IoT Bridge Taxonomy

By Dave Thaler (dthaler@microsoft.com)

Abstract

Today, many organizations develop and implement different schemas for the same kind of things. For example, there are already many different ways to express the notion of a "light bulb". In this position paper, we assume that there will continue to be heterogeneity in protocols and schemas, at least for the forseeable future, given that IoT is relatively early in the technology lifecycle. This position paper focuses on the question "How can we achieve interoperability when different organizations, companies and individuals develop extensions?", and in particular how to translate, or "bridge", between two endpoints with differing protocols and schemas. We define a simple taxonomy of bridging types involving a protocol translation component (or "protocol bridge") combined with one or more schema translation components (or "schema bridges"), and provide recommendations on future directions.

1 Terminology

In this paper we use the following terms:

- "server": an endpoint exposing information (e.g., a smart object such as a light bulb or TV). Also known as a producer or provider in some protocols.
- "client": an endpoint consuming such information (e.g., an app). Also known as a consumer in some protocols.
- "data model": a concrete schema, i.e., a schema expressed in a particular language/format (e.g., in RAML, JSON Schema, D-Bus introspection XML, etc.).
- "resource": a specific instance of something implementing one or more data models for different types of functionality, exposed by a server.
- "bridge": a device designed to translate from one protocol and/or data model to another, to allow a client of one to communicate with a server of another. The bridge may be physically collocated with the server, or the client, or (more typically) in a separate device such as a hub.
- "schema bridge": a bridge component that can translate data expressed in a given data model to another one that expresses the same information in a different way. The conversion might involve conversion of data types (e.g., integers vs string literals), property names, resource granularity (e.g. one data model vs two more specific data models), or other more complex conversions.
- "protocol bridge": a bridge component that can translate from a given protocol to another given protocol, but does not translate the data model. Thus, often a protocol bridge requires one or more schema bridges due to the two protocols using different data models.

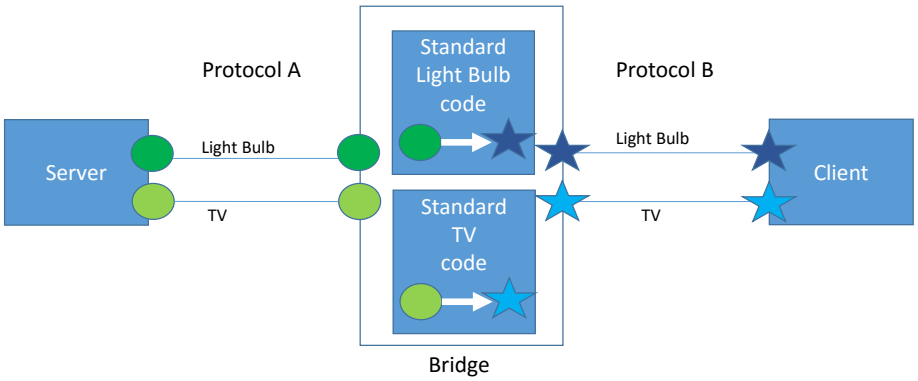
2 Taxonomy

We first define two important axes for any given scenario:

1. Is the server's resource exposing a "standard" data model, or a vendor-specific data model?
2. Is the bridge exposing that resource as a "standard" data model, or a dynamically created data model?

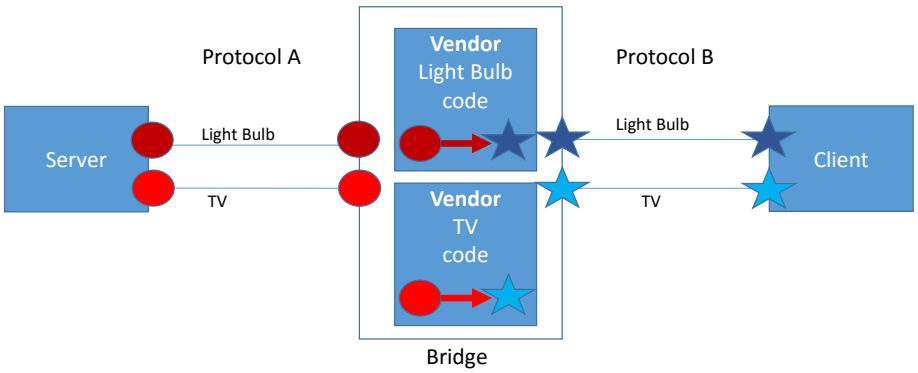
2.1 Bridging standard-to-(other)-standard (S2S)

This scenario typically requires translation code on a per-data model basis, since each standard can express things completely differently. However, since both are standards, such code can be authored by a third party (i.e., someone other than the client and server code owners).



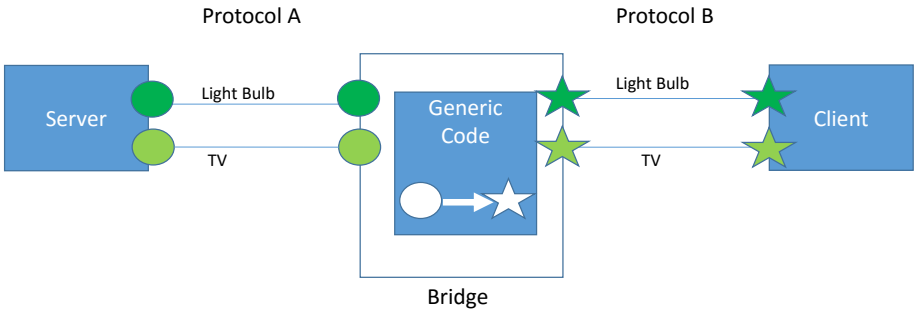
2.2 Bridging proprietary-to-standard (P2S)

This similarly requires bridging code on a per-data model basis. Since the server's data model is not a standard, it often must be authored by the same vendor/coder as the server or by someone who obtains permission from them. Thus the bridge must either be server-vendor-specific itself, or else it must allow vendor-specific extensions to be included or added.



2.3 Bridging any-to-dynamic

In this case, the bridge does not try to translate to a particular schema/data model, but attempts to keep the same data model (whether standard or proprietary) as much as possible and only translate the actual syntax into the language/format used on the other side (e.g., XML to JSON or vice versa), without any domain-specific knowledge of the data model being translated or its semantics. Thus, there is only one dynamic schema bridge needed (at least for a given pair of protocols), and it can be authored by a third party.



The benefit is that the bridge can be used to translate essentially *any* resource, rather than only being able to translate ones it has a priori knowledge of the data model(s) for. The downside is that different servers exposing different data models for the same thing will appear as different sorts of things to the client.

3 Recommendations

In general, we believe that bridges should use specific schema bridges for known data models (which we call "static schema bridges"), and fall back to using a dynamic schema bridge when no specific schema bridge is found for a discovered resource. Furthermore, we believe that static schema bridges are important to convergence (e.g., one client communicating with multiple server implementations), and it is thus important for bridges (that aren't embedded into the server itself) to separate the notion of "protocol bridge" from "(static) schema bridge" and allow extensibility by others. The bridge, or the owner of a bridge, should be able to easily install schema bridges for new and/or proprietary data models.

To ease the job of developing schema bridges and increase semantic interoperability, it is desirable that different data models for the same type of thing (e.g., light bulbs) are as similar as possible for basic functionality. In an ideal world, data models used by different protocols and organizations would express exactly the same information in ways that are algorithmically translatable by a dynamic schema bridge with no domain-specific knowledge. Sharing data models more widely, and having agreements in principle of at least using the same abstract information model, would be very beneficial.

It is also important to keep in mind that the IoT ecosystem is still undergoing significant innovation, especially in terms of new types of resources, and even new variations of existing types of resources. For example, while classic light bulbs historically had on/off, then dimming capability, many now have color settings, and perhaps other bells and whistles. It is important to not preclude innovation and vendor differentiation, in order to grow the IoT ecosystem and allow it to evolve. Thus it is also critically important to not limit use to only standard schemas. Hence the recommendation to support dynamic schema bridges, as well as allowing bridges to acquire new static schema bridges.