Embedded Web Services: Making Sense out of Diverse Sensors

David E. Culler
Gilman Tolle
Arch Rock Corporation

Introduction

How many times have you heard, "I just want to connect a collection of different sensors together and have them JUST WORK"? As providers of sensor related technology, this seemingly naïve statement should give us pause. Why is ease of integration such an increasingly common expectation? And, why is it so hard to achieve in practice? In part, the expectation is natural given the increasingly sophisticated interconnects that are built into our instruments, meters, and gauges, making them a source of physical information, rather than just a measurement device. However, the design space of sensors is so diverse with so many subtle specializations that it very difficult to represent the information that they produce in anything like a plug-and-play fashion, much less to describe how these devices must be utilized in order for that information to be meaningful. Starting from the observation that sensors have become embedded physical information servers, we may draw insights from IT community's long standing efforts to cope with integrating diverse sources of distributed information. Today, every large enterprise, major supply chain, and significant Web portal integrates many different kinds of data sources from physically distributed dynamic processes. After many attempts, the key was to simplify the approach into a Service Oriented Architecture built upon a relatively straightforward foundation of Web Services. In this article, we translate these lessons to Embedded Web Services as a framework for integrating diverse sensor networks.

Three Levels of Embedded Information Integration

The quest for ease of integration for networks of sensors dates back to the 50's with the development of the 4-20 mA "current loop" signaling standard for analog sensors. Of course, while it allows a controller to obtain the current output value from any sensor (by digitizing the voltage across a precision resistor), interpreting that value as a meaningful process variable still requires out of band information about the specific transducer, its calibration, and how to correctly operate and control the device.

In general, integrating diverse sources of information takes place at three levels. The bottom is the communication media that allows information and control actions to be exchanged. Given such an interconnection, the communicating participants must agree on how information is represented. This is variously called the format, data model, or object model. Finally, for such information exchanges to be useful, the participants must have a way to discover what behaviors other devices are capable of performing and how to cause them to take particular actions, i.e., service discovery.

Figure 1. Integrating Diverse Information Sources

The quest entered its digital chapter in the mid 80's with the HART (Highway Addressable Remote Transducer) standard. HART borrowed a page from the telephony book to provide a 1200 baud serial communications bus over the analog 4-20 mA wiring. It fused all three levels in defining a fixed set of transactions (access process variables, set configuration parameters, and perform calibration) with fixed data representation over a particular link. BACnet followed in the early 90s with a much broader integration goal. It was defined over several standard serial-line and LAN communication links (RS232, RS485, ARCNet, Ethernet, LONTalk), specified 23 standard object types, and represented a device as a collection of objects supporting application level services (for data sharing, alarm and event management, trending, scheduling, and remote device and network management). The introduction of BACnet/IP allowed adoption of newer IP-based links, but also allowed a BACnet "network" to be constructed as an application specific 'virtual network' within an IP infrastructure. The object and data representation of all of these is defined in Abstract Syntax Notation (ASN.1) - used widely for management of IP networking devices. To represent intended behaviors, specific device and object definitions are collected into "application profiles". Common Industrial Protocol (CIP)

followed in the late 90's with a more complete object model and a recognition that communication between objects can be defined independently from the physical links that provide transport, starting with DeviceNet and moving to UDP/IP. Devices are physical instances of a class and containing a collection of objects. However, all CIP classes, object types, and data types have fixed 8-bit encodings pre-specified in human-readable documents.

The opportunity for ease of integration in physical terms was dramatically enhanced by the emergence IEEE 802.15.4 making connection wireless and operating at very low power. Multihop mesh routing is used for robustness and range even in harsh, cluttered environments. Zigbee seeks to provide integration at the application level over this particular link through defining the binary representation of a host of object types with application profiles to capture the informal semantics. Meanwhile several groups have begun to incorporate this new link, including Wireless HART and SP100.11a.

Today, most sensors are small chips or circuit elements connected directly to powerful, yet inexpensive, microcontrollers with sophisticated communication links (CMOS radios or wired interfaces) attached. So with intelligence and communication at every device, it should be easy to integrate them into rich networks.

Web Services

The complexity of defining and utilizing distributed information sources in binary form is not unique to sensors. In this same time period we saw the automation of order entry, inventory management, process flow, and logistics. But despite IP-based networks connecting all the parts together, the information representation and behaviors of the customer's system were always different from that of the vendor and all the databases organized the information differently. The IT community introduced its own collection of complex distributed object systems, including Sun RPC, CORBA, and Microsoft's DCOM. Each had its own abstract object representation model, interface inquiry protocols, and service discovery mechanisms.

All this changed with the Web. Today we routinely experience the sophisticated integration of many different kinds of information sources and processes. Even a single shopping site integrates inventory and availability from many different vendors, order processing, shipping, real time feeds, shipping, news, alerts, and customized analytics.

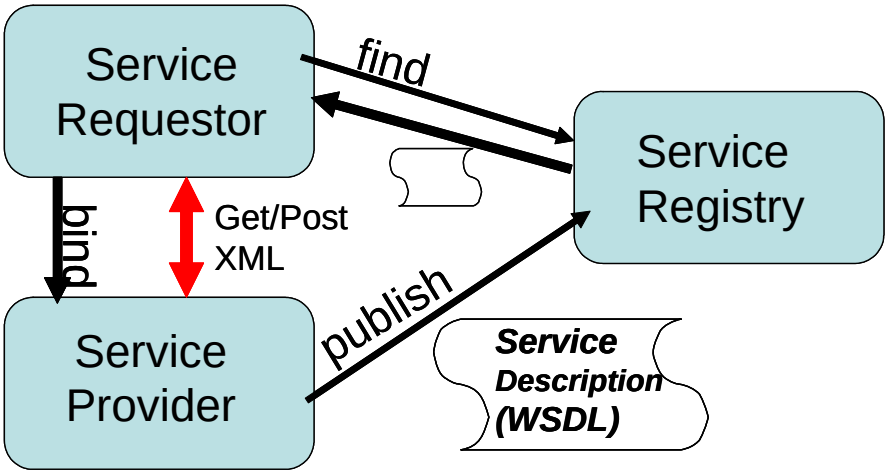

Figure 2. Web Service discovery, bind, and utilize

The integration breakthrough came about through two primary factors. The first factor was a radical simplification at each of our levels. Communication is simplified to transferring sequences of characters between named endpoints on hosts. Information representation is simplified to recognizing nested XML tagged sections, delimited by <tag> … </tag> pairs in the sequence, coupled to a machine-readable schema that is also in XML. The set of behaviors is simplified to two: GET and POST a sequence from or to a named endpoint. The second factor was consistent application of a Service Oriented Architecture (SoA) in which applications are built as a protocol and implementation independent composition of services.

In Web Services, each information source or computational element describes itself in a Web Services Description Language (WSDL) file. As illustrated in Figure 2, a requestor of the service first obtains its WSDL, either directly from the service, as in device discovery, or indirectly from a repository. The WSDL is a complete machine readable description of the service, including how information is represented and what behaviors are provided. It allows the requestor to bind to the provider of the service and establish any necessary translations in a fully automated fashion. Then, data sharing and behavior invocation are conducted efficiently through the established service interface. XML and WSDL provide an automated framework for defining objects and operations. The application domain determines the specific objects and operations that are included in the services.

So the ease of integration question for wireless sensor networks can be rephrased as “Can we have the best of these two worlds?” Can we retain compact binary representations and finely crafted application profiles that we demand for embedded devices and yet the automated service discovery, composition, and ability to deal with diverse information sources that we expect in enterprise applications?

At the communications level, the binding step between the consumer and provider of a service provides the mechanism to deal with a diversity of links and protocols. The higher powered devices are likely to be on IP-based links, such as Ethernet or WiFi, and the binding is certainly simpler if the embedded sensor links also support IP. However, for non-IP embedded links the gateways can participate in the binding step. The only requirement is that it be possible to name the service endpoint in a consistent manner. In IP this is simply the IP address and port number. It is a URL at the HTTP level.

At the information representation level, the set of binary objects used in the exchange (e.g., bits, bytes, and integers) are mapped to a set of XML tags, as are the set of composition rules (e.g., struct and array). This means that for any binary object that is to be communicated at the embedded device there is a canonical XML representation that is independent of that particular device and a mapping between the two. Embedded devices do not have to support general XML, they only deal with a compact subset. The WSDL provides enough information for gateways, proxies, or applications to perform the translation.

The application profile is reflected in the objects and methods described in the WSDL for the device. In fact, WSDLs can be defined for the existing industrial standards, so the WSDL serves as a specification for the device. Or, WSDLs can be generated from the implementation of the service on the device. Then the generated WSDL must match the standard, up to vendor extensions.

These concepts are illustrated in Figure 3 for Arch Rock’s Primer Pack / IP.

- The communication layer uses IPv6 naming and routing (with 6LoWPAN standard compression) over IEEE 802.15.4. This connects to surrounding corporate networks through routers or gateways.
- Data and objects in the embedded devices and network are represented as a compressed binary form of restricted XML. The role of HART process variables, BACnet properties, and CIP attributes is provided by typed *attributes*. The particular attributes, such as sample rate, threshold, and current value, are specific to the application and form the core of the application profile.
- GET and SET methods are defined on attributes, providing basic data acquisition, device and service configuration, and management. Events,

alarms, or scheduled reporting are POSTed from the embedded service provider to the consumer.

- More complex aspects of the application profile, such calibration, local data processing, management, and diagnosis, are provided by commands that are POSTed to the device with an associated set of typed argument and result parameters.

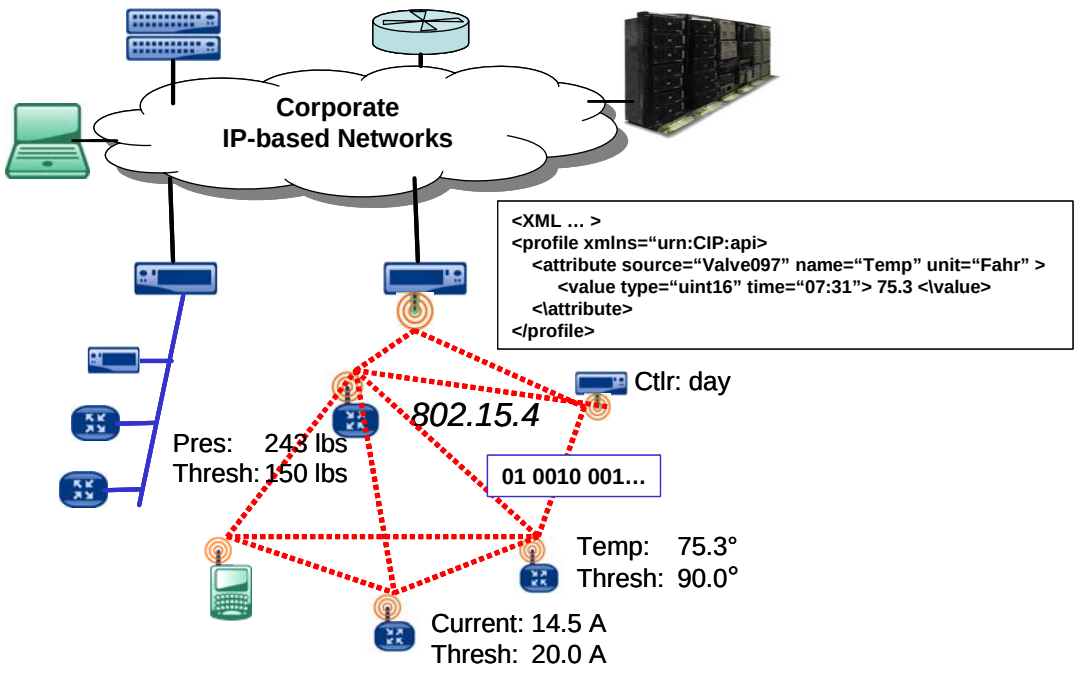

Figure 3. Embedded Web Services

All of these are communicated in compact binary form across the mesh routed low power wireless network, but when they cross a gateway or proxy to a more capable network they are translated to their full, self-describing XML form.

The Arch Rock offering goes one step further and provides application development support for embedded web services as well. The variable that holds the attribute in the embedded code is annotated as so compilation tools can generate the corresponding sections of the WSDL, the XML representation, and all the translations. Similar support is provided for commands and events.

Going Forward

The increasing intelligence, ubiquity and diversity of sensors have made paramount the need for an interoperable, robust means of communicating with sensors and interpreting their output. Although many industrial standards efforts have

approached this problem with complex binary object models working up from the devices, it is likely that the path to ease of information may follow that of web services, which employ a fairly simple model from the top down. As both these worlds seek to easily integrate diverse information source in application specific manners, it is likely they there will grow more similar as we go forward.