Coffee Maker demo page for the work presented in "IoT-DDL Device Description Language for the 'T' in IoT" paper, under-submission

Presented by Ahmed Khaled and Prof. Sumi Helal

IoT-DDL and Atlas-Thing architecture

IoT Device Description Language (IoT-DDL) is a machine- and human-readable XML-based descriptive approach for the thing in smart space. IoT-DDL explicitly tools the thing to self-discover its own capabilities, resources, entities, services and cloud-based thing accessories. Resources are the components that shape the operating system services a thing needs to be part of IoT (e.g., network module, memory unit). Each resource is shaped through a set of properties that configure such operating services. Moreover, thing entities are the different physical devices, software functions and hybrid devices that can be attached to, built-in or embedded inside the thing. Each entity provides set of services to smart space through a set of well-defined interfaces (APIs). Furthermore, a thing can have one or more external accessory named attachment. Thing-Attachment is a cloud-based expansion of the thing that provides further representations and services (e.g., Log-server, Database, Dashboard). Thing mates of a thing in smart space include cloud platforms, edges, and other things (a nearby thing in the same smart space or remote thing from a remote smart space). A thing in smart space engages with thing mates in IoT ecosystem through a set of information-based interactions and action-based interactions. Information-based interactions (referred to as tweets) enable the thing to announce its identity, capabilities, and APIs to other thing mates. Tweet is a way through which a thing can describe what it is, what it does, and what it knows to the other thing mates. Action-based interactions include management commands, lifetime updates and configurations from authorized parties as well as the applications that target thing's capabilities and services.

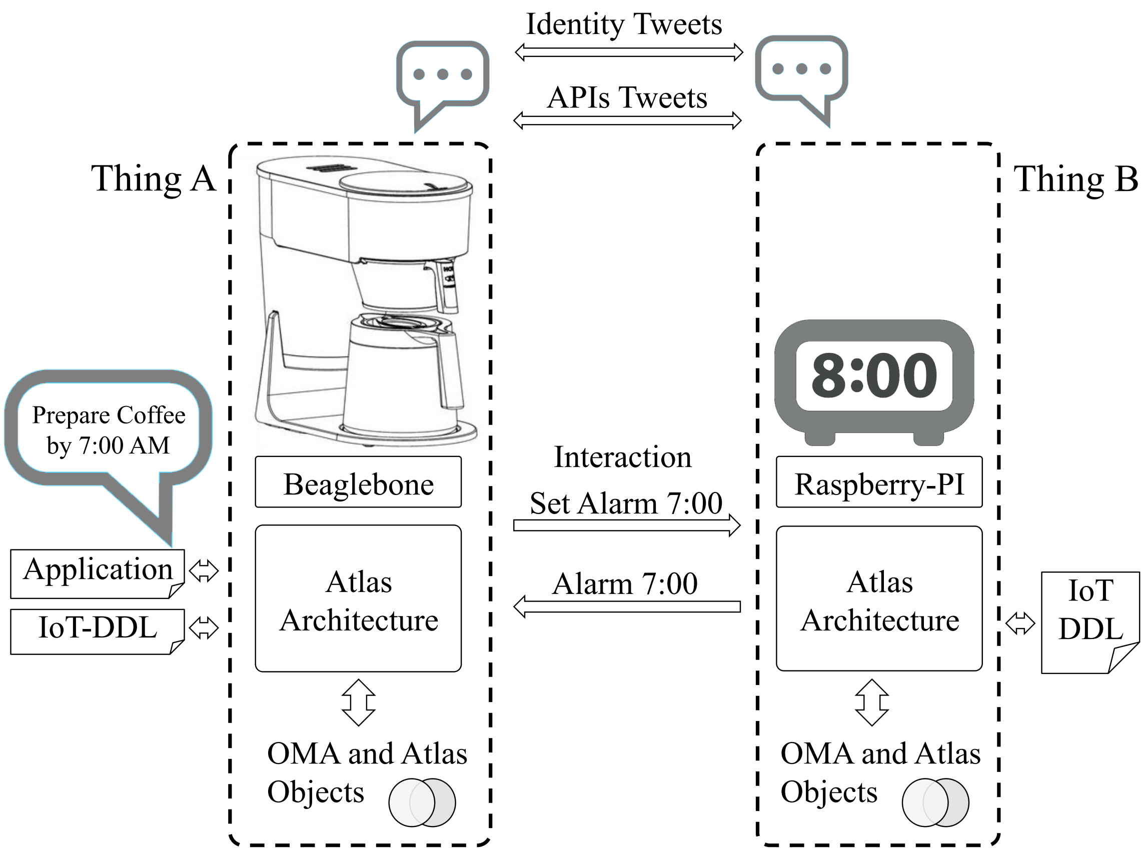
Taking a smart phone as an example of a thing in smart space is equipped with an internal memory and WiFi network module as two essential resources required for the smart phone to be part of the smart space. The smart phone as the same time contains set of sensors (e.g., proximity, accelerometer) as embedded entities that can offer services to smart space. The phone can also be connected to a cloud-attachment for lifetime updates for the operating system provided by the vendor. The three argued requirements and the proposed structure for the thing shape the specifications required to describe thing in smart space as well as highlights the design aspects of the architecture that can fully utilizes such specifications.

The IoT-DDL is proposed within a lightweight architecture named Atlas-Thing architecture that fully utilizes the specifications of IoT-DDL. Atlas-Thing architecture takes advantage of the thing's operating system services to provide new layers and functionalities that introduce novel capabilities we believe a thing must have to engage in ad-hoc interactions and interconnections as well as IoT scenarios and applications. The architecture enables the thing to self-discover its resources, attachments, components and services from the uploaded IoT-DDL. The thing can then formulate APIs of its offered services. The architecture enables information-based interactions and action-based interactions to take place between the thing and thing mates. The architecture also takes the advantage of lightweight device management standards (Open Mobile Alliance lightweight M2M (OMA-LwM2M)), object modeling standard (IP Smart Object (IPSO)), IoT communication standards (Constrained Application Protocol (CoAP) and Message Queuing Telemetry Transport (MQTT)) and security protocols (Advanced Encryption Standard (AES)) to enable thing management and configuration with the least human intervention, and empower secure ad-hoc interactions between thing and thing mates. Such interactions will ultimately fuel powerful IoT programming models, which are currently missing.

Promising scenario for IoT: Prepare morning coffee automatically!

In this demo we involved two different things in a smart space, Raspberry PI model-B sensor platform running Raspbian operating system and Beaglebone-Black sensor platform running Angstrom operating system. On one side, the Beaglebone-Black thing is connected to an external water boiler as an attached hardware entity. On the other side, the Raspberry PI thing offers an alarm clock service as a built-in software entity. T he IoT-DDL configuration files are developed and deployed on both things, point the identity of each thing, their inner entities beside the different resources and services. The Beaglebone-Black thing offers preparing coffee for specific time duration and switching the boiler off services. While the Raspberry PI thing offers to set and to clear alarm clock service. Powering on the two things, the architecture in each starts parsing the different sections and subsections of the uploaded IoT-DDL. Each thing now identifies itself, discover the different services and functions it can offer to space, and then start generating APIs. Each thing starts looking for space-mates by broadcasting thing identity tweets, entity identity tweets and generated APIs' tweets. Upon receiving other's tweets, each thing keeps a record for future use. The prototype starts by assuming an IoT application resides the Beaglebone-Black thing that requires turning on the boiler for a while when the alarm triggers. At the same time, both things register themselves as OMA clients that can connect to the server and register their OMA standard objects and Atlas objects. On the server side, an authorized user can list the different connected clients, read the registered objects and updates the different attributes of them.

Below is a sketch for the demo that highlights the main components of scenario, and a video for the demo can be accessed through this link (https://youtu.be/up2IUNqdv0o):



The IoT-DDL xml files for both Beaglebone Black and Raspberry-PI things can be accessed through the links below:

- Beaglebone-Black IoT-DDL file: https://github.com/AEEldin/IoTDDL_CoffeeMakerDemo/blob/master/BBB_IoTDDL.xml
- Raspberry-PI IoT-DDL file: https://github.com/AEEldin/IoTDDL_CoffeeMakerDemo/blob/master/RPI_IoTDDL.xml

Mobile & Pervasive Computing Laboratory