Hypermedia-driven Socio-technical Networks for Goal-driven Discovery in the Web of Things

Andrei Ciortea, Antoine Zimmermann, Olivier Boissier
Univ. Lyon, MINES Saint-Étienne
CNRS Lab Hubert Curien UMR 5516
F-42023 Saint-Étienne, France
{andrei.ciortea, antoine.zimmermann, olivier.boissier}@emse.fr

Adina Magda Florea
Department of Computer Science
University "Politehnica" of Bucharest
313 Splaiul Independenței
060042 Bucharest, Romania
adina.florea@cs.pub.ro

ABSTRACT

To cope with dynamic environments, Internet of Things (IoT) applications are expected to autonomously discover and interact with services at runtime in pursuit of design or user-specified goals. On the one hand, various paradigms and technologies are available to program goal-driven autonomous software agents, and on the other hand hypermedia-driven environments are central to the development of robust machine-to-machine applications. However, existing approaches for the development of hypermedia-driven environments fall short of meeting the needs of autonomous agents: they either severely restrict the agents' autonomy, or their topological structure is either fragmented or inefficient to navigate at scale. In this paper, we explore the use of *socio-technical networks*, that is networks of people and things interrelated in a meaningful manner via typed relations, as an overlay for enhancing hypermedia-driven interaction in IoT environments. We present a proof of concept and discuss several classes of applications in which this model could prove useful.

CCS Concepts

•**Information systems** → **RESTful web services; Service discovery and interfaces;** *Social networks;*

Keywords

Web of Things; hypermedia APIs; socio-technical systems; autonomous agents

1. INTRODUCTION

The World Wide Web provides people with an Internet-scale hypermedia-driven environment that they can navigate and act on in pursuit of their goals. An example commonly used to illustrate these concepts is purchasing a book on the Web [15]: the buyer goes to a bookseller's website, looks up the book, adds it to her digital shopping cart and proceeds to checkout. The URI of the bookseller's website is the buyer's entry point into the environment. At each step of the way, hypermedia provides *local guidance* by advertising new possible steps to be taken, and the buyer's goal provides *global guidance* by serving as a "utility function" for selecting the next most useful step (e.g., to buy a book, it is more useful to add it to your digital shopping cart than it is to navigate away).[1]

The automation of various tasks, such as the one presented above, has long been envisioned on the Web [2]. Software clients that can operate with minimal human intervention are further motivated in the context of dynamic Internet of Things (IoT) environments [12, 16]. Multi-agent research already offers multiple languages and tools for programming goal-driven behaviors that can provide *global guidance* to software agents. In a recent publication, we demonstrated the successful transfer of multi-agent technology to IoT application development [7]. What is missing, in contrast to the book shopping example, is an Internet-scale hypermedia-driven environment that can provide *local guidance* to software agents. This environment should support:

R1 *heterogeneity*: the environment should span across application domain silos;

R2 *effective crawling*: software agents must be able to navigate the environment in an informed manner;

R3 *autonomy*: software agents must be able to act on the environment in an autonomous and reliable manner.

We consider that a necessary step towards this vision is to search for general models that enable software agents to reliably and effectively participate in large hypermedia-driven environments.

In this paper, we explore the use of *socio-technical networks (STNs)* [6], that is networks of people and things interrelated in a meaningful manner via typed relations, as an overlay for enhancing hypermedia-driven interaction in IoT environments. Section 2 discusses related work on hypermedia-driven environments. Section 3 presents our general approach. Section 4 presents a proof-of-concept. We suggest several classes of applications in which STNs could be useful in Section 5.

2. RELATED WORK

For the purpose of our discussion, we conceive of a hypermedia-driven *environment* as a dynamic set of resources and the relations among them. The environment is exposed to software clients by one or more hypermedia-driven *interfaces*.

[1]We adopt the terms *local* and *global guidance* from Simon Mayer's dissertation (Section 7.2.5) [15].

WoT'16, November 07-09, 2016, Stuttgart, Germany

DOI: http://dx.doi.org/10.1145/2991561.2991573

Several approaches to develop hypermedia-driven interfaces are available. Some of these approaches are general-purpose, in the sense that they do not impose constraints on the resources and relations they expose. We discuss general-purpose approaches in Section 2.1. Other approaches do impose constraints on exposed resources and relations, such as the use of specific types, and with each constraint they imprint additional features on the resulting environments. We discuss these approaches in Section 2.2.

2.1 Hypermedia-driven Interfaces

Hydra [13] defines an approach to develop hypermedia-driven interfaces by means of a vocabulary that servers can use to advertise state transitions to clients via hypermedia controls. The hypermedia controls can augment resource representations, or they can be advertised in a machine-readable API documentation discoverable via the HTTP `Link` header field of retrieved responses.

An approach similar to some extent to Hydra is taken by the Thing Description (TD) currently being developed in the W3C WoT Interest Group[2]. Two important differences from Hydra are that (i) the TD goes beyond the request/response interaction pattern by modeling events, and (ii) the TD is designed with a clear separation of interaction semantics from the underlying protocol and with an emphasize on the diversity of protocols in the IoT landscape (e.g., CoAP, Bluetooth Low Energy, MQTT). Note, however, that both the TD and the Hydra Core vocabulary are actively being developed at the moment of writing this paper.

RESTdesc [27] is another approach for describing hypermedia-driven APIs that attaches functionality to hyperlinks via inference rules defined as a set of preconditions, a request to be performed, and a set of postconditions. RESTdesc was designed with a focus on software agents, who can use the descriptions with a reasoner in order to determine sequences of requests intended to achieve goals, such as buying a book on the Web.

The above approaches have complementary purposes and could potentially be used in conjunction: Hydra is designed for hypermedia-driven interfaces in general, the TD is designed to account for WoT-specific requirements, and RESTdesc is designed with a focus on high-level planning for software agents.

2.2 Hypermedia-driven Environments

In contrast to the general-purpose solutions presented in the previous section, some existing approaches to develop hypermedia-driven interfaces impose constraints on the resources and relations they expose, which leads to environments with topological structures that fall broadly in one of two categories: *directories* (or *containers* or *catalogues*) *of resources* and *graphs of services*.

2.2.1 Directories of resources

In constrained RESTful environments (CoRE), devices use the CoRE Link Format [24] to advertise their resources under a standard endpoint, or to register them to a CoRE Resource Directory [25]. The CoRE Link Format defines the `hosts` relation type, which provides a forward path from a server to a hosted resource. It also allows for the definition of other relation types. In addition, the standard defines an attribute, among others, for attaching application-specific semantic types to resources. We thus conclude that software agents can conceive of a constrained Web server as a kind of "logical directory" (not to be mistaken with a CoRE Resource Directory) that has a flat hierarchy (i.e., one level in the case of the `hosts` relation) and contains *semantically typed resources*. To the best knowledge of the authors, there are currently no standard relation types that would enable software agents to navigate from one logical directory to another. Software agents can act on a CoRE, for instance by publishing new resources to a CoRE Resource Directory.

The Linked Data Platform (LDP) [26] takes a purely data-driven approach to hypermedia-driven interaction, which is similar in some respects to the one taken in CoRE. The LDP defines a protocol for reading and writing linked data on the Web. This protocol is based on HTTP and RDF: it defines conventions for LDP resources, whose state is represented in RDF, and further specifies the HTTP operations that can be performed on those resources. In doing so, it relies on a data model defined in terms of *containers* and *resources*, where containers are resources that can contain other resources. An LDP resource is created inside an LDP container, and thus LDP servers enforce a tree-like structure on their resources. LDP resources are explicitly linked to LDP containers via *containment* and/or *membership* relations, depending on the type of container. The LDP normative requirements are very precise with respect to the lifecycle of LDP resources and enforce hypermedia-driven interaction via server-managed relations (e.g., containment relations are automatically added at resource creation). Hypermedia-driven interaction across LDP servers is not standardized. We conclude that software agents can conceive of an LDP-based environment as a forest of tree-like structures of resources. The environment is fully represented in RDF and software agents can act on it in a reliable manner. Membership relations can be extended with domain- and application-specific knowledge, which enables informed navigation.

HyperCat [4] defines a JSON-based hypermedia catalogue format for collections of URIs, and is very similar to the LDP in several aspects. For instance, a HyperCat *catalogue* contains *items*, where both catalogues and items are described through RDF-like triples. An item can be any URI together with its associated metadata, which implies that catalogues can contain URIs to other catalogues. HyperCat is also based primarily on HTTP, although other protocols can be used. However, HyperCat is different from the LDP in two aspects that are important in the context of our discussion. First, given that items in HyperCat catalogues can be any URIs, HyperCat does not enforce a tree-like structure on its collections: catalogue A can contain a link to catalogue B, and catalogue B can contain a link to catalogue A. Second, the client does not have control over in which catalogue an item is finally registered, the server may organize new items in any pattern it wishes (and must return the URI of the catalogue to which the item was registered).[3] Creating new catalogues is not specified and out of the scope of the HyperCat specification. The HyperCat specification also provides several extensions, for instance, for search and subscription mechanisms. We conclude that software agents can conceive of a HyperCat-based environment as a graph of resources in which an edge denotes a generic containment relation from a directory to a contained resource. The containment relation is implicit and cannot be extended with domain- or application-specific types, and thus software agents cannot navigate the environment in an informed manner. Given that the behavior of a HyperCat server is non-deterministic, software agents cannot act on the environment in a reliable manner.

2.2.2 Graphs of services

Another model for hypermedia-driven environments is given by *graphs of services*. Such is the case of the *computational marketplaces* proposed in [15, 17]. In this approach, developers publish hyperlinks between related services to create graphs that *mashup clients* can then navigate using various preprogrammed strategies. The hyperlinks are annotated with meta-information (e.g., forward

[2] http://w3c.github.io/wot/current-practices/wot-practices.html, Accessed: 28.09.2016.

[3] HyperCat 3.00 Specification: https://drive.google.com/file/d/0B5JKRgbs8OylcmdvMmlGYmtVTU0/view, Accessed: 28.09.2016.

path name, service cost) to provide *local guidance* to clients in their traversal. To enable *global guidance*, the authors propose to annotate hyperlinks with the name of the service mashup traversed by the client.

It is worth to note that graphs of services can be seen at a higher-level of abstraction than graphs of resources, and thus can be added as an overlay of "informational highways" to guide software agents more effectively in massive hypermedia-driven environments. However, if global guidance annotations are used, the autonomy of software agents is severely restricted: there is no longer a clear separation between the environment and the logic that navigates and acts on it. The use of graphs of services without global guidance annotations, but with more complex software agents than the ones introduced in the initial approach seems promising and requires further investigation.

We conclude our discussion of related work with three main observations. First, we note that there are multiple (and diverse) approaches to develop hypermedia-driven interfaces. Second, when considering large-scale hypermedia-driven environments, most existing approaches lead to environments that are either fragmented, cannot be navigated in an informed manner, or cannot be easily extended with domain- and application-specific knowledge. Third, with the noticeable exception of the LDP, software agents generally have little support to act on hypermedia-driven environments. In HyperCat-based environments, the result of their actions is also non-deterministic.

3. HYPERMEDIA-DRIVEN SOCIO-TECHNICAL NETWORKS

In the following, we first present briefly the core abstractions introduced by our model for *socio-technical networks (STNs)*, which was formally defined in [6], and how they address the requirements in Section 1. Then, we present an approach to apply the STN model to the Web.

3.1 Networks of Agents and Artifacts

STNs are dynamic networks of *agents*, *artifacts* and other entities interrelated in a meaningful manner via *typed relations*. We use STNs to model IoT ecosystems: people and things that are actively trying to influence the state of their environment are modeled as *agents*, whereas things that passively augment the environment with new capabilities (i.e., are part of the environment) are modeled as *artifacts*. Agents and artifacts thus provide a clear separation between the logic that manipulates the environment and the one that augments it.

The *agent* and *artifact* abstractions have their roots in the *Agents & Artifacts meta-model* [18], a well-known paradigm in multi-agent research. This alignment enables the transfer to IoT application development of existing languages and tools for programming goal-driven agents and multi-agent systems, as we have shown in [7]. In the same time, we chose to restrain the alignment to only these two core abstractions in order to maintain the generality of the STN model.

Typed relations enable agents to navigate STNs in an informed manner (cf. R2). The STN model defines a generic *social relation* between agents, which can be further extended with domain- and application-specific relation types (cf. R1). Agents can manipulate their relations in order to "rewire" their networks in pursuit of their goals (cf. R3). The STN model also specifies other operations that agents can perform, such as sending messages or creating groups. The set of operations can be further extended to fit domain- and application-specific requirements (cf. R1).

The state of an STN is reflected in the digital world by means of *digital artifacts* that are *hosted* by *platforms*. The hosting relation enables the discovery of platforms and any implementation-specific knowledge that software agents would require in order to interface with the platforms in a reliable and polite manner.

3.2 Socio-technical Networks on the Web

To deploy STNs on the Web, we implement digital artifacts as *information resources* [11]. In doing so, we follow the REST architectural style to identify any mismatches with the Web architecture, and we build a uniform RDF representation of STN-based hypermedia-driven environments using linked data [1] and the *STN ontology*[4], a Web ontology for describing STNs and HTTP bindings of STN operations.

We structure the presentation of our approach following the three orthogonal pillars of the Web architecture [11]: *identification*, *formats*, and *interaction*.

3.2.1 Identification

We use URIs to identify entities in an STN such that dereferencing an entity's URI always returns a useful representation, for instance in RDF (cf. linked data principles [1]). Since real-world entities (e.g., people) cannot be represented as information, we generally use *hash URIs* [22] that dereference to digital artifacts representing those entities (e.g., user accounts).

For illustrative purposes, Listing 1 shows an RDF representation of David's user account: David is identified by a hash URI, he is described as an `stn:Person`, he holds multiple user accounts, and David's URI dereferences to the user account in his home STN.

Listing 1: This listing shows an RDF description of David, who holds user accounts on Facebook, Twitter and his home STN. David also owns a social TV. We use examples of URIs for legibility.

```
@prefix stn: <http://w3id.org/stn/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://home1.example.org/david#me> a stn:Person ;
  stn:holds <http://home1.example.org/david> ;
  stn:holds [ a stn:UserAccount ;
      stn:hostedBy <http://facebok.example.org/#platform> ;
      stn:id "1550387481863557" ] ;
  stn:holds [ a stn:UserAccount ;
      stn:hostedBy <http://twitter.example.org/#platform> ;
      stn:id "daviddoe" ] ;
  stn:owns <http://home1.example.org/tv#thing> .

<http://home1.example.org/david> a stn:UserAccount ;
  stn:name "David Doe"^^xsd:string ;
  stn:description "An IoT enthusiast!"^^xsd:string ;
  stn:connectedTo <http://home2.example.org/bob> .

<http://home1.example.org/tv#thing>
  a <http://example.org#SocialTV> .
```

URIs allow digital artifacts to be referenced globally and independent of context. Most existing Web APIs, however, use implementation-specific resource identifiers, such as David's user accounts on Facebook and Twitter in Listing 1. To cope with this architectural mismatch, we encapsulate in the references to locally-identified digital artifacts the URIs of their hosting platforms via the `stn:hostedBy` property. A platform's URI should dereference to a machine-readable description of its API such that software agents can "learn" on-the-fly how to interact with the platform. The descriptions can be hosted by platform authorities or third parties.

[4] http://w3id.org/stn/

3.2.2 Formats

RDF is the data model of the Semantic Web and a natural choice for representing the states of digital artifacts in hypermedia-driven STNs. Developers then have two options when choosing representation formats: either to reuse standard RDF serialization formats, or to define new serialization formats that fit their needs. The latter can be particularly useful, for instance, to obtain concise representations in constrained environments. In this paper, we take the former approach: we serialize digital artifacts in Turtle [20] using the STN ontology.

However, most existing Web APIs do not use RDF. They typically expose implementation-specific data models using generic media types, such as JSON. To integrate these APIs in STN-based environments, the STN ontology provides a simple mapping language for extracting RDF data from heterogeneous JSON representations.[5] For instance, Listing 2 shows a mapping that uses the STN ontology to enable software agents to extract an RDF description of a user account from a JSON representation of a Twitter account. To extract RDF data from other representation formats, developers can use more general approaches, such as SPARQL-Generate [14] or RML [8].

Listing 2: This listing shows a mapping from a JSON representation of a Twitter account to an RDF model of a user account described using the STN ontology.

```
@prefix stn: <http://w3id.org/stn/core/operations#> .
@prefix stn-http: <http://w3id.org/stn/core/http#> .

<#twitterAccountJSONMapping>
    a stn-ops:Representation ;
    stn-ops:mediaType stn-http:JSON ;
    stn-ops:entityType stn:UserAccount ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:key "screen_name" ;
            stn-http:STNTerm stn:id ;
        ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:key "name" ;
            stn-http:STNTerm stn:name ;
        ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:key "description" ;
            stn-http:STNTerm stn:description ;
        ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:key "url" ;
            stn-http:STNTerm stn:heldBy ;
        ] .
```

3.2.3 Interaction

Interaction in REST-style STNs must be driven by hypermedia. We use several elements for this purpose:

- the *social network metaphor* enforces relations between agents and artifacts, which are reified in the digital world as references across digital artifacts;
- *agent descriptions*, such as the one in Listing 1, enable the discovery of agents' user accounts and thus serve as convergence points for STNs;
- *hosting relations* encoded in digital artifact descriptions (see Listing 1) enable the discovery of machine-readable API documentation, which enables software agents to "learn" on-the-fly how to interface with the platforms they encounter.

The STN ontology provides two modules for describing Web APIs: one for describing common STN operations, and one for describing HTTP bindings of STN operations (see [6]). The latter can be replaced with modules for other protocols, or even with other approaches for describing Web APIs, such as the ones mentioned in Section 2.1.

[5]More precisely, mappings can be from any data source modeled using key/value pairs and arrays of values, but we have only used this approach for JSON representations.

4. PROOF OF CONCEPT: THE SOCIAL TV

In this section, we demonstrate the use of STNs to enable hypermedia-driven interaction across multiple siloed IoT environments. We showed how to apply STNs to constrained environments in [7]. In this paper, our focus is on the STN overlay.

4.1 Application Scenario

The deployed proof-of-concept scenario is depicted in Figure 1. David and each of his friends own *STN Boxes*, that is WoT hubs hosting their home STNs. We assume that devices register to the STN Boxes using a standard interface (e.g., the one defined in [25]) and are then instantiated as agents (see [7] for details). David has one social relation in his home STN, two Facebook friends, and follows two Twitter users (cf. Figure 1). David also owns a social TV that can autonomously discover and connect to other social TVs owned by his friends, for instance, to obtain movie ratings and recommendations. For this purpose, the TV crawls David's distributed social graph.

To implement the STN Boxes, we used multiple instances of an STN platform (see [7] for details). To deploy David's social graphs, we used Facebook's test harness and regular Twitter accounts. The social TV client is preconfigured with the authorization tokens required to access these graphs. David, his friends, their social TVs and all platforms in the deployed environments are identified through URIs and described using the STN ontology. We assume that David set up his URI on the social TV during the installation process. Lastly, to enable software agents to *navigate out of* Facebook and Twitter, we assume that David's friends advertise their URIs as their personal websites (e.g., see Listing 2), a property that is common to user accounts on most social platforms.

4.2 From Agents & Artifacts to RDF Sources

In this scenario, we model David, his friends and their social TVs as *agents*. Each of them *holds* one or more *user accounts* on an STN Box, Facebook or Twitter. An `stn:UserAccount` is an `stn:DigitalArtifact`, and it is assumed that the entity acting through a user account is acting on behalf of the `stn:Agent` that `stn:holds` the account. The platforms in the deployed environments are instances of the class `stn:Platform` (or its subclass `stn:STNPlatform`), and in our scenario implementation they are neither agents, nor artifacts.

Per our discussion in Section 3.2, agents, artifacts and any other entities in the deployed distributed STN are mapped to *resources*. All agents and platforms in our scenario are real-world entities and thus *non-information resources*. The digital artifacts used in our implementation (i.e., user accounts, platform descriptions) are *information resources* that describe real-world entities in the deployed STN. Dereferencing the URI of a non-information resource (e.g., David) returns a representation of an information resource describing it (e.g., David's user account on his STN Box).

The distributed STN in our scenario is deployed across multiple

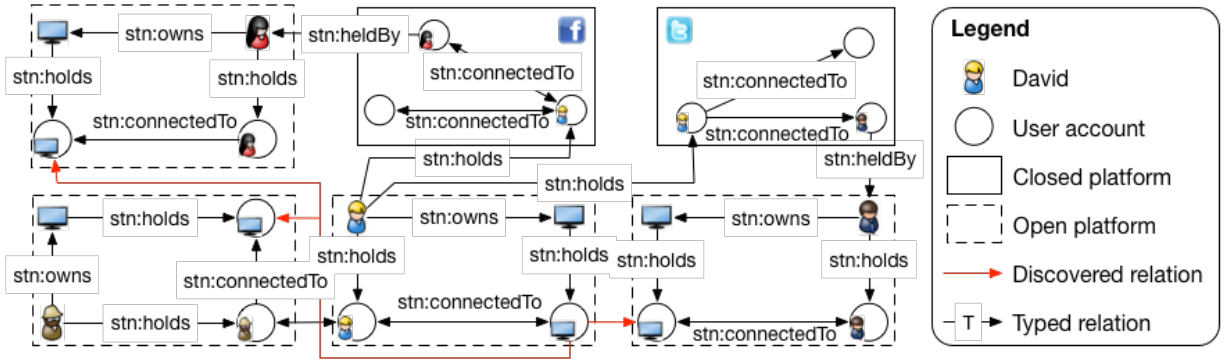

Figure 1: This image depicts a software agent's uniform RDF representation of the distributed STN deployed in our scenario, which spans across Facebook, Twitter and multiple WoT environments. David's social TV can navigate the STN in an informed manner in pursuit of its goal to discover and connect to other social TVs owned by David's friends.

heterogeneous platforms that are exposed through both hypermedia APIs (i.e., the STN platforms) and non-hypermedia APIs (i.e., Facebook and Twitter). The STN platforms use the STN ontology to produce RDF representations of `stn:UserAccount`s. To integrate Facebook and Twitter in this STN-based environment, we used the STN ontology to describe in RDF: (i) the STN operations supported by the APIs of the two platforms, such as retrieving the outgoing social relations of a given user account, and (ii) mappings for extracting RDF representations of `stn:UserAccount`s from the JSON representations produced by the two platforms (e.g., such as the mapping in Listing 2). These platform descriptions provide software clients with uniform wrappers over the heterogeneous APIs of Facebook and Twitter, in which JSON representations of user accounts are lifted to RDF. The end result, from the point of view of a software client, is a uniform RDF representation of the distributed STN, as depicted in Figure 1.

4.3 Autonomous Navigation and Action

The social TV's entry point into the environment is David's URI (see Figure 1), which dereferences to an RDF description similar to the one in Listing 1.[6] Using this description, the TV discovers David's user accounts. For user accounts that use platform-specific identifiers, the TV follows the `stn:hostedBy` relations to discover the APIs of their hosting platforms (cf. Listing 1), and then performs `stn-ops:GetOutgoingRelations` operations on each platform to discover the user accounts of David's friends. Then, for each friend's user account, the TV retrieves a representation and, if necessary, translates it to RDF using the description of the hosting platform (e.g., see Listing 2). Using these RDF representations, the TV follows the `stn:heldBy` relations to retrieve descriptions of David's friends and checks if any of them `stn:owns` an `ex:SocialTV`[7]. The TV creates new relations to all discovered social TVs for future reference (see Figure 1).

4.4 Discussion

In this proof of concept, we applied the STN model to create a hypermedia-driven environment distributed across multiple *heterogeneous* platforms, and to *bridge* otherwise *siloed* IoT environments. Given an entry point, the social TV is able to navigate the environment in an *informed manner* to discover other social TVs. The social TV is also able to act on the environment in an *autonomous and reliable manner*, for instance to create relations to newly discovered TVs. Other social TVs can then reuse these relations, for instance to speed up or expand their searches. We have thus shown that STN-based environments can address the requirements in Section 1.

A challenge for this particular scenario is posed by API rate limiting. For instance, the Twitter Public API v1.1 permits at most 15 authorized requests to most endpoints in a 15 minutes interval.[8] This limit is high enough to crawl the tiny graph in our deployment scenario, but it can hinder the crawling of larger graphs. This challenge, however, also hints to a more general problem:

Rate limiting is a common practice for Web APIs, and we expect that access to resources will become even more problematic in the WoT. To support both interoperability and politeness, it would be useful to have a uniform mechanism to retrieve applicable API rate limits at runtime. Shared caches and other mechanisms for shared access to resources could also be useful to further mitigate this problem.

5. CONCLUSIONS AND PERSPECTIVES

In this paper, we explored the use of *socio-technical networks (STNs)* as an overlay for enabling hypermedia-drive interaction across otherwise siloed IoT environments. We presented an approach that conforms to the REST architectural style and relies on linked data to apply the STN model over both hypermedia and non-hypermedia APIs, integrating them into a single hypermedia-driven environment. We have successfully integrated in this environment both Facebook and Twitter, two of the most widely used social platforms. An autonomous software agent is then able to navigate and act on the distributed STN while being agnostic to the underlying platforms.

We envision several classes of applications in which STNs could be useful. A first class of applications are those focused around people. For instance, our proof of concept leverages relations among people to enhance discoverability in the IoT. Other uses of online social networks have already been explored in the WoT community, such as social-based access control for physical devices [10, 19, 23], providing uniform interfaces for interacting with heterogeneous things [5], or socially-enhanced IoT mashup editors [19]. A second class of applications are those that could benefit from being designed as loosely coupled networks of goal-driven software agents. In home automation scenarios, for instance, STN-based applications

[6]The RDF description in Listing 1 uses example URIs for legibility.
[7]The `ex:` prefix denotes the namespace `http://example.org#`.

[8]https://dev.twitter.com/rest/public/rate-limits/, Accessed: 28.09.2016.

allow things to be deployed and to evolve independently from one another, while still being able to compose their services at runtime in pursuit of design or user-specified goals. We presented one such application in [7]. A third class of applications are those based on people and things working together in pursuit of common goals, a vision that emerges under various overlapping terms, such as *hybrid societies* [9], *social computation* [21], or *the global brain* [3].

6. REFERENCES

[1] Tim Berners-Lee. 2006. Linked Data - Design Issues. http://www.w3.org/DesignIssues/LinkedData.html. (2006). Accessed: 19.09.2016.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific American* 284, 5 (2001), 28–37.

[3] Abraham Bernstein, Mark Klein, and Thomas W Malone. 2012. Programming the global brain. *Commun. ACM* 55, 5 (2012), 41–43.

[4] Michael Blackstock and Rodger Lea. 2014. IoT interoperability: A hub-based approach. In *Proceedings of the 4th International Conference on the Internet of Things*. IEEE, 79–84.

[5] Michael Blackstock, Rodger Lea, and Adrian Friday. 2011. Uniting online social networks with places and things. In *Proceedings of the Second International Workshop on Web of Things*. ACM, 5.

[6] Andrei Ciortea, Olivier Boissier, Antoine Zimmermann, and Adina Magda Florea. 2015. Towards a Social and Ubiquitous Web: A Model for Socio-technical Networks. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Vol. 1. IEEE, 461–468.

[7] Andrei Ciortea, Olivier Boissier, Antoine Zimmermann, and Adina Magda Florea. 2016. Responsive Decentralized Composition of Service Mashups for the Internet of Things. In *the 6th International Conference on the Internet of Things*. ACM.

[8] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. 2014. RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data.. In *LDOW*.

[9] Fabien Gandon, Michel Buffa, Elena Cabrio, Olivier Corby, Catherine Faron-Zucker, Alain Giboin, Nhan Le Thanh, Isabelle Mirbel, Peter Sander, Andrea Tettamanzi, and others. 2013. Challenges in Bridging Social Semantics and Formal Semantics on the Web. In *International Conference on Enterprise Information Systems*. Springer, 3–15.

[10] Dominique Guinard, Mathias Fischer, and Vlad Trifa. 2010. Sharing using social networks in a composable Web of Things. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 702–707.

[11] Ian Jacobs and Norman Walsh. 2004. *Architecture of the World Wide Web, Volume One, W3C Recommendation 15 December 2004*. W3C Recommendation. World Wide Web Consortium (W3C). http://www.w3.org/TR/2004/REC-webarch-20041215/

[12] Matthias Kovatsch, Yassin N Hassan, and Simon Mayer. 2015. Practical semantics for the Internet of Things: Physical states, device mashups, and open questions. In *Internet of Things (IOT), 2015 5th International Conference on the*. IEEE, 54–61.

[13] Markus Lanthaler and Christian Gütl. 2013. Hydra: A Vocabulary for Hypermedia-Driven Web APIs. *Proceedings of the 6th Workshop on Linked Data on the Web* 996 (2013).

[14] Maxime Lefrançois, Antoine Zimmermann, and Noorani Bakerally. 2016. Flexible RDF generation from RDF and heterogeneous data sources with SPARQL-Generate. In *the 20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16)*.

[15] Simon Mayer. 2014. *Interacting with the Web of Things*. Ph.D. Dissertation. Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 22203.

[16] Simon Mayer, Nadine Inhelder, Ruben Verborgh, Rik Van de Walle, and Friedemann Mattern. 2014. Configuration of smart environments made simple: Combining visual modeling with semantic metadata and reasoning. In *Internet of Things (IOT), 2014 International Conference on the*. IEEE, 61–66.

[17] Simon Mayer and David S Karam. 2012. A computational space for the Web of Things. In *Proceedings of the Third International Workshop on the Web of Things*. ACM, 8.

[18] Andrea Omicini, Alessandro Ricci, and Mirko Viroli. 2008. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous agents and multi-agent systems* 17, 3 (2008), 432–456.

[19] Antonio Pintus, Davide Carboni, and Andrea Piras. 2012. Paraimpu: a platform for a Social Web of Things. In *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 401–404.

[20] Eric Prud'hommeaux and Gavin Carothers. 2014. *RDF 1.1 Turtle - Terse RDF Triple Language, W3C Recommendation 25 February 2014*. W3C Recommendation. World Wide Web Consortium (W3C). http://www.w3.org/TR/2014/REC-turtle-20140225/

[21] Michael Rovatsos. 2014. Multiagent systems for social computation. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1165–1168.

[22] Leo Sauermann and Richard Cyganiak. 2008. Cool URIs for the Semantic Web. https://www.w3.org/TR/2008/NOTE-cooluris-20081203/. (2008).

[23] Thomas Schmid and Mani B Srivastava. 2007. Exploiting social networks for sensor data sharing with SenseShare. *Center for Embedded Network Sensing* (2007).

[24] Z. Shelby. 2012. Constrained RESTful Environments (CoRE) Link Format. RFC 6690 (Proposed Standard). (Aug. 2012). http://www.ietf.org/rfc/rfc6690.txt

[25] Z. Shelby, M. Koster, and P. van der Stok. 2016. CoRE Resource Directory draft-ietf-core-resource-directory-07. (2016).

[26] Steve Speicher, John Arwe, and Ashok Malhotra. 2015. *Linked Data Platform 1.0, W3C Recommendation 26 February 2015*. W3C Recommendation. World Wide Web Consortium (W3C). http://www.w3.org/TR/2015/REC-ldp-20150226/

[27] Ruben Verborgh, Thomas Steiner, Davy Van Deursen, Sam Coppens, Joaquim Gabarró Vallés, and Rik Van de Walle. 2012. Functional Descriptions as the Bridge between Hypermedia APIs and the Semantic Web. In *Proceedings of the Third International Workshop on RESTful Design*. ACM, 33–40. DOI:http://dx.doi.org/10.1145/2307819.2307828