Performance Analysis of Constrained Application Protocol Using Cooja Simulator in Contiki OS

Kanchana P. Naik
Computer Network Engineering, Dept. Of ISE
NMAM Institute of Technology
Nitte, India
naikkanchana5@gmail.com

Rakesh Joshi U.
Asst. Professor , Dept. Of ISE
NMAM Institute of Technology
Nitte, India

***Abstract*—As the Internet of Things developed, many devices organized into network, which collects and exchanges the data among themselves. The Constrained Application Protocol (CoAP) is presented in this paper which is an application layer protocol of IoT protocol stack. It is used with low power , resource constrained nodes and constrained networks in the Internet of Things. CoAP is introduced by IETF CoRE Working Group. CoAP provides low overhead and supports machine to machine M2M communication. This paper shows the implementation of CoAP and comparison of CoAP with HTTP with regard to energy consumption and response time of both client server transaction and the results shows that CoAP is more appropriate compared to HTTP. The simulations has been carried out using Contiki Operating system and Cooja simulator which serves for networked, memory constrained systems and low power wireless IoT devices.**

***Keywords*—** CoAP, IoT, IETF, CoRE, HTTP, Contiki OS, Cooja simulator

I. INTRODUCTION

The new web service paradigm is introduced by the IETF CoRE Working Group for the networks which consists of many smart objects. CoAP is a lightweight HTTP protocol. It is a web transfer protocol for the constrained devices. As wireless networks are low-powered and lossy in nature, connection-less UDP, instead of stream-oriented TCP, is mostly used in the IoT. The CoAP consist of certain HTTP functionalities which are altered for the small embedded devices like sensor nodes in order to make it applicable for the IoT applications [3]. The constrained devices are let to interact with the broad Internet using the related protocols.

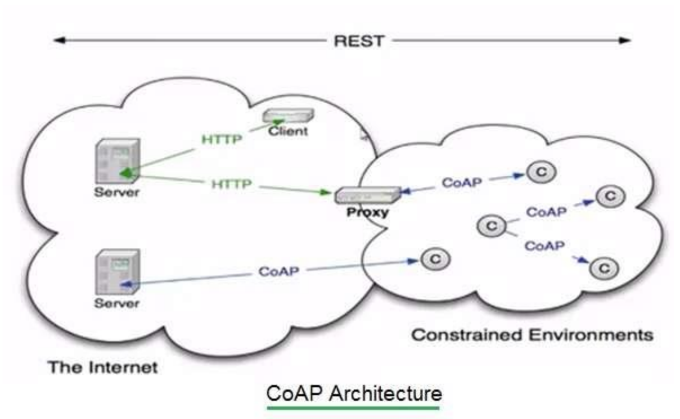


Fig 1:CoAP Architecture

CoAP is developed mainly for IoT and machine to machine (M2M) applications. The REST based CoAP architecture is shown in Fig. 1. The Universal Resource Identifiers (URIs) identifies the resources which are server controlled. The same methods which are used by HTTP such as GET, PUT, POST and DELETE methods are used by CoAP. CoAP is not same as HTTP but it includes a part of HTTP functions which are improved for using it in small embedded devices considering the low processing power and less energy consumption . In addition, to make it appropriate for IoT and M2M applications different mechanisms have been modified and certain recent ones are added to it.

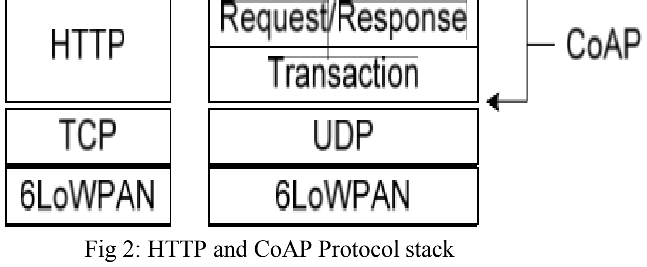


Fig 2: HTTP and CoAP Protocol stack

The transport layer is the main difference between HTTP and CoAP. HTTP depends on the Transmission Control Protocol (TCP). But the TCP's flow control system is not suitable for Low power and lossy networks and its overhead is very large for these fleeting transactions. And multicast is not supported by TCP. CoAP relies on the User Datagram Protocol (UDP) and consequently it has low overhead and it supports multicast. CoAP is systematized in two layers that is the transaction layer and the request /response layer as shown in Fig. 2. The message exchange or communication between end points is handled by the transaction layer[1]. There are four types of messages exchanged on this layer: They are Confirmable, Non-confirmable, Acknowledgment and Reset.

1. confirmable(CON): When the message is sent, the response is needed for this message type; the message receipt should be confirmed by the receiver.

2. Non-confirmable (NON): Here when the message is sent , the response is not required ,and hence there is no need to attest the message receipt.

3. Acknowledgment (ACK): Acknowledgment is obtained in response to the confirmable message which confirms the last reception .

4. Reset (RST): Whenever error is occurred in the message, or if it is not understandable or if the receiver does not want to communicate with sender, then this Reset message is sent.

The diffusion of requests and responses for the resource management is the work of request/response layer. A REST request is carried on a CON or NON message, whereas a REST response is carried on the ACK message. Because of these two layers CoAP provide reliability mechanisms even without using TCP as the transport protocol. Until the Acknowledgement message is obtained from the recipient a Confirmable message will be retransmitted. And it also enables the asynchronous communication. The multicast and congestion control is also supported by transaction layer. The CoAP was designed to maintain the message overhead as slight as possible and to restrict the application of fragmentation. HTTP has the great message overhead which indicate packet fragmentation and it leads to performance declination in the context of low power and lossy networks.

II. CoAP METHODS

CoAP employs the same methods as HTTP such as GET, PUT, POST, and DELETE methods and are used for resource manipulation. A response "405 Method Not Allowed" is generated as a reply to a unicast request. The same properties of HTTP such as safe and idempotent is exhibited by CoAP methods. The GET method is safe where as the GET, PUT and DELETE methods is executed in idempotent manner. The resource is signified by the URI embedded in the request which handles the confined body which is used for data processing and can also create new resources because of the POST method and hence this method is not idempotent [2]. Different CoAP methods are:-

A. GET

The GET method is accustomed to recollect the information of the resource which is identified by the request URI. As the response to this method, the success a 200 (OK) is obtained.

B. POST

The new subordinate resource will be created by the POST method below the parent URI which is requested by it to the server. A 201 (Created) response is sent after creating the resource on the server where as a 200 (OK) response code is sent on failure.

C. PUT

The resource which are identified by the request URI are updated or created by the POST method along with the confined message body. If the specified URI contains the message body, then it is considered as the changed version of a resource and the 200 (OK) response is received otherwise the new resource is created with that URI and the201 (Created) response is received. An error response code is sent if the resource is not created or modified.

D. DELETE

The resource which is identified by the requested URI is deleted using the DELETE method and the 200 (OK) response code is sent if the operation is successful .

III. METHODOLOGY

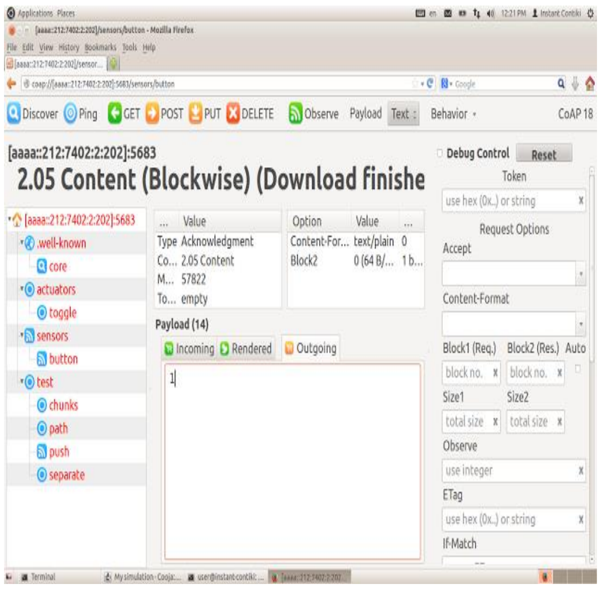
The performance of CoAP is compared with HTTP in order to evaluate its benefits for WSNs. The comparison considers the energy consumption of the server mote and the response time. The same experiments have been carried out on a CoAP client-server pair and on an HTTP client-server pair in order to evaluate the performance of these two protocols. The Contiki operating system and the Cooja simulator is used to carry out the simulations. Contiki being a memory-constrained systems which focuses on the low-power wireless Internet of Things devices and also an operating system for networked system.

The CoAP server is implemented by the sensor mote where 6LoWPAN or RPL is used for the network layer and CoAP for application layer running on Contiki . The same Tmote Sky platform which is used for CoAP server is accustomed for HTTP server implementation. And instead of CoAP server, Contiki is loaded with the HTTP server .The client queries the server at regular intervals and requests temperature and humidity in the CoAP and HTTP client-server transaction .

The experiments is carried out in two different client-server scenarios. The Zolertia Z1 motes is used for evaluation of the response with embedded temperature sensor. The Tmote Sky motes are used for evaluating the energy consumption embedded temperature and humidity sensors. The Tmote Sky is simulated using Cooja which is a simulator for Contiki based motes.

III. RESULTS AND ANALYSIS

Simulated the CoAP using the cooja simulator in contiki operating system. The firefox browser with cu plugin to open the ipv6 address and read the sensor values is shown by below two screenshots. In the Fig. 3 the browser sends the toggle value 1 for Red LED by selecting the POST button which is present in it, the RED LED glows in the Mote upon receiving which denotes that the node is taking the inputs remotely as shown in Fig. 4.



Applications Places
[aaaa::212:7402:2:202]/sensors/button - Mozilla Firefox
File Edit View History Bookmarks Tools Help
coap://[aaaa::212:7402:2:202]:5683/sensors/button
Discover Ping GET POST PUT DELETE Observe Payload Text Behavior CoAP 18
[aaaa::212:7402:2:202]:5683
.well-known
core
actuators
toggle
sensors
button
test
chunks
path
push
separate
Debug Control Reset
Token
Request Options
Accept
Content-Format
Block1 (Req.) Block2 (Res.) Auto
Size1 Size2
Observe
ETag
If-Match
Terminal
My simulation - Cooja:...
user@instant-contiki: ...

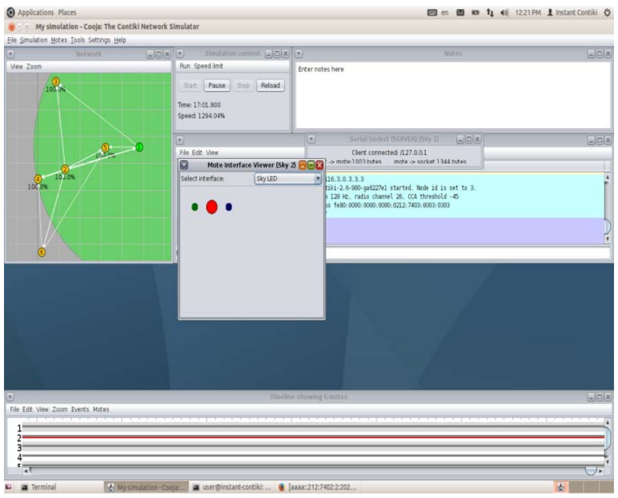
[aaaa::212:7402:2:202]:5683
2.05 Content (Blockwise) (Download finishe

...	Value
Type	Acknowledgment
Co...	2.05 Content
M...	57822
To...	empty

Option	Value	...
Content-For...	text/plain	0
Block2	0 (64 B/...	1 b...

Payload (14)
Incoming Rendered Outgoing
1

Fig. 3: Screenshot of firefox browser with cu plugin



Applications Places
12:21 PM Instant Contiki
My simulation - Cooja: The Contiki Network Simulator
File Simulation Motes Tools Settings Help
Network
View Zoom
Simulation control
Run Speed limit
Start Pause Step Reload
Notes
Enter notes here
Serial Socket (SERVER) (Sky 1)
Mote Interface Viewer (Sky 2)
Select interface: Sky LED
Timeline showing 6 motes
File Edit View Zoom Events Motes
Terminal
My simulation - Cooja:...
user@instant-contiki: ...
[aaaa::212:7402:2:202...

```
Time: 17:01.800
Speed: 1294.04%
Client connected: /127.0.0.1
-> mote:1003 bytes     mote -> socket 1344 bytes
116.3.0.3.3.3
tiki-2.6-900-ga6227e1 started. Node id is set to 3.
128 Hz. radio channel 26. CCA threshold -45
s fe80:0000:0000:0000:0212:7403:0003:0303
```

Fig 4: Screenshot of cooja simulator blinking red LED

A. Energy consumption

By using UDP as the transport protocol, the power consumption and battery lifetime improves with the decrease in header size of the packet in WSN's. The set of web service requests are simulated between a CoAP client and server and HTTP client and server for evaluating the performance improvement of CoAP compared to HTTP in terms of energy consumption. The graph of energy consumption of CoAP and HTTP server motes is shown in Fig. 5.

The Cooja simulator is used for testing and the server mote's energy consumption are calculated for a fixed client request interval time in two different operation modes. That is when the server mote receives packets, which is indicated as RX mode and the second mode is when the server mote transmits the packets which is indicated as TX mode. In HTTP the higher number of bytes are transferred and hence more time is spent by the motes for receiving, transmitting and processing packets which leads to higher power consumption in Rx and Tx. The energy consumed is very less while using CoAP compared to the one consumed while using HTTP when receiving the packets.

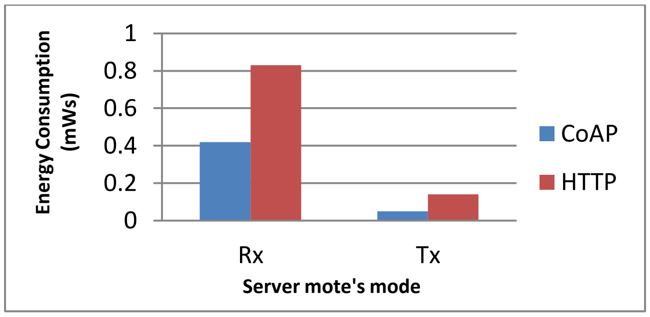


Fig 5:Energy Consumption of CoAP and HTTP server motes

In a second experiment, it is executed using the different value of the client request inter-arrival time such as 5,10,30 etc. By using these fixed length of simulation each experiment has a different total number of the generated client requests. The graph of energy consumption of the CoAP and HTTP server motes for different values of the client request inter-arrival time is shown in Fig. 6. This demonstrates that the higher the number of client-server transactions, the higher the difference in average energy consumption.

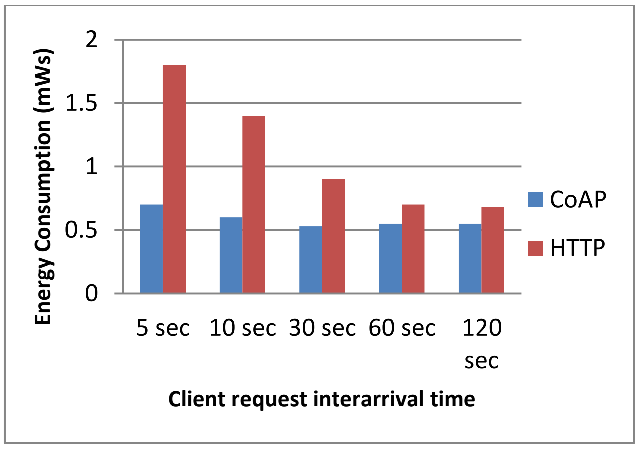


Fig 6: Energy consumption of the CoAP and HTTP server motes for different values of the client request inter-arrival time

B. Response time

The response time should be at low values when accessing sensor resources using web services. For some applications which have the strict requirements in terms of latency, their performance may be impacted by the response time using CoAP and HTTP client-server systems .This also shows how the response time and the packet header size is reduced by the use of UDP.

The response time is the time taken from the moment the client initiates the request till it receives the response with the payload. The simulation has been carried out on a client-server pair consisting of Zolertia Z1 motes. From the Fig. 7 the significant difference can be noticed in the response time for CoAP compared to HTTP.

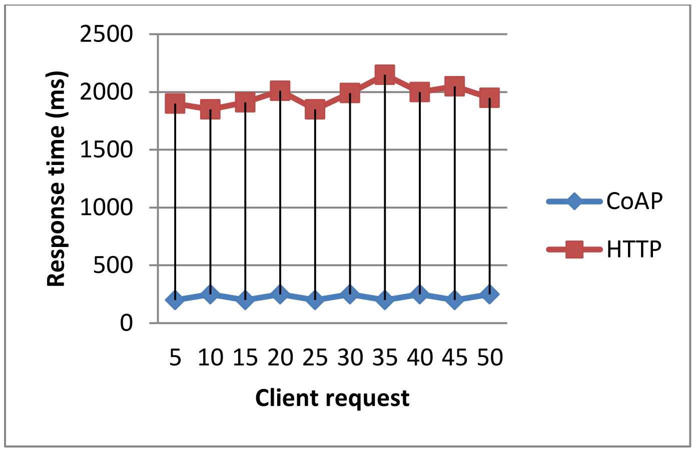


Fig 7: Response time of CoAP and HTTP

IV. CONCLUSIONS

CoAP provides the same methods as HTTP for the resource manipulation. It also supports additional functionalities such as multicast, asynchronous communication and subscriptions for IoT and M2M applications. Simulated CoAP, an IETF protocol providing low power and lossy networks with a RESTful architecture. CoAP uses UDP as the transport protocol and has compact packet overhead. This significantly affects the energy consumption of the motes and the response time. Here the Contiki operating system and the Cooja simulator is used for performing the simulations, and the results obtained shows that the CoAP is better in terms of energy consumption and response time compared to HTTP.

REFERENCES

[1] Walter Colitti, et.al., “Communication stacks: Constrained Application Protocol”, 2011.
[2] Manveer Joshi, et.al., “CoAP Protocol for Constrained Networks”, I.J. Wireless Technologies, 2015
[3] Shahid Raza, “Light weight security solutions for IoT”, IEEE ETFA, 2015.
[4] Tapio Leva, et.al. , “Comparing the cost efficiency of CoAP and HTTP in Web of Things applications”, Decision support systems, Vol. 63, July 2014
[5] Dan Garcia Carrillo ,et.al, “Light weight CoAP based Bootstrapping service for the Internet of Things”, Sensors 2016
[6] Ajit A. Chavan et. al., “Secure CoAP using Enhanced DTLS for Internet of Things”, IJIJCCE, Dec 2014.
[7] Antonio J. Jara, et.al, “The Internet of Everything through IPv6”, Journal of Wireless Mobile Networks, 2010.
[8] Thiemo Voigt, “Contiki Cooja Hands-on crash course”, CONET, 2016.