Provenance in the cointext of personal IoT data from Things (PDT)

Preliminay draft, summer 2016

General scenario:

This is a health monitoring scenario currently under development within our lab, the specific data is a combination of accelerometry and glucose levels data. This is part of a broader research effort but for the sake of this scenario, we can assume goal is to alert users when their activity levels are low and the glucose levels raise above a threshold. The system should be capable of explaining why certain alerts are raised and how the user reacted to them.

**The point of this description is to identify the minimal instrumentation and metadata injection / provenance collection that needs to occur along the processing pipeline to support full accountability. The extra components are those shown in red in the figures below.**



1. Alice and Bob independently produce streams (two each for example) from 4 sensors: S1-S4.

2. Each user’s streams are collected by a *field gateway* , ie GWY1, GWY2. In the prototype these are the users’ smart phones. These are the first edge elements that can do some computation.

3. Each gateway discretises its input stream (for instance by windowing) and does some processing, producing an output stream which is a sequence of tokens (content TBD). The stream is sent to cloud-based consumers where they are re-aggregated and further processed. Comm is mediated by a messaging queue with a set of predefined topics that consumers may subscribe to. We will use MQTT as the reference protocol as it decouples technologies nicely at each end (ie the cloud end is a JMS queue) and it supports arbitrary payloads, so we can inject minimal metadata in the messages.

4. consumers subscribe to topics and pick them from the queue as they come around. They are analytics apps that can generate alerts based on the stream they see.

(sorry I missed out on 5!)

6,7,8. apps may generate alerts which are persistently stored in an alert DB and pushed to the user through the gateway (ie alert popup on the phone)

(users react to the alert, ie dismiss or take other in-app action).

Generating messages



gateways produce messages of the form <s, t, v> where s is the sensor ID, t local timestamp, v is the payload (sensor reading within the window, or simple stats such as avg / var of reading levels, depending on the specific application).

The pair msgid = <s,t> uniquely identifies a message (we could add the gwy id too if the local timestamp is found to be ambiguous).

Enqueueing



We propose to intercept the messages into the (MQTT) queue(s) with a component that generates a ***provlet*** (small provenance document, in this case a single statement), of the form

wasGeneratedBy(msgid, gwyid, t)

to denote that the msg has been generated by a gateway at a certain time.

Consuming



Here we also intercept the messages as they are extracted by each of the consumers, and generate provlets of the form:

used(ag, msgid, t)

where ag is the id of the consumer analytics app (I forget why I called them ag!)

Generating alerts



more red components here to intercept DB writes, with provlets of the form:

{ wasGeneratedBy(al, ag) }, wasderivedFrom(al, msgid)

to indicate that (1) the consumer app ag generated alert al, and (2) alert al is causally connected (derived from) a set of msgs

Pushing alerts



this step is captured by more red components and the corresponding provlets may take one of a few possible forms. One is indicated in the figure:

Alert *al21* was used by a *gwy\_push* activity (on the phone) which pushes the alert to the user (entity *userprompt*) and is performed by *gwy2*.

Then possibly a *userAction1* is performed by *Bob*, which reacts to the prompt and generates a specific *ack*.

So this is the basic infrastructure for provenance collection at various observation points. The provlets are collected in a central place (a Neo4J DB for instance) where one can doing analytics on them, query to discover who reacted to which alert, etc.

I see a secondary but not trival problem of scale as this is gong to generate *a lot* of provlets. So scalability of provenance management in the IoT context is THE problem I think. Some have addressed it using a form of summarisation – we could draw from that work but they are not my own results.

Still, in a paper about provenance in IoT I suspect that scalability will have ot be addressed.

11-5-2016

Challenges:

- upsampling. downsampling in response to interesting events

- diff on different types of data and use that to trigger provenance recording on/off. Threshold in diff. – differences in two datapoints

Notes to Michele:

Simon is working on a demonstrator that currently uses default sensor data from the laptop, as a proxy for accelerometer data, i.e. from a wearable.

The plan is to incrementally implement the metadata injection (sensor id + timestamp) into a queue-based M2M protocol (MQTT).  Simon is using ActiveMQ with MQTT.  The red component in the figure (attached again) are being implemented as well.

My part is to simulate the provlet generation, which is the job of the red components, so we can start playing with a small provenance DB right away and show what we can do with it — i.e. use it to explain alerts, track trajectories of personal data, etc.

One of the challenges is scalability, i.e. how much provenance the red components generate, wrt how much we actually need.  One idea is to have a reactive system where we can up/down sample the sensor readings based on interesting events in the data streams (e.g. outliers, trends), thus increasing/decreasing the density of the provenance information.  Another idea is to use summarisation techniques (developed by others) to effectively compress the provenance, exploiting the regular structure of the graph.

The issue of verifiability of the provenance traces is not addressed, and I believe this is where you might have an interest?  or are there other areas where you would see your interests fit into this effort.