A Connective Fabric for Bridging Internet of Things Silos

Simon Mayer, Erik Wilde, Florian Michahelles
Siemens Corporate Technology
Berkeley, USA
Email: simonmayer@siemens.com

***Abstract*—Social and socially-enabled applications have established themselves as a large and important set of scenarios that make sense in personal as well as in enterprise settings. However, it is still not clear how to best promote an open and extensible ecosystem of social applications. This makes it hard to design and implement applications that are good SOA citizens and can participate in an open and growing set of social interactions. Our work presents an architecture and implementation based on Activity Streams, which uses and slightly extends the format to work as a foundation for a SOA that allows applications to bridge vertical silos in the Internet of Things. In this paper, we discuss the application of our architecture in several different domains and our findings from mashing up applications across silos using Activity Streams. In addition to demonstrating the feasibility of this approach, we believe that our work can serve as illustration and guidance in the ongoing effort to standardize the next version of Activity Streams.**

I. Introduction

The World Wide Web has become the public face of the Internet. Instead of merely representing a technical revolution, the Web has evolved as a social phenomenon that facilitates the building of communities around content and services. The Web's focus on resources and the principle of relating resources by linking URIs allows for maintaining a web of related resources in a collaborative manner. Finally, the concept of applying these core Web principles to the communication between devices – that any component with an identifier can be linked to, that uniform interfaces provide access to general resource representations, and that content types can be negotiated during client/server exchanges – has been extensively studied and is being widely adopted in the *Internet of Things (IoT)* domain [1], [2].

While Web technologies have certainly helped to make services scalable and accessible, the underlying architecture is based on client-server applications – rather, we would like to mimic the nature of social interaction, i.e., loosely coupled event-driven messaging between multiple entities [4], [8]. To enable this, in the *World Wide Web Consortium (W3C)*, proponents of the social Web are pushing the development of so-called *Activity Streams (AS)* (see more details in Section II) as a general-purpose mechanism of loosely coupling social services.

It is the goal of this paper to investigate the application of AS to IoT scenarios and to investigate how the principle of loose coupling can be applied for vertical and horizontal integration of networked physical devices. This paper introduces AS that are brokered by the communication platform *ASbase* as a connective fabric among devices. In this way, we empower devices to generate events in the form of *activities*, describe the status of performed working steps, and allow other devices that are capable of processing and reacting to a particular activity to coordinate and collaborate their actions. We investigate how this mechanism can scale and be applied to various use cases in industry, and show that it allows changes in the system setup to remain local, expensive re-configurations to become obsolete, and system extensions to become seamless such that the overall device ecosystem can evolve with few static interdependencies.

The remainder of this paper is structured as follows. Section II introduces the concept of Activity Streams, an event-based communication mechanism that implements loose coupling of services. Section III presents our implementation of the ASbase broker that allows to produce, consume, and filter device-generated activities. Section IV illustrates the *vertical* integration capabilities of ASbase by means of a healthcare application scenario and Section V targets its *horizontal* integration capabilities as demonstrated by the integration of four implementation projects. Section VI provides an outlook how provenance-enriched activities can be applied to locate the root cause of an event chain by means of a service technician scenario. Finally, sections VII and VIII summarize next steps and lessons learned from the vertical and horizontal integration of IoT devices building upon loose coupling.

II. Activity Streams (AS)

Attempting to provide a foundation for the loose coupling described above, an informal group of developers defined *Activity Streams (AS)*, a simple format which allows the sharing of social activities (such as *like* and *follow* activities by users) across services. The initial AS version never became an official standard. Still, the specification gained traction and was used as an initial submission to the W3C, where it is currently being developed in a second iteration within the *Social Web Working Group (SocialWG)*.

The newer version of the specification, *Activity Streams 2.0 (AS2)*, currently is being specified as a core format [5], and a vocabulary [6] that defines the terms that can be used in AS objects. While the current focus is on a JSON-based syntax, the WG is exploring options to also support an "RDF view"

978-1-4673-8058-4/15/$31.00 ©2015 IEEE

of AS2 by supporting or even requiring JSON-LD [7].[1]

AS allows social services to share information about activities in a social network. The goal is to allow various services to share this information, so that activities within one service can serve as input to other services. At its core, AS does however not go far beyond defining a simple grammar for statements of the form "Alice added a picture to her album." In this example, the activity contains four main elements:

- *Alice* is the *actor* and is the entity performing the activity represented by the AS activity.
- *Added* is the *verb* and identifies the action represented by the AS activity.
- *Picture* is the *object* of the activity, and identifies the primary object of the AS activity.
- *Album* is the *target* and identifies the target of the AS activity. It often is an object that in an English sentence is referred to by the preposition "to."

While AS1 itself only defines a single generic verb ("post"), an additional (but essential) *Activity Base Schema* defines a substantial set of additional *verbs* that mostly originate in the social media space. Examples are the above "add", "invite" for an invitation that has been extended, and "follow" for the activity that some social actor has started following another actor.

The AS1 base schema also defines a variety of additional concepts, such as *object types*, and additional descriptive metadata for specific object types. While the mix and structure of the concepts defined in the AS1 base schema looks a bit eclectic, it simply is a result of implementor feedback that was collected during the initial AS1 development.

One of the most important features of AS is its extensibility: while the base schema defines a set of concepts that are supposed to be shared across implementations, the AS core model is open for extensions, and thus it is possible to introduce new concepts at all levels of the AS base model, such as new metadata for activities, new verbs, new object types, and new properties of objects.

This extensibility of AS is the reason why it can comfortably span use cases from the original social Web scenarios while also accommodating requirements that may emerge in social enterprise and industrial IoT scenarios. By defining new vocabularies, AS-based services can furthermore share information that goes beyond the base schema, as long as there is shared understanding of the concepts represented by these vocabularies. Following the principles of an open and extensible *Service-Oriented Architecture (SOA)*, those vocabularies can either incrementally extend existing concepts (such as adding new properties to objects that are only understood by consumers who implement these properties), or invent new activity types (i.e., verbs) which will not be meaningful to systems that only implement the base schema.

In the following sections, we present an example of how AS can be used outside the classic social Web domain: we present an AS brokerage platform (Section III) as well as a concrete scenario where we use AS as the foundation of a distributed personal health system (Section IV). We also show how the base schema can be extended with new vocabularies to extend the basic social plumbing of AS in a way that is still in line with the base schema of AS, but also allows to enrich AS-based social interactions with new expressiveness.

[1]The first version of AS had serializations based on JSON and XML (using Atom [3] for the latter).

III. The ASbase Platform

We present the ASbase platform as an implementation of an AS broker that allows to quickly create, test, and mashup social services. ASbase accepts activities in the AS format and distributes them to interested consumers, where its API supports both a request/response and a publish/subscribe pattern. To enable clients to specify which types of activities they are interested in, the platform implements a generic filtering language that is based on the query language used by MongoDB[2] and should therefore already be well-known and straightforward to use for many developers.

A. ASbase REST API

The basic functionality of ASbase is to allow AS-based social services to produce and consume AS activities. To enable this, ASbase exposes a REST API that provides straightforward access to its services.

Producers of activities use HTTP POST requests to transmit these to ASbase, where they are parsed using the OpenSocial AS library[3] and stored internally using MongoDB. At the time it receives a new AS, ASbase checks whether this activity corresponds to any subscription of any registered user. If the activity indeed matches one of the subscriptions, the platform pushes it to that user via its callback channel (e.g., a simple HTTP callback) — these callbacks are registered at the time a new user signs up to use the platform. Once a user has been created, it can create subscriptions to specify what kind of AS it is interested in by using our filtering language (see below).

Apart from allowing users to *subscribe* to specific types of activities and get them pushed automatically, clients can also directly *query* ASbase — to enable this, the platform supports a simple mechanism to retrieve AS within a specified time window using HTTP GET requests, and a more advanced HTTP POST-based mechanism where clients may also include an AS filter.

B. AS Filtering

Instead of defining an AS-specific filtering language, we decided to make use of the MongoDB query language for this purpose.[4] This language natively supports comparison operators for individual values and arrays (e.g., `greater-than` or `contained-in`) as well as regular expression-matching and searching in texts. Its support for logical operators means that clients of ASbase can be arbitrarily specific in defining which

[2]`http://www.mongodb.org`

[3]`https://github.com/OpenSocial/activitystreams`

[4]`http://docs.mongodb.org/manual/reference/operator/query/`

kinds of activities they are interested in, thereby constituting a powerful *deep filtering* mechanism. Finally, the language supports GeoJSON-based geospatial queries (e.g., `geoWithin` or `near`). Listing 1 shows an example filter for ASbase — using comparison and geospatial constructs, this filter matches any activity that contains either one of the verbs "post" or "like" and whose actor's location is within 1000 meters of the Golden Gate bridge.

```
{
  "verb" : {
    $in : [ "post", "like" ]
  },
  "actor.location" : {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [ -122.476447, 37.808153 ]
      },
      $maxDistance: 1000
    }
  }
}
```

Listing 1. An example AS filter that matches 'post' and 'like' activities from actors within 1000 meters of the Golden Gate bridge.

IV. Personal Health Scenario

As one concrete example of a scenario that can be realized using ASbase, we present our HealthViz project, in which we investigate the integration of wearable sensors from the consumer wellness domain into a professional health IT platform. In this scenario, individuals use wearables or services on portable devices (such as activity tracker applications on smartphones) that allow them to collect information about their level of exercise and other physical activities. This information can then be used by health coaches or doctors to better assess a patient's lifestyle, and to monitor and suggest lifestyle changes.

This is an interesting use case for the ASbase platform because it requires a mediator between the emerging variety of wearable sensors and the healthcare IT platform. In this scenario, we thus use ASbase to decouple the data collection from concrete devices and services from the use of the collected data in further stages of the data processing pipeline, such as during aggregation and analysis, and also from the final consumption by user interfaces that are targeted at health coaches and doctors: in the same way as social picture sharing services should not depend on the specific devices that were used to take pictures, health-related data analysis should be independent of the specific data collection device.

Devices that we have experimented with so far include continuous heart rate trackers, exercise-based heart rate trackers, body posture monitors, and physical activity categorization services (which distinguish between, e.g., walking, cycling, and sleeping). To inform and enable the automatic translation of data provided by these services in the future, we have also compiled a list of a variety of devices and services and collected the different categories of activities they are aware of.

Fig. 1. A health coach uses our visualization interface to discuss patients' health data with them.

This list is openly available[5] and currently lists 10 services and a total of 435 categories of physical activities, ranging from very basic activity types such as *walking* (all 10 services know this activity) to highly specialized activities such as *High-Intensity Interval Training* (HIIT; only one of our surveyed services is aware of this type of physical exercise).

To facilitate the exchange of data between the different tiers that are involved in our system (data upload, analysis, and visualization), we have furthermore created a simple CSV-based *Exercise Data Format (EDF)*[6] that decouples the representations of specific services and provides a unified way in which downstream processing steps can consume exercise data. This time series-based format is designed to be open and extensible, and currently only supports the rather small set of sensors we have used so far.

A. ASbase Setup

Using the ASbase platform described in Section III, our setup is based on the following general information flow:

1) Devices or services generate measurements that in the majority of cases are available through specific APIs of these data providers. We provide adapters that take available data, transform it into an EDF representation, and then indicate that new data has become available by publishing this information in the form of a new Activity Stream.
2) Health-related services use the ASbase platform to subscribe to these kinds of events (the AS verbs) from the patients (the AS actors) they are interested in, and then process the raw EDF data. One example for such a processing step is an analysis step that takes a patient's heart rate data and scans it for periods of critically elevated heart rate.
3) The analysis step generates reports which again are represented in a way that can serve as input for subsequent processing steps. The availability of these reports is announced through a specific AS verb, "analyze," which allows potential consumers of these reports to subscribe

[5] https://github.com/dret/exercise/tree/master/types
[6] https://github.com/dret/exercise/tree/master/EDF

```
{
  "uuid" : "http://example.org/activities/add/1234",
  "published" : "2015-01-06T15:04:55Z",
  "startTime" : "2015-01-05T17:33:40Z",
  "endTime"   : "2015-01-05T19:08:39Z",
  "status" : "completed",
  "actor" : {
    "objectType" : "person",
    "id" : "http://example.org/TestAthlete",
    "displayName" : "TestAthlete"
  },
  "verb" : "add",
  "subVerb" : "http://strava.github.io/api#Ride",
  "object" : {
    "objectType" : "exerciseRecord",
    "dataFields" : [ "HRM", "Geo", "AirTemp" , "Time" ],
    "id"  : "http://www.strava.com/activities/237180070",
    "url" : "http://www.strava.com/activities/237180070/
        export_gpx"
  }
}
```

Listing 2. An AS that expresses that information about the actor engaging in a physical activity was added to his/her exercise log.

to them and be notified about their availability through ASbase.

4) The described consume/analyze/report cycle can be chained, where multiple inputs can also be merged into more advanced aggregate reports.
5) Finally, the last consumer in our scenario is a visualization interface (see Figure 1) that a health coach or doctor can use to gather data about a patient's personal health status. This interface might at first merely show aggregates and summary reports, but can give health professionals the ability to load additional data on demand — in our example, this additional data concerns information about the specific type of physical exercise (e.g., walking or HIIT) and periods of critically elevated heart rate.

We discuss the AS verbs that we use to enable this information flow in the following two sections.

B. `Add` — Adding new Exercise Data

The *add* activity (see Listing 2 for an example) conveys that the actor engaged in a physical exercise from *startTime* until *endTime* and that information about this activity was added to an exercise log. The activity also carries a unique universal identifier and holds more detailed information about the specific type of physical exercise — whether this information is indeed available depends on the origin of the data. In addition to the AS base schema vocabulary, we use the following properties to package this information:

- `subVerb`: This property gives more information about the detailed kind of exercise that the *actor* performed. In the example in Listing 2, the *subVerb* tells us that the actor was riding a bicycle.
- `object`: This property links to the raw exercise data. The *objectType* must be *exerciseRecord* and the *object* may be further described using the property *dataFields*, which lists the types of available data as an array.

In the example in Listing 2, a patient (the actor) has made new data available representing a physical activity, in this case a bike ride. It is identified through a type that is specific to the origin of the data, which in this case is the Strava API. There also is information about the available data, and a link to it.

In our system, this API-specific AS activity will be consumed by an adapter that performs two basic tasks: First, it takes the service-specific exercise type (the *Strava ride*) and maps it to a vocabulary that is API-independent. Second, it transforms the available data to the EDF format, and then re-publishes the data to ASbase in this new representation to enable consumers of the AS to focus only on dealing with EDF, rather than with the specific export formats of the proprietary APIs. The harmonized exercise data thus carries an independent exercise label and links to the EDF representation of the data — subsequent analysis steps can consume this data and perform analyses on it regardless of where it originated from.

C. `Analyze` — Publishing an Analysis Result

The *analyze* AS activity (see Listing 3 on the next page) signifies that the actor produced new insights from processing input data. We use the following properties to represent different facets of the analysis:

- `subVerb`: Analogous to its usage in the *add* activity type, we use this property to embed more information about the specific type of analysis that was performed.
- `object`: The raw output of the analysis. The *object* may be further described using the *dataFields* property in the same way as for the *add* activity.
- `subject`: The entity that the analysis concerns, e.g. the person that the analysis is about.
- `criticality`: A rating of the criticality of the analysis results. This enables downstream processing units to react appropriately, e.g. by raising an alarm.

The example shown in Listing 3 tells consumers that a report about elevated heart rate episodes was generated and is available at the published URI. The AS activity can optionally contain metadata about the report (e.g., *elevatedHRepisodes*) which enables consumers to create filters on ASbase that consider these properties (e.g., "deliver reports with more than five elevated HR episodes").

D. User Interface

Our visualization interface consumes AS activities and data representations that are linked from them and displays them in a dashboard-style way that gives the health coach or doctor a quick overview of the most relevant analyses of the recent past. If they choose to do so, they can drill down into the data, in which case more detailed representations will be accessed and visualized. Finally, the ASbase subscription API can be used to have the interface dynamically update itself while being used by the health professional.

```
{ "uuid" : "http://example.org/activities/analyze/1234",
  "published" : "2015-01-13T13:14:55Z",
  "startTime" : "2015-01-11T00:00:00Z",
  "endTime"   : "2015-01-12T00:00:00Z",
  "actor": {
    "objectType" : "software",
    "displayName": "The HR Analyzer"
  },
  "verb": "analyze",
  "subVerb": "http://example.org/activitystreams/types#
      ElevatedHRAnalysis",
  "criticality" : "low",
  "subject": {
    "objectType" : "person",
    "id" : "http://example.org/TestAthlete",
    "displayName" : "TestAthlete"
  },
  "object" : {
    "objectType" : "elevatedHRreport",
    "elevatedHRepisodes" : 4,
    "title": "HR episodes for January 11, 2015",
    "url": "http://www.heartrate.com/report/438782763587.
        csv"
  } }
```

Listing 3. An AS that expresses that a new analysis of data from the subject is available.

V. AS for Bridging Vertical IoT Silos

Our personal health project shows that AS can be used as a connective fabric between loosely coupled actors developed by a project team. To verify our claim that AS-based decoupling is also valuable for bridging vertically integrated silos, we did another experiment that involved multiple teams first implementing four different scenarios, and in a second step integrating these verticals using AS.

A. Study Setup

In our study, 17 students worked on implementing different IoT applications within four independent project teams of four/five students each, for eight weeks. The projects were part of their coursework in the IoT: Foundations and Applications lecture at UC Berkeley, with the goal of demonstrating a functional prototype after six weeks. Each group was asked to think about their project in terms of decoupled tiers of computation, implement the decoupling using AS via ASbase, and document the concrete activity types they used in a simple custom format, the *Activity Streams Documentation Language (ASDL)*.[7] The individual projects that were implemented as part of the students' coursework are briefly described here:

- Internet of BART ("BART"): The goal of this project was to automatically recommend one of the multiple cars of a commuter train based on the crowdedness of the individual cars. For this, the students used a camera that is coupled to an accelerometer (both Arduino-based), an image analysis back end, and two user interfaces: an Android application that displays the crowdedness of each car on a smartphone screen, and an LED strip that produces an intuitive visualization of the entire train and color-codes each car's crowdedness.
- Internet of Seats and Spaces ("SEATS"): This project team implemented a distributed management system for seats at crowded spaces, such as the UC Berkeley library. Their implementation consists of a pressure sensor that is attached to seats and detects their occupancy state, iOS and Android applications to send seats reservation requests, and a back-end management server that approves or denies reservation requests.
- Family Fitness Tracker ("STEPS"): The goal of the STEPS team was to motivate family members to work out more by constructing a game experience on top of steps data recorded by a Pebble fitness watch — using their application, one can launch tournaments and have their back-end tournament manager produce reports on the current progress of the competitors and the individuals' steps goals.
- Eye of Things ("EYE"): The EYE team implemented a monitoring application that would allow users to better stay in control of their interacting IoT devices and services. They use the Wolfram DataDrop platform as a common database for recording events in users' smart environments and produce basic co-occurrence maps that are visualized on a Web interface, and can also consume events from the If-This-Then-That (IFTTT) service.

[7] https://github.com/dret/ASDL

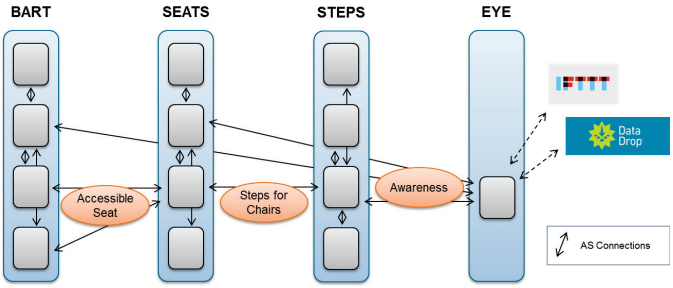

Fig. 2. Overview of the cross-project mashups that our students implemented on top of ASbase.

After six weeks, the groups were given the task of additionally implementing at least one functionality that involves using a service from another project. We asked them to also record all changes they had to make to their implementation (in particular to the internal and external data models they used) for integrating with another project, as well as all changes that they asked any other group to make to their implementation.

B. Implemented Mashups

The students were very creative in designing cross-project mashups (see Figure 2): for instance, the BART and SEATS teams integrated their applications by, additionally to displaying the general crowdedness of commuter rail cars, showing whether the handicapped seating is available or not, again using their LED strip controller. The students described the integration as very straightforward: SEATS subscribed to *trainRecord* activities and sent derived *trainRecordWithSeatingInfo* activities back to ASbase. On top of this mashup, the STEPS team integrated their gamification engine with SEATS' software, thereby enabling seat reservations as rewards for a high level of exercise: multiple actors can compete using the STEPS

platform — whoever wins the contest thereby obtains the right to reserve a seat in a specific space (and, by extension, in a commuter rail car). Finally, the EYE software displays activities that are passed around by all other projects: this group asked the others to augment their activities with a *provider name* property (to support users when browsing the EYE interface) as well as a location object with information about the *city*, *state*, and *country* where an activity took place.

C. Findings

When asked about their experience with using AS, the students liked that ASbase helped with communication and coordination by acting as "glue" for their systems (e.g., BART used it to decouple their Arduino-based on-train system, the image analysis backend, the mobile application, and the LED strip controller). Regarding the integration across projects, the SEATS team in particular liked the clean separation of data from different projects that catered to "simple integration," and that one could view the other projects as black boxes where "everything already works." They also liked that the STEPS and BART teams did not need them to change anything about their activities. Finally, the STEPS team appreciated the modularity and the additional use cases that are enabled by using AS as common connective fabric which they found very valuable for integrating with other ideas: "[AS enables a] level of modular design that is really cool. We hardly needed to get in touch with the Seats and Spaces Team. (...) Everything was already there!"

The main challenges that the students described had to do with the documentation of other groups' Activity Streams. SEATS said that clear communication and public documentation of the used activities would have been great from the start, and that they faced a problem of "moving targets" when integrating their project with others. Due to the same problem, also STEPS was at first confused about how to structure their project to fit with the SEATS group. The SEATS group said that, despite the generally valid and helpful black-box assumption, one still needed to talk to the other team for fully understanding the functionality of their project. Finally, EYE remarked that they faced challenges with others using standardized data formats, e.g., for location information and timestamps.

The teams' comments motivate extending ASbase in two directions: an index of used verbs and activity types should inform users about all activities that are being used by other projects, and an activity monitor should check whether incoming activities are compliant to published ASDLs — both these changes motivate making ASDL descriptions machine-readable.

VI. Provenance-enriched AS for Machine Maintenance

Finally, we investigated the embedding of explicit, simple, provenance information with AS. In our use case, we apply AS to a machine maintenance scenario where different processing tiers interact via ASbase (see Figure 3): (fake) events produced by a wind turbine are received by a (local) turbine monitor that analyzes the data the events link to and classifies the events according to a criticality measure (this is very similar to the personal health use case discussed in Section IV). Another, higher-level, monitor is subscribed to events that are classified as *critical* by the individual turbine monitors and, for each such event, determines whether a maintenance technician should be invited to have a closer look at the turbine. Technicians, in turn, carry mobile devices with an Android application that is subscribed to those *invite* events they are qualified to respond to.

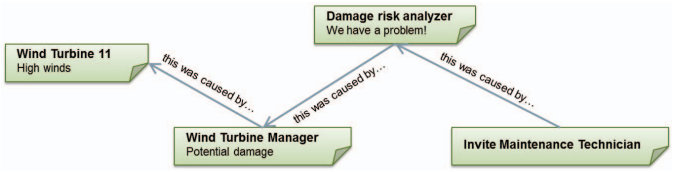

Fig. 3. Provenance chain between different activities that are brokered by ASbase.

```
{
  "id": "http://128.32.78.37:8080/activities/de6cc21bc7404
      ae8977d7bb257d990f4",
  "published": "2015-04-27T19:18:42.399Z",
  "verb": "invite",
  "actor": {
    "displayName": "Windpark Monitor",
    "id": "http://example.org/monitors/windparkmonitor",
    "objectType": "software"
  },
  "object": {
    "displayName": "Maintenance Technician",
    "id": "http://example.org/maintainers/152",
    "objectType": "person"
  },
  "provenance": [
    {
      "relationship": "prov:wasDerivedFrom",
      "object": "http://128.32.78.37:8080/activities/e379a3
          29756345208d6d34a27bc4bc61"
    },
    {
      "relationship": "prov:wasGeneratedBy",
      "object": "http://example.org/monitors/
          windparkmonitor"
    }
  ]
}
```

Listing 4. An AS that is enriched with provenance information based on the PROV ontology.

To access the data that is produced by the wind turbine during the maintenance operation, the technician's application requires a way of relating the *invite* that it received to the corresponding original activity produced by the wind turbine. Additionally, it could be valuable to enable the technician to access the activities that were published by the two intermediaries and potentially contain valuable insights about the problem at hand. To enable this, we augmented the Activity Streams in this scenario with provenance information in the way shown in Listing 4: The listing represents an *invite* of "Maintenance Technician 152" (the object) that is published by the "Windpark Monitor" entity (the actor). The AS additionally carries provenance information using two relationships

from the PROV ontology:[8] the *prov:wasGeneratedBy* property holds information about the publisher (in this case, this is the same as the actor), and the *prov:wasDerivedFrom* property points to the AS that made the Windpark Monitor aware of the potential problem (i.e., the *critical* AS published by the turbine monitor). To support the technician during the maintenance processes, we implemented an Android application that, upon receipt of an *invite* activity, follows the *prov:wasDerivedFrom* links therein to trace the incident to the initial event that was published by the wind turbine and holds a link to the original data – this data is then displayed by the application.

VII. Future Work

Currently, the ASbase implementation is based on the existing stable AS1 format, but we are still actively developing new features of the platform. We expect to switch to AS2 as soon as that specification has reached a suitable level of stability. Our goal is to learn from real-life, industrial-grade applications, discover where a richer set of AS-based services would be beneficial, and extend the platform gradually to provide richer services over time.

Using the MongoDB general-purpose JSON query language serves our clients' purposes for the moment and brings with it powerful tools such as geospatial querying, but it is not operating at the level of abstraction that we would ideally like it to. Rather, our goal is to create an "AS Query Language" that operates with concepts that are meaningful for AS, abstracts from JSON peculiarities, and makes it easier for ASbase users to focus on AS semantics instead of the JSON syntax. We also plan to extend ASbase by integrating more protocols, such as WebSockets, into its subscription API.

On a more abstract level, we are very interested in extending the presented basic provenance model to further scenarios, for instance regarding the analysis activities in our personal health scenario: similar to the wind turbines example, this would allow doctors to trace specific analysis results about a patient's health back to the specific measurements that gave rise to these findings, which might be valuable during diagnosis processes.

VIII. Conclusions

In this paper, we presented our approach to using Activity Streams (AS) for providing the connective fabric between loosely coupled services. Originating in the social media space, we see AS as a good foundation for social enterprise and industrial IoT scenarios as well as use cases in professional healthcare. The openness and extensibility of AS allow to increase the richness of concepts that can be represented by them, and we presented a scenario where AS are used for integrating heterogeneous devices that collect health data, analysis services, and a visualization tool for health professionals within a single vertical silo. We furthermore showed their applicability to bridging such vertical silos in an experiment where multiple project groups used AS to interconnect initially independent implementations and form cross-vertical mashups. Finally, we demonstrated the possibility of enriching AS with provenance information that elicits cause/effect-relationships. We believe that AS provide a good mechanism to coordinate and synchronize collaboration across devices in an emerging (industrial) IoT. Within that space, the development of shared vocabularies is valuable for bridging applications across IoT domains, which should support the creation of a common AS description language.

While our interest in AS and in the specific scenarios is what triggered our research in the beginning, our current main focus is on ASbase, the platform that allows such AS-based scenarios to be built in simple, open, extensible, and decentralized ways. ASbase is still under development, but we hope that by applying it in a variety of scenarios, we will gather more experience with AS as a general foundation for a variety of social applications in different domains. We will extend ASbase with more functionality as required, where we are particularly interested in further experimenting with AS-embedded provenance relationships.

[8] http://www.w3.org/TR/prov-o/

References

[1] Guinard, D., Ion, I., Mayer, S.: In Search of an Internet of Things Service Architecture: REST or WS-*? A Developers' Perspective. In: Proceedings of the 8th International Conference on Mobile and Ubiquitous Systems (MobiQuitous). pp. 326–337 (2011)

[2] Guinard, D., Trifa, V., Mattern, F., Wilde, E.: From the Internet of Things to the Web of Things: Resource-Oriented Architecture and Best Practices. In: Uckelmann, D., Harrison, M., Michahelles, F. (eds.) Architecting the Internet of Things, pp. 97–129. Springer-Verlag, Heidelberg, Germany (May 2011)

[3] Nottingham, M., Sayre, R.: The Atom Syndication Format. Internet RFC 4287 (December 2005)

[4] Pautasso, C., Wilde, E.: Push-Enabling RESTful Business Processes. In: 9th International Conference on Service Oriented Computing (ICSOC 2011). Lecture Notes in Computer Science, Springer-Verlag, Paphos, Cyprus (December 2011)

[5] Snell, J.M.: Activity Streams 2.0. World Wide Web Consortium, Working Draft WD-activitystreams-core-20150722 (July 2015)

[6] Snell, J.M.: Activity Vocabulary. World Wide Web Consortium, Working Draft WD-activitystreams-vocabulary-20150722 (July 2015)

[7] Sporny, M., Kellogg, G., Lanthaler, M.: JSON-LD 1.0: A JSON-based Serialization for Linked Data. World Wide Web Consortium, Recommendation REC-json-ld-20140116 (January 2014)

[8] Zhu, D., Zhang, Y., Cheng, B., Chen, J.: Towards a Flexible Event-Driven SOA Based Approach for Collaborating Interactive Business Processes. In: IEEE International Conference on Services Computing, SCC 2011, Washington, DC, USA, 4-9 July, 2011. pp. 749–750 (2011)