6LoWPAN Stack Model Configuration for IoT Streaming Data Transmission over CoAP

Khalid S. Aloufi [1]

[1] College of Computer Science and Engineering, Taibah University, Madinah, Saudi Arabia.

***Abstract*:** Different protocols have been developed for the Internet of things (IoT), such as the constrained application protocol (CoAP) for the application layer of the IPv6 over low-power wireless personal area networks (6LoWPAN) stack model. Data transmitted over 6LoWPAN are limited by the throughput and the frame size defined by IEEE 805.14.5 standards. Choosing the best configuration for data transmission involves a tradeoff between the application requirements, the constrained network configuration, the constrained device specifications and IoT application protocols. This paper provides an analysis of message size and structure recommendations for the 6LoWPAN stack model for different network topologies using CoAP. CoAP is a promising application protocol for the 6LoWPAN stack model because it can effectively manage the transmission required functionality in small header UDP packets compared to TCP packets. However, a data model is also required to realize an effective IoT model. While fragmentation and reassembly are supported by CoAP, they should be avoided for this type of model. As for any conceptual model, a high configuration between layers is mandatory. Additionally, the proposed message format is useful for semantic web of things application development and for WSN design and management.

***Keywords*:** Internet of Things (IoT); 6LoWPAN; CoAP; Semantic Web; Web of Things (WoT); Smart City; Home Automation**.**

1. Introduction

The number of constrained devices (CDs) connected to the Internet is expected to tremendously grow in the coming few years to approximately fifty billion devices [1]. To accommodate Internet growth, the Internet Protocol (IP) is moving from IPv4 towards IPv6. Consequently, different protocols are being developed in terms of supported data size or transmission criteria. Wireless sensor network (WSNs) have different components. CDs are the main components of the Internet of things (IoT). The IoT is effectively integrated with other Internet and web technologies and has opened a wide new domain of applications. For instance, the web of things (WoT) is a bridge model to connect things to the web. Additionally, the semantic web of things (SWoT) defines the connectivity between the semantic web and a WSN. For the IoT, different models have been developed, such as the IPv6 over low-power wireless personal area networks (6LoWPAN) stack model in 2007. Since then, 6LoWPAN has gone through different research and industry analyses and implementations [2] [3] [4] [5]. One of the main components of the 6LoWPAN model is the adaptation layer to exchange data between the IPv6 and 6LoWPAN networks, where the IP is compressed to operate with constrained data size [6] [3] [7]. The adaptation layer of 6LoWPAN has been developed to support the IP in CDs [3]. To integrate 6LoWPAN with IPv4 networks, a connection could be made through a gateway node at an edge node. The main components of the 6LoWPAN stack model, as shown in table 1, are IEEE 802.15.4, the adaptation layer, 6LoWPAN and application protocols, such as the constrained application protocol (CoAP) or message queuing telemetry transport (MQTT) [1] [8] [9].

For effective functionality, IoT systems require CDs with efficient power, constrained processing capabilities, constrained storage size, effective network design and configuration, a defined data model and an IoT application model. The requirements are summarized in figure 1. The 6LoWPAN stack model layers, as shown in table 1, should share a working configuration to work effectively to realize a defined quality of service (QoS) [10]. For example, CoAP should consider the mechanism of the 6LoWPAN adaptation layer and the data size defined by the data link layer (DLL) when sending data [3]. For CDs, the data memory and processing capabilities are limited.

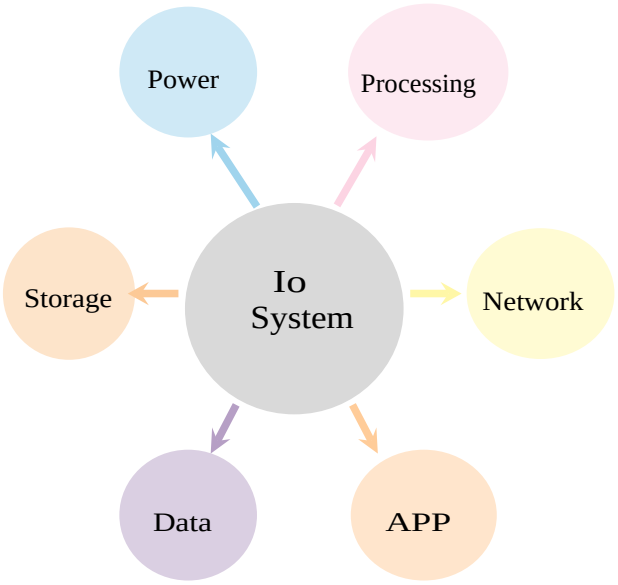

Figure 1. IoT System

One of the IoT application challenges is to deal with large data streams. As the data size increases, the required power and bandwidth also increase. In the IoT, generally the hardware and the network are constrained, however, its applications can go beyond the constrained environment through the integration of a system structure including advanced edge nodes or cloud capabilities.

In the application layer, since CoAP is transmitted over UDP, CoAP has different features that allow its IoT functionality, such as resource discovery and observation [1] [8]. Each CoAP data payload should have structured data formats, which will be detailed later, for integration with other

frameworks, such as the WoT or SWoT. CoAP has two types of implementation, either as a client or as a server. For data exchange between the CoAP client and server, the data payload contains the application data and the CoAP header. For example, when CoAP is used to observe temperature, any change is sent to the client by the server. These features result in large amounts of data and application functionality for smart cities and home automation. CoAP is effectively compatible with the 6LoWPAN model in terms of the application layer requirement if the CoAP payload fits in one 6LoWPAN frame; otherwise, different challenges arise, such as fragmentation and reassembly [10].

Table 1. Internet of Things Stack

Layer	OSI Layers	TCP/IP	6LoWPAN
7	Application	Application	CoAP, MQTT
6	Presentation		
5	Session		
4	Transportation	Transportation	UDP
3	Network	Internet	IPv6
			6LoWPAN Adaptation Layer
2	Data Link	Network Access	IEEE 802.15.4 MAC
1	Physical		IEEE 802.15.4 PYS

The design and configuration of IoT applications requires consideration of the data size for QoS. The model can handle data of large sizes, which require data fragmentation. However, the model layers, such as the network layer and DLL, are designed to work with a small data frame size. Different transportation and application protocols have been developed to work with such frame sizes and the constrained environment [10]. The use of general protocols, such as HTTP and TCP, or large frames sizes is not useful for IoT applications [11]. Aqeel-ur-rehman *et al* [12], Had mentioned how different IoT protocol are used for different IoT applications. It has mentioned that Up uses CoAP for field area network with medium support for security. Additionally, the use of the general practices for data transportation and application development is not recommended for such an environment, such as sending large amounts of data over a limited data frame and using fragmentation [11].

The objective of the study is to define the requirement of an IoT device in different WSN topologies and the structured data configuration for IoT applications over the 6LoWPAN stack model using CoAP. The next section explains the features of constrained devices and their role in IoT applications. Then, the following section presents the constrained application protocol to show the role of the application layer in the 6LoWPAN model. After that, the reason why fragmentation should be avoided is discussed. The relationship between the CD and different network topologies is shown in the next section because, in the network layer, configuration and analysis are required to show the CD requirements in different networks. Then, the general model is presented. All notes and recommendation from earlier sections regarding setting up the model configuration are given. Then, related work is discussed in the next section to show the relationship between the proposed analysis and research activity in the field. Finally, concluding remarks are presented, followed by the references.

2. Related Work

This research mainly evaluates the application layer of the IoT stack model; other studies have evaluated different layers of the IoT stack model, such as the 6LoWPAN layer [25]. There are different studies that have analyzed the suitability of the 6LoWPAN model for IoT applications [26] [10]. In the application layer, CoAP and other IoT protocols are suitable for IoT applications using the push mechanism of messages from the server to the client [26]. There are different models for using IoT data in semantic web applications [28] implying that the integration between the IoT model and the semantic web application model could be at the application layer. Adding more complexity to the model will result in difficult application development. The research objective here is to analyze the data and application requirements of the 6LoWPAN model. Previous studies proposed various development methods for the different layers of the 6LoWPAN model, such as developing the congestion-aware routing protocol (CoAP) to deal with periodic sensor data [29]. According to the requirement of some applications, a software defined network (SDN) is used to enhance the application of 6LoWPAN; a study by Miguel et al. presented a model called the software-defined 6LoWPAN wireless sensor network (SD6WSN) [30].

Intrusion detection and prevention (IDC) has been developed for integration in the 6LoWPAN model to protect CoAP traffic against security threats, such as denial of service (DoS) attacks [31]. Amanowicz and Krygier proposed a solution for the expected high traffic in the 6LoWPAN network, called the inter-session network coding mechanism, to reduce the traffic and consequently reduce energy consumption [32]. Another study proposed a framework for IoT system adaptability for application development and implementation, considering the 6LoWPAN mode [33]. Araújo et al. proposed a methodology to decrease device power consumption and consequently increase network live time by adapting a routing algorithm [34]. Abeele et al. implemented a proxy at the edge of the network to enhance WoT application over an IoT environment of limited resources [35]. Chen et al. presented a routing protocol for agricultural low-power and lossy networks (RPAL) using a scalable context-aware objective function (SCAOF) to adapt low-power and lossy networks (RPL) to the environmental monitoring of agricultural low-power and lossy networks (A-LLNs) [36]. Ludovici et al. presented a technique for forwarding and routing 6LoWPAN fragmented packets [36]. While in this research, fragmentation is not recommended, analysis and enhanced methodology is always helpful in the changing environment of new and continuously evaluated models in both industry and research.

One of the studies proposed by Ludovici et al. showed that 6LoWPAN fragmentation is suitable for a constrained environment; however, in a congested network, a proposed methodology called block wise outperforms the default fragmentation methodology of the 6LoWPAN model [38]. Oliveira et al. presented a security framework to enhance the 6LoWPAN model [39]. Finally, another study showed a model to interconnect CoAP with web applications using a web socket in CoAP [40].

3. Constrained Devices

Three classes of constrained devices are defined to work with low bandwidth and energy and lightweight protocols, as listed in table 2 [13]. Class 0 devices have limited connection capability and data storage. These devices can connect to the Internet and perform some functionalities by connecting to other devices, proxies, gateways, or servers. In some cases, Class 0 devices can use an OS for the IoT, such as by using the RIOT OS or customizing an OS to fit its specification. Class 1 devices can communicate in an IP network with security supported by constrained protocols, such as the constrained application protocol (CoAP) over UDP. Class 2 devices support Class 1 protocols and a full protocol stack because they have more code and data storage. Class 1 devices can replace Class 2 devices in cases where the developer can customize the OS or the protocol stack used. Each CD can have several resources, such as a few sensors, which depends on the application and its capability.

Table 2. Classes of Constrained Devices

Name data (RAM)	code (ROM)	IP Support	Name data (RAM)
Class 0, C0	10 KB	100 KB	No
Class 1, C1	~10 KB	~100 KB	Yes
Class 2, C2	~50 KB	~250 KB	Yes

CDs will have very limited RAM and ROM [14] [15]. The RAM or flash memory size should be sufficient for the OS and application data. The RAM size calculation considers the OS and the application data requirements as shown in Eq. (1). A CD has a RAM of APP_{RAM} and OS_{RAM} bytes reserved for code and the OS. The data sources are assumed to send data messages, limited to 127 bytes each [14]. The total data size is measured by Eq. (2), where n_1 is the number of connections to the CD, and n_2 is the number of cached messages. The CD should be ready to store and process the number of other connected CDs based on its location in the WSN as defined by n_1. The CD can cache message from connected sensors, or another CD as defined by n_2. Thus, the RAM required is the number of bytes required by the OS; the data shown in Eq. (2) define APP_{RAM} in Eq. (1).

$$M_{RAM} = OS_{RAM} + APP_{RAM} \quad (1)$$

$$APP_{RAM} = 127 * (n_1 + n_2) \quad (2)$$

The maximum total size of data a CD can handle is $x*127$ bytes, where x is the number of data messages the CD is sending or caching at a time. For example, for a 10 KB storage, the CD can have a maximum of 78 messages; for a 50 KB storage, the CD can have a maximum of 400 messages at a time.

Memory or ROM is the part of the CD that hosts the operating systems (OS) or the boot loader and application codes, as shown in table 3. The ROM size calculation consider the OS and the application data requirements as shown in Eq. (3). Different applications require different CD specifications. For example, the Contiki OS for constrained devices requires approximately 10 KB of RAM and approximately 30 KB of ROM, similar to a Class 1 device. CoAP implementation with the RIOT OS requires approximately 200 KB of ROM and 100 KB of RAM, similar to a Class 2 device. Each IoT OS can be customized to smaller memory and thus applied to a lower class.

$$M_{ROM} = OS_{ROM} + APP_{ROM} \quad (3)$$

Table 3. Oss for CDs

OS	RAM	ROM	Constrained
Contiki	10 KB	30 KB	Y
TinyOS [15]	< 1 kB	<4 KB	Y
RIOT	1.5 KB	5 kB	Y
mbed OS	512 KB	64 KB	Y
Android	>>*n* GB	>>*n* GB	N
Linux (Raspbian)	>>*n* GB	>>*n* GB	N
iOS	>>*n* GB	>>*n* GB	N

Table 3 shows different operating systems (OSs) that can be used in IoT ecosystems. However, the table shows that some types of OSs are mainly IoT OSs for different kinds of boards and architectures. The table shows that constrained OSs are useful for IoT devices in general ubiquitous and pervasive computing. Other OSs could be useful for edge nodes [17] [18] [19].

4. Constrained Application Protocol

This section analyses the data payload of the application protocol of the 6LoWPAN model. Based on table 1, the match between the model and the message structure is as shown in table 4. In the model, the IPv6 header is compressed to either 2, 12 or 20 bytes when the two devices are in the same network, a different network with a known prefix or a different network with an unknown prefix, respectively [7] [4] [14] [20].

The IP layer payload of the 6LoWPAN consists of the UDP header and payload, containing the CoAP header and payload. Application data are expected to be between 0 and 55 bytes in size for each frame, as shown in table 5. The CoAP header is assumed to be 11 bytes. The CoAP header has 4 bytes of a fixed header. Then, a token between 0 and 8 bytes is included. The option delta, length and value are between 0 and 5 bytes in size, and their sizes increase as the number of options increases. Then, a one-byte marker (0xFF) and the payload are included. The total size is assumed to be 11 bytes on average plus the payload.

Eleven bytes is enough to send the reading of, for example, a sensor along with its variables. If the CD is sending the temperature of a room or a location, then the payload should have enough bytes to send the temperature along with variables to define the specific location of the sensor. In general, the overhead is high because the maximum size is 127 bytes. However, the underlying protocol is designed for a constrained environment. Additionally, CoAP may require more than 11 bytes if data are required to have a specific structure, along with some metadata.

As shown in table 5, different cases are used to configure the CoAP data size. The IEEE 802.15.4 frame is 127 bytes, of which the MAC header takes up the first 25 bytes, followed by 21 bytes of link layer security, leaving approximately 81 bytes. An IPv6/6LoWPAN header requires 40/20/12/2, bytes leaving approximately 28/48/56/66 bytes. For UDP, there are some cases in which only 11 bytes are left when a 6LoWPAN mesh header is used. Additionally, there is a case in which only 53 bytes are left when no fragmentation or mesh headers are used for the 6LoWPAN header stack. For CoAP, there are from 0 to 55 bytes left.

		CoAP Header	CoAP Data	Application layer
	UDP Header	UDP Data		UDP
IP Header	IP Data			IP
Frame Header	Frame Data		Frame Footer	IEEE 802.15.4

Table 4. Message Format of the IoT Model

Different implementations of CoAP in constrained devices are shown in table 6. Each CoAP implementation is suitable for either regular or edge nodes. Some implementations are designed for regular client nodes. The CoAP implementations are available in different varieties of OSs and support different types of hardware. There also as well some implementations to manage the IoT network data in the cloud and it is commercial solution as shown in the end of the table. Some implementations are developed by specific Hardwar developer, which is an advantage of specific hardware users. In general, having too many implementations will require a design consistency when two different implementations exchange messages.

The application layer should have optimization and synchronization between the different layers of the IoT model. When the data frame is more than 127 bytes, data fragmentation is used [11]. When fragmentation is needed in CoAP, each fragment header requires an additional 5 bytes, except for the first fragment, which is 4 bytes [8] [22] [23]. Although fragmentation is supported by the DLL in IEEE 802.15.4, it is better to avoid fragmentation to save power and bandwidth and prevent wasted overhead [11].

The protocol overhead is the percentage of the header bytes in the total message bytes. IoT packets are small, so the overhead percentage is expected to be high in comparison with other network packets. Although fragmentation is supported by the model, the IoT application generally has small amounts of data to send, such as the temperature or air quality in a room.

5. Model based on Constrained Network

The WSN topology is usually either a star, tree or mesh topology. In any network topology, the CD is also resending the data of other sources. Consequently, the project cost may be affected by the network topology. In a star network, the main node requires some bytes of RAM as shown in Eq. (4), where n is the number of network connections. The required RAM for a mesh network is as shown in Eq. (5), where x is the number of nodes. In a tree network, the deeper the tree is, the higher the requirements. For a tree WSN, the amount of RAM required is as shown in Eq. (6), where g is the depth of the tree. In a tree WSN, as shown in figure 2, the edge node connects the nodes to outer connections. However, the WSN nodes are connected to the edge node using internal node connections. A WSN node should have more than one connection to the edge node using one of the hybrid WSN topologies, such as a partial mesh network topology, which is a mix between tree and mesh topologies. In this case, nodes in the tree topology have multiple connections to other nodes in the tree; consequently, the node has more than one option to connect to the edge node.

The design of a WSN involves different considerations; however, for the constrained model of the IoT, the location of the CD and the network design directly affect the CD specifications. The CD requires a specific RAM size depending on its application, its location and the topology of the WSN.

$$127 * n \quad (4)$$

$$127 * (x - 1) \quad (5)$$

$$OS + 127 * (2^{g+1} - 1) \quad (6)$$

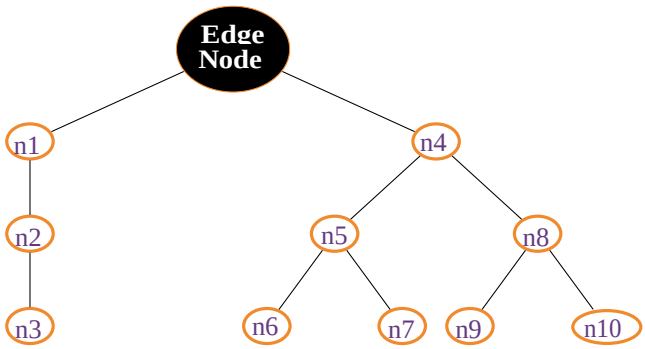

Figure 2. WSN Example

After defining the topology, the recommended configuration of the 6LoWPAN model detailed earlier is presented. The recommendations presented are mainly for the configuration of application data, CD specifications and the network topology. For the application data, CoAP is recommended as the application protocol for the model. Data fragmentation is not recommended when using CoAP. IoT application data should be small enough to fit in one CoAP data payload, between 0 and 55 bytes.

Regarding the CD specifications, the CD should have enough specifications to support the 6LoWPAN model. The OS selected should support the transportation, Internet and network access requirements with enough RAM and ROM in the CD. Based on the node location, the specifications can be defined, either for a regular node or a gateway node. The main node features for the model are the power, storage, processing unit and connectivity. The connectivity should support the IEEE 802.15.4 standard. A gateway node is used for forwarding and processing data. Since the gateway node is an intermediate node between regular nodes and the Internet, it should have an unlimited power source and reasonable processing capabilities. The gateway should have cached value, i.e., a CoAP client observer, of all the resources connected to the nodes, which will help increase the system performance.

Table 5, IEEE 802.15.4 Packet of 127 bytes

scenario	F H/D	SH	Mesh H	Fragment H	6LoWPAN H/D	UDP H/D	CoAP H/D
worst case 0	25/102	21/81	0/64	5/76	40/36	8/28	11/17
worst case 1	25/102	21/81	17/64	5/59	40/19	8/11	11/0
same network	25/102	21/81	0/64	5/76	2/74	8/66	11/55
known network	25/102	21/81	0/64	5/76	12/64	8/56	11/45
unknown network	25/102	21/81	0/64	5/76	20/56	8/48	11/37
case 1	25/102	21/81	0/64	0/81	20/61	8/53	11/43
scenario	F H/D	SH	Mesh H	Fragment H	6LoWPAN H/D	UDP H/D	CoAP H/D

The network topology should be a partial mesh network topology. The data are routed in the network all the way to the gateway nodes using the routing protocol for low-power and lossy networks (RPL) [24]. The throughput in IEEE802.15.4 is limited to 250 kbps [4] [8]. A basic mode is shown in figure 3, Where nodes are connecting to the Internet and advanced information processing is done through an Edge Node.

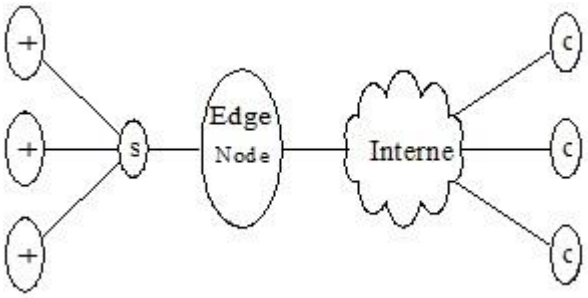

Figure 3. IoT Basic Model

For CoAP, there are four kinds of messages, including confirmable, non-confirmable, acknowledgement, and reset. Each message is either GET, POST, PUT or DELETE. The non-confirmable format is used for repeated values from a resource, such as sensor data [8]. Reset is sent to check the availability of the resources. If a resource is an actual resource, then GET is the suitable method to get the value of the resource. On the other hand, when the resource is virtual, then POST is used to set the resource value. CoAP and HTTP integration can close the gap between CoAP and web applications, which enables real applications for the WoT and SWoT. A piggybacked response is recommended to save bandwidth and energy. A CoAP server has the feature of responding to a "resource discovery" request with its list of resources. Hence, CoAP can be used for sharing resources over the web.

Table 6. Implementation of CoAP in Constrained Devices

Implementation	Client	Server	Language	Con.	OS	Note
Erbium	√	√	C	√	Contiki	official CoAP for the Contiki OS
libcoap	√	√	C	√	All	
tinydtls	√	√	C	√	LWIP/Contiki/TinyOS/RIOT	adds security to other implementations
TinyCoAP	√	√	C	√	TinyOS	
LibNyoci	√	√	C	√	LWIP/Contiki/TinyOS/RIOT	spun off from the SMCP project
microcoap	x	√	C	√	Arduino	-
cantcoap	√	√	C	√	Linux	
Lobaro CoAP	√	√	C	√	All	
MR-CoAP [20]	√	√	Java	√	VM	using IBM Mote Runner
Wakaama	√	√	C	√	All	LWM2M implementation

coap-node	√	x	JavaScript	√/x	Windows/Linus/macOS	LWM2M implementation
coap-shepherd	x	√	JavaScript	√/x	Windows/Linus/macos	LWM2M implementation
Californium	√	√	Java	x	JVM	
nCoAP	√	√	Java	x	JVM	
Leshan	√	√	Java	x	JVM	built on top of Californium
CoAP.NET	√	√	C#	x	Windows	
CoAPSharp	√	√	C#	x	Windows	
gen_coap	√	√	Erlang	x	Windows/Linux	
go-coap	√	√	Go	x	Linux/Windows	
node-coap	√	√	JavaScript	x	Windows/Linux/macOS	(node.js)
txThings	√	√	Python	x	All	
Aiocoap	√	√	Python	x	All	
Ruby-gem	√	x	Ruby	x	Linux/Windows/macOS	
David	x	√	Ruby	x	Linux/Windows/macOS	
coap-rs	√	√	Rust	√/x	All	
Copper	√	x	Browser-based		Linux/Windows/macOS	
iCoAP	√	x	Objective-C	x	iOS, OSX	
SwiftCoAP	√	√	Swift	x	iOS, OSX	
SPITFIREFOX	√	√	Java	x	Android	
Aneska	√	x	Python	x	Android	works with txThings
mbed client	√	√	C	√	mbed OS	ARM mbed, commercial and support IoT management
oneMPOWER	x	x	non	x	All	Commercial, Support IoT management
thethings.io	x	x	non	x	All	Commercial, Support IoT management

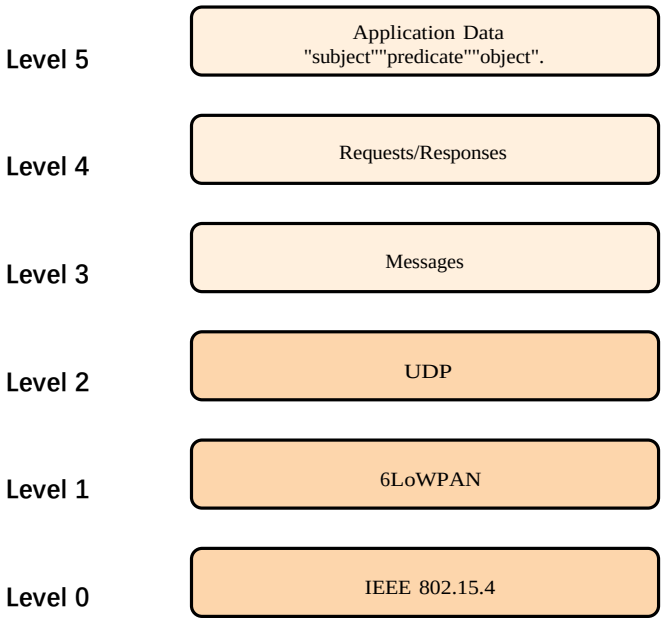

Figure 4. CoAP Model

The model provides semantic structured data, as shown in figure 4, where the data are presented using the resource description framework (RDF). The data are delivered in either request or response messages according to CoAP specifications. The resource can have more than one RDF triple in one transmission if the available CoAP payload is sufficient. If for each of the subjects, the predicate object will require a minimum of 3 bytes, then one triple will require a minimum of 16 bytes each. Two triples will require 32 bytes , etc. In this case, one message will not be able to have more than approximately 4 RDF triples. One example of an RDF triple is the temperature degree response message, which could be "temp""is""22.5◦C".

More triples will be involved if required for location or other parameters. In table 5, worst case 1 will accommodate no triples. If the CoAP header is the minimum size with only 6 bytes, meaning 4 bytes of the basic header, a 1-byte token and a 1-byte marker, then only 5 bytes are left, which will not be sufficient to send any structured data or meaningful parameters unless not stated in the response but taken from the request.

The purpose of the model is to enable utilization of structured response application data. Worst case 1 can have only one triple. The presented model cannot work without IPv6 compression unless carefully configured and tuned for the frame sections mentioned in table 5.

Referring to the previous argument, despite the high configuration requirements, CoAP is a promising application protocol for the IoT. CD specifications, OS requirements, 6LoWPAN stack model header compression and the structure data model are the main things to consider when using CoAP over the 6LoWPAN stack model.

6. Conclusion

IoT data are constrained data sent using a CD and a constrained network. The optimum specifications of a CD can be defined according to the performance evaluation of the constrained model. The objective of this study is to define the amount of CoAP data a CD can handle in each transmission. CoAP messages of the type observe requests consume large amounts of memory. However, the request size in bytes is unknown. In general, the RAM size as has been shown in different studies to directly affect the processing power and power consumption. This research shows the relationship between the memory requirements and data size, functionality and CD location.

The data generated from CDs in total are big data; however, each individual data frame has a few tens of bytes, as mentioned earlier. The data generated from a single CD are considered constrained because of their limited size according to the application protocol used, such as CoAP over 6LoWPAN. The data should be limited to the available CoAP payload. Further studies are required to consider an analysis of the trade-off between energy, processing and bandwidth. When the processing or throughput increases, the power consumption increases.

Hence, the application should design a system such that the processing is maintained at the gateway node. Nodes should only be used for reading and forwarding data. The resource data can be taken from the gateway rather than from a connection to the CD node. Additionally, the retransmission time out (RTO) and round-trip time (RTT) will be computed for general congestion control algorithms of CoAP.

The model is suitable for smart cities, home automation and any other applications where the data size is expected to be in the range mentioned earlier. Most smart city and home automation data are a type of reading or a signal of different sensors or parameters, such as temperature, average traffic wait time or expected time for events or resources.

The network topology should be a partial mesh network topology. The adaptation layer is the key element for network connectivity in the model because of the IP adaptation process and compression [7]. CoAP has been receiving much attention in the research community and from application developers for different OSs. Finally, this article gives recommendations for CoAP implementation in the 6LoWPAN stack mode.

References

[1] C. Bormann, A. P. Castellani and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," in IEEE Internet Computing, Vol. 16, No. 2, pp. 62-67, March-April 2012.

[2] Z. Shelby, Z. Bormann, "6LoWPAN: The Wireless Embedded Internet," Wiley Publishing, 2010.

[3] S. Chakrabarti, G. Montenegro, R. Droms, J. woodyatt, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) ESC Dispatch Code Points and Guidelines," RFC 8066, 2017.

[4] P. Thubert, J. Hui, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, 2011.

[5] G. Montenegro, C. Schumacher, N. Kushalnagar, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," RFC 4919, 2007.

[6] M. Crawford, "Transmission of IPv6 Packets over Ethernet Networks," RFC 2464, 1998.

[7] A. Ludovici, A. Calveras, M. Catalan , C. Gómez, J. Paradells, "Implementation and Evaluation of the Enhanced Header Compression (IPHC) for 6LoWPAN," EUNICE 09. Lecture Notes in Computer Science, Vol 5733. Springer, Berlin, Heidelberg, 2009.

[8] Z. Shelby, K. Hartke, C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, 2014.

[9] A. Banks, R. Gupta, "MQTT version 3.1.1 plus errata 01," O.S.I.A.E., 10 December 2015. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html, Accessed 2019.

[10] N. Ismail, R. Hassan, K. Ghazali, "A study on protocol stack in 6lowpan model. Journal of Theoretical and Applied Information Technology" Vol. 41, No. 2, pp. 220-229, 2012.

[11] J. Olsson, "6LoWPAN demystified," T.I., http://www.ti.com.cn/cn/lit/wp/swry013/swry013.pdf, Accessed 2014.

[12] A. ur Rehman, S. ur Rehman, I. Uddin Khan, M. Moiz, S. Hasan, "Security and Privacy Issues in IoT," International Journal of Communication Networks and Information Security (IJCNIS), Vol 8, No 3 ,2016.

[13] C. Bormann, M. Ersue, A. Keränen, "Terminology for Constrained-Node Networks," RFC 7228, 2014.

[14] G. Montenegro, J. Hui, D. Culler, N. Kushalnagar, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," RFC 4944, 2007.

[15] J. W. Hui and D. E. Culler, "Extending IP to Low-Power, Wireless Personal Area Networks," in IEEE Internet Computing, Vol. 12, No. 4, pp. 37-45, July-Aug. 2008.

[16] P. Gburzyński, E. Kopciuszewska, "On Rapid Development of Reactive Wireless Sensor Systems," CZOTO, Vol. 1, No. 1, pp. 574-582, 2019.

[17] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, C. Lin, "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment," in IEEE Access, Vol. 6, pp. 1706-1717, 2018.

[18] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," in IEEE Internet of Things Journal, Vol. 3, No. 6, pp. 854-864, Dec. 2016.

[19] S. Yi, C. Li, Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues,". Proceeding of the Workshop on Mobile Big Data, NY, USA; Mobidata '15, pp. 37–42. 2015.

[20] S. A. B. Awwad, C. K. Ng, N. K. Noordin, B. M. Ali and F. Hashim, "Second and subsequent fragments headers compression scheme for IPv6 header in 6LoWPAN network," Seventh International Conference on Sensing Technology (ICST), Wellington, pp. 771-776, 2013.

[21] T. Kramp, M. Baentsch, T. Eirich, M. Oestreicher, I. Romanov, "The IBM Mote Runner," ERCIM , 2009.

[22] A. Ludovici, C. Augé, J. Casademont, J. "Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network," Sensors, Vol. 11, No. 1, pp 992–1008, 2011.

[23] J. Hui, D.C.; Chakrabarti, S. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture. IPSO, 2009.

[24] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks.," . RFC 6550, 2012.

[25] R. Garg, S. Sharma, "A study on Need of Adaptation Layer in 6LoWPAN Protocol Stack,". International Journal of Wireless and Microwave Technologies, Vol. 7, pp. 49–57, 2017.

[26] C. Devasena, "IPv6 Low Power Wireless Personal Area Network (6LoWPAN) for Networking Internet of Things (IoT) – Analyzing its Suitability for IoT," Indian Journal of Science and Technology, Vol. 9, 2016.

[27] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core,". RFC 3920, 2004.

[28] M. Ruta, F. Scioscia, E. Sciascio, "Enabling the Semantic Web of Things: Framework and Architecture," IEEE Sixth International Conference on Semantic Computing, pp. 345–347, 2012.

[29] Bhandari, K.S.; Hosen, A.S.M.S.; Cho, G.H. "CoAR: Congestion-Aware Routing Protocol for Low Power and Lossy Networks for IoT Applications," Sensors, Vol. 18, No. 11, 2018.

[30] Miguel, M.L.F.; Jamhour, E.; Pellenz, M.E.; Penna, M.C. "SDN Architecture for 6LoWPAN Wireless Sensor Networks," Sensors, Vol. 18, No. 11, 2018.

[31] J. Granjal, JM. Silva, N. Lourenço, "Intrusion Detection and Prevention in CoAP Wireless Sensor Networks Using Anomaly Detection," Sensors, Vol. 18, 2018.

[32] M. Amanowicz, J. Krygier, "On Applicability of Network Coding Technique for 6LoWPAN-based Sensor Networks," Sensors, Vol. 18, 2018.

[33] C. Martín, J. Hoebeke, J. Rossey, M. Díaz, B. Rubio, F. Van den Abeele, "Appdaptivity: An Internet of Things Device-Decoupled System for Portable Applications in Changing Contexts," Sensors, Vol 18, 2018.

[34] H. Araújo, R. Filho, J. Rodrigues, R. Rabelo, N. Sousa, J. Filho, J. Sobral, "A Proposal for IoT Dynamic Routes Selection Based on Contextual Information," Sensors, Vol. 18, 2018.

[35] F. Van den Abeele, I. Moerman, P. Demeester, J. Hoebeke, "Secure Service Proxy: A CoAP(s) Intermediary for a Securer and Smarter Web of Things," Sensors, Vol. 17, 2017.

[36] Y. Chen, J. Chanet, K. Hou, H. Shi, G. de Sousa, "A Scalable Context-Aware Objective Function (SCAOF) of Routing Protocol for Agricultural Low-Power and Lossy Networks (RPAL)," Sensors, Vol 15, No. 8, pp. 19507–19540, 2015.

[37] A. Ludovici, A. Calveras, J. Casademont, "Forwarding Techniques for IP Fragmented Packets in a Real 6LoWPAN Network." Sensors, Vol 11, No. 1, pp. 992–1008, 2011.

[38] A. Ludovici, P. Marco, A. Calveras, K. Johansson, "Analytical Model of Large Data Transactions in CoAP Networks," Sensors, Vol. 14, No. 8, pp. 15610–15638, 2014.

[39] L. Oliveira, J. Rodrigues, A. de Sousa, J. Lloret, "A Network Access Control Framework for 6LoWPAN Networks," Sensors, Vol. 13, No. 1, pp. 1210–1230, 2013.

[40] A. Ludovici, A. Calveras, "A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications," Sensors, Vol. 15, No. 1, pp. 1217–1244, 2015.