Future Generation Computer Systems 122 (2021) 40–51

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

An AI-enabled lightweight data fusion and load optimization approach for Internet of Things

Mian Ahmad Jan [a,*], Muhammad Zakarya [a,*], Muhammad Khan [b], Spyridon Mastorakis [c], Varun G. Menon [d], Venki Balasubramanian [e], Ateeq Ur Rehman [a]

[a] Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan
[b] ComNets Lab, Department of Computer Science, New York University, Abu Dhabi, United Arab Emirates
[c] College of Information Science & Technology, University of Nebraska Omaha, USA
[d] Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam 683576, India
[e] School of Science, Engineering and Information Technology, Federation University, Mount Helen, VIC 3350, Australia

ARTICLE INFO

Article history:
Received 6 October 2020
Received in revised form 3 March 2021
Accepted 17 March 2021
Available online 8 April 2021

Keywords:
Internet of Things
Data fusion
Load optimization
Evolutionary algorithms
Gateway-Edge configuration
Service migration

ABSTRACT

In the densely populated Internet of Things (IoT) applications, sensing range of the nodes might overlap frequently. In these applications, the nodes gather highly correlated and redundant data in their vicinity. Processing these data depletes the energy of nodes and their upstream transmission towards remote datacentres, in the fog infrastructure, may result in an unbalanced load at the network gateways and edge servers. Due to heterogeneity of edge servers, few of them might be overwhelmed while others may remain less-utilized. As a result, time-critical and delay-sensitive applications may experience excessive delays, packet loss, and degradation in their Quality of Service (QoS). To ensure QoS of IoT applications, in this paper, we eliminate correlation in the gathered data via a lightweight data fusion approach. The buffer of each node is partitioned into strata that broadcast only non-correlated data to edge servers via the network gateways. Furthermore, we propose a dynamic service migration technique to reconfigure the load across various edge servers. We assume this as an optimization problem and use two meta-heuristic algorithms, along with a migration approach, to maintain an optimal Gateway-Edge configuration in the network. These algorithms monitor the load at each server, and once it surpasses a threshold value (which is dynamically computed with a simple machine learning method), an exhaustive search is performed for an optimal and balanced periodic reconfiguration. The experimental results of our approach justify its efficiency for large-scale and densely populated IoT applications.

1. Introduction

In the Internet of Things (IoT), the sensor nodes of various applications gather highly correlated data in their neighbourhoods that affect the outcome of any decision made at the cloud data centres [1,2]. In these applications, the data are unstructured, intermittent and somewhat dynamic. The raw data gathered by the nodes need to be processed locally and analysed at the edge and cloud data centres to optimize the usage of available resources. The raw data need to be fused within the network to reduce the correlation in them. Each node, unaware of its neighbour's sensing range, gathers data in its neighbourhood. The sensing range of two or more nodes may overlap leading to the aggregation of similar data [3]. Each node needs to perform local data fusion to discard multiple copies of the same data. In-network data fusion alleviates the redundancy to trade-off the volumes of data and the available resources at the edge and cloud data centres [4]. The presence of resource-starving nodes means that a data fusion approach needs to be lightweight, robust, and scalable, based on application requirements.

Data fusion alone is not enough to optimize the usage of available network resources. The upstream fused data toward the cloud data centres need to be fairly distributed among the edge servers [5,6]. In the existing literature [7–9], the fused data streams are offloaded to the nearest edge servers. However, this approach is not efficient as some of these servers may overload quickly in comparison to others that remain underutilized. The underlying nodes and network gateways associated with the over-utilized servers may suffer higher latency, packet drop, and bandwidth consumption. For a fair distribution of the network load, a dynamic load balancing approach needs to be adopted to assign the time-consuming tasks to underutilized servers. Based

* Corresponding authors.
E-mail addresses: mianjan@awkum.edu.pk (M.A. Jan), mohd.zakarya@awkum.edu.pk (M. Zakarya).

https://doi.org/10.1016/j.future.2021.03.020
0167-739X/© 2021 Elsevier B.V. All rights reserved.

on the run-time load at the servers, a decision needs to be made for the assignment of data streams. The configuration of network entities needs to be constantly monitored for an optimized and balanced load. Artificial Intelligence (AI)-enabled algorithms can manage the complex relationship among the network entities [10,11]. These algorithms need to be adopted for intelligent load balancing and optimization of the selected paths for reliable transmission of the fused data. They have the ability to reconfigure the devices' connectivity based on their experienced load.

Moreover, heterogeneity of the edge servers and network bandwidth may generate opportunities for application migrations (running within virtual machines) which could be beneficial in further load-balancing, avoiding stranded (wasted) resources, and performance degradation (due to overload situations). Stranded resources are those which cannot be allocated due to the unavailability of another resource e.g. CPU cores are fully utilized but memory is half utilized — half memory cannot be allocated because there are no CPU cores available to run the VM/workload/application. Here, heterogeneity refers to the speed of server to process data or network bandwidth. This is achieved through comparing the current utilization levels of the edge servers and/or the rate of transferring over a network link (channel conditions) to some pre-defined threshold values. If utilization level of an edge server or a network link surpasses a particular threshold value, migrations will happen. However, a static threshold may not be appropriate; therefore, we use a simple machine learning model to compute an adaptive threshold.

In this paper, we propose a novel data fusion and load optimization approach for the IoT-enabled applications. Our approach reduces data redundancy at the node level and fairly distributes the fused data streams among the edge servers. It is scalable and can be used by any application, provided that the threshold values for monitored data are known. It ensures the availability of high-quality data at the cloud data centres for decision-making. The main contributions of this work are as follows.

1. A lightweight data fusion approach that reduces the correlation and redundancy in the gathered data by using *MiniMax* stratified sampling. The buffer of each node is partitioned into multiple stratum, each one holding only two values, i.e., a minimum (*min*) and a maximum (*max*). A comparison with *min* and *max* decides to discard or retain any newly sensed data. After a sampling interval, the stratum of each node transmits only two data readings by discarding all other correlated readings.
2. A dynamic load optimization approach that maintains a balanced traffic in the network using a real-valued Genetic Algorithm (GA) and Discrete Particle Swarm Optimization (DPSO). A Software-Defined Networking (SDN) controller monitors the load on individual edge servers and reconfigures the current Gateway-Edge configuration if an unbalanced load is experienced. For reconfiguration, the SDN controller invokes these evolutionary algorithms to identify the transmission path for each gateway towards a prospective server.
3. The above contribution does not account for dynamic load-balancing, i.e., when on some particular resources, the data get processed quicker than others. A dynamic service migration technique is suggested to balance the load across several edge servers that triggers migration decisions, based on current resource (edge server, network channel) usage. A dynamic threshold is computed using a simple regression model in order to keep resources well-balanced.

The rest of our paper is organized as follows. In Section 2, we provide an overview of the background studies pertaining to our proposed approach. In Section 3, our proposed framework and algorithms are described in detail. This section also offers a service migration technique for load balancing across several edges. The experimental results and performance evaluation are sketched in Section 4. Finally, we provide concluding remarks and future research directions in Section 5.

2. Background

In this section, we provide the background studies pertaining to data fusion in the context of load optimization for IoT applications.

In [12], a cloud-based adaptive sensing belief propagation protocol (ASBP) was proposed. ASBP estimates the quality of links to determine the shortest routes toward the cloud for data gathered from IoT applications. The protocol exploits the spatio-temporal correlation among the data streams at cloud datacentres to reduce the energy consumption, and balance the load by keeping a subset of nodes in active states at a given time. ASBP, however, is unable to evenly distribute the load on edge servers for a large-scale IoT network. Besides, fusing massive amount of sensor data at the cloud incur a significant amount of transmission overhead. A dynamic sensor activation algorithm, SensorRank, was proposed to prioritize the deployed nodes based on their residual energy levels, their relative distance, and their links qualities [13]. SensorRank considered symmetric channels for data transmission among the neighbouring nodes. These channels may lead to an uneven load distribution among the nodes, and on the gateways and edge servers, respectively. A spatio-temporal based novel data mining approach (NDM) was proposed for the removal of redundant data, prior to upstream transmission towards the gateways [14]. NDM uses a packet classification approach to filter out redundant data to maintain the network load on the edge servers. NDM is non-scalable and its iterative nature of load distribution at the edge incurs an excessive overhead at the resource-constrained sensor nodes.

In [11], an optimized mobile sink-based load balancing (OMS-LB) protocol was proposed to achieve balanced load for a large-scale IoT network. OMS-LB offloads the computationally complex tasks from data gathering devices to a Software-defined Network (SDN) controller that is interfaced with cloud datacentres. The proposed protocol uses PSO and GA to determine the optimal paths for a mobile edge server and optimal data gathering points, i.e., gateways. OMS-LB does not define any criteria for data collection from an application perspective. Besides, the presence of a single server makes this protocol non-scalable, and vulnerable to security threats. A multi-edge based architecture was proposed for seamless integration of cloud datacentres in an IoT environment [5]. The proposed architecture used a multilevel protocol for gateways selection and AI-based load balancer for the identification of an optimal load distribution. However, the proposed architecture lacks any information about the heterogeneity of nodes, network latency and bandwidth requirements. In [15], the authors proposed a data aggregation scheme by estimating an accurate sensor matrix from the gathered raw data. A fog server is used to reconstruct the matrix that contains minimal noise and highly refined data. However, the proposed matrix does not take into account the load balancing issue and has limitations imposed on its scalability. Besides, it lacks any information on heterogeneous data fusion and interoperability of IoT devices.

All these existing approaches focused on centralized gateways and edge servers for load optimization and decision-making. The presence of centralized entities affect the scalability, fault tolerance and optimal load adjustment of a network. Besides, these approaches operate without data aggregation and fusion at the network level. As a result, they require excessive processing and storage of redundant data at the network gateways and edge servers. The transmission of redundant data ultimately deteriorate the QoS of an underlying network. In the IoT paradigms, it is inevitable to consider AI algorithms for maintaining a balanced load for various applications. There are numerous AI algorithms developed to resolve the load optimization problem. However, in this paper, we utilize the most embraced evolutionary algorithms, i.e., Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) [16–18]. GA and PSO are population-based algorithms, where the population means a group of all possible solutions, i.e., Gateway-Edge configurations (load balancing and optimization) [19].

GA is a bio-inspired search algorithm in which the population is referred to as a group of chromosomes. The genes of the fittest available chromosomes are utilized to generate new chromosomes, i.e., new optimal Gateway-Edge configurations, via mutation and crossover. On the other hand, in standard PSO, the population of all possible solutions is referred to as a swarm of particles. PSO is inspired by the social behaviour of swarms of ants, a flock of birds, a shoal of fish, etc. In all these cases, the swarm probe the search space for identifying the food with varying velocities. In the case of PSO, each particle is considered a candidate solution for the Gateway-Edge configuration problem. In the case of GA, each chromosome is considered a candidate solution. Since both these algorithms are not directly applicable to integer-based load optimization problems, we have developed a real-valued GA and a Discrete PSO (DPSO) for identifying the optimal Gateway-Edge configurations.

Load balancing is an essential part of the IoT, edge and cloud frameworks that could be achieved in two different ways: (i) dynamic service placement; and (ii) service migration. In respect of (i), two policies are suggested in [20]: cloud-only placement: place all application's modules in the server; and edge-ward placement: favour to run application's modules on various edge devices. Moreover, if allocation of an edge device is not suitable for a particular module, then either resources from other edge devices (server) could be provisioned or it could be migrated somewhere else. Empirical evaluation of both policies suggests that the edge-ward policy significantly improves the application's performance and reduces the network traffic. In respect of (ii), authors in [21,22] suggest that if an application's performance is the worst on a particular edge device (due to more number of connected sensor devices, network congestion etc.), then its migration either to the server or to another edge device could improve its performance and reduces network traffic. Moreover, mobility management in mobile edge clouds (MECs) also involves migrations [22].

Migrations could also be triggered to balance resource utilization levels of edge nodes. For example, if the utilization of an edge node increases certain threshold value (say 80%), some of the application's module may be moved to other edges. Migration can also take place when resources are under utilized i.e. threshold of 20%. This is done to conserve and consolidate resources to save energy [23]. In the later case, energy could be saved through migrating workloads from these underutilized servers to other servers; and switching them off. However, this may cause performance issues, in particular, if demand exceeds suddenly. We, in this paper, prefer the former one as our objective is not saving energy; instead we want to balance the load across different switched on servers. Furthermore, we use a dynamic threshold-based method that estimate these threshold values periodically — using Eq. (12). Service migration could only be achieved if various sand-boxing technologies such as virtualization, containerization are being used to virtualize the server and edge device resources [24]. In practice, resources in public clouds are virtualized, which increases resource utilization levels and saves energy. If various modules of a particular application are being run in a Virtual Machine (VM) or container; then the service can be migrated either off-line or live. In live migration, the service is moved transparently while still running; however, in off-line migration the service is stopped first, moved, and then resumed at the target edge. Using CRIU[1] technology, containers could be more quickly migrated than VMs. In case of live VM migration, where VMs data are kept on a shared storage reachable over the network, the time of migration T_{mig} depends on the total volume of memory used by the VM M_{vm} and available network bandwidth B_{total}. For virtual machine V_i the total migration time is given by:

$$T_{mig_{V_i}} = \frac{M_{V_i}}{B_{total}} \tag{1}$$

The above equation is used to compute only the migration time of a particular VM. Every VM for this T_{mig} time is considered offline, which is also called the downtime of the VM. The downtime (or performance loss) is dependent on the migration duration, as given by Eq. (1) [25]. Increased downtime results in poor performance; therefore it should be minimized for high availability of the datacenter. The performance degradation P_{deg} due to a single migration is calculated using the following formula (as given by Eq. (2)), where U_{V_i} is the CPU utilization of VM V_i, t_0 is the migration start time and 0.1 is the factor that shows the average performance degradation for web application i.e. 10% of the CPU utilization [23,26].

$$P_{deg_{V_i}} = 0.1 \times \int_{t_0}^{t_0+T_{mig_{V_i}}} U_{V_i}(t).dt \tag{2}$$

Note that, the above performance degradation model (10% loss in workload execution time) is benchmarked in [23,27]; and we assume that it already accounts for other time consuming activities such as: the time to initiate a VM migration; the time to transfer page files (dirty pages in case of live or online VM migration); the time to boot/spin up a new server (if there is no currently running server that can accept the VM being migrated); and the time to restart the VM (in the case of cold or offline VM migration) [28].

3. Data fusion and load optimization approach for IoT applications

In Fig. 1, the sensor nodes of various applications transmit their data to cloud data centres via the network gateways and edge servers. Among these applications, the smart city nodes gather and transmit highly correlated data streams. The transmission of these streams affects the decision-making at data centres and creates bandwidth bottlenecks for time-critical and delay-sensitive applications. Moreover, these applications experience excessive latency and degradation in the network throughput if an unbalanced load is experienced at the edge servers. An uneven load distribution results in some of the servers over-utilized while others remain underutilized. The unbalanced load leads to packet loss, longer delays, and network congestion. In this section, we discuss our proposed data fusion and load optimization approach to eliminate data redundancy and maintain a balanced load at the network entities.

[1] https://criu.org/Main_Page.

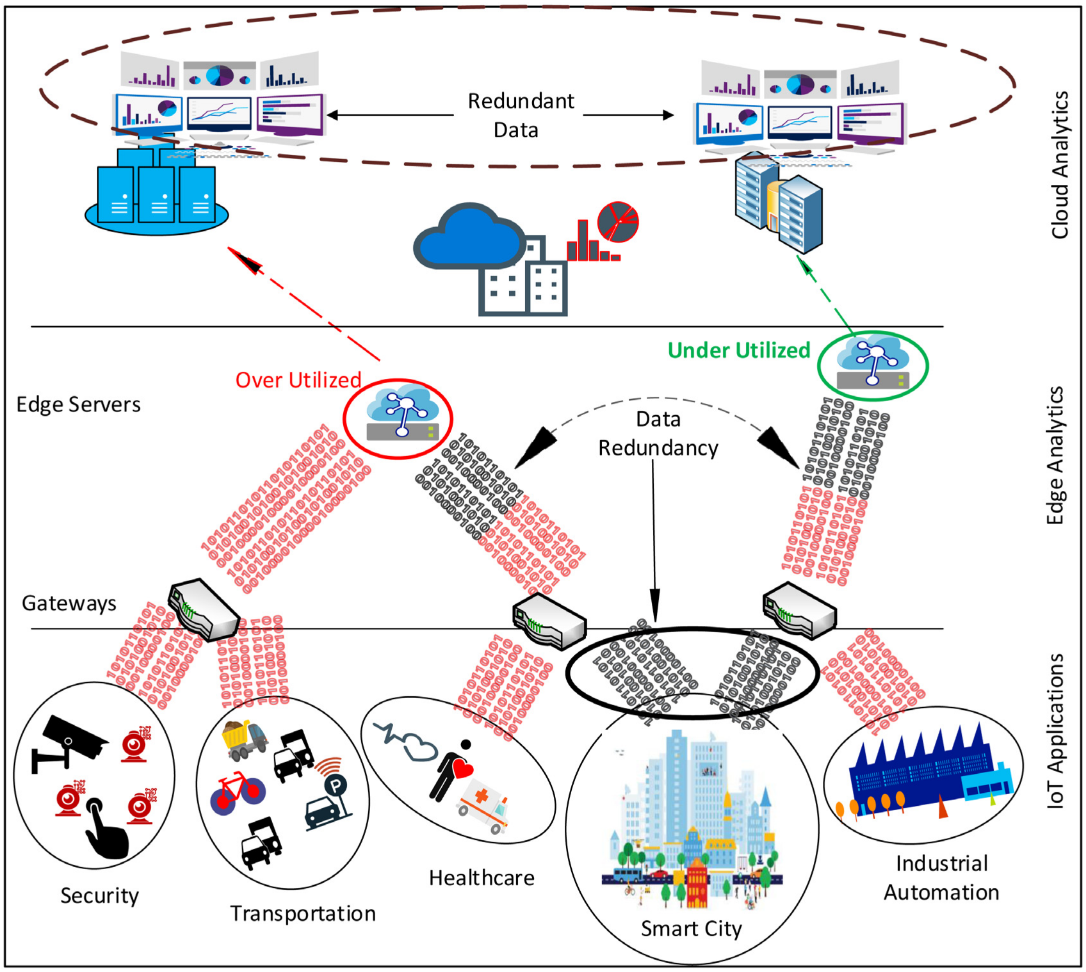


Fig. 1. Data redundancy and unbalanced load in IoT.

3.1. The Proposed Fog-IoT Framework

The proposed model consists of three layers i.e. the cloud layer, the edge layer and the local layer, as shown in Fig. 1. The local layer is responsible to gather important data (related to traffic, healthcare, and crowd, etc.) through various IoT devices and sensors. Once the data is collected, it is processed and/or stored at the edge layer through edge clouds [29]. The edge level processing may also include aggregation that could be achieved through removing redundant data. The filtered data can, then, be sent to the remote cloud layer for further processing such as storage, monitoring and resource management. Transferring the gathered data at local layer directly to the cloud layer, or through edge, may introduce significant delays in the cloud network, which is optimized through data fusion and load balancing methods as discussed in Sections 3.2 and 3.3.

The edge infrastructure is of great use when reading the stored data for processing through machine learning approaches. For example, real-time prediction of the traffic flow might happens at the edge layer, however, prediction for monitoring services (load, service migration) can be performed at the cloud layer [30]. Moreover, if real-time prediction, for example, shortest or safe route estimation, is carried out on the remote cloud, then it will incur significant delays depending on the network quality and capacity. In that scenario, the nearest edge cloud can predict the road conditions, congestion and distance; if the required data is stored locally. However, due to the least storage and processing capabilities of the edge clouds [31], the data may not be available or processed locally. In that case, there are three various options: (i) move the required data from the cloud to the edge, process, take decisions, and discard; (ii) perform the prediction at the remote cloud (in whole); and (iii) train the prediction model at the remote cloud and predict at edge layer (distributed fashion computation). Similarly, the huge amount of collected data may consist of duplicate values that could create network congestion and, therefore, affect the prediction process. The edge cloud can use fusion and aggregation technique to send only appropriate data to the remote cloud.

3.2. Data fusion

In a densely deployed smart city, energy hole problem is a common issue faced by the one-hop neighbours of the base station [32]. These one-hop neighbours not only transmit their own data but also relay the data of downstream nodes to the base station. As a result, their energy is depleted rapidly as compared to other nodes. To fill the void left by energy-depleted nodes and to maintain seamless network connectivity, one or more nodes may either sense multiple regions or move around the field to fill this gap. These nodes continuously sense and aggregate data in their neighbourhood, as shown in Fig. 2. Each node S maintains a coverage area based on its sensing range (R_s), and a radio coverage based on its communication range (R_c), respectively. The R_s enables efficient data monitoring, whereas the R_c ensures the upstream data transmission. These nodes can have a uniform or non-uniform coverage degree. The coverage

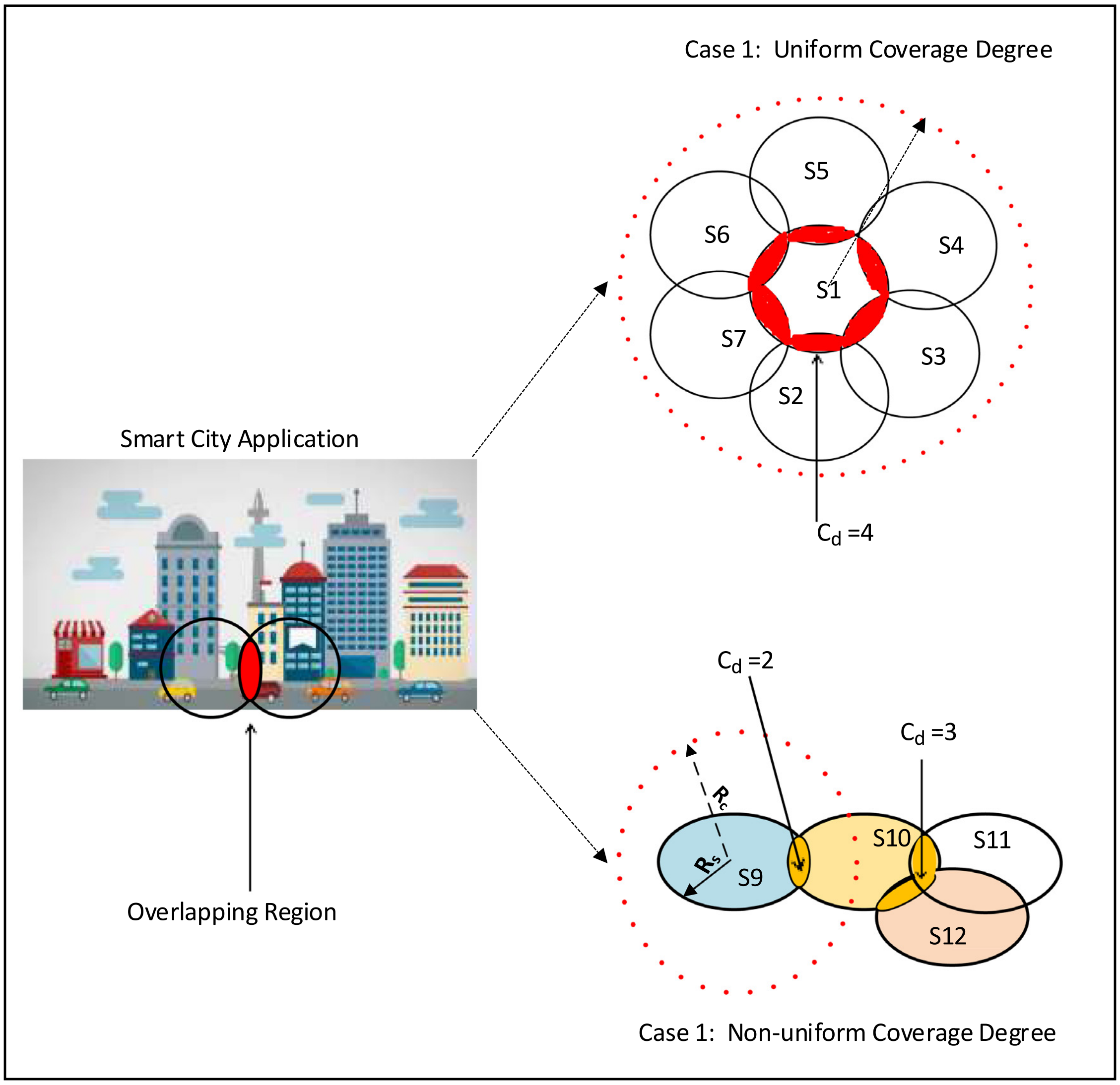


Fig. 2. Data Correlation of varying coverage degrees.

degree represents the number of nodes actively monitoring a particular region, i.e., an overlapped region. For uniform coverage, the value of correlation degree (C_d) remains constant for all the nodes. On the other hand, the value of C_d varies for some or all the nodes in a non-uniform coverage. A larger value of C_d represents highly correlated and redundant data as multiple nodes monitor a particular event in the overlapped region.

To eliminate the correlated and redundant data, we use a lightweight data fusion approach at the node level. The proposed approach uses a *minimax* function for the identification and removal of redundant data. In a smart city, each node is equipped with multiple sensors based on an application requirement. In this paper, we restrict our discussion to the temperature sensors only. However, the flexibility of our approach enables it to be extended for any application provided that the threshold values of monitored data are known. We classify the sensed temperature readings based on the correlation and similarity index among them. Each class, also known as stratum, contains a particular range of temperature readings. For each node, we define ten different stratum that are dynamic and depend on application requirements. They are designed using the concept of stratified sampling, a probability-based sampling technique [33,34].

The strata[2] are defined within the buffer of each node and can store temperature readings ranging from 20 °C to 39.99 °C. Based on these readings, each stratum holds a different range of varying values of up to 2 °C. For instance, the range of first stratum $\ddot{S}t_1$ is $\{20.00, \ldots, 21.99\}$, and the last stratum $\ddot{S}t_{10}$ is $\{38.00, \ldots, 39.99\}$, respectively. The outcome of each stratum can either be a *min* or *max* value. Each stratum has a mean value m that defines its *min* and *max*, respectively. In the beginning, when a new temperature reading T_i is sensed by a node, it is checked against strata of the given node to identify a destination stratum. Once a match is found, T_i is compared against m of the stratum. If T_i is less than m, it becomes the *min*, otherwise, it becomes the *max*, as shown in Eq. (3). The next time a new reading T_{i+1} is sensed within the range of the same stratum, it is compared against m. If T_{i+1} is less than m, a comparison is made with the new *min*. If T_{i+1} is less than *min*, the former turns out to be the new *min*, otherwise, it is discarded. A similar comparison is made with *max*. If T_{i+1} is greater than *max*, it turns out to be the new *max*, otherwise, it is discarded. Irrespective of T_i, T_{i+1} or any other subsequent readings, an exact match with the values

[2] Plural of stratum.

of m, *min* or *max* means that these readings will be discarded.

$$f(St_1, \ldots, St_{10}) = \begin{cases} min, & if \quad T_i < m, \\ 0, & if \quad T_i == m, \\ max, & if \quad T_i > m. \end{cases} \tag{3}$$

The *max* and *min* of a given stratum can be plotted as a stationary point on a curve. A point $P(x_0, f(x_0))$ is considered a stationary point of a function $f(x)$ if $(\frac{df}{dx})$ is 0 at x=x_o. Suppose a function y=$f(x)$ is a stationary point with x=x_o. Then

- if $[\frac{d^2f}{d^2x}]_{x=x_0} < 0$, then x=x_o is the *max* of a stratum.
- if $[\frac{d^2f}{d^2x}]_{x=x_0} > 0$, then x=x_o is the *min* of a stratum.
- if $[\frac{d^2f}{d^2x}]_{x=x_0}$ == 0, then
  - if $[\frac{df}{dx}]_{x=x_0} < 0$ for x>x_o, and $[\frac{df}{dx}]_{x=x_0} > 0$ for x<x_o, then x=x_o is the *max* of the given strata.
  - if $[\frac{df}{dx}]_{x=x_0} > 0$ for x>x_o, and $[\frac{df}{dx}]_{x=x_0} < 0$ for x<x_o, then x=x_o is the *min* of the given strata.

In our data fusion approach, the sampling rate of each node is S_r packets per second, where $S_r \geq 1$. The stratum of each node transmits only two packets, i.e., a *min* and a *max*, after every one minute. If a node constantly maintains its sampling rate at $S_r = 1$ for one minute, our approach achieves a maximum of 3 times reduction in the number of transmitted packets to the gateways. For S_r >1, the number of transmitted packets is reduced even further by a minimum of 3 times. In our approach, the S_r of a node and the number of transmitted packets from its strata to the gateways are inversely proportional to each other. These strata significantly reduce data redundancy, network latency, packet collision probability, and ultimately the network congestion. In our proposed approach, each node maintains a similarity index (Ω) for the data gathered over the S_r interval. The value of Ω ranges from 0.03 to 0.1. If Ω is equal to 0.03, it means that among two temperature packets within the range of 0.03 °C, only one will be retained. For example, in case of two packets with values 20 °C and 20.03 °C, only one will be retained in the stratum. Hence, larger the value of Ω, higher will be the rate at which the data are fused.

3.3. Load optimization

Upon data fusion, each node transmits the refined data to cloud data centres via the network gateways and edge servers. The gateways are relay nodes that need to be monitored for maintaining a balanced load at the edge servers. For this purpose, an optimal Gateway-Edge configuration is required. We use various Key Performance Indicators (KPIs) for an in-depth analysis of the network traffic to identify the optimal configuration. An SDN controller is used for identifying the transmission route for each gateway. It monitors the load on each server, and once it surpasses a threshold value, an alarm is raised to re-configure the current Gateway-Edge connection. If the Gateway-Edge configuration is known at a particular time t, then finding the optimal balanced Gateway-Edge configuration at time $t+1$ is a primary challenge. If N is the number of gateways, and M is the number of edge servers, then the Gateway-Edge configuration at a particular time t can be represented by a vector $G^t = \{G^t_1, G^t_2, \ldots, G^t_n\}$, where $G^t_n \in \{1, 2, \ldots, N\}$. As an example, $G^t_n = m$ means that the nth gateway is transmitting to an mth server at time t. Finding the optimal Gateway-Edge configuration vector at time $t+1$, i.e., $G^{t+1} = \{G^{t+1}_1, G^{t+1}_2, \ldots, G^{t+1}_n\}$, is a prime objective. To solve the Gateway-Edge configuration problem, we consider two KPIs, i.e., Average Residual Energy ($\mathbf{KPI}_{\text{ARE}}$) of the network and Load Fairness Index ($\mathbf{KPI}_{\text{LFI}}$) of the servers.

The LFI is monitored based on Jain's Fairness Index [35]. The normalized weighted sum of these two KPIs is taken into account to maximize the network performance (NP) at time t as shown in Eq. (4).

$$Max(NP) = \alpha \text{KPI}_{\text{ARE}} + \beta \text{KPI}_{\text{LFI}}. \tag{4}$$

Here, NP is a primary objective function for optimization problems, and α and β are the weights assigned to each KPI. These weights represent the priority level of each KPI in the objective function, as shown in Eq. (5).

$$\text{NP} = \alpha(\frac{1}{\text{N}_n} \sum_{i=1}^{\text{N}_n} \frac{\text{R}_i(t)}{\hat{\text{E}}}) + \beta(\frac{1}{\text{M}} \frac{\left(\sum_{i=1}^{\text{M}} \sum_{n=1}^{\text{N}} \text{I}_{n,i}\varphi_n(t) \setminus \varphi_{\max}\right)^2}{\sum_{i=1}^{\text{M}} \left(\sum_{n=1}^{\text{N}} \text{I}_{n,i}\varphi_n(t) \setminus \varphi_{\max}\right)^2}). \tag{5}$$

where, $\frac{R_i(t)}{\hat{E}}$ is the residual energy of a sensor node i at time t and is defined as the remaining energy ($R_i(t)$) of node i to the initial energy ($\hat{E}$) of each node at time t. For all the nodes in the network, $\hat{E}$ is similar at the time of deployment. For the second KPI, we consider the load fairness at the edge servers. $I_{n,i}$ is a binary indicator, i.e., $\text{I}_{n,i}$ is 1, if an nth gateway transmits φ_n packets to ith server at time t, otherwise, $\text{I}_{n,i}$ is 0.

To find an optimal Gateway-Edge configuration, an SDN controller needs to perform an exhaustive search for all possible gateway to edge combinations. Literally, it means that the size of the search space is equivalent to M^N, where M represents the number of edge servers and N represents the number of active gateways at a particular time t. The number of possible configurations increases exponentially with an increase in the number of M and N, respectively. To resolve the Gateway-Edge configuration as an optimization problem, we use the evolutionary algorithms, i.e., GA and DPSO. The following steps are executed for these algorithms to achieve an optimal configuration and a balanced load.

1. Generate a random population R^0 of size Δ. The best possible position for each particle, i.e., Gateway, is initiated such that $Pbest^0_i = r^0_i$, $\forall\ 1 \leq i \leq \Delta$.
2. Discover the fitness value of each particle for DPSO and each chromosome for GA in R^0 (using Eq. (5)) and identify its global best position $Gbest^0$, using Eq. (6).

   $$Gbest^0 = argmax_{1 \leq i \leq \Delta} F(Pbest^I_i). \tag{6}$$

3. For GA, if the best candidate solution for Gateway-Edge configuration is attained or the maximum number of generations has reached, then the search ends, otherwise, Step 4 is executed. For DPSO, if the best candidate solution is achieved, then the velocities of particles in the current population need to be updated using Eq. (7).

   $$v^I_i = j_w v^{I-1}_i + a_1 r_1 (Pbest^I_i - x^I_i) + a_2 r_2 (Gbest^I_i - x^I_i). \tag{7}$$

   Here, x^I_i represents the current position of particle i at I^{th} iteration, r_1 and r_2 are random variables within the (0, 1) range, a_1 and a_2 are acceleration constants used for pulling the particles toward the best position, and j_w reflects the inertia effect of preceding particle's velocity over the updated particle's velocity.
4. Next, a set of the best available γ chromosomes are extracted from the current population for GA. The current population is R^I and the selection probability is P_s. For DPSO, the iteration number is simply updated, i.e., I=I+1.
5. In case of GA, crossover and mutation are performed on γ. All infeasible solutions, i.e., R^I- γ are replaced with μ. Here, μ represents the newly generated chromosomes. In the case of DPSO, if the best candidate solution is attained for Gateway-Edge configuration, then the search ends; otherwise, Step 6 is executed.

6. For GA, all the steps from Step 2 are repeated. For DPSO, the personal best position for each particle is updated using Eq. (8).

$$Pbest_i^l = \begin{cases} Pbest_i^{l-1}, & \text{if } F(r_i^l) \leq F(Pbest_i^{l-1}) \\ r_i^l, & otherwise. \end{cases} \tag{8}$$

7. For DPSO, the global best position is updated using Eq. (9).

$$Gbest_i^l = \begin{cases} argmax_{1\leq i\leq \Delta} F(Pbest_i^l), & \text{if } F(Pbest_i^l) > F(Pbest_i^{l-1}) \\ Gbest_i^{l-1}, & otherwise. \end{cases} \tag{9}$$

8. For DPSO, repeat all the steps from Step 1.

The flowchart of our proposed approach is shown in Fig. 3. The SDN controller constantly performs the Gateway-Edge configuration based on the NP and KPI values. Since the controller needs to collect various information from the network to analyse the NP for the current Gateway-Edge configuration, we have highlighted the data fusion at the node level as well. The above solution can be used for homogeneous edge servers or that have capabilities to execute at approximate equal times. Furthermore, it does not account for dynamic scenarios where some data get processed earlier than the other edges. Therefore, to further balance the load, a service migration algorithm is suggested — which is feasible as, largely, edge servers run Linux-based operating systems.

3.3.1. Service migration technique

In our framework, load balancing can bring, at least, two benefits: performance improvement; and infrastructure energy efficiency. We trigger service migration either if: the utilization level of a particular edge node; and/or the data transfer rate on a particular link (channel condition), exceed certain predefined threshold values (steps 1 to 5) [36]. Once a module is being moved to another edge, it will immediately start receiving packets on another, perhaps, less utilized route. This could be achieved, after copying memory contents of the VM or container through sending a complimentary ARP (address resolution protocol) reply packet to inform the routing devices, within the network, to send data packets to its new location. As a result, both goals, i.e. balanced workload on various edges and reduced network traffic, could be achieved. Once a migration decision is triggered from a particular edge, next is to select a module of a suitable application to move. We move the application's module which can utilize the destination edge more i.e. priority is given to the module which is receiving more packets than other modules (step 6). Lastly, the module is migrated to the least utilized, neighbouring, edge platform; in order to diminish the migration performance impacts over the application's module and migration time (step 7). Finally, the selected module of the application m and the destination server are added to the migration map (step 8–10). The migration map is, then, passed to the load optimization module, as shown in Fig. 3, in order to reconfigure the Gateway-Edge configuration, periodically. The migration steps are described in Algorithm 1:

Algorithm 1: Service migration technique

```
Input: dynamic CPU and memory utilization threshold
       values i.e. $U_{tc}$ and $U_{tm}$, respectively — computed
       periodically using Eq. (12); channel condition $C_t$
Output: migration list $map$ → input for load optimization
        module in Fig. 3
1  compute CPU utilization level of the edge node ($EN_{uc}$);
2  compute memory utilization level of the edge node ($EN_{um}$);
3  compute channel condition ($C_c$);
4  for each node ∈ edge do
5      if $EN_{uc} \geq U_{tc}$ or $EN_{um} \geq U_{tm}$ or $C_c \geq C_t$ then
6          select application $m$ from edge node;
7          choose edge node $n$ as destination node;
8          $map \leftarrow m, n$;
9      end if
10 end for
11 return map
```

where EN_{uc} is the total provisioned CPU resources (cores) and EN_{um} is the total provisioned memory resources (RAM) with respect to their total capacities. Note that, U_{tc} and U_{tm} refer to the utilization level of a particular resource i.e. CPU, memory, respectively. Network resources such as bandwidth can also be considered in this formulation. Moreover, the channel condition B_{ij} is estimated using the transmission rate T_r, as given by:

$$T_r = B_{ij}.log_2(1 + \frac{P_{ij}.h_{ij}}{N}) \tag{10}$$

where B_{ij} represents the bandwidth between edge server i and gateway j, h_{ij} denotes the channel gain for gateway j at edge server i and P_{ij} is the transmission power of gateway j. Furthermore, N is the background noise [37]. Note that, Alg. 1 will approximately take $\mathcal{O}(mnlog(n))$ - where m denotes the total number of edges, n denotes the number of edge nodes and $log(n)$ is the time needed to compute configuration states such as resource utilization levels and channel conditions. The best case occurs at $\mathcal{O}(log(n))$ plus the time needed to complete all possible migrations. However, complexity would increase up to $\mathcal{O}(mn)^2$ for large number of edges, hosts and application requests — if unluckily an application cannot be placed or, in case, enough resources are not available. Note that, from security point of view, service migration in the IoT and VM or container migration in infrastructure clouds are completely different [36]. Usually, in infrastructure clouds, the migration data is transferred over dedicated networks; however, in IoT the data is transferred over the internet. This makes it essential to encrypt the migrated data and to authenticate the service migration messages that are exchanged among various edge devices.

Using static values for thresholds may not be feasible to trigger effective migrations in platforms with dynamic, heterogeneous and unpredictable workloads. This is due to the fact that resources that falls within the range of the least and most utilized (lower and upper thresholds) resources could not be reconfigured i.e. all hosts are equally loaded. In such scenario, threshold values can either be decreased or increased to balance the load amongst the edge nodes. Therefore, threshold values are needed to be adaptive and dynamically estimated using some sort of statistical techniques on historical data [23]. For example, we can adjust the threshold values based on the strength of the deviation of the edge or link utilization levels because higher deviations increase the likelihood of rising utilization levels. In other words, the higher the deviation, the lower the value of the threshold. Various methods such as local regression (LR), median absolute deviations (MAD), and entropy can be used to measure the statistical dispersion. For implementational simplification purposes, we prefer to use the MAD that describes the median of absolute values of deviations (residuals) from the data's median. For a particular dataset $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots, \mathcal{D}_n\}$, the MAD can be computed as:

$$MAD = median_i(abs[\mathcal{D}_i - median_j(\mathcal{D}_j)]) \tag{11}$$

The adaptive threshold value $\mathcal{T}_v$ is given by:

$$\mathcal{T}_v = 1 - \lambda.MAD \tag{12}$$

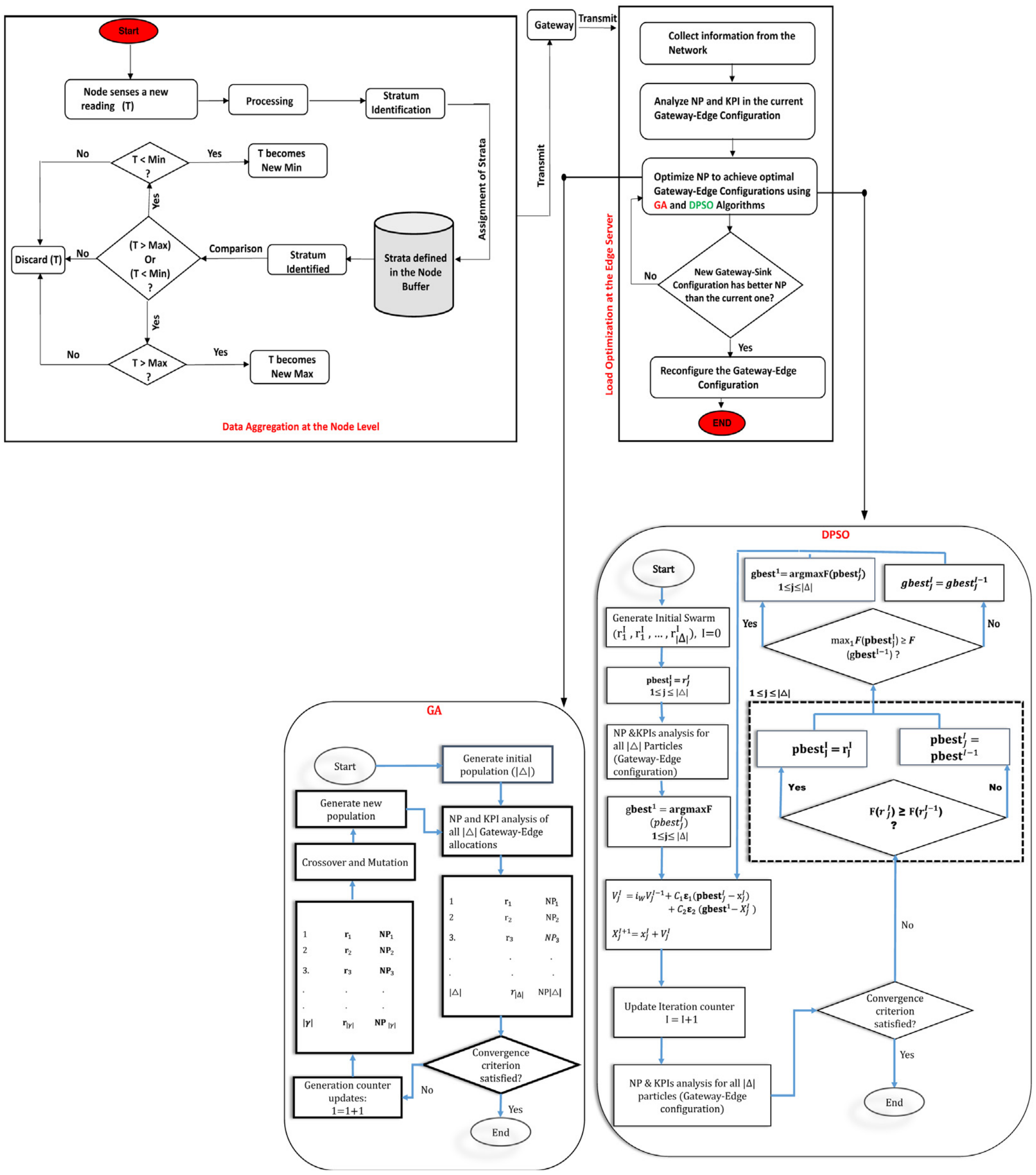

Start
Node senses a new reading (T)
Processing
Stratum Identification
Assignment of Strata
T < Min ?
Yes
No
T becomes New Min
(T > Max) Or (T < Min) ?
Comparison
Stratum Identified
Strata defined in the Node Buffer
Discard (T)
T > Max ?
T becomes New Max
Data Aggregation at the Node Level
Gateway
Transmit
Collect information from the Network
Analyze NP and KPI in the current Gateway-Edge Configuration
Optimize NP to achieve optimal Gateway-Edge Configurations using GA and DPSO Algorithms
New Gateway-Sink Configuration has better NP than the current one?
Reconfigure the Gateway-Edge Configuration
END
Load Optimization at the Edge Server
GA
Generate initial population (|△|)
Generate new population
NP and KPI analysis of all |△| Gateway-Edge allocations
Crossover and Mutation
Convergence criterion satisfied?
Generation counter updates: l=l+1
End
DPSO
Generate Initial Swarm
pbest = r, 1≤j≤|△|
NP &KPIs analysis for all |△| Particles (Gateway-Edge configuration)
gbest = argmaxF(pbest) 1≤j≤|△|
Update Iteration counter l=l+1
NP & KPIs analysis for all |△| particles (Gateway-Edge configuration)
Convergence criterion satisfied?
F(r) ≥ F(r) ?
max F(pbest) ≥ F(gbest) ?

Fig. 3. Flowchart of the proposed approach.

where λ is a parameter that describes how strongly the system tolerates edges over utilizations — lower λ results in higher tolerance to variations in utilization level. Once $\mathcal{T}_v$ is computed, the utilization levels of the node and link are compared to it in order to trigger appropriate migration decisions.

4. Experimental results

In this section, we evaluate the efficiency of our proposed data fusion and load optimization approach in terms of various experimental metrics. For data fusion, we developed a Java-based simulator that utilizes the data collected from sensors, a setup similar to the one adopted at Intel Berkeley Research Lab [38]. Upon fusion, the simulator feeds the refined data to the gateways. For optimal Gateway-Edge configuration, we use Matlab 2018a interfaced with Java. Moreover, we added several Java class files to mimic the notion of containers that simulate a containerized fog infrastructure. The classes were taken from the well-known fog simulator iFogSim [20]. The service migration technique uses either: (i) a static threshold value of 80%; or (ii) dynamic thresholds computed using Eq. (12) in order to trigger migrations of application modules across edges. We further assume that overload (i.e. upper threshold) will not happen due to service placement constraint. To carry out this, the iFogSim default policies for selecting over-loaded servers, containers and target servers were used.

In Fig. 4, the percentage of fused packets transmitted to the gateways is shown for different values of τ. Here, τ represents the number of readings sensed by each node over its sampling interval (S_r). The percentage of transmitted packets is calculated as $\frac{S_p}{\tau} \times 100$, where S_p denotes the number of fused packets sent from the strata of each node. The efficiency of our data fusion approach enhances with an increase in the value of τ. The percentage of transmitted fused packets from the strata of each node drops to 1% for 1000 packets, sensed during S_r. Our approach conserves the energy of resource-starving nodes and at the same time, reduces the burden on the network gateways. In comparison to our approach, the existing schemes deliver higher percentage of redundant data to the gateways. For example, EECC [34] transmits multiple copies of the same data from the strata of each node after S_r interval. As a result, the percentage of fused data delivered at the gateways is proportionally high. Moreover, without data

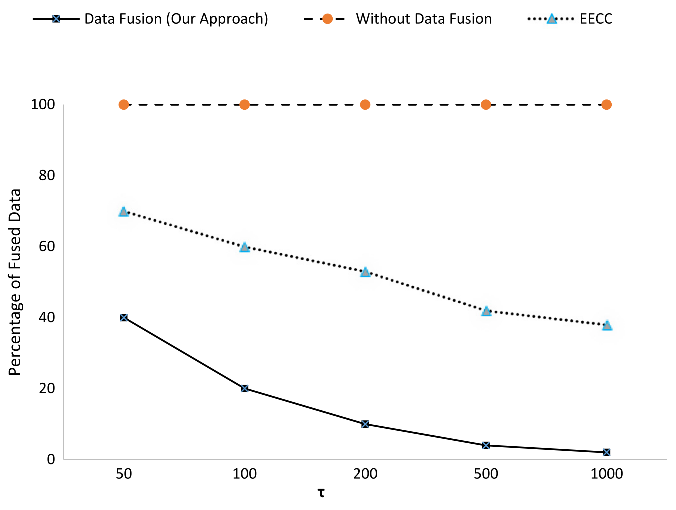


Fig. 4. Percentage of fused data.

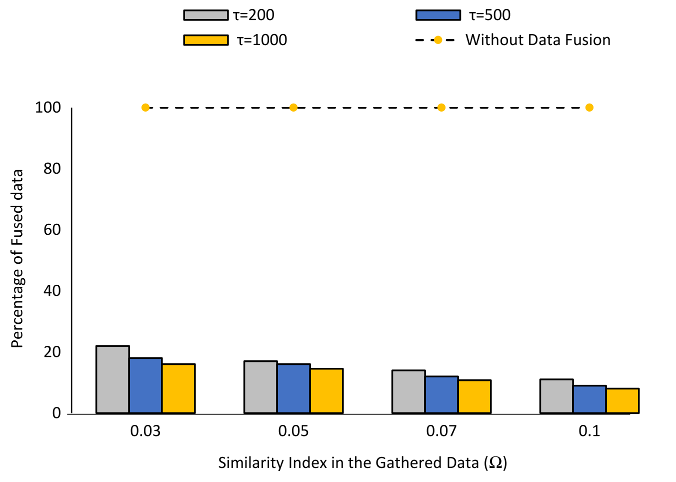


Fig. 5. Data fusion with varying values of similarity index.

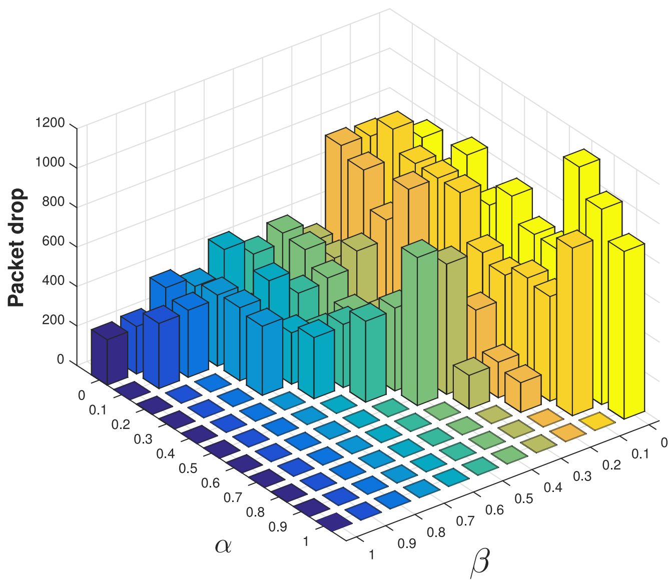


Fig. 6. Packet loss with varying values of α and β.

Table 1
Optimal gateway-edge configurations.

Number of edge servers	Convergence rate		Number of iterations	
	GA	DPSO	GA	DPSO
2	0.925	0.94	14	11
3	0.86	0.895	28	21
4	0.805	0.85	43	31

fusion, all the sensed packets need to be transmitted to the gateways that will adversely affect the decision-making at the data centres.

During data fusion, each node examines the similarity index (Ω) in the data gathered over the S_r interval. This index further reduces the redundancy and at the same time, lowers the processing burden on the nodes and the network gateways. In Fig. 5, the percentage of fused data for varying values of Ω is shown. In this figure, the values of τ varies from 200 to 1000 and Ω from 0.03 to 0.1, respectively. If Ω is 0.03, it means that among multiple readings having a similarity lower than or equal to 0.03, only one reading will be retained and the rest will be discarded. As a result, a higher percentage of readings will be discarded with an increase in the value of Ω. Moreover, our approach achieves a higher percentage of fusion when the value of τ increases. This figure shows that with higher values of Ω and τ, the processing and transmission burdens on the edge nodes and gateways decreases, significantly. In the absence of data fusion technique, a higher percentage of data is delivered to the gateways that in turn increases the processing and transmission burden on the nodes.

The optimal Gateway-Edge configurations achieved by GA and DPSO are shown in Table 1. We considered three benchmark problems P_1, P_2, and P_3 with two, three, and four edge servers, respectively. P_1, P_2, and P_3 contain 1200 sensor nodes including 200 gateways, distributed over the sensing field. In P_1, GA converges to an optimal Gateway-Edge configuration after the 14th generation. DPSO, on the other hand, achieves the optimal configuration after the 11th iteration. Please note that iteration and generation are similar terms. The former is used in PSO and the latter in GA, as discussed in Section 3.3. GA achieves 185 optimal Gateway-Edge configurations over 200 generations with a convergence rate of 0.925, whereas, in DPSO, there are 188 optimal configurations with a convergence rate of 0.94. In P_2 and P_3, the convergence rate of GA and DPSO decreases and larger values of iterations and generations are required to achieve an optimal Gateway-Edge configuration. It is mainly due to an increasing number of edge servers in these benchmark problems. These results show that DPSO reaches an optimal solution in fewer iterations as compared to GA.

We assessed the network performance (NP) for all optimal solutions in term of packet drop by modifying the KPIs such that α and β are set in the ordered form of 0,0.1, …,1 with a constraint $\alpha + \beta <= 1$. To properly tune α and β, an exhaustive search is performed on P_1 using these parameters to achieve an optimal Gateway-Edge configuration. When α and β are set to 0.8 and 0.2, respectively; a minimum packet drop is observed, as shown in Fig. 6. The selection of proper weights for the optimization function, i.e., NP, is challenging and essential for achieving optimal results in the context of evolutionary algorithms.

The service migration technique, as suggested in Section 3.3.1, was implemented to balance the load across the edge servers. Increasing the total number of edge servers decreases the utilization levels and vice versa, as shown in Table 2. Moreover, the number of migrations happened is proportional to the amount of fog servers. The standard deviation actually represents how the current load on each server differs from other servers — the higher this value, the more less balanced is the workload and vice versa. We observed significant reduction in utilization levels,

Table 2
Average utilization levels (%) of the edge servers and number of migrations (using static and dynamic threshold values for load balancing and resource re-configuration).

Number of servers	GA		DPSO	
	Utilization	Migrations	Utilization	Migrations
Static threshold values				
2	62.9 ± 5.99	101	61.4 ± 6.45	89
3	60.4 ± 6.31	105	58.7 ± 8.01	103
4	59.6 ± 6.44	147	59.1 ± 7.32	122
Dynamic threshold values				
2	71.2 ± 1.27	134	75.3 ± 2.02	155
3	62.8 ± 1.71	178	68.4 ± 3.89	189
4	54.3 ± 1.09	201	59.7 ± 2.99	217

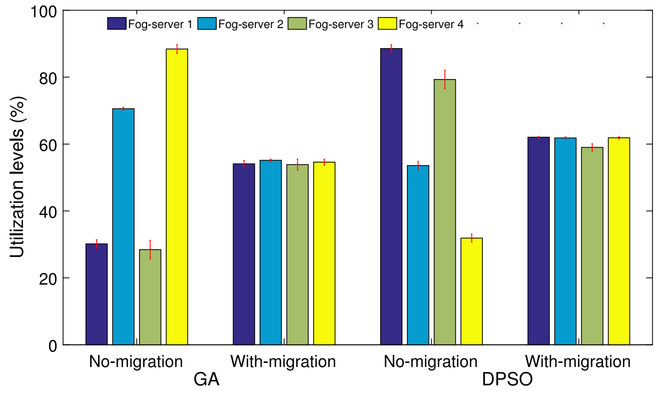


Fig. 7. Load balancing for four fog servers [error bars denote standard deviations from the means].

that, essentially translate to greater energy savings and improved levels of performance. We observed, as shown in Table 2, that using static threshold values reduces the migration opportunities. Moreover, increased levels of variations were observed in server's utilization levels. This means that either resources were utilized more or the least due to less migration opportunities. Using dynamic threshold values, the variations in utilization levels may decrease significantly — as more migrations will occur subsequently. Varying the threshold values will essentially result in variations of outcomes and differences in Gateway-Edge reconfiguration. Fig. 7 shows average utilization levels (along with error bars at five minute intervals) when four fog servers are taken into account. Moreover, number of migrations would have some impacts on network traffic and processing performance. For four fog servers when migrations are not taken into account, we achieved 48.75 ± 23.67 and 41.28 ± 28.56 average utilization levels for GA and DPSO, respectively. The high variations (standard deviations) show the imbalance load across various servers which might happened due to dynamics in workloads execution or processing patterns. With migrations, we were able to significantly reduce these variations as shown in Fig. 7.

5. Conclusions and future work

In this paper, we proposed a lightweight data fusion and AI-enabled load optimization approach for reconfigurable IoT applications. The buffer of each node is partitioned into strata that hold and transmit only non-correlated fused data towards the network gateways and edge servers. We used GA and DPSO to optimize the usage of available resources by identifying the optimal routes for upstream transmission of refined data from the gateways to edge servers. These algorithms monitored the load at the servers, and if an unbalanced load is experienced, the current Gateway-Edge configuration is reconfigured. For load monitoring at the edge, various Key Performance Indicators (KPIs) were used. Our experimental results significantly reduced the processing and transmission burden at the nodes for large-sized data streams. Our approach achieved optimal gateway-edge configurations for varying number of edge servers in a densely populated network setup. Moreover, a migration approach was used to balance the load across different edge servers. Our evaluation of the proposed migration approach demonstrated that all edge servers are relatively utilized uniformly while having lower standard deviations in their utilization levels. Subsequently, this ensures that data is processed at edge which increases performance. In the future, we aim to analyse the network performance by maintaining a balanced load at the network gateways. It will enable the gateways to automate the downstream transmission links towards the nodes. Moreover, we are keen to see the impact of migrations in dynamic scenarios, particularly, on network traffic and transmission delays.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was partially supported by a pilot award from the Center for Research in Human Movement Variability, USA and the NIH, USA (P20GM109090) and the National Science Foundation, USA under award CNS-2016714.

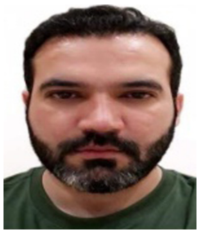
References

[1] Y. Shen, T. Zhang, Y. Wang, H. Wang, X. Jiang, Microthings: A generic IoT Architecture for flexible data aggregation and scalable service cooperation, IEEE Commun. Mag. 55 (9) (2017) 86–93.
[2] F. Khan, M.A. Jan, A.U. Rehman, S. Mastorakis, M. Alazab, P. Watters, A secured and intelligent communication scheme for iIoT-enabled pervasive edge computing, IEEE Trans. Ind. Inf. (2020).
[3] W. Ding, X. Jing, Z. Yan, L.T. Yang, A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion, Inf. Fusion 51 (2019) 129–144.
[4] F. Alam, R. Mehmood, I. Katib, N.N. Albogami, A. Albeshri, Data fusion and IoT for smart ubiquitous environments: a survey, IEEE Access 5 (2017) 9533–9554.
[5] W. Twayej, M. Khan, H.S. Al-Raweshidy, Network performance evaluation of M2M with self organizing cluster head to sink mapping, IEEE Sens. J. 17 (15) (2017) 4962–4974.
[6] M.A. Jan, F. Khan, R. Khan, S. Mastorakis, V.G. Menon, P. Watters, M. Alazab, A lightweight mutual authentication and privacy-preservation scheme for intelligent wearable devices in industrial-CPS, IEEE Trans. Ind. Inf. (2020).
[7] D. Mazza, D. Tarchi, G.E. Corazza, A unified urban mobile cloud computing offloading mechanism for smart cities, IEEE Commun. Mag. 55 (3) (2017) 30–37.
[8] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, C.-T. Lin, Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment, IEEE Access 6 (2017) 1706–1717.
[9] S.K. Sharma, X. Wang, Live data analytics with collaborative edge and cloud processing in wireless IoT networks, IEEE Access 5 (2017) 4621–4635.
[10] H. Yao, M. Li, J. Du, P. Zhang, C. Jiang, Z. Han, Artificial intelligence for information-centric networks, IEEE Commun. Mag. 57 (6) (2019) 47–53.
[11] T.A. Al-Janabi, H.S. Al-Raweshidy, A centralized routing protocol with a scheduled mobile sink-based AI for large scale I-Iot, IEEE Sens. J. 18 (24) (2018) 10248–10261.
[12] F.H. Bijarbooneh, W. Du, E.C.-H. Ngai, X. Fu, J. Liu, Cloud-assisted data fusion and sensor selection for internet of things, IEEE Internet Things J. 3 (3) (2016) 257–268.

[13] N. Nesa, I. Banerjee, Sensorrank: An energy efficient sensor activation algorithm for sensor data fusion in wireless networks, IEEE Internet Things J. (2018).
[14] S. Kumar, V.K. Chaurasiya, A strategy for elimination of data redundancy in internet of things (IoT) based wireless sensor network (WSN), IEEE Syst. J. (2018).
[15] S. Sanyal, P. Zhang, Improving quality of data: IoT data aggregation using device to device communications, IEEE Access 6 (2018) 67830–67840.
[16] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, Vol. 12, MIT press, 1999.
[17] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (6) (2003) 317–325.
[18] M. Abbasi, M. Rafiee, M.R. Khosravi, A. Jolfaei, V.G. Menon, J.M. Koushyar, An efficient parallel genetic algorithm solution for vehicle routing problem in cloud implementation of the intelligent transportation systems, J. Cloud Comput. 9 (1) (2020) 6.
[19] K. Kaur, S. Garg, G. Kaddoum, S.H. Ahmed, M. Atiquzzaman, KEIDS: Kubernetes-based energy and interference driven scheduler for industrial IoT in edge-cloud ecosystem, IEEE Internet Things J. 7 (5) (2020) 4228–4237.
[20] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.
[21] L.F. Bittencourt, J. Diaz-Montes, R. Buyya, O.F. Rana, M. Parashar, Mobility-aware application scheduling in fog computing, IEEE Cloud Comput. 4 (2) (2017) 26–35.
[22] A. Machen, S. Wang, K.K. Leung, B.J. Ko, T. Salonidis, Live service migration in mobile edge clouds, IEEE Wirel. Commun. 25 (1) (2018) 140–147.
[23] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1366–1379, http://dx.doi.org/10.1109/TPDS.2012.240.
[24] A.A. Khan, M. Zakarya, R. Khan, H^2 – a hybrid heterogeneity aware resource orchestrator for cloud platforms, IEEE Syst. J. 13 (4) (2019) 3873–3876.
[25] M. Zakarya, L. Gillam, Managing energy, performance and cost in large scale heterogeneous datacenters using migrations, Future Gener. Comput. Syst. 93 (2019) 529–547.
[26] M. Zakarya, L. Gillam, H. Ali, I. Rahman, K. Salah, R. Khan, O. Rana, R. Buyya, EpcAware: A game-based, energy, performance and cost efficient resource management technique for multi-access edge computing, IEEE Trans. Serv. Comput. (2020) 1–14, http://dx.doi.org/10.1109/TSC.2020.3005347.
[27] A.A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, O. Rana, An energy and performance aware consolidation technique for containerized datacenters, IEEE Trans. Cloud Comput. (2019) http://dx.doi.org/10.1109/TCC.2019.2920914.
[28] M. Zakarya, L. Gillam, A.A. Khan, I.U. Rahman, Perficientcloudsim: a tool to simulate large-scale computation in heterogeneous clouds, J. Supercomput. (2020) 1–55, http://dx.doi.org/10.1007/s11227-020-03425-5.
[29] S. Mastorakis, A. Mtibaa, J. Lee, S. Misra, Icedge: When edge computing meets information-centric networking, IEEE Internet Things J. 7 (5) (2020) 4203–4217.
[30] B. Nour, S. Mastorakis, A. Mtibaa, Compute-less networking: Perspectives, challenges, and opportunities, IEEE Netw. 34 (6) (2020) 259–265.
[31] S. Mastorakis, A. Mtibaa, Towards service discovery and invocation in data-centric edge networks, in: 2019 IEEE 27th International Conference on Network Protocols (ICNP), IEEE, 2019, pp. 1–6.
[32] Q. Wang, D. Lin, P. Yang, Z. Zhang, An energy-efficient compressive sensing-based clustering routing protocol for WSNs, IEEE Sens. J. 19 (10) (2019) 3950–3960.
[33] T. Li, M. Bolic, P.M. Djuric, Resampling methods for particle filtering: classification, implementation, and strategies, IEEE Signal Process. Mag. 32 (3) (2015) 70–86.
[34] S.R.U. Jan, M.A. Jan, R. Khan, H. Ullah, M. Alam, M. Usman, An energy-efficient and congestion control data-driven approach for cluster-based sensor network, Mob. Netw. Appl. 24 (4) (2019) 1295–1305.
[35] R. Jain, A. Durresi, G. Babic, Throughput fairness index: An explanation, in: ATM Forum Contribution, Vol. 99, 1999.
[36] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Commun. Surv. Tutor. (2017).
[37] H. Zhang, F. Guo, H. Ji, C. Zhu, Combinational auction-based service provider selection in mobile edge computing networks, IEEE Access 5 (2017) 13455–13464.
[38] S. Madden, Intel berkeley research lab data, 2003.

Mian Ahmad Jan is an Assistant Professor at the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. He received his Ph.D. in Computer Systems from the Faculty of Engineering and Information Technology (FEIT), University of Technology Sydney (UTS), Australia. His research interests include energy-efficient and secured communication in Wireless Sensor Networks and Internet of Things. He has been guest editor of numerous special issues in various prestigious journals e.g. Future Generation Computer Systems, IEEE Transactions on Industrial Informatics, Springer Neural Networks and Applications, IEEE Sensor, etc. are few to mention. He is an IEEE senior member.

Muhammad Zakarya received the Ph.D. degree in Computer Science from the University of Surrey, Guildford, U.K. He is currently a Lecturer with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interests include cloud computing, mobile edge clouds, Internet of Things (IoT), performance, energy efficiency, algorithms, and resource management. He has deep understanding of the theoretical computer science and data analysis. Furthermore, he also owns deep understanding of various statistical techniques which are, largely, used in applied research. He is an Associate Editor for the IEEE Access Journal and has served as TPC member in various international conferences and workshops.


Muhammad Khan received his Ph.D. degree in wireless communications from Brunel University London, United Kingdom. He is associated with the Wireless Network and Communication Centre (WNCC) at Brunel University London, and the Computer Networks (ComNets) Lab at New York University (NYU). He is currently working on Projects related to 5G mmWave Congestion Control in collaboration with the University of Contabria, Spain and University of Limerick, Ireland. His main research interest is next generation wireless communications, Cloud Radio Access networks, Artificial Intelligence, Machine learning and Congestion Control.

Spyridon Mastorakis (smastorakis@unomaha.edu) is an Assistant Professor in Computer Science at the University of Nebraska Omaha. He received his Ph.D. in Computer Science from the University of California, Los Angeles (UCLA) in 2019. He also received an M.S. in Computer Science from UCLA in 2017 and a 5-year diploma (equivalent to M.Eng.) in Electrical and Computer Engineering from the National Technical University of Athens (NTUA) in 2014. His research interests include network systems and protocols, Internet architectures (such as Information-Centric Networking and Named-Data Networking), edge computing and IoT, and security.

Varun G. Menon is currently an Associate Professor in Department of Computer Science and Engineering, SCMS School of Engineering and Technology, India. He is a Senior Member of IEEE and a Distinguished Speaker of ACM Distinguished Speaker. Dr. Varun G Menon is currently a Guest Editor for IEEE Transactions on Industrial Informatics, IEEE Sensors Journal, IEEE Internet of Things Magazine and Journal of Supercomputing. He is an Associate Editor of IET Quantum Communications and also an Editorial Board Member of IEEE Future Directions: Technology Policy and Ethics. His research interests include Internet of Things, Fog Computing and Networking, Underwater Acoustic Sensor Networks, Cyber Psychology, Hijacked Journals, Ad-Hoc Networks, Wireless Sensor Networks.

Venki Balasubramanian received the Ph.D. degree in body area wireless sensor network (BAWSN) for remote healthcare monitoring applications. He is the Pioneer in building (pilot) remote healthcare monitoring application (rHMA) for pregnant women for the New South Wales Healthcare Department. His research establishes a dependability measure to evaluate rHMA that uses BAWSN. His research opens up a new research area in measuring time-critical applications. He contributed immensely to eResearch software research and development that uses cloud-based infrastructure and a core member for the project sponsored by Nectar Australian research cloud provider. He contributed heavily in the field of healthcare informatics, sensor networks, and cloud computing. He also founded Anidra Tech Ventures Pty Ltd a smart remote patient monitoring company.

Ateeq Ur Rehman is currently working as a Lecturer at the department of Computer Science, Abdul Wali Khan University Mardan, KPK, Pakistan. He received his Ph.D. degree from the University of Southampton in 2017. His main research interests are next-generation wireless communications and cognitive radio networks, Internet of Things, Internet of Vehicles, blockchain technology and differential privacy.