Journal of Network and Computer Applications 140 (2019) 1–22

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Edge computing for Internet of Things: A survey, e-healthcare case study and future direction

Partha Pratim Ray [a,*], Dinesh Dash [b], Debashis De [c]

[a] Department of Computer Applications, Sikkim University, Gangtok, India
[b] Department of Computer Science and Engineering, National Institute of Technology Patna, India
[c] Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, Kolkata, India

ARTICLE INFO

Keywords:
Edge computing
Internet of Things
e-healthcare
Industrial protocol

ABSTRACT

The world has recently witnessed the emergence of huge technological growth in the field of data transmission and smart living through various modes of information and communication technology. For example, edge computing has taken a leading role to embark upon the problems related to the internetwork bandwidth minimization and service latency reduction. Inclusion of small microcontroller chips, smart sensors and actuators in the existing socio-economic sectors have paved the Internet of Things (IoT) to act upon the dissemination of smart services to the end users. Thus, a strong need of understating of the industrial elements of edge computing has become necessary that can share the mutual goal while assimilating with the IoT. This paper advocates the crucial role of industrial standards and elements of the edge computing for the dissemination of overwhelming augmented user experience with conjunction with the IoT. First, we present the taxonomical classification and review the industrial aspects that can benefit from both of the IoT and edge computing scenario, then discuss about each of the taxonomical components in detail. Second, we present two practically implemented use cases that have recently employed the edge-IoT paradigm together to solve urban smart living problems. Third, we propose a novel edge-IoT based architecture for e-healthcare i.e. EH-IoT and developed a demo test-bed. The test results showed promising results towards minimizing dependency over IoT cloud analytics or storage facility. We conclude with discussion on the various parameters such as, architecture, requirement capability, functional issues, and selection criteria, related to the survival of edge-IoT ecosystem incorporation.

1. Introduction

Last decade has seen the overwhelming growth of the smart sensors, smart actuators, networking technology, and low-power consuming chips (Ray, 2016a, 2016b). Thus, leading to immense increase of data transmission over the internetwork backhaul. This has certainly put an over burden on the existing cloud infrastructure to efficiently handle all such requests in timely fashion (Dinh et al., 2013; Satyanarayanan et al., 2009; Bonomi et al., 2014). Edge along with Internet of Things (IoT) have emerged as a novel ecosystem i.e. edge-IoT ecosystem that has inherently held the demising power and capabilities to other computing paradigms such as, grid computing, cloud computing and fog computing (Atat et al., 2017; Wang et al., 2016; Ray, 2016c). Obviously, current socio-economic scenario has led to foresee such a novel venture that could be able to exaggerate the speed of social-development upon a surprising ceiling. Indeed, other existing computing solutions are good enough to cater all these social requests, they lack mainly in two areas e.g. (i) latency to serve users' request and (ii) heavy load on internetwork backbone. Resulting an ineffective closure-start for the futuristic smart socialization (Ray, 2016d). This problem becomes more when the industrial involvement come into the scene (Ray, 2015a, 2015b).

To solve this problem, edge computing has been proposed in recent past. It observed that the capability of the edge computing and IoT could be merged to create a new genre of ecosystem which would benefit the overall growth of the information and communication technology-based development (Ray, 2014a, 2015c). It is already validated that positioning of network end-devices to near proximity of user/application enhances the speed of response i.e. the network latency is minimized (Ray, 2014b; Wu et al., 2016). Further, the data-intensive applications are readily become efficient when such interventions are deployed in practice. Such capabilities could lead the edge-IoT ecosystem to outperform other existing computing paradigms. Moreover, placing the IoT solutions along

* Corresponding author.
E-mail address: parthapratimray@hotmail.com (P.P. Ray).

https://doi.org/10.1016/j.jnca.2019.05.005
Received 4 December 2018; Received in revised form 18 March 2019; Accepted 7 May 2019
Available online 12 May 2019

with the edge computing provides an excellent opportunity to serve localized smart applications. Doing so in turn reduces the burden of data propagation through the network backhaul (Satyanarayanan et al., 2015; Premsankar et al., 2018; Mao et al., 2017; Ray, 2016e). This may drastically lower down the cost of network processing and maintenance while supporting for green computing revolution (Ray, 2016f, 2016g, 2017a). Because, lower load on the active network would passive the effect of running the Hugh power hungry cloud data centers and network base stations in background (Chiang and Zhang, 2016; Canzian and Van Der Schaar, 2015; Ray and Agarwal, 2016; Ray, 2016h). The edge-IoT ecosystem can also pursue the geographically distributed aspects while allowing the mobility of end user. Surely, these two virtues give more expansion to its technological wings. It is also known that Content Delivery or Distribution Networks (CDNs) and Information-Centric Networking (ICN) work in similar fashion (Ray, 2015d; Elkhatib et al., 2017). But, they lack in interaction-free delivery services which has been given utmost importance in the edge-IoT ecosystem.

To this end, it is worth to mention that industrial standards and elements could also play a significant role when the edge-IoT ecosystem would be deployed in the reality (Farris et al., 2015; Elias et al., 2017). The reason is simple i.e. the use of popular and contemporary facilities which are currently accommodated with other computing services (Taleb et al., 2016; Kumar et al., 2013; Lin et al., 2015). For example, edge analytics server, NoSQL database engines, communication protocols, and data segments etc. Such industry elements have the power to fill the gap of the edge-IoT ecosystem to make it sharper and technologically visible (Xia et al., 2017; Dutta et al., 2016; Ray, 2017b). Till date, no such literature or document is available that has enlightened the fact of canvassing the industrial elements into the edge-IoT ecosystems. It is believed that upon such integration would provide immense power to the edge-IoT solutions to act efficiently on the undisclosed aspects. This survey presents an in-depth study and analysis of edge computing for IoT based scenario whereby implying the industry-protocols, standards, vendors, communication, data type, ecosystem and usage policies. Further, the presented survey is associated with a novel case study which makes this article unique in flavor and quality than the existing review or survey articles (Ray, 2016i; Hong et al., 2015; Jarschel et al., 2011; Kämäräinen et al., 2014; Fitzgerald et al., 2008).

The major contributions of this article are twofold. First, we present and discuss the taxonomical classification of the industrial edge-IoT computing. We review different factors of the taxonomy behind the edge-IoT. Second, we propose and evaluate a novel e-healthcare architecture that relies upon industrial elements of the edge-IoT ecosystem i.e. EH-IoT. Third, we discuss about the various pillars of the key requirements, functional capabilities, operational issues and architectural structure of the edge-IoT ecosystem to make it more strengthen in near future. The case study was specially demonstrated as a proof-of-concept to confirm that e-healthcare services can be harnessed from the EH-IoT. This indirectly advocates the efficiency of the industrial edge-IoT ecosystem. Many authors and researchers have effectively used and successfully incorporated edge computing paradigm to enhance robustness of their system. A quick search on the IEEE Xplore digital library with the keyword “edge computing”, “edge computing + e-health”, “edge computing + Edgent”, “edge computing + Apache Edgent”, “edge computing + Edgent + healthcare” gave a search result of 390 results out of which a handful of papers were found useful in context of applicability, parallelism and appropriateness. Some of these are discussed as follows: (Huang et al., 2014) validates the efficiency and resourcefulness of edge computing by providing extensive survey on edge systems and the also presents comparative study of cloud computing system. Their results show the edge system performs better than the cloud system. Ahlgren et al. (2012) developed a system which continuously monitors patient's heartrate with the help of heartrate sensor, this data is continuously analyzed by the KAA edge computing server, the analyzed data is sent to the user's smartphone. A design by Vallati et al. (Griffin et al., 2014) achieves a histrionic decrease in service latency and ensures the security of locality information. Sapienza et al. designed an edge computing-based smart urban-citizen application that can successfully perceive certain critical events e.g., man-made disasters (Ren et al., 2018). In Pan and McElhannon (2018) and Peterson et al. (2016), the researchers selected an edge computing-based architecture to address various network-related issues in vehicular technology. An efficient scheduling and adaptive offloading scheme is proposed that minimizes the computation complexities in the prescribed vehicular network. It is also known that the edge architectures might play a vital role in e-health applications which could save the lives of many patients. As edge computing guarantees a faster response and higher throughput, the decision-making process becomes faster and easier in the e-health applications. For instance, Ali and Ghazal (Ryden et al., 2014) proposed a real-time mobile detection service for heart attack by using the edge computing. The scheme showed lower service latency when it was combined with the geographical awareness that can rightly detect the patient's location. Lastly, Fl'avia Pisani et al. proposed their framework that is capable to execute cross platform code in the NodeMCU 1.0 (Habak et al., 2015).

The rest of this article is organized as follows: Section 2 discusses on the classifications of the edge-IoT taxonomy. Section 3 presents a novel EH-IoT architecture and test-bed is evaluated. Section 4 shows various parametric considerations required for sustainability for the edge-IoT ecosystems in terms of discussions. Lastly, Section 5 concludes the article.

2. State-of-the-art on the Edge-IoT taxonomy

This section presents an in-depth orientation on the state-of-the-art edge-IoT taxonomy. It covers eight classes of industrial components such as, (A) edge software and analytics, (B) edge-IoT ecosystem, (C) edge-IoT cloud platform, (D) edge-IoT key vendor, (E) edge-IoT data types, (F) edge-IoT open database systems, (G) Edge-IoT SoC platform and (H) Edge-IoT communication. More details on each of these components are prescribed later. Fig. 1 presents the taxonomy of industrial component of edge-IoT computing.

2.1. Edge software & analytics

1. *AWS Greengrass:* Amazon AWS Greengrass works in and around three key specifics such as (1) AWS Lambda functions: helps to respond local IoT edge devices as early as possible upon triggering of localized events by using lambda functions (2) machine learning inference: makes Greengrass easier to perform learning inferences which are ordinarily trained in Amazon cloud sites, and (3) AWS IoT core: it utilizes the core authentication engine to restrict IoT devices from exchanging data between such devices which are authentic in the same network. Moreover, data synchronization helps to orient the change in local-database with the remote cloud ones. As lambda function minimizes response time by implying local computational power, offline services are also possible. A set of simplified programming models help developers to code own device per AWS lambda specifics, thus overall cost reduction is achieved for IoT related applications development.
2. *Cisco Fog Director:* Cisco Fog Director leverages in-built capability to manage large-scale IoT-based fog applications. It is assisted with visual-web programming environment that helps to integrate APIs with currently used managerial options. Four key parameters are involved in this model that includes (1) seamless and rapid adoptability: (2) continuous IoT-fog application monitoring, (3) Consistent management services and (4) on-demand remote assistance and debugging support. Thus, it extends solutions to the existing benefits such as, improvement over operational effectiveness over the single-point production associativity, business agility and scalability over the application life-cycle quality enhancement.
3. *Cisco IOx:* Cisco IOx edge development environment relies upon Cisco IOx Application Framework (CAF) and Fog Director application

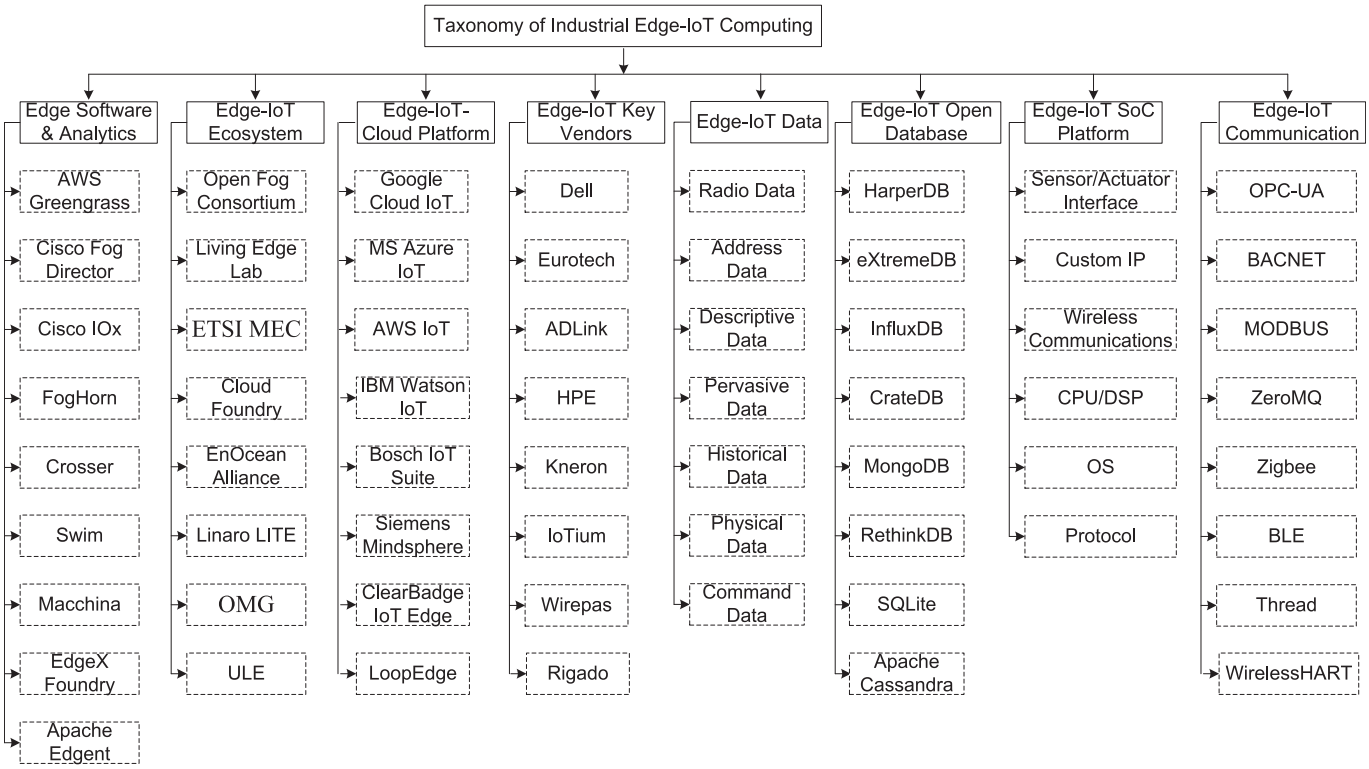

Fig. 1. Taxonomy of industrial edge-IoT computing scenario.

management modules. It is meant to be connected with Cisco IOS® software platform. It features following capabilities such as (1) IoT application life cycle management, (2) Platform-as-a-Service (PaaS), (3) Docker-container deployment, (4) VPN as well as IPSec based tunneling for enhanced security, (5) orientation with the IOx client tool for choice-based product maximization and (6) DevNet-based trial runs. It has supports for Cisco 800, 4000, 1000 and IR510 series network infrastructures. Similarly, IOx core options are leveraged by the following means such as, IOX/809-Core/829-Core/IE4K-CORE/CGMSRV-CORE/TBD). Among others, help in rapid IoT growth, flexible application deployment and closed-loop data service are the most important benefits of the Cisco IOx.

4. *Foghorn:* Foghorn is one of the emerging vendors in the area of intelligent edge service domain. The aim of the FogHorn is to bring the power of big data and near real-time stream processing at the local user end. Its high performance and heterogeneous applications are best suitable for the closed-loop edge environment. Due to very small footprint, it can be embedded into wide range of gateways, industrial digital systems and PLCs. It has supports for (1) data ingestion from local sensors, (2) operational technology differentiator, (3) machine learning from local historian-based services, (4) complex event processing engine for the domain specific language, (5) cloud agnostic monitoring and (5) management of deployed instance. Further four key benefits are achieved from this platform that includes, predictive use case normalization, lowering bandwidth and real-time security maximization of the implied system.
5. *Crosser:* Crosser is an excellent inclusion in the enterprise level edge analytics solution providers list. It provides seamless integration to any type of enterprise application scenario. In-built orchestration capability gives it space for high end facility to leverage lightweight IoT-edge solution. It can collect data from any local sensor and connect IoT-based communication and messaging protocols to on-premise, off-premise or mobile assets. Metadata generation, cleaning and preparation are specialization of this solution. Anomaly and outlier detection in real-time aspect gives it power to customize edge-cloud managing algorithms to effectively deal with. Besides, multi-hierarchy-based action, alter and notification services are produced on various severity levels of work in low-latency time period. Event driven enterprise data flow is easily acted by the Crosser application life cycle management unit. It is good at benefitting low cost Machine-to-Machine communication, minimized network cost, lowering bandwidth and the cloud-independence.
6. *Swim.ai:* Swim.ai is a machine learning-based edge analytic solution that processes real-time distributed IoT applications. It is specially built to tackle the convolutional neural networks (CNN) for the IoT-based time series data stream. Its EDX module help in-built artificial intelligence models to self-train from the sensed edge data. Thus, event prediction, system maintenance, failure assumption and activity patterning are done in seamless manner. The “digital twin” model is incorporated to find the critical information of the present system and detect the futuristic behavior. The EDX made sensor-data fabric efficiently reduces, learns and analyzes big edge data in low-cost network design model.
7. *Macchina.io:* Macchina.io is newly introduced IoT-edge platform that integrates IoT-edge device software development kit and the remote manager kit to efficiently manage the resources deployed in the on-premises as well as cloud infrastructure. It hosts the device application server that is run by java script-based engine that facilitates IoT-based communication protocols. Several key benefits involving the hardware lock-in avoidance protocol, industry-proven software platform and reduced time-to-implement cost are achieved from the Macchina platform. Further, remote IoT-edge devices can be securely accessed and managed via web browser, mobile APPs, VNC and SSH.
8. *EdgeX Foundry™:* EdgeX Foundry™ is a Linux-based open source framework developed for IoT-edge computing service orchestration. It has interoperable, collaborative and operating system-agnostic plug-and-play-based software reference platform. Several open source and industry standard groups and protocols are merged at one point of Foundry setup. Both of the IP and non-IP specific loosely-coupled microservices are hereby included to facilitate cloud-

independent orientation. The foundry is deployed for following key values such as, data orchestration, edge database, edge system management and edge analytic services. Several north-bound infrastructure and applications are leveraged with microservices intercommunicate via core APIs and services.

9. *Edgent:* Edgent is an Apache open source incubation application platform. Various range of edge devices including Java 8/7, Raspberry Pi B and Android are applicable for Edgent implementation. It acts as back-end system for edge-device application. Four types of back-end message buses including MQTT, IBM Watson IoT Platform, HTTP, JDBC, Apache Kafka, customized message bus, communicate with the Edgent. Several centralized streaming analytics (Apache Storm, Flink, Samza and IBM Streams) could be harnessed for Edgent linkage. It uses Java8 Lambda expressions to package and export edge applications. It is based on the TStream.map() function for string handling purposes. Edgent connectors are used as interfacing media with similar or other entities. A comparison among edge software and analytics is shown in Table 1.

A generic architecture for edge-IoT ecosystem is hereby proposed in Fig. 2. The overall ecosystem is restricted between "NORTHBOUND" and "SOUTHBOUND" infrastructure, applications and "things" respectively. It is a three-core layered architecture that consists of (i) edge-IoT platform, (ii) IoT platform and (iii) cloud services. The architecture is framed such way that the edge-IoT platform is capable to handle device, system and metadata management. This low-level is supported with several other effective components that includes Software Development Kit (SDK), local management console and command pool manager. The manager is responsible for serving aggregation, filtering, data collection and anomaly detection in the edge-IoT layer. A built-in connector protocol connects this layer with the IoT platform i.e. core component of the architecture. As expected, it accommodates a few of selective access, orchestration and controlling mechanisms. System integration services serve the monitoring, data ingestion, alerting and triggering activities with help of the context aware services. A number of heterogenous IoT platforms can work together to assist the edge-IoT platform i.e. lowest layer of architecture. The loosely-coupled micro-service component belongs to the cloud layer i.e. highest level of architecture. Container deployment and data flow controlling is done by this layer. Further, rule engine optimization, scheduling, real-time analytics and device configuration are served by the cloud services. A bunch of industrial cloud services can work seamlessly integrated fashion within this architecture.

2.2. IoT edge ecosystem

1. *Open Fog Consortium:* The mission of this consortium is to cultivate and align standard development tasks related to fog and related computing paradigms. The goal is to develop an open reference fog computing architecture for building operational technology models and deployable test beds. Several working groups (WG) are involved in these tasks that includes architecture WG, communications WG, manageability WG, security WG, software infrastructure WG and testbed WG. Cisco, Intel, Microsoft, Dell and ARM holdings are the key members of this initiative.
2. *Living Edge Lab:* The goal of this lab initiative is to implement a large, open and flexible test-bed for edge computing application and research in public domain. The key benefits that it aims at are (1) disruptive enhancement of client's experience and (2) providing reduced data-trafficking by bringing cloud service at the edge. Several edge services including cloud to edge migration, device to device migration, end to end parametric testing and end user trails are available through the implied test beds.
3. *ETSI Multi-access Edge Computing (MEC):* This initiative is running under the roof of the Industry Specification Group (ISG) of the European Telecommunications Standards Institute (ETSI). The purpose of this MEC is to leverage seamless and open framework for integrating various edge computing-based applications originating from the vendors, developers and third-party service providers. All authentic operators can run their specific Radio Access Network (RAN) at the edge of the deployed MEC. Services like video analytics, augmented reality, data caching, local content delivery and IoT are expected to be harnessed from it.
4. *Cloud Foundry Foundation:* It is a collaborative outcome from the organizations such as, Cisco, Google, IBM, Microsoft, Pivotal, SAP and SUSE to make faster and ease of access deployable cloud-based applications. The aim is to facilitate the end users and developers to focus on application development rather than platform related overhead. This framework helps to connect any cloud by any applications to access the services and interoperable integration. Further, container specific service and multi-cloud based multi services are handled by this framework. Edge related services are thus automatically harnessed.
5. *EnOcean Alliance:* It provides a flexible, cost efficient and energy efficient ecosystem for closed periphery activities, especially for smart building automation. It maintains a standardized wireless (868 MHz, 902 MHz and 928 MHz) application protocol stack that runs upon IoT-based systems. It implies the self-powered wireless IoT-based sensor to be controlled at the edge of the networks. It also performs following tasks such as, remote management and commissioning, product labelling, energy harvesting, over IP security and smart acknowledge.
6. *Linaro IoT and Embedded (LITE) Group:* It facilitates various open source as well as proprietary software integration services for the IoT and edge-computing scenario. Applications on top of FreeRTOS, ThreadX, Zephyr, mbed etc., could be easily developed within this software framework. Several industry standard projects based on Arm v8, LAVA, Kernel CI, and LSK are under process. Advanced boot loaders and TrustZone services could be formulated by using LITE implementation. LinaroEdge is a special solution that is developed to support for Time Sensitive Networking (TSN) mostly applicable in the edge era.
7. *Object Management Group:* This group focuses on developing the industrial-IoT (IIoT) based standards and protocols. It also specializes in leveraging interoperable

Table 1
Comparison among edge software.

Edge Software	Real-time response	Machine learning	IoT Core Support	In-built SDK	Open Source	Cloud Agnostic	Enterprise Solution
AWS Greengrass	Yes	Yes	Yes	No	No	No	Yes
Cisco Fog Director	Yes	No	Yes	No	No	No	Yes
Cisco IOx	Yes	No	Yes	Yes	No	Yes	No
Foghorn.io	Yes	Yes	Yes	Yes	No	Yes	No
Crosser.io	Yes	Yes	No	No	No	No	Yes
Swim.ai	Yes	Yes	No	No	No	No	No
Macchina.io	Yes	No	Yes	Yes	No	No	Yes
EdgeX Foundry™	Yes	No	Yes	No	Yes	Yes	Yes
Apache Edgent	Yes	No	Yes	Yes	Yes	Yes	No

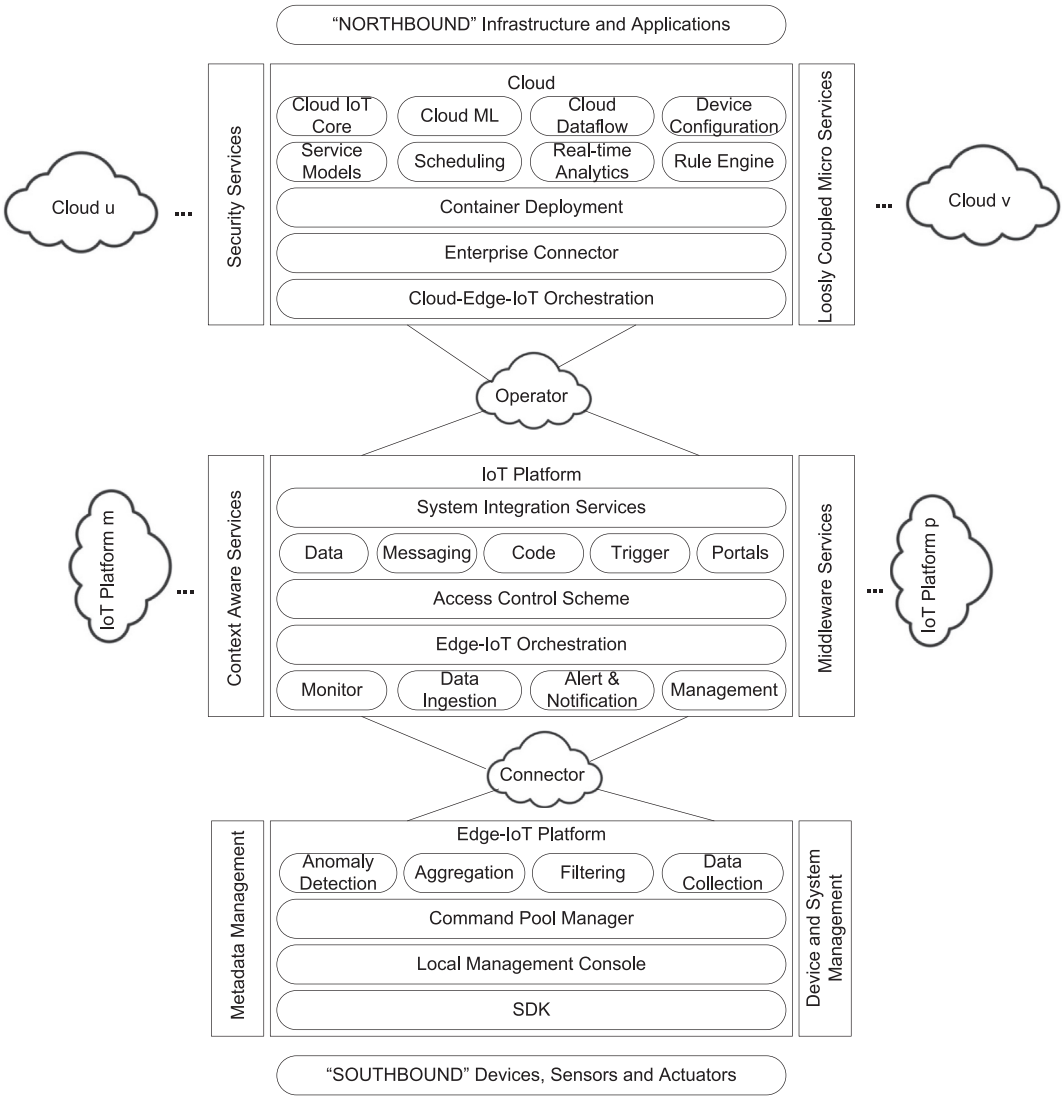

Fig. 2. Proposed generalized architecture of the edge-IoT ecosystems.

IIoT test-beds, especially for healthcare, retail and modular open systems. Many distributed as well as edge IoT solutions are under process that caters the in-built Business Process Modelling (BPM).

8. *ULE Alliance:* It focuses on development and processing of ultra-low energy and low latency networking with optimized communication strategies. Such wireless solutions have moderate data rate specifically meant for the Digital Enhanced Cordless Telecommunications (DECT). It runs on top of the Home Area Network Functional protocol (HAN FUN) complying several interoperability test events. Table 2 shows the comparison among these ecosystems.

2.3. IoT edge-cloud platforms

1. *Google Cloud IoT:* Google provides an intelligent cloud service platform for IoT and edge enabled scenarios. The framework is segregated into three layers such as, (1) edge device, (2) cloud data analytics and (3) data usage. A real-time machine learning facility is incorporated into the edge device layer that facilitates services for the edge IoT core and Linux-based android-things. Four key functionalities are harnessed at the cloud data analytics component where Cloud Bigtable, BigQuery, Cloud Dataflow and Cloud Machine Learning particulars take part in data analysis. Cloud Datalab and Cloud Studio-based solutions cater the data usage and storage policy.

Table 2
Comparison among edge-IoT ecosystems.

Edge Ecosystem	Computing Paradigm	Key factor	Access Type
Open Fog Consortium	Fog, Edge-IoT	Open reference fog architecture	Open Source
Living Edge Lab	Edge, IoT	Flexible test-bed	Open Source
ETSI MEC	MEC, IoT	Radio access network	Closed/ Proprietary
Cloud Foundry Foundation	Cloud, Fog, Edge, IoT	Multi-cloud framework	Open Source
EnOcean Alliance	IoT, Edge	Energy efficient smart building automation	Closed/ Proprietary
LITE	IoT, Edge	Time sensitive networking	Open Source
OMG	IoT	Industrial business process modelling	Closed/ Proprietary
ULE	Edge	Ultra-low energy networking	Closed/ Proprietary

2. *MS Azure IoT:* Microsoft provides IoT application development facilities at the edge of the network with help of its Azure IoT platform. It relies on the cost per usage policy with not termination time period. It depends on the powerful integration of Azure IoT Edge that leverages near real-time response, clod agnostic and reduced solution cost. Its

secure and intelligent edge helps to deploy AI and analytics services at the edge.

3. *AWS IoT:* Amazon's AWS IoT is a cloud served IoT on the edge service that comprises of Amazon FreeRTOS, AWS Greengrass, IoT Core, IoT Device Management, IoT Device Defender, AWS IoT Analytics, IoT 1-Click and IoT Button-based components. It serves all types of integrated micro-services at the edge by incorporating low-footprint OS, centralized device behavior monitoring system, localized data caching, and on-board point of contact application deployment opportunities.
4. *IBM Watson IoT:* It is a cloud hosted intelligent IoT service which could be run at the edge devices of any network. Watson API is key of this framework that provides real-time as well as social-sentiment analysis on the device generated data. Anomany detection and performance validation are also doe by the risk manager component. Seamless IoT device connection and management are formulated by the Watson's intelligent engine at the backend.
5. *Bosch IoT Suite:* Bosch has incorporated a software service for IoT-edge integration that relies on top of a compact module of open, flexible and intelligent suite. Machine-to-machine communication is paved through this suite framework. A remote manager solution is integrated with the underlying edge-IoT analytics software. The price of usage is served as trial basis.
6. *Siemens Mindsphere:* Siemens provides MindConnect solution to the edge-IoT application developer. The MindConnect platforms is correlated with the hardware and software stack of Siemens. It can be hosted on the MindSphere cloud be it provide or public infrastructure. It also comes with an open IoT OS to facilitate millions of distributed devices around various application domain.
7. *ClearBlade IoT Edge Platform:* It provides an enterprise level cloud-assisted edge service platform to connect IoT devices deployed in the industrial sectors. It is a small as well as scalable edge platform that enables organizations to configure, deploy, filter, stream, synchronize and manage state-of-the-art IoT devices with help of a ClearBlade software stack. No SDK, nor local software module is paved to do such tasks. All the programs, data, messaging service and triggering mechanism are pooled on the edge site.
8. *LoopEdge:* It provides an industrial IoT support through a complete package of edge platform that runs on backend cloud or on-demand on-premise. It supports two types of hardware for deployment that includes (1) IoT device and (2) gateways. Arduino, Mbed, Spark MSP430, Chipkit Max32, TI Stellaris are examples of IoT devices that connect sensor and actuators for sensing and out delivery purpose. Gateways perform message format translation task at the edge. Dell, Nexcom, HPE, HMS, Intel x86, Kontron IPC are examples of such gateways. Another special feature called connector is used within its platform such as, MongoDB, Cassandra, MySQL, Hbase, Redis, MariaDB, Informix, IBM DB2, LoopDB, Bigtable, SAP Sybase etc. These are various type of DBMS that store data and message streams for possible analysis and extraction purpose. Table 3 presents the comparison among the edge-IoT platforms,

2.4. IoT edge hardware vendors

1. *Dell:* Dell is engaged at development of cutting edge IoT solutions in terms of edge-gateway, edge controller and edge-cloud platform. It is presently seeking involvement into the EdgeX Foundry for possible collaboration and cooperation to build a unified edge-IoT platform.
2. *Eurotech:* It is specialized into the development of smart IoT gateways which are currently assembled with the open source Eclipse Kura platform. The aim of Eurotech is to manage edge-IoT aspects with help of an open source cloud platform to disseminate the specialized services to its clients via AWS IoT and MS Azure.
3. *ADLink:* Currently, ADLink is busy with manufacturing of the IoT gateways to be installed at the edge of the networks. It seems to have two strategies to counter the IoT marketplace such as, (1) collaboration with Wind River software solution, and (2) deploy the Vortex Edge software to get operated from various embedded device lists. Interestingly, it is also keen to pursue a strong bonding with the open source initiatives like Eclipse Cyclone and Fog05 etc.
4. *HPE:* It manufactures smart IoT gateway device called Edgeline. It is a series of gateways with variety of modular design and specifications for different set of applications. However, they rely on the third-parties for development of the software packages to run the Edge-line series gateways.
5. *Kneron:* Kneron is an edge Intellectual Property (IP) developer company. It specializes in cultivation of the Kneron NPU IP and visual recognizing software for edge-based solution. Recently, it has started the production of the dedicated AI-based low power deep learning processor production which runs on CNN mechanism.
6. *IoTium:* Distributed edge provisioning for the IIoT is the only goal of the IoTium. The edge software is capable to run on top of Intel x86 machines as well as certified Dell, Advantech, Cisco and Lanner gateways. All types of wireless and cellular communication facilities are in-built into the IoTium software supported nodes.
7. *Wirepas:* It provides a mesh topology-based distributed and scalable IoT infrastructure. It runs with help of Wirepas Mesh software that can accommodate with any type of radio communication on any embedded IoT device at massive scale. Currently, it has been successfully tested on the smart buildings, smart city, smart logistics and smart energy monitoring and management.

Table 3
Comparison among edge-IoT platforms.

Edge-Cloud Platforms	Type of Ownership	Key incorporations	Target Hardware	Platform Access
Google Cloud IoT	Proprietary	Cloud Bigtable, BigQuery, Cloud Dataflow, Cloud Machine Learning	AndroidThings/Linux kernel	Restricted (Trial)
MS Azure IoT	Proprietary	Azure IoT Edge, IoT Central	Linux/Windows	Pay per Usage
AWS IoT	Proprietary	Amazon FreeRTOS, AWS Greengrass, IoT Core, IoT Device Management, IoT Device Defender, AWS IoT Analytics, IoT 1-Click, IoT Button	AWS Partner Network (APN) supported device	Pay per Usage
IBM Watson IoT	Proprietary	Watson API, Watson IoT Cloud	Linux, Windows	Pay as you Go, LITE
Bosch IoT Suite	Proprietary, Flexible Open	Software as a Service, Bosch IoT Remote Manager	Bosch IoT Gateway	Pay per usages, Subscription
Siemens Mindsphere	Proprietary, Open	MindConnect, Open IoT OS	Siemens hardware	Pay per usage, Subscription
ClearBlade IoT Edge Platform	Proprietary	EdgeLite, IoT Portal, IoT Packet Manager	NXP, Dell, HPE, Phoenix Contact, Owasys, AMD, Mica, Arms, Lanner, Moxa, Sercom, Raspberry Pi, Toshiba	Restricted (Trial)
LoopEdge	Proprietary	Connector, IoT Device Driver and Gateways	Arduino, Mbed, MSP430, TI Stellaris etc.	Pay as you go

8. *Rigado:* Rigado facilitates IoT Edge-as-a-Service evaluation kit for seamless connectivity between edge and IoT-cloud infrastructure. Most of these wireless modules come under monthly subscription basis to provide cost-effective and on-demand edge computing facilities. It is capable to go with ubuntu core, java, python and node-red to minimize the time-to-market with ease of application development. Three key components are currently in market place that includes (1) Cascade-500 IoT gateway with BLE, Zigbee, Thread, Wirepas Mesh, ethernet, 2G, 3G and Wifi connectivity, (2) Edge-as-a-Service; Edge Connect, Edge Direct and Edge Protect and (3) Wireless modules: Nordic and NXP enabled BLE 4.2, 5.0 and Thread connectivity.

2.5. Edge IoT data type

1. *Radio Data:* Such type of data is generated from radio frequency generating communication module. In major cases, data originated from the Radio Frequency Identification Module (RFID), Bluetooth, ZigBee, LoRA etc., include different packet structure. Thus, heterogeneous radio packet data-structure is of utmost importance to get catered with.
2. *Address Data:* Multiple number of devices under the same edge-IoT scenario require unique addressing schemes, hence the need of hosting similar data structures. These data types can vary from one device to another and from one genre of communicating protocol to a unique type of device.
3. *Descriptive Data:* Every edge-IoT-based process, system and object should be framed and formulated in such way so that it could be described effectively. Thus, descriptive data structure must be hold for any relevant edge-IoT sub systems.
4. *Pervasive Data:* IoT frameworks are frequently used to deal with the environment and positional applications. Pervasiveness thus becomes very important in this regard. Current edge-IoT frameworks must facilitate similar data structure.
5. *Historical Data:* Every node and controller of the edge-IoT network processes long chain of data streams at different time stamps of operation. Earlier originated data may be preserved for some future analytical or debugging situations. So, there should be some provisions to restore the historical data in suitable format.
6. *Physical Data:* Mathematical modelling is a very interesting and important aspect for the edge-IoT design verification and validation. As and when these procedures are over, it may be prescribed for comprehension of similar objectives in the near-future. Such reality emulating data may be harnessed for processing.
7. *Command Data:* Edge-IoT networks are filled with actuators, for example, alarm, auto sprinkler, robotic motorized hand etc. These actuators work upon receiving of appropriate commanding signal from the master. These types of data should be assimilated to dig in-depth for more detailed apprehension of actuator behavior.

2.6. Edge IoT open database systems

1. *InfluxDB:* This is an example of combination of the key-value and Time Series DBMS models. It is specialized in leveraging NoSQL type query processing in IoT environment. Moreover, it can handle the NewSQL and increasing data model query structures. It can interpolate some of the missing data from the in-built database.
2. *CrateDB:* It is one of the most innovative DBMSs in current IoT market that handles SQL structures. It is capable to scale linear data ingestion into the growing IoT data samples, cluster data-balancing and in-memory data replication. It can work in conjunction with the Kafka, NodeRED, Grafana, Apache Lucene, Netty, and Presto in no lock-in manner. It can be run in both of the edge and or cloud.
3. *MongoDB:* It is used as a general-purpose document-oriented database for IoT-based applications. It uses JSON-like formats for processing the data. Recently it has started to support BSON formatting. Master sharding and MapReduce techniques are used as the query processing methods.
4. *RethinkDB:* It is an open source, document store class database API. It implements Unified Chainable Query Language (UCQL) to access to JSON data store. Real-time commands are herein used as a plug-and-play function for WWW interaction. It utilizes real-time push architecture for instantaneous data rehabilitation. It also offers asynchronous query processing while implementing the Eventmachine. SSL connectivity is also paved by the underlying architecture.
5. *SQLite:* It comes with a very small footprint for implementation in the end application, especially for embedded devices. It is a self-contained as well as highly reliable open source RDBMS. It does not depend on the client-server architecture, instead processes SQL while getting inside the embedded systems.
6. *Apache Cassandra:* It is one of the most popular wide column store class NoSQL query processing engines. Innovative query processing techniques such as CQL and Thrift are utilized in this system. It comes with "no single point of failure" capability for utilizing real-time series data. Decentralized, scalable and elastic characteristics are core part of this database solution.

However, two database systems are getting equal popularity though they belong to proprietary class as described later (Cuervo et al., 2010; Ray et al., 2017; Ray, 2017c). Table 4 presents the comparison among these database systems.

HarperDB: It comes as a web based graphical user interface (GUI) that can be run on edge-device without intervention from the cloud. The web interface is capable to perform management of data for users, customized schema and modify roles within it. Both of SQL and NoSQL queries can be responded by the HarperDB engine. Moreover, real-time chart preparation, graphical visualization and query processing are easily made possible. All configurations are handled by JSON. Its small footprint is capable to handle multiple workloads with a single modelled dynamic schema by using REST API.

eXtremeDB: It features the in-memory and persistent DBMS facilities. Primarily it comes under the RDBMS and Time Series DBMS models. It also caters as the Key-value store model as its secondary model. It has the capability to provide clustering and sharding mechanisms in its forefront. Its footprint is extremely small in size, thus appropriate for the embedded IoT device implementation. SQL queries in accordance to another NoSQL is facilitated.

2.7. Edge-IoT SoC specifications

1. *Sensor/Actuator Interface:* Edge-IoT devices are enabled with different types of SoC as well as ASIC architectures. These SOC/ASIC modules are complied with the multiple sensor/actuators. Thus, the SOCs should be capable of handling various interfacing genres for efficient communication with sensor or actuator data. Some of the most important interfaces are Analog-Digital convertor (ADC), Digital-Analog Convertor (DAC), General Purpose Input Output (GPIO), Serial Peripheral interface (SPI), Intern IC Bus (I2C) and MIPI-based Inter IC Sensorwire Bus (I3C).
2. *Custom IP:* SoCs are responsible for hosting several edge-IoT-based computationally intensive workload handling. Customized IP core are thus needed to cater such operations. Out of many, following are the most important cores that includes edge analytics processing, enhanced processor security, intelligent control logic functions and heterogenous sensor/actuator fusion.
3. *Wireless Communications:* Each of the edge-enabled SoCs are integrated with a variety of popular and mostly used wireless communication standards such as ZigBee, LoRA, Bluetooth, WiFi, WirelessHart. Recently, visible light communication modems are also getting popular with the other well-known solutions.
4. *CPU/DSP:* All the deployed SoCs are run by either specialized CPU or DSP processing units. A long-list of such processor core developing

Table 4
Comparison among edge-IoT database systems.

DBMS	Access Type	Class	Query Language	Latency	Compressed Foot print	JSON Object
InfluxDB	Open Source	Time Series, Key-Value	NewSQL	Low	23.7 MB	Yes
CrateDB	Open Source	RDBMS	SQL	Real-Time	5.5 MB	No
MongoDB	Open Source	Document Store	DOBL	Low	Cluster Based	Yes
RethinkDB	Open Source	Document Store	UCQL	Real-Time	10.9 MB	Yes
SQLite	Open Source	RDBMS	SQL	Low	2.17 MB	Yes
Apache Casandra	Open Source	Wide Column Store	CQL, Thrift	Low	35.6 MB	No

DOBL: Dynamic Object-Based Language.

companies are currently upholding the IoT market share. Following are some of the crucial market players in this regard, such as, MIPS, CEVA, Texas Instruments, ATMEL, ARM, Cadence, ARC Synopsis and ANDES.

5. *OS:* Several low footprint operating systems are being deployed into the vast range of SoCs for effective system interfacing purposes. Most of those support many of the RISC/CISC cores to get implied with. A handful of them are included in this study due to their popularity and usefulness that includes Free RTOS, RIoT, Tiny OS, Contiki, Generic Linux for embedded applications and ARM Mbed OS. Table 5 presents the comparison among the edge-IoT OS.
6. *IoT Protocols:* Different types of data management and messaging protocols are used in edge-IoT ecosystem. Employed SoCs are hence made capable to pursue some of the very important and reliable IoT protocols. Some examples of these protocols are as follows, Message Queuing Telemetry Transport (MQTT), Data Distribution Service (DDS), Light Weight Machine to Machine (LWM2M), Constrained Application Protocol (CoAP), and IPv6 Low Power Personal Area Network (6LoWPAN).

A novel industrial edge-IoT protocol stack is proposed in this regard as shown in Fig. 3 (Ray, 2017d, 2017e, 2017f). As seen, it is a 4-layered protocol stack which is better suited for SoC implementation. It is well understood that any edge-IoT protocol should be placed over one of the wide-range of SoCs available in market. The ARM, ANDES, Cadence, MIPS, ARC and CEVA are the most popular as well as leading market players who develop SoC cores for CPU and DSP. All such CPU and DSP cores are readily deployable in several bunches of embedded sectors in the edge-IoT application. The next higher layer of the stack comprises of customized intellectual property (IP) that deals with edge analytics, security, sensor fusion and control logic related deployments. Now-a-days, a long list of OS variants is existing in public domain. But, a handful of those are open-source as well as easily burn-able namely, TinyOS, FreeRTOS, RIoT OS, Mbed OS, and Contiki OS. The next higher level of protocol stack leverages communication and sensor/actuator interfacing protocols. Bluetooth, LoRa, WirelessHART, ZigBee, WiFi are mostly used protocols in SoC stacks. Similarly, I2C, SPI, ADC, DAC, I3C are the mostly promising interfacing protocols that come in the list of choice of the developers. The top most layer of the protocol stack deals with the IoT protocols that covers a long list of efficient and light-weight means such as, DDS, MQTT, LoWPAN, CoAP, XMPP, WebSocket etc (Chun et al., 2011; Ra et al., 2011).

2.8. Edge-IoT communication protocols

1. *OPC-UA:* It the most prevalent edge-IoT communication protocol for the IIoT scenario. This Open Platform Communication standard for is designed to cater the platform independent Unified Architecture for the industrial edge-IoT services. It is capable to handle AES 128-256 encryption in each session of communication while leveraging support for HTTPS and SOAP for various applications. Apart from these, it can accommodate message signing, user authentication and firewall-friendly user control activities. It is one of the mostly used extensible and information modelling framework for rigorous industrial use (Gordon et al., 2012; McKeown et al., 2008).
2. *BACnet:* Smart city and smart home automation revolution are at the verge of the current technological scenario. Thus, an efficient Building Automation and Control Networks are of the utmost requirement. BACNET is such kind of data communication protocol that leverages supports for multiple industry standards, for example, American National Standard Institute (ANSI), International Organization for Standardization (ISO) 16484-5 and American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) etc. It defines a wide number of physical/data link layer protocols that includes the RS-22, RS-485, Ethernet, ZigBee and LonTalk (Pan et al., 2016; Ranjan et al., 2015).
3. *MODBUS:* It is the *de facto* communication standard for connecting industrial electronic equipment. MODBUS comes with a variety of protocols such as remote terminal unit (RTU), TCP/IP, UDP, Plus, Pemex and Enron. It relies on mesh networking architecture and able to communicate with the supervisory control and data acquisition (SCADA) systems over the industrial, scientific and medical (ISM) radio bands and General Packer Radio Service (GPRS). Despite of such facts, it lacks in high speed and accurate timing issues.
4. *ZeroMQ:* It a newly released distributed message queuing protocol that runs without incorporation of a centralized message broker. It works with helps of four in-built pattern technologies such as, request-reply, publish-subscribe, push-pull (pipeline) and exclusive pair. The underlying library is developed in C++ and licensed over LGPL. The socketing technology is very similar to the Berkley sockets.
5. *ZigBee:* It is developed over the IEEE 802.15.4 specifications for catering the Personal Area Network (PAN) while complying with the low-power, small-bandwidth requirement and low-data rate applications. Generally, its range is within 10–100 m in line of sight. However, recently the range has been tested successfully up to 40 KM.

Table 5
Comparison among edge-IoT OS.

OS	Min. RAM	Min. ROM	C Support	C++ Support	Multi-Threading	MCU w/o MMU	Modularity	Real-Time
RIOT	~1.5 kB	~5 kB	Yes	Yes	Yes	Yes	Yes	Yes
Linux	~1 MB	~1 MB	Yes	Yes	Yes	No	Partial	Partial
Tiny OS	<1 kB	<4 kB	No	No	Partial	Yes	No	No
Contiki	<2 kB	<30 kB	Partial	No	Partial	Yes	Partial	Partial
Mbed OS	>4 kB	>32 kB	Yes	Yes	Yes	Yes	Yes	Yes

https://www.riot-os.org/.

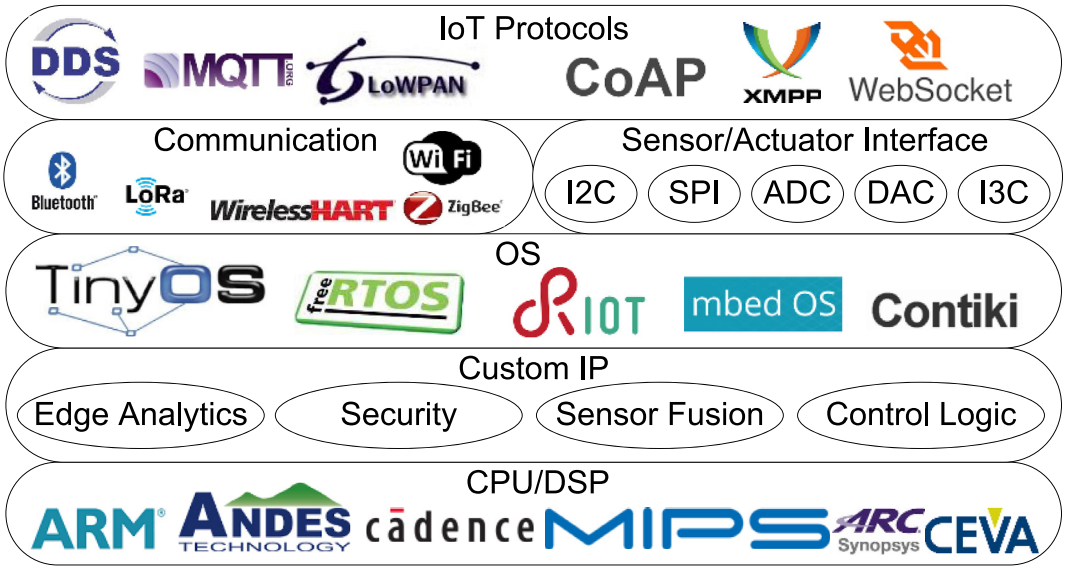

Fig. 3. Edge-IoT SoC protocol stack.

On average the data rate is around 250 kbps (Wang et al., 2017; El-Sayed et al., 2018).

6. *Bluetooth:* It is based on the IEEE 802.15.1 standard specifications that is used to exchange data in close proximity, average 10 m distance. A spectrum band of 2.4–2.845 GHz Ultra High Frequency (UHF) is allotted for peer-to-peer communication. Recently released Bluetooth 5 has proven to have 2 Mbps physical layer transmission rate. The range has been increased to 50–100 m in line of sight with minimized power consumption (Madukwe et al., 2017).
7. *Thread:* It is made to deliver communication services for the building automation related tasks in the edge-IoT ecosystem. It is developed on top of 6LoWPAN that indirectly relies on the IEEE 802.1.5.4 protocol. It implements a very low-power mesh networking topology for future-proof, reliable and secure IoT applications (Vallati et al., 2016; Sapienza et al., 2016).
8. *WirelessHART:* It is another IEEE 802.15.4 compliant popular wireless sensor networking (WSN) leveraging solution that is developed on the Highway Addressable Remote Transducer Protocol (HART). The WSN works on the basisof the Time Synchronized Mesh Protocol (TSMP). Several communication protocols can be interoperable with the WirelessHART (Ray et al., 2018a; Feng et al., 2017).
9. *LonWorks:* It is a standard developed by the Echelon Corporation for establishment of communications among edge-IoT devices through the fiber optic and any other wired channel. It follows the ANSI/EIA 709.1, ISO/IEC 14908-1, 14908-2, 14908-3, and 14908-4 Standards. It is licensed under the proprietary cost-benefit model. Normally its usage cost is very high due to its licensing policy. Practically, it is used for control, home automation, and transportation fields. Table 6 provides the comparison among these communication protocols (Zhang et al., 2017; Ali and Ghazal, 2017; Edge IoT protocol stack, 2018).

2.9. Use cases

1. *Waggle:* It is an ongoing scientific project hosted by the Argonne National Laboratory. The main aim of this research project is to leverage edge-IoT supported hardware-software platforms for conducting smart city research. It comes with a set of hardware modules specially designed for scientific experiments in the fields of edge and IoT specific areas for example bee hive monitoring, environmental monitoring, e-healthcare, city pollution monitoring and smart transportation. It consists of three components namely, (i) Waggle sensor node, (ii) Waggle edge-IoT framework and (iii) Waggle cloud infrastructure. The sensor module consists of temperature, humidity, infrared hyperspectral imager camera, sound detector, CO, CO2, NO2, and particulate sensors. It also consists of the motor control, air pump, robotic sampler and LED status-based actuators. It can also monitor e-health vitals. Such sensed data are temporarily stored at the

Table 6
Comparison among popular edge-IoT communication standards.

Protocols	Monitoring Organization	Specifications	Access	Cost	Use
OPC-UA	OPC Foundation	C, C++, Java, .NET, Python, SOA architecture	Free/Open	Low	Not Given
BACnet	ASHRAE	ANSI/ASHRAE Standard 185; ISO-16484-5; ISO-16484-6	Free/Open	Low	Industrial, Building automation, Transportation
MODBUS	Modicon Inc.	IEC 61158 Standard, ASCII, RTU, TCP/IP	Free/Open	Low	HVAC, Building automation, Access control
ZeroMQ	iMatix	Inter Process TCP, IPC, TPIC, C++, JNI implementation	Free/Open	Low	Asynchronous, distributed/concurrent Industrial automation
ZigBee	ZigBee Alliance	IEEE 802.15.4 Standard, Low power, 100–500 kB/s data rate	Restricted/ Proprietary	Moderate	Industrial automation, WSN
Bluetooth	Bluetooth SIG	IEEE 802.15.1, 2.4 GHz, ISM band, 10–100 m, 100-1 MB/s data rate	Restricted/ Proprietary	Low	Home automation, Industrial, Personal Networking
Thread	Thread Group	IEEE 802.15.4 standar, 6LowPAN	Free/Open	Low	Industrial automation, Control
WirelessHART	HART Communication Foundation	IEC 62591, IEEE 802.15.4 standard, ISM band, TSMP technology, HART protocol	Royalty Free, Copyright License	Low	Industrial WSN
LonWorks	Echelon Corporation/ Motorola	ANSI/EIA 709.1; ISO/IEC 14908-1, 14908-2, 14908-3, 14908-4 Standard	Proprietary	Very High	Control, Home automation, Transportation

local edge-node which is made from low power-core computer-based system. Highly secured uplink manager assists the edge to communicate with the Waggle cloud repository to store and analyze the data. An on-demand cloud scaling mechanism is incorporated with the deployed Waggle cloud. Overall system is thus intelligent and supports the attentive sensing and actuating events efficiently. Fig. 4 presents the underlying brief architecture.

2. *Array of Things:* Array of Things (AoT) is an initiative by the Argonne National Laboratory that aims at solving problems related to the urban environment and infrastructure monitoring by using the sensed-data from its modular and programmable sensor nodes deployed at various locations in any city. The analyzed data are often open-sourced to the public use-purposes and advanced analytical researches. Mainly, the pollution and environment related sensor data are processed by the AoT framework. The inferred information is later sent to the needy, for example real-time pollution level is informed to the subscribed even/morning walker of the city and the chronic asthma patients. It also suggests the pedestrians about the safest timing for move around the city during day or night. Moreover, traffic lights are also used to inform the citizens about the current status of traffic congestion and other related information. Currently, AoT nodes are being experimentally installed in the sea shore of Chicago city for which the real-time graphical visualizations are handled by the Plenario services (Plenar.io). The hardware designs and software codes are readily available at https://arrayofthings.github.io. Fig. 5 represents the architectural framework for the AoT implementation.

3. EH-IoT: a case study on Edge-IoT for e-healthcare

3.1. Problem definition

Since Edge computing paradigm is a new concept, there is lack of availability of lightweight solution. Though many researches tend to implement this concept mostly by creating their own edge application which is specifically dependent to a particular hardware, they lack in open source aspect, ease of customization and applicability. Developers around the world is contributing for the development of open source software called Apache Edgent. Though, Edgent is still in its infancy stage, not much research articles have been published and very few detailed documentations are available as open research platforms. Moreover, e-healthcare related applications have not been much tested and validated against the Edgent periphery.

3.2. Objectives

This case study proposed and developed the novel Edgent enabled edge-IoT computing system for e-healthcare i.e. EH-IoT that primarily focuses the below:

- Deployment and configuration of the Edgent on the Raspberry Pi 3 as the edge-IoT system;
- Exploring the connector classes of Apache Edgent to communicate with the external entities back end systems;
- Setup the proof-of-concept test-bed for provisioning of edge services on the IoT-based scenario; and
- Minimization of the amount of data transmission to analytics IoT cloud server.

3.3. Materials used

This study uses several hardware and software modules as described below: (The MachNation IoT Architecture; Functional architecture of edge-IoT system; Selection of best IoT platform; Requirements of edge-IoT paltforms)

Temperature Sensor: LM35 is an industry standard temperature measuring chip, used for measuring body temperature for proof-of-concept.

Raspberry Pi 3: It supports for developing Edgent-based edge-IoT system. In this experiment it is used to deploy the local edge computing serving device i.e. EH-IoT.

Arduino Uno: Itis an open-source microcontroller board based on the ATmega328P microcontroller. It is hereby used to host the

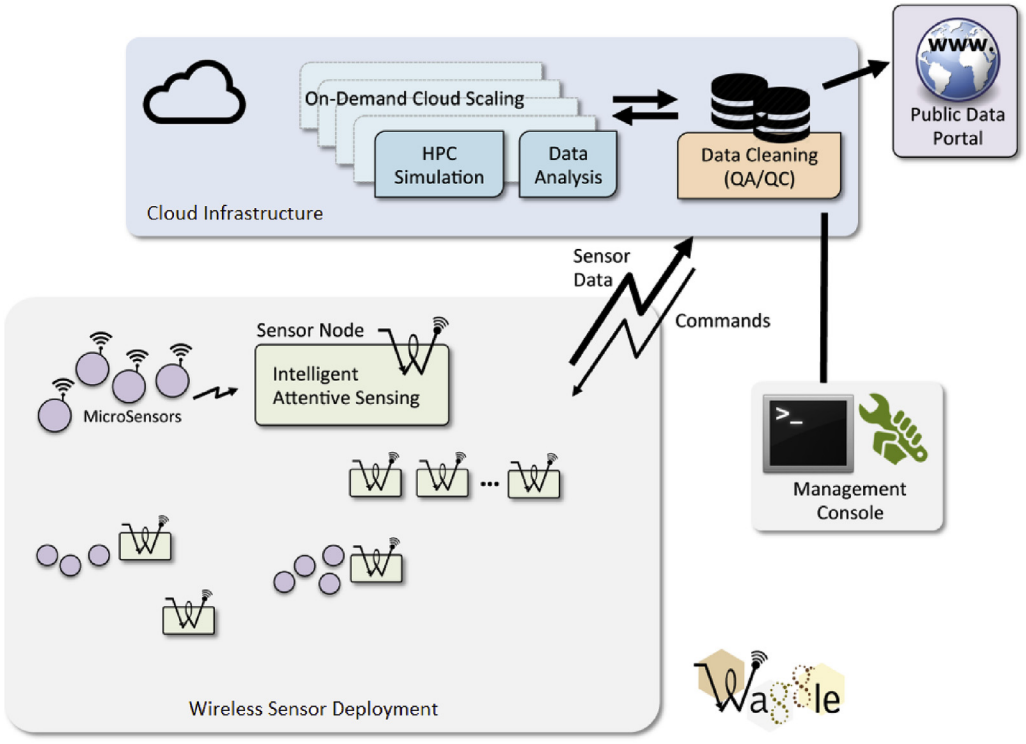

Fig. 4. Waggle-based sensor to cloud information exchange architecture (Waggle-based sensor to cloud information exchange architecture).

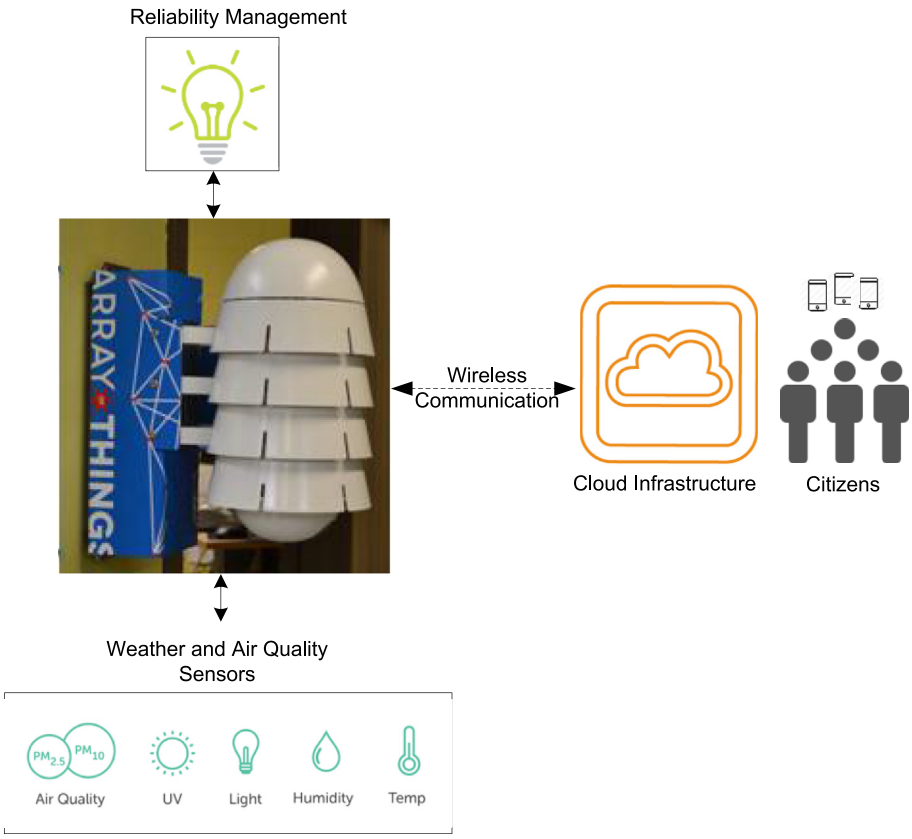

Fig. 5. Citizen-based cloud information exchange architecture.

communication and data flow from LM35 to the EH-IoT system.

Raspbian OS: Raspbian is used to host the Edgent on top of Raspberry Pi 3. In this study it is used for leveraging seamless edge analytics services through the EH-IoT.

Apache Edgent 1.2.0: It is an open source development tool to analyze real-time data on the edge of IoT scenario. Its main purposes in this study are as follows: (i) reduction of the amount of data transfer to centralized remote IoT cloud analytics server and (ii) minimization of the amount of data transfer that gets stored in the IoT cloud repository.

ThingSpeak: In order to analyze, view and store the sensed data, the ThingSpeak provides an efficient and easy IoT cloud analytics platform to cater the required needs.

3.4. Methodology

The proposed EH-IoT deployment comprises of three key modules such as, (i) Apache Edgent engine, (ii) embedded hardware-based sensing unit and (iii) IoT-based cloud repository.

(i) *Apache Edgent Engine:* Apache Edgent is an open source programming model designed basically for edge devices with low computation power. It provides necessary APIs and classes for the development of Edgent application. The Edgent engine is responsible for handling all the necessary tasks such as receiving streams of incoming data from the data source, process and filter those data to determine critical readings and send it to the cloud for further analysis. The main features of the Edgent are as follows: (i) Functional flow API for streaming analytics (Map, Flat map, Filter, Aggregate, Split, Union, Join, Dead band filter), (ii) Connectors (MQTT, HTTP, WebSocket, JDBC, File, Kafka), (iii) Java API allows you to send JSON to an MQTT device, (iv) bi-directional communications with the backend, (v) web-based interface to view application graph and metrics, (vi) availability of Junit, and (vii) usefulness of the Java-based Lambda expressions.

(ii) *Embedded hardware-based sensing unit:* This module comprises of the basic hardware components which is responsible for generating raw data (unprocessed data), which is sent to the edge device for processing and analysis. The hardware included in this module comprise of IoT devices which generates frequent (continuous) streams of data such as temperature sensor for healthcare applications. Sensing unit acts as data source to the edge device. Here the data source is LM35 temperature sensor (used as a proof of concept) which reads human body temperature.

(iii) *IoT based cloud repository:* Data can be sent from the EH-IoT framework to the IoT-based cloud analytics unit i.e. back-end system, if needed to accomplish analysis that cannot be performed on the EH-IoT, such as: (i) running complex analytic algorithm that necessitates huge resources (CPU or memory), (ii) maintaining large amount of state information about the system, and (iii) correlating data from the device with data from other sources. The EH-IoT communicates with back-end system through the following message hubs (i) MQTT, (ii) IBM Watson IoT platform, (iii) Apache Kafka - an enterprise message bus service, (iv) custom message hubs, and (v) ThingSpeak – open IoT platform with in-built Matlab analytics.

Fig. 6 presents the proposed architecture behind the EH-IoT. As mentioned earlier, the architecture is segregated into four parts such as, (i) hardware: it contains Raspberry.

Pi 3, Raspbian OS, Java 1.8.0 and Apache JVM, (ii) Edgent engine: it comprises of publish/subscribe connector tool that communicates with other Edgenet-based application programming interfaces; further

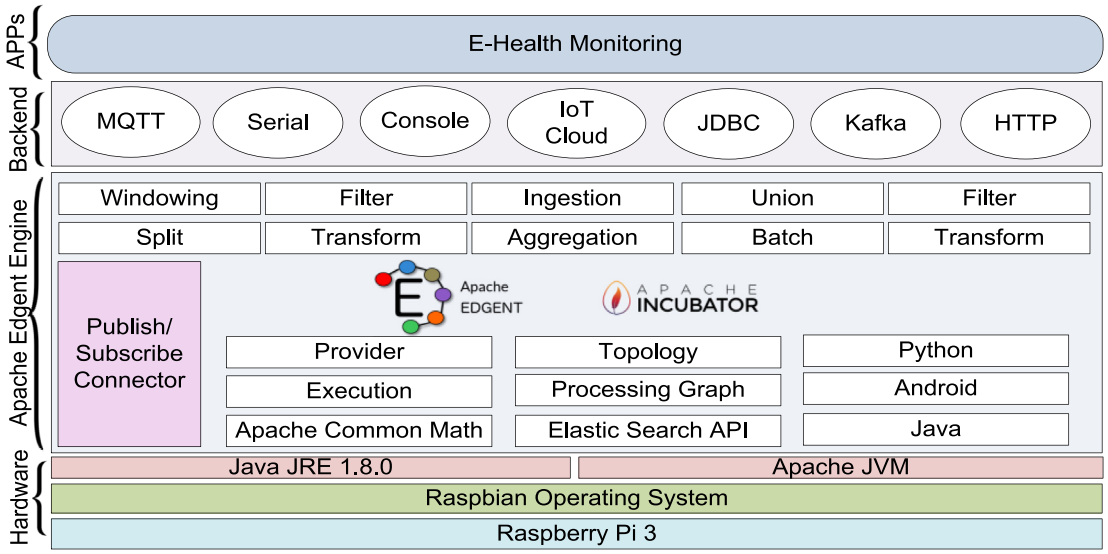

Fig. 6. Proposed architecture of the EH-IoT edge-IoT system.

ingestion, filtering, transformation, windowing and aggregation etc., are executed, (iii) backend: the backend is a collection of a list of supports such as MQTT, serial connector, console, IoT cloud connector, JDBC, Apache Kafka and HTTP, and (iv) applications: in this study e-health monitoring is given the most important priority, however other applications including industry 4.0, smart automation and smart agriculture could also be involved.

The proposed EH-IoT system model is presented in the Fig. 7. The system model shows the demo body-temperature being sensed from the user's body and transmitted to the Arduino Uno as a source of data. The raw data is then transferred to the Edgent-based Raspberry Pi 3 edge-IoT device which performs all types of filtering, aggregation and dissemination of temperature data either to IoT cloud via the IoT gateway or for local processing. In this study, critical or filtered data is only transferred to the IoT cloud analytics server. If the data is within the specified range then it is locally processed, thus reducing the computation time and overhead on the internet backhaul. Pseudocode 1, 2, and 3 are prescribed for efficient activity of the EH-IoT system model that measure body temperature, data filtering cum cloud integration and Eh-IoT to cloud communication. Fig. 8 (a) and (b) present the EH-IoT testbed implementation and physical attachment with a user, respectively. Flow chart of three components are illustrated in Fig. 9. (a), (b) and (c) that depict following, data source, data sink and filtering mechanism, respectively.

Pseudocode 1
EH-IoT Body Temperature Measurement.

```
Public Class TempSensor Implements Supplier<Double>
{Static Final Long SerialVersionUID = 1L, raw_data, str;
 @Override Public Double Get()
          {      SerialPort[] ports = SerialPort.getCommPorts();
                 SerialPort serialPort = ports[chosenPort - 1];
           try {        SerialPort.setComPortParameters(9600,        5,        1,
SerialPort.NO_PARITY);
                 @SuppressWarnings("resource")
                 Scanner data = new Scanner(serialPort.getInputStream());
                 if(data.hasNextLine())
                    {    str = data.nextLine();
                 raw_data = Double.parseDouble(str);
                 }
        }catch (Exception e ) {} serialPort.closePort(); return (raw_data);
 } }
```

Pseudocode 2
EH-IoT Data Filter and Cloud Integration.

```
Public Class TempSensorApplication
{Initialize Static OPTIMAL_TEMP_LOW = 96.0
Initialize Static OPTIMAL_TEMP_HIGH = 99.5
Initialize Static Range<Double> optimalTempRange
optimalTempRange = Ranges.closed(OPTIMAL_TEMP_LOW,
OPTIMAL_TEMP_HIGH)
Public Static Void Main (String[] args) throws Exception
    {       TempSensor sensor = new TempSensor()
            DirectProvider dp = new DirectProvider()
            Topology topology = dp.newTopology()
            SendToThinkSpeak thinkspeak = new SendToThinkSpeak()
            TStream<Double> tempReadings = topology.poll(sensor, 500,
TimeUnit.MILLISECONDS)
            TStream<Double> filteredReadings = tempReadings.filter(tuple ->
            !optimalTempRange.contains(tuple))
            filteredReadings.sink(tuple -> System.out.println("abnormal
temperature detected! "
          + "It is " + tuple + "\u00b0F!"))
          long time = System.nanoTime() - start
          filteredReadings.sink(tuple -> thinkspeak.sendDataOverRest(tuple))
          tempReadings.print()
          dp.submit(topology)
  }
}
```

Pseudocode 3
EH-IoT to IoT Cloud.

```
Public Class SendToThinkSpeak Implements Callback<JsonNode>
{       Public Void SendDataOverRest(double temp)
       {          Unirest.post("https://api.thingspeak.com/update.json")
              .header("accept", "application/json")
              .field("api_key", "BJH2BQDISA2P4UT8")
              .field("field1",temp)
              .asJsonAsync(this);
       }
       @Override Public Void Completed(HttpResponse<JsonNode> response)
        {
            int code = response.getStatus()
              JsonNode body =response.getBody()
            InputStream rawBody = response.getRawBody()
          }
}
```

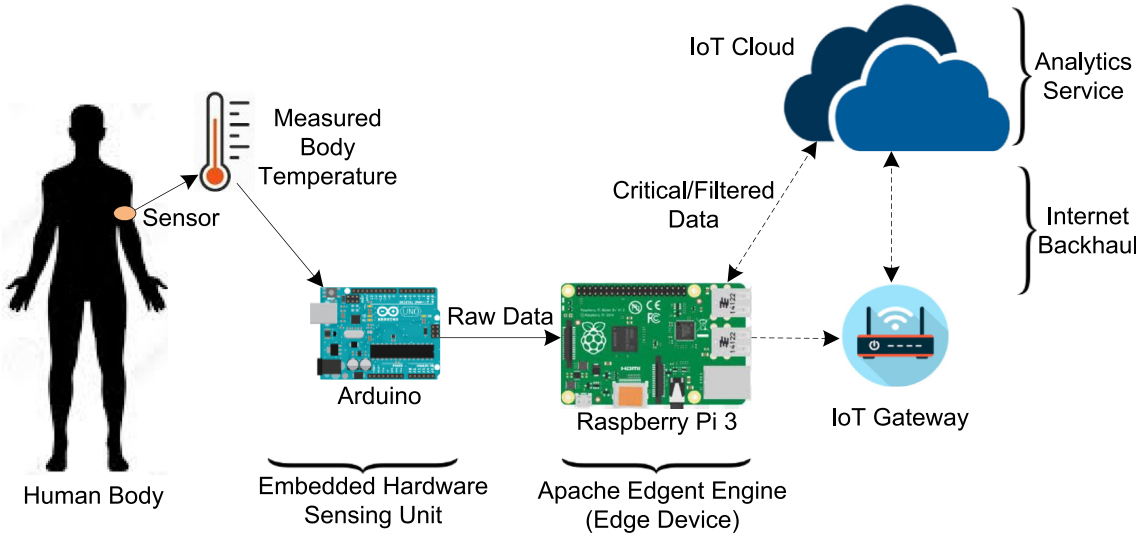

Fig. 7. Proposed system model of the EH-IoT system for e-healthcare demo.

3.5. Study design

(i) *Real-time Data Transmission:* Data sensed by the sensing unit of the EH-IoT is continuous so in order to handle those data an efficient technique is required. Since the Arduino is connected via serial port to the Raspberry Pi 3. Data transmission and Serial Port selection is handled by the JSerialComm. JSerialComm is a java library designed to provide platform-independent way to access standard serial ports without requiring external

(ii) libraries, native code, or any other tools. It is meant as an alternative to RxTx and the (deprecated) Java Communications API, with increased ease-of-use, an enhanced support for timeouts, and the ability to open multiple ports simultaneously. Fig. 10 presents the data flow transition in the EH-IoT.

(iii) *Transition of data from Arduino to Edgent Enabled Device:* Arduino Uno is connected via USB to the Raspberry Pi3. Temperature sensor LM35 is connected to the Arduino Uno with its data pin connected to the analog pin A0 and a program containing the control to access the sensor is burned to the Arduino Uno. On the other hand, a Java program containing the main method is executed in Eclipse IDE on the Raspberry Pi3. The serial communication between the Arduino and the Java program is handled by jSerialComm library. On executing the main Java program, it creates an instance of the DataSource program which is responsible for getting data from the Arduino and feeding it to the main Java program. A flowchart depicting the flow of program is given below.

(iv) *Filtering Tuples:* If we are interested in determining when the temperature is strictly out of the optimal range suppose say 96 °F and 99.5 °F (which is the normal temperature range of human body), a simple filter can be used. The *filter* method can be applied to *TStream* objects, where a filter predicate determines which tuples to keep for further processing. In this case, we want to keep temperatures below the lower range value *or* above the upper range value. This is expressed in the filter predicate, which follows Java's syntax for the *Lambda expressions.* Then, we terminate the stream (using *sink*) by printing out the warning to standard out.

3.6. Results and discussions

(i) *General Analysis:* The application runs on any device which supports Java version 8. Some promising results was obtained based on number of test case scenarios which is described in the following sections. Tests was performed on two scenarios: (i) user without fever and (ii) user with fever. The temperature sensor is placed on armpit of the patient. Normal temperature readings range from 96 °F to 99.5 °F on healthy human body. Whereas temperature exceeding the range is considered as abnormal or the body is in fever state. When the body temperature of the patient is normal, the sensor readings is processed by the edge device itself i.e., the data is not uploaded to the cloud. The temperature is only uploaded when the readings goes beyond the range. Readings taken from the experiment is shown in the Fig. 11. The green line in the above indicates that the body temperature of the patient is

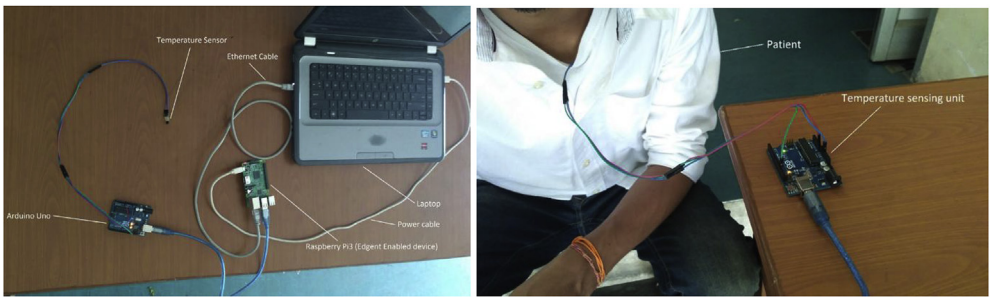

Fig. 8. (a) Prototype of EH-IoT system. (b) Reading temperature from the user with support from EH-IoT.

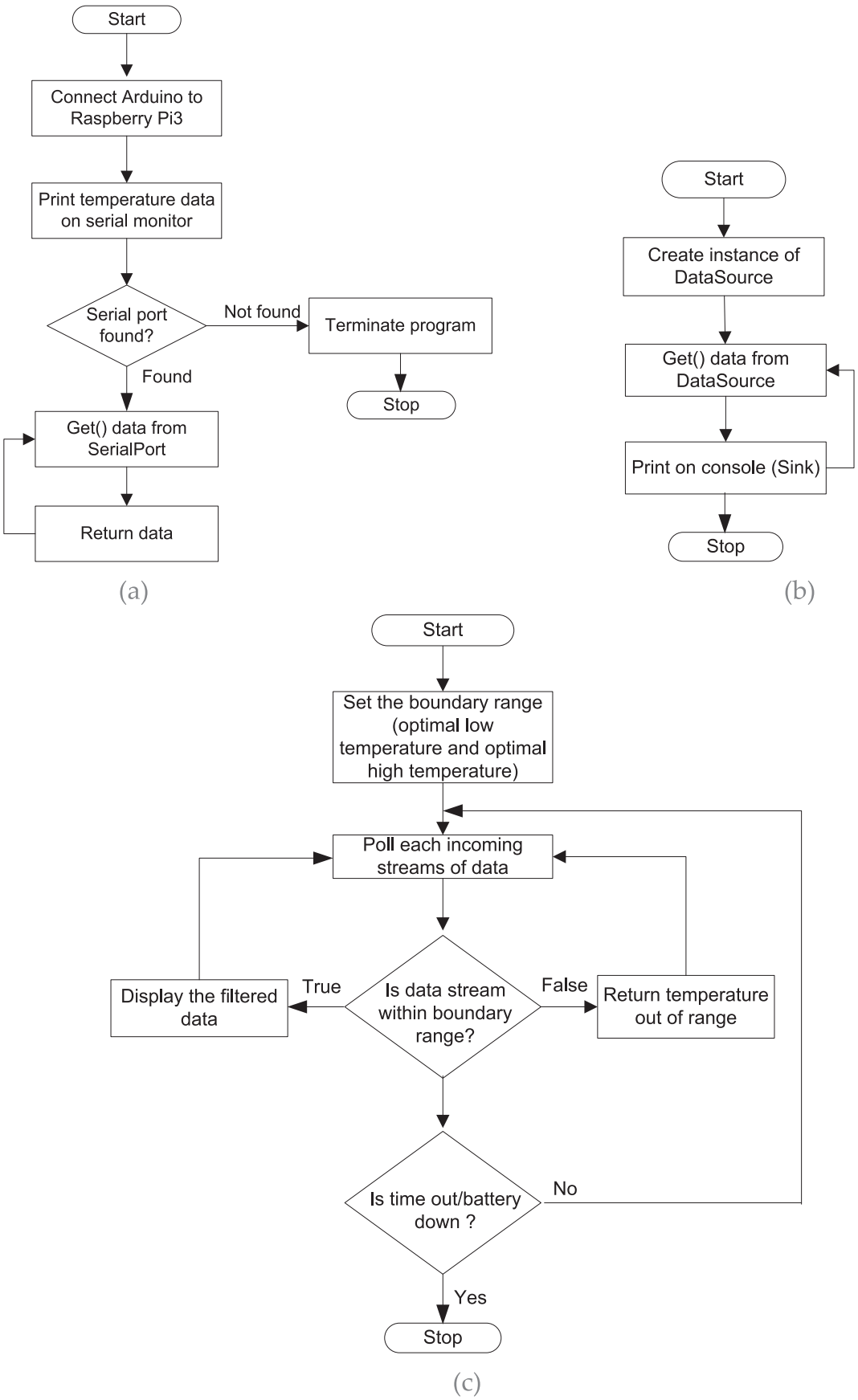

Fig. 9. EH-IoT flowchart depicting (a). Data Source (producer). (b). Data Sink (consumer). (c) Filtering mechanism.

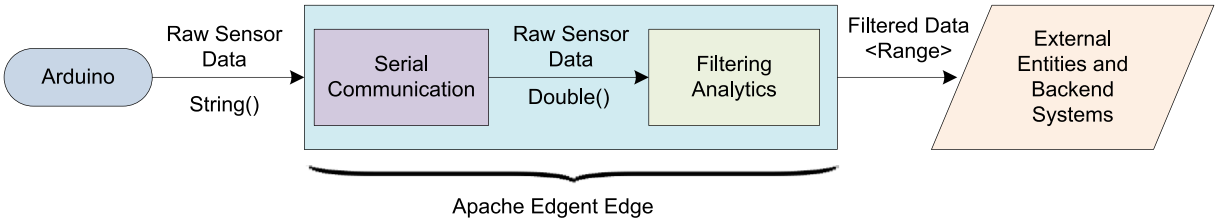

Fig. 10. Data flow transition within EH-IoT system.

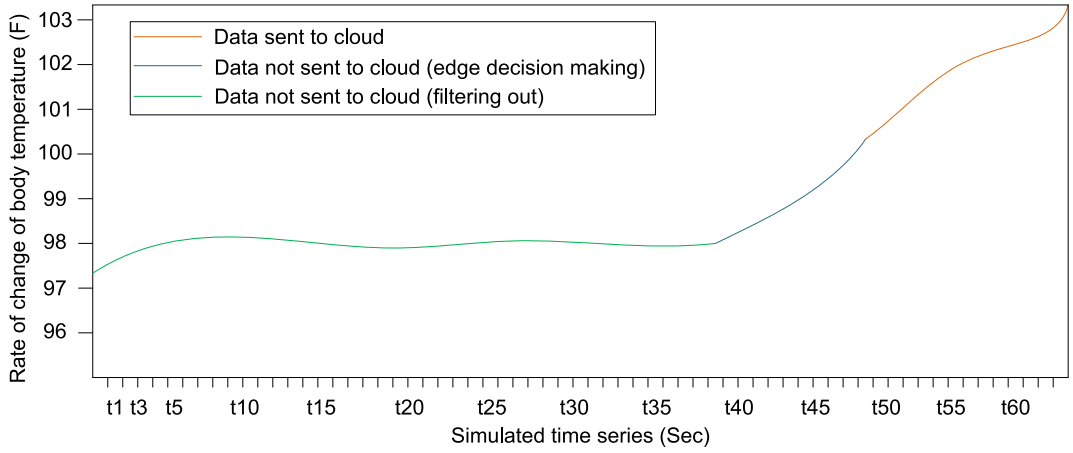

Fig. 11. The graph depicting rate of change in data values of EH-IoT temperature readings.

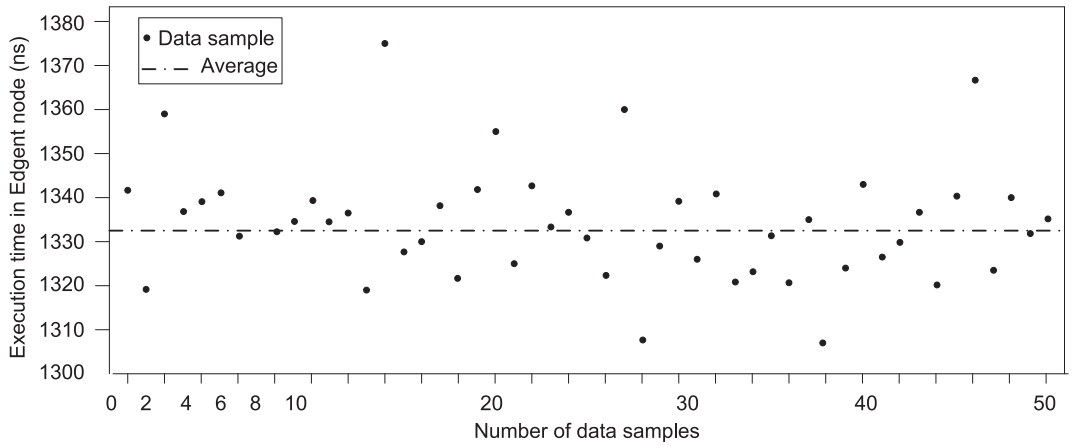

Fig. 12. Computation time samples in EH-IoT.

normal. When the temperature rises steadily and crosses the upper optimal range i.e. 99.5 °F the readings are forwarded to the cloud which is depicted by the red line. Further analysis can be done on these data on the cloud. The main motive of the EH-IoT is to reduce the workload on the cloud and also reduce the bandwidth. Sending only selected data values to the cloud reduces the bandwidth significantly. Our comparison of data processing done by edge to the cloud gave us promising results which has been depicted below. From our analysis on healthy human body, the temperature tends to remain in the normal range for longer duration. So as long as the temperature is normal there is no need to send it to the cloud which only increases bandwidth. Out of 13 readings taken, 9 readings were found to be in the range of optimal temperature (normal healthy human body). However, 4 readings were found to be out of range when the temperature began to rise (when the body is in fever). Calculation of result gave us the significant difference in percentage which was 73% data handled by the edge device and only 27% was forwarded to the remote IoT cloud. Fig. 11 shows the data transition from edge to IoT cloud.

(ii) *Computation Time:* Calculation of computation time is done using the *System.nanoTime*() method. The time is initialized to a variable at the beginning of the program. Another variable is assigned for the end time at the end of the program. Now the difference in start time and end time gives the total time required to execute the program. The time required to run EH-IoT application is measured in nanoseconds which has been depicted in the chart below. Ten computation time samples have been taken from the program

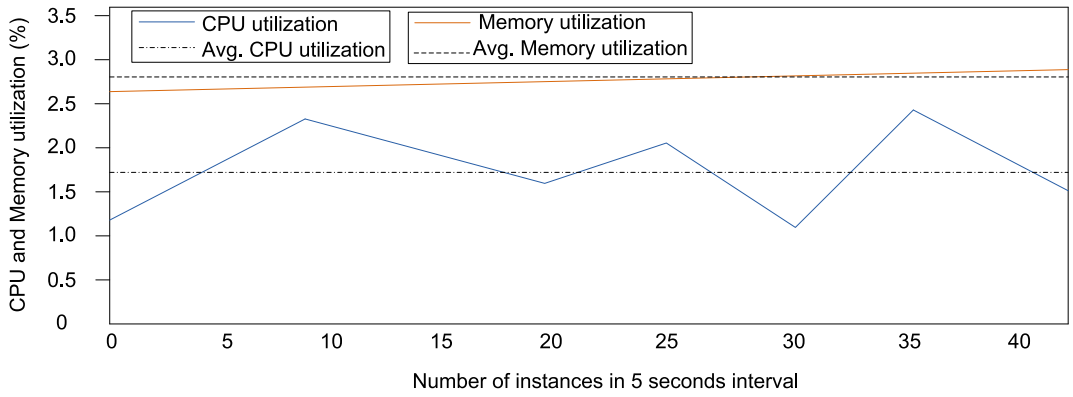

Fig. 13. CPU and memory utilized by the EH-IoT.

execution ranging from minimum of 1300 ns to maximum of 1380 ns. The average computation time was 1332 ns. Fig. 12 presents the computation time samples.

(iii) *Resource Utilization:* Raspberry Pi 3 is equipped with a quad-core 64-bit ARM (Advanced RISC Machines) Cortex A53 running at 1.2 GHz processor and 1 GB of RAM. The resource utilization has been monitored using Linux *top* command on a specific process id: *top –p* 971. Memory and CPU were monitored with the interval of 5 s also the readings were taken accordingly. Fig. 13 depicts the usage of CPU and memory utilized in terms of percentage. From the readings taken from the resource utilization it was seen that minimum CPU utilized was 1.3% and the maximum was up to 2.7%. However, memory utilization was not more than 2.9% (27.2 Mb) out of the available 1 GB of RAM.

(iv) Latency comparison: For the calculation of latency while uploading data to the ThingSpeak IoT cloud a mqtt class called *TimerPingSender* from the interface *MqttPingSender* was used. Which considers the time (in ms) required for a packet to complete one round trip time (RTT). It shows the trip time required for a packet to complete one round trip from the edge device to the ThingSpeak IoT cloud. The latency calculation for the data sent by the Arduino Uno was done with the help of the timer function called *Timer*() in Java. A timer function called *millis*() is started in Arduino Uno as soon as the data is printed on the Serial monitor. The timer method also runs in ET-IoT system in parallel. The difference in time delay required by the ET-IoT system to capture the packet is calculated. The delay in receiving packet data from the Arduino Uno to ET-IoT is also achieved from the ET-IoT system. It is clear that the latency of the IoT cloud is far more than that of the ET-IoT system. The Figs. 14 and 15 illustrate the latency of data transmission to IoT cloud and local processing, respectively.

4. Discussions and issues

This section enlightens edge-IoT architecture, functionality, operational issues, requirement, capability and selection criteria that are the most crucial factors for sustainability for the edge-IoT ecosystem in technological struggle.

4.1. Edge-IoT architecture, functionality and operational issues

1. *Protocol-wise Architecture:* Fig. 16. Presents the protocol-wise generic edge-IoT architecture for the existing ecosystem. The architecture is categorized into three levels (i) embedded devices (ii) edge-IoT and (iii) cloud. Th embedded device category comprises of two components such as, hardware device and device-to-edge communication protocols. The hardware devices list embedded systems, embedded cores and popular device specific architectures (ARM, DSP core etc.). On the other hand, BLE, WiFi, Visible Light Communication (VLC), I2C, Serial communication, ZigBee, Thread etc., are the mostly used and selective popular protocols that edge-IoT ecosystems heavily reply upon for communication purposes. The next higher layer comprises of four segments such as, edge-IoT solutions, edge-IoT protocol adapter, edge-IoT applications and edge-IoT-cloud connectivity protocols. The edge-IoT layer could be assumed as the core engine of the presented architecture. Several edge-IoT rules engines, data storage facilities, analytics and data normalization techniques

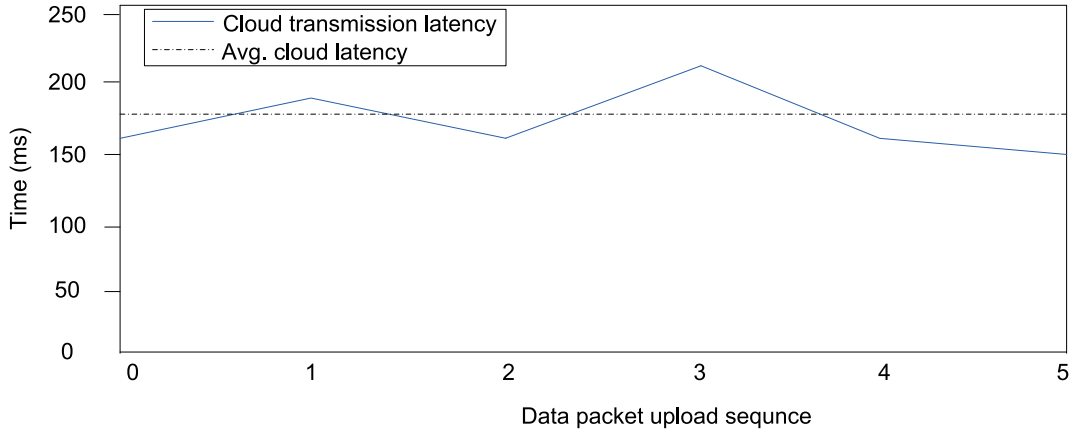

Fig. 14. ET-IoT latency while uploading data to the IoT cloud.

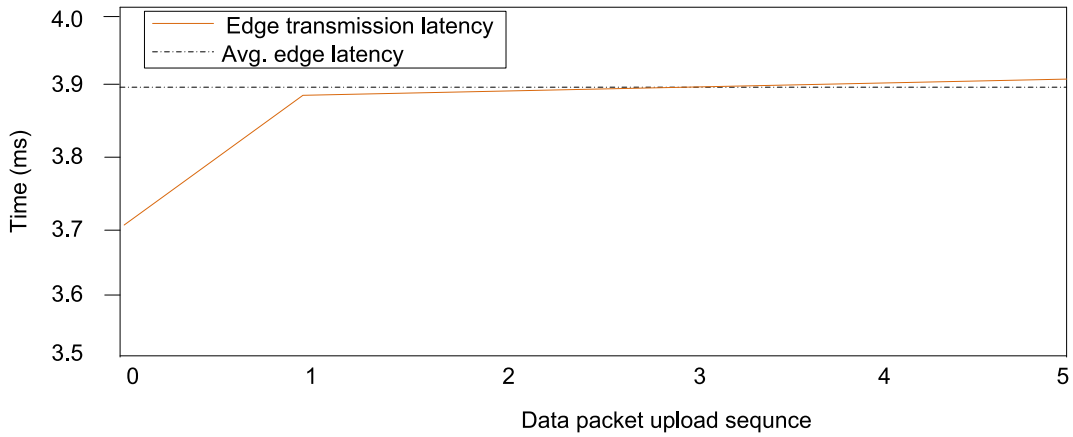

Fig. 15. Localized processing latency within ET-IoT.

are involved into this layer. Among the cloud connectivity protocols, MQTT, CoAP, AMQP, WebSocket, DNS-SD etc., are to name a few most viable opportunities. In the top-most layer, the cloud services orchestrate several different facilities for the underlying layers. Mainly, the device metadata management, data ingestion, access control and on-platform analytics are the commendable inclusions of the existing cloud services. Moreover, its enterprise application-perspectives can be associated with the device management core. As a whole, the layered components play very crucial role to infer meaningful and significant edge-IoT-based services to the stakeholders.

2. *Functionalities:* There exist 6 key functionalities that are inferred from the edge-IoT architectures such as, (i) edge-IoT application: IoT application deployed to the end executed from the edge, (ii) edge rule engine: ability to execute several actions that includes notifications and callouts executed at the edge, (iii) edge-cloud connectivity: serves operations such as, device data transfer, data aggregation points and edge-IoT gateway provisioning (iv) edge analytics: quantitative exploration of actual data and meta data at the edge, (v) edge data normalization: conversion of machine and unstructured data into compressed and structured digital data, and (vi) edge data storage: storage and assessment of long-term and transient machine data at the cloud repository. Fig. 17 presents the key functionalities in brief.
3. *Operational issues:* The edge-IoT covers many operational issues that originate during real-life and industrial applications scenarios. The main tasks of the operational issue handling involve the following such as, (i) improving processing speed, (ii) reducing data security issues, (iii) improving asset performance, (iv) reducing latency issues, (v) analyzing and controlling of device, (vi) enhancement to reduce downtime and (vii) improvement and optimization of production. Herein presented issue solvers are in-deed important to get resolved in presented scenario of edge-IoT ecosystem and architecture. Fig. 18 elaborates the key issues with respect to the functionality of the edge-IoT platforms.

4.2. Edge-IoT requirement, capability and selection criteria

1. *Requirement category:* In deed an edge-IoT ecosystem has some requirements to successfully act on the related tasks. Four are of utmost importance such as, (i) edge data processing, (ii) edge system management, (iii) architecture and integration and, (iv) business strategy and business association. Efficient edge data collection, orchestration, filtering and aggregation are key, without which expected results may not arise. Similarly, stack-based architecture with full-fledged standardized protocol support is a must for an excellent quality edge-IoT ecosystem. Overall management is obviously a very important factor

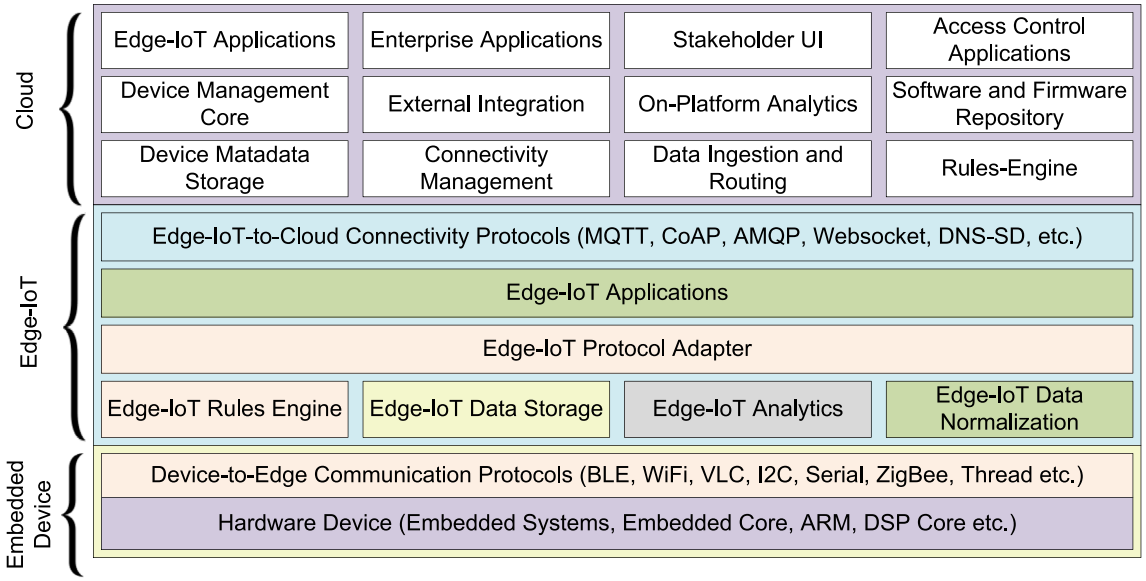

Fig. 16. Protocol-wise Edge-IoT ecosystem architecture (The MachNation IoT Architecture).

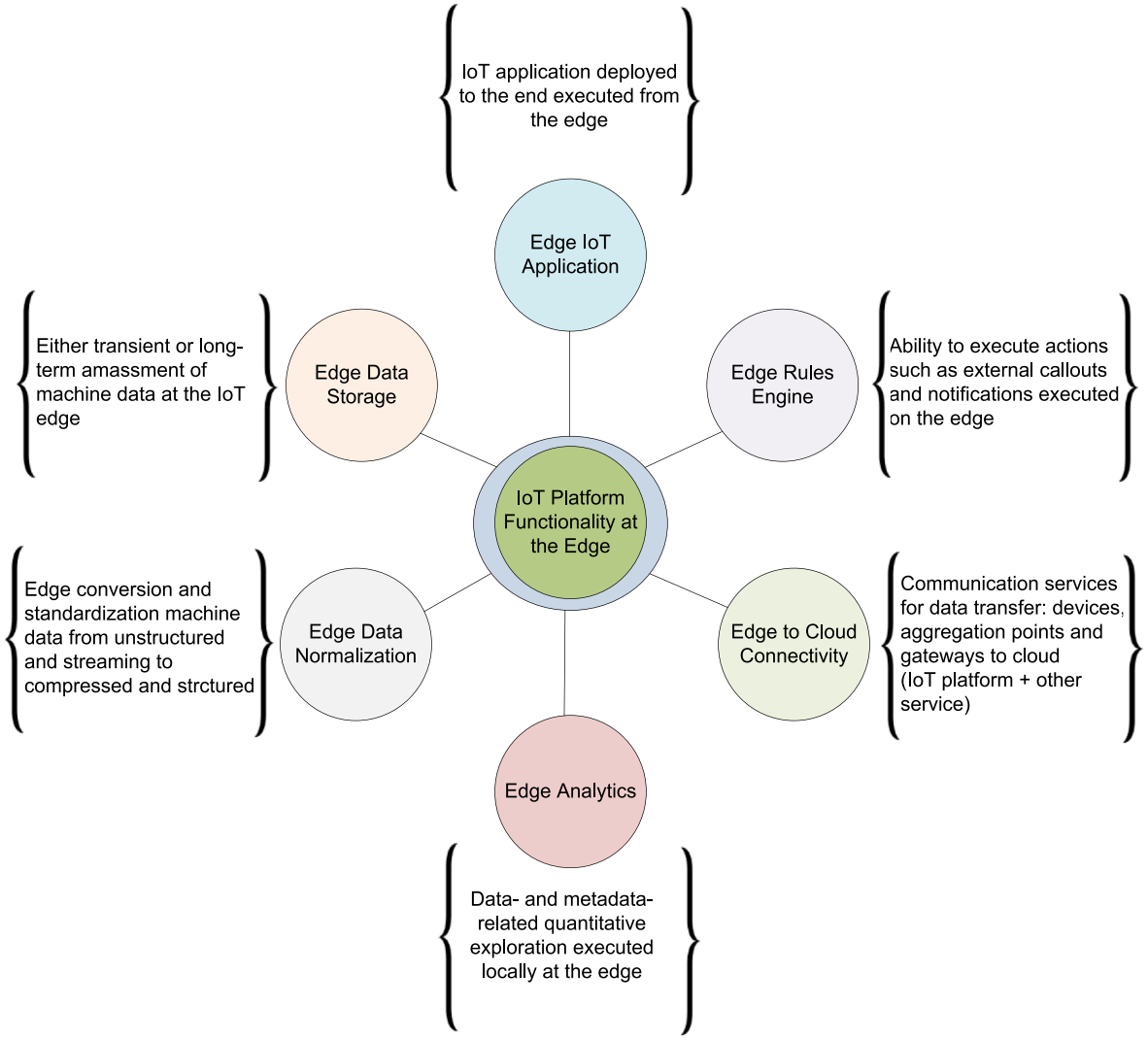

Fig. 17. IoT platform functionalities at the edge (Functional architecture of edge-IoT system).

that needs to be accurately handled. As edge-IoT ecosystem can leverage industrial-specific orientation, thus a state-of-the-art strategy and business management is required. Fig. 19 present the key requirements for related scenario (Selection of best IoT platform; Requirements of edge-IoT paltforms).

2. *Capability category:* As mentioned earlier, edge-IoT platform needs to have five key capabilities without which it may not succeed in solving the expected problems. Some of the extraordinary capabilities are as follows, (i) use of the comprehensive analytics and visualization tools, (ii) hardware agnostic scalable service, (iii) cloud-based orchestration capabilities to support device life cycle management, (iv) robust capability for offline functionality and (v) extensive protocol support for edge-IoT data ingestion. A, edge-IoT platform should follow the prescribed capability criteria to host and facilitate seamless support to underlying applications (IIoT in living industrial environments).
3. *Selection criteria:* At this end, it becomes very important to discuss about the important selection criteria of selecting any edge-IoT ecosystem for deployment into a real-life situation. Followings are inferred from specialized sources to get selectively attached with the edge-IoT ecosystem that includes: (i) hybrid deployment, (ii) device lifecycle management, (iii) connectivity networks, (iv) interoperability functions, (v) cloud platforms and strategy, (vi) application enablement, (vii) support for ecosystem alliances, (viii) rules engine and event processing, (ix) data normalization facility, (x) end-to-end security, (xi) user friendliness and efficiency, (xii) hardware assets, (xiii) scalability, flexibility and modularity, and (xiv) integration with business systems etc. It is understood that many other aspects could also be taken into consideration for selecting an edge-IoT ecosystem for any intensive work environment. However, it is advised to follow the prescribed criteria to initially screen the ecosystems before going into detailed analysis for a specific usage (Edge IoT specifications; Edge-IoT requirements and capability).

4.3. Future research directions

1. *Edge Gateway:* It is obvious that edge-IoT ecosystem relies on the end-network device i.e. gateway. Special attention shall be given toward development and integration of edge gateway to leverage such provisions. The edge gateway must be capable of catering near-by as well as long range network connectivity protocols. A nexus among edge gateways could be developed to create ad-hoc network persuasion.
2. *Edge Analytics:* Advanced, though light weight java script enabled frameworks and platforms may be used together to provide edge analytics service to the end-user. For instance, a patient may be facilitated with real-time prescription or diagnosis by the e-health based edge analytics service.

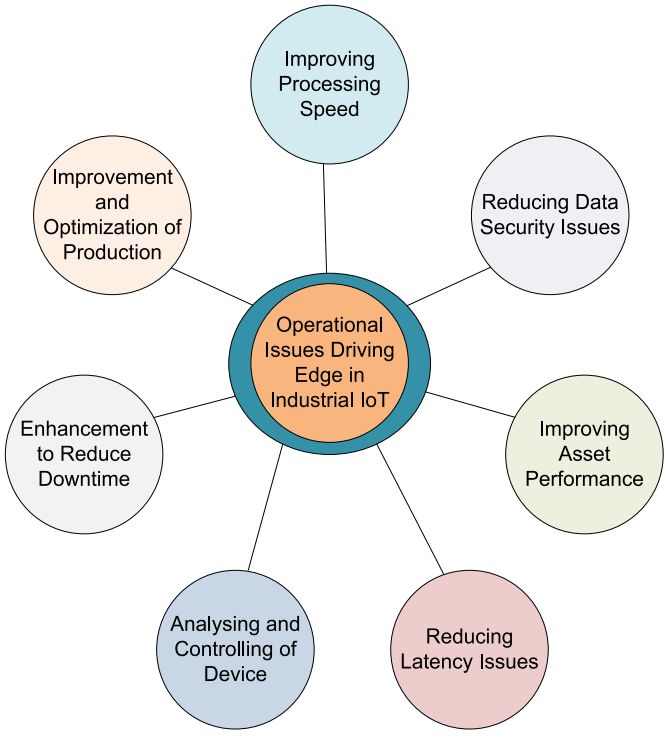

Fig. 18. Key factors driving the edge-IoT operational issues.

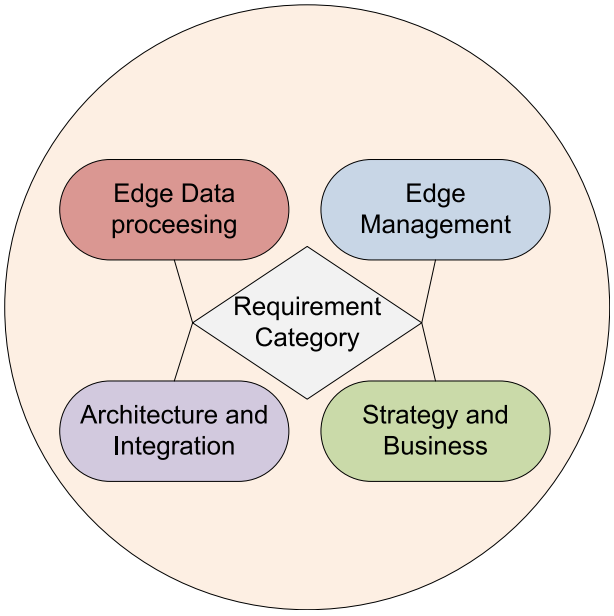

Fig. 19. Requirement category for edge-IoT ecosystem.

3. *Edge Architecture:* Novel edge-architecture should be prescribed to necessitate the obligations related to the edge-IoT ecosystem in future. While designing such architecture, following key point must be taken care of i.e. module-based interactions among the various system-components inside the edge platform. Layered architecture comprising different levels may play promising role in this regard.
4. *Sensor-Actuator Integration:* It is envisaged that the future edge-IoT ecosystem shall carry a number of smart sensor and actuators for particular application. Thus, a need of I2C and SPI centric common approach could be leveraged for the benefit of the better edge service. For example, sensors needing I2C could be assimilated with the actuators that require SPI interfacing with the system.
5. *Context-Awareness:* As the edge computing is inferred to cater the near real-time services to the end-users, it becomes an obligation for the edge service to incorporate context-awareness regarding the scenario. For instance, an e-health based edge service should be aware of the patient's disease pattern based on the seasonal change.
6. *Dew Computing:* Dew computing is a recent inclusion into the existing computing paradigm that solves the super-user experience and interwork agnostic approach to facilitate real-time service to the user at the extreme edge of any network. The main focus of dew computing is to articulate the natural phenomenon of actual the cloud-dew drop relationship into the digital domain (Wang, 2015; Hu et al., 2018; Wang, 2016; Doing More with the Dew). Dew computing utilizes the concept of the dew-cloud architecture, dew computer, dew DNS and related technologies to make the system more transparent and delay-sensitive (Cloud--dew architecture; Yingwei, 2015; Ray).
7. *Blockchain:* Blockchain is another option that could be associated with the future edge-IoT ecosystem. It is a continuously growing ledger that is immutable, permanent and open consensus protocol-based approach to leverage distributed and peer-to-peer communication without the need of centralized authority (Ray, 2019; Zheng et al., 2017). Thus, "Internet of Edge- Block" may be envisaged to assist the existing edge-IoT ecosystem to act autonomously and transparently while processing the decentralized end-users' requests.
8. *Digital Twin:* Albeit another form of novel technology called the Digital Twin is current getting popular that minimizes the process of data distribution between the local and remote cloud by creating a localized as well as a virtual copy to be stored at the remote site. Future edge-IoT service may be equipped with the digital twin capabilities to offer efficient data distribution, resulting minimal dependency over the network backhaul (Chen and Guan, 2018).
9. *Auto Machine Learning:* Data security is a key area where rigorous investigation should be performed, especially in security. Edge-IoT ecosystem being heterogeneous, autonomous and super flexible, there will be high chance of attacks into the system. Autonomic machine learning algorithms could be sought to fill this gap to assure the integrity of the end-user database.
10. *Unified framework of Integration:* As technology advances, various branches of platforms development approaches also grow. This generates a passive situation where a node belonging to a specific edge system may be technologically un-interoperable to the other category. Here, a unified framework of integrating all such notions should be merged to pacify the issues related to the interoperability (Ray et al., 2018b).

5. Conclusion

Edge computing plays significant role in IoT applications where instant processing of data is required. This paper first reviews the existing industrial standards and solutions available for the edge-IoT. A novel edge-IoT architecture is prosed and discussed. Further, discussions are made to identify the requirement, capability and selection criteria of the edge-IoT ecosystem. Moreover, its architecture, functionality and operational issues are discussed. A novel case study while utilizing the Edgent platform is proposed and developed. A physical demonstration against the e-healthcare application was deployed. The results and analysis section make it clear that the deployment of the EH-IoT system the bandwidth between the IoT cloud and local edge device can be increased and processing latency can be decreased significantly. If number of such devices are installed on every IoT applications one could imagine how much bandwidth could be saved and latency could be reduced providing quality of service.

Acknowledgement

We are thankful to Mr. Arun Subba and Mr. Kiran Regmi for their support to conduct this research.

References

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B., 2012. A survey of information-centric networking. IEEE Commun. Mag. 50 (7), 26–36.
Ali, S., Ghazal, M., Apr. 2017. Real-time heart attack mobile detection service (RHAMDS): an IoT use case for software dependent networks. In: Proc IEEE 30th Can. Conf. Elect. Comput. Eng. (CCECE), pp. 1–6.
Atat, R., Liu, L., Chen, H., Wu, J., Li, H., Yi, Y., 2017. Enabling cyber-physical communication in 5g cellular networks: challenges, spatial spectrum sensing, and cyber-security. IET Cyber-Phys. Syst.: Theor. Appl. 2 (1), 49–54.
Bonomi, F., Milito, R., Natarajan, P., Zhu, J., Mar 2014. Fog computing: a platform for Internet of Things and analytics. In: Big Data and Internet of Things: A Roadmap for Smart Environments. Springer, pp. 169–186.
Canzian, L., Van Der Schaar, M., 2015. Real-time stream mining: online knowledge extraction using classifier networks. IEEE Netw. 29 (5), 10–16.
Chen, Xiuyu, Guan, Tianyi, 2018. Research on the predicting model of convenience store model based on digital twins. In: IEEE International Conference on Smart Grid and Electrical Automation (ICSGEA), pp. 224–226.
Chiang, Zhang, T., 2016. Fog and IoT: an overview of research opportunities. IEEE Internet Things J. PP (99), 1.
Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., Patti, A., 2011. Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the Sixth Conference on Computer Systems. ACM, pp. 301–314.
Cloud-dew architecture: realizing the potential of distributed database systems in unreliable networks - Semantic Scholar. Retrieved 2018-06-30.
Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P., 2010. MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. ACM, pp. 49–62.
Dinh, T., Lee, C., Niyato, D., Wang, P., Dec 2013. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Commun. Mobile Comput. 13 (18), 1587–1611.
Doing More with the Dew: A New Approach to Cloud-Dew Architecture - Semantic Scholar. Retrieved 2018-06-30.
Dutta, S., Taleb, T., Ksentini, A., 2016. QoE-aware elasticity support in cloud-native 5G systems. In: IEEE Int'l. Conf. On Comm. (ICC 2016). IEEE, pp. 1–6.
Edge IoT protocol stack. https://www.open-silicon.com/iot-edge-soc-platform/. (Accessed June 2018).
Edge IoT specifications. twitter.com/CraigDResnick/status/986604163067867141. (Accessed October 2018).
Edge-IoT requirements and capability. www.idc.com/getdoc.jsp?containerID=EMEA43589818. (Accessed October 2018).
El-Sayed, Hesham, Sankar, Sharmi, Prasad, Mukesh, Puthal, Deepak, Gupta, Akshansh, Mohanty, Manoranjan, Lin, Chin-Teng, Fellow, IEEE, 2018. Edge of things: the big picture on the integration of edge, IoT and the Cloud in a Distributed Computing Environment, vol. 6.
Elias, R., Golubovic, N., Krintz, C., Wolski, R., 2017. Where's the bear?-automating wildlife image processing using iot and edge cloud systems. In: Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on. IEEE, pp. 247–258.
Elkhatib, Y., Porter, B., Ribeiro, H.B., Zhani, M.F., Qadir, J., Rivi`ere, E., 2017. On using micro-clouds to deliver the fog. IEEE Internet Comput. 21 (2), 8–15.
Farris, I. Militano, Nitti, M., Atzori, L., Iera, A., 2015. Federated edge-assisted mobile clouds for service provisioning in heterogeneous iot environments. In: Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, pp. 591–596.
Feng, J., Liu, Z., Wu, C., Ji, Y., Dec. 2017. AVE: autonomous vehicular edge computing framework with ACO-based scheduling. IEEE Trans. Veh. Technol. 66 (12), 10660–10675. https://doi.org/10.1109/TVT.2017.2714704.
Fitzgerald, D., Trakarnratanakul, N., Dunne, L., Smyth, B., Caulfield, B., 2008. Development and user evaluation of a virtual rehabilitation system for wobble board balance training. In: Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE. IEEE, pp. 4194–4198.
Functional architecture of edge-IoT system. Dima Tokar-Machnation@machnationiot. https://www.machnation.com/2017/09/18/functional-architecture-iot-platforms/?utm_source=iscoop. (Accessed August 2018).
Gordon, M.S., Jamshidi, D.A., Mahlke, S., Mao, Z.M., Chen, X., 2012. COMET: code offload by migrating execution transparently. In: Pre- Sented as Part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), pp. 93–106.
Griffin, D., Rio, M., Simoens, P., Smet, P., Vandeputte, F., Vermoesen, L., Bursztynowski, D., Schamel, F., June 2014. Service oriented networking. In: EuCNC 2014.
Habak, K., Ammar, M., Harras, K.A., Zegura, E., 2015. Femto clouds: leveraging mobile devices to provide cloud service at the edge. In: 2015 IEEE 8th International Conference on Cloud Computing. IEEE, pp. 9–16.
Hong, H.-J., Chen, D.-Y., Huang, C.-Y., Chen, K.-T., Hsu, C.-H., 2015. Placing virtual machines to optimize cloud gaming experience. IEEE Trans. Cloud Comput. 3 (1), 42–53.
Hu, Yu-Chen, Tiwari, Shailesh, Mishra, Krishn K., Trivedi, Munesh C. (Eds.), 2018. Intelligent Communication and Computational Technologies. Lecture Notes in Networks and Systems. ISSN: 2367-3370, vol. 19, ISBN 978-981-10-5522-5. https://doi.org/10.1007/978-981-10-5523-2.
Huang, C.-Y., Chen, K.-T., Chen, D.-Y., Hsu, H.-J., Hsu, C.-H., Feb.-March 2014. GamingAnywhere: the first open source cloud gaming system. In: Proc. 4th ACM Multimedia Systems Conf. (MMSys '13), Oslo, Norway.
IIoT in living industrial environments. www.arcweb.com/blog/iiot-living-edge-industrial-environments. (Accessed October 2018).
Jarschel, D. Schlosser, Scheuring, S., Hoßfeld, T., 2011. An evaluation of QoE in cloud gaming based on subjective tests. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on. IEEE, pp. 330–335.
Kämäräinen, T., Siekkinen, M., Xiao, Y., Ylä-Jääski, A., 2014. Towards pervasive and mobile gaming with distributed cloud infrastructure. In: 2014 13th Annual Workshop on Network and Systems Support for Games. IEEE, pp. 1–6.
Kumar, K., Liu, J., Lu, Y.-H., Bhargava, B., 2013. A survey of computation offloading for mobile systems. Mob. Netw. Appl. 18 (1), 129–140.
Lin, Y., Kämäräinen, T., Di Francesco, M., Ylä-Jääski, A., December 2015. Performance evaluation of remote display access for mobile cloud computing. Comput. Commun. 72, 17–25.
Madukwe, Kosisochukwu J., Ezika, Ijeoma J.F., Iloanusi, Ogechukwu N., 2017. Leveraging edge analysis for internet of things based healthcare solutions. In: 2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON).
Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B., 2017. A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor.
McKeown, T. Anderson, Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.
Pan, Jianli, McElhannon, James, Feb. 2018. Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J. 5 (1), 439–449.
Pan, J., Ma, L., Ravindran, R., TalebiFard, P., 2016. Homecloud: an edge cloud framework and testbed for new application delivery. In: Telecommunications (ICT), 2016 23rd International Conference on. IEEE, pp. 1–6.
Peterson, L., Al-Shabibi, A., Anshutz, T., Baker, S., Bavier, A., Das, S., Hart, J., Palukar, G., Snow, W., 2016. Central office re-architected as a data center (cord). IEEE Commun. Mag. 54 (10), 96–101.
Plenar.io. http://plenar.io. (Accessed June 2018).
Premsankar, Gopika, Di Francesco, Mario, Taleb, Tarik, 2018. Edge computing for the internet of things: a case study. IEEE Internet Things J. 5 (2), 1275–1284.
Ra, M.-R., Sheth, A., Mummert, L., Pillai, P., Wetherall, D., Govin- dan, R., 2011. Odessa: enabling interactive perception applications on mobile devices. In: Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services. ACM, pp. 43–56.
Ranjan, R., Benatallah, B., Dustdar, S., Papazoglou, M.P., 2015. Cloud resource orchestration programming: overview, issues, and directions. IEEE Internet Comput. 19 (5), 46–56.
Ray, P.P., An Introduction to Dew Computing: Definition, Concept and Implications IEEE Journals & Magazine. ieeexplore.ieee.org. Retrieved 2018-06-30.
Ray, P.P., 2014a. Home health hub internet of things (H3IoT): an architectural framework for monitoring health of elderly people. In: Proceeding of IEEE ICSEMR, ISBN 9789380222745, pp. 1–4. https://doi.org/10.1109/ICSEMR.2014.7043542. Chennai.
Ray, P.P., 2014b. Internet of things based physical activity monitoring (PAMIoT): an architectural framework to monitor human physical activity. In: Proceeding of IEEE CALCON, Kolkata, pp. 32–34.
Ray, P.,P., 2015a. A generic internet of things architecture for smart sports. In: IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, pp. 405–410. https://doi.org/10.1109/ICCICCT.2015.7475313.
Ray, P.,P., 2015b. Internet of things based smart measurement and monitoring of wood equilibrium moisture content. In: IEEE IEEE International Conference on Smart Structures and Systems (ICSSS). https://doi.org/10.1109/SMARTSENS.2015.7873612.
Ray, P.P., 2015c. Internet of Things for Sports (IoTSport): an architectural framework for sports and recreational activity. In: Proceeding of IEEE International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO), Vizag, pp. 79–83. https://doi.org/10.1109/EESCO.2015.7253963.
Ray, P.,P., 2015d. Towards an internet of things based architectural framework for defence. In: IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, pp. 411–416. https://doi.org/10.1109/ICCICCT.2015.7475314.
Ray, P.P., 2016a. A survey on internet of things architectures. J. King Saud Univ. - Comput. Inf. Sci. https://doi.org/10.1016/j.jksuci.2016.10.003. Elsevier.
Ray, P.,P., 2016b. A survey of IoT cloud platforms. Future Comput. Inf. J. 1 (1–2), 35–46. https://doi.org/10.1016/j.fcij.2017.02.001. Elsevier.
Ray, P.,P., 2016c. Towards internet of things based society. In: IEEE International Conference on Signal Processing, Communication & Embedded Systems (SCOPES), Paralakhemundi, Odisa, India, pp. 345–352. https://doi.org/10.1109/SCOPES.2016.7955849.
Ray, P.,P., 2016d. Internet of things cloud based smart monitoring of air borne PM2.5 density level. In: IEEE International Conference on Signal Processing, Communication & Embedded Systems (SCOPES), Paralakhemundi, Odisa, India, pp. 995–999. https://doi.org/10.1109/SCOPES.2016.7955590.

Ray, P.,P., 2016e. Communicating through visible light: internet of things perspective. Current Sc. 111 (12), 1903–1905. Indian Academy of Sciences.
Ray, P.P., 2016f. Internet of robotic things: concept, technologies and challenges. IEEE Access 99. https://doi.org/10.1109/ACCESS.2017.2647747.
Ray, P.,P., 2016g. Creating values out of internet of things: an industrial perspective. J. Comput. Netw. Commun. https://doi.org/10.1155/2016/1579460. Hindawi.
Ray, P.,P., 2016h. An internet of things based approach to thermal comfort measurement and monitoring". In: IEEE International Conference on Advances in Computing and Communications (ICACCS), pp. 1–7. https://doi.org/10.1109/ICACCS.2016.7586398, 2015.
Ray, P.P., 2016i. Internet of things cloud enabled MISSENARD index measurement for indoor occupants. Measurement 92, 157–165. https://doi.org/10.1016/j.measurement.2016.06.014. Elsevier.
Ray, P.P., 2017a. Obligations behind quantum internet dream. Curr. Sci. 112 (11), 2175–2176.
Ray, P.P., 2017b. Data analytics: India needs agency for health data. Curr. Sci. 112 (6), 1082.
Ray, P.,P., 2017c. Internet of things for smart agriculture: technologies, practices and future road map. J. Ambient Intell. Smart Environ. 9, 395–420. https://doi.org/10.3233/AIS-170440. IOS Press.
Ray, P.,P., 2017d. Understanding the Role of Internet of Things Towards Providing Smart e-Healthcare Services. Bio Med. Res. 28 (4), 1604–1609. Allied Academics.
Ray, P.P., 2017e. An IR sensor based smart system to approximate body core temperature. J. Med. Syst. 41, 123. https://doi.org/10.1007/s10916-017-0770-z. Springer.
Ray, P.P., 2017f. A survey on visual programming languages in internet of things. Sci. Program. https://doi.org/10.1155/2017/1231430. Hindawi.
Ray, P.P., 2019. Minimizing dependency on internetwork: is dew computing a solution? Trans. Emerg. Telecommun. https://doi.org/10.1002/ett.3496. Wiley.
Ray, P.,P., Agarwal, S., 2016. Bluetooth 5 and internet of things: potential and architecture. In: IEEE International Conference on Signal Processing, Communication & Embedded Systems (SCOPES), Paralakhemundi, Odisa, India, pp. 1461–1465. https://doi.org/10.1109/SCOPES.2016.7955682.
Ray, P.P., Mukherjee, M., Shu, L., 2017. Internet of things for disaster management: state-of-the-art, challenges, and future road map. IEEE Access 5 (1), 18818–18835. https://doi.org/10.1109/ACCESS.2017.2752174.
Ray, P.P., Dash, D., De, D., 2018a. Approximation of Fruit Ripening Quality Index for IoT based Assistive e-Healthcare. Microsyst. Technol. Springer.
Ray, P.P., Dash, D., De, D., 2018b. Internet of Things-based Real-Time Model Study on e-Healthcare: device, Message Service and Dew Computing. Comput. Network. Elsevier.
Ren, Ju, Pan, Yi, Goscinski, Andrzej, Beyah, Raheem A., Jan.-Feb. 2018. Edge computing for the internet of things. IEEE Netw. 32 (1), 6–7.
Requirements of edge-IoT paltforms. https://www.networkworld.com/article/3247801/internet-of-things/the-top-5-user-requirements-of-iot-edge-platforms.html. (Accessed September 2018).
Ryden, M., Oh, K., Chandra, A., Weissman, J., 2014. Nebula: distributed edge cloud for data intensive computing. In: Cloud Engineering (IC2E),s 2014 IEEE International Conference on. IEEE, pp. 57–66.
Sapienza, M., Guardo, E., Cavallo, M., La Torre, G., Leombruno, G., Tomarchio, O., May 2016. Solving critical events through mobile edge computing: an approach for smart cities. In: Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), pp. 1–5.
Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N., Oct.-Dec. 2009. The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. 8 (4), 14–23.
Satyanarayanan, P. Simoens, Xiao, Y., Pillai, P., Chen, Z., Ha, K., Hu, W., Amos, B., Apr.-June 2015. Edge analytics in the internet of things. IEEE Pervasive Comput. 14 (2), 24–31.
Selection of best IoT platform. https://www.i-scoop.eu/best-iot-platform-selection-criteria/. (Accessed September 2018).
Taleb, T., Ksentini, A., Jantti, R., Nov 2016. Anything as a Service for 5G mobile systems. IEEE Netw. PP (99), 12–19.
The MachNation IoT Architecture with IoT platform functions divided into 8 categories (the colors) on the levels of device, edge and cloud. https://www.machnation.com/2017/09/18/functional-architecture-iot-platforms/?utm_source=iscoop. (Accessed July 2018).
Vallati, C., Virdis, A., Mingozzi, E., Stea, G., Oct. 2016. Mobile-edge computing come home connecting things in future smart homes using LTE device-to-device communications. IEEE Consum. Electron. Mag. 5 (4), 77–83.
Waggle-based sensor to cloud information exchange architecture. https://wa8.gl/architecture/software/. (Accessed July 2018).
Wang, Yingwei, 2015-09-16. Cloud-dew architecture. Int. J. Cloud Comput. (IJCC) 4 (3), 199–210.
Wang, Yingwei, 2016. Definition and categorization of dew computing. Open J. Cloud Comput. ISSN: 2199-1987 3 (1).
Wang, Y. Wang, Sun, Y., Guo, S., Wu, J., 2016. Green industrial internet of things architecture: an energy efficient perspective. IEEE Commun. Mag. 54 (12), 48–54.
Wang, J., Pan, J., Esposito, F., October 14, 2017. Elastic urban video surveillance system using edge computing. In: ACM Workshop on Smart Internet of Things 2017 (SmartIoT 2017), San Jose, CA.
Wu, S. Guo, Li, J., Zeng, D., 2016. Big data meet green challenges: big data toward green applications. IEEE Syst. J. 10 (3), 888–900.
Xia, W., Zhao, P., Wen, Y., Xie, H., 2017. A survey on data center networking (dcn): infrastructure and operations. EEE Commun. Surv. Tutor. 19 (1), 640–656.
Yingwei, Wang, 2015. The Initial Definition of Dew Computing. Dew Computing Research.
Zhang, K., Mao, Y., Leng, S., He, Y., Zhang, Y., Jun. 2017. Mobile-edge computing for vehicular networks: a promising network paradigm with predictive offloading. IEEE Veh. Technol. Mag. 12 (2), 36–44. https://doi.org/10.1109/MVT.2017.2668838.
Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H., 2017. An overview of Blockchain technology: architecture, consensus, and future trends. In: 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, pp. 557–564.

Partha Pratim Ray received the B.Tech. degree in computer science and engineering and the M.Tech. degree in electronics and communication engineering, with specialization in embedded systems, from the West Bengal University of Technology, Kolkata, India, in 2008 and 2011, respectively. He is currently a full-time Assistant Professor with the Department of Computer Applications, Sikkim University, Gangtok, India. His research interests include Internet of Things, Dew computing, and Pervasive bio-medical informatics. He received the VIRA Young Scientist Award and Bharat Vikas Award in 2017, for outstanding contribution in his field.

Dr. Dinesh Dash has received PhD from IIT Kharagpur, M.Tech from the University of Calcutta in 2013, 2004, respectively. Currently he is Assistant Professor in Computer Science and Engineering, National Institute of Technology, Patna, India. His area of interest includes Sensor Network, Mobile AdHoc Network, Algorithm, and Computational Geometry. He has served as the reviewer of IEEE Transaction on Mobile Computing and IEEE Access. He has received research grants from the Science & Engineering Research Board, DST Govt. of India.

Prof. Debashis De earned his M.Tech from the University of Calcutta in 2002 and his Ph.D (Engineering) from Jadavpur University in 2005. He is the Professor in the Department of Computer Science and Engineering of the West Bengal University of Technology, India, and Adjunct research fellow at the University of Western Australia, Australia.He is a senior member of the IEEE. Life Member of CSI and member of the International Union of Radio science. He worked as R&D engineer for Telektronics and programmer at Cognizant Technology Solutions. He was awarded the prestigious Boyscast Fellowship by the Department of Science and Technology, Government of India, to work at the Herriot-Watt University, Scotland, UK. He received the Endeavour Fellowship Award during 2008–2009 by DEST Australia to work at the University of Western Australia. He received the Young Scientist award both in 2005 at New Delhi and in 2011 at Istanbul, Turkey, from the International Union of Radio Science, Head Quarter, Belgium. His research interests include mobile cloud computing, Green mobile networks, and nanodevice designing for mobile applications. He has published in more than 200 peer-reviewed international journals in IEEE, IET, Elsevier, Springer, World Scientific, Wiley, IETE, Taylor Francis and ASP, seventy International conference papers, four researches monographs in springer, CRC, NOVA and ten text books published by Person education.