International Journal of Latest Trends in Engineering and Technology
Special Issue SACAIM 2017, pp. 179-183
e-ISSN:2278-621X

AN ANALYSIS OF COMMUNICATION WITH IOT DEVICES USING WEBSOCKETS

Preethika .R[1], Ganesh Prabhu[2] & AravindaPrabhu .S[3]

Abstract – The WebSocket allows bi-directional communication between a Web-based application and a Web-server. WebSockets can be used to develop remote communication applications, instant messaging services and many more real-time applications. Since, WebSocket allows asynchronous full-duplex communication, it can be used to develop simple Web-based Internet of Things(IoT) applications. In this paper, we try to analyse the functionality of WebSockets alongside IoT applications.
Keywords – Internet of things, communication, WebSockets, messages.

1. INTRODUCTION

Internet of things have been developing at a very fast pace. There are four communication models for the Internet of Things namely Device-To-Device communications, Device-to-Cloud communications, Device-to-Gateway model, Back-End Data-Sharing Model. We will be mainly focussing on establishing device-to-device communications using WebSockets. WebSocket allows communication among two devices. This makes it easy for the users to send command to the devices and vice versa.

2. WEBSOCKET COMPONENTS AND FUNCTIONALITY

2.1 WebSocket Components-

WebSockets send data using a continuous connection that is established between the client and the server.The user builds up a WebSocketconnection through a process known as the WebSocket handshake. Theuser then sends a normal HTTP request to the server. An Upgrade header is incorporated into this request informing the server that the user wishes to build up a WebSocketconnection. When the handshake is finished, it replaces the HTTP connection to the WebSocket connection which uses the same hidden TCP/IP connection. This allows both parties to send data among each other.

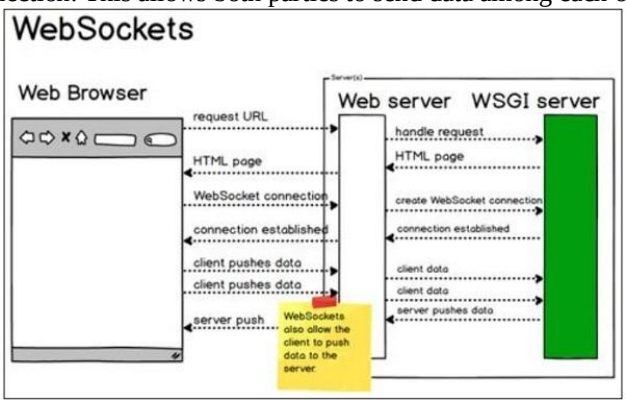

Figure 1. Websocket Components[1]

2.2 Functionality-

a) Websockets runs along with any protocol that runs over the top of a pure TCP connection.
b) WebSocket has transport layer as the topmost layer at which any other protocols can run. WebSocket API has the capability to write sub-protocols: protocol library that can interpret specific protocols.
c) WebSockets require the browser side to run a javascript library that can identify WebSocket handshake, establish and maintain a WebSocket connection.
d) On the server side, it can use any existing protocol library that run of top of tcp and leverage a WebSocket gateway.

[1] MCA V Semester, St Aloysius College, AIMIT, Mangaluru, Karnataka, India
[2] MCA V Semester, St Aloysius College, AIMIT,Mangaluru, Karnataka, India
[3] Asst Professor, Department of Computer Science & Bio Informatics, AIMIT, St Aloysius College Mangalore, Karnataka, India

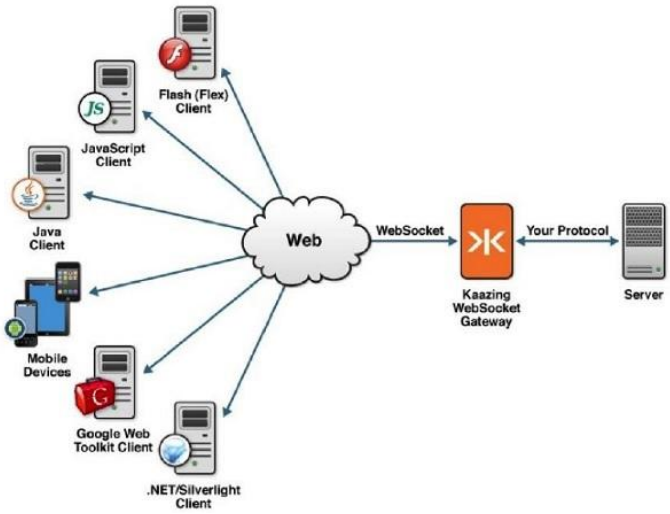

Figure 2. Websocket functionality[2]

3. IOT ARCHITECTURE AND USAGE OF WEBSOCKET IN IOT

Internet of Thing(IoT) can be anything ranging from a person with a heart monitor transplant to small present in the electronic device which has been assigned an IP address and provided the ability to transfer data into network constantly. The application of IoT can be found in many industries today, which includes healthcare, automobile, agriculture and electronic device manufacturers. The IoT appliance dates back to 1980s, with a Coke Machine at Carnegie Melon University being the first known device. This machine was programmed to connect over the internet and generate the status to determine whether there was a cold drink awaiting at the machine, or should the machine be tripped down. The general architecture of IoT is represented below.

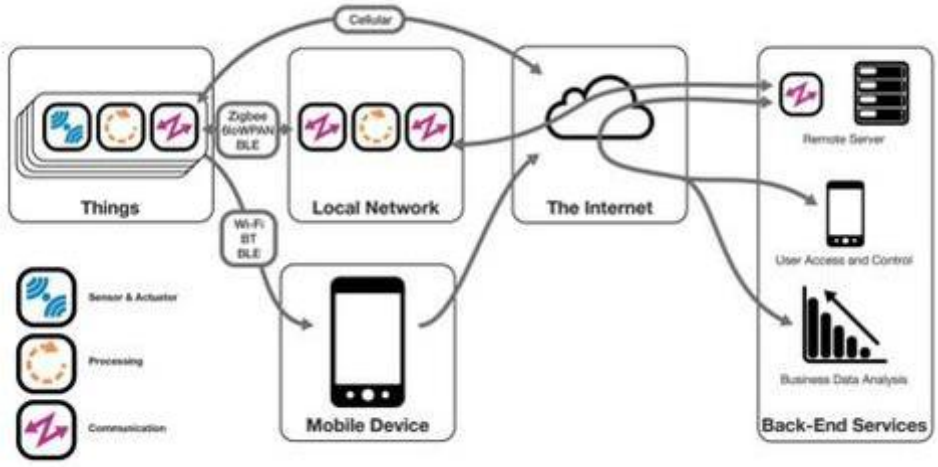

Figure 3. Simple IoT architecture[3]

There are various IoT Communication models

1. Device-to-Device Communications
2. Device-to-Cloud Communications
3. Device-to-Gateway Model
4. Back-End Data-Sharing Model

We use Device-to-Device Communication model to establish a network amongst the Devices.The device-to-device communication model works alongside two or more devices that connects directly and communicateswith each other. These devices communicate over variousnetworks. These devices use protocols like Z-Wave, ZigBee or Bluetooth to determine direct device-to-device communications.

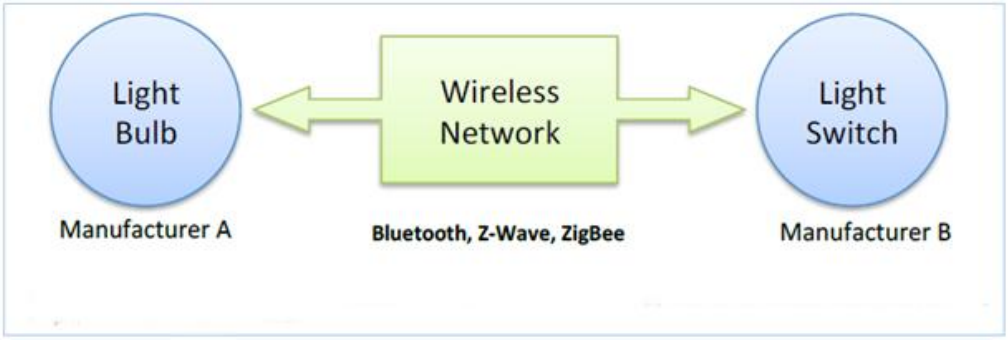

Figure 4. Representation of Device-to-device communication model[4]

To establish Device-To-Device communication we will be using WebSockets, for this we need to establish connection between client and server using WebSocket programs.

4. IOTIMPLEMENTATION

To establish a WebSocket connection. We first have to develop a client program which we can use HTML, CSS and JavaScript.

Step 1-

Thesnippet of establishing a Socket connection is shown below.

```
socket.onopen = function() {
  if (socket) {
    sock=socket;
     clearTimeout(timer);
     servers.push(socket.url);
  }
};
```

Figure 5. Socket Connection code

Step 2-

Create a server program, which listens to socket handshake and responds.

For Demo purposes, we will use Python server code which also contains an autobahn library

```
class BroadcastServerFactory(WebSocketServerFactory):

    """
    Simple broadcast server broadcasting any message it receives to all
    currently connected clients.
    """

    def __init__(self, url):
        WebSocketServerFactory.__init__(self, url)
        self.clients = []
        self.tickcount = 0
        #self.tick()

    def tick(self):
        self.tickcount += 1
        self.broadcast("tick %d from server" % self.tickcount)
        reactor.callLater(1, self.tick)

    def register(self, client):
        if client not in self.clients:
            print("registered client {}".format(client.peer))
            self.clients.append(client)

    def unregister(self, client):
        if client in self.clients:
            print("unregistered client {}".format(client.peer))
            self.clients.remove(client)
```

Figure 6. Server side code

Step 3-

After establishing connection between both client and server. We write a code to broadcast messages.

```
def broadcast(self, msg):
    print("broadcasting message '{}' ..".format(msg))
    for c in self.clients:
        c.sendMessage(msg.encode('utf8'))
        print("message sent to {}".format(c.peer))
```

Figure 7. Broadcast snippet

Step 4-
Create a Javascript file for the webpage. This accepts responses and receives broadcast from the server file.

```
function broadcast() {
   var msg = document.getElementById('message').value;
   if (sock) {
      sock.send(msg);
      log("Sent: " + msg);
   } else {
      log("Not connected.");
   }
};
function broadcastLight() {
  if (sock) {
      sock.send("Light On");
      log("Sent: " + msg);
   } else {
      log("Not connected.");
   }
};

function log(m) {
   ellog.innerHTML += m + '\n';
   ellog.scrollTop = ellog.scrollHeight;
};
```

Figure 8. JavaScript that receives from broadcast

4.1 Working-
Assume a simple webpage, which has two buttons, that allows the users to control either fan or light appliance.

Figure 9. HTML Demo Page

For understanding purposes, we have used a simple log created using HTML and CSS. This log display current status of the client and server

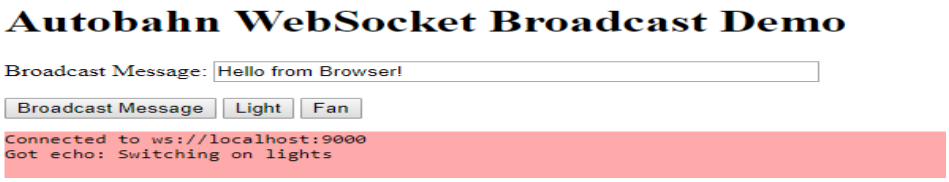

Figure 10. HTML Demo page with "Light" clicked

When we click on the "Light" button, user sends a message to the listening server using a WebSocket to switch on the lights.

```
D:\broadcast sacaim>python sac_broadcast.py
2017-11-02 20:29:54+0530 [-] Log opened.
2017-11-02 20:29:54+0530 [-] BroadcastServerFactory starting on 9000
2017-11-02 20:29:54+0530 [-] Starting factory <__main__.BroadcastServerFactory object at 0x041C17D0>
2017-11-02 20:29:54+0530 [-] Site starting on 8080
2017-11-02 20:29:54+0530 [-] Starting factory <twisted.web.server.Site instance at 0x041D1378>
2017-11-02 20:29:55+0530 [-] registered client tcp4:127.0.0.1:50325
2017-11-02 20:53:17+0530 [-] broadcasting message 'Switching on lights' ..
2017-11-02 20:53:17+0530 [-] message sent to tcp4:127.0.0.1:50325
```

Figure 11. Python server side program

The server python file receives the message and sends an function to underlying IoT device to “Switch on” the lights.

5. CONCLUSION

Websockets can be effectively used with IoT devices. Since WebSocket can communicate over any underlying TCP/IP protocols. It can be used with respect to the most of the modern programming languages. The biggest advantage of WebSocket is its network compatibility. The major benefit to the WebSocket protocol is its server-side library, it makes implementation on a server easier becauseof the wide variety of WebSocket servers. Similar to HTTP, WebSocket is capable of using TCP and, thus, it has capability of using TLS and used its wider availability in existing network stacks.

6. REFERENCES

[1] https://kaazing.com/doc/legacy/4.0/about/kaazing-websocket-html5.html

[2] https://www.fullstackpython.com/websockets.html

[3] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, "Internet of Things (IoT): A vision architectural elements and future directions", Future Gener. Comput. Syst., vol. 29, no. 7, pp. 1645-1660, Sep. 2013.

[4] Y. Furukawa “Web Based control application usng Websocket”, ICALEPPC 2011, 2011, Grenoble, France

[5] http://websocket.org/