Review of Interoperability approaches in Application Layer of Internet of Things

Varun M Tayur
Dept. of Computer Science
Jain University, Bangalore
Varun.tayur@jainuniversity.ac.in

Suchithra R
Dept. of Computer Science
Jain University, Bangalore

***Abstract*—Internet of Things (IoT) encompasses numerous old and new technologies under its umbrella; which means there is a huge diversity in types of devices, protocols and mechanisms that are to be supported. Things by themselves need to inter-network and collaborate to provide the necessary level of service expected out of them, working in isolation does not help in many cases. Interoperability issues have been attempted to be solved in broadly five different ways - publishing standards, reference architectures and frameworks, defining protocols and media-type standards and by using abstract interface definition languages and semantic ontologies. An interoperable environment requires a well-established data-model to provide properties, behavior and message formats and a framework to support selecting the right protocol along with an ontology indicating what the data means based on the context. Application layer is the primary interface for interacting with the systems and services; hence it makes it important to provide an interoperable environment at this layer enabling dynamic composition and rapid development of new services. IoT requires a common construct that is adaptive, expressive and dynamic in nature abstracting out the communications, behavior and models that allows diverse set of device stacks to interoperate. Such constructs enables device vendors to publish changes systematically and incrementally without affecting the advances in the rapid development of applications.**

Keywords—interoperability;IoT interoperability methods; IoT frameworks; IoT standards;IoT semantics

I. INTRODUCTION

Interoperability has been a major challenge for IoT's rapid advancement. This can be primarily attributed to the existence of a number of diverse set of technologies both old and new. Interoperability is critical in IoT as most of the communication is machine to machine (M2M). The challenge has remained a tough nut to crack even with the existence of many standards and reference architectures. Interoperability has also been attempted to be solved by using Semantic Ontologies which provide deeper understanding of the raw sensor data enabling machines to take decisions based on simple rules. Design of devices for interoperability from ground up can also be achieved by defining comprehensive centralized metadata and defining abstract models out of them that can be automatically generated for multiple programming languages.

Standards have been proposed by various alliances and consortia – For example the IP Smart Objects (IPSO) alliance proposes to have a common mechanism for interaction between all smart objects identified by the IP, European Telecommunications Standard Institute (ETSI) has proposed end-to-end M2M standardization. IoT-A forum has proposed a reference model and architecture for IoT which has been used as a model for one of the implementation's called OpenIoT. AllSeen alliance has come up with an open, universal, secure and programmable connectivity framework with the aim of enabling interoperation with other AllJoyn enabled products. Similar framework called IoTvity with the goal of an interoperable IoT has been defined by the Open Interconnect Consortium (OIC).

Semantics provide a different dimension to the data interoperability at a higher level than what just raw data gives; Sensor Semantic Network (SSN) provides a comprehensive set of ontologies for interpreting the sensor data. SenML (Sensor Markup Language) has proposed to provide a common media-type for sensor data exchange – it has proposed to use JSON. Design of devices for interoperability from ground up can also be achieved by defining comprehensive metadata for all sensors as done by eclipse Vorto project or the eclipse Ponte project which proposes to provide a common ground for application layer protocols like CoAP, MQTT and HTTP.

The current IoT and in general M2M is too much diversified. Diversification is a result of many mechanisms, often some custom protocols in use due to the nature of solutions that have been developed over time. Many of the standards have been an afterthought after the actual solution has been developed while in many cases the standard isn't generic enough for adoption. With the introduction of more standards there is definitely a churn in the adoption cycle. At the same time the rapid advancement in hardware is influencing development of more efficient solutions. The standards, reference architectures and interface definition languages have to a large extent abstracted out these varied differences. The advancement of IoT or in general M2M can be made possible when the applications and solutions are built with a focus on interoperability at all layers of the application stack. A formal specification or a language construct can be defined at the application layer. This specification can be used to generate bindings at various layers of the application stack automatically, these can be used to expose standard interfaces which are aware of the communication standards, data types etc. Such mechanisms can foster development of autonomous systems yet be interoperable.

II. RELATED WORK

A. Application Layer Transport

HTTP is a stateless protocol developed for use in distributed hypermedia systems. Its request methods, error codes and headers have been used for building systems independent of the data being transferred. The HTTPs Uniform Resource Identifier (URI) naming scheme has been effectively modeled to uniquely address the devices and sensors by IPSO proposal. The real challenge for IoT in its adoption of HTTP regardless of its immense popularity is the requirement of rather bulky stack and usage of TCP. This has led to the development of Constrained Application Protocol (CoAP) which is tailored for resource constrained nodes and Low-powered Lossy Networks (LLN). CoAP is developed to use UDP unlike TCP/HTTP and also retains the support for REST/URI access. Unlike HTTP, CoAP is an optimized for the IoT with a focus on efficiency very much essential for resource constrained environments. While HTTP and CoAP are comparable transport mechanisms in the Application layer they are largely used for one-to-one communication. In some cases where many devices want to communicate with each other, a message board pattern would be useful, that is achieved by few of the publish/subscribe paradigms like MQTT, AMP. Table 2 provides a comparison of the options available at the application layer.

B. Standards and Frameworks

IPSO defines a RESTful mechanism to address sensors/devices [1] working over CoAP/HTTP. Sensors are addressed by a well-defined URIs called the Function Sets [2]. Function-sets are published with well-defined root-paths for Device, General Purpose IO, Power, Load Control, Sensors, Light Control, Message, Location and Configuration. For ex. '/dev' gives complete information about the smart device including manufacturer, model, serial-number etc. The URIs resource-types, data-types, interfaces and units are specified as per the IETF CoRE link format specification. The function-sets can be used to configure an end-point, discover configuration parameters and observe sensor events. For interoperability reasons the object model is based on the Open Mobile Alliance Light Weight M2M (OMA LWM2M) specification. Lightweight M2M (LWM2M) is a device management protocol [3] specialized for IoT devices considering the factors such as low bandwidth and Lossy networks. LWM2M application protocol implements the interface between M2M device and M2M Server based on CoAP/Datagram Transport Layer Security (DTLS) on UDP and Short Message Service (SMS). LWM2M defines a simple object model running on the CoAP REST API, a device management architecture using REST objects, functions for client registration, service enablement, reporting, including control of asynchronous notifications.

ETSI/oneM2M alliance has proposed a standard for end-to-end M2M Communications [4]. The functional architecture enables development of interoperable applications by providing a comprehensive end-to-end description of the functional entities, the related reference points and the service capabilities, information model, security, and management, charging and implementation guidance. OneM2M also has a published semantic ontology [5] that can enable integration with other external systems. Semantics can enable automatic discovery, interpreting and using M2M data from different sources thus enabling creating higher level services.

AllJoyn is open-source software framework [6] for applications that can discover nearby devices, communicate with each other directly regardless of brands, categories, transports, and OSes without the need of the cloud. It is designed to run on multiple platforms, ranging from small embedded RTOS platforms to full-featured OSes. It supports multiple language bindings and transports (BLE, Zigbee, Z-Wave). One advantage of the AllJoyn framework is that devices can be on the local network without the need for the cloud to function reducing the number of devices connected to the internet. IoTivity [7] is yet another competing framework similar to the AllJoyn and it provides discovery, data transmission, data management and device management functions. Discovery supports multiple discovery mechanisms, data transmission is based on a messaging and streaming model and supports information exchange and control, data management supports collection, storage, and analysis of data from various resources and device management supports configuration, provisioning and diagnostics of device. IoT-A forum has proposed an architectural reference model [8] (ARM) for establishing a common ground for IoT applications. The IoT ARM should provide a common structure and guidelines for dealing with core aspects of developing, using and analyzing IoT systems. The IoT-A allows developers to make choices that make the architecture fit the device they are developing and are at the same time providing them with guidelines to follow. The IoT ARM does not guarantee interoperability between any two concrete architectures, but it is a tool in helping to achieve interoperability between IoT systems. OpenIoT is based on the IoT-A ARM and it provides a middleware for getting information from the sensor clouds without knowing actually about the sensor type.

Weave [9] is a common language for IoT devices created to provide a shared understanding between devices, humans, and smart-phones. It is based on standardized schemas that are used to describe devices' properties. Weave is designed to be cross-platform by design which enables a smart-phone to turn on an oven on another platform. Weave can be used both on existing software stacks as well as over Android stacks. Thread [10] is a protocol based on mesh networks designed for secure and reliable connection of hundreds of products around the home designed specifically for the home environments with self-healing mesh networks. Interoperability is achieved by design using open standards and IPv6 technology with 6LoWPAN as the foundation. It is designed to support a wide variety of products for the home: appliances, access control, climate control, energy management, lighting, safety, and security. Thread seems to be yet another protocol developed by consortia of companies with focus on interoperability; however its adoption is going to be a challenge since it requires major software and hardware updates to be in place.

Vorto [11] provides facility to define the capabilities of the devices in entirety – which is called the information model [12]. This information model is stored in a central repository

which can be used by vendors to build solutions reusing them. Automatic code generators ensure the model is translated into appropriate code representation which simplifies development and enables reuse and interoperability without the trouble of multiplicity in data-types and formats. Domain Specific Language (DSL) can be developed using the information models as functional building blocks. Such High-Level constructs enable rapid development of applications and superior interoperability. Ponte [13] provides a unique capability that allows publishing and receiving the data using HTTP, MQTT and CoAP with the ability to mix and match the various protocols ex. It enables request to be submitted with CoAP and subscribe to events via MQTT. Using HTTP and things are available at http://<your ponte>/resources/<your thing>. Using MQTT, things are available at mqtt://<your ponte>/<your thing>. Using CoAP, things are addressed using coap://<your ponte>/r/<your thing>. The real-world events can be instantly made available using the mechanism of publish/subscribe brokers.

The raw sensor data itself cannot be "understood", it needs some metadata; SenML requires no additional metadata or schema for building generic applications that can use the value and unit for wide array of uses [14]. SenML uses JSON format that gives a good trade-off between simplicity and efficiency. SenML being an IETF standard makes it a good choice for building interoperable applications that can be simple, autonomous and yet be efficient.

C. *Semantic Ontologies*

Ontologies are formal specifications of how to interpret the relations and attributes within the data. Ontology consists of classes, relations and attributes. Classes define abstract concepts and they can be extended to be more specific in scope by having one or more children. The Classes have attributes and these can represent the classes' properties and characteristics. Relations between the classes or their properties are the edges that connect them. One such ontology is the Semantic Sensor Network (SSN) [15] ontology. It describes sensors and sensor observations, and related concepts. It does not cover every thinkable aspect of the IoT, but what it does not cover can be included from other ontologies via OWL imports. Semantic Sensor Observation Service (SemSOS) [16] models the domain of sensors and sensor observations in a suite of ontologies, adding semantic annotations to the sensor data, using the ontology models to reason over sensor observations, and extending an open source SOS implementation with a semantic knowledge base. Connecting IoT Sensors to Knowledge-Based Systems by Transforming SenML to RDF is discussed in [14]. Converting SenML to RDF enables higher level of machine intelligence whose knowledge can be used to take decisions leading to an interoperable world across domains.

III. DISCUSSION

Standards, Domain Specific Languages/Code Generation and Semantics attempted to solve the interoperability challenge in unique different ways. Standards and reference architectures like IoT-A, IoTvity provides an efficient tested blueprint for harmonious existence of devices in the IoT space. M2M communication is enabled by the use of standardized interfaces and data-types published in the blueprints. The challenge in the current scenario is that there are many standards bodies and vendor alliances that are building parallel specifications each attempting to solve the similar set of issues. Major challenge is now in getting the adoption rates up. However, multiple such standards bring in more challenges to interoperability as they are inherent incompatibilities in the interfaces and data-types. Reference Architectures published by IoT-A forum, AllSeen Alliance, OIC consortium have many things in common but there are differences too leading to multiple implementations like OpenIoT, IoTvity etc. The fragmentation in the frameworks and implementation is another cause of interoperability issues plaguing rapid adoption of IoT.

Domain Specific Languages (DSL) and Code generation based on a shared information model can reduce differences between the models and aid in development of interoperable solutions. The Eclipse Vorto provides a shared repository of such information models and schemas which describe the sensors and devices attributes which can be reused by other vendors and be extended without worrying about how it will be used. Weave is another high level language construct that defines the actions, attributes that enable multi-vendor interoperation. Thread is consortia of vendors that have developed a protocol based on the IPv6 standards and based on the popular MAC protocols. DSL and code generators alone cannot solve the problem of interoperability though it can eliminate the problem of multiplicity by introducing a shared model that can be extended by multiple vendors keeping the base interoperation active. Such solutions can lead to development of more autonomous systems as interfaces and data-models are well known to all vendors and developers.

Semantics and ontologies provide detailed information about the data model itself. SSN defines IoT/sensor specific ontologies required for IoT and allied areas, though it is not very comprehensive other ontologies can be linked with by importing it. SemSOS also defines the standard ontologies for sensor/actuator data observations. Interoperability can be addressed with ontologies as it defines a well-defined construct enabling machines to make sense of the sensor data instead of just observing and processing them. However, this requires converting SenML/JSON/binary object types into RDF so that semantic techniques can be applied on them for aiding machine understanding and analysis. This also means actions taken based on the analysis based on RDF needs conversion back to SenML/JSON or data-types that the device can understand. Eclipse Ponte can be useful in cases where application data can be seamlessly transported using HTTP, CoAP or MQTT.

The requirement for Interoperability in IoT cannot be satisfied with just one of the approaches described above but with the combination of all these techniques. It is difficult to have a mechanism that can address the requirements of all the IoT devices and sensors. Table 5 shows the requirements for interoperability in IoT – specifically at the application layer. The primary requirement for IoT is a data model can be consumed directly for automation. The data-model should essentially provide rich support for expressing the properties and behavior together with being able to use the semantic

knowledge base. Adding to all these the standards are required to supplement these interfaces and data models so that there is one of its kind which is extensible to meet most requirements. Figure 1 shows the requirements for high level constructs to be available at the design of the application so that workflow definitions can be made possible. Protocols at the application layer can be HTTP, CoAP, MQTT. Message format specifies the encoding and structure of the data – which can be in JSON, XML or Binary. Semantics are made available via established ontologies which work on the raw sensor data converted into RDF or similar formats for interpretation of the meaning and context. Behaviors indicates the list of operations either configuration or management that are available and it must be context specific. Properties define the list of attributes and properties of the device that can be used for configuration and operations (example can be vendor name, light on/off status).

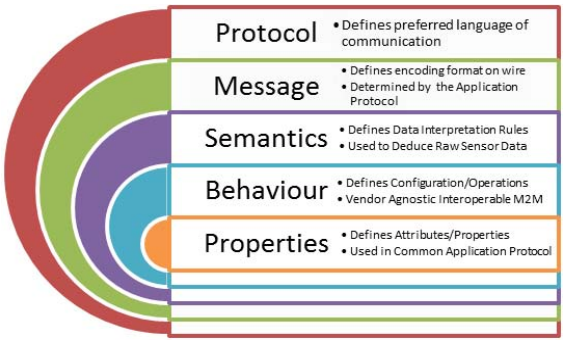

Figure 1: Application layer interoperability requirements

Table 1 presents the summary of various frameworks/standards and the support that they provide for composing the workflows. oneM2M and openIoT provides most of the requirements required for composing workflows dynamically.

Table 1: Application Layer frameworks/standards comparison

Standard/ Framework	***Features***	***Application Layer Interoperability Requirements***				
		Properties	***Behavior***	***Semantics***	***Message***	***Protocol***
IPSO	IP based addressing of smart objects addressed using standard HTTP verbs and URIs.	Yes	Yes	No	No	No
ETSI/ oneM2M	Provides ontologies that can be used to deduce context, standard operations and models can enable workflows to be defined	Yes	Yes	Yes	Yes	No
IoT-A/ OpenIoT	Reference Model provides an end-to-end device management solution from the cloud with no explicit support for workflow building	Yes	Yes	No	No	No
AllJoyn/ IoTvity	Custom data-models and behaviors – challenge for interoperability and definition of workflows	Yes	Yes	No	No	No
LwM2M	Provides only management interfaces and operations and workflow definition not possible	Yes	Yes	No	No	Yes
SenML	Provides only the data-model in JSON format, no schema is available and the interpretation of data is entirely left to the application	Yes	No	No	No	No
Weave	Language based control of multi-vendor device communication, workflow composition with heterogeneous device set is possible but lacks context awareness	Yes	Yes	No	Yes	Yes
Thread	Reliable communication protocol for the Things with multi-protocol support – no evidence of direct support for workflows, it can be used in the lower layer of the communication for seamless workflow execution	Yes	Yes	No	Yes	Yes
Vorto	Shared Information model ensures interoperability and abstract models gives flexibility to define workflows but lacks context awareness	Yes	Yes	No	No	No
Ponte	Provides abstraction for various application protocols with no direct use in composing workflows	No	No	No	No	Yes
CoAP	Built on same lines of HTTP, interoperability with some additional brokers, content negotiation, REST/URI support, supports observer pattern, only one-to-one communication between client-server	No	No	No	No	Yes
Message Brokers	Scalable solution for many-to-many communication, devices need to know the content formats, built to be light-weight	No	No	No	No	Yes

IV. CONCLUSION

Interoperability issues have been attempted to be solved in broadly three different ways - publishing standards, reference architectures and frameworks, defining protocols and media-type standards and by using abstract interface definition languages and utilizing semantic ontologies. The challenge for interoperability cannot be solved by just a standards specification alone or an application layer protocol or a common domain specific language it must be a combination of all these. The requirements for an interoperable IoT environment is a language that can define the properties and behavior of the devices, automatic code generation from the model for various levels in the application stack, Coupled with protocol(s) for efficient data exchange based on the particular application type and ontologies that can give out the meaning of what the data means. Languages like YANG have a good potential for use in IoT applications as it provides a mechanism for defining the attributes, properties and behaviors which can be used by vendors to augment their changes easily without breaking existing implementation and it supports both configuration and operational interfaces. YANG also has the ability to be converted into the XML/JSON variant without any loss of interpretation. YANG can be integrated with semantic ontologies and Linked Object Vocabularies to enable deeper understanding of the data exchanged between the devices

REFERENCES

[1] "The IPSO Application Framework", http://www.ipso-alliance.org/wp-content/uploads/2016/01/draft-ipso-app-framework-04.pdf, Accessed Aug 2016

[2] "IPSO Smart Objects", http://www.ipso-alliance.org/wp-content/uploads/2016/01/ipso-paper.pdf, Accessed Aug 2016

[3] "Light Weight M2M", http://technical.openmobilealliance.org/Technical/Release_Program/docs/LightweightM2M/V1_0-20160407-C/OMA-TS-LightweightM2M-V1_0-20160407-C.pdf, Accessed Aug 2016

[4] "oneM2m architecture", http://www.etsi.org/deliver/etsi_ts/118100_118199/118101/01.01.00_60/ts_118101v010100p.pdf. Accessed Aug 2016

[5] "oneM2M Base Ontology", http://www.onem2m.org/ontology/Base_Ontology/oneM2M_Base_Ontology.owl, Accessed Aug 2016

[6] "AllJoyn Framework", https://allseenalliance.org/framework/documentation/learn/core/standard-core, Accessed Aug 2016

[7] "IoTvity", https://wiki.iotivity.org/, Accessed Aug 2016

[8] "IoT-A ARM", http://www.iot-a.eu/public/public-documents/documents-1/1/1/D1.3/at_download/file, Accessed Aug 2016

[9] "Weave", https://developers.google.com/weave/, Accessed Aug 2016

[10] "Thread", http://threadgroup.org/Portals/0/documents/whitepapers/Thread%20Stack%20Fundamentals_v2_public.pdf, Accessed Aug 2016

[11] "Vorto", http://www.eclipse.org/vorto/documentation/overview/introduction.html , Accessed Aug 2016

[12] "Vorto Repository", http://vorto.eclipse.org/#/details/examples.functionblockmodels/TemperatureSensor/1.0.0 , Accessed Aug 2016

[13] "Ponte", http://www.eclipse.org/ponte/#architecture , Accessed Aug 2016

[14] Elhadi Shakshuki et el., The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014) Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF, Procedia Computer Science, Volume 32, 2014, Pages 215-222, ISSN 1877-0509, http://dx.doi.org/10.1016/j.procs.2014.05.417.

[15] Michael Compton et al., The SSN ontology of the W3C semantic sensor network incubator group, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 17, December 2012, Pages 25-32, ISSN 1570-8268, ttp://dx.doi.org/10.1016/j.websem.2012.05.003

[16] C. A. Henson et al., "SemSOS: Semantic sensor Observation Service," *Collaborative Technologies and Systems, 2009. CTS '09. International Symposium on*, Baltimore, MD, 2009, pp. 44-53. doi: 10.1109/CTS.2009.5067461