A Communication Framework for the Internet of People and Things Based on the Concept of Activity Feeds in Social Computing

Thomas Vilarinho, Babak A Farshchian, Jacqueline Floch and Bjørn Magnus Mathisen
SINTEF ICT
SINTEF
Trondheim, Norway
{thomas.vilarinho , babak.farshchian, jacqueline.floch, bjornmagnus.mathisen}@sintef.no

Abstract— **Social networks connect people while internet of things platforms connect things. Although both platforms use various communication tools in efficient ways, platforms for things often don't communicate and interoperate with social networks. As a consequence, there is a lack of unified programming interfaces and platforms to enable internet-based interactions between people and things. This hinders deployment of services where people and things need to co-exist such as in ambient assisted living and collaborative sensing scenarios.**

We propose activity feeds as a unified communication framework to address this integration challenge. The activity feed concept is widely deployed in social networks and recently also in internet of things platforms. It is a flexible and easy-to-understand concept. We propose a communication framework based on activity feeds consisting of a set of concepts, a set of patterns for using activity feeds, an API for developing applications, and a reference implementation of the framework on smart phones.

We describe how we tested this framework in two different applications in the areas of ambient assisted living and crisis management. For these two applications we were able to easily deploy the framework by invoking the activity feed API. We report on how we use communication patterns based on activity feeds and how the framework managed to facilitate people and thing communication.

Keywords-component; Activity Stream; Activity Feed; Internet of Things; Social Computing; Ambient Intelligence; Ubiquitous Computing; Computer Supported Cooperative Work; Ambient Assisted Living; Crisis Management;

I. INTRODUCTION

Internet of Things (IoT) is upon us and already offers a number of new opportunities for using ICT in innovative ways in order to solve societal problems. Sensing systems allow us to sense environmental changes such as increased air pollution. Health- and lifestyle-related devices and sensors allow elderly and people with chronic diseases to live and function independently. At the same time these scenarios introduce highly sophisticated technical devices into our everyday lives. Seamless and transparent communication between our human systems and the technological infrastructure of IoT, that we are so increasingly depending upon, is crucial.

Traditionally in the distributed computing world, communication and information sharing technologies have been created for two different audiences: "Things" (e.g. devices and sensors) and “People”. Things have talked languages that people without specialized training have difficulties to understand, while people have been talking languages that things often don't understand. This issue was raised by Tim Berners-Lee et al. in the concept of the semantic web[1] where web pages are not only made for human consumption but also for processing, understanding and reasoning by computers. Though the semantic web in its outset had little to do with IoT and was aimed at intelligent software agents, the basic thesis of the semantic web has become even more relevant with the advent of IoT. Our day-to-day and minute-to-minute encounter with technological "things" needs a common lingua franca.

In our research we explore the affordances of physical things in extending the interaction possibilities among physically distributed groups of people. We believe that physical things, due to their specialized form factors and affordances, can offer rich and natural interaction mechanisms in many social interaction scenarios. Our aim is to use things to enrich the digital representations of people and things so that interactions in the digital world can become richer and more natural and resemble those in the physical world, in what we call Physically Embedded Social Interaction (PESI).

Through an iterative process of developing and evaluating a set of proof-of-concept prototypes we have developed a communication framework for PESI. This framework is based on well-known concepts and standards from social computing, especially the concept of Activity Feed that we use as the main means to support communication among people and things. This paper discusses the applicability of Activity Feeds as a basis for a communication framework for PESI. The rest of this paper is organized as follows. First we will introduce the main concepts of the PESI communication framework, and illustrate the concepts through two scenarios, one from crisis

978-0-7695-5038-1/13 $26.00 © 2013 IEEE
DOI 10.1109/IE.2013.24

management and one from ambient assisted living. We then describe the current implementation of this framework for Android devices called UbiShare. We compare our contribution to state of the art, and also shortly describe our research methodology. We conclude the paper with some direction for future research.

II. MAIN CONCEPTS AND USE CASES

The central concepts for our work are those of **Activity Feed** (shortly called a Feed in the rest of this paper) and **Activity**. Most internet users are familiar with feeds in various forms. Figure **1** shows a feed from a popular internet-based service. Activities are published continuously into a Feed and are often visually shown in a chronological order, with the latest Activity at the top of the Feed. This is done to allow the users to see the latest activities, while hiding activities that have already been seen by the user. In some cases, Activities are also displayed to the user according to other criteria, e.g. popularity and context in Facebook. In the next two sections we describe the concepts of Feed and Activity in more detail.

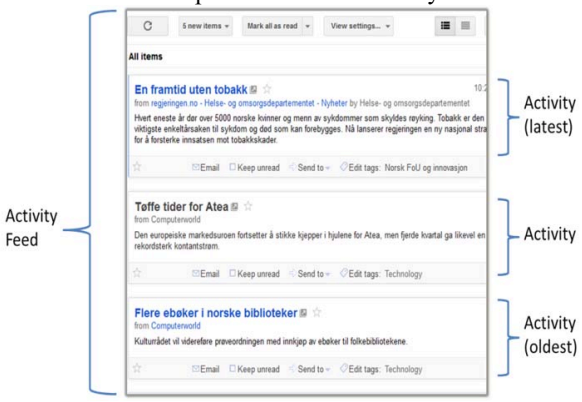


Figure 1: An example of an Activity Feed from the web.

A. *Activity Feeds*

Activity Feeds (also known as Activity Streams[2]) were popularized mainly by social networking applications as a means for showing the "latest news" from websites and social networks. Feeds have been implemented in different ways by different services, and partly standardized by the OpenSocial APIs[3] and the Activity Streams specification[2]. Various implementations have converged into the concept of a Feed as a chronological list of events related to an entity and which other entities can subscribe to (and possibly publish to). A Feed as used in our framework has the following properties:

- A Feed is assigned to one entity and is **owned** by that entity. In our framework these entities are currently people, things and communities (to be discussed later).
- A Feed has a universal resource identifier (**URI**), which means it can be accessed unambiguously on the internet.
- One entity can **publish** Activities to own or another entity's Feed. In the same way, one entity can **subscribe** to another entity's Feed and in this way receive notifications about Activities in that Feed.
- Activities are accumulated in a Feed and are kept as a **history**. This allows a Feed to function as a chronological context for the entity.

B. *Activity*

A Feed is created by entities publishing Activities into the Feed based on real world events. An Activity is a construct that provides information about the nature of the event and its context. We have based our definition of an Activity on the definition provided by the Activity Streams specification[2]. An Activity is constituted by the following parts:

- Actor: represents the entity that performs the action.
- Verb: represents the type of the action performed.
- Object (optional): represents the entity primary object (subject) of the action.
- Target (optional): the substantive to which the action refers.
- Provider (optional): the identity of the entity that publishes the activity on behalf of the Actor.

Figure **2** shows how a real world event "John sends a letter to Maria" maps onto an Activity.

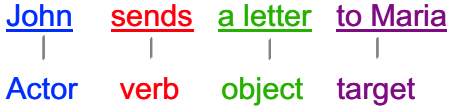


Figure 2: Activity example

The main motivation for defining a standard for Activities and Feeds (such as in OpenSocial) has been to facilitate exchange of Activities among various social computing sites. Our research builds on this line of development. In addition we use Activities and Feeds to support communication among people and things. We will see how this is done later as we describe some of our implementation examples.

C. *Persons, Things and Communities*

The entities in our framework currently belong to one of the three types of **Person**, **Thing** and **Community**. A Person represents a real human being, while a Thing represents a physical or digital artifact. A Community represents a collection of Persons and Things. All the entities, regardless of their type, have a Feed assigned to them (see Figure **3**). An entity can publish to its own Feed or publish to the Feed of another entity. In the same way an entity can subscribe to its own Feed or subscribe to another entity's Feed (publishing and subscribing are normally subject to access rights in an implementation of the framework). A Community is different from the two other entity types in that it also contains an explicit list of members that is visible to all the Persons and Things that are the members of the Community.

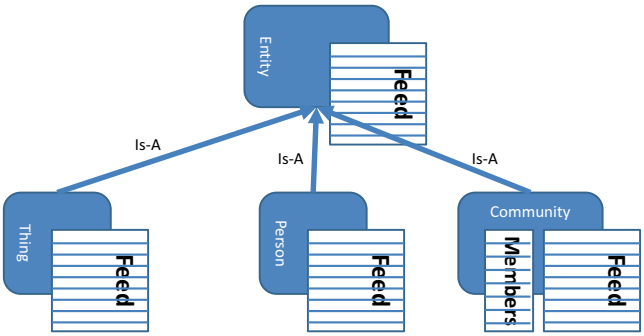


Figure 3: Entities in our framework are Things, People and Communities.

D. Example use cases

Based on the above concepts we describe here a set of example use cases that illustrate how the framework can be used to achieve the goal of implementing a light-weight communication framework for PESI.

1) Publishing to a feed as a means of sending a command

One of the most important use cases is to allow a Person or a Thing send a command to another Thing. The goal is to activate the receiving Thing to do something. In this case an Activity is created to contain the command where the verb is the desired action on the side of the receiving Thing. In Figure **4**.A the activity is "John (actor) /brew (verb) /coffee (object) /coffee machine (target)". This activity is then published to the coffee machine Feed triggering the machine to brew coffee.

2) Subscribing to a feed as a means of being notified about context changes

Subscribing in our framework allows an entity to be notified when an Activity is published in a Feed. This can be used to signal changes in context, which in turn can lead to actions. For instance, subscribing to the coffee machine Feed in the example above will notify the subscribing entity when the coffee is ready, i.e. when the coffee machine publishes an Activity into its own Feed (see Figure **4**.B).

3) Using a Community as a means to share context

Publishing or subscribing to a Community's Feed can be used to maintain a shared context for a group of Persons and Things. In this case entities join a Community (become members) in order to demonstrate interest in shared context. All members of a Community can publish to the Community's Feed, and all members are notified about published Activities. For instance, John can publish an Activity to the Feed of the Community created for his work team in order to inform members that coffee will soon be fresh-brewed. The Community is in addition responsible for providing social transparency by maintaining a list of members that is visible to all members.

4) Light-weight and loose composition

As we will see later in the example applications, Things can publish and subscribe in various combinations. The framework can be used as a light-weight composition mechanism. For instance, the coffee machine can publish an Activity to a bell and make the bell ring when coffee is brewed.

5) Using Feed history as a means to learn and personalize

All activities published to a Feed are in theory available for later inspection, analysis, merging, aggregation etc. This means that the Feed can be used as a source of learning about the entity and personalizing the behavior of the system according to the learned knowledge.

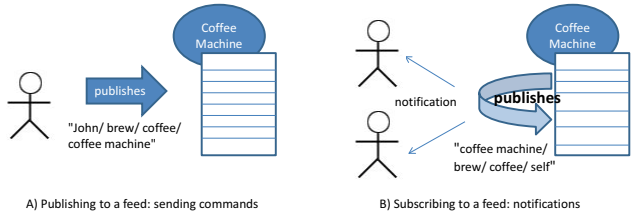


Figure 4: Two example use cases for the framework.

III. Applications of the Framework

In this section we describe two applications of our communication framework. These applications illustrate how the framework is used in two realistic scenarios, and allow us to some extent answer the following two questions:

- Can we use concepts from social computing, in particular the Activity Feed concept, to implement a light-weight communication framework for PESI?
- Will this framework make it easier to develop new applications for PESI?

The applications described here are from two very different domains. The first application is from crisis management domain where communication among the members of a rescue team is supported using a smart jacket. The second scenario is from ambient assisted living domain, where a fall detection application is used to support the inclusion of informal caregivers in fall management. The fall itself is detected using sensors connected to a smart phone. Both applications exist in form of prototypes implemented in the Android operating system on top of UbiShare, which is an implementation of our communication framework for Android-based mobile devices (UbiShare will be described later).

A. Supporting Rescue Work

A major challenge for rescuers operating in a crisis field is to support cognitively demanding tasks with the use of communication and collaboration tools. For instance, even the most user-friendly mobile devices today require full attention when interacting with them. But such full attention cannot be expected from a rescuer in the field. To address this challenge, we explored physical user interfaces that can be integrated in a smooth and non-obtrusive manner with the rescuer's physical environment. A smart jacket that we have developed[4] is an example of such an interface. The smart jacket, called iJacket, is equipped with a number of actuators: an LCD display, a

vibrator and a loudspeaker. iJacket allows the coordinator of a rescue team to draw the attention of the rescuers in the team and to provide them with information and commands. Using a normal jacket as the platform we avoid distracting rescuers from their work. iJacket communicates with Android smart phones using Bluetooth, and is implemented using the Arduino hardware platform for physical prototyping[5].

Our application for rescue work consists of two parts, iJacket and iDisaster (see Figure **5**). iDisaster is an application that allows team coordinators to create and organize rescue teams. iDisaster also supports interactions among team members by sending text messages and using iJacket for communication. iJacket, supports unobtrusive communication among rescuers by mediating text messages sent to rescuers. When the coordinator sends a text message to the rescuers, iJacket's vibrator and loudspeaker are activated and the message is shown immediately on the LCD display.

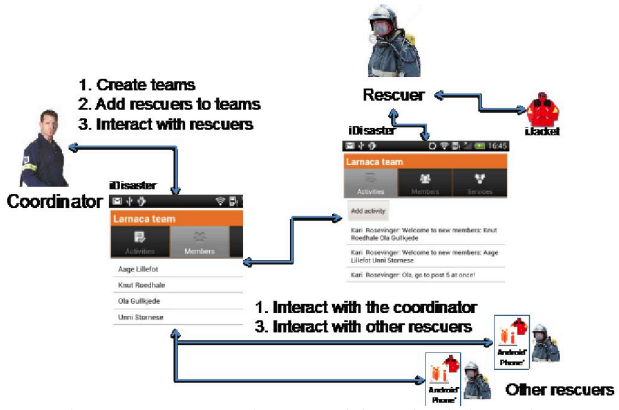


Figure 5: Rescue scenario: SW modules and main interactions.

Figure **6** shows how our communication framework is used to support this application. The first step is for the coordinator to create a rescue team consisting of the coordinator and rescuers (step 1 in Figure **6**). Each rescuer has an iJacket that subscribes to the rescuer's Feed. When the coordinator sends a text message, this message is published in form of an Activity in the Community's Feed (step 2) and notifications are sent to rescuers who are members of the team (step 3). When the notification is received, iDisaster publishes a new Activity in the rescuers' Feed (step 4) which triggers a notification to the iJacket of the rescuer(s) to which the notification is addressed (step 5). iJacket does its job (vibrating, sounding alarm and displaying the message), and publishes an Activity to its own Feed (step 6). Note that publishing Activities to Feeds is a convenient way of notifying all the interested People and Things (i.e. sharing context information). For instance, the iJacket Feed might trigger a notification back to the Community (not shown in the figure) in this way letting the coordinator know that messages were received. In addition, using Activity Feeds as a publish/subscribe hub enables loose coupling among people and things. For instance, if the rescuer adds a new Thing, such as a helmet with a wireless headset, the Thing can be integrated easily by creating a subscription relation between the person and the helmet. Moreover, it is also possible (not implemented in our application) to have Things that are associated with the community. For instance a satellite surveillance Thing could be added directly as a member of the Community and be accessible by all the members of the team.

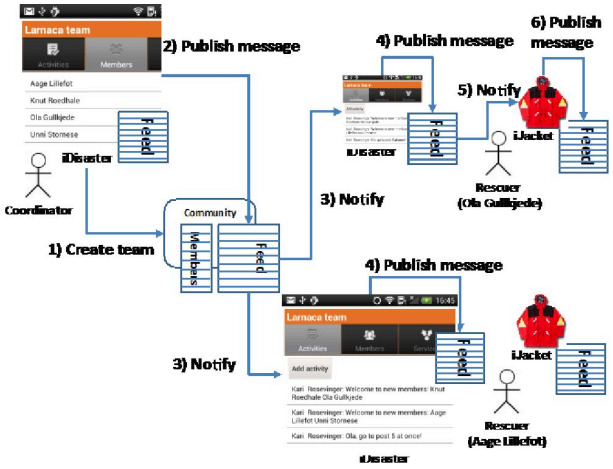
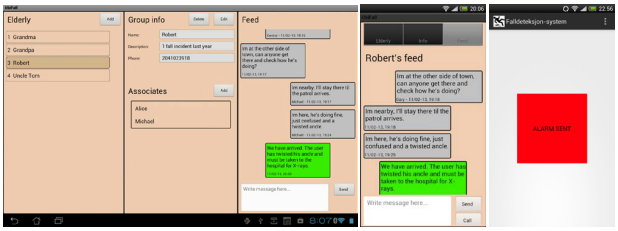


Figure 6: Using the framework in the rescue application

B. *Managing Falls Among Elderly*

Our next example application is from the domain of ambient assisted living. Falls are one of the most serious incidence groups among elderly. Falls often result in hip fractures. In most cases the injury results in permanent disability. Falls are psychologically demanding, both for the victim and for the informal network of family members and friends.

Our application for fall detection, called UbiFall (see Figure **7**), is developed based on the assumption that detecting falls at e.g. home can happen faster and more efficiently if the informal social network of the person at risk is involved in the process of fall management[6]. UbiFall allows a fall manager (e.g. in a monitoring center in a hospital or care center) to register a person at risk of fall and associate a group of family members or friends to that person. This is shown in Figure **7**.A. The left-most part of the window here shows the list of the elderly that are monitored by the monitoring center (with number 3 Robert selected). The middle part shows information about the person being monitored (Robert). At the bottom of this middle part is the list of people added as the members of this person's informal network (Alice and Michael). The right-most window in Figure **7**.A shows an ongoing communication among the involved people regarding the situation after a fall sensor has sent a message that a fall is detected.



A) UbiFall for Fall Manager B) UbiFall for the Network C) UbiFall for Elderly

Figure 7: The Fall detection application: screenshots

Figure 7.B shows UbiFall when used by the members of the informal network of the person at risk. Here the user sees messages sent by the fall manager, the sensors and the other members of the network. The person at risk has a much simpler user interface to UbiFall (Figure 7.C). This interface is mainly a large button that can be pushed to prevent an initiated fall alarm from being sent. The elderly person can push this button in cases when the sensor creates a false alarm.

Figure 8 shows how our communication framework is used to support the implementation of UbiFall. Each person involved (Fall Manager, Daughter, Neighbor and Elderly) is represented by a Person with a dedicated Feed. In addition, the sensor on Elderly's smart phone is represented as a Thing with its own Feed. As the first step, Fall Manager creates a Community to represent the network of the people involved in fall management (step 1 in Figure 8). Shortly after the network is set up, the fall sensor detects a fall. The sensor publishes a "sensor/detect/fall" Activity to its own Feed (step 2), which generates a notification to the Elderly application (step 3). Once Elderly UbiFall application receives the notification it allows Elderly to prevent the alarm from being sent (e.g. if the fall sensor generated a false alarm). If the alarm is not prevented (i.e. Elderly has really fallen) the Elderly application publishes an "elderly/trigger/alarm" Activity to the Feed of its network (step 4). The network notifies all the people in its member list (step 5). The alarm Activity shows up in the Community Feed visible to community members (as show in Figure 7.B) and can trigger further actions. The members of the network can continue publishing messages or other types of Activities into the Community Feed in order to coordinate their actions.

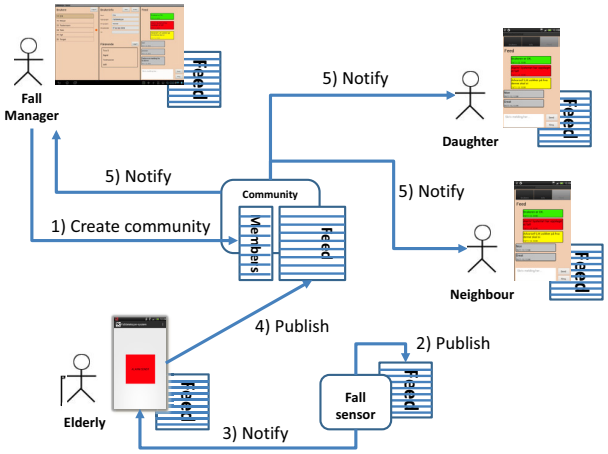


Figure 8: Using Feeds in the Fall Detection application

The above two applications show how the basic patterns of publishing and subscribing to feeds can support fairly complicated scenarios involving many different actors. The applications also show that the same concepts are used for People, Things and Communities, allowing a uniform set of concepts and simplifying application development. In the next section we will introduce UbiShare, a reference implementation of the PESI communication framework.

IV. REALIZATION

We have created a reference implementation of the PESI communication framework for Android smart phones with a set of APIs for creating and managing Persons, Things, Communities, Feeds and Activities. The goal for this reference implementation is to validate the concepts and to investigate whether we can simplify the development of applications such as the ones described in the previous section. The above two applications are implemented on top of this reference implementation as shown in Figure 9 and described below.

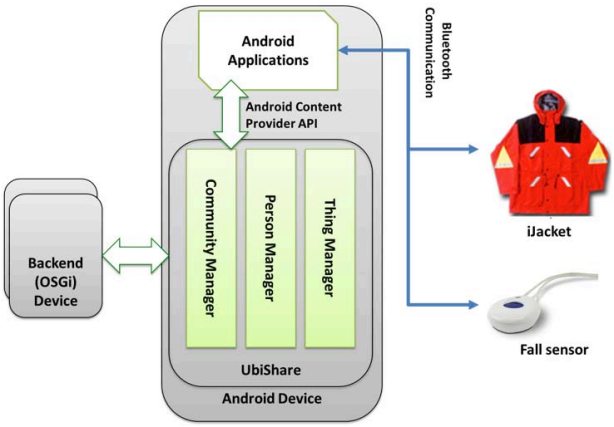


Figure 9: The reference implementation for the framework.

Here we describe mainly the part of the reference implementation for smart phones since this part is most relevant for the described applications. Framework concepts such as Persons, Things, Communities, Feeds and Activities

are managed on smart phones by a component called UbiShare[7] (see Figure 9 the middle part). UbiShare utilizes the content provider concept in Android[8] in order to provide standardized access to a data model consisting of the main framework concepts. A content provider is the Android standard mechanisms to encapsulate access to shared structured data. A content provider provides access to data using a simple CRUD (Create, Retrieve, Update, and Delete) API. It is easy for an Android programmer to create applications on top of a content provider due to its simple and standardized API. The API provided by UbiShare allows an application programmer to:

- Create and manage Persons, Things and Communities.
- Create and manage membership of Persons in Communities.
- Create and manage membership of Things in Communities.
- Publish Activities to any type of Feed.
- Get notified when an Activity is published to a Feed.
- Query and display all the data related to the communication framework.

The data that is created and stored in UbiShare is synchronized with back-end services (the left part of Figure 9) using Android's standard mechanism called a sync adapter. It is possible to create sync adapters for multiple back-end services, allowing UbiShare's data (or parts of it) to be synchronized with various services. Currently we have two synchronization mechanisms. One sync adapter uses the XMPP protocol [9] to synchronize data towards an OSGi[10] backend. A second sync adapter uses a cloud-based file system (Box.com) for synchronization.

Communication with Things such as iJacket and fall sensors is implemented by creating an Android-based proxy application for each Thing (shown as "Android Applications" in Figure 9). In this way the Things can get access to the API provided by UbiShare, get their own private Feed, and subscribe and publish to other Feeds. Communication with the physical Thing itself can be done using any communication mechanism available on the device. Our current implementations of iJacket and fall sensors are based on Bluetooth, which was our preferred choice due to the wide-spread availability of Bluetooth on mobile and embedded devices. Currently the iJacket and the fall sensor used in our applications are implemented using the Arduino physical prototyping platform[5]. We have implemented a dedicated Bluetooth library that allows wireless communication between an Arduino and an Android application[11].

Developing an application on top of UbiShare requires a few steps to be taken by the developers:

- Defining an Activity glossary: Activity types are defined by application developers and depend on what the developers consider to be meaningful actions performed on allowed types of Persons, Things and Communities. For instance, each of the application domains of rescue work and fall management will have their own glossary of Activities. The developer has to define the meaning of the Activities. In particular a glossary of verbs and objects should be defined. We anticipate that such glossaries will be reused as more applications are built using our framework.
- Configuring Feed settings: the developer has to configure the Feed settings for the Persons, Things and Communities in his/her application. This means deciding which entities can subscribe/publish to the entity's Feed and how the entity will publish/subscribe to other entities' Feeds. This is an important step, as the Feeds to be used will be the communication channels to/from each entity. As we have seen in the case of rescue and fall management applications, different options exist when using the publish/subscribe patterns.
- Using the API to publish/subscribe to the feeds: The developer has to incorporate in his entity's source code the API calls to publish/subscribe to the proper Feed. Publishing to a Feed requires calling an insert method on the content provider API. Subscribing to a Feed requires implementing a so-called "content observer" (callback) and processing notification messages.
- Translating Activities into actions: The remaining work for the developer consists of filtering and parsing incoming or outgoing Activities in accordance with the glossary defined in step 1, and performing the actions that match the Activities.

V. Related Research

Communicating with things through seemingly humane means is not limited to science fiction or AI-based robotics. For quite a long time people have been communicating with their things using the same ICT-based tools that they use to connect with their colleagues and friends. Early examples include software engineering teams, where instant messaging or email tools are used to communicate with continuous integration systems. So it is not uncommon to have Jenkins (a continuous integration tool) and Nabaztag (a cute rabbit pet) as your "friends" in Jabber or Gtalk, and have them talk with you about build failures[12]. With the advent of social networks as a major means of online communication among people, we also see a growing number of initiatives trying to use similar techniques (e.g. feeds) to interconnect things. For instance, Ericsson's social web of things[13], SenseFace[14] and the Twitter applications reported in[15] are all attempts to integrate data from physical sensors with online social networks. Our work builds on the experience from this type of research, and attempts to take this research a step further by building a unified framework and APIs for integration of people and things. In fact, most of the reported work in this area focuses on building a separate internet of things (albeit using techniques from online social media) while our aim is in

addition to investigate the effect of mixing people and things in hybrid social networks.

Activity feeds in particular have gradually become an important part of any social media. The idea of having a data feed for things is used in a number of IoT platforms. One popular example is Cosm[16], a commercial platform that allows its users to assign a web address to their things such as sensors and devices. The web address works a lot like a feed, and can be used to query sensor data, receive notifications, and provide an RSS (Really Simple Syndication) feed to the data collected from the sensor. Similar other platforms include open sen.se[17], Evrythng[18] and ThingSpeak[19], all of which provide some form of feed to access sensor data. These platforms are all underpinning our research, and provide a solid platform for the realization of our conceptual framework. However, we need to keep in mind that these platforms are mainly developed for things and not people. In particular the concept of a feed used in these platforms is often specialized in the direction of connecting and controlling things and not for communication between/among humans and things. We hope our framework can be used to guide the further development of such platforms and help experimentation with thing-human integration. In fact, our future research includes integrating the Thing Feed in our framework with some of these platforms.

In general, we can claim that our research advances state of the art 1) by focusing on the integration of human-thing communication and not only on connecting things to social media, and 2) by attempting to develop a unified framework and APIs that can be used by application developers to support human-thing communication.

VI. Conclusions and Future Work

In this paper we have presented a framework for PESI based on using a familiar concept from social networking services, the activity feed. We have defined the framework and have demonstrated its utility using two different sets of applications from the domains of rescue work and elderly fall management. We have also described UbiShare, a reference implementation of the framework demonstrating a set of APIs used to build the applications. All the applications and UbiShare itself are available as open source implementations and can be downloaded from GitHub [20] or obtained by contacting the authors.

The work presented in this paper demonstrates the feasibility of using the same framework of concepts (and the same set of APIs) to develop applications consisting of things, people and communities of things and people. The advantages of using activity feeds as the basic concept are many. Activity feeds are supported by many existing social computing and IoT platforms, which makes them ideal as a tool for integration. As demonstrated here and in related research, feeds can also be used as a shared language understandable by both people and things. The historical data about events stored in feeds can be used to perform advanced learning and personalization tasks (see e.g. [21]). We believe a unified framework can accelerate the development of a number of interesting applications where people and things need to co-exist.

We have not yet validated whether programming efforts on the side of application developers will be reduced by using our framework. This is one important direction for our future research. Achieving this goal will depend on how good the APIs and the implementations of the framework will be. But we can informally notice that the number of concepts that the programmers need to know is considerably reduced. Our future work consists of refining the framework and its APIs by developing new applications that can demonstrate its utility. The framework in its current form is quite generic. We need more experience in using the framework in order to discover the most useful patterns and the most efficient ways of developing applications.

Regarding the implementation of the platform, our future work will focus on implementing a number of new sync adapters for some of the IoT platforms discussed in Related Research. We will also continue simplifying application development by providing utility skeleton classes for different types of applications.

Acknowledgement

The work presented here is supported by the European R&D projects SOCIETIES and FARSEEING. We thank all our colleagues in these two projects for fruitful discussions. iJacket was first developed by Group 10 in subject IT2901 at NTNU[22] and later refined by us. Box.com synchronization for UbiShare is being implemented by MSc student Kato Stølen at NTNU. UbiFall applications are developed by MSc student Andreas Lund at NTNU. We thank the members of Group 10, Kato and Andreas for their efforts.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[2] Diso Project, “Activity Streams - a format for syndicating social activities around the web,” 2013. [Online]. Available: http://activitystrea.ms/. [Accessed: 03-Mar-2013].

[3] M. Häsel, “Opensocial: an enabler for social applications on the web,” *Communications of the ACM*, vol. 54, pp. 139–144, Jan-2011.

[4] M. Divitini, B. A. Farshchian, J. Floch, B. M. Mathisen, S. Mora, and T. Vilarinho, “Smart Jacket as a Collaborative Tangible User Interface in Crisis Management,” in *Proceedings of the Workshop on Ambient Intelligence for Crisis Management*, Pisa, Italy, 2012.

[5] M. Banzi, *Getting started with Arduino*. Beijing; Cambridge: Make:Books / O’Reilly, 2009.

[6] A. Lund, “UbiCollab: Twitting Falls- UbiFall,” Norwegian University of Science and Technology (NTNU), Trondheim, Norway, TDT4501, 2012.

[7] UbiCollab.org, “UbiShare,” *GitHub*, 2013. [Online]. Available: https://github.com/UbiCollab/UbiShare. [Accessed: 03-Mar-2013].

[8] Android Developers portal, “Content Provider Basics,” 2013. [Online]. Available: http://developer.android.com/guide/topics/providers/content-provider-basics.html. [Accessed: 05-Mar-2013].

[9] The XMPP Standards Foundation, "The Extensible Messaging and Presence Protocol (XMPP)," 2013. [Online]. Available: http://xmpp.org/about-xmpp/. [Accessed: 03-Mar-2013].
[10] R. Hall, K. Pauls, S. McCulloch, and D. Savage, *OSGi in Action: Creating Modular Applications in Java*, 1st ed. Manning Publications, 2011.
[11] A. B. Eie, H. Goldsack, J. Jansen, A. Lucassen, E. Di Santo, J. Svarvaa, and B. H. Wold, "oSNAP- Open Social Network Arduino Platform," Norwegian University of Science and Technology (NTNU), Trondheim, Norway, IT2901 final report, 2012.
[12] Jenkins developer community, "Nabaztag Plugin for Jenkins," 2013. [Online]. Available: https://wiki.jenkins-ci.org/display/JENKINS/Nabaztag+Plugin. [Accessed: 03-Mar-2013].
[13] M. Alendal, "The social web of things – a social network for your devices - Ericsson." [Online]. Available: http://www.ericsson.com/thinkingahead/idea/110217_social_network_for_you_1968920151_c. [Accessed: 03-Mar-2013].
[14] M. A. Rahman, A. El Saddik, and W. Gueaieb, "SenseFace: A sensor network overlay for social networks," in *IEEE Instrumentation and Measurement Technology Conference, 2009. I2MTC '09*, 2009, pp. 1031 – 1036.
[15] M. Demirbas, M. A. Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu, "Crowd-sourced sensing and collaboration using twitter," in *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, 2010, pp. 1 –9.
[16] Cosm, "How Cosm Works," 2013. [Online]. Available: https://cosm.com/how_it_works. [Accessed: 03-Mar-2013].
[17] Sen.se, "Feel, Act, Make sense • Sen.se," 2013. [Online]. Available: http://open.sen.se/. [Accessed: 03-Mar-2013].
[18] Evrythng, "About | EVRYTHNG," 2013. [Online]. Available: http://www.evrythng.com/about-us/. [Accessed: 03-Mar-2013].
[19] ThingSpeak, "Internet of Things - ThingSpeak," 2013. [Online]. Available: https://www.thingspeak.com/. [Accessed: 03-Mar-2013].
[20] UbiCollab.org, "UbiCollab@GitHub," 2013. [Online]. Available: https://github.com/UbiCollab/. [Accessed: 05-Mar-2013].
[21] J. Freyne, S. Berkovsky, E. M. Daly, and W. Geyer, "Social networking feeds: recommending items of interest," in *Proceedings of the fourth ACM conference on Recommender systems*, New York, NY, USA, 2010, pp. 277–280.
[22] UbiCollab.org, "Your jacket can now talk to Facebook!," 2012. [Online]. Available: http://ubicollab.org/z/2012/05/25/your-jacket-can-now-talk-to-facebook/. [Accessed: 05-Mar-2013].
[1]