Towards a Social and Ubiquitous Web: A Model for Socio-technical Networks

Andrei Ciortea
Laboratoire Hubert Curien
UMR CNRS 5516, Institut Henri Fayol,
Mines Saint-Etienne, Saint-Étienne, France
University "Politehnica" of Bucharest,
Romania
Email: andrei.ciortea@emse.fr

Antoine Zimmermann
and Olivier Boissier
Laboratoire Hubert Curien
UMR CNRS 5516, Institut Henri Fayol,
Mines Saint-Etienne, Saint-Étienne, France
Email: antoine.zimmermann@emse.fr,
olivier.boissier@emse.fr

Adina Magda Florea
University "Politehnica" of Bucharest,
Romania
Email: adina.florea@cs.pub.ro

***Abstract*—The Web is to experience considerable growth by extending to physical devices, i.e. the so-called *Web of Things (WoT)*. To make this growth sustainable, users need mechanisms for managing and interacting with large networks of devices and services. Existing search engines and mashup editors for the WoT partially address this problem by enabling developers and tech savvy users to search for "things" and connect them to one another. However, manually "wiring" large networks of things does not scale. Furthermore, static mashups cannot adapt to dynamic environments. Our approach is to apply the social network metaphor to the WoT to create a semantic socio-technical overlay that may be reliably processed and used by both people and things. Things can crawl this overlay to discover one another, but also manipulate it such that they may wire themselves. In this paper, we introduce a model for *socio-technical networks (STNs)*. We apply this model to wrap up Twitter and Facebook as STNs, and to develop ThingsNet, our own prototype STN platform. Finally, we present a software client that uses the STN model to function across all three platforms, as well as any other platforms that may be translated to this model.**

I. Introduction

Metcalfe's law suggests that the value of a network-like system is proportional to the square of the number of nodes in the system. The World Wide Web seems to be no exception. Bob Metcalfe also suggests that his law is mostly applicable to small systems, and once a network reaches a certain size, the affinity (i.e., value per connection) might suffer to the point where it overwhelms the quadratic growth of its value.[1] In other words, growth must be sustainable.

The Web is taking a new leap: it extends to the physical world by integrating places, devices and physical things as first-class entities, i.e. the so-called *Web of Things (WoT)* [1]. It seems reasonable to predict a significant growth – not only in size, but also in functionality: writing the state of a Web resource will not only result in updating a database, it may literally open a door in the physical world. A leap that brings huge value if the Web manages to balance its affinity by making this growth sustainable. In doing so, it is necessary to minimize all implied costs: the costs of integrating things into the Web, the costs of developing WoT applications, the costs of accessing and using the WoT.

Developing Web-enabled things has already been proven to be cost-efficient [2]. Search engines for the WoT [3], [4], an emerging application domain, aim at enabling human users and software clients to look-up services on the WoT, thus reducing the costs of usage while supporting automatic service integration in composite applications. Reducing development and usage costs is also addressed by platforms that allow developers and tech savvy users to "wire" the Internet of Things (IoT) / WoT by creating mashups of devices and services[2] [5], [6]. Manually wiring large networks of things, however, does not scale. Furthermore, once created, these mashups are static and cannot adapt to dynamic environments.

Therefore, we need a uniform mechanism that goes a step beyond service look-up and static mashups, one that enables users to manage and interact with large networks of heterogeneous things and the functionality they provide. Our proposal is to apply the social network metaphor to the WoT to create a *semantic* socio-technical overlay that may be reliably processed and used by both people and things. Things can crawl this overlay to discover one another, but also manipulate it such that they may handle their own connections to people or other things, for instance according to rules embedded in their application logic or defined by an external authority. This approach falls within an emerging application domain, also referred to as the *Social Web of Things (SWoT)* [7], [8], [9] or the *Social Internet of Things* [10]. The key contribution that we introduce in this paper is a model for *socio-technical networks (STNs)*, which are the building blocks of a SWoT.

To motivate our vision for the SWoT, we introduce the principles on which we build our approach in Sec. II. Sec. III presents the STN formal model and how it can be deployed on the Web. Sec. IV applies this model to create STN wrappers for existing social platforms, such as Twitter and Facebook, but also in the development of a new generation of thing-friendly social platforms. Related work is discussed in Sec. V.

II. Towards a Web of People and Social Things

Applying the social network metaphor to the WoT leads to the following sub-objectives:

[1] Guest Blogger Bob Metcalfe: Metcalfe's Law Recurses Down the Long Tail of Social Networks. Nov 2006. http://bit.ly/1AZCsQJ

[2] Node-RED: http://www.nodered.org, Accessed: 15.05.2015.

1) to develop *social things* that may autonomously participate in online interactions with people and other heterogeneous things on the Web;
2) to facilitate the development of software clients that may dynamically access and use heterogeneous *social platforms*[3];
3) to enable people and social things to access and manipulate Web resources in a secure fashion, without violating the privacy of other users.

In order to achieve these sub-objectives, we build our proposal on the following principles:

1) **Things as first-class entities** of a socio-technical overlay: The API of an STN platform should generally support all operations required to participate in the STN. In other words, STNs should generally accommodate things as full-fledged citizens. For instance, on Twitter users may be anyone or anything[4], and most actions are available via its API. This principle supports sub-objective 1.
2) **Interoperability**: STN platforms should provide machine-readable interfaces such that social things are able to access and participate in the socio-technical overlay without being manually configured against each of its underlying platforms. In addition, the socio-technical overlay should integrate existing social platforms. This principle supports sub-objectives 1 & 2.
3) **Autonomy & Regulation**. Social things should be able to (semi-)autonomously manage their relationships with other entities. In addition, proactive sharing of information should be encouraged for exploiting the full potential of the SWoT. Regulation implies that social things should be able to interpret and reason about externally defined policies and norms. In an STN platform, all regulations should be explicit and available in machine-readable format. Norm enforcing, smart disclosure and social control mechanisms are also necessary. This principle supports sub-objectives 1 & 3.

We consider autonomy and regulation to be two facets of the same coin. Without regulation, autonomy is potentially dangerous. Without autonomy, regulation is not needed. It is worth to note that most social networks regulate the actions of users and developers both explicitly (e.g., terms of services and rate limiting) and implicitly (e.g., social norms of usage).

Following these principles, we propose a layered architecture for the SWoT, which is built on top of the WoT and structured along four layers (from bottom to up): *agency*, *social*, *normative* and *application* [9].

Agency Layer. In our proposal, a thing may be any kind of entity: physical (e.g., a device or a book), digital (e.g., an e-mail) or abstract (e.g., a research group). Obviously, not all things in the universe of discourse need to be social. People and social things are both abstracted by *agents* and are first-class entities in the SWoT. Agents exhibit characteristics such as autonomy, goal-driven behavior and social ability. Our aim is to profit from the vast amount of models and technologies available for programming software agents and systems of autonomous agents [11], [12]. Non-social things are modeled as *artifacts*, that is to say, as resources usable by agents to act on or perceive the physical or digital world.[5] For instance, a door whose state (e.g., locked or unlocked) is read or written via a Web API may be modeled as an artifact. If the door may also interact with people or vacuum cleaning robots[6] to unlock itself for authorized agents, then it may be modeled as an agent. Securely and uniquely identifying agents is also addressed at this layer, for instance by means of single sign-on systems, such as OpenID Connect [14] or WebID [15]. This layer is motivated by principles 1 & 3.

Social Layer. At the social layer, agents become connected in STNs. The different relations (e.g., friendship, ownership, provenance, colocation) among agents, and between agents and artifacts, are represented in social graphs as structured data, such that both people and social things may reliably process and reason upon them. For instance, all appliances in a house, together with its inhabitants, may form an STN. Similarly, a world-wide network of environmental sensors may form a widely distributed STN. People and social things should be able to access and participate in linked, open and interoperable STNs, which may be distributed across multiple platforms. This layer is motivated by principles 1 & 2.

Normative Layer. This layer is concerned with models and mechanisms for expressing, monitoring and regulating policies or norms. Any applicable policies and norms should be made available to social things in a machine-readable format. While regulation may be achieved at the agency layer by hardcoding rules into agents, being able to interpret externally-defined norms is desirable, if not necessary. For instance, software agents should be able to retrieve a platform's rate limiting policies and terms of services in a machine readable format. Similarly, an intelligent vehicle should be able to dynamically obtain and interpret the driving regulations that apply to its current country or region. User-defined privacy policies related to the exchange and sharing of information should also be available to social things. Any mechanisms for social control (e.g., trust and reputation) are also located at this layer. This layer is motivated by principle 3.

In this paper, due to space considerations, we focus our discussion on the social layer. The key concern that we address is to define, model and develop open and interoperable STNs as building blocks of a socio-technical overlay for the WoT.

III. Socio-technical Networks

An STN has to represent networks of people and things, taking into account their position in the physical world, the data they produce, norms that may affect their evolution, and operations that enable them to participate in the STN. In this section, we first introduce a general model for networks of agents (see Sec. III-A). This general model is then made more specific to describe *digital STNs* (see Sec. III-B). Then, having introduced the mathematical structures underlying our envisioned socio-technical overlay, Sec. III-C discusses how digital STNs may be deployed on the Web.

[3] A *social platform* is any platform that enables communication and interaction with other autonomous entities.

[4] https://dev.twitter.com/overview/api/users/, Accessed: 15.05.2015.

[5] Approach inspired by the *Agents & Artifacts* meta-model [13] from the field of multi-agent systems.

[6] Such as the iRobot Roomba: http://www.irobot.com/for-the-home/vacuum-cleaning/roomba.aspx, Accessed: 15.05.2015.

A. A general model for STNs

We use $\mathcal{E}$ to denote the infinite set of entities that may be part of an STN, and $\mathcal{AG} \subset \mathcal{E}$ to denote the infinite set of agents. We denote the infinite set of STNs by $\mathcal{S}$ and formally define an STN $s \in \mathcal{S}$ as the following construct:

$$s = (G_t, Ops, Norms, O), \tag{1}$$

where:

- G_t is a graph that represents the state of s at any given point in time t, and we call it the *socio-technical graph (STG)*;
- *Ops* is a set of operations through which agents in $\mathcal{AG}$ may retrieve information from or modify the state of G_t;
- *Norms* is the set of norms that regulate the use of operations in *Ops*;
- O is an ontology in a given knowledge representation language L (e.g., an OWL ontology) that encapsulates domain-specific knowledge for s.

The signature of O defines two sets of non-logical symbols: *ETypes*, which are the types of entities in s, and *RTypes*, which are the types of relations among entities in s. For all $s \in \mathcal{S}$, we define symbols $Agent \in ETypes$ and $\{type, connectedTo\} \subset RTypes$, where *type* associates types from *ETypes* to entities in s and *connectedTo* represents a directed social relation between two agents.[7]

G_t is then a directed edge-labeled multigraph given by:

$$G_t = (N, E), \tag{2}$$

where: $N \subset \mathcal{E}$ is the finite set of nodes in G_t, that is to say the entities in s at moment t, and $E \subseteq N \times RTypes \times (N \cup ETypes)$ is the finite set of directed labeled edges in G_t.

For illustration, consider a social network formed by two friends, *John* and *Jane*. We choose to define a DL ontology O that describes two concepts (i.e., entity types), *Agent* and *Person*, and three roles (i.e., relation types), *type*, *connectedTo* and *friendOf*, where: $Person \sqsubseteq Agent$, *friendOf* is symmetric and $friendOf \sqsubseteq connectedTo$. We define two individuals, *John* and *Jane*, where *John* and *Jane* are persons, and *John* is a friend of *Jane*. Following our definitions, we have:

$$\begin{aligned}
&ETypes = \{Agent, Person\}\\
&RTypes = \{type, connectedTo, friendOf\}\\
&G_{t,il} = (\{John, Jane\},\\
&\qquad \{(John, type, Person), (Jane, type, Person), (John, friendOf, Jane)\})
\end{aligned}$$

We write $O \cup G_t \vDash_L (a, rel, b)$, where $a, b \in N$ and $rel \in RTypes$, to say that we use knowledge encapsulated in O and G_t to infer that (a, rel, b) holds in s.[8] For instance, in the above illustration, it may be inferred that:

$$\begin{aligned}
&O \cup G_t \vDash_L (John, type, Agent)\\
&O \cup G_t \vDash_L (Jane, type, Agent)\\
&O \cup G_t \vDash_L (Jane, friendOf, John)\\
&O \cup G_t \vDash_L (John, connectedTo, Jane)\\
&O \cup G_t \vDash_L (Jane, connectedTo, John)
\end{aligned}$$

We denote the restriction of N to all nodes of a given type $T \in ETypes$ by $N[T] = \{n \in N \mid O \cup G_t \vDash_L (n, type, T)\}$. Similarly, we denote the restriction of all edges of a given type $rel \in RTypes$ by $R_{rel} = \{(a, b) \in N \times (N \cup ETypes) \mid O \cup G_t \vDash_L (a, rel, b)\}$. In our previous example, $N[Agent] = \{John, Jane\}$ and $R_{connectedTo} = \{(John, Jane), (Jane, John)\}$.

We use $\mathcal{G}$ to denote the infinite set of STGs. Given $s \in \mathcal{S}$, an agent operation $op \in Ops$ is formally defined as a function of the form:

$$op : \mathcal{G} \times \mathcal{AG} \times 2^{Input} \to \mathcal{G} \times 2^{Output}, \tag{3}$$

where we use *Input* and *Output* as technical notations for the sets of all possible input and output parameters of an agent operation. An operation is always performed by an agent and, in most cases, it defines a transition from one state of the STN to another.

STGs and agent operations provide the mathematical foundation for modeling the social layer (cf. Sec. II). The social layer is governed by the normative layer via norms. Norms may encourage operations (e.g., all social things owned by the same user should connect to one another), regiment operations (e.g., a social thing should not register to new STN platforms without explicit consent from its owner), or they may filter the output of operations (e.g., things owned by a user should be advertised only to close friends). Research in multi-agent systems proposes several models that may be used for precisely defining, formalizing and regulating norms [18], [19]. We do not address the normative layer in this paper, however this does not influence our contribution or the clarity of the rest of the paper.

B. Digital STNs

We further specialize our model to define the class of *digital STNs*, that is STNs reified as explicit digital representations that agents may use and manipulate. In order to support compatibility with existing platforms (see Sec. II), and to preserve the design and implementation autonomy of new platforms, our model for digital STNs extracts the commonality from existing social platforms with high user adoption[9], widely accepted standards [20], [21] and ontologies [16], [22], [23] in order to define a core set of types, agent operations, and digital artifacts[10].

1) Types: In addition to the types defined previously, for all digital STNs, $\{Entity, Person, SocialThing, DigitalArtifact, Platform\} \subset ETypes$ and $\{represents, hostedBy, owns\} \subset RTypes$, where:

- $N[Person] \cup N[SocialThing] \subset N[Agent]$, and $N[Person] \cap N[SocialThing] = \varnothing$;[11]
- $N[Agent], N[DigitalArtifact], N[Platform]$ are mutually disjoint subsets of $N[Entity]$.

A digital artifact *represents* an entity and it is always *hostedBy* a platform. We express this formally by defining two

[7] As a convention, symbols for entity types start with a capital letter. Therefore, *Agent* is an entity type, while *type* and *connectedTo* are relation types.

[8] In category theoretic terms, this notation is formally correct if there is a span in the category of institutions [17] between the institution for L and the one for G_t such that the pushout exists.

[9] Such as Facebook, Twitter, Google Plus, and Foursquare.

[10] *Digital artifacts* are artifacts that exist only in the digital world, such as user accounts or digital messages

[11] That is to say, all persons and social things are agents, and a node cannot be typed as both a person and a social thing.

relations: $R_{represents} : N[DigitalArtifact] \nrightarrow N[Entity]$,[12] and $R_{hostedBy} : N[DigitalArtifact] \rightarrow N[Platform]$. The third relation type, *owns*, is used to represent that a social thing may be owned by another entity, in most cases an agent or a group of agents. We define $R_{owns} \subseteq N \times N[SocialThing]$.

We introduce a few notations that we use throughout the rest of this section. We use $\mathcal{DA} \subset \mathcal{E}$ to denote the infinite set of digital artifacts. The digital image of an entity $e \in \mathcal{E}$ within an STN $s \in \mathcal{S}$ is defined as $dgIm_s(e) = \{d \in \mathcal{DA} \mid (d, e) \in R_{represents}\}$. We define the digital image closure of a set of entities $A \subseteq \mathcal{E}$ within $s \in \mathcal{S}$ as $A^*_s = \bigcup_{a \in A} dgIm_s(a) \cup A$. For instance, the digital image closure of a set of agents in a given STN is the set of agents and the digital artifacts that represent them within the STN, such as user accounts.

2) Agent operations: The general form of an operation was given by Equation 3 in Sec. III-A. Operations are described further by specifying the context in which they may be applied, their parameters, and any side-effects they might have on an STG.

Social relations or *connections* in digital STNs may be created between agents or digital artifacts that represent them. For instance, on Twitter connections are created between *user accounts*, while in FOAF [16] they are created between *agents*, and not the actual FOAF profile documents that describe them. We define an operation for creating an outgoing connection from a performing agent a_p as follows:

createConnectionTo:
1. Desc: Agent a_p performs this operation to create a connection from itself (or a digital artifact that represents it) to a targeted entity u, where u is an agent or a digital artifact that represents an agent.
2. Input: $u \in \mathcal{AG}^*_{\mathcal{S}}$.
3. Preconds: $u \in N$ or $\exists u' \in N$ s.t. $u' \in \{a_p\}^*_s$.
4. Postconds: $u \in N$ and $\exists u' \in \{a_p\}^*_s$ s.t. $u' \in N$ and $(u', u) \in R_{connectedTo}$.
5. Output: $desc_s(u)$.

Agent a_p is the source of the intended connection, while the target of the connection, which is a required input parameter, is either an agent or a digital artifact that represents it. The precondition requires that either the source or the target must already be in the STN. The postcondition specifies that successfully completing this type of operation implies there exists a *connectedTo* edge between the source and the target, and both have been added to the set of nodes. The output of a *createConnectionTo* operation returns a description of the target. Given an STN $s \in \mathcal{S}$, function $desc_s : N \rightarrow 2^E$ returns the description of a node $n \in N$ as a set of edges. The returned description is specific to each STN and may be affected by norms, such as privacy policies (see the previous section).

Implementers may choose to further restrict this operation within their STNs, for instance by requesting that both the source and the target of the connection should already exist within the STN, thus making it a closed STN. Operations for creating incoming connections or deleting existing connections are defined similarly. Most other operations in a digital STN, however, would depend on the digital artifacts within the STN.

3) Common digital artifacts: Digital artifacts that may be commonly used in STNs include user accounts, groups, messages or places. For each artifact, the model defines artifact-specific relations and operations. Due to space considerations, we only discuss user accounts, however other artifacts are defined in a similar fashion.

A *user account* is typically held by an agent, meaning that the entity acting via the user account is assumed to have been delegated by and acting for the agent.[13] The user account may have an associated name, description or local identifier within the STN. In addition, the user account may also hold information about the represented agent, such as the owner of a social thing. Formally, we define $UserAccount \in ETypes$, $\{id, name, description, heldBy\} \in RTypes$, and say that $\forall n \in N, (n, UserAccount) \in R_{type} \Rightarrow (n, DigitalArtifact) \in R_{type}$. Furthermore, we define $R_{heldBy} \subseteq N[UserAccount] \times N[Agent]$ such that $\forall s \in \mathcal{S}, O \vDash_L (u, a) \in R_{heldBy} \Rightarrow (u, a) \in R_{represents}$, that is to say an account *heldBy* an agent *represents* the agent.

Creating a user account is defined as follows:

createUserAccount:
1. Desc: Agent a_p creates a user account on platform p.
2. Input: $p \in \mathcal{P}$.
3. Preconds: $\exists d \in \mathcal{DA}$ s.t. $d \notin N[DigitalArtifact]$.
4. Postconds: $d \in N[UserAccount], (d, a_p) \in R_{heldBy}, (d, p) \in R_{hostedBy}$.
5. Output: $desc_s(d)$.

A *createUserAccount* operation requires as input a platform $p \in \mathcal{P}$, where $\mathcal{P} \subset \mathcal{E}$ is the set of all platforms. In most cases, platforms would require additional parameters to create a user account (e.g., a name to be displayed). As a precondition, the platform must be able to provide a digital artifact that is not already in use. If the operation is completed successfully, a new *UserAccount* artifact is created, which is held by a_p and hosted by platform p, and its description is returned. Operations for updating or deleting user accounts are defined similarly.

Our mathematical model for digital STNs provides an unambiguous foundation for the envisioned socio-technical overlay. Going a step further, we next discuss how digital STNs may be deployed on the Web.

C. Web-compliant STN platforms

Similar to existing social platforms, a digital STN is centered around an STN platform, i.e. a system that provides a collection of features for participating in the STN. In a Web-compliant STN platform, digital artifacts are mapped to Web resources and operations are implemented as HTTP requests. Furthermore, in accordance with the interoperability principle (see Sec. II), platforms should produce semantic representations of digital artifacts and provide machine-readable interfaces. For this purpose, we have created an OWL ontology, which we call the STN ontology, that implements our digital STN model. The STN ontology is available online[14] and has been described in detail in [25].

The STN ontology defines three modules: *STN-Core*, which provides the core concepts and properties required to describe STGs, *STN-Operations*, which may be used to describe agent operations, and *STN-Operations-HTTP*, which may be used to describe implementations of agent operations as HTTP requests. The ontology has a modular design such that it may

[12] $R_{represents}$ is a partial function, artifacts may also be purely digital entities.

[13] An agent may hold multiple user accounts, usually on different platforms.

[14] http://purl.org/stn/core/spec

be easily extended. For instance, the *STN-Operations-HTTP* module may be substituted with any other similar vocabularies for RESTful APIs, such as Hydra [24], or even with modules for different protocols, such as CoAP[15], a Web protocol for resource-constrained devices. In the following section, we apply the STN ontology to create semantic descriptions of existing platforms.

IV. Validation and Discussion

This section presents the current validations of our STN model. We first introduce a prototype STN platform (see Sec. IV-A). We then report on the use of the STN ontology to create semantic descriptions for Twitter and Facebook (see Sec. IV-B). Sec. IV-C presents a software client that may uniformly access all three platforms based on their semantic descriptions.

A. ThingsNet: an STN platform

ThingsNet[16] is an STN platform that follows our digital STN model (see Sec. III-B). The platform uses two types of digital artifacts, user accounts and messages. Artifacts are identified through HTTP URIs, which makes them globally accessible, and their states are represented in RDF. ThingsNet uses the STN ontology and supports Turtle representations. Its API supports agent operations for creating, retrieving and deleting user accounts, connections and messages. All operations are described in ThingsNet's semantic description:[17]

```
@base <http://www.thingsnet.com> .
@prefix stn: <http://purl.org/stn/core#> .
@prefix stn-ops: <http://purl.org/stn/operations#> .
@prefix stn-http: <http://purl.org/stn/http#> .
@prefix http: <http://www.w3.org/2011/http#> .

<#platform>
    a stn:STNPlatform ;
    stn:name "ThingsNet" ;
    stn-http:baseURL <http://localhost:9000> ;
    stn-http:supportsAuth stn-http:WebID ;
    stn-ops:consumes stn-http:Turtle ;
    stn-ops:produces stn-http:Turtle ;
    stn-ops:supports <#createAccount> ,
        <#getAccount> ,
        (...)
        <#deleteMessage> .
```

From this description, a software client may identify ThingsNet as an `stn:Platform`, it may extract the supported STN operations and other general information about the platform (e.g., the base URL of its API). A social thing should validate that it supports the required authentication standards (i.e., WebID) and representation format (i.e., Turtle). Each STN operation is described further, such as the *createUserAccount* operation (cf. Sec. III-B3):

```
<#createAccount>
    a stn-ops:CreateUserAccount ;
    stn-ops:implementedAs [
            a stn-http:AuthSTNRequest ;
            http:methodName "POST" ;
            http:requestURI "/users" ;
        ] ;
    stn-ops:hasRequiredInput [ a stn-ops:SocialThingClass ] ;
    stn-ops:hasRequiredInput [ a stn-ops:Owner ] ;
    stn-ops:hasRequiredInput [ a stn-ops:DisplayedName ] ;
    stn-ops:hasInput         [ a stn-ops:Description ] ;
    stn-ops:hasOutput [
            a stn-http:TurtleRepresentation ;
            stn-ops:representationOf [ a stn:UserAccount ] ;
        ] .
```

ThingsNet relies on WebID to uniquely identify the agents.[18] The platform parameter is implicit (cf. Sec. III-B3). In addition, ThingsNet requires a number of other parameters for this operation: the social thing's class, which may be `stn:SocialThing` or any of its subclasses, an URI identifying the thing's owner, and a name to be displayed within the STN. Optionally, a description may also be given. The operation returns a Turtle representation of a *UserAccount* (cf. Sec. III-B3). This operation is implemented as an authenticated `HTTP POST` request to the `/users` endpoint. The body of the request must contain a Turtle representation of the user account to be created with all required input parameters. For instance, a social TV may register to ThingsNet using a WebID provided by its manufacturer, the WebID of its owner, a preconfigured class and a user-configured name. ThingsNet responds with a Turtle representation of the created user account:

```
HTTP/1.1 201 Created
Content-Type: text/turtle
Content-Length: 784

@prefix stn: <http://purl.org/stn/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix ex: <http://www.example.com#> .

<http://localhost:9000/users/f8d...a16>
    a stn:UserAccount ;
    stn:description
        ( "A TV with a twist!"^^xsd:string ) ;
    stn:heldBy
        <http://api.mymanufacturer.com/tvs/874...260#thing> ;
    stn:hostedBy
        <http://www.thingsnet.com#platform> ;
    stn:name
        "John's Social TV"^^xsd:string .

<http://api.mymanufacturer.com/tvs/874...260#thing>
    a ex:SocialTV ;
    stn:ownedBy ex:John .
```

It is worth to note that ThingsNet is an open platform: in addition to dynamic registration, its registered users may add connections to agents on other platforms. Other operations, such as retrieving the profile of a registered user, are made available to agents outside the network, while they may still be subject to sharing policies.

B. Compatibility with existing social platforms

We have chosen Twitter and Facebook to highlight the generality of our approach with respect to the integration of existing social platforms (see Sec. II): while Twitter users may be anyone or anything[19], Facebook restricts its users to people. The complete semantic descriptions for the two platforms are available online[20,21].

[15] https://tools.ietf.org/html/rfc7252/, Accessed: 15.05.2015.

[16] http://github.com/andreiciortea/thingsnet

[17] Throughout the rest of this paper, we write RDF snippets in Turtle representation and use the prefix bindings presented here.

[18] While the prototype does not in fact implement the WebID authentication protocol, it simulates it by passing the WebID as an HTTP header in an "authenticated" request. The prototype returns a `401 Unauthorized` response whenever authentication data is missing.

[19] https://dev.twitter.com/overview/api/users/, Accessed: 15.05.2015.

[20] http://purl.org/stn/twitter

[21] http://purl.org/stn/facebook

1) An STN description for Twitter: The STN description we have created for Twitter begins as follows:

```
@base <http://www.twitter.com/> .

<#platform>
    a stn:Platform;
    stn:name "Twitter" ;
    stn-http:baseURL <https://api.twitter.com/1.1/> ;
    stn-http:supportsAuth stn-http:OAuth ;
    stn-ops:consumes stn-http:JSON ;
    stn-ops:produces stn-http:JSON ;
    stn-ops:supports <#getAccount> ,
        <#follow> ,
        (...)
        <#getDirectMessages> .
```

Social things identify the Twitter platform by using the declared URI `http://www.twitter.com/#platform`. Twitter uses OAuth for authentication and supports only JSON representations. The Twitter API supports most of the operations available to human users, such as: following and unfollowing users; sending, retrieving and deleting direct messages; posting, retrieving and deleting tweets. All these operations are included in its STN description. The *createConnectionTo* operation (cf. Sec. III-B2) is applied to Twitter as follows:

```
<#follow>
    a stn-ops:CreateConnectionTo;
    stn-ops:implementedAs [
            a stn-http:AuthSTNRequest ;
            http:methodName "POST" ;
            http:requestURI "/friendships/create.json" ;
        ] ;
    stn-ops:hasRequiredInput [
            a stn-ops:UserAccountID ;
            stn-ops:paramName "screen_name" ;
            stn-http:paramIn stn-http:Body;
        ] ;
    stn-ops:hasOutput <#twitterAccountJSONMapping> .
```

A social thing may follow a Twitter account by composing and sending an authenticated `HTTP POST` request to `/friendships/create.json`. The request must include a JSON payload with the identifier of the targeted user account. Acquiring the Twitter screen name of the targeted account is part of the application logic (e.g., via crawling or it may be provided by a human user). If the operation is completed successfully, Twitter responds with a JSON representation of the targeted account. The STN description includes a mapping between a *UserAccount* digital artifact (cf. Sec. III-B3) and Twitter's user account data model:

```
<#twitterAccountJSONMapping>
    a stn-http:JSONRepresentation ;
    stn-ops:representationOf [ a stn:UserAccount ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:thirdPartyKey "screen_name" ;
            stn-http:STNterm stn:id ;
        ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:thirdPartyKey "name" ;
            stn-http:STNterm stn:name ;
        ] ;
    stn-ops:contains [
            a stn-http:Mapping ;
            stn-http:thirdPartyKey "description" ;
            stn-http:STNterm stn:description ;
        ] .
```

This mapping enables social things to extract from Twitter's JSON response the semantic information necessary for building an RDF representation of an `stn:UserAccount`. Other Twitter operations and data model mappings are described in a similar fashion.

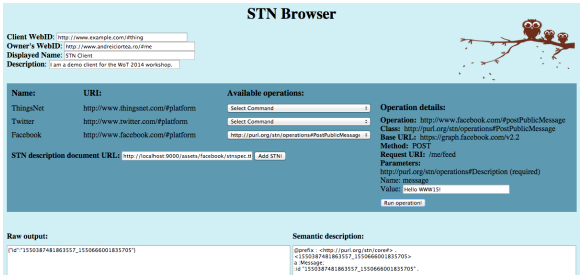

Fig. 1. The STN Browser: posting a status update on Facebook.

2) An STN description for Facebook: Facebook's STN description is similar to the one we have created for Twitter and is available online[22]. The Facebook API supports a rich variety of endpoints for reading data from its social graph. In particular, several endpoints explore the different facets of a user's profile, such as TV shows or music bands the user *liked*. On the other hand, support for writing data is somewhat more limited. For instance, a software client may post status updates or retrieve private messages on behalf of a human user, however it may not send private message or create/accept connection requests. Operations covered in the STN description include retrieving user profiles, connections, messages, feeds (i.e., the news feed, user feeds, group feeds), and group operations (e.g., groups of a given user, members in a given group, posting messages in groups). The limited expressivity of the STN ontology is apparent in the case of Facebook's rich user profiles. However, our objective was to create a generic model that would provide the backbone of a semantic socio-technical overlay. The STN ontology may be extended further with domain-specific vocabularies.

C. A Browser for STNs

We have developed a Web application for browsing the socio-technical overlay. The *STN browser* may retrieve, parse and use STN descriptions to translate agent operations to HTTP requests and extract semantic information from the bodies of HTTP responses. It supports Turtle and JSON, self-descriptive parameters are preconfigured (e.g., its WebID, owner's WebID), while other parameters may be obtained from a human user (e.g., a Twitter account to follow). Results are displayed in two areas, as depicted in Fig. 1: the area on the left shows the raw body of the received response, and the one on the right shows the extracted semantic information. We structure our discussion on accessing and using the three platforms based on the principles introduced in Sec. II.

Things as first-class entities: On ThingsNet, social things are first-class entities. The APIs of Twitter and Facebook reflect the important conceptual difference in how they define their users. On Twitter, social things may use most of the operations available to people and are thus first-class entities. Still, social things require manually created Twitter accounts. On Facebook, social things may perform limited actions and only on behalf of a human user. Thus, they remain outside

[22] http://purl.org/stn/facebook

of the network itself, however they may use Facebook as an artifact (e.g., as a source of information, for user input[23]).

Interoperability: By using the STN descriptions we have created, the STN browser is decoupled from the APIs of the three platforms. Its application logic may use the higher level concepts and operations provided by the STN model. However, both Twitter and Facebook use OAuth for authentication and authorization, and provide proprietary single sign-on systems, which implies that the browser has to be configured to access them. ThingsNet relies on WebID (see Sec. IV-A for details), and thus any STN client supporting the WebID authentication protocol may access it. ThingsNet is also an open platform, and thus it may be linked to other STNs.

Regulation: Twitter's documentation for developers describes the rate limiting policy in detail and its API provides an endpoint for retrieving the rate limit status of an application as structured data. Endpoints for retrieving the platform's privacy policy and terms of services as unstructured text are also available. Therefore, on Twitter regulations are explicit, yet not all of them in a machine-readable format. Facebook does not publish official limits for API requests and applications cannot dynamically obtain such information. ThingsNet does not currently address this issue.

The STN browser validates that our STN model and approach enable software clients to access and participate in heterogeneous social platforms in a uniform fashion. This interoperability gain provides the foundation for creating the envisioned socio-technical overlay. What is missing is explicitly linking closed networks, such as Twitter and Facebook, into the overlay. This may be achieved by means of WebID profiles [15], which may use the STN ontology to create bridges by specifying user accounts held by a thing or human user on different social platforms, thus making them discoverable. Software agents may then crawl the overlay to discover people, things and services, extend it with domain-specific knowledge, or perform more complex tasks according to their application logic.

V. Related Work

We position our proposal with respect to related work in the social Web (see Sec. V-A) and in the emerging landscape of social systems for the IoT/WoT (see Sec. V-B).

A. An Open and Distributed Social Web

The final report of the W3C Social Web Incubator Group offers a thorough analysis of the state of the Social Web in 2010 and investigates the requirements and enabling technologies for a "truly universal, open, and distributed Social Web architecture" [21]. We build upon this vision. The main addition in our proposal is that we focus on integrating things as first-class entities of an open and distributed social overlay. This brings a series of new challenges to be addressed. An important step that we take in this paper is modeling socio-technical networks.

OpenSocial [20] provides a collection of APIs for building social Web applications, allowing developers standard access to social information. The initiative was launched by Google, MySpace and other social platforms, and has strong industry support. Community support is critical in order to define a standard API for social platforms. We take a different approach. First, we create a core model for STNs, and not only social networks. We then translate our STN model to heterogeneous platforms by means of semantic descriptions. The advantage of our approach is that it has no impact on an existing API and it does not necessarily require buy-in from social platforms. The obvious drawback, however, is that the expressivity of the STN ontology, as it is, might be limited for some use cases (see Sec. IV-B2). However, the goal of the STN model and ontology is to create the backbone of a semantic socio-technical overlay that may be used for autonomous reasoning, and they may be easily extended further with domain-specific knowledge.

The STN ontology is aligned with several well-know vocabularies, such as FOAF (see [25] for details). FOAF is a widely accepted vocabulary for describing characteristics of people, social groups and relations among them. There is a close matching between many concepts in FOAF and the STN ontology, and thus social networks described in FOAF can be easily translated to STNs described using the STN ontology. FOAF, however, was not designed to accommodate things as first-class entities in social networks. For instance, while `foaf:knows` is symmetric and "relates a `foaf:Person` to another `foaf:Person` that he or she knows" [16], the `stn:connectedTo` property describes a more generic type of unidirectional connection between persons, smart things and user accounts. The two properties may be used in conjunction, and a `foaf:knows` relation may be represented as two reciprocal `stn:connectedTo` properties.

B. Social Systems in the IoT / WoT

In recent years, the idea that pervasive computing could benefit from social networks is gaining momentum. An extensive discussion of this subject is provided in [26].

A number of research projects have used social networks for sharing Web-enabled things. For instance, SenseShare [27] proposed to use Facebook as front end for their system, relying on the platform's social graph to provide a set of privacy policies for sharing. The Social Access Controller [28] went a step further by providing support for several social platforms and a central point of access control. Paraimpu [7] is another social tool that allows users to share their things, discover and bookmark things shared by other users and compose them for personalized applications. In Paraimpu, social networks may also be used as sensors/actuators. Paraimpu is thus similar to a social mashup editor for the IoT/WoT. In these approaches, things are digital artifacts controlled and composed by people.

In other work, social networks have been proposed as user interface containers for WoT applications [8]. The user interface of platforms such as Facebook or the ones implementing OpenSocial may be extended with plug in applications, thus providing familiar interfaces for the WoT. Unlike SenseShare, the authors emphasize the need for WoT applications not to rely on one given social platform, but rather to be open to multiple such platforms.

Other research goes a step further and proposes an architecture for a Social Internet of Things (SIoT) [10], in which things

[23]Facebook may be used as a user interface container for applications by integrating tabs in Facebook Pages.

become autonomous through embedding rules to manage a predefined set of categories of relations (e.g. objects from the same production batch, objects belonging to the same user). Things are also compliant to any privacy policies imposed by their owners. One of the central problems approached in SIoT-related research is network navigability and finding strategies for search optimization [29]. In our work, we are more interested to integrate things into the Web's social overlay and build open and interoperable STNs. Our model is generic enough to be extended with any set of predefined categories of social relations. In our model, rules, policies and norms may be expressed in a normative layer that is external to the agents, therefore providing greater flexibility for adding new categories of relations, as defined in the SIoT, and for dynamically changing the rules associated to each category.

VI. Conclusions

Modern Web applications are driven by social constructs. We believe that extending these constructs to things, and thus applying the social network metaphor to the WoT, would not only help users manage the envisioned complexity of the WoT, but would also add huge value to the development of WoT applications. In this paper, we have introduced a model for socio-technical networks, which are the building blocks of a semantic socio-technical overlay for the WoT. This overlay may be reliably processed both by people and social things and used for autonomous reasoning. Social things are first-class entities in this overlay. They may autonomously manage their connections, either according to their application logic or rules defined by an external authority, and perform any other domain-specific operations supported by the underlying platforms. We have applied our STN model and ontology in the development of an open STN platform, but also to create STN descriptions for Twitter and Facebook, two of the social platforms with the highest user adoption[24]. We have implemented a software client that may access and use all three platforms based on their semantic descriptions, while being agnostic to their APIs.

References

[1] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[2] D. Guinard, "A web of things application architecture: Integrating the real-world into the web," Ph.D. dissertation, 2011.

[3] S. Mayer, D. Guinard, and V. Trifa, "Searching in a web-based infrastructure for smart things," in *Internet of Things (IOT), 2012 3rd International Conference on the*. IEEE, 2012, pp. 119–126.

[4] B. Ostermaier, K. Romer, F. Mattern, M. Fahrmair, and W. Kellerer, "A real-time search engine for the web of things," in *Internet of Things (IOT), 2010*. IEEE, 2010, pp. 1–8.

[5] M. Blackstock and R. Lea, "Iot mashups with the wotkit," in *Internet of Things (IOT), 2012 3rd International Conference on the*. IEEE, 2012, pp. 159–166.

[6] R. Kleinfeld, L. Radziwonowicz, and C. Doukas, "glue. things–a mashup platform for wiring the internet of things with the internet of services," in *Proceedings of the 4th International Conference on the Internet of Things*, 2014.

[7] A. Pintus, D. Carboni, and A. Piras, "Paraimpu: a platform for a social web of things," in *Proceedings of the 21st international conference companion on World Wide Web*. ACM, 2012, pp. 401–404.

[24] As of March 31, 2015, Facebook reports 1.44 billion monthly active users (http://newsroom.fb.com/company-info/).

[8] M. Blackstock, R. Lea, and A. Friday, "Uniting online social networks with places and things," in *Proceedings of the Second International Workshop on Web of Things*. ACM, 2011, p. 5.

[9] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea, "Reconsidering the social web of things: position paper," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 2013, pp. 1535–1544.

[10] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[11] R. H. Bordini, L. Braubach, M. Dastani, J. J. Gomez-Sanz, J. Leite, A. Pokahr, and A. Ricci, "A survey of programming languages and platforms for multi-agent systems," 2006.

[12] R. H. Bordini, M. Dastani, J. Dix, and A. E. F. Seghrouchni, "Multi-agent programming: Languages, tools and applications," 2009.

[13] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems," *Autonomous agents and multi-agent systems*, vol. 17, no. 3, pp. 432–456, 2008.

[14] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0," http://www.openid.net/specs/openid-connect-core-1_0.html, February 2014, accessed: 2014-05-15.

[15] A. Sambra and S. Corlosquet, "WebID 1.0 - Web Identity and Discovery, W3C Editor's Draft 30 April 2015," W3C IG Editor's draft, Apr. 30 2015. [Online]. Available: https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html

[16] D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.99 (Paddington Edition)," Jan. 14 2014. [Online]. Available: http://xmlns.com/foaf/spec

[17] J. A. Goguen and R. M. Burstall, "Institutions: Abstract model theory for specification and programming," *Journal of the ACM (JACM)*, vol. 39, no. 1, pp. 95–146, 1992.

[18] J. F. Hubner, J. S. Sichman, and O. Boissier, "Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3, pp. 370–395, 2007.

[19] M. Dastani, D. Grossi, J.-J. C. Meyer, and N. Tinnemeier, "Normative multi-agent programs and their logics," in *Knowledge Representation for Agents and Multi-Agent Systems*. Springer, 2009, pp. 16–31.

[20] The OpenSocial Foundation, "OpenSocial Specification 2.5.1," http://opensocial.github.io/spec/2.5.1/OpenSocial-Specification.xml, August 2013, accessed: 2015-05-15.

[21] H. Halpin and M. Tuffield, "A Standards-based, Open and Privacy-Aware Social Web," http://www.w3.org/2005/Incubator/socialweb/XGR-socialweb-20101206/, December 2010.

[22] U. Bojars and J. G. Breslin, "SIOC Core Ontology Specification," http://rdfs.org/sioc/spec/, March 2010, accessed: 2015-05-15.

[23] J. Lieberman, R. Singh, and C. Goad, "W3C Geospatial Vocabulary," http://www.w3.org/2005/Incubator/geo/XGR-geo/, October 2007, accessed: 2014-03-24.

[24] M. Lanthaler and C. Gütl, "Hydra: A vocabulary for hypermedia-driven web apis." in *LDOW*. Citeseer, 2013.

[25] A. Ciortea, O. Boissier, A. Zimmermann, and A. M. Florea, "Open and interoperable socio-technical networks," in *To appear. Post-proceedings of the 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2014.

[26] A. M. Ortiz, D. Ali, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges."

[27] T. Schmid and M. B. Srivastava, "Exploiting social networks for sensor data sharing with senseshare," *Center for Embedded Network Sensing*, 2007.

[28] D. Guinard, M. Fischer, and V. Trifa, "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, 2010, pp. 702–707.

[29] M. Nitti, L. Atzori, and I. P. Cvijikj, "Network navigability in the social internet of things," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*. IEEE, 2014, pp. 405–410.