CSYE 6510: IoT Assignment #3, 100 points total)

Due on 3/28/20

OPTION 1:

Please read the following paper (presented in May 2018 in the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)) and answer the following questions:

1) When describing HTTP, the authors say "Hypertext Transfer Protocol (HTTP) is a connectionless client/server protocol ubiquitous in IT and the web.". But HTTP runs over *connection oriented* TCP, what do they mean by *connectionless* ?

2) Figure 5 shows the message flow for a level 1 QoS MQTT scenario. How does it change when either level 0 or level 2 is used instead ?

3) Under CoAP/HTTP, this paper describes a typical IoT scenario with sensor/gateway interaction. Can you describe what hardware is used in each case ?

4) Under CoAP, this paper mentions client and server. What device (sensor/gateway) is the client ? What device (sensor/gateway) is the server ? How does differ from the examples shown in RFC 7252 ? Moreover, what are the implications of the client/server scheme presented in this paper when compared to that of the examples in RFC 7252 ?

5) How do you think the results shown on Figure 10 would change if the interface was IEEE 802.15.4 instead of Ethernet/Wifi ?

6) Assume you are a performing peer-review of this paper for a journal editor: what are the paper weak points? what is it missing? what would you modify?

OPTION 2:

Goal: Run MQTT traffic

Requirements:

1. Install an MQTT broker (i.e. MQTT mosquitto).
2. Install an MQTT client (i.e. Paho MQTT or MQTT mosquito subscriber/publisher).
3. Launch the broker on a VM/computer.
4. Have one client instance (app) (on another VM/computer) subscribe to a "temperature" topic.
5. Have another client instance (sensor) (on another VM/computer) publish a "temperature" event.

Questions:

1. Perform the following steps:
    1. Capture (*and paste*) wireshark traces between clients and broker.
    2. Measure how long it takes for the sensor event to reach the app (use wireshark timestamps).

for uniform 0%, 5% and 10% packet loss introduced at the sensor. Make a table *packet loss* vs *latency* for each quality level *at most once*, *at least once* and *exactly once*.

2. Describe the messages that are exchanged between devices and broker.

3. How does quality affect latency? Why?

MIPRO 2018, May 21-25, 2018, Opatija Croatia

Energy efficiency of Machine-to-Machine protocols

Marko Pavelic, Vatroslav Bajt, Mario Kusek
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Zagreb, Croatia
marko.pavelic@fer.hr; vatroslav.bajt@gmail.com; mario.kusek@fer.hr

***Abstract*—The paper provides a comparison of energy consumption for a few commonly used M2M protocols, used on energy constrained devices in M2M communication systems. It was implemented on Arduino hardware platform. Energy consumption for each protocol is measured on boards with both Ethernet and WiFi communication modules. Servers with which Arduino is communicating are all implemented on a Raspberry Pi, which also serves as an M2M gateway. Traffic is captured for each protocol on both wired and wireless interface and compared to energy consumption.**

Keywords—Arduino, M2M, energy consumption, Raspberry Pi, HTTP, CoAP, MQTT

I. INTRODUCTION

Machine-to-Machine (M2M) communication is considered as one of the most demanding, but promising technologies for implementation of new generation of networks. M2M communications are typically characterized by connecting a large number of autonomous devices to other devices in both wireless and wired systems. These devices communicate with a central controller or with each other without human intervention. Many of these devices do not have unlimited energy supply, they are battery-powered and therefore, energy efficiency is an important factor in M2M systems.

Energy efficiency is defined as ratio of useful energy used for work out of total energy available from energy sources. The goal is to achieve maximal energy output with minimal energy waste and potentially negative effect on the environment.

One of the most common applications of M2M communication is a smart home system. These systems have wired and wireless devices connected to it, depending on the device function and location. Some of these devices have to have their own power source and on a home network with good signal strength, most of the power consumption is due to the efficiency of the device itself and efficiency of the running program. As there are many M2M protocols that can be implemented to run on such devices we take a look at performance of a few of these protocols implemented on an Arduino hardware platform. Protocols which are analyzed are Hypertext Transfer Protocol (HTTP) [1], Message Queuing Telemetry Transport (MQTT) [2] and Constrained Application Protocol (CoAP) [3].

II. RELATED WORK

In [4] the most important messaging technologies proposed as the foundation of the next generation of Internet of Things (IoT) and more specifically the Industrial Internet applications are reviewed. An understanding of both the architecture and the message/data sharing requirements of each target system is an important pre-requisite for choosing the most appropriate messaging solution.

Similarly [5] presents the most representative application layer protocols that have gained attention for IoT, while providing a comparison among each other and argue about their suitability for the future of the IoT.

The architecture proposed in [6] consists of the IoT Home Gateway for aggregating data from devices, the Web Based Service Definition Engine for defining the user's required services and the IoT Service Platform for executing the service via the aggregated data and the user's defined services. The proposed IoT Home Gateway provides device management to remove heterogeneity of various devices, the Auto-configuration for dynamic device discovery and the device information exposure to provide required information to third-party and other IoT service platforms. Also it supports discovery of constrained devices such as Arduino by the Auto-configuration mechanism. As a result, we showed implementation results that control various devices according to home energy saving scenario.

The paper [7] proposes an energy consumption model for energy constrained devices in Machine-to-Machine communication system. The model specifies generic tasks that are executed on M2M devices. It was implemented on devices on Waspmote and Arduino hardware platforms. Power consumption was calculated for the states of the proposed model. The model with calculated consumption values is applicable for testing algorithms and techniques for achieving higher energy efficiency. Furthermore, the paper presents a basic operating cycle in which tasks from the specified model are executed on M2M devices and M2M gateway in an energy efficient way. Energy consumption during 24 hours by using the suggested operating cycle is compared on Arduino and Waspmote hardware platforms.

III. IoT Application Protocols

A. HTTP Protocol

Hypertext Transfer Protocol (HTTP) is a connectionless client/server protocol ubiquitous in IT and the web. Because there are countless open source tools that use HTTP, and every coding language has HTTP libraries, it is very accessible. The focus on HTTP in IoT is around Representational State Transfer (REST), which is a stateless model where clients can access resources on the server via requests. In most cases, a resource is a device and the data that a device contains. HTTP provides a transport, but doesn't define the presentation of the data. As such, HTTP requests can contain HTML, JavaScript, JavaScript Object Notation (JSON), XML, and others. In most cases, IoT is standardizing around JSON over HTTP.

HTTP communication is usually established over TCP/IP connections. HTTP can be also established other protocol, such as unreliable UDP [1].

B. MQTT Protocol

Message Queue Telemetry Transport (MQTT) protocol is an application layer protocol designed for resource-constrained devices. It uses a topic-based publish-subscribe architecture. This means that when a client publishes a message M to a particular topic T, then all the clients subscribed to topic T will receive the message M. Like HTTP, MQTT relies on TCP and IP as its underlying protocols. However, compared to HTTP, MQTT is designed to have a lower protocol overhead.

The reliability of messages in MQTT is taken care by three Quality of Service (QoS) levels. QoS level 0 means that a message is delivered at most once and no acknowledgments of reception is required. QoS level 1 means that every message is delivered at least once and confirmation of message reception is required. In QoS level 2, a four-way handshake mechanism is used for delivery of a message exactly once [2].

C. CoAP Protocol

Constrained Application Protocol (CoAP) is a recently developed application layer protocol intended to be used in the communication of resource-constrained devices. This protocol is based on REST architecture and supports request-response model like HTTP. In addition to request-response model, CoAP also supports publish-subscribe architecture using an extended GET method. Unlike MQTT, the publish-subscribe model of CoAP uses Universal Resource Identifier (URI) instead of topics. This means that subscribers will subscribe to a particular resource indicated by the URI U. When a publisher publishes data D to the URI U, then all the subscribers are notified about the new value as indicated in D.

The major difference between CoAP compared to MQTT and HTTP is that CoAP runs on top of the UDP while the latter run on top of TCP. As UDP is inherently not reliable, CoAP provides its own reliability mechanism. This is accomplished with the use of "confirmable messages" and "non-confirmable messages". Confirmable messages require an acknowledgement while non-confirmable messages do not need an acknowledgement [3].

IV. measurement model

Measurement model specifies a simple task executed on M2M device that is the same for each protocol measured. The goal is to achieve minimum energy consumption for the same amount of usable data sent.

Measurement model defines two types of devices: peripheral device and central device. Peripheral device is a device which collects data from its environment by using one or more sensors. Peripheral devices usually have limited source of energy and their energy consumption should be as low as possible. Device which collects data from peripheral devices is called central device. Peripheral device periodically sends the message to the central device and then waits in powered mode for the timer to expire.

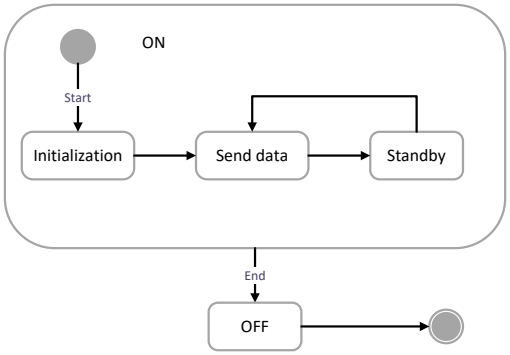

Figure 1 Peripheral device state diagram

In this measurement models all protocols are configured to use features of the protocols which ensure minimal data loss.

For HTTP the POST method is used to communicate with the central device and the response is always status code 204. Similarly to HTTP, method PUT is used in CoAP communication with response status code 2.05. The request and response messages for both protocols can be represented by a sequence diagram in Figure 2.

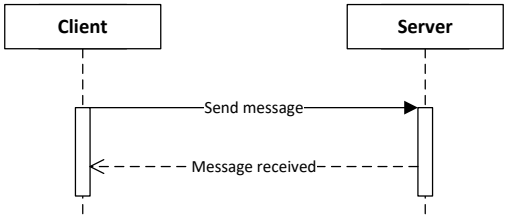

Figure 2 HTTP and CoAP message received acknowledgment

The messages received and sent on server side are shown in Figure 3 for HTTP and Figure 4 for CoAP.

Request

```
> Transmission Control Protocol, Src Port: 4117 (4117), Dst Port: 8889 (8889), Seq: 71, Ack: 1, Len: 57
> [3 Reassembled TCP Segments (127 bytes): #104(69), #113(1), #118(57)]
v Hypertext Transfer Protocol
  > POST /arduino HTTP/1.1\r\n
    Host: localhost\r\n
    Connection: close\r\n
    Content-Type: application/json\r\n
  > Content-Length: 13\r\n
    \r\n
    [Full request URI: http://localhost/arduino]
    [HTTP request 1/1]
    [Response in frame: 120]
v JavaScript Object Notation: application/json
  v Object
    v Member Key: "tempIn"
        Number value: 20
```

Response

```
> Transmission Control Protocol, Src Port: 8889 (8889), Dst Port: 4117 (4117), Seq: 1, Ack: 128, Len: 142
v Hypertext Transfer Protocol
  > HTTP/1.1 204 No Content\r\n
    X-Powered-By: Express\r\n
    ETag: W/"a-oQDOV50e1MN2H/N8GYi+8w"\r\n
    Date: Tue, 07 Jun 2016 11:36:47 GMT\r\n
    Connection: close\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.005526000 seconds]
    [Request in frame: 118]
```

Figure 3 HTTP POST request and response

Request

```
> User Datagram Protocol, Src Port: 4097 (4097), Dst Port: 5683 (5683)
v Constrained Application Protocol, Confirmable, PUT, MID:20515
    01.. .... = Version: 1
    ..00 .... = Type: Confirmable (0)
    .... 0000 = Token Length: 0
    Code: PUT (3)
    Message ID: 20515
  > Opt Name: #1: Uri-Path: arduino
    End of options marker: 255
  v Payload: Payload Content-Format: text/plain; charset=utf-8 (no Content-Format), Length:
      Payload Desc: text/plain; charset=utf-8
    v Line-based text data: text/plain
        {"tempIn":1}
```

Response

```
> User Datagram Protocol, Src Port: 5683 (5683), Dst Port: 4097 (4097)
v Constrained Application Protocol, Acknowledgement, 2.05 Content, MID:20515
    01.. .... = Version: 1
    ..10 .... = Type: Acknowledgement (2)
    .... 0000 = Token Length: 0
    Code: 2.05 Content (69)
    Message ID: 20515
    End of options marker: 255
  v Payload: Payload Content-Format: text/plain; charset=utf-8 (no Content-Format), Length:
      Payload Desc: text/plain; charset=utf-8
    v Line-based text data: text/plain
        Message received
```

Figure 4 CoAP PUT request and response

As MQTT functions as a pub/sub model the response depends on the Qos specified, in this case it is Qos-1. As mentioned in Section III. and shown in Figure 5. the response to published message comes when all clients have received the message at least once. These messages are shown in Figure 6.

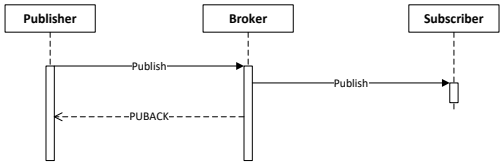

Figure 5 MQTT message delivered acknowledgment

Publish

```
> Transmission Control Protocol, Src Port: 50684 (50684), Dst Port: 1884 (1884), Seq: 1, Ack: 1, Len: 23
v Data (23 bytes)
    Data: 3415000761726475696e6f00037b74656d70496e3a317d
    [Length: 23]
0000  b8 27 eb b9 13 6e e8 99  c4 9d 60 33 08 00 45 00   .'...n.. ..`3..E.
0010  00 4b 4a d3 40 00 40 06  66 0d c0 a8 04 30 c0 a8   .KJ.@.@. f....0..
0020  04 4c c5 fc 07 5c 2e 9d  2f 30 df 13 b1 f8 80 18   .L...\.. /0......
0030  05 59 90 3e 00 00 01 01  08 0a 05 bc 68 5c 00 0a   .Y.>.... ....h\..
0040  74 81 34 15 00 07 61 72  64 75 69 6e 6f 00 03 7b   t.4...ar duino..{
0050  74 65 6d 70 49 6e 3a 31  7d                       tempIn:1 }
```

PUBACK

```
> Transmission Control Protocol, Src Port: 50684 (50684), Dst Port: 1884 (1884), Seq: 24, Ack: 24, Len: 4
v Data (4 bytes)
    Data: 40020001
    [Length: 4]
0000  b8 27 eb b9 13 6e e8 99  c4 9d 60 33 08 00 45 00   .'...n.. ..`3..E.
0010  00 38 4a d5 40 00 40 06  66 1e c0 a8 04 30 c0 a8   .8J.@.@. f....0..
0020  04 4c c5 fc 07 5c 2e 9d  2f 47 df 13 b2 0f 80 18   .L...\.. /G......
0030  05 59 04 58 00 00 01 01  08 0a 05 bc 68 64 00 0a   .Y.X.... ....hd..
0040  78 a5 40 02 00 01                                 x.@...
```

Figure 6 MQTT message delivered acknowledgment

V. PROTOCOL ENERGY CONSUMPTION COMPARISON

This section describes the measurement process for models shown in Figure 1.

The server for all protocols is run on a Raspberry Pi 3 device, which runs a Node.js application that implements HTTP API, MQTT broker and CoAP server. Wireshark is also run to help us identify the time frames when the client is sending messages.

The client for each protocol is implemented on an Arduino Uno hardware platform, with either WiFi or Ethernet Shield.

On Arduino we run a program specified in our model, therefore after initialization the device can be in either of two states shown in Figure 1.

Figure 7 shows our measurement setup.

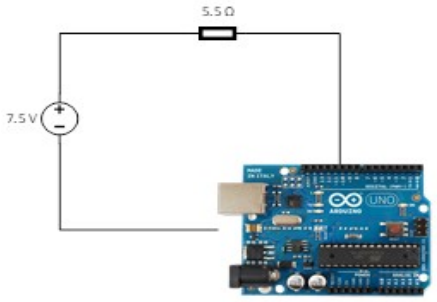

Figure 7 Measurement power circuit

Total amount of energy consumed by the device in a certain time t can be calculated by using the following expression:

$$E = P_D \times t \tag{1}$$

Where P_D is the power of the device which can be calculated by using the expression:

$$P_D = U_D \times I_D \tag{2}$$

Where U_D is the voltage on the device which can easily be measured by using an oscilloscope, while I_D is the current which can be calculated by using Ohms law:

$$U = I \times R \tag{3}$$

In order to see the amount of energy consumed in each state for each protocol of the model proposed in Section IV, a measurement needs to be conducted for each variation. Based on this information, the exact amount of energy which can be saved by using different protocols and/or connection interfaces. Arduino is connected to a constant 7.5V power supply to provide a more stable power input. For our measurement, we had at our disposal an oscilloscope Rigol DS1102D.

Since we are unable to measure the current on the device by using an oscilloscope, we used a known resistance which was connected with our device in a power circuit so that the current passing through the resistor is the same as the current passing through the Arduino device (serial connection). Our power circuit is shown in Figure 5 and Figure 6. We used a resistor with impedance of 5.5 Ω and precision tolerance of 1%.

In the serial connection of the Arduino device and a resistor with known impedance, Kirchoffs voltage law can be applied:

$$U_{total} - U_R - U_D = 0 \quad (4)$$

By using the expressions 3 and 4 we get the equasion:

$$U_{total} = U_R + U_D = I_R \times R_R = I_D \times R_D \quad (5)$$

Since in serial connection we can assimilate the current going through all the elements, we have:

$$I_{total} = I_R = I_D \quad (6)$$

By measuring voltage U_R on the resistor with the oscilloscope, we can easily calculate the current on the known resistor which is equal to the current on the device:

$$I_D = U_R / R_R \quad (7)$$

To help us identify the time frames when Arduino is sending packages we compare measured voltages to the time when the packages are captured on the server. We use U_{base} as an average voltage when the device is in standby state and U_{avg} as the average voltage when Arduino is sending packages. Both values are calculated in designated time period by using the equation:

$$U_{avg} = \frac{1}{T}\int_0^T u(t)\,dt \quad (8)$$

VI. RESULTS

As shown in Figure 1, average power consumption of a certain protocol is calculated for both states and on both interfaces and shown in Tables 1 and 2 and Figures 3 and 4. The data is collected and averaged over 25 cycles. A cycle is a transition between states, described in Figure 1, and lasts 5 seconds.

TABLE I. ETHERNET POWER CONSUMPTION

Protocol	State	
	Send data	***Standby***
CoAP	1338 mW	1293 mW
HTTP	1333 mW	1287 mW
MQTT	1347 mW	1297 mW

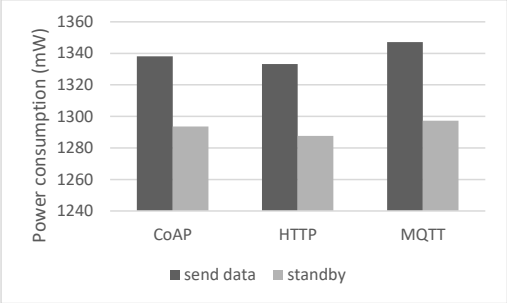

Figure 8 Ethernet power consumption

TABLE II. WIFI POWER CONSUMPTION

Protocol	State	
	Send data	***Standby***
CoAP	987 mW	934 mW
HTTP	1037 mW	985 mW
MQTT	988 mW	940 mW

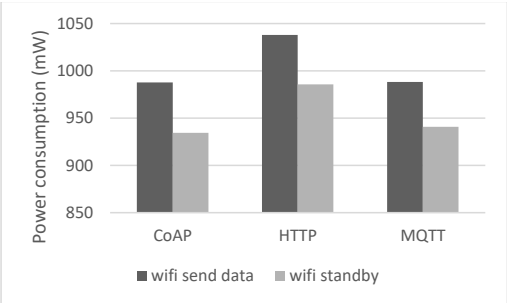

Figure 9 WiFi power consumption

We see from these results that Arduino with Ethernet Shield draws more power than WiFi shield. Furthermore, the difference in power consumption between states is also greater on Ethernet. On average, all protocols consume similar amount of power (within a margin of error) on Ethernet, while on WiFi HTTP consumes about 50 mW more power while either state.

Energy consumption of protocols on an Arduino device with Ethernet or WiFi Shield is presented in Table 3 and Figure 5.

TABLE III. ENERGY CONSUMPTION

Protocol	Interface	
	Ethernet	***WiFi***
CoAP	26,7 mWh	19,7 mWh
HTTP	53,3 mWh	41,5 mWh
MQTT	26,9 mWh	19,7 mWh

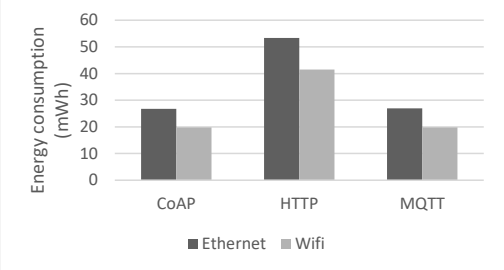

Figure 10 Energy consumption

From the results it can be seen that all protocols consume more energy when they are sending data via Ethernet interface. While CoAP and MQTT both consume almost equal amounts of energy, HTTP consumes two times as much. This is due to larger headers that HTTP has compared to CoAP or MQTT, which results in more packages sent and received, i.e. power is consumed over a longer period.

CONCLUSION AND FUTURE WORK

Energy efficiency is an important aspect which needs to be taken into consideration while designing M2M systems with devices that do not have unlimited power supply. We proposed a basic model for M2M devices and M2M gateways which specifies a generic task executed on devices. This model is used to measure energy consumption of protocols HTTP, CoAP and MQTT.

As shown in our results in Section V. protocols CoAP and MQTT preformed significantly better than HTTP. It is not surprising as CoAP and MQTT were designed as lightweight protocols to be used specifically in IoT, unlike HTTP. Our results also show that both CoAP and MQTT consume almost equal amounts of energy for sending data. While there are many energy efficient protocols that can be used in M2M protocols, an understanding of both the architecture and the message/data sharing requirements of each target system is an important pre-requisite for choosing the most appropriate messaging solution.

There are a lot of ways to expand on this work. Firstly we could include other protocols like Advanced Message Queuing Protocol (AMQP), Extensible Messaging and Presence Protocol (XMPP), Data Distribution Service (DDS) and Java Message Service (JMS), mentioned in [4] and [5]. In this work we only presented energy consumption of an Arduino Uno device when it is sending messages to a server. The device can also be set to receive packages rather than only sending them, or a combination of both. As described in [7] we could also do our measurements on different devices, not just Arduino.

ACKNOWLEDGMENT

This work has been supported in part by research project “Machine-to-Machine Communication Challenges” (M2MCC), funded by Ericsson Nikola Tesla, Zagreb, Croatia.

REFERENCES

[1] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[2] MQTT Version 3.1.1. Edited by Andrew Banks and Rahul Gupta. 29 October 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[3] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

[4] Prism Tech, “Messaging Technologies for the Industrial Internet and the Internet of Things”, November 2013 http://www.prismtech.com/sites/default/files/documents/MessagingComparsionNov2013USROW_vfinal.pdf

[5] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," Transaction on IoT and Cloud Computing, vol. 3, no. 1, pp. 11-17, 2015, https://jesusalonsozarate.files.wordpress.com/2015/01/2015-transaction-on-iot-and-cloud-computing.pdf

[6] P. Skocir, S. Zrncic, D. Katusic, M. Kusek and G. Jezic, "Energy consumption model for devices in machine-to-machine system," Telecommunications (ConTEL), 2015 13th International Conference on, Graz, 2015, pp. 1-8.

[7] S. M. Kim, H. S. Choi and W. S. Rhee, "IoT home gateway for auto-configuration and management of MQTT devices," Wireless Sensors (ICWiSe), 2015 IEEE Conference on, Melaka, 2015, pp. 12-17.