SemIoTic: Interoperability In Smart Spaces
Rayan Al-Atab; Professor Sharad Mehrotra; Dr. Georgios Bouloukakis; Dr. Roberto Yus,
1University of California, Berkeley, 2University of California, Irvine

Abstract
This project proposes a middleware framework for IoT smart spaces that provides a semantic domain relevant view of the smart space. It allows developers to implement applications that interact with the smart space on a high level, hiding the complexity of having to deal with low level information generated by sensors and actuators. However, these IoT devices are heterogeneous with different operating and hardware characteristics. The lack of a single standard interaction protocol makes it more challenging to obtain and exchange information in the system. To solve this interoperability issue, we propose the Data eXchange (DeX) connector model, which comprehensively abstracts communication middleware protocols.

Introduction
SemIoTic facilitates the development of IoT smart space applications by enabling users to express their requests/commands on a high level.

It solves two main challenges:
1- Interoperability: Supports high level interactions with device wrappers to extract information from sensors and actuators regardless of the protocol they support.
2- Reusability: Presents a metamodel that is based on SSN/SOSA ontology used for describing static and dynamic sensors and their observation. It supports algorithms that translates high level user actions into appropriate IoT device actions.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Interactions</th>
<th>Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoAP</td>
<td>request/response; streaming</td>
<td>reduced power requirements</td>
</tr>
<tr>
<td>REST</td>
<td>request/response</td>
<td>greater variety of data formats</td>
</tr>
<tr>
<td>MQTT</td>
<td>streaming</td>
<td>fast transaction rates</td>
</tr>
<tr>
<td>WebSockets</td>
<td>streaming</td>
<td>reduces unnecessary network traffic and latency</td>
</tr>
</tbody>
</table>

Table 1. Protocols used in the implementation of the DeX API

Design

DeX Client - Server Publisher - Subscriber Data Streaming
post send publish push
get receive listen accept
scope operation filter stream_id
message item event data

Table 2. Primitives of core models mapped to DeX primitives

Implementation

Figure 6. DeX direct mediation

Future Work
• Implement streaming interaction in DeX API.
• Continue developing the SemIoTic wrapper.
• Extend the DeX API to support more communication protocols.

Acknowledgements
I would like to thank Professor Sharad for having me in his lab this summer. I would also like to thank my mentors Georgios Bouloukakis and Roberto Yus for their advice, assistance and guidance in completing this project. I would also like to thank Dr. Nalini Venkatasubramanian and Dr. Sharnnia Artis for their unlimited support in helping me become a better researcher. Last but not least, a big thank you to the NSF for their funding and sponsorship.

References