A Context-Aware Fog Enabled Scheme for Real-Time Cross-Vertical IoT Applications

Diptendu Sinha Roy, *Member, IEEE*, Ranjit Kumar Behera, K. Hemant Kumar Reddy, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—As the Internet of Things (IoT) paradigm is maturing, innovative, and novel services are being envisioned. An upcoming trend is the depiction of services enacted through seamless integration of multiple vertical IoT services, termed as cross-vertical or unified IoT services in this paper. Traditional Cloud-based centralized network architectures cannot cater to real-time responses demanded by such unified IoT applications. Moreover, introducing Fog nodes within the network architecture, though a promising alternative, cannot sustain the burden of a huge number of applications that culminates in massive data handling. In this paper, we envision employing lessons learned from context-aware computing, specifically context sharing among interdependent vertical IoT applications to address this delay requirement of such unified IoT applications by enacting context sharing among Fog nodes for minimizing system delay. The detailed network model and context sharing mechanism have been presented and the service time minimization has been framed as an optimization problem. Algorithms for context sharing and delay tolerant load balancing have been presented and simulation results carried out demonstrate the efficacy of the proposed methodology.**

***Index Terms*—Cloud computing, context sharing, cross-vertical Internet of Things (IoT) applications, delay tolerant load balancer, fog computing, IoT, service delay.**

I. Introduction

The Internet of Things (IoT) is an emerging conglomeration of information and communications technology and related technologies, where billions of devices and things are interconnected toward enabling a variety of intelligent applications, such as, smart city and homes, transportation, logistics, healthcare, industries, and so on [1], [2]. However, the next achievement in the landscape of IoT is the remarkable transformation of such applications to cross-vertical applications and services [3]–[6]. Cross-vertical IoT applications are also term as cross-domain IoT applications. Cross-vertical applications are the integration of various solo IoT applications domains. For instance, collaboration of smart grid and intelligent home can unlock new business models for IoT and transform IoT business processes. In a smart city for instance, a full service like traffic management is achieved by combining separate vertical services, like parking monitoring, road status, traffic guidance, and so forth. Similarly, the potential for energy savings and optimization across a city can be realized by aggregating and analyzing data across various buildings, heating ventilation and air conditioning, car charging stations, and other lighting devices. Thus, the design of the cross-vertical applications is needed to materialize such holistic smart IoT systems.

A significant part in the realization of cross-vertical applications is interoperability between their discrete communication protocols and network infrastructures [7] to share data and their context. However, interoperability between different IoT applications can be implemented at various levels of the protocol stack and at diverse places in an end-to-end solution. For instance, at higher level of network, mainly semantic ontologies can be implemented to transfer the requested data across different applications and domains [8]. An alternative approach is to implement data transfers at the network level, at the point of data aggregation itself, like at the IoT gateways or in the Cloud [5]. However, Soursos *et al.* [5] have focused on interoperability at lower levels of network where data are gathered. IoT devices transfer raw data to the Cloud for their analyses and storage. Thus, the Cloud is a collection point of IoT data, irrespective of their communication protocols. Cloud can derive meaningful insights from data as well as from different context instances using context-aware computing [9]. Additionally, the Cloud stores insights and context instances for realization of cross-domain applications. For instance, Feel@Home [8] is a project that employs context management framework for enabling cross domain IoT applications by sharing context instances. Moreover, upon queries from other applications in any domain, an application of other domains can share its context instances. Context means the information that can characterize different situations of an entity with respect to their dependable entity. An entity can be a person, place, or things that have relevant interaction and dependency with other entity of different applications. This definition is widely used by researchers to ascertain context from IoT data. Each learned context or information can be termed as a context instance. Context instances can be produced by processing of raw data. Interoperability issues are not relevant here because data are at the same place and are

Manuscript received May 26, 2018; revised July 7, 2018 and July 28, 2018; accepted August 28, 2018. Date of publication September 10, 2018; date of current version May 8, 2019. *(Corresponding author: Diptendu Sinha Roy.)*

D. Sinha Roy is with the Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong 793003, India (e-mail: diptendu.sr@nitm.ac.in).

R. K. Behera and K. H. K. Reddy are with the School of Computer Science and Engineering, National Institute of Science and Technology, Berhampur 761008, India (e-mail: ranjit.behera@gmail.com; khemant.reddy@gmail.com).

R. Buyya is with the School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/JIOT.2018.2869323

already converted into a uniform format, which is needed to be shared. However, sharing of insights or context information should be achieved in real-time to meet the quality of service (QoS) of cross-domain applications. Thus, Cloud computing is not suitable for providing such services in real-time for massive IoT data [11]. As a consequence, we focus on Fog computing infrastructure for cross-vertical applications which is expected as a next-generation technology to analyze IoT data locally and provide real-time services efficiently.

For providing cross-domain services, Fog nodes can act as an interface between discrete IoT applications where all data are gathered and are orchestrated between different applications. Also, since Fog computing is a distributed network infrastructure, the orchestrated system will expectedly be more reliable and efficient in terms of QoS. Thus, Fog computing offers a promising solution owing to its low latency for realizing interoperability between different applications in real-time and at the very point of data accumulation, irrespective of any communication protocols and format. The applications at each Fog node can share the context instances among different Fog nodes and applications upon request by other applications. However, IoT devices delivering extensive data to Fog nodes can increase the overhead at the Fog nodes, since Fog nodes are limited by their computational capabilities. Therefore, an efficient mechanism is necessary to control the massive data flow and provide cross-domain applications. However, past researchers have not addressed realization of cross-domain IoT applications using Fog computing. Therefore, we present the realization of cross-domain applications using Fog computing in this paper. To the best of author's knowledge, this is a first of its kind work considering the implementation of the cross-domain applications using Fog computing and context-aware computing.

In this proposal, the Fog nodes receives data from any applications as different contexts and sent to Cloud for historical analyses and storage. Context generation handles massive data challenges at Fog nodes by keeping only context instances. Second, context instances can be of immense help to realize cross-domain services. The point in realizing cross-domain services is to share their different context information periodically from sensors already set up for different applications. Therefore, our proposed approach provides an efficient methodology to achieve cross-domain services by context sharing using Fog computing irrespective of communication protocols. Additionally, we optimize the delay of context sharing time among Fog nodes in this paper. Therefore, our proposed method is a first-of-its-kind attempt to design efficient cross-vertical IoT applications or unified IoT services and minimize the delay for the operating time in the system.

The remainder of this paper is organized as follows. Section II presents the related work with intuitive reasoning of our proposal. The definition of cross-domain, unified services and the role of context sharing are introduced in Section III along with relevant examples and a generalized service model for context sharing with Fog computing is also presented. Section IV describes our proposed method for efficient sharing of context instances between different applications and among Fog nodes. Section V discusses the experiments carried out with analyses of results attained thereof for the minimization of operating time using context sharing among Fog nodes. Finally, the concluding remarks are provided in Section VI.

II. Related Work

This section provides a brief outline of the relevant technological developments in areas related to this paper. All such background information have been organized in the subsequent three sections.

A. *Context-Aware Cross-Domain IoT Applications*

IoT is a paradigm where things like, any objects or devices, besides the users, can also be connected to the Internet, and they can communicate with each other seamlessly using any communication platform. The vision of IoT is interdisciplinary in nature and hence while defining IoT, authors have given utmost emphasis to realization of cross domain interactions of IoT applications. Cross-domain IoT applications is an essential concept for realization of IoT. Besides, it is one of the foremost differences between wireless sensor networks and the IoT. Wireless sensor networks generally deal only with a fixed domain. However, IoT requires the capability to work with multiple domains. Interoperability is the main issue in realizing cross domain IoT applications. Interoperability issue can be overcome by sharing of context in Cloud computing [5]. Context-aware computing has played a major role over the last decade in ubiquitous computing domain and is expected to play a significant role for IoT paradigm as well [9]. Perera *et al.* [9] have presented a large majority of research and commercial solutions proposed in the field of context-aware computing conducted over the last decade by considering 50 different projects. Most of these projects are modeled with context-aware middleware containing different modules for context management, and these modules follow a general rule which is the context lifecycle. Li *et al.* [12] presented a general context lifecycle (which defines the time period from its obtainment to destruction) that is demarcated by six major events, including context acquisition, context modeling, context reasoning, context distribution, context repository, and context visualization. Kamienski *et al.* [13] developed a project, namely IMPReSS for context-aware energy efficiency management for smart buildings. Very recently Fallahzadeh *et al.* [14] proposed a context-aware system design for remote health monitoring.

All such solutions have employed middleware to enable context sharing for realization of cross domain IoT applications. However, such solutions are applicable for a limited number of applications and not for holistic systems such as IoT. Therefore, there is need of context sharing at lower level of network where data are gathered. So, any applications from any domain can query for context instances. Such holistic systems can be enabled by emerging Cloud computing.

However, another challenge is the number of IoT devices. As per a data given from CISCO, by 2020, 50 billion devices will be connected to the Internet [15], which in turn will generate humongous amount of data. To store and process those data, to provide services and to cater to the needs of people in real life, industrialists, entrepreneurs, and individuals primarily rely on the advancement of Cloud computing paradigm. Some researchers have already proposed models for integrating Cloud and IoT to address computation and storage related deficiencies [16]. But for latency sensitive IoT applications, use of Cloud becomes a problem [17]. Fog computing is promising for the realization of cross-domain IoT applications [18]. Therefore, this paper proposes a model for cross-domain IoT applications by using context sharing and Fog computing. However, Fog computing suffers from many inherent challenges, such as limited network availability, network bandwidth, storage capacity, etc. The work related to the limitation of Fog computing is provided in the following section.

B. Fog Computing

Fog computing, a revolutionary concept, acts as an intermediate layer between end devices and traditional Cloud computing offers highly virtualized platforms [19], [20]. Fog was introduced as an infrastructure for distributed computing in order to handle millions of Internet-connected devices and was proposed using principles of edge computing, where services are hosted within edge devices, such as switches, routers, and access points. Chang *et al.* [17] proposed a hierarchical distributed architecture that extends from-the-edge-of-the-network to the core and illustrated a few real life applications of Fog computing with distinct role of Fog for IoT and with insights of Fog–Cloud interactions. Sanaei *et al.* [21] presented a mobile Fog computing-based programming model that supports large-scale IoT applications for provisioning of latency-sensitive and geographically dispersed applications. Similarly, Bonomi *et al.* [19] highlighted the importance of Fog computing and its role for IoT and analytics. Recently, some research has assessed the use of Fog computing in different domains and its importance [22]–[24]. Lv *et al.* [22] presented a detailed report on most promising and challenging scenarios in IoT, particularly focusing on three interrelated obligations, namely, mobility, reliability, and scalability for IoT scenarios. In [23], various different approaches for data-to-decision with Fog computing paradigm at a superficial level. Tang *et al.* [24] investigated different computing platforms for the feasibility of building a highly reliable and fault-tolerant Fog computing platform. Recently, researchers have focused on resource allocation in Fog computing platforms and sufficient amount of work has also been presented in [25]–[27]. A few researchers have also explored security and privacy aspects of Fog computing [28]–[30]. Yousefpour *et al.* [31], [32] have proposed models for minimization of service delay. However, they provide only a general framework for IoT applications through Cloud and Fog collaboration by means of offloding tasks from Fog to Cloud had been provided. Sharing of context instances among the Fog nodes was not dealt with in those studies. Most of the works proposed on Fog computing have primarily focused on the principles, basic notions, and the doctrines of it and not much research has contributed to more intricate aspects, like service delay of the Fog paradigm for context sharing among different domain or applications in real time. Therefore, this paper focuses on the service delay problem in Fog computing from the view point of context sharing for the realization of cross-domain IoT applications.

III. Cross-Domain IoT Application and Role of Context Sharing

In this section, we present those aspects considered crucial for the sustainable evolution of IoT landscape toward cross-domain interoperability, i.e., unified services of IoT. Later, we shall present some application scenarios of unified IoT services. Additionally, this section discusses the role of context sharing to realize unified IoT services.

A. Convergence of Vertical IoT Solutions

The current landscape of IoT is evolving around a plethora of vertical solutions, each separately suited to a specific scenario and development of specific protocol as required. However, recent studies show the need for cross-domain or cross-vertical IoT applications that can cover broader aspects of our daily lives and provide more innovative services. Cross-domain or cross-vertical applications are seamless integration of discrete vertical IoT solutions, such as smart homes, intelligent transport systems, smart grid, and so forth. We use the term unified IoT services in this paper to convey such services that are enacted through seamless integration of many vertical IoT services. For example, the integration of smart grid and intelligent transportation with devices in a smart home can provide efficient energy management services. An innovative unified IoT service can be developed such that a smart home can estimate the arrival time of a user from office, thus adjust the room temperature based on current weather conditions at the user's home upon his arrival and thus save energy for the rest of the time. This service seamlessly integrates smart home data, traffic data, weather data, and smart grid data. Similarly, there can be other innovative examples of unified services like smart mobility and urban ecological routing [3]–[6] and so on.

The development of unified IoT services highlight the need for interoperability between communication and network infrastructure of different IoT applications for efficient sharing of their sensing and actuation resources and data. The first challenge in the realization of unified services is interoperability among discrete communication protocols of vertical IoT applications. A standard communication protocol for the entire IoT system is not feasible simply because of different characteristics of sensing environment and their respective data formats. However, the accomplishment of cross-vertical applications can occur at network level like data aggregation point of IoT, such as the Cloud, which hosts all IoT applications. Devices transmit data to their respective IoT application at the Cloud. The applications analyze the data and respond back with appropriate control signals for actuation. However, Cloud stores data from different IoT applications. So, it can be possible that an application requests data from other IoT applications in Cloud and Cloud network shares much of the requested data to the application from storage. Further, the application analyzes all the received data and can provide innovative unified IoT services. So, the Cloud can solve interoperability issues by sharing data among different applications. However, sharing all the raw data among IoT applications can be a burgeoning overhead in terms of communication and computation required. Moreover, IoT applications need the context and reasoning of IoT data for the realization of cross-domain applications [3]. Therefore, IoT applications can share the context of data among themselves. Recent works [8], [9], [11] have discussed context-aware computing in IoT which helps to characterize IoT data into different events according to instantaneous environment conditions and can help us to understand the raw sensor data through reasoning. It also helps in making a scalable and automatic decision without human intervention. Thus, context-aware computing generates different context instances. Context instances are the detected events of an IoT application which occurs at specific locations and at certain time instants. So, this paper has considered the sharing of context instances in spite of raw data among the IoT applications. To the best of the author's knowledge, this paper is first to use context instances sharing for the realization of cross-domain IoT applications.

Moreover, the convergence and analysis of IoT data and context instances for cross-domain applications need to be processed in real-time to obtain quality and reliable services. The massive amount of IoT data at analytics network, i.e., at the Cloud can increase service delay [33]. Thus, the Cloud cannot provide real-time services and can degrade the QoS. Therefore, this paper has considered Fog computing as a convergence point for cross-domain applications. The Fog computing is a distributed computing paradigm, unlike the centralized Cloud computing, which makes it fault tolerant. Moreover, Fog computing holds promise of analyzing data locally and delivery services in real-time. Hence, the emergence of Fog computing paves the path for cross-domain IoT applications.

However, IoT devices may still deliver an enormous amount of data to Fog nodes but from a confined locality only. The analysis of massive amount of incoming data at a Fog node can also lead to high latency. Additionally, moving an enormous amount of IoT data between different vertical applications can be another challenge for realizing the cross-domain applications. Therefore, the next section discusses the network model of cross-domain applications using context sharing and Fog computing. The following sections explain research challenges in implementation of context sharing at Fog nodes.

B. Context Sharing

Schilit *et al.* [34] had coined the term context-awareness in 1994, in connection with ubiquitous computing. Recently a few researchers [35], [36] have considered context-aware services for IoT and it is expected to play a significant role in IoT as well. However, the definition of context is in itself a research question. It has been defined in several ways but we in this paper, define it for the purpose of this paper. This paper has used the most comprehensive definition of context as provided by Abowd *et al.* [37] and Dey *et al.* [38]. Context is information that can characterize the different situations of an entity with respect to their dependable entity. An entity can be a person, place or things that have relevant interaction and dependency with other entity of different applications. This definition is widely used by researchers to ascertain context from IoT data. Each learned context or information can be termed as a context instance. Context instances can be produced by processing of raw data. Examples of context instances include things state, profile, climate, taken actions, etc. However, generation of context information is a well-addressed research problem; therefore, in this paper, we have not included context instance generation in details. The context instances are used for providing unified IoT services by sharing relevant data with services from other domains. Applications provide services to actuators after analyzing different context instances from their dependent applications. In this paper, Fog computing and context instance sharing are applied for realizing unified IoT services. Before, discussing our efficient method for realization of unified IoT services, the assumed network model for unified services is introduced in the following section.

IV. Assumed Network Model and Problem Statement

In this section, we introduce the network configuration of unified IoT services. In this model, Fog computing and context instance sharing is employed for realizing efficient unified IoT services with QoS. In addition, a few examples are discussed to demonstrate the functioning of the given model and also to highlight a few challenges for context-aware Fog computing.

A. Network Model for Unified IoT Services

We focus on efficient context sharing among Fog nodes, in order to realize the cross-domain or unified IoT services. The assumed network model comprises of numerous sensors attached to various things and a few Fog nodes that collect data from these sensors. Additionally, Fog nodes are also interconnected to each other through a cellular network. Hence, Fog computing can also be termed as mobile edge computing in this paper where such nodes can also be considered as base stations. However, this paper has used the term Fog node throughout this paper to mean the same. The system architecture is shown in Fig. 1. Fog nodes collect data simultaneously from these devices within their locality using their respective communication protocols. Data are delivered to their respective applications deployed at Fog nodes. Each application processes these raw data through several levels in order to produce a high-level description of the environment with discrete semantic states, known as contexts [39]. These discrete states are termed as context instances in this paper. Each Fog node is endowed with context management capability and comprises of different context engines associated with each application. However, current context management engines and in Iot are implemented within their own domain and maintains its own user data and users, which does not promote the unified IoT services. Therefore, our proposed method considers context sharing between different vertical IoT applications such that it can realize scalable and innovative services. The context management unit is responsible for sharing the context instances. Context instances can be shared between

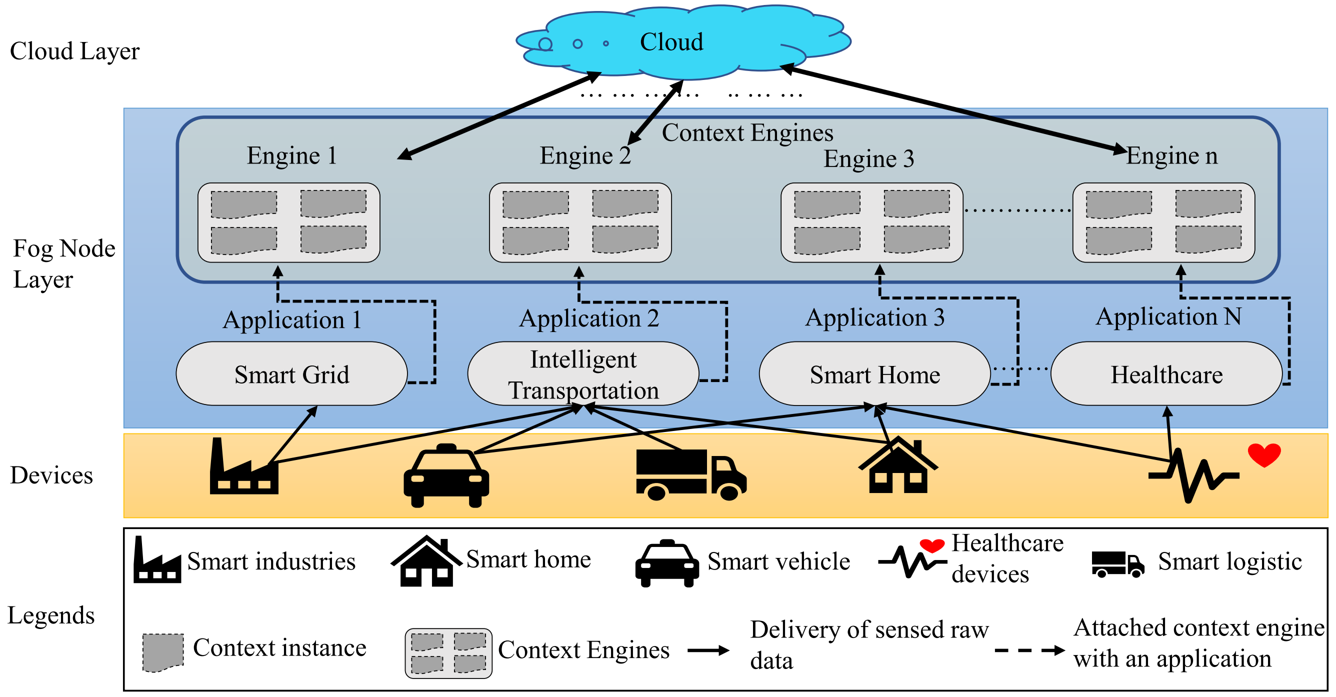
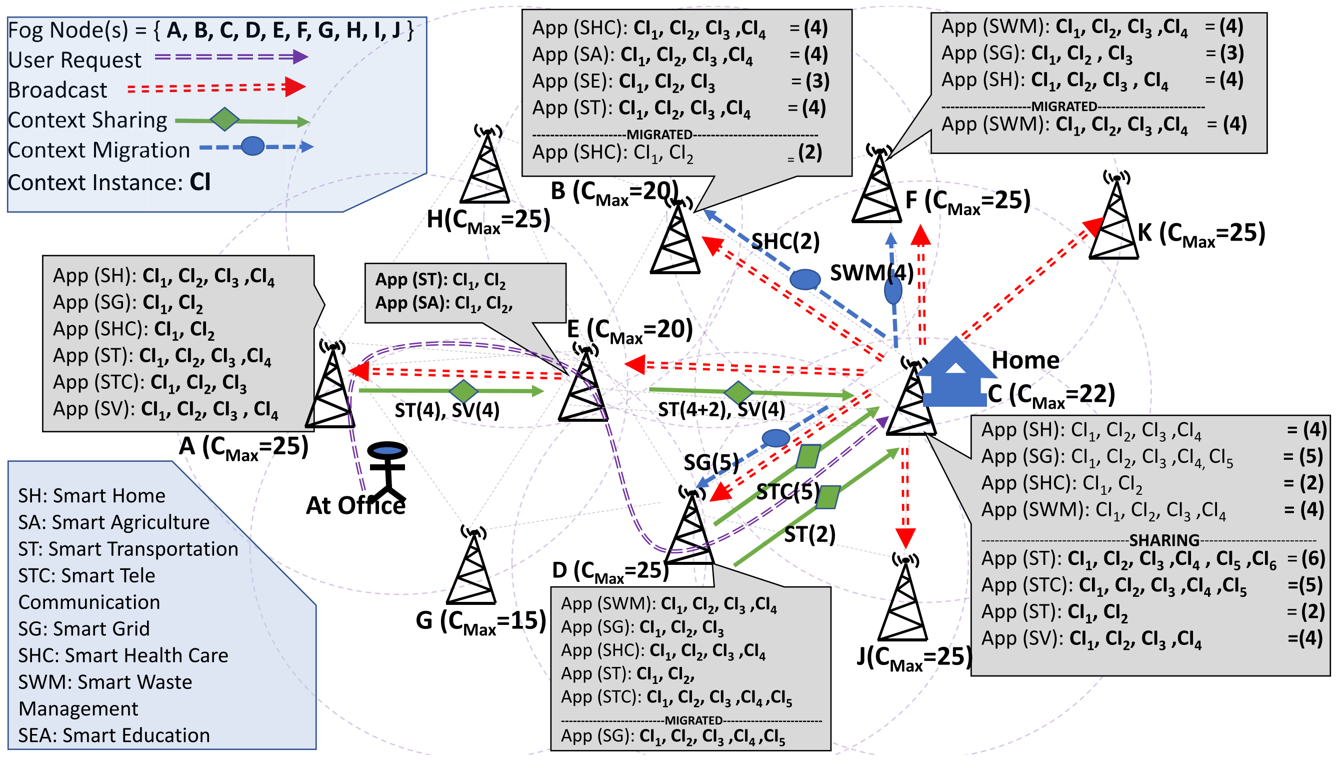


Fig. 1. Cross-domain system architecture.



Fog Node(s) = { A, B, C, D, E, F, G, H, I, J }
User Request
Broadcast
Context Sharing
Context Migration
Context Instance: CI
App (SHC): CI_1, CI_2, CI_3, CI_4 = (4)
App (SA): CI_1, CI_2, CI_3, CI_4 = (4)
App (SE): CI_1, CI_2, CI_3 = (3)
App (ST): CI_1, CI_2, CI_3, CI_4 = (4)
MIGRATED
App (SHC): CI_1, CI_2 = (2)
App (SWM): CI_1, CI_2, CI_3, CI_4 = (4)
App (SG): CI_1, CI_2, CI_3 = (3)
App (SH): CI_1, CI_2, CI_3, CI_4 = (4)
MIGRATED
App (SWM): CI_1, CI_2, CI_3, CI_4 = (4)
H(C_{Max}=25)
B (C_{Max}=20)
F (C_{Max}=25)
K (C_{Max}=25)
SHC(2)
SWM(4)
App (SH): CI_1, CI_2, CI_3, CI_4
App (SG): CI_1, CI_2
App (SHC): CI_1, CI_2
App (ST): CI_1, CI_2, CI_3, CI_4
App (STC): CI_1, CI_2, CI_3
App (SV): CI_1, CI_2, CI_3, CI_4
App (ST): CI_1, CI_2
App (SA): CI_1, CI_2,
E (C_{Max}=20)
Home
C (C_{Max}=22)
ST(4), SV(4)
ST(4+2), SV(4)
A (C_{Max}=25)
At Office
SG(5)
STC(5)
ST(2)
App (SH): CI_1, CI_2, CI_3, CI_4 = (4)
App (SG): $CI_1, CI_2, CI_3, CI_4, CI_5$ = (5)
App (SHC): CI_1, CI_2 = (2)
App (SWM): CI_1, CI_2, CI_3, CI_4 = (4)
SHARING
App (ST): $CI_1, CI_2, CI_3, CI_4, CI_5, CI_6$ = (6)
App (STC): $CI_1, CI_2, CI_3, CI_4, CI_5$ =(5)
App (ST): CI_1, CI_2 = (2)
App (SV): CI_1, CI_2, CI_3, CI_4 =(4)
SH: Smart Home
SA: Smart Agriculture
ST: Smart Transportation
STC: Smart Tele Communication
SG: Smart Grid
SHC: Smart Health Care
SWM: Smart Waste Management
SEA: Smart Education
D (C_{Max}=25)
G (C_{Max}=15)
J(C_{Max}=25)
App (SWM): CI_1, CI_2, CI_3, CI_4
App (SG): CI_1, CI_2, CI_3
App (SHC): CI_1, CI_2, CI_3, CI_4
App (ST): CI_1, CI_2,
App (STC): $CI_1, CI_2, CI_3, CI_4, CI_5$
MIGRATED
App (SG): $CI_1, CI_2, CI_3, CI_4, CI_5$

Fig. 2. Network model for context management.

dependent applications only. The dependent applications can reside in same Fog node or at different Fog nodes, thus making context instances sharable. Without loss generality, we consider that all Fog nodes are reachable from each other. Context manager broadcasts the request of each application to other Fog nodes periodically. Each context manager has knowledge about dependent applications. The dependent applications at each Fog node respond back with their generated context instances within a predefined time interval. After analyzing all context instances, the applications deliver services to actuators and users. Fig. 2 shows the aforesaid network model.

In this network model, we consider the connected cellular network of Fog nodes as shown in Fig. 2, as an undirected graph $G = (k, L)$ where k is the set of connected Fog nodes and L represents the links between those nodes. Each Fog node has a set of applications which are denoted by A. An application $j \in A$ generates a set of context instances periodically. The set of such context instances is denoted as C. The details pertaining to context switching and management is explained in Section V. For sharing of context between Fog nodes, we assume that the number of subcarriers is fixed, say Sn. We fix the number of the subcarriers in order to maintain the quality of other services, like data delivery or sending a control signal to actuators. Without loss of generality, all Fog nodes have the same number of subcarriers with the same bandwidth and same data rate. Moreover, it is assumed that the arrival of context happens randomly as a Poisson process [40]. For the convenience of readers, the major notations mentioned till now and to be used later are listed in Table I.

Our assumed network model utilizes Fog computing as a promising technology for realizing the cross-domain applications through context sharing. However, the main service requirement of Fog computing is to share context in real-time with allowable delay while having limited computation capabilities, unlike Cloud computing. The dynamic behavior of IoT applications and devices can create the delay problem. For example, the accumulation of IoT devices at a single Fog node can overload the resources at that node. Hence, such issues faced by context sharing approach in Fog node should be focused upon. Therefore, the next section introduces the problem statement related to context sharing in Fog computing.

B. Problem Statement

The service requirement of context sharing in Fog is to provide with low service delay. It can be more challenging for real-time IoT applications. However, Fog computing has low latency since it brings the Cloud capabilities at the edge of the network. Nevertheless, Fog is not always delay-aware or latency-aware because of the dynamic behavior of IoT devices. For example, many devices data may be aggregated within a Fog node thus overloading the resources therein. However, another Fog node might not be as stressed at that instant. The congested node can thus deliver delayed services and can worsen its situation further. Therefore, context sharing should be effectively designed to reduce such kind of delays. Low service delay can be achieved if the delay in communication (transmission delay) amongst sharing Fog nodes are reduced and also if the time taken for processing these context instances from different Fog nodes (processing delay) is minimized. Existing literature have neither considered context sharing in IoT devices nor have they studied context sharing using Fog computing. The objective of this paper is to minimize overall service delay for deployment of unified IoT services by context sharing using Fog computing. Our proposed approach is reducing transmission delay and processing delay simultaneously rather than focusing on them separately.

TABLE I
LIST OF NOTATIONS

Acronym	*Definition*
C_{max}	Maximum number of context instances
N_F	Total number of Fog nodes
N_{IC}	Number of incoming context from a node
N_p	Number of Fog node sharing context
N_{TC}	Number of migrated instances to nearby node
N_m	Number of nodes participating in migration
NA	Number of available context instances
R/S	Recipient node/supporting node
k/i/j	Fog node number/context instance
d_{kR}	Distance between nodes
T_{ctr}	Transmission time of context instances
T_M	Migration time of context instances to other Fog nodes
T_p	Processing delay
T_s	Total time for context sharing
t_{Tr}	Transmission time for transmitting one
bk 0/1)	Boolean variable to decide the participating
λ	Arrival rate of context instance
μ	Service rate of incoming context instances
D_k	Communication delay
L_i	Length of context instance i
DR	Data rate
S_n	Number of carrier associated for each Fog node for context instance transmission
R	Data rate at each subcarrier
$AppC_{total}$	Is total contexts of a specific applications
$AppC_{Min}$	Is the minimum contexts required to process a request
nc	Is the Number of context instances
C_{total}	Is the total number of context available at a Fog node
C_{extra}	Is the number of extra contexts
App.delayTolerance	It denotes the latency sensitive of an applications
App.delayTolerance=2	It denotes the applications with high delay tolerance
App.delayTolerance=1	It denotes the applications with medium delay tolerance
node.Cmax-node.NA	To get available capacity of of a Fog node

The transmission and processing delay can be reduced by limiting the maximum number of context instances residing at each Fog node. $C_{\max}$ denotes the maximum number of context instances at a Fog node. Limiting the number of context instances can have two advantages: first, it minimizes the processing delay by limiting the number of context instances as per computation capability (servicing rate); second, it can manage the storage capacity of Fog node which is also limited. However, this paper does not focus on minimization of storage capacity. An efficient algorithm for context sharing and balancing the number of context instances if it crosses the maximum number of context instances can minimize the transmission delay. Such an efficient method for context sharing across Fog nodes while considering its limitations is proposed in the next section.

V. Smart Context Sharing Model for Enabling Unified IoT Services

To solve the aforesaid problem, a new and efficient method is needed to enable unified IoT services by sharing contexts of different applications using Fog computing. In this vein, this section presents a smart context sharing (SCS) algorithm. Furthermore, our algorithm improves the service delay during context sharing among Fog nodes, which helps in realization of real-time,cross-domain IoT services. The details of our proposal are explained in the next section.

A. Proposed Smart Context Sharing Model

The proposed model aims at transmitting context instances from different applications residing at various Fog nodes to requested applications in another Fog node. In our proposed approach, applications at each Fog node generate context instances periodically. Generation of context instances is a well-known approach; whereas sharing the context of the heterogeneous IoT applications is yet a novel one. Therefore, we have not focused on the generation of context instances. Let the context instances generated by Fog node k and application j is C_{ij}^{k}. The context manager will prioritize all context instances of different IoT applications as per their QoS requirement such as delay criticality or delay tolerance. Without loss of generality, we are presenting the algorithm of our approach from the viewpoint of a Fog node. It can be a model for other Fog nodes similarly. A Fog node broadcasts context requests for instances by an application periodically. After receiving the broadcast message, each Fog node within the range of the broadcasting node responds with the context instances. The applications dependent on the requesting application share the context instances only.

As we observed in the previous section, the maximum number of context instances residing at a Fog node is limited, i.e., $C_{\max}$. Therefore, the incoming number of context instances (N_{IC}) and existing number of context instances (C_{total}) should not exceed $C_{\max}$. If it is less than $C_{\max}$, the broadcasting Fog node will accept all the incoming context instances, otherwise the exceeding context instances should migrate to another nearby free Fog node and this process is termed as balancing of context instances which has been shown in Algorithm 2.

The event of surpassing the limit of $C_{\max}$ can be termed as congestion for a context instance. The context manager migrates the exceeding context instances based on a priority decided ahead of time. The context manager finds the closest free node which can accommodate the extra context instances and can thus migrate N_{TC} such instances to the closest node. The algorithm of our proposal is shown in Algorithm 1. The notations and the comments of conditions of Algorithms 1 and 2 are also shown in Table I.

However, our objective is to provide an efficient method for context sharing. The service delay depends on the maximum number of context instances at each Fog node. Service delay can be optimized if the $C_{\max}$ value is optimal and thus the service delay is a function of $C_{\max}$. Because of this, the main idea behind our proposal is to find an optimal value of $C_{\max}$ in order to optimize the service delay of context sharing. In mathematical terms, this is reduced by solving an optimization problem. The service delay in our proposal depends on three variables, namely transmission time of context instances to

Algorithm 1: SCS Algorithm

```
Begin
Generate different context instances from the different
services of every application at each Fog node. Prioritize
the context instances as per IoT applications QoS
requirement
@Sender site each:
for Fog node do
  NeighborList:= Collect all neighbor Fog node info
  within range.
  @Repeat
  @Sender:
  for each Fog node do
    $C_{Max}$ = Random(Max1,Max2)
    if ($AppC_{total} < AppC_{Min}$) then
      Broadcast (SenderId, “msg: request for required
      context instance”,NeighborList)
    end if
  end for
  @All Receiver site:
  Check for availability of requesting contexts instances
  and send the number of instances (nc) of requested type
  Send(SenderId, nc, ReceiverId);
  @Sender site:
  t=delay tolerance of an application;
  while t <= (resTime+commTime do
    $C_{total}$= SUMMATION nc from all Receivers.
  end while
  if ($C_{total}$ +$N_A$) > $C_{Max}$ then
    $C_{extra}$=( $C_{total}$+ $N_A$ )-$C_{Max}$
    if $C_{total} < N_A$ then
      Process the Application contexts
    else
      $C_{extra}$= ($C_{total}$+$N_{IC}$) $C_{Max}$
      DelayTolerantLoadBalancing(NodeId, NeighborList ,
      Cextra)
      Migrate $C_{extra}$ context instances to neighbor Fog
      nodes
      $C_{total}$ = $C_{total}$+$N_{IC}$
    end if
    Broadcast (SenderId, msg: ready to receive contexts,
    NeighborList)
    @All Receiver site: ——————
    for all node : NeighborList do
      if available then
        Send(SenderId, ContextList, ReceiverId);
      end if
    end for
    @Sender site:
    t=0;
    while (t<=(resTime+commTime)) do
      Receive(contextList, senderId, receiverId );
      $N_{IC}$ =count(contextList);
      $C_{total}$=$C_{total}$+$N_{IC}$;
      t=t+1;
    end while
  else
    Balance the Fognode($C_{extra}$)
  end if
  Until the ‘t’ time units are finished
end for
```

Algorithm 2: DelayTolerantLoadBalancer (NodeId, NeighborList, Cextra)

```
1: Begin
2: @Sender site:
3: for all node : NeighborList do
4:    NeighborList .Capacity=node.$C_{max}$ - node.$N_A$
5: end for
6: while ($C_{extra}$ > 0) do
7:    closestNode= minDistance(NeighborList)
8:    for all App:SenderNode do
9:       if (App.delayTolerance = 2) then
10:         MigAppList. Add(App)
11:      end if
12:   end for
13:   if (MigAppList is NOT NULL) then
14:      for all App: MigAppList with
         RL-MinTime(MigAppList) do
15:         if (App = closestNode.App AND Cextra > 0) then
16:            if (closestNode.Capacity > closestNode.Capacity )
               then
17:               Migrate(App, App.Capacity, closestNodeId)
18:               Cextra= Cextra - App.Capacity
19:            else
20:               Migrate(App, closestNode.Capacity,
                  closestNodeId)
21:               Cextra= Cextra - closestNode.Capacity
22:            end if
23:         end if
24:      end for
25:   else
26:      for all App: SenderNode do
27:         if (App.delayTolerance = 1) then
28:            MigAppList. Add(App)
29:         end if
30:      end for
31:      if (MigAppList is NOTNULL) then
32:         for all App: MigAppList with
            RL-MinTime(MigAppList) do
33:            if (App = closestNode.App AND $C_{extra}$ >0) then
34:               if (closestNode.Capacity >
                  closestNode.Capacity) then
35:                  Migrate(App, App.Capacity, closestNodeId)
36:                  $C_{extra}$= $C_{extra}$ - App.Capacity
37:               else
38:                  Migrate(App, closestNode.Capacity,
                     closestNodeId)
39:                  $C_{extra}$= $C_{extra}$ - closestNode.Capacity
40:               end if
41:            end if
42:         end for
43:      else
44:         Msg(Contexts cant be migratedİ)
45:      end if
46:   end if
47: end while
48: end
```

the broadcasting node (T_{ctr}), the migration time of context instances to other nodes during congestion (T_M), and the processing delay (T_P). Our proposal has considered both computation and communication delay. We can say that average service delay can be optimized by optimizing below formulas

$$T_S = T_{\text{ctr}} + T_M + T_P. \tag{1}$$

The initial step is to formulate transmission time, migration time, and processing time that are, T_{ctr}, T_M, and T_P, in order to write final objective function.

The transmission time depends on the number of incoming context instances from different participating Fog nodes and the transmission time of each context instance. Therefore, the transmission time can be enumerated using

$$T_{\text{ctr}} = \sum_{k=0}^{N_p} N_{\text{IC}} \times t_{\text{Tr}}^{kr} \tag{2}$$

where N_p is the number of participating node, N_{IC} is the number of incoming instance from each participating Fog node, and t_{Tr} is the transmission time of a context instance from participating node to broadcasting node. However, our proposed approach has considered a Poisson arrival rate for incoming context instances.

Moreover, the incoming context instances are affected by delayed context instances due to delay in their arrival. Therefore, at time t, the number of context instances can be expressed as

$$N_{\text{IC}}^k = \lambda t - \lambda d_k = \lambda(t - d_k) \tag{3}$$

where λ is the arrival rate of context instances from a node, t is the observation time, and d_k is the communication delay. To the best of our knowledge, no literature has incorporated the arrival rate of context instances and despite all efforts, the authors could not provide values of arrival rate based on concrete data. Therefore, a Monte Carlo simulation (MCS) has been employed for estimating the arrival rate of context instances. The MCS approach is used for visualization of the actual process and random behavior of systems in a well-designed way [39]. This approach is used to find the number of time an event happens within a defined mission time. In this paper, MCS is chosen to estimate the arrival rate by integrating the probability density function of context instances which has been assumed to be a Poisson process. The value of λ can be obtained by (8)

$$f(t) = \lambda \exp^{-\lambda t} \tag{4}$$

$$F(t) = \int_0^t f(t)dt \tag{5}$$

$$F(t) = \int_0^t \lambda \exp^{-\lambda t} dt \tag{6}$$

$$F(t) = 1 - \exp^{-\lambda t} \tag{7}$$

$$\lambda = -(\ln(1 - F(t)))/t. \tag{8}$$

Using MCS, the value of 1-F(t) can be randomly generated.

$$\lambda = -(\ln \alpha_1)/t \tag{9}$$

$$\alpha_1 = 1 - F(t). \tag{10}$$

Once arrival rate is determined using MCS approach, the number of incoming context instances without delay can be estimated. However, the delay can happen due to network congestion or packet loss due to high signal-to-noise ratio. Therefore, the number of delayed context instances can be found based on

$$d_k = \frac{L_i^k}{S_n \times C} \tag{11}$$

where d_k is the communication delay, S_n is the number of subcarriers available for context instance transmission, and L_i^k is the size of context instances. However, channel capacity depends on different parameters but we have employed the Shannon–Hartley theorem [41] to estimate the channel capacity between Fog nodes

$$C = B\log_2\left(1 + \frac{S}{N}\right). \tag{12}$$

Here, B is the bandwidth of the channel in Hertz, S is the average received signal power over the bandwidth, and N is the average noise and interference over the bandwidth. We have used the predefined value these parameters suitable for cellular networks. Substituting the value of C from (12) in (11), the delay in transmission of context instances can be found. The communication delay (d_k) value and value of λ are substituted in (3) to estimate the number of incoming context instances. The number of incoming instances and transmission time of a context instance are substituted in (2) from (3) and (13), respectively, in order to estimate all context instances transmission time from broadcasting Fog nodes

$$t_{\text{Tr}} = \frac{d^{kr}}{Y} \tag{13}$$

where d^{kr} is the distance between the broadcasting Fog node k and receiving Fog node r and Y is the propagation speed in wireless medium. Next, we need to formulate the migration delay of congested context instances. Congested context instances are extra instance which is N_{TC}. The migration time is defined

$$T_M = \sum_{k=1}^{N_m}\left(N_{\text{TC}}^k \times t_{\text{Tr}}^{ks} + N_{\text{TC}}^R \times t_{\text{Tr}}^{RS}\right) \tag{14}$$

where N_m is the number of participating nodes, N_{TC}^R and N_{TC}^k are the number of context instances migrated from participating nodes and broadcasting nodes, respectively. Also, t_{Tr}^{ks} is the transmission delay between the broadcasting node and its closest node and t_{Tr}^{Rs} is the delay between the participating node and its closest node. However, as per our assumption, transmission time between all Fog node is the same, i.e., t_{Tr}. Therefore, (14) can be written as

$$T_M = \sum_{k=1}^{N_m}\left(N_{\text{TC}}^k \times t_{\text{Tr}} + N_{\text{TC}}^R \times t_{\text{Tr}}\right). \tag{15}$$

The value of N_{Tc} needs to be estimated for calculating migration time. The number of migrating context instances is the difference between the incoming and available context instance at a Fog node k. Therefore,

$$N_{\text{TC}} = N_{\text{IC}}^k - C_{\max}^k \tag{16}$$

where N_{IC} can be calculated from (3) and $C_{\max}^k$ is the optimal value which will be estimated later. The value of N_{IC} from (16) and value of t_{Tr} from (13) can be substituted in (15) to estimate the migration time of context instances.

Thereafter, we need to estimate the processing time at each Fog node. The formulation of processing delay is as follows:

$$T_p = \frac{1}{(\mu(t) - \lambda(t)) \times N_{\text{IC}}^k} \tag{17}$$

$$\mu = -\frac{\left(\ln(1 - F'(t))\right)}{t} \tag{18}$$

$$1 - F'(t) = \alpha_2 \tag{19}$$

$$\mu = -\frac{\ln \alpha_2}{t} \tag{20}$$

$$\lambda = -\frac{\ln \alpha_1}{t}. \tag{21}$$

$\mu(t)$ is the service rate function at t of a Fog node, which is calculated similar to an arrival rate using Poisson process and MCS. Substituting the value from (20) and (21) in (17), the value of processing time can be estimated. Equations (2), (15), and (17) give the required values of transmission delay, migration delay, and processing delay, respectively. The objective function to minimize the service delay of context sharing is written below with its constraints

$$\begin{aligned} \text{minimize} \quad & T_s = T_{\text{ctr}} + T_M + T_p \\ \text{s.t.} \quad & N_A^k + N_{\text{IC}}^k \le C_{\max}^k \\ & N_{\text{TC}}^k + N_A^S \le C_{\max}^S \\ & C_{\max}^k, C_{\max}^S > 0 \end{aligned} \tag{22}$$

where N_A^k is the available context instances at Fog node k, N_{IC}^k is the incoming context instances to Fog node k, and $C_{\max}^k$ is the maximum number of context instances allowed at a Fog node k. The sum of incoming context instances and available context instances should be less than or equal to a maximum number of context instances allowed at node k, which is depicted in (22). Similarly, the limit of context instances that a supporting or nearby node should satisfy is specified in (22). In (22), N_{TC}^k is the migrated instance to the closest node S from the broadcasting node k whereas is the number of available context instances at the nearby node to the Fog node or supporting node.

Their summation should also be less than $C_{\max}^S$. The final constraint is that $C_{\max}^k$ and $C_{\max}^S$ should be greater than 0. Equation (22) should be minimized subject to given constraints. We can get the optimal value of $C_{\max}^k$ after solving (22). Similarly, our proposal calculates optimal value for all other Fog nodes. The optimal values are assigned to all Fog nodes. Thereafter, each Fog node implements Algorithms 1 and 2. Algorithm 1 presents in a nutshell the essential functionalities of SCS approach, whereas Algorithm 2 presents in summary the essential functionalities of a delay tolerant load balancing approach.

Each Fog node implements both the functionalities to provide the service to user requests. For each request that arrives at a Fog node, it checks the presence of a sufficient number of context instances and also the required number of context instances in it. Once these two criteria are satisfied, then it proceeds to check whether it can provide service response within the stipulated service delay. If so, then it proceeds toward processing. If any of the above three criteria is not satisfied, then it invokes the SCS algorithm (Algorithm 1) to manage.

The first two @Sender sections deal with collecting of neighbors information via a request message to all neighbors for sending the required type of context instances with minimum service delay. Next @Receiver section provides the details about the operations that all receivers entail. They react to sender request after checking the presence of context instances of requested type. If available, then they send to requester. Next @Sender sections in the algorithm describes the delay calculation for the delay incurred for sending request and receiving context instances along with the required buffer size ($C_{\max}$) for temporarily incorporating within Fog node for processing the user requests. If available $C_{\max}$ is less than what is required, then some contexts are migrated to other node temporary storage, either for unused context instances or for extra context instances of an application or for context instances for a low priority application. If none of these criteria are satisfied, then the service is itself migrated to some other Fog node for processing with the help of DelayTolerantLoadBalancing (Algorithm 2).

Algorithm 2 presents in details the load balancing approach without the violating service delay constraints. Initially it collects all neighbors' information such as available capacity, pending load, presence of context instances of required application and its sufficient number of instances for processing. Based on this information, the algorithm decides to migrate the sender service request to other Fog nodes for processing. The first section of this algorithm deals with delay sensitive applications and the second part deals with applications having medium and high delay tolerance. Lower and upper sections of this algorithm are used to predict a suitable node to migrate. In order to improve the performance of the prediction technique, we incorporated a machine learning approach namely, reinforcement learning (RL), which is adaptive in nature to the environment.

VI. Performance Evaluation

In this section, we have discussed details related to the performance evaluation of the proposed SCS and load balancing algorithms in terms of their efficacy in minimizing service delays. The simulation for the network architecture has been setup in NS-2 simulator. In this section, the various details and initial values pertaining to the simulation setup are presented. The initial values of parameters have been chosen as per those provided in [20], [25], and [42].

A. Simulation Step

1) Network Topology: The simulation of Fog computing network was build with a set of Fog nodes, the number of such nodes being varied from 10 to 100 with increments of 10. An appropriate structure is used to store all the relevant information pertaining to a Fog node, such as its neighbors, its access range, response time, and $C_{\max}$ size. The list of such structures, named Fn, is created for maintaining information for all Fog nodes and its adjacent nodes in the network. For instance, if the topology considers 100 Fog nodes for simulation at any instant, Fn[1...100][1...100] is used to provide all required information.

2) Network Traffic: The data traffic is generated at different Fog nodes using a random function within the range of 10%–50% of $C_{\max}$ size.

3) Terminal Nodes: The total number of Fog nodes in the simulation study has been kept fluid within a range of 10 to 100 in order to build a system that is more realistic and to measure the system performance against varying conditions. Capacity of Fog nodes ($C_{\max}$) is assigned in the simulations randomly within a range of 500 to 8000, with increments of 500. The response time of these Fog nodes are kept within a limit in between 2 and 10 ns and this is also generated randomly.

4) Application: In order to simulate the above environment, a total of 20 different applications were considered with different computational requirements. The number of applications and the type of applications for each Fog node is generated using a random number generated, a unique number being mapped to represent a certain application. A realistic characterization of such IoT applications can be found in [42]. In order to simplify the evaluation of our proposed algorithm for various IoT infrastructures, we classified customers' requests into three classes depending upon their delay sensitiveness. For example, Class-I considers highly delay tolerable applications like smart searching, smart agriculture, smart home, and smart parking. Class-II considers medium delay tolerable application like smart shopping, smart retailing, smart pay, and so on, whereas Class-III considers highly delay sensitive applications from health care, defense, and intrusion detection areas. The delay tolerance of the above 20 applications were set to any one of the three different categories, namely low, medium, and high and their values were assumed to be 50 ns, 200 ns, and 500 ns, respectively. The applications were classified into three different priority levels (low, medium, and high) depending upon its application type. High priority applications are provided preference for processing at Fog nodes.

Different application requests are generated from these three classes in order to make the simulation environment more realistic. Context information of Fog nodes are dynamically varied with time. The proposed SCS algorithm is thus employed to appropriately prioritize the service requests and share their contexts accordingly among Fog nodes along with efficient duty cycling. Simulated network environment is highly dynamic and the network topology is dynamically changed with time.

5) Migration Policy: When the number of context instances exceed beyond the maximum capacity of a Fog node, the context instances of low priority applications are migrated to nearby neighbors with minimum response time. The response time of each Fog node are set at the beginning of the simulation setup. The response time of these Fog nodes are varied according to the current load and this current load depends on the number of applications and type of applications being handled by the nodes at that instant. To measure the efficacy of the system in terms of service delay, the above simulation setup is employed with varying $C_{\max}$ values ranging from 500 to 8000 for a constant number of user requests. The service delay is recorded for each $C_{\max}$ value which is varied uniformly with increments of 500. Service delays are recorded at different Fog nodes under varying load scenarios. Simulations are repeated for 20 times and mean values of obtained results are presented. To measure the effect of $C_{\max}$ on the overall system performance, we simulated our

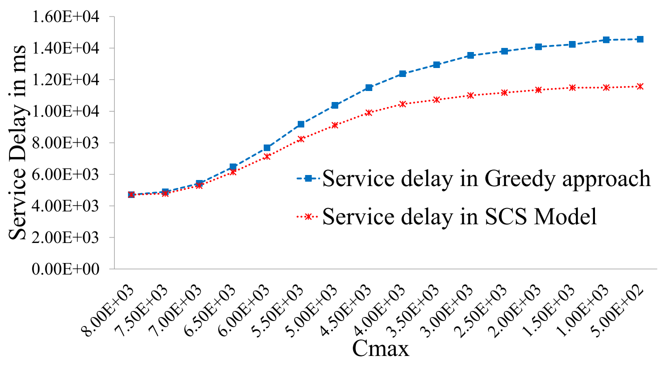


Fig. 3. Service delay with varying $C_{\max}$.

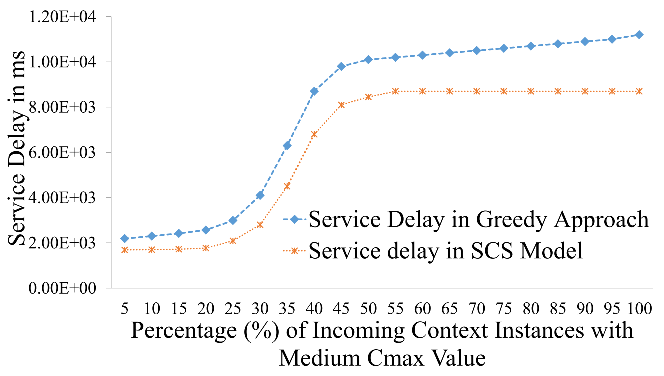


Fig. 4. Service delay over percentage of incoming context instances with low $C_{\max}$ values.

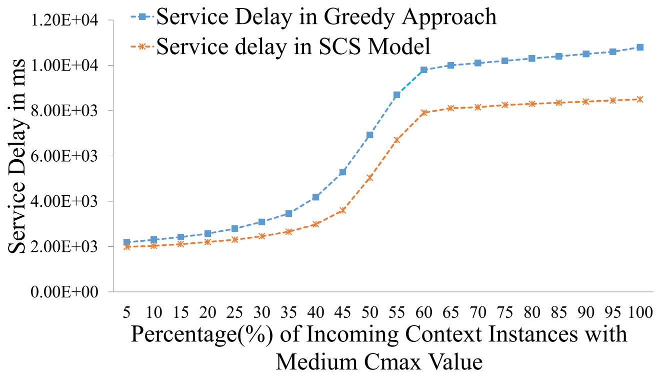


Fig. 5. Service delay over percentage of incoming context instances with medium $C_{\max}$ values.

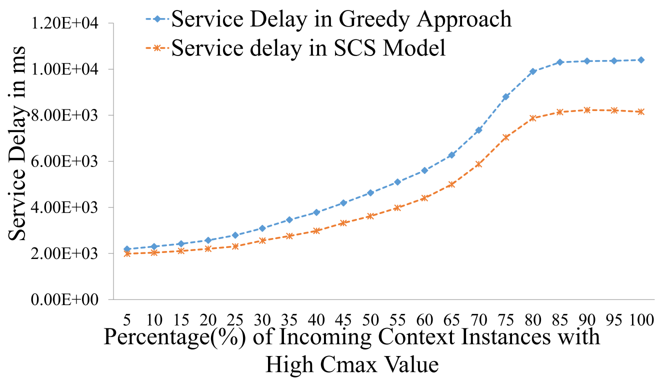


Fig. 6. Service delay over percentage of incoming context instances with high $C_{\max}$ values.

model under three different ranges of $C_{\max}$. Details regarding the range of $C_{\max}$ values are given in the results section. We have also simulated a greedy approach for the above scenarios with the same ranges as defined previously. Greedy approach always selects the choice that seems to be the best at that moment. During migration, greedy approach selects a neighbor for migrating a user request based on the information available with sender node. On the contrary, in the SCS model, we employed a prediction approach which predicts the response time and delay required to complete the required task. This is done intuitively since Fog node characteristics are dynamic in nature, varying from time to time depending upon arrival of requests. Inputs are generated from users and these users are deployed at different Fog nodes with different application requirements. Using this information, in our model, an RL prediction scheme is employed to predict the best Fog node for a particular purpose, unlike the greedy approach which decides upon the neighbor simply on the basis of static information it has.

B. Results and Discussion

To validate the proposed SCS model, we carried out a number of experiments and performance of our proposed model is compared with the performance of the greedy approach. During our experimental analysis, we observed service delay of our model by varying context instances. Fig. 3 depicts the service delay with varying $C_{\max}$ size from 500 to 8000 ($C_{\max}$ size indicates the number of context instances can store in a Fog node) with a fixed incoming user request. In our experiment, we set the $C_{\max}$ value to three categories like high, medium, and low with ranges of 5001 to 8000, 3001 to 5000, and 500 to 3000, respectively. Fig. 3 demonstrates that the difference in efficacy among the two approaches is very less up to a certain $C_{\max}$ value (up to around 5500), and thereafter SCS model outperforms the greedy approach in terms of service delay. Figs. 5–7 capture the relative performance of the proposed SCS scheme over its greedy counterpart for service delay with different $C_{\max}$ values and a particular number of incoming user requests. From Fig. 4, it can be observed that our proposed SCS provides performance improvement of around 5%–24% in service delay over its greedy counterpart. Fig. 5 depicts that performance improvement of the proposed SCS model is up to 21% with medium $C_{\max}$ value, whereas Fig. 6 depicts the performance improvement of SCS model up to 19% over its greedy counterpart with high $C_{\max}$ value.

Fig. 7 depicts the performance differences between SCS model and greedy approach with variations in migrated context migration extents. The efficacy of both models are close as long as 40% of contexts are migrated. However, beyond that, there is a marked performance degradation observed for greedy approach resulting in up to 31% degradation in service delays. In order to achieve reduced service

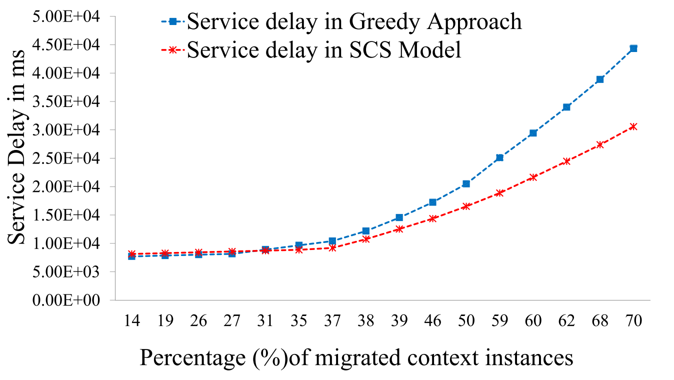


Fig. 7. Effect of percentage of migrated context instances on service delays.

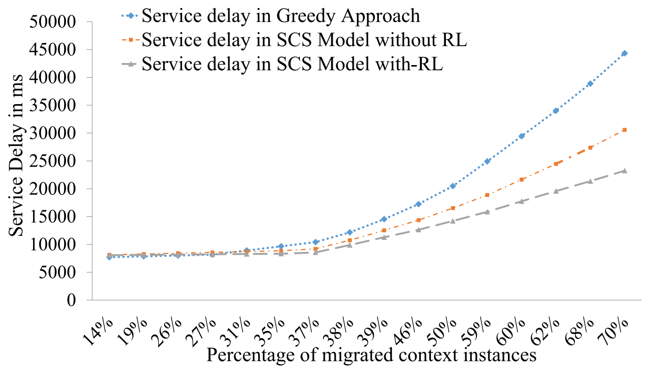


Fig. 8. Efficacy comparison of RL approach over migrated context instances.

delay from context migration point of view, we incorporated a delay tolerant load balancer depicted with RL in Algorithm 2. Fig. 8 depicts the performance differences between greedy, SCS model and SCS with reinforcement learning (SCS-RL) approaches. The efficacy of all three models are close as long as context migrations are less. However, with increase in migrated context instances, SCS model outperforms the greedy approach considerably up to 27%–29% and SCS-RL outperforms the SCS model considerably up to 30%–33%.

VII. Conclusion

Myriad existing IoT applications have paved the path for next generation cross-vertical IoT applications to be envisioned. Meeting the QoS requirements of such unified IoT services requires improvised network architectures that leverage context sharing among IoT applications within the Fog nodes. This paper presents a novel scheme for materializing such network architecture comprising of Cloud and Fog resources that is oblivious to underlying communication protocols. To that end, this paper proposes a novel SCS scheme that in conjunction with context migration among Fog nodes can reduce service delay. Additionally, an SCS scheme has been shown to outperform greedy schemes by approximately 5%–20% depending upon the Fog nodes' capabilities. In future, we shall investigate the ramifications that such a scheme can have on the energy consumption of the network architectures.

References

[1] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Smartphone-based user activity recognition method for health remote monitoring applications," in *Proc. PECCS*, 2012, pp. 200–205.

[2] I. Bisio, A. Delfino, F. Lavagetto, and A. Sciarrone, "Enabling IoT for in-home rehabilitation: Accelerometer signals classification methods for activity and movement recognition," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 135–146, Feb. 2017.

[3] M. I. Al, P. Patel, S. K. Datta, and A. Gyrard, "Multi-layer cross domain reasoning over distributed autonomous IoT applications," *Open J. Internet Things*, vol. 3, no. 1, pp. 75–90, Nov. 2017.

[4] S. K. Datta, J. Haerri, C. Bonnet, and R. F. D. Costa, "Vehicles as connected resources: Opportunities and challenges for the future," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 26–35, Jun. 2017.

[5] S. Soursos *et al.*, "Towards the cross-domain interoperability of IoT platforms," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Athens, Greece, Jun. 2016, pp. 398–402.

[6] A. Gyrard, S. K. Datta, C. Bonnet, and K. Boudaoud, "Cross-domain Internet of Things application development: M3 framework and evaluation," in *Proc. 3rd Int. Conf. Future Internet Things Cloud (FiCloud)*, Rome, Italy, Aug. 2015, pp. 9–16.

[7] M. Blackstock and R. Lea, "Toward interoperability in a Web of Things," in *Proc. ACM Conf. Pervasive Ubiquitous Comput. Adjunct Publ.*, Zürich, Switzerland, Sep. 2013, pp. 1565–1574.

[8] A. Gyrard, C. Bonnet, and K. Boudaoud, "Enrich machine-to-machine data with semantic Web technologies for cross-domain applications," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Seoul, South Korea, 2014, pp. 559–564.

[9] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.

[10] B. Guo, L. Sun, and D. Zhang, "The architecture design of a cross-domain context management system," in *Proc. 8th IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, Apr. 2010, pp. 499–504.

[11] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge node resource management," *IEEE Trans. Services Comput.*, to be published, doi: 10.1109/TSC.2017.2753775.

[12] X. Li, M. Eckert, J. F. Martinez, and G. Rubio, "Context aware middleware architectures: Survey and challenges," *Sensors*, vol. 15, no. 8, pp. 20570–20607, Aug. 2015.

[13] C. Kamienski *et al.*, "Context-aware energy efficiency management for smart buildings," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Milan, Italy, Dec. 2015, pp. 699–704.

[14] R. Fallahzadeh, Y. Ma, and H. Ghasemzadeh, "Context-aware system design for remote health monitoring: An application to continuous edema assessment," *IEEE Trans. Mobile Comput.*, vol. 16, no. 8, pp. 2159–2173, Aug. 2017.

[15] D. Evans, "The Internet of Things: How the next evolution of the Internet is changing everything," San Jose, CA, USA, CISCO, White Paper, Apr. 2011.

[16] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and Internet of Things," in *Proc. Int. Conf. Future Internet Things Cloud*, Barcelona, Spain, Dec. 2014, pp. 23–30.

[17] C. Chang, S. N. Srirama, and R. Buyya, "Indie fog: An efficient fog-computing infrastructure for the Internet of Things," *Computer*, vol. 50, no. 9, pp. 92–98, Sep. 2017.

[18] M. Bansal, I. Chana, and S. Clarke, "Enablement of IoT based context-aware smart home with fog computing," *J. Cases Inf. Technol.*, vol. 19, no. 4, pp. 1–12, Oct. 2017.

[19] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, vol. 546. Cham, Switzerland: Springer, 2014, pp. 169–186.

[20] (Oct. 21, 2017). *Cisco Delivers Vision of Fog Computing to Accelerate Value From Billions of Connected Devices*. [Online]. Available: https://newsroom.cisco.com/press-releasecontent?type=webcontent&articleId=1334100

[21] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, Feb. 2014.

[22] Z. Lv, H. Song, P. Basanta-Val, A. Steed, and M. Jo, "Next-generation big data analytics: State of the art, challenges, and future research topics," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1891–1899, Aug. 2017.

[23] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1910–1920, Aug. 2017.

[24] B. Tang *et al.*, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2140–2150, Oct. 2017.

[25] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and 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.*, vol. 47, no. 9, pp. 1275–1296, Jun. 2017.

[26] M. Aazam and E. N. Huh, "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT," in *Proc. IEEE 29th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Gwangiu, South Korea, Mar. 2015, pp. 687–694.

[27] M. Aazam and E.-N. Huh, "Dynamic resource provisioning through fog micro datacenter," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, St. Louis, MO, USA, Mar. 2015, pp. 105–110.

[28] C. Dsouza, G. J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proc. IEEE 15th Int. Conf. Inf. Reuse Integr. (IEEE IRI)*, Redwood City, CA, USA, Mar. 2014, pp. 16–23.

[29] S. J. Stolfo, M. B. Salem, and A. D. Keromytis, "Fog computing: Mitigating insider data theft attacks in the cloud," in *Proc. IEEE Symp. Security Privacy Workshops*, San Francisco, CA, USA, May 2012, pp. 125–128.

[30] S. Kulkarni, S. Saha, and R. Hockenbury, "Preserving privacy in sensor-fog networks," in *Proc. 9th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, London, U.K., Dec. 2014, pp. 96–99.

[31] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Honolulu, HI, USA, 2017, pp. 17–24.

[32] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[33] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1457–1477, 3rd Quart., 2017.

[34] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proc. 1st Workshop Mobile Comput. Syst. Appl.*, Santa Cruz, CA, USA, Dec. 1994, pp. 85–90.

[35] T. Gu, H. K. Pung, and D. Q. Zhang, "A service oriented middleware for building context aware services," *J. Netw. Comput. Appl.*, vol. 28, no. 1, pp. 1–18, Jan. 2005.

[36] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–267, Jan. 2007.

[37] G. D. Abowd, M. Ebling, G. Hung, H. Lei, and H. W. Gellersen, "Context-aware computing," *IEEE Pervasive Comput.*, vol. 1, no. 3, pp. 22–23, Jul. 2002.

[38] A. K. Dey, G. D. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *J. Human–Comput. Interact.*, vol. 16, no. 2, pp. 97–166, Dec. 2001.

[39] J. Venkatesh, B. Aksanli, C. S. Chan, A. S. Akyürek, and T. S. Rosing, "Scalable-application design for the IoT," *IEEE Softw.*, vol. 34, no. 1, pp. 62–70, Jan./Feb. 2017.

[40] (Oct. 21, 2017). *Random Poisson Process*. [Online]. Available: https://books.google.co.in/books?id=JRs3DwAAQBAJ&pg=PA127&dq=random+Poisson+process&hl=en&sa=&redir_esc=y#v=onepage&q&f=false

[41] L. M. Surhone, M. T. Timpledon, and S. F. Marseken, *Shannon–Hartley Theorem*. Saarbrücken, Germany: VDM, 2010. [Online]. Available: https://books.google.co.in/books?id=K89mcAAACAAJ

[42] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for IoT applications," *Ad Hoc Netw.*, vol. 62, pp. 101–115, Jul. 2017.

Diptendu Sinha Roy (M'14) was born in Hooghly, India. He received the B.Tech. degree from Kalyani University, Kalyani, India, in 2003, and the M.Tech. and Ph.D. degrees from the Birla Institute of Technology Mesra, Ranchi, India, in 2005 and 2010, respectively.

He is currently with the National Institute of Technology Meghalaya, where he is the Head of the Department of Computer Science and Engineering. His current research interests include distributed, grid computing, cloud computing, fog computing, software reliability, and optimization in engineering. He also performs research on design and analysis of distributed infrastructure of power systems.

Ranjit Kumar Behera was born in Balasore, India. He received the B.Tech. and M.Tech. degrees from the Biju Patnaik University of Technology, Rourkela, India, in 2004 and 2010, respectively.

He is currently with the National Institute of Science and Technology, Berhampur, India. His current research interests include distributed and grid computing, cloud computing, fog computing, and service oriented architectures.

K. Hemant Kumar Reddy was born in Berhampur, India. He received the M.Tech. and Ph.D. degrees from Berhampur University, Bhanja Vihar, India, in 2008 and 2014, respectively.

He is currently with the National Institute of Science and Technology, Berhampur, India, as an Assistant Professor. His current research interests include distributed and grid computing, cloud computing, fog computing, and service oriented architectures.

Rajkumar Buyya (S'03–M'03–SM'08–F'15) is a Professor of computer science and software engineering and the Director of the Cloud Computing and Distributed Systems Laboratory with the University of Melbourne, Melbourne, VIC, Australia. He is a Adjunct Professor with the National Institutes of Technology Meghalya, Shillong, India. He served as a Future Fellow of the Australian Research Council from 2012 to 2016. He has authored over 525 publications, 7 textbooks, and edited several books. He is one of the highly cited authors in computer science and software engineering worldwide (citations).

Dr. Buyya was a recipient of the Bharath Nirman Award and Mahatma Gandhi Award along with Gold Medals for his achievements in the information technology field and services rendered to promote greater friendship and India international cooperation, and the Highly Cited Researcher Award of the Web of Science by Thomson Reuters in 2016. Software technologies for grid and cloud computing developed under his leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Manjrasofts Aneka Cloud technology developed under his leadership has received the 2010 Frost & Sullivan New Product Innovation Award. He served as the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON CLOUD COMPUTING. He is currently serving as the Co-Editor-in-Chief of the *Journal of Software: Practice and Experience*, which was established over 45 years ago. Please visit his cyberhome: www.buyya.com for additional information.