ACTA TECHNICA CORVINIENSIS
– Bulletin of Engineering
Tome IX [2016] , Fascicule 2 [April – June]
ISSN: 2067 – 3809

[1.] Tanmoy SARKAR, [2.] Niva DAS

WEB OF THINGS – PRAGMATIC APPROACH

[1.] Neudesic India Pvt. Limited, Hyderabad, INDIA
[2.] Kelton Tech Solutions Limited, Hyderabad, INDIA

Abstract: In recent years IoT (Internet of Things) is gaining interest among researchers and many IoT development platforms, SDK's have been proposed. There are lots of low powered, efficient devices available in market which can be used to create IoT solutions. The main drawbacks of IoT is that every manufacture proposed it owns protocol and provide their own API's to work with it. You need to be an avid programmer to implement, understand, customized and use this API's. Since lots of these libraries are open source not much information can be gathered online. To solve this protocol agnostic problem in IoT many researchers are now focusing on WoT i.e. Web of Things. The main advantage of implementing WoT is that we can use familiar Web protocols to build up our solutions. In this paper we will discuss SOA architecture, proposing a high level WoT network implementation at home-using embedded devices and Web protocols like REST web services and JavaScript to achieve interoperability. We will also discuss security concerns in WoT and best practices to avoid vulnerabilities.

Keywords: WoT, Web of Things, IoT, Internet of Things, REST, SOA

INTRODUCTION

With the popularity of IoT more and more devices manufactured by different vendors are coming to market. Each device has their own specification and protocols supported.

However, the central focus of IoT is to achieve interconnectivity among these devices and how to achieve interoperability while ensuring trust and security [1] [9] [10] [11]. Also, different operating system proposed in IoT for resource constraint devices [2]. In [19] author proposes to use cognitive radio with IoT and in [18] author discuss about secure communication in cognitive radio. To overcome this mismatch many researchers are moving to WoT which is a subset of IoT but here to communicate between devices they are using Web protocols [3]. WoT, unlike IoT which works on all layers of OSI, mainly works on OSI layer 1. Because of this high level of abstraction, it is easier to connect devices and send messages among them. Also, with the growing maturity of cloud computing, data can be easily stored, shared and retrieved. To implement physical mashup communication where physical devices communicate with virtual platforms web technologies like JavaScript, REST can be used. JavaScript has been tested and tried from last 10 years and is mostly used nowadays in both client side as well as server side scripting language in Web. It can become WoT programming language in near future with the increasing user base. Also, REST is well defined protocol which works with many different data format like XML, JSON and used pre-defined verbs like GET, POST, PATCH, and PUT. Using these well-defined protocols and programming languages in our system we can develop our solution without any worry of new protocol implementation. JSON data is preferable over XML data in WoT because unlike latter JSON data are much faster to parse and doesn't need strict validation. Today nearly every device can consume REST services and parse JSON data with ease. In [4]-[8] authors have applied REST to smart devices mainly considering interoperability, mash ability and complexity.

SERVICE ORIENTED ARCHITECTURE

Service Oriented Architecture (SOA) is collection of services which communicate with each other and pass data among them. In SOA the basic components are service and connection. A service is independent, fully functional, well defined function. To connect these services the most common connection we used is web service.

The two most popular Web Services approaches are:

WS-* Architecture: This is most commonly known as SOAP (Simple Object Access Protocol) initially builds by Microsoft which becomes standard by

W3C. SOAP implement various WS-* services. It basically woks with XML file for data transmission, WSDL for Service definition and UDDI for Service discovery. Before parties can communicate with SOAP they need to share the message structure, protocol use among themselves. Any deviation from accepted structure can result in message rejection in SOAP. SOAP is a protocol which implements WS*-Architecture.

a. REST architecture: REST (Representation State Transfer) is not a protocol but rather define some style on how to consume services using standard HTTP protocol. REST doesn't dependent on any single communication protocol but any protocol which supports URI can use REST. Also, unlike SOAP no prior knowledge of data is required. REST uses HATEOAS (Hypermedia as the Engine of Application State) constraint to achieve this functionality. It's a constraint in REST services where client communicate with services based on hypermedia provided dynamically by application servers.

In WoT, REST is considering to be the first class citizen to communicate data among devices over HTTP because of following reasons:

1. No prior knowledge of data required beyond media type which the resource can provide.
2. REST permits different data format but SOAP only works with XML.
3. REST can be used with different communication protocol.
4. REST supports JSON data which is lightweight than XML and can be easily parsed.
5. REST can use HTTP protocol security mechanism like SSL for secure communication.
6. SOAP messages are more secure than REST because it can implement WS-Security which offers confidentiality and integrity of data. But, these securities overhead consume lot of resources and many embedded devices are resource constraint. So, in spite of REST is not as secure as SOAP we can limit the security vulnerabilities by following some best practices.

In [4], the author uses REST API's which gets data from Bluetooth enable embedded devices and visualize data/functionality like power consumption, power on/off on web-page.

SECURITY CONCERNS

Security is a major concern for any communication nowadays. In paper [6], the author discuss about different security and privacy issue in IoT. There are ways to secure IoT communication [8] but again will extra overhead on embedded devices. In WoT, all the communications are REST API based. So, attacker can attack the API's and insert malicious data with techniques like script injection, SQL injection. The best way to avoid this is to follow some REST best practices [17] and limit the attack surface.

1. Since REST API are stateless authentication should not depend on cookies or sessions. Each request must come with authentication data like API key. API key can either be incorporated into URL or message header. The problem with API key in URL is anybody can copy the key and share it with others. So, it's better to have API key within header than URL.
2. The API key within header is still being traceable because the credentials are travelling through wire. So, it always better to signed request. For this, it's always better to use HTTPS over HTTP.
3. Avoid forwarding failure request to less secure API. Always better to send 404 errors in case of authorization failure with proper message.

Some best practices of JavaScript:

1. Reduce the use of global variables.
2. Reduce the possibility of undesirable re-declarations
3. Reduce the use of anonymous functions for better debugging experience and maintainability.
4. Avoid using Eval statement.
5. Always use HTTPS and avoid HTTP communication.

But still after following these best practices there are always room for well know network attacks like man-in-the-middle attack, DoS attack. The most useful technique to break secure communications which are using HTTPS is SSLStrip. SSLStrip acts as a proxy for HTTPS traffic and send the request as HTTP. Since the HTTP traffics are not secure attacker can use tcpdump to retrieve user credentials. To overcome SSLStrip attack the best way is to always host API using HTTPS and doesn't forward the request to HTTP in case of failure. In this paper we are discussing setting up home WoT network so chances of network attack are minimal.

TOPOLOGY USED

In this network topology Fig.1 we are using Raspberry Pi as a central point of communications which interacts with different devices to get data and forward it to cloud storage. Here, Arduino is used to sense room temperature/humidity with the help of sensor DHT11 and post message (interval can be set programmatically) to Raspberry pi by consuming REST API exposed by Raspberry Pi with the help of ESP8266 Wi-Fi module. End systems like smartphone/Laptop are request initiating devices which request data and get response from cloud storage. The data flows as follows:

1. Arduino sends sensor data to raspberry pi (within interval) to raspberry pi.

2. Raspberry pi in turn sends the data to cloud storage.
3. Request initiating devices like smartphone/ laptop request room temperature/humidity by consume API exposed by cloud service.
4. Once the cloud receives the request it sends the latest data got from Raspberry pi.
5. Request initiating devices like smartphone, laptop can then use these data for analytics purpose.

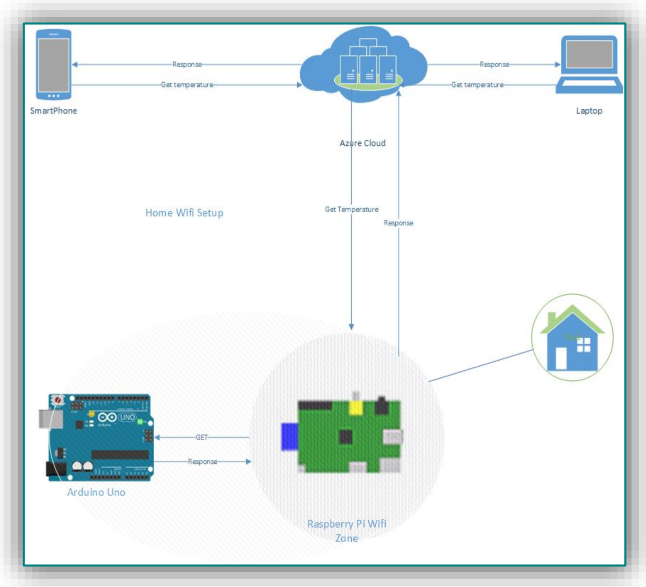

Figure 1. Network topology diagram of WoT devices

ALGORITHM

To implement below algorithm we have used JavaScript in Raspberry Pi and C language in Arduino. Today JavaScript become the language of Web and we can leverage its power in WoT. There are client side JavaScript modules available like angularjs [14] as well as server side JavaScript like Nodejs [16]. For, cross platform mobile application Ionic framework [15] is used which can be programmed using angularjs, HTML5 and CSS3 [12]. There are lots of open source JavaScript modules available which helps in fast development.

Arduino doesn't have built in Wi-Fi module. But for Wi-Fi communication Arduino Wi-Fi Shield is available. In this experiment we are using ESP8266 cheap, efficient Wi-Fi module for communication with Raspberry Pi.

Communication between Arduino and Raspberry Pi.

```
1. Initialize variables
2. Start initiating connection
3. We have introduced interval of 10 mins so that
   Raspberry Pi should not be flooded with data.
4. While(ESP8266 is available){
   a. Configure header with registered Device id.
      The device registration should be done
      beforehand and must be unique for audit
      purpose
   b. POST data to Raspberry Pi Rest API using
      ESP8266 command.
   }
5. While(wait for response){
   Log full response for audit purpose;
   }
6. Close connection
```

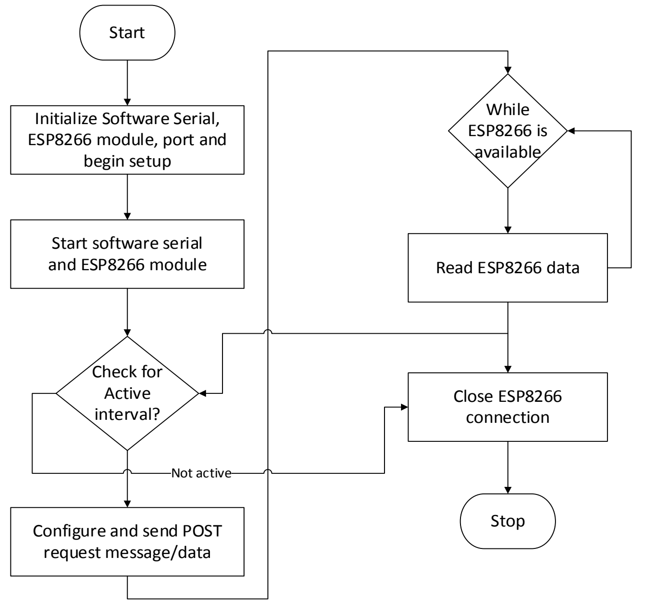
Start
Initialize Software Serial, ESP8266 module, port and begin setup
Start software serial and ESP8266 module
Check for Active interval?
Not active
Configure and send POST request message/data
While ESP8266 is available
Read ESP8266 data
Close ESP8266 connection
Stop

Figure 2. Arduino workflow

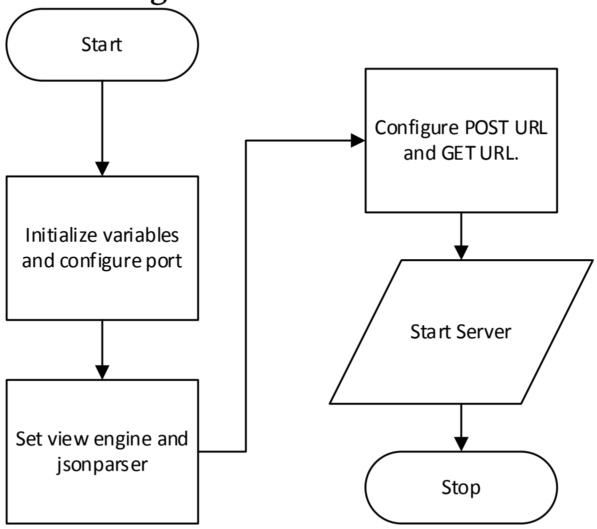
Start
Initialize variables and configure port
Set view engine and jsonparser
Configure POST URL and GET URL.
Start Server
Stop

Figure 3. Node js workflow

Communication between Raspberry Pi and Cloud Service

```
1. Initialize variables and configure port
2. Start initiating connection
3. While(Listen for incoming messages)
   {
   a. Configure POST message using device name
   b. Configure request header
   c. Apply authorization key to header
   d. Send data to cloud storage
   }
4. While(wait for response)
   {
   Log full response for audit purpose;
   }
```

For complete code and configuration visit [13].

The communication between Arduino and Raspberry Pi is not secure and doesn't follow REST best practices. We keep it simple with the assumption that since devices are connected to LAN rather than WAN, they are more secure from outside attacks.

CONCLUSION

With the invent of WoT, more and more researchers are moving towards it because of its simplicity, interoperability and use of well-defined/tested web technologies But with the increasing number of devices security becomes more and more vulnerable. The way we can limit WoT vulnerabilities is by following Web technologies best practices. Also, the rises of web languages like JavaScript, HTML5 help WoT researchers and enthusiasts' jobs much easier. Now with vibrant open community available WoT can be the future of connected devices.

References

[1.] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Comput. Network. vol. 54, pp. 2787–2805, October 2010.

[2.] T. Borgohain, U. Kumar and S. Sanyal, "Survey of Operating Systems for the IoT Environment" in arXiv preprint arXiv: 1504.02517, 2015

[3.] D. Guinard and V. Trifa, "Towards the Web of Things: Web Mashups for Embedded Devices," in Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain, Apr. 2009.

[4.] D. Guinard, C. Floerkemeier, and S. Sarma, "Cloud computing, rest and mashups to simplify rfid application development and deployment," in Proceedings of the 2nd International Workshop on the Web of Things (WoT 2011). San Fransisco, USA: ACM, June 2011.

[5.] D. Yazar and A. Dunkels, "Efficient application integration in IP-based sensor networks," in Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, ser. BuildSys '09. New York, NY,

[6.] W. Drytkiewicz, I. Radusch, S. Arbanowski, and R. Popescu-Zeletin, "pREST: a REST-based protocol for pervasive systems," in Mobile Ad-hoc and Sensor Systems,

[7.] V. Stirbu, "Towards a RESTful Plug and Play Experience in the Web of Things," in IEEE International Conference on Semantic Computing, Aug. 2008, pp. 512–517.2004 IEEE International Conference on. IEEE, 2004, pp.340–348.

[8.] E. Wilde, "Putting things to rest," School of Information, UC Berkeley. Report 2007-015., Tech. Rep., 2007. [Online]. Available: http://escholarship.org/uc/item/1786t1dm

[9.] T Borgohain, A Borgohain, U Kumar and S Sanyal, "Authentication Systems in Internet of Things", arXiv preprint arXiv: 1502.00870, 2015

[10.] T Borgohain, U Kumar and S Sanyal, "Survey of Security and Privacy Issues of Internet of Things", arXiv preprint arXiv: 1501.02211, 2015

[11.] U Kumar, T Borgohain and S Sanyal, "Comparative Analysis of Cryptography Library in IoT", arXiv preprint arXiv: 1504.04306, 2015

[12.] J. Kopeck, K. Gomadam, and T. Vitvar, "hRESTS: an HTML microformat for describing RESTful web services," in Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. IEEE Computer Society, 2008, pp. 619– 625.

[13.] https://iotguys.wordpress.com/

[14.] https://angularjs.org/

[15.] http://ionicframework.com/

[16.] https://nodejs.org/en/

[17.] http://www.programmableweb.com/news/how-to-secure-your-rest-api-right-way/2014/05/22

[18.] S. Sanyal, R. Bhadauria, and C. Ghosh, "Secure communication in cognitive radio networks," in Proc. Computers and Devices for Communication (CODEC), 2009, pp. 1–4.

[19.] Q. Wu et al., "Cognitive Internet of Things: A new paradigm beyond connection," IEEE Internet Things J., vol. 1, no. 2, pp. 129–143, Apr. 2014

ACTA Technica CORVINIENSIS
BULLETIN OF ENGINEERING
fascicule 1
fascicule 2
fascicule 3
fascicule 4
ACTA TECHNICA CORVINIENSIS BULLETIN OF ENGINEERING

ACTA Technica CORVINIENSIS
BULLETIN OF ENGINEERING

ISSN:2067-3809