications. Our work, however, has two key differences with [24]. First, while [24] focuses on merely capturing the edge linkages between the graph nodes (representing stream operators), we are interested in additionally efficiently capturing the evolution of each individual node (context state). Moreover, we explicitly dynamically modify the provenance capture to track the states of only those nodes which actually affect the *triggering* context at any instant. Our method of contextual provenance representation and reconstruction also utilizes the low-overhead model-based TVC approach to stream provenance introduced in [14]. In this approach, the dependencies between the output and input elements of any stream operator are specified in terms of a functional model, enabling easy reconstruction of incoming data elements (in our case, finer-grained context) that contributed to the generation of an output data element (in our case, higher-level composed context).

VII. Conclusions

In this work, we have presented MediAlly, a mobile phone-based system for capturing relevant medical data streams from a remote subject. MediAlly is a context-aware system that processes the medical data streams based on user-specified context rules. Additionally the system produces a metadata stream describing the contextual provenance surrounding the data collection process. We have currently tested our setup in a laboratory environment. In ongoing work, we are working with medical practitioners to launch clinical trials. In the future, such trials will give us a clearer understanding of the accuracy of context inference for specific application needs, as well as establish the increase in operational lifetime observed in actual practice.

References

[1] The SHIMMER sensor platform http://shimmer-research.com.

[2] D. Husemann, C. Narayanaswami, and M. Nidd. Personal mobile hub. In ISWC 04: Proceedings of the Eighth International Symposium on Wearable Computers (ISWC04), 2004.

[3] R. Jafari, F. Dabiri, P. Brisk, and M. Sarrafzadeh. Adaptive and fault tolerant medical vest for life-critical medical monitoring. In SAC 05: Proceedings of the 2005 ACM symposium on Applied computing, pages 272279, 2005.

[4] T. Gao, et al, The Advanced Health and Disaster Aid Network: A Light-weight Wireless Medical System for Triage, IEEE Transactions on Biomedical Circuits and Systems, Volume 1, Issue 3, September 2007.

[5] A. Shimpi., No 3G on the iPhone, but why? A Battery Life Analysis, Anandtech.com whitepaper, http://www.anandtech.com/printarticle.aspx?i=3036

[6] A. Misra, B. Falchuk and S. Loeb, Server-assisted Context-Dependent Pervasive Wellness Monitoring, in WIPH'09: 2009 International Workshop on Wireless Pervasive Healthcare, March 2009.

[7] C. J. James, et al, Personalized Ambient Monitoring of the Mentally Ill, 4th European Conference of the International Federation for Medical and Biological Engineering, November 2008.

[8] E. Shih, P. Bahl and M. Sinclair, Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices, in Proceedings of the 8th annual international conference on Mobile computing and networking (Mobicom), September 2002.

[9] A.Esuli, F.Sebastiani, SENTIWORDNET: A Publicly Available for Opinion Mining, In Proc. of the 5th Conf. on Language Resources and Evaluation (LREC06), Genoa, 2006

[10] P. Lukowicz, et al, AMON: A Wearable Medical Computer for High Risk Patients, iswc, pp.0133, Sixth International Symposium on Wearable Computers (ISWC'02), 2002.

[11] Y. B. Kim, M. Kim and Y. Lee, COSMOS: a middleware platform for sensor networks and a u-healthcare service, SAC 2008: 512-513

[12] C. Kirsch, M. Mattingley-Scott, C. Muszynski, F. Schaefer, and C. Weiss. Monitoring chronically ill patients using mobile technologies. IBM Systems Journal, 46(1):8593, 2007.

[13] E. Lubrin, E. Lawrence, and K. F. Navarro. Motecare: an adaptive smart ban health monitoring system. In BioMed06: Proceedings of the 24th IASTED international conference on Biomedical engineering, 2006.

[14] M. Blount, J. Davis, A. Misra, D. M. Sow and M. Wang. A time and value centric provenance model and architecture for medical event streams, in HealthNet, 2007.

[15] I. Mohomed, A. Misra, M. Ebling, W. Jerome. HARMONI: Context-aware Filtering of Sensor Data for Continuous Remote Health Monitoring. PerCom 2008.

[16] S. Gaonkar, J. Li, R. Roy Choudhury, L. Cox and A.Schmidt. Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation, MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services, June 2008.

[17] C. Lorincz, G. Challen, A. Roy Chowdhury, S. Patel, P. Bonato and M. Welsh. Mercury: A Wearable Sensor Network Platform for High-Fidelity Motion Analysis. SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, November 2009.

[18] K. Dave, S. Lawrence, and D. M. Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In WWW '03:Proceedings of the 12th international conference on World Wide Web, pages 519528. ACM, 2003.

[19] R. McArthur, Uncovering deep user context from blogs, Proceedings of the second workshop on Analytics for noisy unstructured text data (AND), 2008.

[20] B.Falchuk, S.Loeb, T.Panagos, "A Deep-Context Personal Navigation System", Proc. ITS America 15th World Congress on Intelligent Transportation Systems, 2008.

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

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

[23] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. Proc. Second Biennial Conference on Innovative Data Systems Research (CIDR '05), January 2005.

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