Performance Analysis of Constrained Device Virtualization Algorithm

Shariq Haseeb, Aisha Hassan A. Hashim, Othman O. Khalifa, Ahmad Faris Ismail

Abstract*: *Internet of Things aims to automate and add intelligence into existing processes by introducing constrained devices such as sensors and actuators. These constrained devices lack in computation and memory resources and are usually battery powered for ease of deployments. Due to their limited capabilities, the constrained devices usually host proprietary protocols, platforms, data formats and data structures for communications and therefore, are unable to communicate with devices from different vendors. This inability leads to interoperability issues in Internet of Things which, is in fact against the spirit of Internet of things which, envisions interconnection of billions of devices and hence, results in an isolated, vendor-locked and close-loop deployments of IoT solutions. Various approaches have been made by the industry and academia to resolve the interoperability issues amongst constrained devices. However, majority of the solutions are at different layers of the communication stack but do not provide a holistic solution for the problem. In more recent research, there have been theoretical proposals to virtualize constrained devices to abstract their data so that its always available to applications. We have adopted this technique in our research to virtualize the entire Internet of Things network so that virtual TCP/IP based protocols can operate on virtual networks for enabling interoperability. This paper proposes the operations of the Constrained Device Virtualization Algorithm and then simulates it in CloudSIM to derive performance results. The paper further highlights open issues for future research in this area.

Keywords* : *Internet of Things, virtualization, software defined networks, FoG computing.

I. INTRODUCTION

The aspirations of IoT as defined by the European Research Cluster is a technology that allows people and things to be connected anytime, anyplace, with anything and anyone, ideally using any path/network, and any service [1]. Technical requirements based on this definition imply that IoT has very high requirement for interoperability between devices or things.

A "thing" in IoT is basically an autonomous, physical or digital object with sensing or actuation capability depending on the application. These objects are designed to bridge the connection between the software domain and the physical world [2].

To date, there are many IoT architectures designed to incorporate these constrained IoT devices. Some architecture is for real-time and batch processing [3], some are edge network based [4], some architectures exploit the resources of the Cloud [5] and some of the more advanced architectures exploit the capabilities of data analytics and AI within the IoT network [6], [7]. However, they do not deviate much from the typical IoT reference architecture shown in Fig. 1.

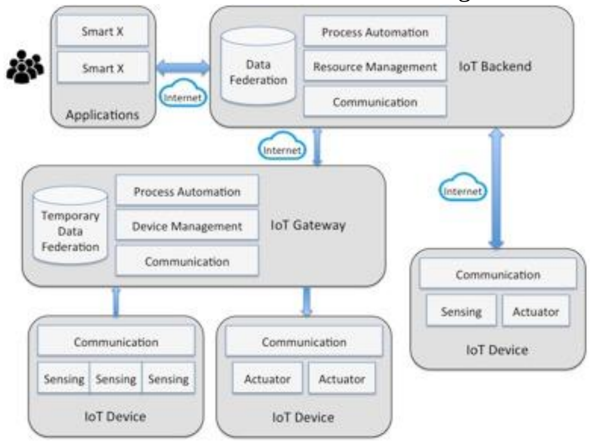


Fig. 1.A typical IoT reference architecture

It can be seen in Fig. 1, that the bottom most layer of the architecture is where the constrained IoT devices reside. These IoT devices have communication capabilities to either communicate with a gateway or directly with the IoT backend for data exchange. An IoT gateway is typically a computing device that is able to communicate with the constrained IoT devices. Some gateways have the ability to store and process data, perform simple automation tasks and manage constrained IoT devices. The IoT backend could reside locally within the IoT network or on a Cloud datacenter. It is responsible for data aggregation, IoT rules and process automation and also serves as a standardized interface to the smart IoT applications.

The typical IoT reference architecture has been implemented in various IoT deployments around the world. Even though, architecturally the IoT deployments are similar, the protocols used within the architecture is very different due to the nature of the constrained IoT devices. In fact, there are more than 300 IoT backends available from different vendors [8] to cater for the heterogeneous nature of the IoT devices.

Beyond just architectural requirements, heterogeneity in IoT devices leads to interoperability issues that have been presented in various literature and standards documents [9]. Interoperability issues are responsible for about 17% of added cost in IoT deployments [10].

Revised Manuscript Received on February 06, 2020.

***** Correspondence Author

Shariq Haseeb*, Faculty of Engineering, International Islamic University Malaysia, Gombak, Malaysia. Email: shariqkhan1@yahoo.com

Aisha Hassan A. Hashim, Faculty of Engineering, International Islamic University Malaysia, Gombak, Malaysia. Email: aisha@iium.edu.my

Othman O. Khalifa, Faculty of Engineering, International Islamic University Malaysia, Gombak, Malaysia. Email: khalifa@iium.edu.my

Ahmad Faris Ismail, Faculty of Engineering, International Islamic University Malaysia, Gombak, Malaysia. Email: faris@iium.edu.my

In order to solve interoperability issues in IoT, we are proposing a new virtualization architecture for constrained IoT devices coupled with constrained device virtualization algorithms. We will then simulate the operations of the algorithm and elaborate on the obtained results to evaluate the feasibility of the new algorithm.

II. DEFINITION OF A CONSTRAINED DEVICE

Constrained IoT devices are typically end nodes in an IoT network. They have the capability to sense one or more parameters from the environment and some constrained devices also have the ability to actuate their physical environments to achieve a specialized purpose [11].

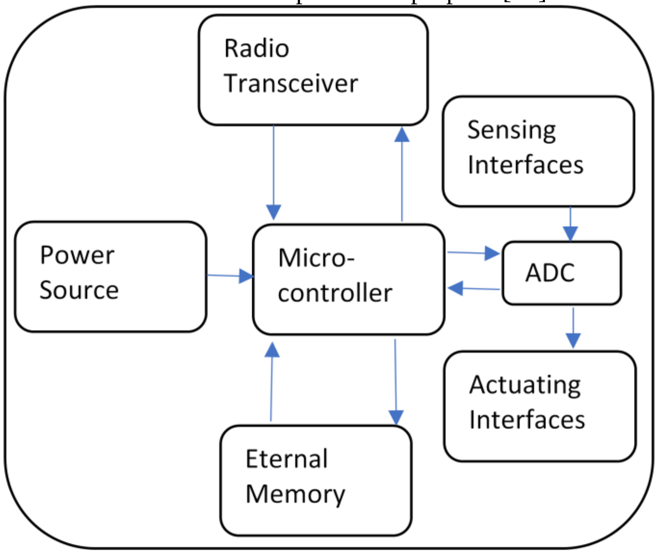


Fig. 2. A typical constrained device architecture

A typical constrained device architecture is shown in Fig. 2. It consists of a micro-controller that functions as a central unit powered by a power source. It is responsible for receiving, transmitting and processing sensor data through and Analogue to Digital Converter (ADC). If required, micro-controller can also trigger other devices through the actuating interface available on the constrained device.

Architecturally, constrained IoT devices can communicate via a gateway or sometimes communicate directly to backend IoT Cloud based platforms for data transfer. More often than not, they operate in lossy wireless conditions employing protocols such as Bluetooth Low Energy (BLE), 802.15.4 (6LoWPAN, Zigbee, Thread, WirelessHART etc.) and more recently Low Power Wide Area Network (LPWAN). They are also mostly battery powered for ease of remote deployments.

Constrained IoT devices have following characteristics:

- Battery powered energy source: in order to conserve energy, many constrained IoT devices follow a sleeping schedule that allows them to perform a transaction such that transmit data or check status and then enter hibernation state before waking up to perform next transaction.
- Limited processing ability: in order to minimize cost and manage power constraints, most constrained IoT devices employ only a limited processing capability that makes them highly specialized.
- Limited memory size: to manage memory restriction, constraint IoT devices limit the size of state and buffers. Hence, they deploy only simple codes and limited communication stacks.
- Limited capability: to perform tasks within the constraints, these devices are highly specialized in nature.
- Vulnerable radio conditions: these devices usually operate under low throughput and lossy network conditions.
- Highly asymmetric link characteristics: in order to maintain specialization, their uplink and downlink conditions are usually not symmetrical. In other words, a sensor is optimized to send while an actuator is specialized to receive.
- No direct human interaction: these devices are designed to be deployed in remote conditions and hence, have very limited human interaction ability for troubleshooting or management. Most devices even compromise on a user interface since it is not required.
- Physical size and cost: these devices are ideally designed to be small and cheap so that they can be easily deployed and replaced when needed.

In order to simplify and understand the constrained IoT devices, Internet Engineering Task Force (IETF) has published and RFC 7228 that categorizes these constrained IoT devices into three simple classes as follow [12]:

- Class 0: these devices are considered to be very constrained and tend of have constraints on memory and processing capabilities. Typically, their memory size is less than 10KB and flash memory is below 100KB. These devices can not directly communicate over the Internet and have to be connected to gateways or other intermediate devices for Internet communication. Based on a blog [13] that conducted a study on these type of devices, the most minimal network stack takes up most of the resources of class 0 devices and no other protocols can be loaded onto the device.
- Class 1: these devices are quite constrained in code space and processing abilities. They cannot easily communicate over the Internet using TCP/IP protocols such as using HTTP, Transport Layer Security (TLS), and related security protocols and XML-based data representations. However, they can employ specially designed low power IoT stacks such as User Datagram Protocol (UDP), CoAP, light weight security protocols like Datagram Transport Layer Security (DTLS) for communication. In order to enable full Internet communication, an intermediate device such as a gateway is recommended [13].
- Class 2: these devices are less constrained and are able to support similar protocol stack to those supported by a mobile phone, notebook or a server [12]. However, they still need to deploy lightweight protocols and energy-efficient algorithms to operate efficiently. It is recommended to use this type of devices to promote interoperability in IoT [13].

- Beyond Class 2: energy constraints may exist on these types of devices. However, they are not constrained by protocol and are able to communicate over the Internet using the full communication stack.

This paper focusses mainly on class 0 and class 1 devices that are widely deployed in the IoT applications.

III. INTEROPERABILITY ISSUES DUE TO CONSTRAINED DEVICES

Constrained devices have limited capabilities and hence host their own flavor of communication protocols leading to heterogeneous nature of such devices. Below are reasons for this heterogeneity:

- Connectivity Issues: constrained IoT devices may host multiple communication interfaces. These interfaces operate on different frequency bands, use different Media Access Control (MAC) and Internet Protocol (IP) for communication [14]. If two devices don’t use the same communication interface, they would never be able to communicate. This is the main reason for constrained IoT device interoperability issues.
- Multi-Vendor Devices: although standardization bodies propose recommendations for developing IoT devices, most vendors don’t follow these recommendations because they drive up cost and does not allow product differentiations [8]. Hence, devices originating from different vendors tend to be closed looped.
- Legacy IoT Deployments: many closed-loop vertical smart services like building management, home automation, vehicle tracking, personnel tracking, etc. have existed even before the term ‘IoT’ was defined [15]. The problem with these solutions is that they were never meant to interoperate hence, they do not deploy protocols for Internet based communication.
- Multiple IoT Platforms: more than 300 IoT platforms have been presented in the literature [16]. Each of these platform implement their own mechanisms for data abstraction and manipulation. Hence, rather than promoting interoperability, they tend to lock down the IoT solutions to their own domain.
- Multiple Syntax: each model or make of IoT device generates data in a particular format. Even if two devices generate exact same data but in different order of format, they will not be able to communicate with each other. This usually happens because data formats are dictated by the applications [17].
- Data Semantics: IoT device data is not coupled with explanations about the data. This leads to almost no understating of messages between IoT systems. Furthermore, many times different unit systems may be used across devices from different regions and hence interoperability between the devices in not possible [18].

IV. APPROACHES TO TACKLE INTEROPERABILITY ISSUES

To date, there have been various attempts by the industry and academia for solving the interoperability issues in IoT. These approaches have been classified across following major areas:

- Architecture and Platform Standards: platform consolidations have been the central focus of most research. The aim of this exercise is to consolidate IoT platform from similar use case and industry to propose common platform. The rationale behind this is to allow devices from the same industrial use to be able to communicate with each other [19]. Although this is a good initiative, it only resolves the interoperability issues within a single industry and does not offer a holistic solution to the interoperability problem.
- MAC Layer Consolidation: MAC layer related initiatives focus on limiting the number of communication protocols through standardization of MAC layer for IoT use [20]. In our opinion, this is not an ultimate solution to the interoperability issue because firstly the choices of communication will be greatly limited and secondly it does not solve the problem even if only two types of MAC protocols are allowed.
- IP Layer Approach: one of the other proposals is to use IP protocol for communication. Although this is a fantastic idea because this allows IoT devices to communicate over the Internet but the only problem with this approach is that all existing IoT devices need to be upgraded [21]. Furthermore, the constrained devices will not be able to support the complete TCP/IP communication stack.
- Infrastructure Approach: at the infrastructure layer, technology such as Software-defined Networks, Fog Computing and sensor virtualization have been employed to achieve interoperability. SDN approach to achieving interoperability is by splitting the control and data planes in SOs. It can be observed from the research in [22] that SDN allows different IoT device, using completely different protocols and connected to completely different networks can communicate with each other over common IPv6 protocol. SDN is able to achieve this by abstracting data from the device and since data is not constrained by protocol, interoperability can easily happen [23]. However, this approach only abstracts the data from the devices and does not cater of the operations of the device which, is needed for long term network maintenance and device manageability.

After evaluating all the approaches, we believe that the sensor virtualization approach can be expanded further to incorporate protocol virtualizations to offer a complete solution for achieving interoperability in IoT deployments.

V. CONSTRAINED DEVICE VIRTUALIZATION ARCHITECTURE

Proposed IoT architecture leverages on the existing architecture and the lessons learned from the software defined telecommunication architecture.

The modified architecture is necessary for hosting virtual constrained devices. The new architecture is shown in Fig. 3 where, the IoT device could simply be a dumb device with purely a communication interface coupled with sensing or actuating capabilities. They could either directly connect to the IoT backend or connect with the help of an IoT Gateway. The IoT gateways would also be lightweight because they only need to maintain a device registry to know the IoT devices connected to it. The gateways also need to host two communication stacks for connecting to the IoT devices and the IoT backend.

The real changes have to be made on the IoT backend that has to host more capabilities compared to the traditional IoT backend. The IoT backend would typically reside on the Cloud or a Fog computing node if Fog based topology is employed. This is to take advantage of the virtually unconstrained resources available in the Cloud or Fog infrastructure. The IoT backend would need to support device virtualization where each dumb IoT device would be represented by a virtual IoT device. The virtual IoT device emulates the physical constrained device capabilities but is represented as a software code in the IoT backed.

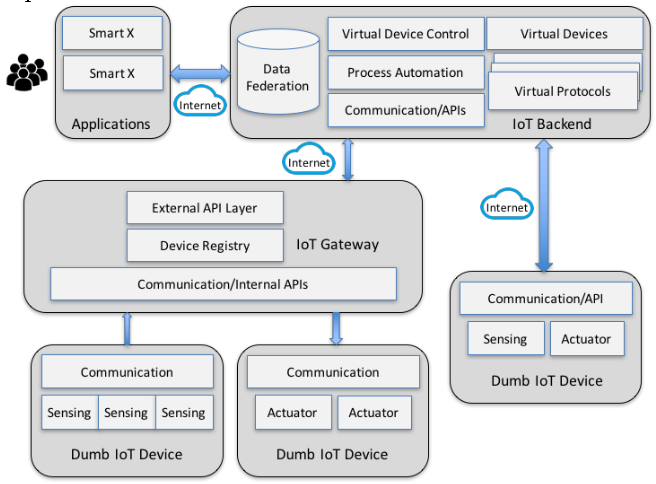


Fig. 3. Virtualized IoT network architecture

VI. CONSTRAINED DEVICE VIRTUALIZATION ALGORITHM

The first steps in this algorithm is to virtualize the constrained IoT devices. In order to virtualize a physical constrained device, it is important to identify the most important attributes of a device that needs to be abstracted. The critical parameters which define a physical constrained device are as follows:

- Device ID: a unique device ID to identify it on the IoT network. This is could follow the standard MAC address, or any incremental addressing format configured by the device manufacturer.
- IoT Device IP (Optional): this is the IP address of the constrained device in its current network if the device supports IP protocol.
- Gateway ID: a unique gateway ID to identify it on the IoT backend. This is could follow the standard MAC address, or an incremental addressing format configured by the device manufacturer.
- Gateway IP (Optional): this is the IP address of the gateway in its current network.
- Device Architecture (Optional): the current hardware specifications of the device such as processor, memory, battery, communication interface etc.
- CPU Utilization (Optional): the CPU utilization of the IoT device as a data stream to the IoT backend.
- Memory Utilization (Optional): the memory utilization of the IoT device as a data stream to the IoT backend.
- Battery Level (Optional): the battery utilization of the IoT device as a data stream to the IoT backend.
- Data Stream: data and its attributes that the IoT device is generating.
- Service Stream: types of services offered by the IoT device.
- Status Stream: current status of the IoT device that could be online, offline, asleep etc.
- Network Stream: network related parameters gathered by the physical constrained device.

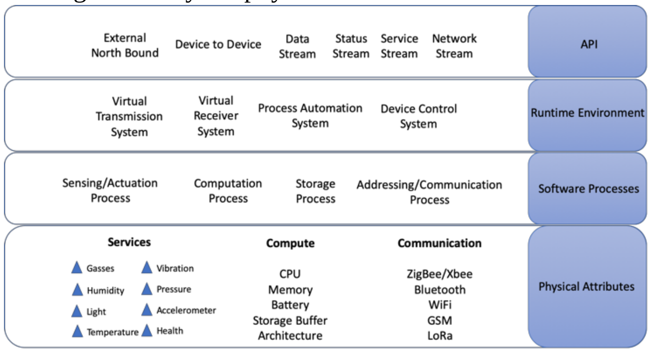


Fig. 4. System architecture of virtual constrained device

With all the known attributes, a software implementation of the physical constrained device can be developed based on the proposed system architecture shown in Fig. 4. Where the physical attributes layer represents the services, compute-ability and communication abilities of the physical constrained device. This layer also forms the data point that defines the ability of the virtualized constrained device. The software processes layer contains snippets of software codes that don't need to execute in real-time but can be executed when required. This layer abstracts the computation intensive behavior of a physical constrained device. The runtime environment layer hosts the code snippets and processes that are always running. These processes are related to the transmission, receiving, periodicity of data and controllability of the virtual device. They are designed to emulate the physical constrained device abilities. The final layer is the API layer that forms the entry point of interfacing with the virtual constrained device. Some of these APIs could be periodic and stream based such as data, service, status and network streams. While, others could be triggers through applications or other devices within the network.

In order for a virtual constrained device to appear to operate like a physical constrained device, certain IoT backend adaptations are required.

The first of those changes is the creation of a virtual interface. As soon as a virtual constrained device is activated, the IoT backend host (Cloud or Fog) would need to create a virtual network interface associated to its own physical interface. This is critical for a virtual device to be backwards compatible with the network elements.

The second important step for the IoT backend is to request and assign an IP address from the address broker within the network. This will allow the virtual interface to be addressable on the network.

The final step is for the IoT backend to maintain a binding of an IP address to the virtual interface to the virtual device. This bookkeeping step is critical to determine when a traffic is meant for the virtual constrained device and it will also help in clean-up process when the virtual constrained device is terminated or migrates to a different network. The flow of the steps is sown in Fig. 5.

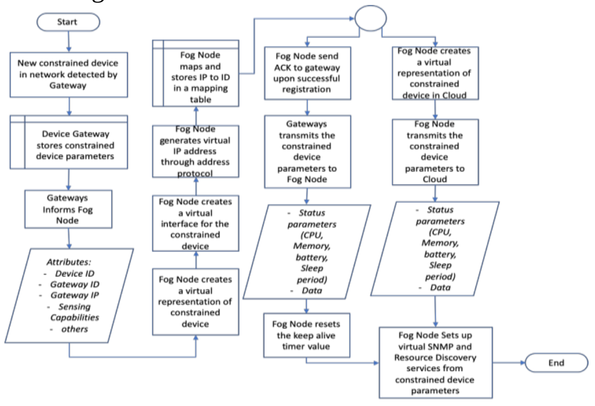

Start
New constrained device in network detected by Gateway
Device Gateway stores constrained device parameters
Gateways Informs Fog Node
Attributes:
- Device ID
- Gateway ID
- Gateway IP
- Sensing Capabilities
- others
Fog Node creates a virtual representation of constrained device
Fog Node creates a virtual interface for the constrained device
Fog Node generates virtual IP address through address protocol
Fog Node maps and stores IP to ID in a mapping table
Fog Node send ACK to gateway upon successful registration
Gateways transmits the constrained device parameters to Fog Node
- Status parameters (CPU, Memory, battery, Sleep period)
- Data
Fog Node resets the keep alive timer value
Fog Node creates a virtual representation of constrained device in Cloud
Fog Node transmits the constrained device parameters to Cloud
- Status parameters (CPU, Memory, battery, Sleep period)
- Data
Fog Node Sets up virtual SNMP and Resource Discovery services from constrained device parameters
End

Fig. 5. Constrained device virtualization algorithm

VII. PERFORMANCE PARAMETERS

The variable parameters of the simulation are configured as follows:

Parameters	Characteristics
Number of constrained devices	5, 10,15,20,25,30,35,40,45,50
Number of Fog Gateways	1
Constrained Device Virtualization Mode	• Hybrid Cloud and Fog • On Cloud only • On Fog only

The fixed parameters of the simulation are configured as follows:

Parameters	Characteristics
Total Available Network Bandwidth	Unlimited
Network Link Characteristics	Constant
Packet Drops Characteristics	Blocking with no packet drop
Packet Loss Characteristics	Nil
Simulation Time	10800 sec (3 hours)
Resource Management Interval	100 sec
Network Device	Intel x86
Architecture	
Network Device OS	Linux
Network Device Time Zone	+8.0 (Malaysia)
Virtual Machine Manager	Xen

The device characteristics are set according to the mode of operation as follows:

Device	Characteristics
Cloud Datacenter	• CPU Cores: 16 cores • MIPS: 44800 MHz • RAM: 40000 MB • MIPS utilization cost: 0.01 • Hierarchical Level: 0 • Busy Power: 103/core • Idle Power: 83.25/core
Cloud VM Instance	• CPU Cores: 1 core • MIPS: 2800 MHz • RAM: 4000 MB • MIPS utilization cost: 0.01 • Hierarchical Level: 0 • Busy Power: 103/core • Idle Power: 83.25/core
Internet Proxy (Hybrid Cloud and Fog mode)	• CPU Cores: 1 core • MIPS: 2800 MHz • RAM: 4000 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: 107.339/core • Idle Power: 83.433/core
Internet Proxy (Cloud only mode)	• CPU Cores: 0 core • MIPS: NA MHz • RAM: 0 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: NA/core • Idle Power: NA/core
Internet Proxy (Fog only mode)	• CPU Cores: 16 core • MIPS: 44800 MHz • RAM: 40000 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: 107.339/core • Idle Power: 83.433/core
Fog Gateway (Hybrid Cloud and Fog mode)	• CPU Cores: 1 core • MIPS: 2800 MHz • RAM: 4000 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: 107.339/core • Idle Power: 83.433/core
Fog Gateway (Cloud only mode)	• CPU Cores: 0 core • MIPS: NA MHz • RAM: 0 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: NA/core • Idle Power: NA/core
Fog Gateway (Fog only mode)	• CPU Cores: 1 core • MIPS: 2800 MHz • RAM: 4000 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 1 • Busy Power: 107.339/core • Idle Power: 83.433/core
Constrained Device	• CPU Cores: 1 core • MIPS: 500 MHz • RAM: 1000 MB • MIPS utilization rate: 0.0 • Hierarchical Level: 3 • Busy Power: 87.53/core • Idle Power: 82.44/core

The link characteristics are set as follows:

Links	Characteristics
Cloud and Internet Proxy (Typical Internet Link)	• Downlink Bandwidth: 10000Mbps • Uplink Bandwidth: 100Mbps • Link latency: 100ms
Network Proxy and Fog Gateway (Typical Intranet Link)	• Downlink Bandwidth: 10000Mbps • Uplink Bandwidth: 10000Mbps • Link latency: 2ms
Fog gateway and Constrained Device (Typical Wireless Sensor Network Link)	• Downlink Bandwidth: 10000Mbps • Uplink Bandwidth: 10000Mbps • Link latency: 15ms
Constrained Device and its sensors/actuators (Typical Bus Link)	• Link latency: 1ms

The traffic characteristics are set as follows:

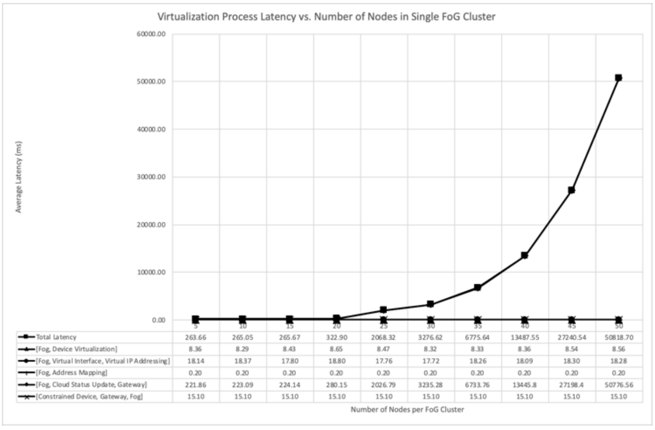
Traffic Source	Characteristics
Sensors	• Deterministic Transmit Distribution: every 5ms • Status signaling: 1 status packet/data packet • Packet size: 127 bytes
Actuators	• Periodic Instructions: every 5ms • Status signaling: 1 status packet/data packet • Packet size: 127 bytes
Inter Software Module Instance	• Reactive Transmit Distribution: in response to a trigger • Packet size: 1280 bytes

Tuple characteristics are set as follows:

Tuple Type	CPU utilization (million instructions)	Network length (bytes)
Sensor Data	100	127
Actuator Message	100	127
Low Computation Software Module Instance	500	1280
Medium Computation Software Module Instance	1000	1280
High Computation Software Module Instance	1500	1280

VIII. RESULTS AND DISCUSSIONS

Average end-to-end latency can be evaluated for each operation modes through simulation of the above scenario. Graph in Fig. 6 shows the results for End-to-end latency and its components for the virtualization setup phase in hybrid operation mode.

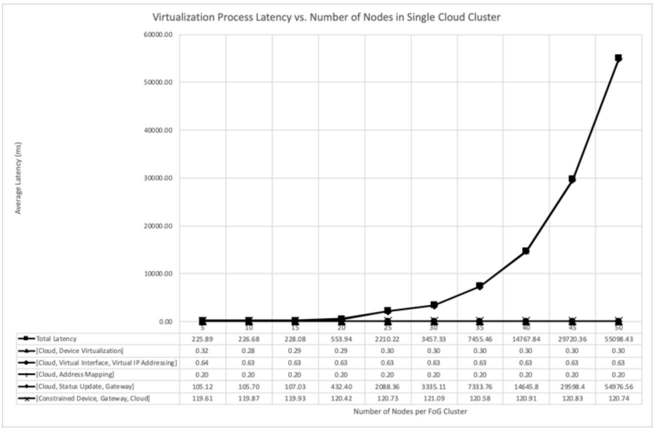


	5	10	15	20	25	30	35	40	45	50
Total Latency	263.66	265.05	265.67	322.90	2068.32	3276.62	6775.64	13487.55	27240.54	50818.70
[Fog, Device Virtualization]	8.36	8.29	8.43	8.65	8.47	8.32	8.33	8.36	8.54	8.56
[Fog, Virtual Interface, Virtual IP Addressing]	18.14	18.37	17.80	18.80	17.76	17.72	18.26	18.09	18.30	18.28
[Fog, Address Mapping]	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
[Fog, Cloud Status Update, Gateway]	221.86	223.09	224.14	280.15	2026.79	3235.28	6733.76	13445.8	27198.4	50776.56
[Constrained Device, Gateway, Fog]	15.10	15.10	15.10	15.10	15.10	15.10	15.10	15.10	15.10	15.10

Fig. 6. Virtualization phase end-to-end latency in hybrid mode

The first observation that can be drawn from the result in Figure 5.3 is that, 15 to 20 constrained devices per virtual cluster is the most optimal number of devices that should be virtualized in a single Fog node to keep the overall virtualization latency to minimum. Beyond 20 constrained devices, the average latency starts to increase exponentially, and this would adversely affect the performance of the IoT application that is deploying this algorithm in hybrid mode. Further, breakdown of the average latency into its subcomponents reveals that the virtualization process itself is not responsible for the exponential latency but the synchronization steps with the Cloud is the major contributor. This is because limited computation on a single Fog device becomes the bottleneck to the Cloud.

Graph in Fig. 7 shows the results for End-to-end latency and its components for the virtualization setup phase in Cloud only operation mode.



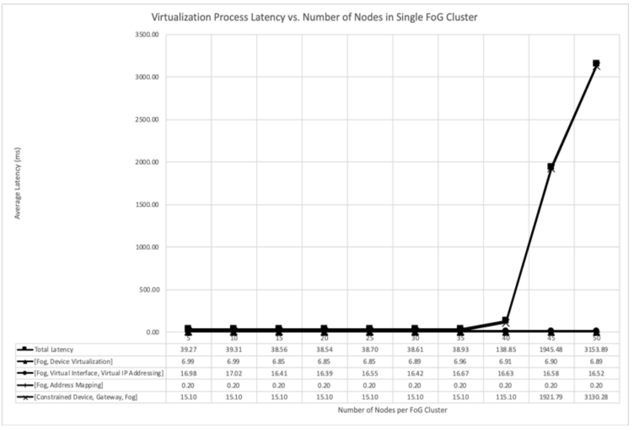
	5	10	15	20	25	30	35	40	45	50
Total Latency	225.89	226.68	228.08	553.94	2210.22	3457.33	7455.46	14767.84	29720.36	55098.43
[Cloud, Device Virtualization]	0.32	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.30	0.30
[Cloud, Virtual Interface, Virtual IP Addressing]	0.64	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63	0.63
[Cloud, Address Mapping]	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
[Cloud, Status Update, Gateway]	105.12	105.70	107.03	432.40	2088.36	3335.11	7333.76	14645.8	29598.4	54976.56
[Constrained Device, Gateway, Cloud]	119.61	119.87	119.93	120.42	120.73	121.09	120.58	120.91	120.83	120.74

Fig. 7. Virtualization phase end-to-end latency in Cloud only mode

From the graph in Fig. 7, it can again be observed that 15 to 20 constrained devices per virtual cluster is the most optimal and the major contributor to the overall latency is again the synchronization step between the Cloud and the gateway which becomes the bottleneck to the Cloud.

Graph in Fig. 8 shows the results for end-to-end latency in pure Fog only mode of operation. Here, it can be observed that even up to 40 constrained devices per virtual cluster gives and acceptable latency.

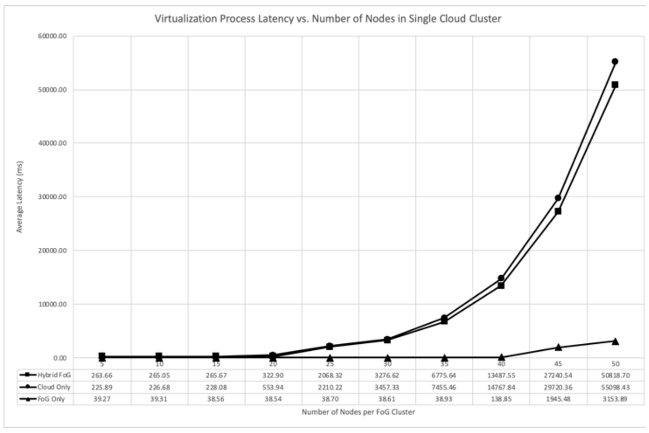
Even though computation is no longer a concern in this mode, beyond 40 constrained devices, the constant synchronization of states between the physical constrained device and the virtual constrained device becomes the major contributor to the latency as the single Fog node becomes the bottleneck.



	5	10	15	20	25	30	35	40	45	50
Total Latency	39.27	39.31	38.56	38.54	38.70	38.61	38.93	138.85	1945.48	3153.89
[Fog, Device Virtualization]	6.99	6.99	6.85	6.85	6.85	6.89	6.96	6.91	6.90	6.89
[Fog, Virtual Interface, Virtual IP Addressing]	16.98	17.02	16.41	16.39	16.55	16.42	16.67	16.63	16.58	16.52
[Fog, Address Mapping]	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
[Constrained Device, Gateway, Fog]	15.10	15.10	15.10	15.10	15.10	15.10	15.10	115.10	1921.79	3130.28

Fig. 8. Virtualization phase end-to-end latency in Fog only mode

Fig. 9 compares the total latency of each mode of operation and it is fairly obvious that the best mode of operation is Fog only mode and the worst is the Cloud only mode. However, Fog only mode is not the most practical mode of operation because it is virtually impossible to provide such a high computation resource at the network edge.



	5	10	15	20	25	30	35	40	45	50
Hybrid FoG	263.66	265.05	265.67	322.90	2068.32	3276.62	6775.64	13487.55	27240.54	50818.70
Cloud Only	225.89	226.68	228.08	553.94	2210.22	3457.33	7455.46	14767.84	29720.36	55098.43
FoG Only	39.27	39.31	38.56	38.54	38.70	38.61	38.93	138.85	1945.48	3153.89

Fig. 9 Virtualization phase end-to-end latency in all modes

IX. CONCLUSION

In conclusion, it can be noted that a physical constrained IoT device can be virtualized along with its communication protocols to provide low latency setup that is suited to the real-time use cases as long as the number of nodes is below 15 nodes/cluster. Virtualization of constrained IoT devices also allows virtual protocols to be executed on constrained IoT devices. Virtual protocols can then be used for communication between devices to achieve interoperability without considering the actual protocols on the physical constrained IoT device.

REFERENCES

1. B. N. Gopalsamy, “Communication Trends in Internet of Things,” in igi-global.com, 2017, pp. 284–305.
2. B. Cheng, S. Zhao, J. Qian, Z. Zhai, and J. Chen, “Lightweight Service Mashup Middleware with REST Style Architecture for IoT Applications,” IEEE Trans. Netw. Serv. Manag., vol. 15, no. 3, pp. 1063–1075, 2018.
3. S. K. Datta, C. Bonnet, and N. Nikaein, “An IoT gateway centric architecture to provide novel M2M services,” in 2014 IEEE World Forum on Internet of Things, WF-IoT 2014, 2014, pp. 514–519.
4. R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future internet: The internet of things architecture, possible applications and key challenges,” in Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012, 2012, pp. 257–260.
5. F. Carrez, T. Elsaleh, D. Gomez, L. Sanchez, J. Lanza, and P. Grace, “A Reference Architecture for federating IoT infrastructures supporting semantic interoperability,” in EuCNC 2017 - European Conference on Networks and Communications, 2017, pp. 1–6.
6. J. Ren, H. Guo, C. Xu, and Y. Zhang, “Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing,” IEEE Netw., vol. 31, no. 5, pp. 96–105, 2017.
7. D. Singh, G. Tripathi, A. M. Alberti, and A. Jara, “Semantic edge computing and IoT architecture for military health services in battlefield,” in 2017 14th IEEE Annual Consumer Communications and Networking Conference, CCNC 2017, 2017, pp. 185–190.
8. M. Noura, M. Atiquzzaman, and M. Gaedke, “Interoperability in Internet of Things: Taxonomies and Open Challenges,” Mob. Networks Appl., pp. 1–14, Jul. 2018.
9. M. K. Shin, K. Nam, S. Pack, S. Lee, and R. Krishnan, “Verification of NFV Services : Problem Statement and Challenges (Research Report No.02),” 2016.
10. Interoperability and Open-Source Solutions for the Internet of Things, vol. 10218. 2017.
11. R. Sutaria and R. Govindachari, “Making sense of interoperability: Protocols and Standardization initiatives in IOT,” in 2nd International Workshop on Computing and Networking for Internet of Things (CoMNet-IoT) held in conjunction with 14th International Conference on Distributed Computing and Networking (ICDCN 2013), 2013, pp. 2–5.
12. RFC, “Terminology for Constrained-Node Networks [RFC 7228],” Ietf Lwig. pp. 1–17, 2014.
13. Nagasai, “Classification of IoT Devices - CISO Platform.” CISO Platforn, Bangalore, 2017.
14. R. Sanchez-Iborra and M. D. Cano, “State of the art in LP-WAN solutions for industrial IoT services,” Sensors (Switzerland), vol. 16, no. 5. 2016.
15. S. K. Datta and C. Bonnet, “Extending DataTweet IoT architecture for virtual IoT devices,” Proc. - 2017 IEEE Int. Conf. Internet Things, IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017, vol. 2018-Janua, no. Vid, pp. 689–694, 2018.
16. T. Pflanzner and A. Kertesz, “A survey of IoT cloud providers,” in 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2016 - Proceedings, 2016, pp. 730–735.
17. E. S. Pramukantoro, W. Yahya, and F. A. Bakhtiar, “Performance evaluation of IoT middleware for syntactical Interoperability,” 2017 Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS 2017, vol. 2018-Janua, pp. 29–34, Oct. 2018.
18. D. Andročec, M. Novak, and D. Oreški, “Using semantic web for internet of things interoperability: A systematic review,” International Journal on Semantic Web and Information Systems, vol. 14, no. 4. pp. 147–171, 2018.
19. S. Haseeb, A. H. A. Hashim, O. O. Khalifa, and A. F. Ismail, “Connectivity, interoperability and manageability challenges in internet of things,” in AIP Conference Proceedings, 2017, vol. 1883.
20. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” IEEE Commun. Surv. Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.
21. B. Tank, H. Upadhyay, and H. Patel, “A survey on iot privacy issues and mitigation techniques,” in ACM International Conference Proceeding Series, 2016, vol. 04-05-Marc.
22. Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, “SDIoT: a software defined based internet of things framework,” J. Ambient Intell. Humaniz. Comput., vol. 6, no. 4, pp. 453–461, Aug. 2015.
23. C. Prazeres and M. Serrano, “SOFT-IoT: Self-organizing FOG of things,” Proc. - IEEE 30th Int. Conf. Adv. Inf. Netw. Appl. Work. WAINA 2016, pp. 803–808, 2016.

AUTHORS PROFILE

Shariq Haseeb is an IoT Specialist at TMR&D and REDtone IoT, Shariq Haseeb has architected several IoT solutions under the Smart City concept. He is the key architect being the TMR&D Smart Helmet, Smart Streetlight, Smart Vehicle and REDtone IoT's citizen engagement solution (CitiAct). Prior to joining REDtone IoT, Shariq was Head of Project in MIMOS Berhad. He has over 12 years research and prototyping experience in the field of computer communication and networks. He has worked cross functionally within protocol, network, infrastructure and device development to invent bleeding edge technology for global market space. He has filed more than 50 patents within Malaysia and Internationally and has more than 35 publications in journals and conferences.

Aisha Hassan Abdalla Hashim received her Ph.D in Computer Engineering (2007), M.Sc. in Computer Science (1996) and B.Sc. in Electronics Engineering (1990). She won the Best Graduating Ph.D Student Award during the IIUM Convocation ceremony in 2007. She joined IIUM in 1997 and is currently a Professor at the Department of Electrical and Computer Engineering. Professor Aisha has taught several courses related to Communication and Computer Engineering and is actively involved in curriculum development and programme accreditation. She has been a member of the Department Board of Studies for several years. She received the Best Teacher Award during IIUM Quality Day in 2007. Prof. Aisha has been appointed as external examiner/visiting professor/adjunct professor at different universities. Professor Aisha who is actively involved in research and postgraduate programmes, has published more than 200 journal/conference papers, and supervised/co-supervised more than 60 Ph.D/Master students. She received the Promising Researcher Award in 2009 during IIUM Quality Day. She has also received many medals/awards in different national/international research exhibitions. One of her research exhibitions won the Promising Commercial Value Award (Second Runner Up) in IRIIE 2014. As a researcher, she has secured research grants from IIUM, Ministry of Higher Education (MOHE) and Ministry of Science, Technology and Innovation (MOSTI). She has actively contributed as a reviewer /technical committee member in many journals/conferences. Professor Aisha has established several teaching/ research networks between IIUM and overseas universities. She has participated in initiating several MoUs as well as encouraging the PhD Student Mobility programme between IIUM and overseas Universities.

Othman Omran Khalifa received his Bachelor's degree in Electronic Engineering from Garyounis University, Libya in 1986. He obtained his Master degree in Electronics Science Engineering and PhD from Newcastle University, UK in 1996 and 2000 respectively. He worked in industry for eight years and he is currently a Professor and at the department of Electrical and Computer Engineering, International Islamic University Malaysia. His area of research interest is Communication Systems, Digital image / video processing, coding and Compression, Wavelets, Fractal and Pattern Recognition. Prof. Khalifa is a Charter Engineer (CEng) and Senior member of IEEE, USA and a member IET, UK. and a member of the Council of Professors of Malaysia. Prof. khalifa was the chairman of the International Conference on Computer and Communication Engineering (ICCCE), 2006, 2010, 2012, 2014. Prof. Khalifa has extensively contributed through his writings in international journals, conferences and book. He published more than 450 publications including 10 books. He is a member of many international advisory boards for many international conferences a member of many editorial boards of many international.

Ahmad Faris Ismail is a Professor and the Dean of Engineering at the International Islamic University Malaysia (IIUM). He was the IIUM Deputy Rector (Research & Innovation) from July 2009 until June 2013. He served as the Dean of Engineering from 1997 until 2009. He obtained his B.Sc. in Chemical Engineering, in 1988, from the University of Houston, USA, and Ph.D in Engineering from Rice University, USA, in 1993. He is the Chief Editor for the IIUM Engineering Journal. Prof. Ismail was a Visiting Academic at the University of Southern Queensland in 2014 and a Visiting Scientist at Kyoto University in 2004. He has been invited as keynote speakers at various international conferences and congresses. He is also a co-inventor for at least eight filed patents of research products and has published more than 200 papers in refereed journals and conference proceedings. His research topics include energy and environment, simulation and modelling, computational fluid dynamics, combustion, and nanofluidss.