Edge Computing and Programming languages

Bharath Joshi
IMT2018016
IIIT Bangalore
bharathkumar.joshi@iiitb.ac.in

Krishna Phalgun
IMT2018038
IIIT Bangalore
Krishna.Phalgun@iiitb.ac.in

Shathir Hussain
IMT20180170
IIIT Bangalore
Shathir.Hussain@iiitb.ac.in

***Abstract*—The advent of Internet of Things applications and their specific demand gave rise to a new distributed computing paradigm Edge computing replacing Cloud computing. To solve the problem of latency, limited bandwidth edge computing was introduced which provides a robust, computing and storage resources closer to the user. The choice of programming language plays a pivotal role in shaping up any paradigm. This report provides an in-depth examination of the specific features, demands and issues and how they influence the design and selection of a programming language for developing edge computing software applications.**

I. Introduction

Cloud computing has revolutionized the IT industry by providing utility computing, virtualization, service-oriented architecture and parallel computing[1].Cost savings, increased collaboration, efficient recovery, and scalability are few of the advantages of cloud computing. Along with the numerous benefits it provides, it also has some disadvantages, such as high centralization(w.r.t. the cloud), inability to handle the growth of many latency-sensitive devices (IoT devices), security, and other issues which gave rise to Edge/Fog computing.

Edge Computing

Edge computing is a distributed computing paradigm in which substantial compute and storage resources are placed at the edge of the Internet, in close proximity to mobile devices, sensors, end users, and Internet of Things devices[2]. Edge computing is interchangeably used with fog computing. Fog computing is a technology to address computing and networking bottlenecks in large scale deployment of IoT applications[3]. Though they are termed differently both serve the same purpose, i.e. bringing the computation nearer to the end user. While fog computing brings it further down from the cloud servers, the edge computing brings either to the end device itself or the nearest gateway, the **edge**.

Depending on the task we're attempting to complete, the "edge" takes different forms. The edge of a telecommunications system is a cell phone or a cell tower. A car could be the edge of an automotive system. An edge in the manufacturing field could be a machine in the shop. A laptop could be the edge for an IT company.[4][5].

Figure 1 from [6] gives an insight about the architecture of traditional Edge computing paradigm. It can be thought of as a three-layer architecture, with the first layer consisting of users and some edge devices. The edge nodes or servers in the second layer are where tasks can be offloaded. The traditional cloud makes up the third layer.Scheduling and offloading take place in the first or second layer, respectively. Due to resource constraints, edge nodelets are not always capable of serving tasks when compared to cloud computing. The tasks that the edge nodelets can't handle are sent to the cloud server.

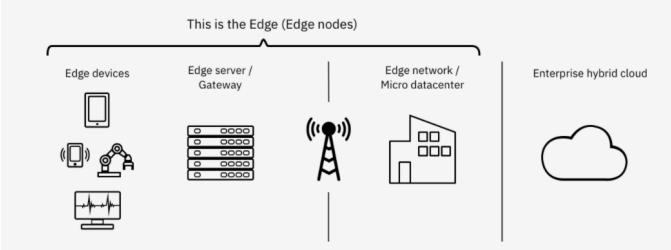

Fig. 1. Edge Architecture

Edge computing has gained a lot of popularity as a result of the various features it offers to its users. A few of them are:

- **Low latency**: Real time applications require a low latency. QoS (Quality of Service) is an important thing to provide the users with. Edge computing makes it easier for services to deliver streaming video and audio bringing high-demand content closer to end users through caching.
- **Scalability and flexibility** : According to [7] the number of connected devices in 2012 was around 15 billion and is also expected to rise above 50 billion by 2050. Edge computing being extremely scalable and flexible comes in handy to deal with the massive data being produced.
- **Security**: Distributed Denial of Service (DDOS) attacks and power outages are a big challenge in cloud computing. The characteristics of edge computing, i.e. having a large number of nodes and a wider range avoids any single point of disruption.
- **Reliability**, **Lesser Turn Around Times** and reducing the overall network traffic are some other features.

Due to the various advantages it has, the paradigm is used in many parts of the industry. A few of them are:

- M2M communication
- Smart grid, home, applications
- IoT Devices

- Autonomous Driving Vehicles
- Medical Care Systems

Along with the numerous benefits, it also has some drawbacks which needs to be dealt with. A few of them are

- Huge constraints in resources
  - Computational Resources: Due to the increase in the number of servers, providing the same type of computational resources (RAM, processors) as that of cloud computing isn't feasible. Edge computing also processes computes information after dividing them into sets of data. Due to the same, synchronization is a tedious task.
  - Less Storage space in the servers and more storage space in the edge devices. Due to the comparative small databases in the edge servers it isn't possible for the server to serve all the tasks. At the same time the edge computing softwares take up a considerable amount of spaces on the devices.
- Costly Affair: Setting up the infrastructure requires lot of investment. This is due to the addition equipment and more hardware.
- Maintenance: Distributed computing paradigms have high maintenance costs due to its nature of complex distribution. Edge computing also has complex distribution and enormous number of nodes which can make maintenance not an easy task.

A. *Security Problems*

The transition to edge computing from cloud computing has been successful in providing users with Qos (Quality of service) but haven't been totally transitioned to provide complete security. Now let us look at some of them,

- Malicious Hardware/Software Injections
  - Cyber criminals can use a variety of hardware and software-based tools to corrupt ,change, or delete data circulating inside the edge nodes.
  - There's also camouflaging (a common terminology in cyber security), in which attackers insert a fake edge computing server that is like any other node processes.
- Routing Information Attacks
  - Rooting attacks at the communication layer. Routing attacks impact latency by interfering with the way data is transmitted within a network.
  - In a Black hole attack,all packets are simply deleted; in a greyhole attack selected packets are deleted and in wormhole packets data is manipulated (the source address, headers)
- Distributed Denial of Service (DDoS) Attacks
  - Another edge computing security risk to be aware of is distributed denial of service (DDoS) attacks, which overload an established network resource with traffic from other compromised resources within the network. This is a common attack in all internet related paradigms.
  - Outage attack causes all the nodes to stop completely. While battery draining attack causes the lives of servers to go down, by giving high intense tasks to process.
- Other issues, such as our data being scattered around many locations rather than being clustered, are caused by the paradigm's architecture.
- Some researchers also claim that since data is sent to many locations, it is difficult to obtain all of the information.

B. *Programming languages*

We basically need them to interact with computers. There are many different languages for dealing with various problems in various domains, each with its own unique characteristics.

Because the existing problems are so diverse, creating a single universal programming language that meets all requirements is nearly impossible. Therefore, programming languages are frequently revised and even combined with other languages to meet our evolving technological needs over time.

The storm of digital revolution has brought a lot of change in everyone's life. New devices were introduced, there was a shift from the mechanical technology to the digital electronics. A lot of computers, computer programs were introduced which marked the beginning of Information Age[8].

The computer programs are run and implemented by/in programming languages. Programming languages refer to different types of expressions and logical structuring rules that serve to generate recurring and systematic tasks[9]. They are the basis for all the big programs that are used to run the web applications, mobile applications, operating systems, what not!. There are many programming languages that help us in different ways with different approaches. There is a great variety in the characteristics of programming languages, hence there could be a lot of divisions based on the features.

Choosing an appropriate programming language for a paradigm is a quintessential task as the basic working, behaviour of the application would be highly impacted by the choice of programming language.Edge computing being introduced to overcome the drawbacks of cloud computing had shown a lot of promise, but as discussed edge computing itself has a lot of demands to be satisfied along with its own advantages and issues. So for the rest of the paper we will concentrate on the choice of a programming language used to develop software applications for edge computing

The rest of the paper is organized as follows. Section 2 describes the literature survey, section 4 specifies the applications and paradigms which use edge computing, section 5 the programming languages and choice for software applications,section 6 does the same w.r.t. edge servers, section

7 concludes the study. A study is done on SICIoT (simple instructor Compiler for IoT) in the appendix.

II. Literature Survey

Qualitative work has been in selecting the choice of programming languages in the edge computing paradigm. Many times a new programming language was implemented which had the characteristics of the older programming languages. The authors in [10] implements a new Garbage collector and integrate with languages like Python and C. A new programming model and programming language MOLE was introduced in [11]. MOLE featured a declarative domain specific language (DSL). Emphasis on D language has been given by the authors in [12].Though D is similar to that of C++, it is a distinct language. Authors in [13] have proposed the choice of programming based on different approaches potentials and pitfalls in WSN (Wireless Sensor Networks).[14] has recommended using of Chronus for the automatic Wireless Sensor Networks. Authors in [18] have recommended the use of Seal Language. SICIoT[23] proposes a simple instruction compiler that can translate straightforward verbal instructions into a new IoT device functionality is presented in this paperwhich was the first kind of thing we heard. More about this would be discussed in the apendix.

III. The Applications

A. The Internet of Things

The Internet of Things (IoT) is a network of interconnected, internet-connected objects that can collect and transmit data without the need for human intervention over a wireless network. The Internet of Things (IoT) is widely discussed in research as well as in industry[15] and it has been evolving not only as a domain which requires hardware optimizations but also software optimizations. Devices for a remote dashboard,servers,bridge device, and sensors are all used in IoT systems. Platforms, embedded systems,and middle ware are all used in IoT software to address the key areas of networking and action. While hardware mostly focuses on data collection, the software emphasizes on pre processing, networking, data integrity , real time analytics.

B. Wireless Sensor Networks

A new generation of smart devices has emerged as a result of the convergence of low-cost wireless communication, computation, and sensing. Wireless sensor networks are a new technology created by combining tens to thousands of these devices in self-organizing networks (WSNs)[16]. They maybe seeming like the IoT applications, but there is subtle difference. WSN form a subset of IoT. Within an IoT system, WSN is frequently used. In an IoT system, a large number of sensors, such as those in a mesh network, can be used to collect data and send it to the internet via a router.[17]

C. Selected Use Cases of Edge Computing enabled Applications

1) Autonomous Vehicles: An autonomous vehicle can sense its surroundings and operate without the need for human intervention. Sensors, actuators, complex algorithms, machine learning systems, and powerful processors are used to execute software in autonomous vehicles. Calculations and decisions can be made much faster due to the low latency of edge computing.

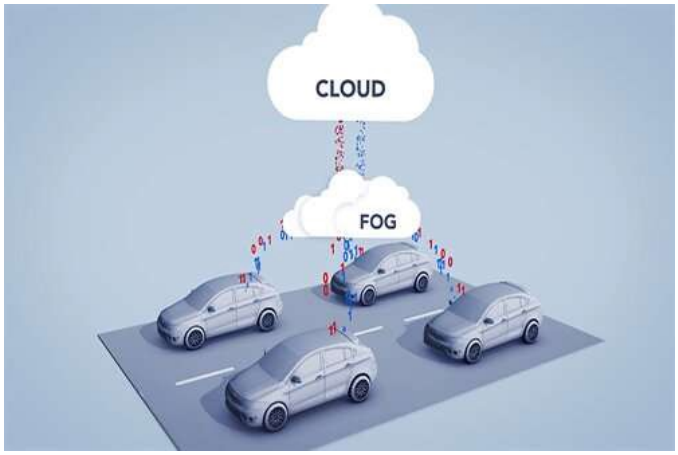

Fig. 2. Autonomous Vehicle

Programming Language Requirements

1) It should include API libraries for dedicated hardware-accelerated facilities.
2) While making quick decisions, the program execution should not wait for data to be passed around or for redundant copies of large data structures to be made. Garbage collectors in modern programming languages, for example, pause execution on a regular basis to free memory, and the entire system randomly comes to a complete halt on a non-deterministic basis.
3) In order to execute the driving algorithm, it should be able to handle a large amount of data and have libraries to support computation.

2) Smart Health Care: Technology that contributes to better diagnostic tools, patient care, and devices that improve everyone's quality of life is what smart health care is all about. The internet of things (IoT), big data, cloud computing, and artificial intelligence are all used in smart health care. To build a robust smart health care system, connected devices must respond quickly and analyse data in a real-time cloud environment.

Physicians can monitor and contact patients remotely. Sensors collect various types of data from patients and report it to their caregivers, just like an autonomous vehicle. The difference is that because the sensor output is small,

processing it does not necessitate a large amount of computing power.

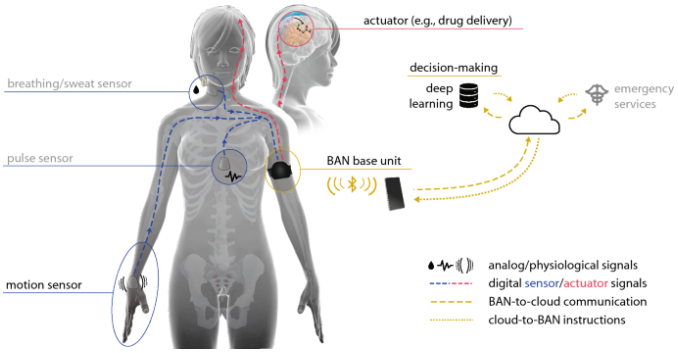

Fig. 3. Smart Health Care

Programming Language Requirements

1) Because we deal with people directly, there will be concerns about privacy and security. Not all languages of programming are secure. Only when the developer adopt best practices and best security policies and techniques, an application is secured. The popular programming languages are in general more safe because the more security flaws are widely known and the most of them could be fixed by professional developers.
2) Because these sensors are typically small and may be implanted within our bodies, the programming language must be able to manage memory effectively. Because if the application has a memory leak, it will eventually run out of memory and crash.
3) The application should be able to manage the exceptions in the workflows otherwise it might have unintended consequences.

IV. Programming Languages

Given that the IoT applications, WSN applications, all the applications which sought to benefited from edge computing are generally very simple in nature, have a lot of constraints both in terms of computational resources, storage. This all makes the development of software applications very hard. Selecting of an appropriate programming language for the software application could be tricky as many different models, architectures could have different needs. The languages used are expected to provide support to different platforms, hardware, libraries which could be used.Certain specific features of programming languages necessitate the use of large runtime libraries. These libraries must be compiled and linked to the executable for the target architecture[19]. A choice shall also be made on the level of language (low level, high level) as well as the development time required for a particular time. Differently typed programs give different features and they have to be considered. The features may include development speed, testing speed, flexibility, readability.Garbage collection can also be an important factor. Extrinsic properties also shall play a bigger role in selecting the choice of PL. Few of them are as follows:

- Libraries Available
- Community support
- Tools (IDEs, debuggers,etc)
- Compatibility

V. Proposed Choice of Programming Languages

Based on the different architectures of different applications and their needs as mentioned in the previous sections, we would reflect upon design and choice of programming languages used to develop software applications for edge computing.

A. Important Features

- Design of programming language
  - The typedness of a programming language plays an important role. Statically typed programs and Dynamical programs have both positive and negative implications which would be discussed.
  - Imperative programs and Declarative programming; Imperative programming is preferred in the applications used for edge computing due to its functional composition.
- Concurrency
  - Concurrency is a programming language property that describes the fact that multiple activities are performed at the same time, and that the activities can interact among themselves in some way.
  - It is very important in this paradigm because of it improves the applications by reducing latency; A unit of work is executed in shorter time by subdivision into parts that can be executed concurrently.
  - Increase throughput;The overall system throughput can be improved by running several tasks at the same time.
- Garbage Collector
  - Garbage collectors identify and recover unused memory on the programmer's behalf, a job that would otherwise be time-consuming and error-prone.
  - Real time services generate a lot of data and manual implementation of deallocation, freeing memory isn't feasible which made Garbage Collectors rose to high importance in the edge computing paradigm.
  - Novel Hardware Garbage collectors have also been a prime interest of research.
- Object Oriented Programming
  - If we were developing a smart home application, the features we would want to include in one space would be almost identical to those we would want to include in another room, with a few exceptions.
  - The abstraction provided by object oriented programming comes to our aid.
  - The encapsulation provided also lets us safeguard our private data. Given that the applications have our

data tramsitted to different places, we wouldn't like other applications to view our private data such as passwords, contact details.

- Provide Security
  - As discussed in the previous sections, security stands a great threat in edge computing paradigm.
  - Some languages provide a great support and flexibility in dealing with malicious trojans, DDoS attacks.

Table 1 describes the features various programming language provides with. The checkmark denotes that the feature is well suitable for applications that use edge computing.

B. C Programming Language

C programming Language, as known is one of the earliest programming languages designed. It is an imperative programming language which basically uses statements that change the program's state. Declarative programming isn't suitable for developing applications for edge computing as the paradigm is more about *what to do* rather *than how to do*. C programming language being statically typed also have an edge over other dynamically typed languages. The advantage of type constraints offer more opportunities for compiler optimizations. The testing time may also be low, but the developing time may balance it.

Almost all the software applications dealing with edge computing have constraints such as low power, low RAM and storage. C language could be a direct solution for writing programs which deal with a lot of hardware. Also by its very nature, the C programming language was designed to be compiled to provide low-level memory access and language constructs that map efficiently to machine instructions with minimal run-time support. C also on being able to run on most of the embedded devices could serve most of our purposes. The community support, available libraries make C an attractive programming language in the field of edge computing.C programming language is used intensively to write and deal with malicious Trojans. Trojans posing a great treat to the edge computing paradigm, C programming language attracts a lot of consumers.Since cyber security teams can disassemble malware to analyse its architecture, distribution, and implications using C language in reverse engineering, it makes it easier to look after the security aspects. On the other hand difficulty in implementing complex data structures, no support to user interface, no exceptional handling and low level of abstraction maybe a problem.

C. JAVA

Java being statically typed, imperative brings the advantages of all the benefits the C programming brings (due to the nature of C being statically typed imperative as discussed in the previous subsection). Along with that Java has been extensively used in all the industries due to its *write once, use anywhere* property. This portable nature makes Java an attractive language to be used. Java is highly interoperable. Interoperability within the cluster of edge devices, nodes and the remote cloud is easily achieved through Java. Java offers an integrated, secure, comprehensive platform for the entire Internet of Things architecture across all vertical markets by providing end to end security, easy integration with IT systems.[20].Embedded systems are the focus of several of the Oracle Java ME Embedded APIs.

The Object Oriented Programming in Java makes the lives of IoT so easy. The encapsulation property also helps us protecting private data. The security aspect can be improved by the use of Object Oriented programming. Java has the ability of automatic up-gradation, which makes it cost-effective and an excellent choice for IoT systems. Garbage Collection has been an important breakthrough in the programming languages paradigm. Garbage Collection (GC), is an automation for a more efficient use of memory, an automation that must be customized according to the target system Java having an inbuilt garbage collector could be an ideal choice. Java on the other hand also has a very large community support and proper documentation.

D. Go programming language

GO (also referred as golang) is a relatively new programming language which first appeared in 2009. Like C, Java it is a statically typed language, functional, imperative. It is also an object oriented program. So it brings in all the advantages provided by the above features as mentioned in the above subsections. One important feature that attracts a lot of developers in the edge computing paradigm is the concurrency the programming language along with the easy implementation style that of python.Garbage collection is an other feature of golang which is an valuable asset to have.Most malware aims to get into target systems undiscovered, making Golang perfect for it to deal.This language also has vast libraries that make the malware creation process very smooth. The pluggins which would be discussed, have a lot of troubleshooting mechanisms which would help further. One of the most interesting thing about goland is the additional pluggins, framework it provides.A third-party transpiler GopherJS is also available which coverts Go to JavaScript for front-end web developement[21]. Gobot is an another framework which is specially developed for rpbots, drones, and the Internet of Things[22].Gobot has a framework for connecting to hardware devices that can be expanded. Currently, 35 platforms for robotics and physical computing are supported which include arduino, ESP8266, Rasberry Pi. It also supports GPIO (General Purpose Input/Output), Analog (I/O) drivers, I2C drivers, SPI drivers. Gobot has a RESTful API that allows you to check the status of any connection, computer, or robot in your swarm. It also has the ability to send commands to your computers and robots directly. Gort, a Command Line Toolkit (CLI) provides tools to scan for connected devices, upload firmware such as aurdino and it works perfectly with Gobot. Golong though having a very detailed documentation has no big community support which may turn up into a problem.

TABLE I
PROGRAMMING LANGUAGES AND THEIR FEATURES

	Statically typed	Imperative	Compiled	OOPS	Garbage Collector	Error handling	Hardware Supp	Packages	Community support	Security
C	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓
Java	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗
Golang	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Python	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓
Javascript	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓
Rust	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
C#	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓

E. *Python*

Python is an imperative program like C, Java but is an dynamically typed program. The dynamical typing of python brings a lot of flexibilty in the functionality of program but at the cost of large testing time. Python also being a high level programming language reduces the developing time. Python like Java is portable. Python also being an interpreted language doesn't have to compiled. Prototyping is easier. Python like Java is an object oriented programming language (abstraction and other properties make a great difference).

Data handling in python is very easy and can easily handle the enormous amount of data generated near the end users. Python also has a lot of modules,libraries for data visualization which attracts a lot of users.Extrinsic factors plays a big role in choosing python over other programming languages. Intrusion detection system (IDS) that monitors for malicious traffic and detects policy violations can serve as effective countermeasures against routing information attacks. Implementation of IDS is very simple, as various machine learning algorithms are used. Python also automates tasks and performs malware analysis along with that an extensive third-party library of scripts is readily accessible, meaning help is right around the corner. Attractive IDEs like pycharm, enormous community support along with every almost all libraries/modules make python an atteractive choice. Python being slow in nature and also the large testing time associated with dynamically typed programming languages is surely a problem.

F. *Other programming languages*

1) Rust: Rust again is a statically typed, imperative, concurent functional programming language.Rust offers developers speed, safety, high level abstractions and an efficient development environment. Rust is also a low level language which suits IoT devices as discussed with the C programming. Rust is also a memory-safe system programming language which also provides low runtime overhead and fine-grained memory management. It is a programming language with strong safety guarantees which prevents memory corruption and has the potential to solve the problems that can occur when using C language.(It used the Data ownership model)Though they are separate books on Rust and Internet of Things, rust not being familiar with the community and also not completely functional may be a bad sight for few developers.

2) Javascript: Javascript is a dynamically, weakly typed programming language which again gives us a lot of advantages with the flexibility nevertheless may also give rise to negative implications. Javascript is the langauge if we want to capture cookies, exploit event handlers, and carry out cross-site scripting. This greatly improves the security of the system. Node.js attracts developers all over the world to use the Javascript for developing applications which may use edge computing. Node.js has been seemlessly programmed to work with dynamically changing data. Data collected from sensors in IoT, WSN can be easily managed with node. IoT protocols such as sockets and MQTT (Message Queuing Telemetry Transport used for transporting small messages) are easily integrated to Node.js. The Node Package Manager has an enormous number of packages available for modules needed to deal with arduino, rasberry pi , other hardware. Javascript also which is one of the most used languages for frontend stands as an attractive choice. Weakly typed programming has brought a lot of side effects in Javascript which surely is a problem.

VI. PROGRAMMING LANGUAGE FOR EDGE SERVERS/GATEWAYS

In this section we'll give a brief about the programming languages that can be used for edge servers/gateways.

A. *What are edge servers?*

Edge servers are servers that perform processing at a point near the edge, which can be anywhere on the spectrum. Another word often used is edge nodes, which may refer to both a wider collection of computing resources (including end-devices) and a cluster of edge servers. The servers handle communication and data processing for a variety of devices connected to various networks.

B. *Java Programming Language*

Java as discusses in the previous sections has been used extensively in many industries due to its robustness, flexibility and portability. The object oriented programming style can be used to create modular and reusable code. Java also provides powerful frameworks to support microservers and enormous large programs(Spring Boot). Spring Boot makes it easy to create applications. It examines our classpath and the beans we've set up, makes educated guesses about what we're

missing, and then adds those things. Java acing in all the extrinsic factors also attract a lot of developers, software.

C. C# (C-Sharp)

C# (C-Sharp) is a programming language developed by Microsoft that runs on the .NET Framework. It is a statically, strong typed language, imperative functional programming language. In simple words it a simple, object oriented and type safe programming languages. It is quite robust and scalable. C# allows the the design of reliable and useful applications. It also provides a lot of security using the in-Built authentication. C# is used in the .NET developing platform for building different type of applications. .NET Core is a cross platform .NET implementation which also uses C#.NET for edge development is known for its high performance and efficient use of computer resources. Concurrency, parallel algorithms, and composability are all supported well by the platform.

VII. Conclusion

The choice of programming language depends on a lot of aspects based on the demanding needs of the applications that are being developed. It has to be accepted that no programming language is perfect and all are built to support specific needs which may have negative implications on other needs. The choice shall be based on the needs such as low latency, security and mapped according to the various intrinsic and extrinsic properties of the programming languages.

Acknowledgment

The authors would like to thank Professor Sujit Kumar Chakrabarti along with the Teaching Assistants Amogh Johri, Anirudh C and Rahul Murali Shankar for their enormous support and guidance throughout the project and the entire course.

References

[1] S. Namasudra, P. Roy and B. Balusamy, "Cloud Computing: Fundamentals and Research Issues," 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), Tindivanam, 2017, pp. 7-12, doi: 10.1109/ICRTCCM.2017.49.

[2] M. Satyanarayanan, "Edge Computing," in Computer, vol. 50, no. 10, pp. 36-38, 2017, doi: 10.1109/MC.2017.3641639.

[3] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi and A. Leon-Garcia, "Fog Computing: A Comprehensive Architectural Survey," in IEEE Access, vol. 8, pp. 69105-69133, 2020, doi: 10.1109/ACCESS.2020.2983253

[4] Ms. Dalbina Dalan.An Overview of Edge Computing.International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Volume 7, Issue 05

[5] Gezer, Volkan & Um, Jumyung & Ruskowski, Martin. (2018). An Introduction to Edge Computing and A Real-Time Capable Server Architecture. International Journal of Intelligent Systems. 11. 105.

[6] https://www.ibm.com/cloud/architecture/architectures/edge-computing

[7] Ton Steenman, "Accelerating the Transition to Intelligent Systems", Intel Embedded Research and Education Summit, February 2012

[8] https://en.wikipedia.org/wiki/Digital_Revolution

[9] https://epodcastnetwork.com/the-importance-of-programming-languages/

[10] A. A. García, D. May and E. Nutting, "Garbage Collection for Edge Computing," 2020 IEEE/ACM Symposium on Edge Computing (SEC), 2020, pp. 319-319, doi: 10.1109/SEC50012.2020.00044.

[11] Z. Song and E. Tilevich, "A Programming Model for Reliable and Efficient Edge-Based Execution under Resource Variability," 2019 IEEE International Conference on Edge Computing (EDGE), 2019, pp. 64-71, doi: 10.1109/EDGE.2019.00026.

[12] T. Severin, I. Culic and A. Radovici, "Enabling High-Level Programming Languages on IoT Devices," 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2020, pp. 1-6, doi: 10.1109/RoEduNet51892.2020.9324882.

[13] T. B. Chandra and A. K. Dwivedi, "Programming languages for Wireless Sensor Networks: A comparative study," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 1702-1708.

[14] J. A. Stankovic, "Wireless Sensor Networks," in Computer, vol. 41, no. 10, pp. 92-95, Oct. 2008, doi: 10.1109/MC.2008.441.

[15] Khanna, A., Kaur, S. Internet of Things (IoT), Applications and Challenges: A Comprehensive Review. Wireless Pers Commun 114, 1687–1762 (2020).

[16] T. B. Chandra and A. K. Dwivedi, "Programming languages for Wireless Sensor Networks: A comparative study," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 1702-1708.

[17] https://shiverware.com/iot/iot-vs-wsn.html

[18] A. Elsts and L. Selavo, "A user-centric approach to wireless sensor network programming languages," 2012 Third International Workshop on Software Engineering for Sensor Network Applications (SESENA), 2012, pp. 29-30, doi: 10.1109/SESENA.2012.6225731.

[19] D. Bacon, P. Cheng, and D. Grove. "Garbage Collection for Embedded Systems". pages 125–136, 01 2004.

[20] https://www.oracle.com/technical-resources/articles/java/java-maker-iot.html

[21] https://en.wikipedia.org/wiki/Go_(programming_language)

[22] https://gobot.io

[23] Angel Zúñiga, Gerardo Sierra, Gemma Bel-Enguix, Javier Gomez, SICIoT: A simple instruction compiler for the Internet of Things, Internet of Things,Volume 12,2020,100304,ISSN 2542-6605, https://doi.org/10.1016/j.iot.2020.100304.

[24] (https://github.com/azuniga-ii/sicIoT

[25] https://github.com/contiki-ng/contiki-ngb

[26] http://anrg.usc.edu/contiki/images/a/a8/Sensor-acq.docx

[27] https://www.trentonsystems.com/blog/is-edge-computing-secure

Appendix

A. Study on Simple Instruction Compiler, SICIoT

In this section we would explain the work done in [23].

With the assumption that IoT will play a revolutionary role in people's future and especially those who have very limited knowledge on technical aspects, the authors wanted to build a simple instruction compiler that could translate the words of the user (verbal instructions) to an IoT device functionality. The compiler was called SICIOT which stands for Simple Instruction Compiler for the Internet of Things.

The authors felt that the IoT paradigm should demand minimum user interaction but at the same time meet all the needs, so the idea of SICIoT came into existence. To programme an IoT/WSN computer, a person must be familiar with the device's programming language, which means they must be programmers. This requirement, however, would be insufficient because the programmer must be aware of specific technical information for this type of system. As a result, only a small number of people who become expert IoT/WSN programmers are able to programme IoT/WSN devices.

So the basic idea is that the user (who is assumed that has no skills with IoT/WSN) uses the compiler proposed and the compilers tells the respective IoT setup (the hardware setup) what to do. For example let us thing we have setup IoT devices to know the hardness of water in the big water heater we have. Heating hard water would cause scales in the pipe which causes a great damage. So we need to inform (via a mail or a

message) to the operators to not heat the water when the water is hard. But we only have the set up ready and have no skills to programme. The SICIoT comes to our aide. If we could tell verbally to SICIoT that "Mail to operator@email.com when the hardness of water is above x", the SICIoT would generate the code for the functionality.

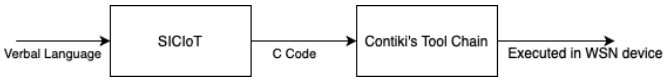

Fig. 4. Compiler overview

The target language for SICIoT was decided to be C as it is used as an intermediate programming language for many compilers. Contiki, a widely used WSN operating system, will take the C code and will function it in WSN device as shown in figure 4.

1) Compiler Architecture: Like the classical architecture (as shown in figure 5) the Lexical Analyser of compiler takes in a source program and produces tokens. The tokens would

Fig. 5. Classical architecture

be try to kept as generic as possible. For example, *too hard* and *hard* would give the same token. The tokens are sent to the parser and parser creates an AST (abstract syntax tree).AST is basically tree represntation of CFG where the

- Grammar symbol → Nodes.
- Starring Symbol → Root Node
- Non-terminals → internal nodes
- Terminals → leaves
- Productions → Edges

. Then different types of analyzers are selected based on their time of computation as follows:

- Recursive Descent
- Top Down Predictive
- Bottom-up Predictive

Generally, the AST generated by the parser is then fed into the semantic analyser which provides us with an annotated AST. We try to find and filter out the semantic errors here. The process would allow us incorrect sentences such as *"heat the mail operator@email.com"*.But the Semantic analyser wasn't yet created in SICIoT which mean *"heat the mail operator@email.com"* would give some random behaviour. The code generator take the AST (not Anotated AST) and then produces output code. Due to no big research in the domain, translation of each natural language to some predefined corresponding target was done. The authors revealed that connecting the natural language semantics with programming langauge semantics was a uphill task. The software tools used by the authors were; flex for the lexical analyzer;Bison the parser (or the syntatic analyzer)

Let us now discuss about the sentence "Notify the operator", firstly tokens Verb(Notify), Det(the), noun(operator) from lexical analyzer is given to the semantic analyzer where the tree is formed. The AST is shown in figure 6.

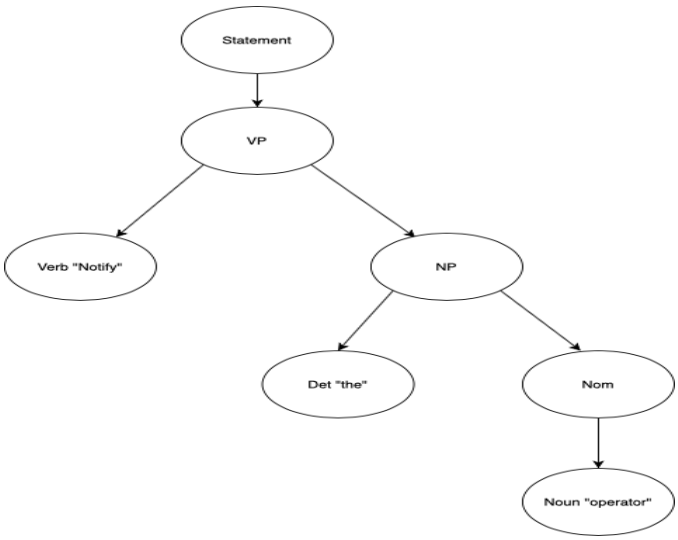

Fig. 6. Tree

(VP) - Verb phrase, (NP) - noun Phrase, (Nom) - Nominal. The CFG (context free grammar) then formalizes the rules. Then the tree is traversed for code generation.The authors use the prototype mentioned in [24][25]. The authors presented many cases and demostrated them. They were confident that the compiler prototype could be extended further to support more complex IoT applications. The SICIoT was restricted to the WSNs which use Contiki as OS. It has to be extended to make it compatiable with other devices like Arduino, Rasberry Pi. It was indeed interesting to know that works have been going on in this field.