Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
December 2020, vol. 25, no. 2, pp. 117–123
https://doi.org/10.2478/acss-2020-0013
https://content.sciendo.com

Lightweight Coordination Patterns for Applications of the Internet of Things

Waseem Akhtar Mufti*
Aalborg University, Aalborg, Denmark

***Abstract* – Applications of the Internet of Things (IoT) are famously known for connecting devices via the internet. The main purpose of IoT systems (wireless or wired) is to connect devices together for data collection, buffering and data gateway. The collected large size of data is often captured from remote sources for automatic data analytics or for direct decision making by its users. This paper applies the programming pattern for Big Data in IoT systems that makes use of lightweight Java methods, introduced in the recently published work on ClientNet Distributed Cluster. Considering Big Data in IoT systems means the sensing of data from different resources, the network of IoT devices collaborating in data collection and processing; and the gateways servers where the resulting big data is supposed to be directed or further processed. This mainly involves resolving the issues of Big Data, i.e., the size and the network transfer speed along with many other issues of coordination and concurrency. The computer network that connects IoT may further include techniques such as Fog and Edge computing that resolve much of the network issues. This paper provides solutions to these problems that occur in wireless and wired systems. The talk is about the ClientNet programming model and its application in IoT systems for orchestration, such as coordination, data communication, device identification and synchronization between the gateway servers and devices. These devices include sensors attached with appliances (e.g., home automations, supply chain systems, light and heavy machines, vehicles, power grids etc.) or buildings, bridges and computers running data processing applications. As described in earlier papers, the introduced ClientNet techniques prevent from big data transfers and streaming that occupy more resources (hardware and bandwidth) and time. The idea is motivated by Big Data problems that make it difficult to collect it from different resources through small devices and then redirecting it. The proposed programming model of ClientNet Distributed Cluster stores Big Data on the nearest server coordinated by the nearest coordinator. The gateways and the systems that run analytics programs communicate by running programs from other computers when it is essentially required. This makes it possible to let Big Data rarely move across a communication network and allow only the source code to move around the network. The given programming model greatly simplifies data communication overheads, communication patterns among devices, networks and servers.**

***Keywords* – Analytics, Big Data, ClientNet, cluster, IoT, methods, orchestration.**

I. Introduction

Internet of Things (IoT) [1], [2], [3], [4] is an emerging technology, which is applicable in almost every aspect of human life, such as Smart Systems [5], Transportation [6], e-Businesses [7], e-Health [8], Geological Systems [9], Aeronautics [10], Industry [11], etc. The use of IoT has merged different technologies enabling engineering, science, arts, business, indoor/outdoor living style to benefit more than what it used to be. The basic idea behind IoT is not very much complex. The idea is to capture the information, initially raw data, of each object (an object may be physical, or abstract that may be an event happening inside computers, machines or living things including humans) around the environment and make the outcomes from its analytics.

The capturing of raw data is to attach sensors [12] with physical or abstract objects to detect their current state or outputs. These sensors are of different types such as: sensors for temperature, pressure, humidity, light, radiation, camera or lasers for object movement, water levels, sounds, chemicals, odours, flavours, weight, electronic signals, computing processes and their exceptions, conditions, values and flags during the transactions, e-business chains or scientific data processing. The next step is to implement the system that can take the data into computers, clouds or inputs to stored programs embedded into electronic chips for decision making or further processing. Verbally the discussed scenario is not difficult to visualise but the implementation of such a system raises many technical issues that require computing and engineering skills and the ability to understand those physical or abstract and non-computing systems under consideration.

Most of the problems are faced due to the involvement of different fields, e.g., electronics, mechanics, architecture, computer programming, communication networks etc. For instance, the technical issues are: the compatibility of sensors with the objects under consideration, connectivity of IoT platforms [13], [14] with sensors and computers, deciding the appropriate types of data collected from sensors, the complexity of dealing with speed, size and other issues of resulting Big Data [15], [16], the design of communication network (wireless and wired) connecting all the actors, the issues with integrating the complete system with cloud or remote servers and the

* Corresponding author's e-mail: wmufti@gmail.com

number of sensing devices distributed across different locations if it is a big size industrial unit.

This paper is one of the efforts giving the possible solutions to some of the issues mentioned above: (1) The Big Data issues especially in Wireless Networks and (2) The design of Efficient Communication Network that connects computers, IoT platform (software and hardware) and the sensors. Since the IoT system in a complex industrial environment is weaved in wireless networks, this paper focuses on the solutions for wireless systems. More often the wireless systems (short range or long range) are sensitive to network capacity and the size of data, and both factors are directly related to how the devices are programmed to communicate. Considering the given fact, these issues can be generalized as well as their source as the different **patterns of coordination** among all actors involved in IoT system, environment and its actual beneficiaries. The generalization of these issues, i.e., patterns of coordination would require the solutions that can exploit the existing computing resources and new methods of data access and processing. The generic solution to these problems is related to the way the different components of IoT system interact with each other. Thus, if the communication interactions are added with more efficient algorithms, the overall system would become efficient enough.

The author believes that the lightweight communication techniques can solve the problems of a very complex Industrial IoT (IIoT) system [17], [18]. The proposed lightweight communication techniques are in fact the communication patterns of point-to-point data transfers leveraging the programming constructs of Java used in similar situations, e.g., Java Remote Method Invocation (RMI) [33]. Moreover, the algorithmic approach used to achieve lightweight computing makes the best use of famously known sensor network designs: Fog and Edge computing. The details of RMI and the function calling model, which is inspired by a classical visitor pattern, is not given in this paper. To understand this paper, it is advised to have a look on these concepts; however, a reader with basic understanding of distributed systems can understand the communication model in Figs. 1 and 2, and the programming model given in Fig. 3.

II. Advantages of Lightweight Coordination Patterns

Given the limitations of a wireless network connecting IoT devices where there is possibility of thousands of sensor devices attached with different physical units or environment and generating terabytes of Big Data, thinking of efficient data acquisition is not possible without efficient data capturing and processing algorithms. Wireless IoT systems impose critical limitations on heavy data transfer, battery power, storage and processing. The efficient network design and algorithmic solutions can have a positive impact on the limitations of IoT systems in wireless networks. The proposed lightweight coordination patterns with its Edge and Fog models and powerful IoT platform can minimise traffic in a wireless network allowing for greater throughput and fewer bottleneck issues.

Taking the full advantage of Edge computing, the lightweight computing techniques can utilise the Raspberry Pi [31] (one of the commercially available IoT platforms) allowing for more processing at the bottom level of the communication stack. These devices are capable of running programs, e.g., to parse the raw data right on the sensing sites that can reduce the network traffic and data processing time for upper layers of the system. This activity also prevents unnecessary data to float on the wireless network, thus saving bandwidth and time to transfer data from sensors to Edge to Fog and to Data centres or cloud. Moreover, since the IoT platforms are attached nearest to the sensors, these devices have limited processing and storage capacity. This issue can be solved by enabling Raspberry Pi to download the Java programs from its server and run like visitor functions.

This saves network bandwidth because data do not float on the network instead visitor functions are called. This is one of the remarkable applications of coordination patterns that makes use of classical visitor patterns of object-oriented software design [19], [20].

III. IoT Communication Model

A communication model for IoT is given in Fig. 1. It has five layers, which are responsible for data transfer from the first step of data sensing to the last step of sending data to data centre or cloud. The **Physical layer** represents the sensor network where each sensor senses data from a device or the environment it is attached with. These sensors of different types come with different hardware brands and APIs compatible to their supporting IoT platforms. Therefore, the physical layer includes sensors and connectivity with IoT platform, e.g., Raspberry Pi [13], [14], Arduino [21], etc. The connection of sensors with IoT platform is normally hardwired, or uses wireless protocols such as LTE [22], LP-LAN, WiMAX, Wi-Fi, or any wireless personal network. The IoT platforms are packaged with their own operating systems, applications and compilers of different programming languages depending on the capacity of hardware board. Finally, the sensor network is attached with an IoT device and power supply.

In an industrial environment an **Edge layer** [23] performs initial tasks, e.g., instead of sending raw data to a cloud or fog server, it executes small tasks of assembling data into a particular format. The initial data may be the sensor id, device code, IoT platform IP address and name on Wireless LAN, data items (values and types), time stamps of data capture and other parameters like GPS data (most probably with moving objects, e.g., in Supply Chain Systems or Transportation). These segments of data form tuples that can be transformed into XML, JSON, Java objects or other formats portable to anywhere on LANs, data lakes or generally anywhere on the internet. In this paper, Java objects are used at Edge and the other formats such as JSON, etc, are discussed in case of cloud or data centre uploads.

Since IoT platforms are lightweight units, it is possible, to provide more space to store the collected sensor data. There should be a supporting file/database server that can dump large sets of history events. It is because in most of the cases the rate

of data collection is often faster than that of its processing, for instance, the temperature of a device is collected every couple of seconds. If these data sets are directly transmitted to fog in real-time, then no matter how fast the network is there are chances of bottle-necks. The efficient design requires supporting a server by directly serving the network of wireless IoT devices and allowing for the storage of the bulk data from where the data must be streamed to fog servers. This will balance the edge-fog communication and prevent overflow conditions at the edge. However, latest Raspberry devices can work as a server.

Fog layer [24] receives pre-formatted data from IoT platform most of the time in a wireless network (WLAN). Since it is TCP/IP communication, IoT devices must form streaming or socket connections with a fog server. This is done by application programs written for application-to-application data transfer in a wireless or wired network, e.g., in this case it is Java RMI based communication. The servers, in a local area, dedicated for a fog layer include database and application servers to process and store the transactional data required for on-premise data filtering, map counting, aggregations and classification into different categories. These activities help reduce network traffic, efforts and computing resources while uploading the resulting big data up in the cloud. Users may also use fog servers for data evaluation, i.e., querying or spontaneous data analytics for internal use.

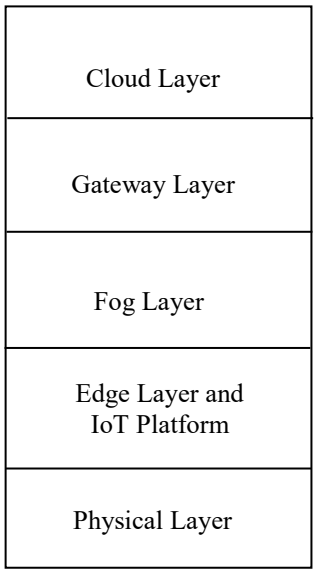

Fig. 1. IoT data communication model. Edge-Fog-Cloud communication stack that defines the five stages of message passing and coordination.

The **Gateway** is buffer between fog servers and cloud. Once the data is processed in the fog, it is then decided exactly how much data size and what data items are needed to send on which type of cloud, or directed to some other destination. Before sending to the cloud if the user wants to delete some data, inject some more data or update, then the cloud receives only valid data, which are the only required data sets for global analytics. In other words, gateway servers facilitate cloud to feed it what exactly is required to upload, where to upload, or change the decision in anyway accordingly.

IV. Edge-Fog Lightweight Communication

This section discusses how the fog and edge computing can be helpful in an industrial scenario. The paper presents the basic design of fog, edge and Raspberry Pi platform in an IIoT setup. Then it is discussed how the idea of visitor functions can leverage Java RMI to provide thin IoT clients and the servers. The resulting system aims at providing lightweight data transfer and processing in IIoT-WLAN. The basic idea is to divide data processing functionalities into elementary and secondary processing through an intermediate or bridge server to balance between data ingestion and data transfer from edge to fog. This intermediary server or bridge provides buffering to prevent bottle-necks in WLAN and data overflow at edge; (see Fig. 2).

Industry can leverage the power of IoT in different use cases, e.g., asset tracking and monitoring, predictive maintenance, improving safety and security, energy efficiency for buildings, supply chain systems, automation of manual processes, etc. To get into the idea and its applicability in IIoT, this paper does not provide a particular use case [32]. The aim is to understand IIoT programming model for lightweight communication patterns based on function calling in Edge-Fog layers applicable for all industrial use cases.

This model executes on top of the wireless communication model, which comes with different protocols and hardware brands. All the use cases involve sensor reading in one or another way. Therefore, this paper assumes a generic scenario where different types of sensors are connected with Raspberry Pi as given in Fig. 2. Since the Pi can run network-based Java programs and contain the storage, this device best suits the aim of the proposed distributed programming model for IoT and Big Data.

V. Edge and Fog in Industrial Internet of Things (IIoT) Architecture

Figure 2 is a generic industrial scenario of IoT platform in Edge-Fog configuration in Wireless LAN. The given network is scalable to any number of nodes depending on the size of data generated by sensors. The bottom layer contains sensors of temperature, humidity, pressure, light and image. The number of sensors can be a network of thousands of sensors attached at some positions on physical objects. As the number of sensors grows, the number of bridge servers grows accordingly depending on the size and speed of data.

In any of the IoT setup whether it is a smart building or industrial unit, the wireless network selection is almost similar. The size of network is important because it means the number of sensors and their corresponding physical objects. If the area is short, then Wi-Fi based high speed WLAN routers are suitable, or if it is in miles then WiMAX and LTE are a good choice. In both scenarios, the size of generated data is the main factor to consider because it guides the choice of network type and the programming model to deal with big data issues. If data is in terabytes, then huge data cannot come to the network to make the network slow.

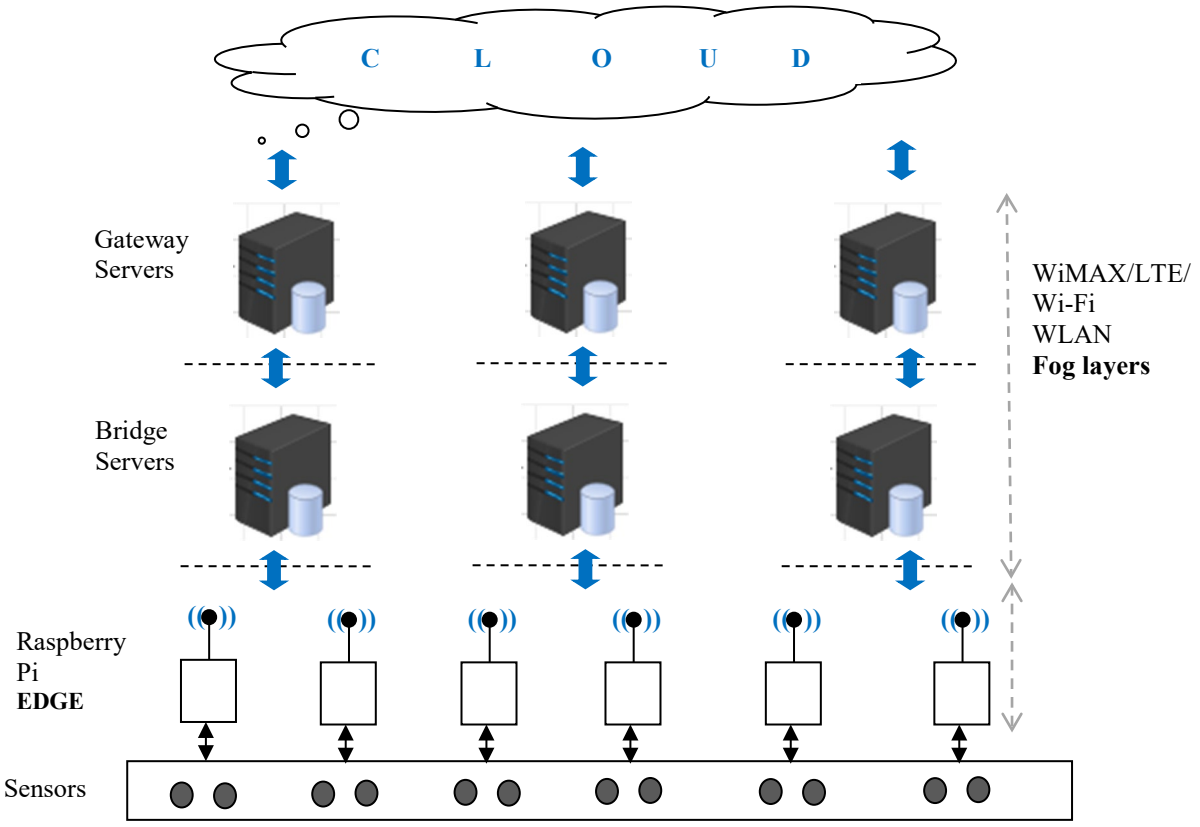

Fig. 2. IIoT platform in Edge-to-Fog WLAN-to-Cloud. It is the implementation of a 5-level lightweight communication model.

VI. Thin IoT Client Programming: Visitor Functions and Java-Based Networking

The use of Raspberry Pi allows easy accessibility of data because it is made exactly for the same reason. This platform can run lightweight Java applications to capture sensor data and send it directly to corresponding bridge servers. For example, the raw data from temperature sensors includes: sensor id, temperature value, time of data capture. Normally there are only few data items, for instance, temperature value and time stamp, where the corresponding sensor id can be generated by the program itself. The data items are then encapsulated in an object and forwarded to a bridge server through a remote method call by invoking the function of a bridge server that accepts an object as a parameter. The number of times the remote method is invoked (within WLAN in this case) is equal to the number of data capture events for a particular sensor. A sensor can generate data in milliseconds, seconds or hours that depends on the type of an object under consideration. Figure 2 does not show the details of wireless routing devices connecting different nodes. The devices with newly available protocols, which are discussed above, are highly scalable to add or remove more nodes dynamically.

For example, many laptops can connect to a Raspberry Pi device by creating a Wi-Fi network, or many Pi devices can connect to their wireless gateway (short rage or long range) that can connect with fog servers. One of the main factors that should be considered when selecting the type of a network is the size of the generated data by sensors. The lightweight programming model for industrial use cases where big data is generated is given in Fig. 3. In this model, a client makes remote calls on a bridge server and sends sensors data as an object through a function argument.

If the number of objects increases (depends on the number of sensors), more clients are created. This is a software architectural design decision that is made based on the specification of the physical system of sensors. This programming model is flexible enough to add or remove clients according to the requirements without major changes in other programs. The given programming model becomes more complicated if it is extended with more computing nodes. It is when many clients would concurrently communicate with servers involving more objects, each associated with sensors and data. A client may be an autonomous object responsible for communicating on its own or a client may be a GUI page that can be operated manually by users to decide when to transfer data.

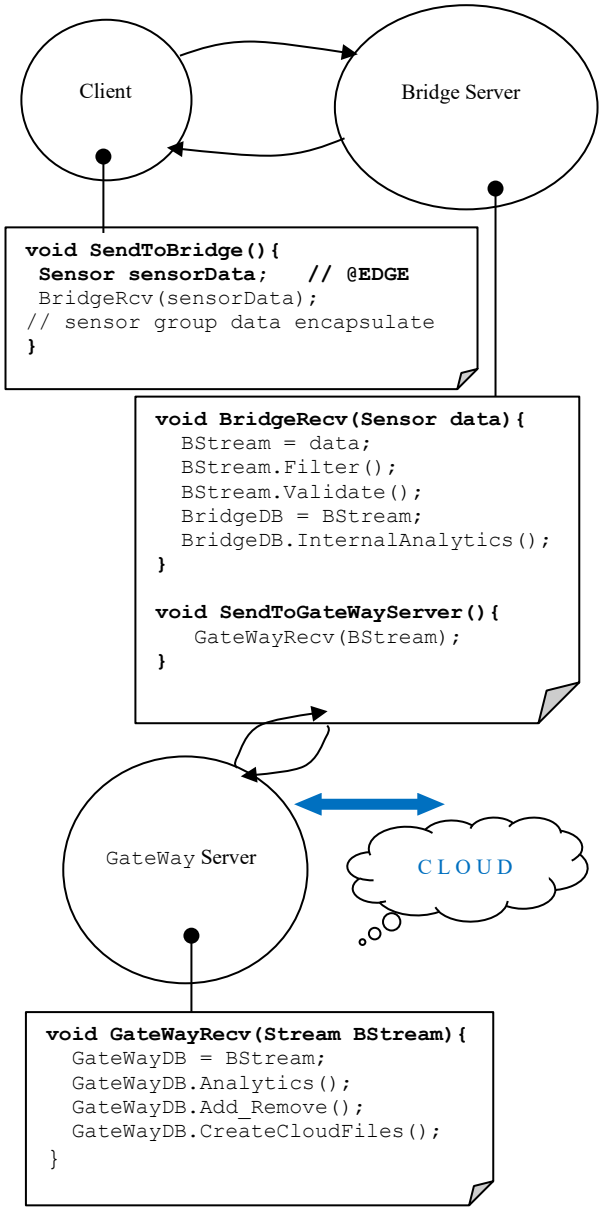

Fig. 3. Lightweight IoT clients and visitor functions. Code level implementation of a 5-level lightweight communication model.

It is worth noticing how the system of computing nodes over a wireless network provides lightweight communication through Java RMI and the semantics of visitor functions. The two programming methods help maintain the controlled packet traffic on a wireless medium by not allowing the heavy data transfer. For instance, the **Sensor data** is defined in a **Client** (program that runs on Pi) available at the **Edge**. Simply the function of Bridge is invoked that receives Sensor data as a function argument. This is implemented by establishing streams between different computing nodes (clients and servers) and maintaining local databases as buffers or keeping the history of transactions. As a result, the collected moderate size of big data is applied, such as filters, redundancy deletions, validations and analytics functions (Fig. 3). This is achieved by deploying visitor functions on these nodes. The visitor functions perform the necessary processing and then transform data sets into final acceptable data to be uploaded on a cloud. The visitor functions can also be fetched as an on-demand service from other dedicated servers. Therefore, it is all about software architecture consideration where to deploy the functions and when to access the on-demand functions for analytics and other services. Once these decisions are taken, the implementation becomes automatic and flexible. The interactions between computing nodes is completely based on the idea of lightweight functions, which contain only the source code that prevents the wireless network from heavy data.

The idea behind 5G and future generation networks involves rigorous use of data processing and filtering on wireless base stations [25], [26]. This makes the base stations smarter, i.e., they consume more electric power and generate more radiation threatening the life on the earth. The given programming model can reduce the use of heavy data transfer and, therefore, would consume less power and generate less radiation.

Finally, the file objects are constructed at the top of the fog servers (gateway nodes) to upload on a cloud. The file objects, e.g., JSON, CSV, XLS/XLSX, Plain Text, LOG, TSV, Parquet Avro, BigQuery Tables, etc. can be created in Java or imported from other third-party applications. The network-based applications developed in Java are flexible enough to connect with any of the remote data lake by exchanging the common file formats. Therefore, if it is required; to support the overall efficiency and throughput, the given programming model can leverage Java's interoperability with newly created databases and streams.

VII. Discussion

Considering the interoperability of Java with other formats given above, the Java Object Serialization and De-serialization enable applications to connect with other formats. There are plenty of Java parsing APIs and JDBC drivers available for the file types discussed above to serialize into Java objects or de-serialize to convert strings, thus creating new file format objects. Moreover, Java can run applications developed in other languages, e.g., C, C++, C# and Python, etc. On the other hand, Java Native Interface (JNI) and Web Services can communicate any of the applications running on other platforms. JNI can go deeper into cross-platform (hybrid platform) hardware level to extract data from hardware controllers and electronic interfaces by running a native code of that hardware.

It is only the matter of requirements and specifications of the domains where IoT is deployed. IIoT is not different from other domains of IoT. All the application tools and wireless connection protocols and devices used in one domain of IoT are applicable in other domains as well. The only difference is the types of sensors and supporting circuitry that can differ from domain to domain, e.g., sensor kits used for smart buildings must be different from those used in the agriculture domain due to the adaptability within that environment.

VIII. Related Work

The idea of lightweight coordination patterns presented in this paper is inspired by the ClientNet Cluster [27]. The solution to transferring Big Data sets using a mobile code is given in

ClientNet Distributed Cluster model leveraging the full potential of Java network programming. This paper uses some of the components, e.g., Client/Server and visitor functions from ClientNet. The system is highly scalable to any number of computing nodes depending on the specifications and requirements. This can also be considered a generic distributed programming model that can be used to implement Edge-Fog-Cloud architecture.

In the literature of IIoT in Edge-Fog-Cloud [28], [29] different approaches are given to solve the similar problems of Big Data. In all of the given architectures, people have tried to ingest data as much as possible at Edge, which is similarly given in this paper, but the key difference is no programming model is clearly given in other papers. For example, as given in the present paper, the use of visitor functions and the typical deployment of these functions according to the layer of Fog servers is not given anywhere in the known literature or commercial implementations.

The famously known protocol MQTT [30] provides real-time data acquisition, but it does not provide lightweight distributed data processing within WLAN or similar networks. However, it can reduce the network traffic near a cloud layer but the way of dealing with bottle-necks and latency issues at low layers (Fig. 1) is missing in most of the research and commercially available protocols.

Moreover, the proposed lightweight distributed programming model shares similar characteristics of 5G networks such as data-intensive processing near a base station. This functionality is deployed on a gateway server, which can be considered one of the component servers of a base station because the communication tower that connects IIoT system with cloud is attached directly with gateway servers. These servers are also able to receive visitor functions or data sets from other local or remote computers (may be some data lake).

IX. Conclusion

The main contribution of this paper is defining a data communication model of Industrial IoT and IoT, in general, and proposing the algorithm for a lightweight interaction model for Big Data in Edge-Fog-Cloud communication stack. The proposed lightweight model is object oriented based on visitor function invocations on local and remote computers. The communication takes place using Java RMI and the visitor functions are deployed on distributed computers in such a way that Big Data does not move rather the functions are passed or invoked with objects as parameters.

The functions return results (data sets) which are light enough to communicate on a wireless network. Initially data is generated from sensors, defined by Edge, which is passed by making Java streams to the Fog servers for data ingestion. The streams are passed as references and the Fog functions are invoked from thin clients of Raspberry Pi to delegate heavy processing on a Fog server. Similarly, the fog servers are connected to another level of data ingestion and manipulation servers, which are called GateWay Servers.

These servers further perform the tasks that filter data to keep only the real data sets to upload on a cloud. The proposed distributed programming model, careful network design and the efficient and pre-planned deployment of visitor functions across all over the nodes can be useful in decreasing the traffic on a wireless network that can solve Big Data problems.

Acknowledgment

I would like to thank for most of the knowledge available online, which saves time in understanding the philosophy of computing. The cutting edge information published by companies and personal blogs are very much helpful in solving many of the difficult problems.

References

[1] S. Kraijak and P. Tuwanut, "A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends," in *11th International Conference on Wireless Communications, Networking and Mobile Computing*, IET, 2015, pp. 1–6. https://doi.org/10.1049/cp.2015.0714

[2] C. Wang, M. Daneshmand, M. Dohler, X. Mao, R. Q. Hu, and H. Wang, "Guest editorial - special issue on internet of things (IoT): architecture, protocols and services," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3505–3510, Oct. 2013. https://doi.org/10.1109/JSEN.2013.2274906

[3] S. K. Lee, M. Bae, and H. Kim, "Future of IoT networks: A Survey," *Applied Sciences*, vol. 7, no. 10, article number 1072, Oct. 2017. https://doi.org/10.3390/app7101072

[4] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (IoT): A Literature Review," *Journal of Computer and Communications*, vol. 3, no. 5, pp. 164–173, May 2015. https://doi.org/10.4236/jcc.2015.35021

[5] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of industrie 4.0: An Outlook," *International Journal of Distributed Sensor Networks*, vol. 12, no. 1, Jan. 2016. https://doi.org/10.1155/2016/3159805

[6] X. Lin, C. Zhou, and J. Song, "IoT-based early-warning system of unsafe behaviors for horizontal-transportation in shield tunnel," *China Safety Science Journal*, no. 11, pp. 109–115, 2014.

[7] D. Bilgeri and F. Wortmann, "Barriers to IoT business model innovation," in *13. Internationale Tagung Wirtschaftsinformatik*, Universität St. Gallen, 2017, pp. 987–990.

[8] M. Almulhim and N. Zaman, "Proposing secure and lightweight authentication scheme for IoT based E-health applications," in *2018 20th International Conference on Advanced Communication Technology*, IEEE, 2018, pp. 481–487. https://doi.org/10.23919/ICACT.2018.8323801

[9] K. Ryu, Y. Koizumi, and T. Hasegawa, "Name-based geographical routing/forwarding support for location-based IoT services," in *2016 IEEE 24th International Conference on Network Protocols*, IEEE, 2016, pp. 1–6. https://doi.org/10.1109/ICNP.2016.7785321

[10] U. Durak, "Flight 4.0: The Changing Technology Landscape of Aeronautics," in *Advances in Aeronautical Informatics*, (U. Durak, J. Becker, S. Hartmann, N. S. Voros, Eds.), Springer, 2018, pp. 3–13. https://doi.org/10.1007/978-3-319-75058-3_1

[11] J. H. Kim, "A Review of Cyber-Physical System Research Relevant to The Emerging IT Trends: Industry 4.0, IoT, Big Data, And Cloud Computing," *Journal of Industrial Integration and Management*, vol. 2, no. 3, Sep. 2017. https://doi.org/10.1142/S2424862217500117

[12] X. Su, H. Zhang, J. Riekki, A. Keränen, J. K. Nurminen, and L. Du, "Connecting IoT Sensors to Knowledge-based Systems by Transforming SenML to RDF," in *5th International Conference on Ambient Systems, Networks and Technologies and 4th International Conference on Sustainable Energy Information Technology*, Elsevier, 2014, pp. 215–222. https://doi.org/10.1016/j.procs.2014.05.417

[13] J. D. Brock, R. F. Bruce, and M. E. Cameron, "Changing the world with a Raspberry Pi," *Journal of Computing Sciences in Colleges*, vol. 29, no.2, pp. 151–153, Dec. 2013.

[14] H. Chaudhari, "Raspberry Pi Technology: A Review," *International Journal of Innovative and Emerging Research in Engineering*, vol. 2, no. 3, pp. 83–87, 2015.

[15] B. Furht and F. Villanustre, "Introduction to big data," in *Big Data Technologies and Applications*, (B. Furht, F. Villanustre, Eds.). Springer, 2016, pp. 3–11. https://doi.org/10.1007/978-3-319-44550-2_1

[16] S. Kaisler, F. Armour, and J. A. Espinosa, "Introduction to Big Data: Challenges, Opportunities, and Realities Minitrack," in *2014 47th Hawaii International Conference on System Sciences*, IEEE, 2014, pp. 728–728. https://doi.org/10.1109/HICSS.2014.97
[17] M. N. O. Sadiku, Y. Wang, S. Cui, and S. M. Musa, "Industrial Internet of Things," *Journal of Advances in Scientific Research and Engineering*, vol. 3, no. 11, Dec. 2017. https://doi.org/10.7324/IJASRE.2017.32538
[18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, Oct. 2018. https://doi.org/10.1016/j.compind.2018.04.015
[19] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st edition. Addison-Wesley Professional, 1995.
[20] B. C. D. S. Oliveira, M. Wang, and J. Gibbons, "The visitor pattern as a reusable, generic, type-safe component," in *23rd ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications*, 2008, pp. 439–456. https://doi.org/10.1145/1449764.1449799
[21] J. Vaughn, "Hands-on computing with Arduino," *Journal of Computing Sciences in Colleges*, vol. 27, no. 6, pp. 105–106, Jun. 2012.
[22] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, "LTE-advanced: next-generation wireless broadband technology," *IEEE Wireless Communications*, vol. 17, no. 3, pp. 10–22, Jun. 2010. https://doi.org/10.1109/MWC.2010.5490974
[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016. https://doi.org/10.1109/JIOT.2016.2579198
[24] L. M. Vaquero and L. Rodero-Merino, "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, Oct. 2014. https://doi.org/10.1145/2677046.2677052
[25] S. Buzzi, I. Chih-Lin, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5G networks and challenges ahead," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 697–709, Apr. 2016. https://doi.org/10.1109/JSAC.2016.2550338
[26] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: challenges and research advances," *IEEE Network*, vol. 28, no. 6, pp. 6–11, Nov. 2014. https://doi.org/10.1109/MNET.2014.6963798
[27] W. A. Mufti, "ClientNet cluster an alternative of transferring big data files by use of mobile code," in Xia Y., Zhang LJ. (eds) Services - SERVICES 2019. Lecture Notes in Computer Science, vol. 11517. Springer, Cham, 2019. https://doi.org/10.1007/978-3-030-23381-5_8
[28] F.-J. Ferrández-Pastor, H. Mora, A. Jimeno-Morenilla, and B. Volckaert, "Deployment of IoT Edge and Fog Computing Technologies to Develop Smart Building Services," *Sustainability*, vol. 10, no. 11, article number 3832, Oct. 2018. https://doi.org/10.3390/su10113832
[29] M. Habib ur Rehman, P. P. Jayaraman, S. U. R. Malik, A. U. R. Khan, and M. Medhat Gaber, "RedEdge: A Novel Architecture for Big Data Processing in Mobile Edge Computing Environments," *Journal of Sensor and Actuator Networks*, vol. 6, no. 3, article number 17, Sep. 2017. https://doi.org/10.3390/jsan6030017
[30] D. Soni and A. Makwana, "A survey on MQTT: a protocol of internet of things (IOT)," in *International Conference on Telecommunication, Power Analysis and Computing Techniques*, 2017.
[31] S. Monk, *Programming the Raspberry Pi: Getting Started with Python*. McGraw-Hill, 2013.
[32] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, Oct. 2018. https://doi.org/10.1016/j.compind.2018.04.015
[33] E. Pitt and K. McNiff. *Java.rmi: The Remote Method Invocation Guide*. Addison-Wesley, 2001.

Waseem Akhtar Mufti is associated with teaching and research since January 2000. Since then he has been teaching undergraduates as well as Master students of computer engineering, computer science and software engineering at a number of universities in Pakistan. Waseem has a Master degree in Software Engineering from Aalborg University (Denmark) and Bachelor of Engineering in Computer Systems from Hamdard University, Karachi (Pakistan). He is the permanent member of Pakistan Engineering Council and member of technical committees of several international conferences and high impact journals. His research and development interests are distributed systems, super-computing, parallel and concurrent programming in object oriented and functional languages, formal methods in software engineering and verification, automatic computing and software design patterns.
E-Mail: wmufti@gmail.com
ORCID iD: https://orcid.org/0000-0002-2278-8754