IoT Operating Systems : Perspective, Availability and Proposed New Architecture

** Parthavi Gor*
*** Taslim Divan*
**** Divya Makwana*
***** Himani Sharma*
****** D. K. Jhala*

Abstract

Internet of Things (IoT) is penetrating our lives faster than earlier due to the fact that it can create valuable real time data without any human interference. As there is no human involved, data generation is certain irrespective of any situation. The accuracy depends upon the sensors used in the implementation. Operating System is a basic and important part of any computational system, so is the case with IoT. However, IoT is still in its infancy, proper standards are yet to be developed and formalized. There are many IoT Operating Systems available in the market, both open source and closed. Users select from the available OS as per their suitability and requirements. However, there is no standard Operating System which can suit all IoT installations/designs. This paper discusses the main features of existing IoT Operating Systems and parameters suitable for IoT Operating Systems. While using these existing Operating Systems, we have come across many issues, both technical and management. These issues are discussed in this paper with utmost care and we have tried to address the users' concern with these. Based on these issues, we tried to derive revised architecture of IoT Operating System which can take care of these issues if they are implemented. We have taken into account the existing architecture of Contiki and RIOT OS while deriving revised architecture. Since IoT is slowly but surely spreading its wings to industrial applications, security is utmost important. Many security issues are still to be addressed by existing Operating Systems; hence, our main concern is to address such issues.

Keywords : Contiki, IoTOS, RTOS, Security

I. INTRODUCTION

IoT has become a buzzword in the field of Information and Communication Technology (ICT) due to its applicability in a wide variety of areas. Internet computing beyond computers has become a reality with the emergence of IoT. IoT has become a darling of users due to its affordability and vast reach. It has touched almost all areas/domains of human life for betterment. The core competence of IoT is to generate and store data all the time without any human intervention. This has created big shift in Big Data analytics and possibility exists to have real time data for better monitoring and control. IoT needs seamless integration of many technologies which include cellular (4/5G etc.), wireless network technologies, fog computing, Big Data, operating systems etc. [4]. Research in these areas will bring more advanced and intelligent IoT which can pave the way for the next industrial revolution, which is popularly known as Industry 4.0. Integration of IoT with Machine Learning/ Deep Learning will give birth to an altogether new domain of real time data collection (by

Manuscript received July 14 2019; revised August 5, 2019; accepted August 18, 2019. Date of publication October 5, 2019.
* G. P. Nishilbhai is a student of Bachelor of Engineering, Gujarat Technological University, Ahmedabad, Gujarat –382 424.
** D. T. Rajabsha is a student of Bachelor of Engineering, Gujarat Technological University, Ahmedabad, Gujarat – 382 424.
*** D. M. Jagdishbhai is a student of Bachelor of Engineering, Gujarat Technological University, Ahmedabad, Gujarat – 382 424.
**** H. S. Vishnubhai is a student of Bachelor of Engineering, Gujarat Technological University, Ahmedabad, Gujarat – 382 424.
***** D. K. Jhala is Visiting Faculty at Bhaskaracharya Institute for Space Applications and Geo-Informatics, Gandhinagar, Gujarat, 382 007. (email : dk.jhala@gmail.com)
DOI : 10.17010/ijcs/2019/v4/i5/149456

IoT devices) and control/forecasting [using Machine Learning (ML) / Deep Learning (DL)] techniques. This is likely to lead to complete or partial elimination of human interference in many domains, specially industry.

Operating Systems (OS) are one of the very important ingredients of IoT solutions. Operating Systems are traditionally used as an interface between programs and under lying hardware to relieve programs from handling complex functions which enable them to focus more on their core functionality based on specific applications. In most cases, OS were mainly used on high end computing devices having sufficient processing and memory power and having no or less power constraints.

Emergence of IoT paradigm has brought a paramount shift in the use and requirement of Operating Systems as IoT domain demands diverse requirements. Operating systems (OS) are essential foundations for enabling IoT devices. Traditional personal computers (PCs) and mobile devices were built on dominant operating systems, namely Linux/Window/Android etc. While these operating systems provide standardized software platforms, they cannot successfully function within the economic constraints that drive the IoT space. IoT devices are expected to be deployed in billions and trillions, at much lower capital and operational expenditure than those required for PCs and smart phones. An operating system tuned for IoT devices needs to deliver scalability, portability, connectivity, security, usability, and multi-app execution.

II. IoT DESIGN

Design of IoT solutions needs to balance many things such as hardware, software, network requirements etc. Application software can influence hardware implementation and physical design. As designs become more compact and even wearable, the mechanical design can constrain the hardware and hence, the software capabilities of the device. Hence, IoT domain needs an operating system that can operate on constrained hardware while still delivering the app-driven experiences and services expected of IoT devices.

Use of IoTdevices depends upon the scale and scope of particular applications and locations where they are implemented. Recently, the Engineering Internet Task Force (IETF) standardized a classification of these devices in three categories based on memory capacity [14].

Class 0 devices have the smallest resources (<<10 KB of RAM and <<100 KB Flash), for example, a specialized mote in a Wireless Sensor Network (WSN).

Class 1 devices have medium-level resources (approximately 10 KB of RAM and 100KB of Flash) allowing richer applications and more advanced features than rudimentary motes, for example, routing and secure communication protocols.

Class 2 devices have more resources, but are still very constrained compared to high-end IoT devices and traditional internet hosts.

For specialized and resource constraints class 0 devices, normally any type of Operating System is not suitable. Instead, hardware specific software is preferred. For class 1 and 2 IoT devices, hardware independent OS needs to be used which provides flexibility for large scale software development and maintenance. Hence, this paper discusses the IOT OS mainly used for class 1 and 2 types of devices. This paper also analyzes more prominent IOT OSs, both open source and commercial (closed source), which are compatible with memory size, IP protocol, programming tools, and large section of popular hardware platforms.

III. IMPORTANT PARAMETERS FOR IoT OS

IoT domain is constrained by many things as compared to normal computing domain. IoT domain requirements are altogether different and cannot be satisfied by usual computing procedures or tools. This section provides major and important parameters driving the selection of IoT OS. It is interesting to note that few parameters cannot be fulfilled by existing OS in totality.

A. Footprint [1]

As IoT devices are required to be deployed in large numbers at remote locations, they always have memory limitations in terms of both RAM and ROM unlike traditional PCs or computing devices. Hence, IoT OS should be optimized to provide required IoT functionality and required data structure. It is expected that IoT OS should have low memory, power and processing requirements to have minimum overhead.

B. Heterogeneity [1]

IoT solutions use different types of hardware having different make micro controllers. At times different hardware boards are used for the same solutions. Scale of diversity is very large for hardware boards as well as communication protocols unlike traditional computing

solutions. IoT OS should be able to cater to these requirements of supporting vast variety of hardware and communication/network protocols.

C. Scalability, Portability, and Modularity

IoT OS should have all these three embedded in them. Scalability should be from node to gateways. OS should be able to port to any hardware. Modularity should provide kernel as core and other application requirements as add on packages. Modular approach will help reduce the size of OS as developers can select components as per design requirements.

D. Connectivity

Network is core to IoT. In any IoT solutions, IoT devices are connected to have communication among themselves as well as over internet. Hence, network requirements in IoT are very different to have multiple protocol support. IoT uses a variety of network protocols; both wireless (IEEE 802.14, Blue tooth etc.) and wire line, at data link and network layer, the underline IoT OS should support multiple stacks to fulfill these requirements.

E. Power Requirements

IoT devices are normally implemented at remote locations and should be powered on all the time. This requires use of batteries which make them power deficient. Since power is consumed by many sub components of IoT (sensors, hardware circuits, radios etc.), it is necessary that OS should have various provisions to save energy. Also, due to location constraints, IoT device batteries should provide power for longer period of time with one time charging. Power saving measures can be achieved by providing sleep mode facility, minimizing or reducing tasks to be executed etc.

F. Security

Any networked systems should adhere to the required security standards. IoT being connected to the internet and part of any critical infrastructure should also adhere to security and safety standards. Hence IoT OS should provide security to the device by way of secure boot, SSL, drivers, and components for encryption. Security is a continuous process, therefore, OS should have provision for remote security updates for IoT devices in the field.

G. Reliability

This is one of the most important parameters for IOT OS as devices are at remote locations which can't be maintained frequently, and should work for years without failure. This should be achieved by proper certification of IoT OS for certain applications.

IV. EXISTING OPEN SOURCE AND COMMERCIAL IoT OS

There exist many OS which are used by developers for IoT applications. Selection criteria depends upon application requirements, availability of network, location etc. This paper will discuss some of the prominent open source IoT OS which are mainly used for different IoT projects/applications.

Contiki [5] is a popular OS because of its light weight, maturity, and flexibility. It is used for networked, memory-constrained, and low-power wireless IoT devices. Few examples of its use include street lighting, sound monitoring for smart cities, and radiation monitoring etc. Contiki supports wide variety of hardware boards and network protocols. Major features include multi-tasking, pre-emptive multi-threading, support of web browser and internet protocol including IPV6. Cooja, single thread simulator makes it slow for specific network scenarios. Existing versions may not support new IoT hardware platform and need to be modified/developed.

RIOT [6] is using micro kernel architecture with low memory use like Contiki. Developers can use C and C++ for writing application programs. RIOT also has multi-threading and real time capabilities. It also provides SSL/TSL libraries. RIOT can run on 8, 16, and 32 bit processors and is compliant with many existing IoT hardware boards. It supports network stacks, namely IPV6, 6LoWPAN, RPL, UDP, TCP and CoAP.

TinyOS [8] is an embedded, component-based operating system and platform for low-power wireless devices, such as those used in wireless sensor networks (WSNs). It is written in C programming language called nesC. It is fully non-blocking and has one call stack. Hence I/O operations which are longer are treated asynchronously and can be called back. It maintains high concurrency with one stack by stitching many small events. For large computations, tasks are used. It supports IPV6 stack using 6LoWPAN. Better network life time is possible due to low power listening. Tiny OS demands some adjustment for communication between hardware and software due to low voltage restrictions. Popular

applications include smoke detection devices, military activity, temperature control, bank's security systems, resource monitoring etc.

Ubuntu Core is a specially designed IoT OS which is light weight, secure, and transactionally updated. An Ubuntu Core system is built using snaps: a core snap, a kernel snap, a gadget snap, and usually runs one or more application snaps. Major features include faster, reliable, highly secure Linux distribution. It also provides public/private key based validation and authentication.

Nucleus RTOS [10] is a real-time operating system (RTOS) offered by the Embedded Software Division of Mentor Graphics, supporting 32 and 64 bit embedded platforms. The Nucleus RTOS is designed for real-time embedded systems for use in medical, industrial, consumer, aerospace, and IoT applications. Its main features include power management, 64 bit support, process model, safety certification etc. The Nucleus provides wide networking stack that supports over 60 networking protocols including IPV4 and IPV6. It is one of the major OS used in many implemented IoT installations.

MBED OS [7] is open source OS which only supports ARM processors. This OS has footprint in Smart Home and Wearable devices domain of IoT. The OS differs from many other embedded operating systems because it is single-threaded as opposed to multi-threaded. Mbed OS provides the Mbed C/C++ software platform and tools for creating microcontroller firmware that runs on IoT devices. It consists of the core libraries that provide the microcontroller peripheral drivers, networking, RTOS, and runtime environment, build tools, and test and debug scripts. These connections can be secured by compatible SSL/TLS libraries.

Wind River VX Works [11] is one of the most used commercial OS today in IoT domain. It is highly scalable and also provides security features which are important for IoT applications. Vx Works is very well-known in the industrial, medical, and aerospace fields because it is one of the few RTOS that have met the necessary certification requirements to be used in these industries. VxWorks supports Intel architecture, POWER architecture, and ARM architectures. The RTOS can be used in multi core asymmetric multiprocessing (AMP), symmetric multiprocessing (SMP), mixed modes, and multi-OS designs on 32- and 64-bit processors.

Window10 [9] for IoT is a family of operating systems from Microsoft. It is designed for use in embedded systems. Microsoft currently has three different subfamilies of operating systems. First is Windows 10 for IoT Mobile, which supports the ARM architecture. Second is Windows 10 for IoT Core which supports Raspberry Pi and Intel Atom and third, Windows 10 for IoT Enterprise, more or less full-blown Windows 10 Enterprise, but restricted to running a single application. This has advantage of having large MS community users who can develop applications using Visual Studio. This is useful for in house app development.

Google Brillo [12] is an Android-based embedded operating system platform by Google. It is aimed to be used with low-power and memory constrained IoT devices, which are usually built from different MCU platforms. As an IoT OS, it is designed to work with as low as 32–64 MB of RAM. It will support Bluetooth Low Energy and Wi-Fi. Along with Brillo, Google also introduced the Weave protocol which these devices can use to communicate with other compatible devices. This means that smart devices don't necessarily need to have embedded Android as their OS, they only have to have the ability to communicate using Weave.

Table I gives a comparison of OS.

IoT OS Comparison

The fact is that IoT domain requirements are very diverse. One particular OS cannot fulfill all its

TABLE I.
OS COMPARISON

OS	Min RAM	Min ROM	Support C	Support C++	Multi Threading	Architecture	Type of Scheduler
Contiki	<2KB	<30KB	Partial	No	Partial	Monolithic	Cooperative, Preemptive
RIOT	<=1.5KB	<=5KB	Yes	Yes	Yes	Micro Kernel	Tickless, Preemptive, Priority based
Tiny OS	<1KB	<4KB	No	No	Partial	Monolithic	Cooperative
Mbed OS	<=5KB	<=15KB	Yes	Yes	Yes	Monolithic	Preemptive
Brillo	<32MB	<128MB	Yes	Yes	Yes	Monolithic	

requirements, and therefore, developers need to choose and pick as per their choice [4].

V. ISSUES AND CONCERNS

We have observed that there is no single IoT OS which can fulfill all requirements of any IoT design. Lack in standards seems to be the major concern for IoT OS. Due to varying nature of IoT implementations, it is very difficult to have all required features to build on one particular OS. This diverse need of IoT domain has created major security issues in this field. A multitude of IoT Operating Systems is a bad news for the internet [3]. Present internet has one major advantage in terms of use of Operating Systems by its computing devices. We have very limited OS which rules these devices. Due to this, it is very easy to handle security related issues like patching the OS, finding the vulnerability. With fewer versions of patches, we are able to mitigate the risk of security breach very fast. Same will not be the case with IoT in place, as we have seen that IoT OS has very large domain with variety of vendors in market. This issue will aggravate the security of internet and IoT devices. When magnitude of IoT devices will cross 20 billion mark by 2020 [3], security issue will become unmanageable unless standards are defined for IoT do main including IoT OS. Another problem is how to install patches in IoT devices as and when required. There is no defined mechanism in place, as in many cases IoT devices are installed in network constrained places. Same is the problem with virus mitigation risk. Present computing devices are normally connected to centralized virus systems for getting periodical updates. However, in absence of such mechanisms in case of IoT, there is always a risk of getting virus in the IoT system.

IoT industry has already experienced notorious IoT-based DDoS attack named Mirai because of the exploitation of a known vulnerability in the operating systems used by dozens of CCTV cameras and DVRs. A newly discovered Trojan malware, dubbed Rakos uses brute force SSH login attack to compromise IoT devices embedded with vulnerable versions of Linux [2]. When IoT is being extended on large scale for Home Automation and Industry 4.0, such concerns are of paramount importance [2].

VI. PROPOSED REVISED IoT OS ARCHITECTURE

We have considered Contiki and RIOT OS architecture while deriving at revised architecture for IoT OS. The following aspects are proposed/ considered while building robust IoT OS.

1. Standard modular architecture for IoT OS to be developed which can cater to any footprint of memory. Stringent standards should be defined for authentication and authorization. Audit and logging standards should also be in place[3]. It is proposed to have configuration module for various network stacks and type selection of scheduler.
2. Interoperability by defining common interface among various network stacks for inter-communication of IoT devices with diverse OS. This will also partly take care of scalability. It is proposed to have "common-wl-interface" module which is software for communication with other IoT devices over wireless by providing paring (similar) software module in other devices.
3. Push mechanism from centralized node should be adopted for patch update wherever network connectivity is not a bottleneck. This module will keep track of previous updates installed for continuity of patch update. This can be online or offline depending on memory availability.
4. Security module can provide various provisions for access, authentication, and keeping track of important security configuration related parameters. This module also verifies the user who communicates through "common-wl-interface from other IoT devices. This module needs to maintain proper logging of outside users coming through this interface.
5. OS certification should be made mandatory. Certification details may be linked to security module for verification.

VII. CONCLUSION

It is certain that IoT technology of future is going to change the use of internet, and the way we have used it so far. Any new technology, when in infancy always has many issues and challenges. IoT is also no exception. This paper aims to discuss various options available for IoT OS in the present IoT domain space with important parameters impacting the selection of OS. It is very clear that lack of standards in this field has created confusion in the IoT community. This needs to be addressed to mitigate various technological problems related to security, modularity, and interoperability. Revised IoT OS architecture is also proposed for betterment and approaches towards some standard in this field. It is expected that research guidance provided here will help

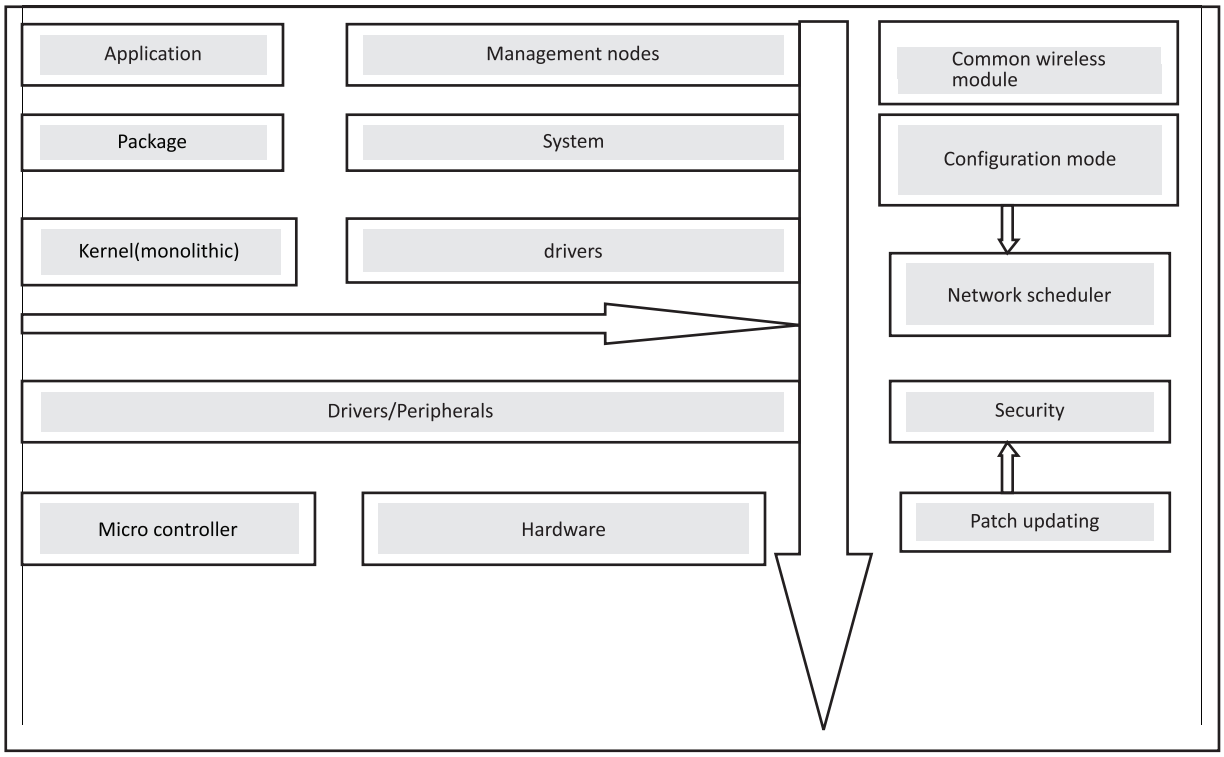

Fig. 1. Revised Architecture Diagram

the community in better development of products.

REFERENCES

[1]Devopedia, "IoT operating systems," Version 9, August 23, 2018. Accessedon: Aug. 23, 2019 [Online]. Available: https://devopedia.org/iot-operating-systems

[2] Jain, H.,"A multitude of IoT Operating Systems is bad news for the safety of the internet." [Online]. Available: https://www.fortinet.com/blog/industry-trends/a-multitude-of-iot-operating-systems-bad-news-for-the-safety-of-the-internet.html

[3] A. Banafa, "3 major challenges IoT is facing," 2017. [Online]. Available: https://www.bbvaopenmind.com/en/technology/digital-world/3-major-challenges-facing-iot/

[4] Y. B. Zikria, S. W. Kim, O. Hahm, and M. K. Afzal, "Internet of Things (IoT) operating systems management: Opportunities, challenges, and solution," 2019. doi: 10.3390/s19081793

[5] Contiki: The Open Source OS for the Internet of Things. Available [Online: https://en.wikipedia.org/wiki/Contiki

[6] sRIOT: The Friendly Operating System for the Internet of Things. [Online]. Available: https://www.riot-os.org/

[7] MbedOS. [Online]. Available https://en.wikipedia.org/wiki/Mbed

[8] TinyOS. [Online]. Available: https://en.wikipedia.org/wiki/TinyOS

[9]WindowsIoT. [Online] .Available:https://en.wikipedia.org/wiki/Windows_IoT

[10] Nucleus RTOS. [Online]. Available: https://en.wikipedia.org/wiki/Nucleus_RTOS

[11] VxWorks. [Online]. Available: https://en.wikipedia.org/wiki/VxWorks

[12]Google Brillo.[Online].Available: https://en.wikipedia.org/wiki/Android_Things

About the Authors

Gor Parthavi Nishilbhai is final year student of Bachelor of Engineering at Gujarat Technological University. Her major areas of interest are Internet of Things and Machine Learning.

Divan Taslim Rajabsha is final year student of Bachelor of Engineering at Gujarat Technological University. Her major areas of interest are Internet of Things and wireless networks.

Divya Makwana Jagdishbhai is final year student of Bachelor of Engineering at Gujarat Technological University. Her major areas of interest are Internet of Things and software development.

Sharma Himani Vishnubhai is final year student of Bachelor of Engineering at Gujarat Technological University. Her major areas of interest are Internet of Things, Machine Learning and software development.

D. K. Jhalais Visiting Faculty at Bhaskaracharya Institute for Space Applications and Geo-Informatics, Gandhinagar, Gujarat. He has about 35 years of experience in the field of ICT and related areas. He has experience in the field of Networking as well He has worked on designing of ICT projects, their development, and implementation at grass root level.