Poster Abstract: sMAP – Simple Monitoring and Actuation Profile

Xiaofan Jiang, Stephen Dawson-Haggerty, and David Culler
Computer Science Division
University of California, Berkeley
{fxjiang, stevedh, culler}@cs.berkeley.edu

ABSTRACT

We present the architecture, specification, and implementations of a simple monitoring and action profile (sMAP), optimized for sensors, meters, and actuators in building environments. Our architecture is built on HTTP/REST and uses JSON as the object format for interoperability. We implement sMAP on a variety of resource monitors and actuators inside a commercial building, including mote-based wireless sensors and meters running IPv6/6LowPAN, Modbus based panel meters, and external data sources. We show that sMAP is widely implementable and efficient, and our API and schema definitions are expressive and concise. We demonstrate that our architecture is well suited for resource constrained devices using compressed JSON and proxies.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication NetworksNetwork Architecture and Design

General Terms

Design, Experimentation, Management

Keywords

Energy, Building, Wireless, Sensor Networks

1. INTRODUCTION

The building has long been under-appreciated as a sensor network, when the fact is that modern commercial buildings contain thousands of sensing and actuation points. Current state-of-the-art building systems, however, hamper rather then help the innovation that is necessary to reduce building energy consumption: the systems typically use older technologies (e.g. Modbus over RS-485) and often are accessible only though obscure interfaces. In this work, we make the first steps towards bringing a flexible distributed architecture to the building by describing a simple RESTful API for accessing building sensors and actuators. Using this interface, we are able to represent an interesting set of building sensors including environmental sensors, plug-load and panel electricity meters, and HVAC systems.

This profile is not developed for its own sake; this is a small piece of a system which, when complete, will include a data storage service, building modeling, and security policy that will allow modeling, supervisory control, and personalized energy feedback applications. This work is also relevant to work currently in progress in academia and industry, such as in the Zigbee Alliance and Google [4, 3, 2], which will be used in a myriad of energy-consuming devices. We show that sMAP is a viable solution for typical building sensors as well as embedded devices running on top of 802.15.4, 6LowPAN, HYDRO (a RPL predecessor), and HTTP/TCP.

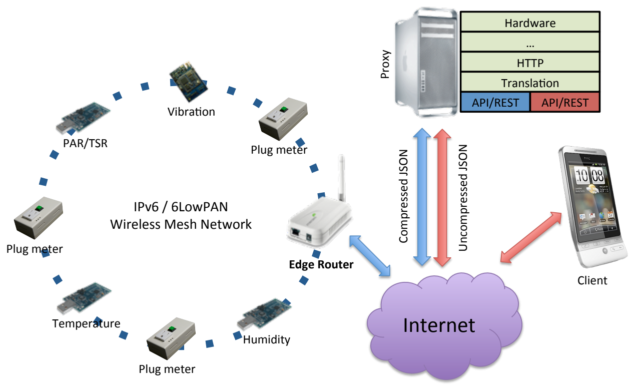


Figure 1: Resource-constrained devices such as motes implement sMAP on top of embedded IP stacks. JSON objects are compressed and decompressed transparently between IP end-points through the use of proxies.

2. ARCHITECTURE

Our architecture is built on HTTP/REST because it is simple and widely compatible. We use JSON as the object interchange format because it is compact, allows validation, and can be easily converted to other representations (e.g. XML, binary). We are able to uniformly represent a number of types of sensors, meters, and actuators, and provide consistent interfaces to access their data, configuration, and historical profiles.

In this architecture, devices with limited resources are supported using proxies, as shown in Figure 1. Each mote implements sMAP on an IPv6/6LowPAN stack [1], and exports this resource via its URL. An edge router connects the wireless mesh network to the Internet. When a client tries to query any of these devices, the HTTP request goes through the proxy, which compresses the JSON data object (if supplied), and relays is to the mote. Subsequently, the response

```
/                     # list resource under URI root [GET]
  /data               # list sense points under resource data [GET]
    / [sense_point]   # select a sense points [GET]
      /meter          # meters provide this service [GET]
        /[channel]    # a particular channel [GET]
          /reading    # meter reading [GET]
          /format     # calibration and units [GET/POST]
          /parameter  # sampling parameter [GET/POST]
          /profile    # history of readings [GET]
          /report     # create and query periodic reports [GET/POST]
```

```
POST requests supply JSON objects as arguments:
        POST: http://meter1.cs.berkeley.edu/report
        { "ReportResource" : "/data/325/meter/*/reading",
          "ReportDeliveryLocation" :
                  "http://webs.cs.berkeley.edu/receivereports.php",
          "Period" : 60, "Minimum" : 50, "Maximum" : 100 }
```

Figure 2: Top figure shows a portion of the API for the meter resource. Bottom figure shows creating a periodic request via HTTP_POST.

instance

```
{"UnitofMeasure" : "kW",
 "Multiplier" : 1,
 "Divisor" : 1,
 "UnitofTime": "second",
 "MeterType" : "electric" }
```

schema

```
 {"UnitofMeasure" : {"type" : "string",
    "options":[
      {"value":"kW", "label":"kW/kWh"},
      {"value":"lph", "label":"Liters per Hour"},...]},
  "Multiplier" : {"type" : "integer",
    "optional" : "true"},
  "Divisor" : {"type" : "integer", "optional" : "true"},
  "UnitofTime" : {"type" : "string",
    "options": [
      {"value":"millisecond"},
      {"value":"second"},...]},
  "MeterType" : {"type" : "string",
    "options": [
      {"value":"electric"},
      {"value":"gas"},...]}}
```

Figure 3: Example of a JSON object and its schema.

is in the form of compressed JSON which is decompressed by the proxy before reaching the client.

2.1 HTTP/RESTful API

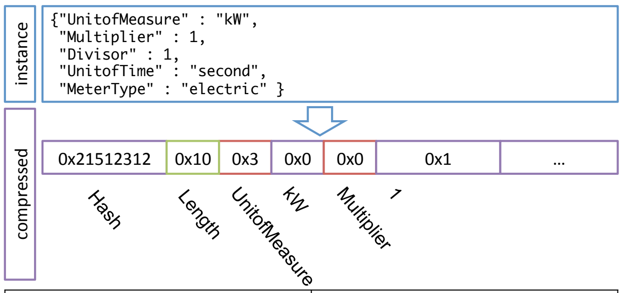
Resources such as sensors, meters, and actuators expose a RESTful API over HTTP. This API defines a sets of HTTP resources which allows a user to sample a sensor or read a meter, view configuration and calibration information, or set up periodic reporting. We define the structure and semantics of this API, as partially shown in top of Figure 2. Standard HTTP GET and POST requests are used to communicate with resources, as shown in bottom of Figure 2, which sets up periodic reporting on meter 325.

2.2 Data Schema

Data is encapsulated in JSON objects defined by a set of data schemas. This set of JSON schemas are defined to be expressive, and enables resource monitors of different classes to interoperate. Figure 3 shows an instance of a JSON object and its schema.

2.3 Proxy / Middlebox Layer

HTTP allows arbitrary proxies or middleboxes to be interposed between endpoints of an HTTP session, and is valuable when dealing with resource-constrained devices. As in other applications of HTTP, these middleboxes may provide a number of services such as load balancing between equivalent sensors, caching of requests, object translation, HTTP header size reduction, and security offload (IPSec proxy). We use proxies to compress and decompress JSON data as shown in Figure 4.



```
{"UnitofMeasure" : "kW",
 "Multiplier" : 1,
 "Divisor" : 1,
 "UnitofTime" : "second",
 "MeterType" : "electric" }
```

application/json	**94 octets**
application/binaryjson	22 octets

Figure 4: Compression of JSON objects is valuable for resource constrained devices.

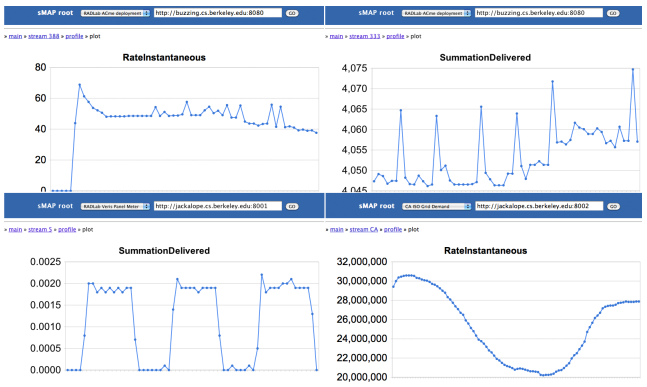


Figure 5: Top two graphs show instantaneous and accumulated energy of two plug-load appliances. Bottom left shows energy consumption of a Veris branch meter circuit. Bottom right shows a portion of California energy usage during a day. All data are obtained through sMAP compliant devices.

3. IMPLEMENTATIONS

We have implemented sMAP on a number of devices including plug-load meters, light sensors, and light switches running TinyOS, a Veris branch electric meter with 30 channels that natively speaks Modbus, and Cal ISO's California live energy feed, as shown in Figure 5.

We invite readers to read more about sMAP at `http://webs.cs.berkeley.edu/smap` and explore our sMAP based application gateway at `http://webs.cs.berkeley.edu/smapview`.

4. REFERENCES

[1] blip: An ipv6 stack for tinyos. `http://smote.cs.berkeley.edu:8000/tracenv/wiki/blip`.

[2] Google powermeter. `http://www.google.org/powermeter/`.

[3] Zigbee smart energy. `http://www.zigbee.org/Markets/ZigBeeSmartEnergy/tabid/224/Default.aspx`.

[4] L. Schor, P. Sommer, and R. Wattenhofer. Towards a zero-configuration wireless sensor network architecture for smart buildings. In *First ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings (BuildSys)*, 2009.