An Open Source Platform for the Internet of Things

Michael J Koster
February 11th, 2013

The Internet of Things connect people and the physical world together in a network of sensing, reasoning, and action. It is real-time, context-aware, hyper-local, and operates at a granular scale.

It is the social graph and the physical graph together: people connecting to things, things connecting to things, people connecting to people.

The Internet of Things: connecting people and things together with software.

The Internet of Things is often described as "machine-to-machine communication" (M2M) or simply that everything will be connected. Connected to what? How does your refrigerator know what to say to your bathroom mirror?

It doesn't, really. The information in your refrigerator is only relevant in the context of you, the user, where and when you interact with your "things", and your intentions at the time. Those connections can only be made by application software.

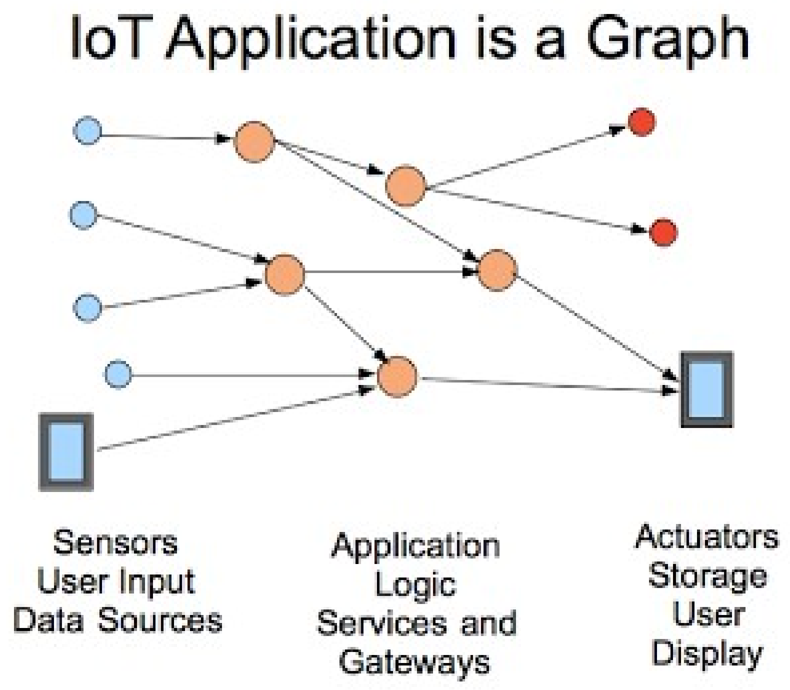
So the Internet of Things is really about connecting people and things together with software.

For people, it may mean greater DIY customization of their experience, through connected awareness, automation, telepresence, and in general a more "hands free" interaction with the physical systems of everyday life.

For business and enterprise, it means exploiting the network effect of connecting physical objects and features together with software. This will produce results like the conversion of capital assets to managed resources, with an expected multiplicative effect on efficiency and resource utilization.

IoT Applications are user-directed graphs

For the purpose of this discussion, we can to think about IoT applications as consisting of sensor and actuator end points, user device end points, and application software that connects the endpoints in the form of a directed graph.

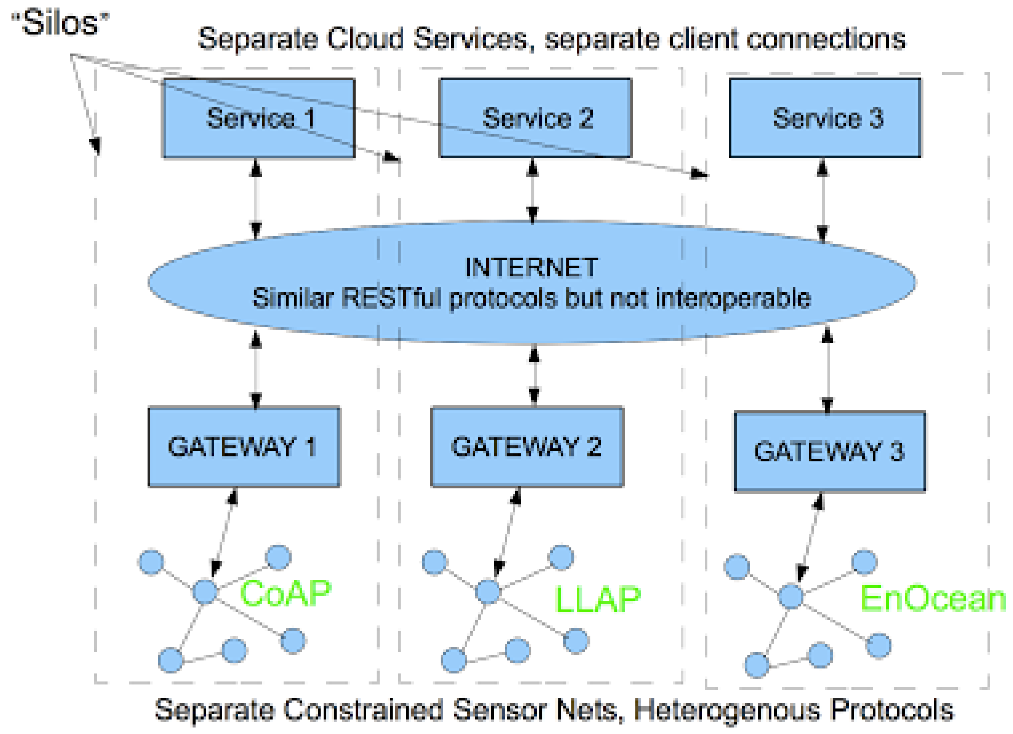


The application software may exist on more than one server, as internet applications do today, but the important distinction is that these connections are made based on user context and intentions rather than javascript programmers, and that the graph may be persistent and dynamically updated per user needs.

Once we understand that the goal is to connect people and things together with software, we can look at ways of using network and data technology we have today, and that which we might invent, to solve the problem.

IoT is Deployed in Silos today

The Internet of Things is being built today, but in a highly vertical fashion. Many companies are emerging with a top-to-bottom technology stack including sensors, actuators, gateways, cloud services, and phone apps. These are typically not interoperable with any other vendor's system at any level (sensor, gateway, service, or app). They all use the internet in similar but incompatible ways.


"Silos"
Separate Cloud Services, separate client connections
Service 1
Service 2
Service 3
INTERNET
Similar RESTful protocols but not interoperable
GATEWAY 1
GATEWAY 2
GATEWAY 3
CoAP
LLAP
EnOcean
Separate Constrained Sensor Nets, Heterogenous Protocols

There are many problems with this situation, which I dive into in my other articles: users vulnerable to single vendor, lack of robustness in solutions, duplication of core platform services, developer code locked to a vendor, and slow industry progress.

This is to be expected early on in an emerging area like the IoT, as many different solutions are being tried. This has been going on for several years now in the IoT, and a loose infrastructure has emerged based on a common set of patterns and data models. While the various systems are not seamlessly interoperable at any level, the common patterns that are emerging suggest that interoperability would be a practical goal.

WSN Interoperability

Wireless Sensor Networks are key technology for the IoT, so much that WSN was probably the number one issue in the early deployment. There are many competing standards: Zigbee, Z-Wave, EnOcean, KNX, XRF, IEEE 802.15.4, Bluetooth, WiFi, RFID, RFM12B...

WSN diversity is a good thing: the differences between the systems are important, such as power, range, data rate, packet size, and there is no one-size-fits-all nor is there likely to be. Some systems support ultra low power energy harvesting like a temperature sensor that runs from a fingernail-sized PV cell and works 100 hours in the dark, vs. richness of protocol such as a complex appliance that needs real time monitoring.

As a guess, there may be 3-4 most common WSN technologies e.g. BT4 moving to 802.15.4 (body area, fitness), Enocean (energy harvesting home and bnuilding automation), WiFi Direct (powered appliances), RFID (sensor tags and keys), 802.15.22 (TVWS) for metro/wide area sensor networking, and these will change and evolve over time.

The new IPV6 protocol for the internet has a set of low power constrained network protocols that allow WSN connection to the internet through a simple router and proxy. These protocols are forward-thinking toward the Internet of Things, and will likely take over more of the WSN domains mentioned above, as Moore's law pushes greater logic complexity into ultra-low power circuits.

Service interoperability

Connected things connect through WSN gateways and routers to Internet services that fulfill the application logic for the user. Today, for the most part, each vendor provides a cloud service for the devices they sell, e.g. Twine. There are also cloud services that allow any connection, providing an API for anyone to connect, for the purpose of integrating multiple devices.

There is the beginning of an ecosystem here, where some devices are being built to use existing services, e.g. Good Night Lamp uses Cosm as their cloud service. Other services that allow open API connectivity include Thingworx and Digi Device Cloud. These services all use very similar RESTful APIs to JSON and XML objects, but have different underlying data models. As a result, sensors and gateways must be programmed for each service they need to interact with.

Proprietary Cloud Service + Open Client

Many IoT services today are based providing easy access to the devices and gateway, with open source client code and reference hardware designs, selling hardware on thin margins, and kickstarter campaigns. There is typically a proprietary cloud service with a proprietary or ad-hoc API from the device or gateway to the service, and a structured API to the service offering "cooked" data.

These systems contain a highly visible open source component, but much of the functionality comes from the cloud service. If a user wishes to use the open source part of the system with another service, the APIs will need to be adapted on either the device/gateway end or service end, or both. It's not exactly a lock-in, but there is a fairly steep barrier to user choice.

This also leaves users vulnerable to outages of a single provider. Much better and more robust would be an ability to configure more than one service provider in parallel in an application graph, for a measure of robustness in the face of service outages. Even more, it should be possible to run user application code in IoT gateways, local user-owned servers, or user-managed personal cloud services. Today's infrastructure and business models are at odds with this level of robustness for users.

User Agency

Another issue that goes even beyond the IoT is that of user identity and user control of resources. While many are imagining the next Facebook of the IoT or the next Twitter of the IoT, these systems are examples of users giving up control of their data, their experience, and often their very identity.

The very nature of the Internet of Things requires a new paradigm based on user agency. Users need to be enabled to customize and define their own IoT applications. We want to give users full control of their identity, their resources, and their experience on the Internet of Things.

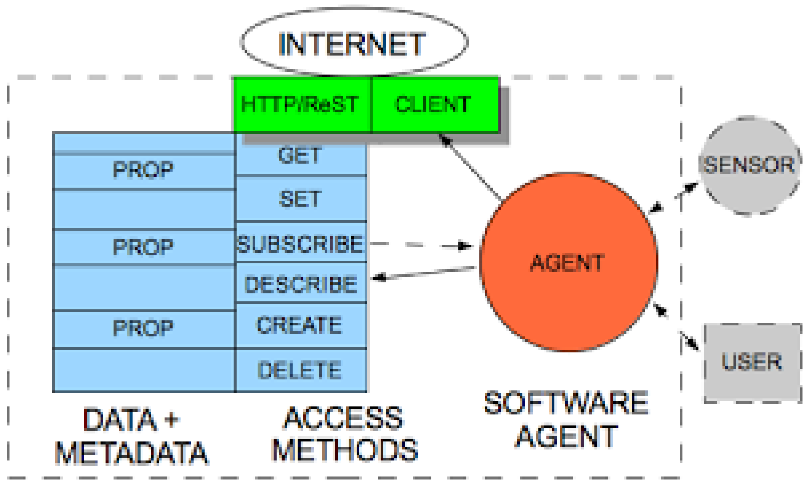
Toward An Internet of Things Platform

A **platform** for the Internet of Things need to broadly address the needs of users, developers, and vendors. The features described here are a subset of the entire platform, but those proposed to address the above needs and requirements.

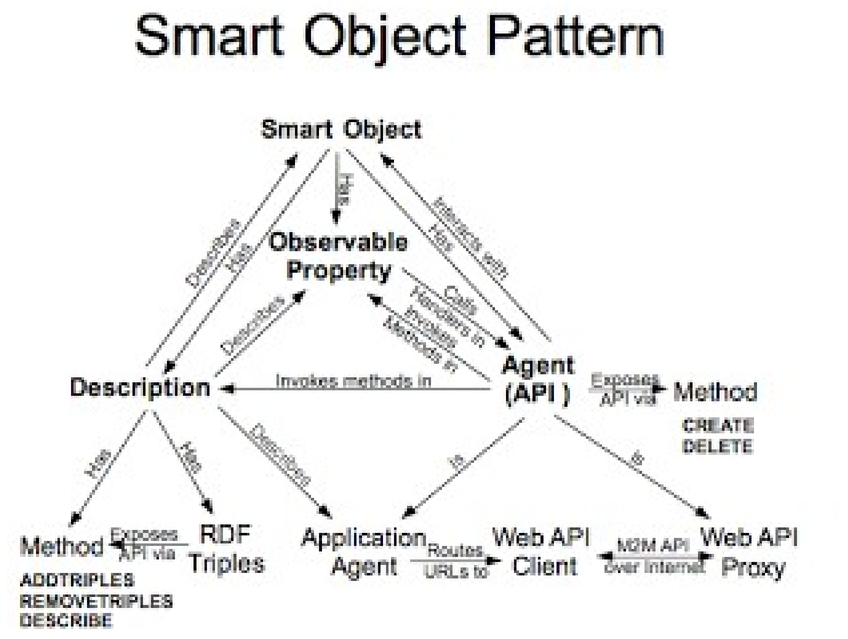
Here are what we propose as base features for an open source Internet of Things platform that promotes interoperability between sensor networks, gateways, services, and user devices.The individual elements have been developed, and are discussed in more detail, in earlier articles of this series.

The Smart Object API defines a data model for broad interoperability

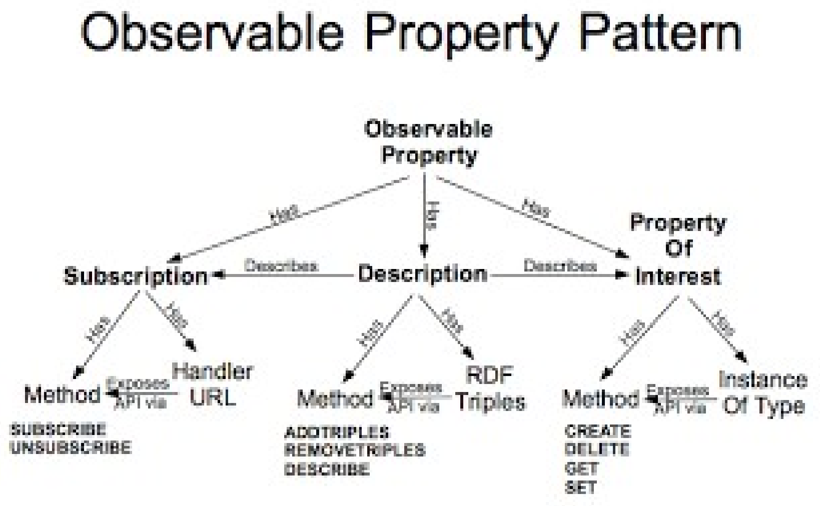
The Smart Object API is a RESTful API to an encapsulation of resources at a node in an application graph. A Smart Object contains IoT data, metadata, and software agent code (application software event handler) resources.



The Smart Object API is also a Semantic Web Linked-Data application that provides integrated, semantic web based, resource discovery and linkage. Semantic Web standards are also used to structure the resources inside Smart Objects. This allows all of the software to be driven by graphs rather than pre-programmed state machines. New APIs can easily be added without breaking any existing code.



The resources presented by the API include RDF descriptions in various web formats, Agent resources to manage application software handlers, and Observable properties, which carry data and data streams.



Observable properties are also structured resources, with their own RDF descriptions and Subscription (also called Observer) resources to enable changes in the observable property to be pushed to URI endpoints in the application graph. Observers may use a number of protocols, for example, Websockets, HTTP POST, PubSub, MQTT, XMPP depending on the application.

This enables a distributed data flow model using software agents + event handlers to implement application logic state machines, filters, triggers, etc. based on changes in data or context.

Use of an abstract observable property enables flexible data typing based on semantic constructors and JSON, XML data objects.

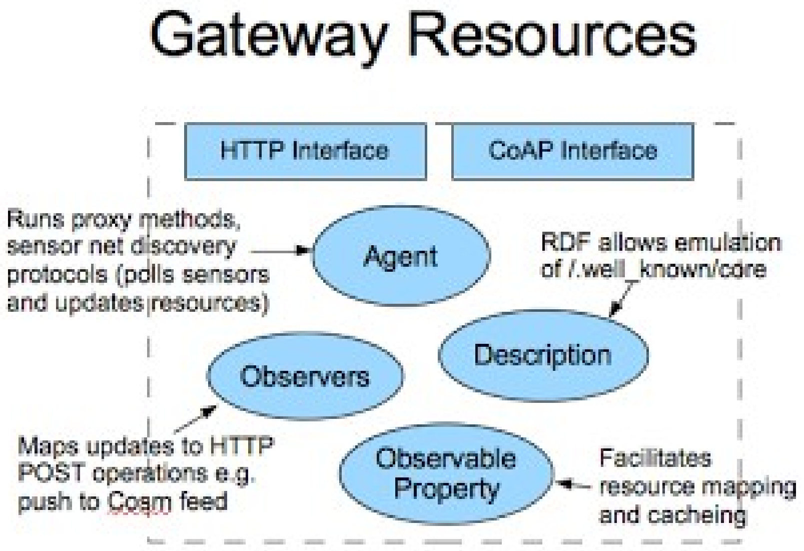
Programming languages

The semantic + event model can be integrated into general purpose languages through libraries. High level methods for semantic linkage and handler connection will allow developers to build graphs and define event processing rules, filters, and triggers.

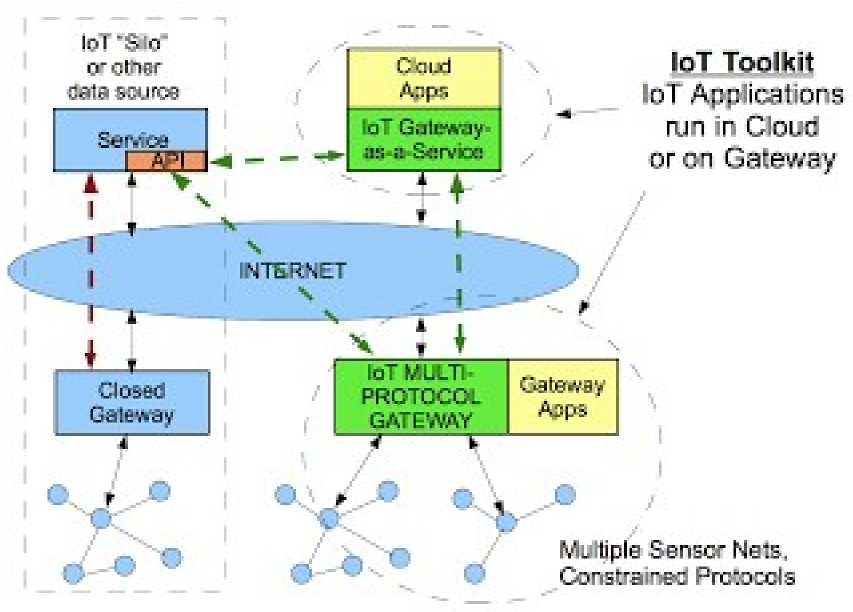
The RESTful internal resource model enables APIs to easily and automatically be built for any language. Special purpose languages may also be developed, sporting embedded Semantic Event processing models and control structures.

Universal Gateways integrate local sensor net resources

Smart Object gateways will host all smart object resources, performing Semantic integration and tagging of WSN resources, WSN-to-internet proxy, adding encryption and security to constrained-protocol WSNs, caching and buffering of accesses, and function as an always-on resource for data collection from energy harvesting sensors and other chirpy data sources. Software agents may be locally executed in the gateway.


Gateway Resources
HTTP Interface
CoAP Interface
Runs proxy methods, sensor net discovery protocols (polls sensors and updates resources)
Agent
RDF allows emulation of /.well_known/core
Description
Observers
Maps updates to HTTP POST operations e.g. push to Cosm feed
Observable Property
Facilitates resource mapping and cacheing

The Smart Object API resources provide a semantic backplane for multiple WSNs and multiple services, enabling integration of IoT resources from top to bottom of the stack, and even accommodating IoT vendor "silos" that only provide high level cooked APIs from their cloud services.


IoT "Silo" or other data source
Service
API
Cloud Apps
IoT Gateway-as-a-Service
IoT Toolkit
IoT Applications run in Cloud or on Gateway
INTERNET
Closed Gateway
IoT MULTI-PROTOCOL GATEWAY
Gateway Apps
Multiple Sensor Nets, Constrained Protocols

Smart Object gateways and services talk to each other using the same API, which is strongly consistent with the low level object programming model provided to the library. This enables application code to run anywhere, in any cloud service or gateway, providing robust service availability in the face of network and service outages.

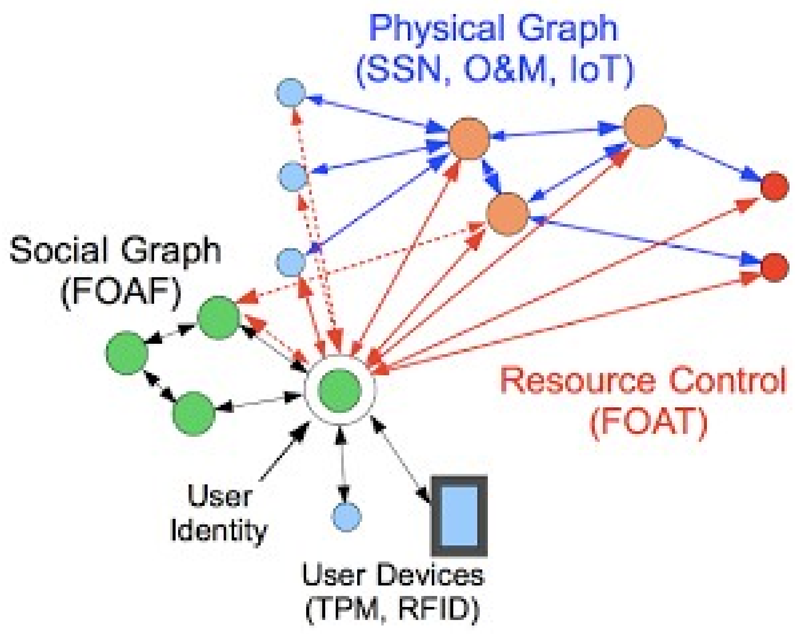
Native applications using the Smart Object API will have access to load balancing, failover, redundant persistence, and strong object storage models.

Personal Object affords user control of identity, resources, and experience

Personal identity and agency are of special importance for the Internet of Things. We have decided to build in a strong model for user agency into the system from the beginning. This allows users to control their identity, access to their resources based on their strong identity, and also to control their experience on the Internet of Things.

The Personal Object is a special case of a Smart Object that represents a user's identity and preferences on the IoT. A Personal Object can be constructed using the Friend Of A Friend ontology, which provides for creation of a social graph that connects people to other people that they know and who can verify their identity. A subgraph of this socal graph can then be used as a strong identifier for a person.

To provide for IoT interaction we propose to connect the social graph to the physical graph using concepts from a new Friend Of A Thing (FOAT) ontology. These concepts and relationships will define people's relationships to the physical resources they control and that also help establish their graph-based identity.



This method allows for the definition of very granular resource access control based on strong identity. Resources are shared with people in one's social graph by making connections to other nodes which can be used to verify both the identity and the capability of resource sharers. These links allow very granular resource control of each resource or property according to specific capabilities. At a minimum there will be API level capability keys, extensible to more granular endpoints e.g. graph objects but not subjects.

Personal Cloud service gives users a vendor-neutral identity and point of control

People today are basing more and more of their digital existence on cloud services such as Facebook, Google+, and Twitter. These vendors often make us the product and hold our data hostage. Even with the best of the platforms, we are ever more vulnerable to channels and storage and software that others control.

Going forward, it's difficult to imagine an IoT experience where service vendors like Facebook set the constraints and define the parameters of how you turn your lights on and off.

It's also difficult to see how multiple competing vendors could provide truly compatible services that allow users to freely move their devices from one to the other.

We will therefore provide in the platform the means for users to create personal cloud services. This will enable vendor-independent storage and service locations to host user's Personal Object and Smart Objects.

Personal cloud services can provide redundancy and persistence, and can be a user's IoT home location on the internet.

Personal cloud services can also function as Gateway-as-a-Service to integrate other cloud APIs into the common data model.

Perhaps most importantly, the creation of a personal cloud system allows users to root all of their social and physical graphs at their own service, analogous to a personal computer, providing a user locus of control. One's facebook graph can be rooted in and synced to one's Personal Object running in their personal cloud service.