OneIOT API

Vijay Subramanian,

ENG Labs,

Cisco

Problem Statement

- To realize the potential of Internet of things (IoT), we need to be able to program policies that listen to and detect events.
- Once events are detected, we need to take actions or react to them.
- To enable IoT applications, we need
  - Data Acquisition and /or analytics at the edge: In particular, we need content-centric rule application and event detection at the edge
  - A programmatic model / framework to make it easy to apply the rules and policies at scale over the sensor space.

How do we write rules/policies that enable us to acquire sensor data efficiently from a large number of sensors while transporting only the data that is needed? How do we write queries that can look at and understand sensor data content?

User Program using Eclipse

IoT API

Our focus is on developing an open API to register rules and policies to obtain sensor data of interest.

Network

IoT API

Sensor Gateway/ Router

Rules Engine

Goal is to make it easy to program the data acquisition from sensors.

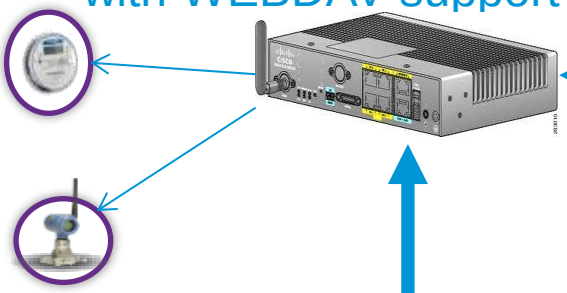
MQTT

Requirements

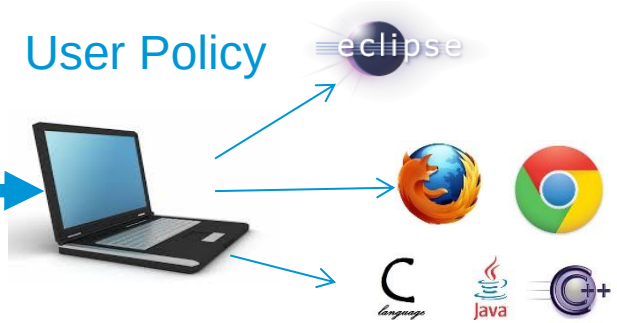
- Examples of rules and policies would be obtaining only i-frames from video to minimize the data sent back and then requesting more data if needed.
- Reading sensor data (temperature / pressure) from field sensors and performing comparison operators on the payload fields. (Example: Send back gps coordinates if temperature is between 20 and 30 F or if pressure is > x psi for more than 30 seconds.)
- Sensor data may be encapsulated in HTTP headers or may be in UDP or TCP packets directly. The format of the data mey be structured (JSON/ XML) or it may be unstructured (raw or proprietary format).

Overall Architecture

IOT Edge Device such as 819
with WEBDAV support

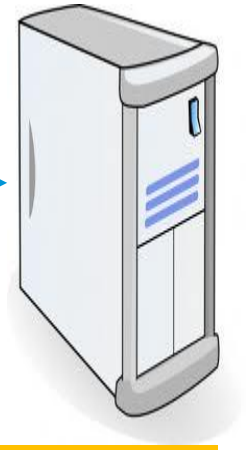


User Policy



Register policies using
REST API

Apache Web server



Results returned using REST API

RESTful Architecture with Pub/Sub semantics

Sample Format of the JSON message

```
{
        "operation" : "POST",
        "rulename" : "Rulename",
        "context" : "cgi-bin/test",

        "network"      :   {
                                "dstaddr" : "172.27.231.12" ,
                                "sport" : "80" ,
                                "dport" : "4001" ,
                           },

        "http"         :   {
                                "Content-Type" : "application/video" ,
                                "Content-Length" : "30" ,
                                "Host" : "www.youtube.com" ,
                           },

        "directives"   :   {
                                "header" : "1" ,
                                "payload" : "1" ,
                                "timer" : "3" ,
                                "cache" : "1024" ,
                           },

        "response"     :   {
                                "method" : "http" ,
                                "addr" : "10.1.1.1" ,
                                "port" : "5001" ,
                           },

        "application" :    {
                                "query" : "resolution=720p" ,
                           },

}
```

Functions

int **set_rule_directives** (int handle, int timer, int cache)
This function is used to set the timer and cache size directives values that are triggers for processing the streams of interest. More...

int **set_app_condition** (int handle, char *query)
This function is used to set conditions that the application payload has to meet for the rule to be a hit. More...

int **set_response_format** (int handle, char *method, char *addr, char *port)
This function can be used to specify where the response should be sent. Responses are sent to the specified url (method://addr:port) as a HTTP message. data. More...

int **set_net_filter** (int handle, char *saddr, char *daddr, char *sport, char *dport)
This function is used to specify the network related fields of interest. With this set, only flows that match these characteristics will be inspected. More...

int **set_register_params** (int handle, char *ip, char *port)
This function is used to specify where the rule should be sent for registration with the device. More...

int **set_http_response_type** (int handle, int header, int payload)
This function is used to specify what data should be sent back in the response. More...

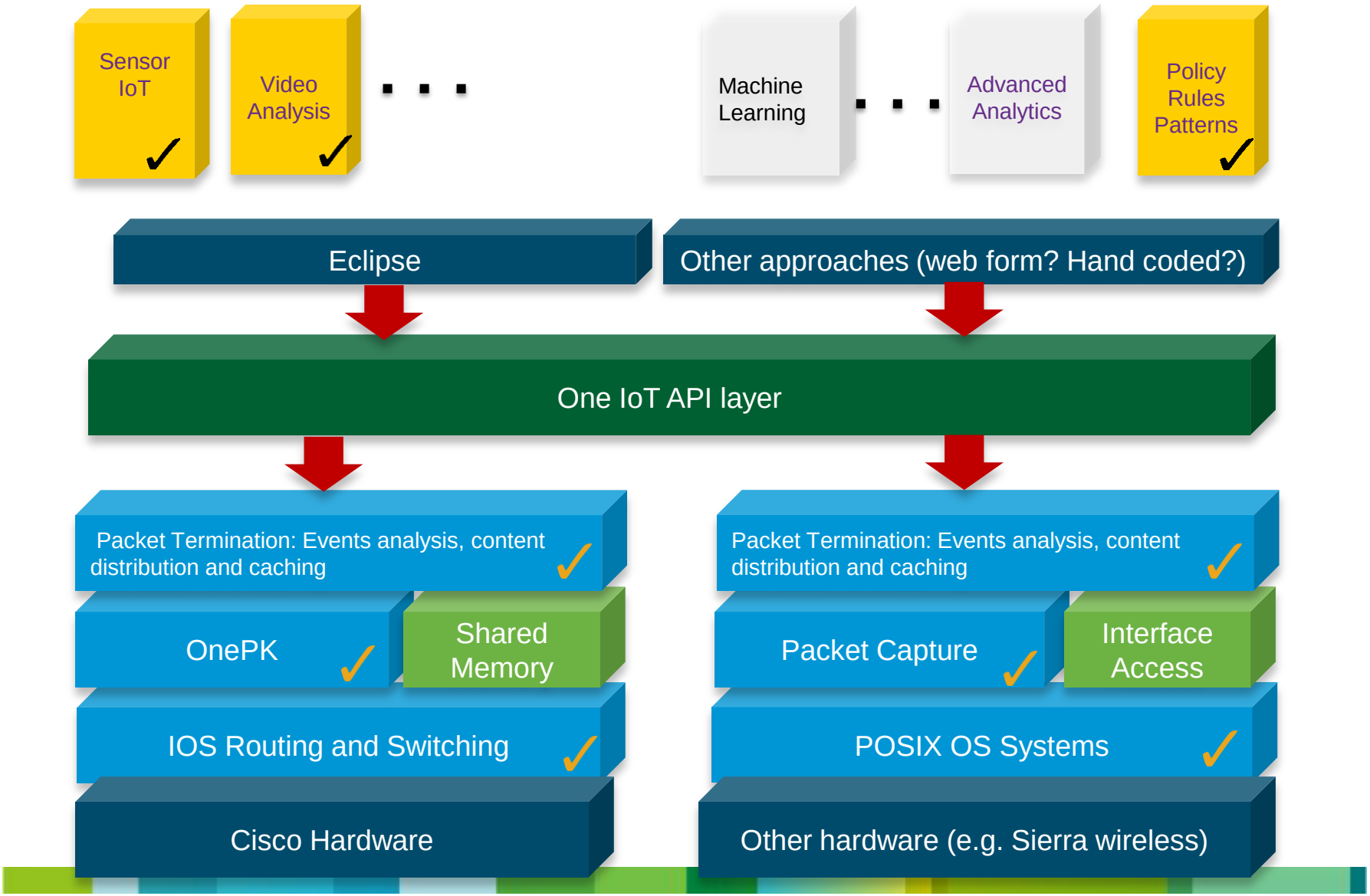
int **set_http_filter** (int handle, char *content_type, int clength, char *host)
Set the HTTP fields by which the stream is to be filtered. More...

int **create_request** (char *name, char *context, char *operation)
Create a request using the given parameters. More...

int **init_api** (void)
Initialize the API library. More...

int **register_request** (int handle)
Register a request. More...

OneIOT API for Data in Motion



Thank you.