On the Role of Semantic Descriptions for Adaptable Protocol Stacks in the Internet of Things

Vangelis Gazis, Konstantinos Sasloglou, Andreas Merentitis, Kostas Mathioudakis
AGT Group (R&D) GmbH, Hilperstrasse 35, 64295, Darmstadt, Germany
e-mail: {vgazis,ksasloglou,amerentitis,kmathioudakis}@agtinternational.com

Abstract—With Internet of Things applications covering several application domains (e.g., Smart Grid, Smart City, e-Health) with a diverse range of requirements, it is now becoming increasingly recognized that the variety of wireless technologies employed for Machine-to-Machine (M2M) communication is here to stay. Due to the heterogeneity of M2M appliances and communication modes, research has been investigating autonomic principles as an efficient instrument for the integration of large populations of dissimilar M2M devices to the M2M gateway and the overall end-to-end M2M architecture. Herein we present the status of semantic support for the IoT field by briefly surveying research efforts in IoT dealing with semantic concerns at the architecture level. We also summarize the state of the art in adaptable protocol stacks and make the case for their application as an efficiency enabler instrument for IoT. To this end, we present our work in an ontology used to semantically describe the adaptation options of the dynamic protocol stacks for future communication devices. We present the ontology design principles and artifacts and elaborate on the level of semantic support it enables for IoT. Finally, we conclude the paper with our next steps in the area of semantic support for IoT.

Index Terms—Internet of Things, Machine to Machine Communications, Semantics, Protocol Stacks, Dynamic adaptation

I. INTRODUCTION

The Internet of Things (IoT) is commonly understood as a pervasive and self-organising network of connected, identifiable and addressable physical objects that by leveraging embedded intelligence, enable application development in and across key vertical market sectors. It is estimated that, by 2020, the annual global economic potential of the Internet of Things across all affected market sectors will range from USD 1.4 trillion to USD 14.4 trillion [1]. The rise of Machine-to-Machine (M2M) communications in support of Internet of Things applications, has resulted in significant research and development efforts, by both academia and industry, into communication modes suitable for low-rate low-power applications scalable to billions of nodes.

A. Technology diversification

The phenomenal growth of wireless technologies over the last decade has spawned a wide gamut of wireless systems, from wide area cellular (e.g., GSM, GPRS, EDGE, etc.), to metropolitan area (e.g., IEEE 802.16 WiMAX, 3GPP EPC LTE/LTE-Advanced, etc), broadband local area systems (e.g., IEEE 802.11a/b/g/n), broadcast systems (e.g., DAB, DVB-T) and short-range connectivity systems (e.g., Bluetooth, IEEE 802.15.4). These standards are now being amended in order to support the particular requirements of M2M communication. For instance, 3GPP has included support for so-called Machine-Type Communications (MTC) in its feature set for 3GPP system releases from R10 onwards [2]. Similarly, the IEEE 802.11ah upcoming extension to the IEEE 802.11 standard is targeting applications characterized by short-burst data transmissions in outdoor settings [3].

With M2M targeting several application domains (e.g., Smart Grid, Smart City, e-Health, etc.), it is now being increasingly recognized that the variety of wireless technologies and its (even greater) accompanying variety of M2M devices is here to stay. For instance, the options of wireless access standard for sensor network installations now include IEEE 802.15.1 (Bluetooth), IEEE 802.15.4, Zigbee, IEEE 802.11a/b/g/n, RFID, DASH7 [4], Weightless [5] and Z-Wave [6], to name a few.

As the heterogeneity of M2M devices and M2M networks complicates their integration at the M2M gateway level, research has been investigating autonomic principles as an efficient instrument for the integration of large populations of dissimilar devices to the M2M gateway and the overall end-to-end M2M architecture. On the networking level, autonomous operation is founded upon functions that support the flexible configuration of the protocol stack according to specific service requirements and enable its dynamic adaptation during runtime. On the other hand, integration concerns on the end-to-end system architecture concerns mandate that these adaptation capacities are defined and instrumented in a technology agnostic manner, so that further evolution of the system is not hampered by legacy bindings.

The rest of the paper is structured as follows: Section II surveys prior work in a twofold manner: first, by presenting research efforts addressing semantic concerns at the architecture level for the Internet of Things, and, secondly, by briefly presenting research efforts in adaptable protocol stacks. Section III introduces an ontology to describe the adaptation options for protocol stack structures by presenting the respective classes and associations in UML notation. We then discuss the application of the ontology to describe the protocol stacks for the 3GPP M2M features and their assembly options. Finally, Section IV concludes the paper with directions for future work.
II. STATE OF THE PLAY

Over the last decade, there has been a growing research interest in the Internet of Things, a disruptive technology, according to the US National Intelligence Council [7]. An early definition for the Internet of Things envisioned a world where computers would relieve humans of the Sisyphean burden of data entry by automatically recording, storing and processing in a proper manner all the relevant information about the things involved in human activities [8]. The European Commission envisions it as an integrated part of the Future Internet where things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts [9]. The use of standard technologies in the World Wide Web to instrument the Internet of Things is frequently referred to as the Web of Things [10].

A. Research efforts in IoT

In the European Union, several collaborative research efforts have addressed the Internet of Things. The FIREworks project was a coordination and support actions funded by the European Union (EU) dealing with the interworking of testbed activities in Europe and their link to related global initiatives [11]. Its work is now handed over to the Future Internet Research and Experimentation (FIRE) initiative that strives to accelerate the diffusion of technologically innovative ideas from revolutionary research efforts through experimental validation in large-scale federated testbed facilities [12]. SmartSantander is an indicative FIRE initiative that aims to provide a city-scale experimental research facility utilizing Internet of Things technologies to support innovative applications and value-adding services for a Smart City [13].

The CASAGRAS and CASAGRAS2 projects are coordination and support actions funded by the European Union (EU) to foster global RFID activities and related standardization [14]. Their objective is to address the major international issues that are important in providing the foundations and cooperation necessary for realizing the Internet of Things as a global initiative. They thus provide a liaison between research carried out in the IERC cluster and the RFID in Europe project to standardization bodies, including the ETSI Technical Committee (TC) on Machine to Machine (M2M), GISFI, CESI YRP/UNL, AIM, CEN and ISO.

The SPITFIRE project aims to bridge the gap between the level of semantic support available in the World Wide Web and Linked Data domains and the informational description of the physical world as quantified by sensor measurements in an Internet of Things context [15]. Semantic Web technologies are thus leveraged to develop suitable vocabularies integrating descriptions of sensor devices and physical things to the Linked Open Data architecture. Notably, the state of such described things is inferred from information acquired through sensor devices. This enables a more dynamic and contextualized mode of interaction where relevant things are identified on the basis of their state.

IoT-A is an EU lighthouse project for the IoT architecture [16]. One of the key concepts in its design proposition is the notion of a Virtual Entity as the digital counterpart of a physical entity. The IoT-A information model uses the Virtual Entity as its core data structure which is used to access and control devices associated to physical entities (e.g., actuators). The disadvantage of this approach is that sensor data tagged with geolocation information are not efficiently supported by IoT-A reference model. In addition, explicit support for semantics is not included in the model.

The HOBNET project is striving to maximize the use of FIRE platforms for the development of innovative applications for energy efficient smart buildings [17]. Its approach is to conceive, design and evaluate scalable IPv6/6LoWPAN network architectures to support Future Internet services and applications. To this end, it defines an interface layer between the Building Management System (BMS) and FIRE experimentation. Focusing on scalability leaves semantic concerns out of the scope of work in HOBNET.

The IoT6 project aims at exploiting the potential of IPv6 and related standards (6LoWPAN, CORE, COAP, etceteras) to realize the full potential of the Internet of Things [18]. Its task objectives include research, design and development of a highly scalable Service-Oriented Architecture (SOA) based on IPv6 to achieve interoperability, mobility, integration to cloud computing and distribution of intelligence among a large population of heterogeneous smart things components, applications and services. Here IoT6 focus is on proposing a complete IPv6 enabled architecture for IoT, rather than ensuring semantic support for the IoT architecture.

The SEMSORGGRID4ENV project defined a three tier architecture relying on service discover and data integration and built on Web Services to support rapid development of applications using heterogeneous data sources (e.g., sensor networks, database tables, etc.). The Application tier comprises services that provide domain-specific support to applications and bridge the gap between the service-oriented and resource-oriented paradigms. The Middleware tier instruments services that provide value through the use of semantic techniques. The Data tier consists of services that provide access to data which can be published either through a stream (e.g., as acquired from a sensor network) or through data stores (e.g., as retrieved from a database). In the semantic front, the project has developed an extension of RDF called stRDF that enables spatiotemporal annotations as part of resource descriptions and an extension of the W3C SPARQL language called stSPARQL that provides the querying capacity over spatiotemporal capabilities [19]. We consider this by far the most notable contribution focused on semantics by a research project.

B. Research efforts in adaptable protocol stacks

The capacity to adapt (i.e., adaptability) is determined by the respective architecture and the degree of flexibility it supports. From the viewpoint of software engineering, protocol
stack flexibility has been engineered by means of a) adaptable protocols, or, b) composable protocols [20].

Adaptable protocols, comprise two parts: a generic layer that implements protocol functions common to multiple protocols (e.g., framing) and a custom extension layer that implements additional protocol functions required to realize the function of a specific protocol. As the smallest unit of design is an entire class of protocol layers (e.g., layer protocol protocols) the analysis granularity of adaptable protocols is undeniably somewhat coarse. Conduits+ [21] is a black-box framework (i.e., class libraries) for network protocol software. The Conduits+ design organizes the common structure and behavior of protocol specific parts in reusable software components. The latter can be appropriately combined into different protocol implementations without any source code modifications. Based on the Java platform, JChannels [22] exploits dynamic class loading, concurrency and code portability to support the development of configurable modular protocol stacks by specifying classes for protocol layer and protocol stack software development. HIPPARC [23] proposed the ESTEREL development environment for tasks dealing with a) the specification of protocols in a natural language, b) the validation of protocol specifications by means of formal languages and verification techniques, and, c) the implementation of protocol functions.

In composable protocols the overall protocol functionality is broken down into elementary protocol functions, which are subsequently used as the building blocks for a customized protocol stack. Depending on the particular architecture style and inter-layer communication paradigm, the composition of protocol functions then follows a flat or hierarchical model. The x-kernel framework [24], [25] was one of the first research efforts in customizable protocol stacks and configururable communication services. In the x-kernel framework, micro-protocol objects are assembled according to a graph definition to realize a particular protocol functionality. The Cactus and Appia systems [26], [27] extended the x-kernel concept to provide a hierarchy in the composition mechanism of protocol objects based on specific QoS requirements. Coyote [28] divides protocol functionality into orthogonal functions that are implemented as fine-grain micro-protocol modules that are then assembled into composite protocols. DROPS [29] proposed adaptable protocols composed of micro-protocol modules that realize elementary functions. Horus [30] is an object-oriented protocol composition framework that enables the development of pluggable protocol software libraries. The FRACTAL framework [31] employs a recursive component model that enables sharing of sub-components between components. The THINK framework [32] supports the build of flexible component-based operating system kernels. By considering an entire protocol layer as a component, DiPS/CuPS [33] provide for the dynamic adaptation of protocol stacks according to a given set of requirements. In [34] the authors position self-management and self-adaptation as an essential capacity for IoT devices. This is achieved by a monitoring agent that collects measurements related to protocol operation and assesses the optimality of the current control parameters by evaluating a utility function. When deemed necessary, adaptation is realized by coordinating the policies applied to selected elements of the protocol stack by assigned controller agents. For an extensive survey of research efforts in the area of adaptable protocol stacks, the reader is referred to [35].

III. AN ONTOLOGY FOR ADAPTABLE PROTOCOL STACKS
A. Design-rationale

The layering of protocol layers into protocol stacks is based on the assumption that each protocol layer is impervious to the functionality embedded within other protocol layers. In general, each protocol layer offers specific services to higher protocol layers and expects specific services from lower protocol layers. Different protocol layers exchange control and data information via Service Access Points (SAP) instances. Each SAP provides access to a selected subset of services offered to higher protocols and the respective SAP primitives. However, these descriptions lack semantic support. As a result, preserving the semantic consistency of a protocol stack (i.e., an unambiguous description of the services realized in and offered by protocol stack) across adaptations becomes impossible. This would require at least a semantic description of the set of services realized by each individual protocol layer is necessary, so that basic consistency checks regarding adjacent protocol layers can be made. In turn, this would enable the development of algorithms than validate the semantic consistency of a particular protocol stack in an incremental (i.e., in a layer-by-layer) manner. In this section we provide an overview of an ontology we developed to describe individual protocol layers in a manner that supports the incremental assembly of protocol stacks by validating the consistency of adjacent protocol layers with regard to the services provided and required by each particular protocol layer.

B. Overview of the protocol stack ontology

At its core, our ontology is structured around classes that describe services, products, protocols and protocol stacks. There are also concepts that describe a particular binding of a service to protocol, i.e., using a particular protocol (out of multiple alternative ones) to realize a particular service. And also, classes to describe the implementation of a protocol along with its deployment artifacts. The relationships that bring these concepts together capture the 'provides' and 'realizes' semantics that arise between each individual protocol layer and the sets of services it provides and realizes, respectively.

Product, the root abstract class in our ontology, is further subclassed by the Service and Protocol classes. The Service class refers to a precisely defined functionality (e.g., IP-in-IP encapsulation) as specified by its textual description attribute. The latter may bear formal semantics (e.g., OMG IDL, ITU SDL), as long as those semantics support a textual representation. The Protocol class represents the behavioral specifications of a single protocol and includes author, version, release, description and summary attributes. It stands as an
abstract concept for the specification of a protocol (rather
than for the specification of what services the protocol real-
izes). Thus it provides a first-class abstraction for protocols
developed and published by authoritative bodies, e.g., the
GPRS Tunneling Protocol (GTP) published by 3GPP. The
ProtocolStack class serves as a container for Protocol instances
related by standardisation. Thus it models protocol stack
specifications that reference (as opposed to specialise) other
protocols, possibly published by another authoritative body
(i.e., a standardization body different to the one responsible
for the specification of the protocol stack in question). For
example, the M2M specifications published by ETSI is a stan-
dard whose protocol stacks leverage protocols (e.g., TR-069)
developed and published by the BroadBand Forum (BBF).
The ProtocolImplementation class of Product that refers to a
software artifact that realizes the functionality of at least one
Protocol instance. Primarily, it models the real-life software
implementation of a protocols functionality. However, it may
also represent functionality that is not associated to a particular
protocol, such as utility functionality (e.g., integrity checks,
buffer management, etc.). A protocol implementation may be
developed in different programming languages (e.g., C, C++)
and packaged in various deployment formats. These concerns
are modeled in the DeploymentArtifact class (Figure 1). Due
to space constraints, the reader is referred to [36] for a detailed
in-depth presentation of the protocol stack ontology.

C. Ontology instrumentation in OWL-DL

To validate our ontology representation, we populated the
instance graph of the protocol stack ontology with information

---

**Fig. 1:** The RCM ontology for dynamically adaptable protocol stacks.

**Fig. 2:** Example protocol stacks of an M2M system with LTE backend.
based on the ETSI M2M protocol stacks and the 3GPP protocol stacks relevant to the MTC features. The latter define a sufficiently wide set of instrumentation options, thus fitting the description of valid protocol stacks as an assembly of appropriate protocol layers. For instance, the device management protocol stacks at the M2M gateway may comprise an BBF TR-069 solution (fitting a wireline sensor network), an OMA-DM solution (fitting a wireless sensor network), or both. Fig. 2 shows the protocol stack on the user plane of an example configuration of an M2M system with a 3GPP LTE backend. Please note that the illustration is simplified, as multiple M2M network technologies may coexist in the so-called last-mile network segment. To ease the reader’s understanding, selected routing protocols in the control plane (i.e., RPL, BGP) are included in the illustration in grey highlight.

Fig. 3 illustrates the M2M segment of the overall OWL ontology linked together by the respective relationships. Instances of the Service class are prefixed by ‘S’ while instance of the Protocol class are synonymous to the respective protocol (and prefixed by ‘P’). Between a Protocol instance and a Service instance, range lines indicate a ‘provides’ relationship while blue lines indicate a ‘requires’ relationship. A segment of the respective OWL ontology data in XML representation is included in Ontology 1. These ontological descriptions decouple algorithm logic from information representation matters and thus facilitate the development of algorithms driving autonomous adaptation regardless of the technological details of the supporting instrumentation platform details (i.e., operating system, information model, data representation).

D. Proof and validation

We validate the proposed ontological description by proving that the proposed ontology suffices for the description of any protocol stack in the M2M domain. To this end, we elaborate on the abstract description of any protocol stack as a protocol graph and demonstrate that the selected ontology realizes an isomorphic mapping between the protocol graph and the ontology graph.

We begin by observing that a multi-protocol stack is typically represented as an abstract protocol graph, where graph vertices represent distinct protocol layers while graph edges represent an adjacency relationship between a pair of protocol layers. Let us consider a protocol stack \( S = \{ p_1, p_2, \ldots, p_N \} \) where \( \forall i \in \{1, 2, \ldots, N - 1\} \) protocol layer \( p_i \) resides directly below (i.e., is adjacent to) protocol layer \( p_{i+1} \). In abstract form, this is represented by an undirected graph \( G = [E, V] \), where \( E = \{ v_1, v_2, \ldots, v_N \} \) the set of vertices and \( E = \{ e_1, e_2, \ldots, e_M \} \) the set of edges in the graph, if and only if for every pair of adjacent protocol layers \( (p_i, p_j) \) in the protocol stack, a distinct edge \( e_k = (v_i, v_j) \); \( e_k \in E, v_i, v_j \in V \) can be identified in the protocol graph and there is no edge \( e_k \in E \) without a matching pair of adjacent protocol layers in the protocol stack. A multiprotocol system (i.e., one of multiple protocol stacks) is also represented by a protocol graph under this convention.

Ontology 1 A segment of the OWL ontology in XML.

```xml
<DOCTYPE Ontology [  
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" [  
]>  
<!Declaration>Class IRI="#Service"/></Declaration>  
<!Declaration>Class IRI="#Product"/></Declaration>  
<!Declaration>Class IRI="#Protocol="/</Declaration>  
...

<Declaration>ObjectProperty IRI="#isProvidedBy"/></Declaration>  
<Declaration>ObjectProperty IRI="#isRequiredBy"/></Declaration>  
...

<Declaration>NamedIndividual IRI="#P_802154"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_HTTP"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_TCP"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_TLS"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_802154"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_HTTP"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_TCP"/></Declaration>  
<Declaration>NamedIndividual IRI="#P_TLS"/></Declaration>  
...

<SubClassOf><Class IRI="#Protocol"/>  
<Class IRI="#Product"/></Class>  
</SubClassOf>  
<SubClassOf><Class IRI="#ProtocolStack"/>  
<Class IRI="#Protocol"/></Class>  
</SubClassOf>  
<SubClassOf><Class IRI="#Service"/>  
<Class IRI="#Product"/></Class>  
</SubClassOf>  
...

<Declaration>NamedIndividual IRI="#S_Encryption"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Forwarding"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Link"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Network"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Physical"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Session"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Transport"/></Declaration>  
<Declaration>NamedIndividual IRI="#S_Tunneling"/></Declaration>  
...

<ObjectSomeValuesFrom><ObjectProperty IRI="#provides"/></ObjectSomeValuesFrom>  
<ObjectSomeValuesFrom><ObjectProperty IRI="#requires"/></ObjectSomeValuesFrom>  
<ObjectSomeValuesFrom><ObjectProperty IRI="#provides"/></ObjectSomeValuesFrom>  
<ObjectSomeValuesFrom><ObjectProperty IRI="#requires"/></ObjectSomeValuesFrom>  
...
</Ontology>
```
In representing a given protocol stack in the RCM ontology, we apply the mapping convention illustrated in Fig. 4. In particular, each pair of adjacent protocol layers \((p_i, p_{i+1})\) is mapped to instances of the ontology concepts as follows:

- Protocol layer \(p_i\) is mapped to an instance \(P_i\) of the Protocol class.
- Protocol layer \(p_{i+1}\) is mapped to an instance \(p_{i+1}\) of the Protocol class.
- The stratification of protocol layer \(p_i\) below protocol layer \(p_{i+1}\) is mapped to an instance \(p_{(i,i+1)}\) of the Service class.
- An instance of the provides relationship is created from the instance \(P_i\) of the Protocol class to the instance \(p_{(i,i+1)}\) of the Service class.
- An instance of the requires relationship is created from the instance \(P_{i+1}\) of the Protocol class to the instance \(p_{(i,i+1)}\) of the Service class.

For each pair of adjacent protocol layers, the aforementioned mapping extends the corresponding protocol graph by one additional vertex and a pair of edges adjacent to it. Being reversible, this particular mapping preserves the isomorphic properties of the protocol graph in the sense that the original one-to-one relation to the original protocol stack is not invalidated; hence the latter can be consistently represented through the RCM ontology. Consequently, the aforementioned mapping is sufficient to represent any protocol stack in the notation of the RCM ontology.

IV. CONCLUSIONS

The vision of a large scale IoT systems that encompasses billion of M2M devices and scales gracefully with diversified traffic patterns implies a wide range of operational parameters.

This requirement cannot be accommodated by a single protocol stack, due to the inherent trade-off between flexibility and efficiency in protocol stack operation. Therefore, this vision is best served by a versatile networking infrastructure capable of instrumenting on demand the set of protocol stacks most suitable to accommodate the given traffic patterns. Achieving this requires system support for the dynamic adaptation of individual protocol layers as well as entire protocol stacks in a manner semantically-rich manner. To this end, we have introduced an ontology to describe the alternative ways that
individual protocol layers can be successfully assembled into protocol stacks realizing specific services. We validated the descriptive capacity of our model against the protocol stack standards of 3GPP that consider M2M network applications. Extension of our work will consider the protocol stacks of the M2M gateway currently being standardized in oneM2M for semantic description and the further validation of protocol reconfiguration use cases.

REFERENCES


