Resource and Service Management for Fog Infrastructure as a Service

by

Shehenaz Shaik

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 03, 2019

Keywords: Internet of Things, Fog computing, Cloud computing, Edge computing, Service management, Smart city

Approved by

Sanjeev Baskiyar, Chair, Professor of Computer Science and Software Engineering
Adit Singh, Professor of Electrical and Computer Engineering
Xiao Qin, Professor of Computer Science and Software Engineering
Anthony Skjellum, Professor of Computer Science and Software Engineering

Abstract

Cloud computing stops short in its offerings towards deployment of latency-critical and bandwidth-intensive applications. Fog computing is emerging as complementary to cloud computing in realizing the deployment of large-scale Internet of Things (IoT) environments for such applications. It supports geographically dispersed IoT devices and users leveraging compute nodes of varied resource capacities in vicinity of the devices/users. Researchers have demonstrated the necessity of fog computing towards deployment of latency-critical and bandwidth-intensive applications. Fog computing infrastructure is recommended for hosting the new generation application environments such as autonomous vehicles, emergency services and personalized healthcare. The Fog Infrastructure as a Service (FIaaS) model facilitates leasing of shared infrastructure resources such as compute, network, and storage on fog nodes which are deployed by Fog Service Providers.

Several research problems must to be solved prior to the real-world deployment of large-scale fog computing environments. This research identified and attempted to solve some of them in the FIaaS model. It has developed an event-driven simulator, PFogSim, to facilitate the simulation of large-scale, dynamic, and mobile fog computing environments comprising thousands of devices with defined multi-layered hierarchical structure representing smart city deployments using IoTs. Towards efficient management of fog, this research proposes a Hierarchical and Autonomous Fog Architecture (HAFA) to organize heterogeneous fog nodes into multi-layered connected hierarchy based on several parameters such as location, distance

from IoT devices and/or users, resource configuration, privacy and security. It groups fog nodes to facilitate resource pooling and local control, and logically links such groups to facilitate disaster readiness and autonomy. Leveraging HAFA, this research proposes and tests a distributed approach to select a cost-efficient fog node to host any given application service among prospective fog nodes with available infrastructure resources considering both computation and communication costs. To cater to the unique needs of the fog environment, it also develops a novel location and network-aware approach for service pricing in FIaaS environments.

Dedication

I lovingly dedicate this dissertation to my father, *Mr. Shaik Mehaboob Vali*, and my mother, *Mrs. Shaik Meharunnisa*.

Acknowledgements

Thank you almighty for the immeasurable blessings.

I gratefully acknowledge that this dissertation was made possible due to the support of Department of Computer Science and Software Engineering at Samuel Ginn College of Engineering, Auburn University and my supervisor Dr. Sanjeev Baskiyar. This research was supported in part by NSF award nos. 0966278 and 1659845. Furthermore, I thank the committee members Dr. Anthony Skjellum, Dr. Xiao Qin, Dr. Adit Singh, and Dr. Allan David for the feedback provided during various discussions and dissertation review.

The role of REU student coordinator gave me an opportunity to work with smart, enthusiastic, and knowledgeable undergraduate students from different disciplines. Thank you Avraham Rynderman, Jessica Knezha, William McCarthy, Clayton Johnson, Jacob Hall, and Qian Wang for your contributions towards development of PFogSim simulator. Working with each of you has been a true pleasure, an experience I will cherish for years to come.

I thank my father, Mr. Shaik Mehaboob Vali, my mother, Mrs. Shaik Meharunnisa and my sister, Shaik Sheeraz, who have been my pillar of strength not just during the past five years into the program, but all through my life. You are the reason for me to embark on this journey of research. I can't thank you enough for your endless support, guidance, and patience.

My daughter, Asmin Shaik, who was a three-year old when I started the program, who lived away from me for several years and still holds the heart to give me back the role of her mom the moment she saw me in the airport, who pulled several all-nighters to give me company while I work on this dissertation, who tried to contribute to the extent of her abilities, who has been patiently waiting for me to complete this program so that we can see the sunlight from outside the lab room, and whose love for me is unconditional. I do not have words to thank her enough. I am looking forward to return her the favor, the day she works on hers, hopefully.

I have deep regards for all of you.

Table of Contents

List of Tables

List of Figures

Abbreviations

Table 1: Abbreviations

Abbreviation	Explanation
5G	Fifth Generation Cellular Networks
AR	Augmented Reality
BAN	Body Area Networks
CA	Cognitive Assistance
CDN	Content Delivery Network
CIoT	Cognitive Internet of Things
CPE	Customer Premises Equipment
CSV	Comma Separated Values
DC	Data Center
EEG-BCI	Electro Encephalography Brain Computer Interface
ESB	Equal Share Bandwidth
ESM	Edge Server manager
FIaaS	Fog Infrastructure as a Service
FSDN	Fog based SDN
FSP	Fog Service Provider
FT	Fog Tenant
GPS	Global Positioning System
HAFA	Hierarchical and Autonomous Fog Architecture
HC	Healthcare
HIPAA	Health Insurance Portability and Accountability Act of 1996
IaaS	Infrastructure as a Service

ICN	Information Centric Network
IoNT	Internet of Nano Things
IoT	Internet of Things
ITS	Intelligent Transportation System
LAN	Local Area Network
M2M	Machine to Machine
MEC	Mobile Edge Computing
ML	Machine Learning
MMC	Mobile Micro Cloud
MMOG	Massively Multi-Player Online Gaming
NIST	National Institute of Standards and Technology
P2P	Peer to Peer
PaaS	Platform as a Service
PTZ	Pan-Tilt-Zoom
QoE	Quality of Experience
QoS	Quality of Service
RSC	Road Side Cloud
RSU	Road Side Unit
SaaS	Software as a Service
SDN	Software Defined Network
SIoT	Social Internet of Things
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SP	Service Provider
STLS	Smart Traffic Light System
VM	Virtual Machine
WAN	Wide Area Network

WAP	Wireless Access Point
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network

Terminology

Table 2: Terminology

Term	Explanation
Application	A collection of interdependent application services which collectively perform a complex task.
Fog Broker	The entity which executes the functionality as defined by orchestrator policy to identify the fog node.
Host	A physical or virtual machine with sufficient compute, memory, disk, and network resources to store a copy of application service locally or has access to it over a local network and execute the application service instance for provided input.
Orchestrator	Orchestrator defines the policy to select a specific node from the set of fog nodes with available resources.
Resource	Infrastructure resource(s) refer to compute, memory, storage, and/or network resources.
Service	An autonomous application component which can perform specific operation on given input and generate required output.

1. Introduction

Internet of Things (IoT), a term coined by Kevin Ashton in 1999, refers to interconnection of physical objects such as vehicles, household devices, buildings, machinery, etc. which are embedded with sensors, software and have the capability to connect to other such objects or standard computing / networking elements. IoT paradigm allows communication with physical objects and among the objects themselves and facilitates the development of innovative applications to help improve the quality of our lives.

By 2020, the number of connected IoT devices worldwide is expected to be more than 50 billion. These IoT devices continually sense their surroundings and generate data either at regular intervals or upon occurrence of an event of interest to the subscriber. These data items need to be processed to extract information and thereby knowledge, prior to making it available to the application users.

1.1. New generation applications

In this section, we introduce some of the emerging new generation applications, along with their data characteristics and resource requirements.

1.1.1. Connected home

Lately, several devices at home are beginning to be connected to internet e.g. phone, kids' watches, multiple video cameras monitoring all rooms in house, connected appliances such as washing machines, light switches, kitchen appliances, A/C thermostat, television for Netflix pre-downloads, baby/pet care and monitoring devices, robotic vacuum cleaners, automatic chef cooking devices, and many more.

Varied nature of devices generate different types of data and their corresponding applications have different resource requirements. Continuous monitoring results in lot of raw data generated at high velocity. The data generated by these devices is usually of personal interest and need to be processed and managed in an efficient manner as the number of devices will only grow in future.

To ensure safety, and security, timely processing of sensed data and generated events as well as quick response is critical. The devices are usually connected over home Wi-Fi network, thus relieving bandwidth constraints in system, provided the application services are hosted by devices within the same network. Appropriate supporting infrastructure to host IoT applications and data can be made available in the form of IoT gateways, Wi-Fi routers, home storage, etc.

Integration with other smart systems is required e.g. voice recognition systems, face recognition systems, surveillance system, smart cities, etc. Cloud integration is recommended for disaster scenarios, backup etc.

Connected home environments are individual, less dynamic, and comparatively small scale deployments. Additionally, lack of relevant technical expertise at home makes it more cost-efficient to leverage third party services towards their maintenance, as well as SaaS based application services to manage and access the application data.

1.1.2. Smart city

These are large scale city wide IoT environments supporting various application domains for large number of users as well as massive number of IoT devices. Applications supported include local traffic monitoring, local weather monitoring, timely updates on disaster events such as tornadoes, etc., accident or rerouting traffic events, vehicle tracking, culprit tracking, city wide surveillance, roadside assistance, lane change assistance, energy management, traffic control at signals, etc. Infrastructure needs to be provided to offer application services with required QoS characteristics.

The system is highly dynamic as the users and devices may be mobile, and may join/leave the system at will, especially during events such as game days, concerts, disaster scenarios, etc. Cloud integration is required to support overload scenarios as well as long term archival services.

Owing to the scale of infrastructure required to support the large system, smart city deployments can have exclusive technical personnel to maintain the local system components.

1.1.3. Smart healthcare

Smart healthcare environments include Body Area Networks (BAN), Internet of Nano Things (IoNT), assisted living, on-body devices such as pacemakers, insulin injectors, etc. These devices generate huge amount of data due to continuous polling of a lot of environmental and patient parameters from several devices.

The corresponding applications usually have strict latency requirements and any non-compliance may result in loss of life at times. Thus timely processing of raw data and application response are critical for smart healthcare applications, which should be ensured by infrastructure hosting the applications and data.

As the sensing devices move with the person being monitored, mobility support is required for applications and corresponding data is expected to be available for applications to make knowledgeable decisions.

For advanced monitoring, intelligent decision making and appropriate timely action, integration with other smart environments is usually required. Integration with cloud is mandatory for data / applications to be accessed from different geographically distant locations.

Smart healthcare applications are usually accessed by/for individuals and their associated institutions and care takers, who are likely not tech-savvy. Additionally, as applications are usually standard ones, they can be accessed in the form of SaaS based services, which avoids the need for deployment and management of any personal hardware and applications.

Scale of the deployment depends on demand for a given application, which is variable at a given location due to mobility of users. Several other factors such as access control and security are also critical due to HIPAA etc. regulations.

1.2. New generation applications characteristics

Based on the type of application, data generated by IoT environments has specific characteristics such as:

- **Volume:** Large volume of data, from large number of sensors. Cloud storage systems may not be able to process the large volume of small data write operations in an efficient manner.
- **Small data:** Large numbers of small sized data, e.g. environmental sensor data. Very small data, as generated by small size, low power sensors. From the perspective of network, it is inefficient to transfer small data generated by individual sensors to be transferred directly to the cloud.
- **Variety:** Multiple data types due to various types of sensors.
- **Data sources:** Massive number of data sources.
- **Velocity:** Data generated at high velocity.
- **Structure:** Mostly unstructured data such as pictures, videos. etc.
- **Variability:** Continual data generation at ad hoc or regular intervals.
- **Value:** Transient data, or of long term value.
- **QoS needs:** Real time processing requirements. For latency-sensitive data generated by sensors, IoT applications are expected to process the data and send controls to actuators in real-time e.g. Smart Traffic Light Systems, Vehicular networks, Wireless Sensor and Actuator Networks, etc.
- **Geo-distribution:** Massively dispersed data sources at the edge.
- **Veracity:** Redundant data, as sensors are deployed such that the events are captured in a redundant manner to ensure reliability and robustness of system and to account for any sensor failures. Cloud compute, network and storage resources can be leveraged in an optimal manner by removing the redundant data generated by the sensors near to the source, rather than in cloud.

- **Location-awareness:** Location-dependent data.

Authors in [1] listed various characteristics of IoT data such as polymorphism, heterogeneity, largeness in quantity, latency sensitivity. Dey et. al. [2] discussed workload types of various IoT applications and compared in terms of the data sizes involved, arrival rate of the jobs and computation needs. Ali and Elkheir [3] discussed IoT data life cycle, energy bottlenecks in IoT data management and various storage-centric and communication-centric approaches for green data management for IoT. Ma, Wang, and Chu [4] listed characteristics of IoT data, proposed a layered reference model for management of IoT data and discussed open research challenges, along with a survey of middleware systems for IoT data management.

Bandyopadhyay and Sen [5] proposed a layered architecture for IoT and discussed key technologies and application domains for IoT. Miorandi et. al. [6] discussed IoT system features, research challenges and applications. Stankovic [7] presented a vision of future world with IoT devices and services and discussed various research challenges towards realization of IoT. Atzori, Iera, and Morabito [8] described IoT paradigm as things oriented vision, internet oriented vision, and semantic oriented vision and discussed enabling technologies.

Wu et. al. [9] proposed a new network paradigm, Cognitive Internet of Things (CIoT), which extends IoT with cognitive capabilities i.e. the objects learn, think and understand the physical and social worlds by themselves. Ortiz et. al. [10] proposed a generic architecture for Social Internet of Things (SIoT), which is defined as the cluster between Internet of Things and social networks. Yue et. al. [11] proposed an architecture for future Internet based on ICN, referred as DataClouds, to support data-centric services of IoT. Gubbi et. al. [12] presented a cloud centric vision for a large scale and worldwide implementation of IoT.

1.3. Insufficiency of Cloud Computing

In current world of IT, users are accessing their data and applications deployed on cloud from their mobile devices. The performance of applications and data access depends on the bandwidth and QoS characteristics of network providing the access to cloud. The problem of low performance is felt more during overload scenarios such as public events, large gatherings, etc. This problem is aggravated by the introduction of Internet of Things paradigm and deployment of IoT supporting applications in real world. The number of devices connect to the internet are expected to reach 50 billion by the year 2020. The sheer volume of data generated by these communicating devices will impact the cloud-hosted application performance.

The data generated by IoT environment needs to be made available to cloud-based applications. But the characteristics of IoT data listed earlier restrain the transfer of sensor data in its entirety to a centralized cloud in a timely manner due to network bandwidth limitations and the six Vs of IoT data – Volume, Velocity, Variety, Veracity, Variability, and Value. In addition, for most of the IoT applications, it might be unnecessary to transfer all the sensor data to cloud. The long distances between IoT devices and centralized cloud may result in variable end-to-end network delay, thus rendering it infeasible to deliver real-time and latency sensitive IoT applications such as those in healthcare, actuation, etc.

Irrespective of the futuristic increase in network bandwidth, it will always fall short for the application needs as, with IoT, data is being generated at the rates faster than it can be consumed by the cloud or even transferred over the core network to the cloud. Hence to support the application SLAs, it is preferable to move application instances to the edge of network rather than moving all the generated data to the central cloud to be processed. Instead, edge nodes can process the data and only the output / results can be uploaded to cloud for long term storage. In

addition, raw data can be uploaded to cloud for further global big data processing. But this data transfer can accommodate higher data transfer latencies and processing on cloud can be done in background without impact to user applications.

2. Fog Computing

In this chapter, we introduce the concept of fog computing, and provide a brief comparison of the same with other distributed computing approaches.

2.1. Introduction

Cisco proposed [13] the concept of fog computing to realize deployment of large scale IoT environments and low latency real-time services, leveraging large number of resource-constrained, heterogeneous fog nodes which are distributed across vast geographical areas and located closer to the users and data sources, as compared to core cloud which is usually located at centralized data centers, far away from users and data sources. Sometimes, IoT devices, such as cameras, themselves may have sufficient free resources available and hence can serve as lower level fog nodes to offer IoT services with very low latency, and might relieve network bandwidth requirements along with compute and storage resource needs in core cloud.

Fog is a cloud-to-thing continuum where WSN, IoT devices, edge nodes of different sizes, distributed grids and MEC (Mobile Edge Cloud), centralized clouds, etc. may all be considered as fog nodes, among others. Thus, application services float around while user application state is dynamically deployed / migrated as needed based on several pre-defined criteria with no user-intervention or knowledge. Applications and data are expected to be always available irrespective of user location, network connectivity, and availability of other fog resources.

2.1.1. Definition

Several definitions are available for fog computing in literature, some of which are presented below.

Yi et. al. [14] have defined fog computing as follows: “Fog computing is a geographically distributed computing architecture with a resource pool consists of one or more ubiquitously connected heterogeneous devices (including edge devices) at the edge of network and not exclusively seamlessly backed by cloud services, to collaboratively provide elastic computation, storage and communication (and many other new services and tasks) in isolated environments to a large scale of clients in proximity.”

Vaquero and Merino [15] have defined fog computing as follows: “Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so.”

NIST [16] [17] [18] defined it as “Fog computing is a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centers. This paradigm supports vertically-isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity.”

OpenFog Consortium Architecture Working Group [19] defined it in reference architecture for fog computing as “A horizontal, system-level architecture that distributes

computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum."

2.1.2. Characteristics

Fog computing extends the cloud computing paradigm to the edge of networks especially wireless networks. Unique characteristics of fog computing environments are as listed below:

- Fog nodes are located away from main cloud data centers.
- Fog nodes are widespread and geographically available in large numbers.
- Completely distributed system.
- Multi-layered hierarchy.
- Fog applications are fully distributed over fog nodes.
- Offers low and predictable latency to applications.
- Applications are provided with awareness of device geographical location and device context.
- Fog nodes cope with mobility of devices.
- Fog nodes offer special services that may only be required in IoT context e.g. translation between IP- and non-IP transport.
- Typically accessed by devices over wireless networks.
- Lots of data processed locally to avoid data transfer to cloud.
- Processing latency sensitive data and actuation.
- Interoperation with cloud.
- Overflow to cloud or at times of fog unavailability.

2.1.3. Fog node

Queries from devices with response time > 100 millisecond are best served by cloud. Those with < 100 millisecond are best served by fog. Small fog nodes usually are embedded devices with resource configurations:

- Memory, usually hundreds of MBs of RAM
- Storage, usually a few GBs
- Compute, 1 or 2 different cores

Fog nodes vary in several ways as listed below:

- Physically, fog nodes may have variable type and configuration of resources such as compute (number of cores, speed), memory (speed, capacity), storage (type, capacity, speed), network (protocol, latency, bandwidth), etc.
- Have device-specific features such as cameras, user interfaces, scientific / application-specific features, etc.
- Static or mobile, w.r.t. data sources and service consumers.
- Geographically widely dispersed with determinable physical location.
- Nodes with standard resource configurations deployed by organizations / service providers, or voluntary nodes with non-standard resource configurations.
- Resource-constrained and usually with lesser resources as compared to cloud servers.
- Predominantly wireless access and intermittent network connectivity.
- Ad hoc, dynamically formed network, due to mobility.
- Less reliable nodes due to mobility, and presence of voluntary nodes.

- Dynamic environments as fog nodes join and leave arbitrarily attributing to mobility or limited power.

Fog nodes can be stationary or mobile relative to the IoT devices (or data sources), listed below are some such environments:

- Stationary fog nodes and devices
    - e.g. physical monitoring of buildings, etc., face recognition.
- Stationary fog nodes and mobile devices
    - E.g. street lights as fog nodes
    - E.g. cars as devices
- Mobile fog nodes and stationary devices (relative to fog node)
    - E.g. train as fog nodes
    - E.g. commuters as data sources
- Mobile fog nodes and mobile devices
    - E.g. VCC leveraging vehicles as fog nodes
    - E.g. vehicles as sensors
    - E.g. application - dissemination of road blockage info

Examples of fog node are IoT device, mobile device, IoT gateway, home Wi-Fi router, small node at bus stop, street light, vehicle on the move, parked car, RSU, cellular base station, surveillance camera, wireless access point, network router, server machine, etc.

2.2. Comparison of various computing approaches

Table 3 provides a comparison of various computing approaches with fog computing in terms of resource and application characteristics.

Table 3: Comparison of various computing approaches

Feature	Cluster Computing	Grid Computing	Cloud Computing	Ad-Hoc Networks	Wireless Sensor Networks	Fog Computing
Application characteristics						
Deployment purpose / Killer Application	High Availability and Load balancing of services	Scientific applications	Web based applications	Ad-hoc connectivity	Sensing environment	Internet of Things
Compute power	Compute intensive	Compute intensive	Compute intensive	N/A	Very small compute resources required	Low / Medium / Large
Latency sensitivity	Latency-sensitive e.g. databases, financial applications	Latency-tolerant	Latency-tolerant (due to remote user access over WAN)	N/A	Latency-sensitive, due to continuous data generation & limited storage resources	Latency-critical applications e.g. healthcare, vehicular applications, etc.
Network throughput	Normal	Normal	Normal	Normal	Data intensive	Data intensive
Data location	Co-located with compute nodes	Co-located with compute nodes	Co-located with compute nodes	Co-located with compute nodes	Sink node, co-located with sensor nodes	Distributed over large geographic areas
Application / Service deployment	Static	Static	Dynamic (based on Service Oriented Architecture)	N/A	Limited support to run applications, e.g. preprocessing of data to clean up, etc.	Dynamic, due to node churn and change in workload due to user mobility.
Deployment characteristics						
Number of nodes	10s of nodes	100s of nodes	1000s of nodes	10s of nodes	Large number of nodes	Very large number of nodes
Node configurations	Homogeneous	Homogeneous	Homogeneous	Heterogeneous	Homogeneous	Heterogeneous
Node resources	High	High	High	Medium	Very low	Low / Medium / High
Inter-node connectivity	High speed local network	High speed local network	Data Center LAN	Wireless LAN	Wireless LAN	Wired / Wireless LAN / WAN
Deployment area	Co-located nodes	Co-located nodes	Co-located nodes in data center	Co-located nodes	Geographically dispersed nodes	Geographically dispersed nodes
Power availability to nodes	Continuous	Continuous	Continuous	Battery-powered (Usually large)	Battery or solar powered (Usually small)	Continuous / Battery powered
Life of nodes	Forever	Forever	Forever	Short lived	Short lived	Dynamic node join / leave
Mobility of nodes	Not supported	Not supported	Not supported	Not supported	Limited support (Move collectively, as a system)	Supported feature

Reliability of nodes	Highly reliable	Highly reliable	Highly reliable	Unreliable	Unreliable	Mix of reliable / unreliable nodes
Inter-Node latency	Negligible	Negligible	Negligible	Negligible	Negligible	Significant (Due to geographic dispersion)
Deployment count over a geographic area	Sparse	Sparse	Sparse	Sparse	Dense	Dense
Access to Cloud or external node (for computation purpose)	Direct connectivity over LAN	Direct connectivity over LAN	Direct connectivity over LAN	Direct connectivity over LAN	Through sink node only	Direct connectivity over LAN / WAN
Node hierarchy	Flat structure	Flat structure	Flat structure	Flat structure	Limited hierarchy (Sensor node / Sink node)	Multi-layered hierarchy
Managed environment	Managed	Managed	Managed	Unmanaged	Unmanaged	Mix of Managed & Unmanaged nodes
Cloud-specific features	Not supported	Not supported	Multi-tenancy, Self-Service, Pay-per-use, SOA, Web-based access to services	N/A	Not supported	All are supported.
Location awareness	No	No	No	N/A	Yes	Yes
Real-time interactions with devices and users	Not supported	Not supported	Limited support (due to WAN latency & large distance)	N/A	Not supported	Supported
Available for general usage	No. Application specific deployment	No.	Yes	N/A.	No.	Yes

2.3. Fog service models

In a service-based computing model, multiple users share physical or virtual resources to concurrently execute their applications with no knowledge of the workload running on the system by other users. Resources and applications can be offered in fog as services similar to that in cloud.

2.4. Fog Infrastructure as a Service (FIaaS)

With Fog Infrastructure as a Service (FIaaS) model, shared physical or virtual infrastructure resources such as compute, storage, and network are leveraged by multiple concurrent users to deploy and execute arbitrary applications. FIaaS offers benefits for tenants similar to those from IaaS in cloud such as on-demand access, self-service, pay-as-you-go, resource pooling, multi tenancy, rapid elasticity, and measured service.

2.4.1. FIaaS Entities

FIaaS service model involves three primary actors–fog service provider, fog tenant, and end user. Fog service provider may be a city municipality, telecom services company, educational institution, non-profit organization offering service to its members, or a web scale company offering physical or virtual IT infrastructure resources for shared use. Fog tenants lease these resources offered by fog service provider in a self-service, pay-as-you-go manner to deploy various application environments. End users access applications deployed by tenants on leased infrastructure resources.

2.5. Case study: Smart city fog infrastructure

In a prospective Smart City environment, owing to the widely varied resource requirements for deployment of IoT, 5G, vehicular, etc. new generation applications, different types of fog nodes need to be deployed across the breadth of the city, details of which are discussed below.

Miniature fog nodes–with very low compute, storage, and network resources, these are deployed in dense manner. At times, these may be co-located with IoT devices themselves and may generate data. These may be stationary or mobile, may have limited energy resources, usually accessed over wireless or cellular network, and may join or leave the network intermittently. Thus, they are less reliable and have low availability. Located closest to data

sources and users, these can be used for quick, short computation or data transfer and cannot be relied upon to store data for future access due to low storage resources.

Small fog nodes–with low resource configurations, these nodes offer continuous availability due to co-location or being relatively static to IoT devices / users, and are usually accessible over wireless networks. Due to higher availability, these nodes can be used for preprocessing of data and simple computing, but they cannot be used to store data for longer durations.

Community fog nodes–with medium resource configurations, these are usually shared by multiple entities in neighborhood to support different types of applications. These nodes are setup individually in an unmanaged environment, are mostly stationary, and can be leveraged to perform basic analytics. Execution of large scale analytics is not possible due to lack of historical data, as data cannot be stored for long term on these nodes due to limited resources.

Edge Clouds–are small clusters of co-located or relatively static fog nodes formed in an ad-hoc manner upon need, or placed in a managed environment. These nodes have higher availability resulting from being stationary, in a physically controlled environment, and/or cluster configurations. These fog nodes have resource configurations sufficient to support local analytics and storage of data for a short time duration.

Micro Data Centers–are service provider managed mini clouds with ample resources i.e. nodes placed in a managed and controlled physical environment, resulting in higher reliability of these nodes. They retain control of application and data ownership, and hence recommended for security and privacy reasons when applications / data are required not to leave premises of an organization or campus, or when they have only local significance. These are higher in number

and located at shorter distances to users as compared to large data centers; thus offering less variable, lower access latency on average, and are usually costlier as compared to cloud resources.

Infrastructure Cloud–consists of huge number of nodes placed together in one or more large data centers. These are service provider managed, globally accessible, very large environments with logically unlimited and infinitely scalable resources available for large scale analytics, compute- and data-intensive operations, and long term storage and archival of any amount of data. Managed data center environment provides high reliability and availability to these nodes.

Due to the scale of co-located deployment, cloud resources are usually cheaper as compared to any other type of fog node, but increase security and privacy risks as applications are hosted on public nodes and application data travels over large network distances. As only a handful of these are setup across the globe by an organization, access latency is large on average, depending on user location, as well as highly variable due to large network distance and access over multiple networks.

The discussion in this section showcases the presence of heterogeneity in fog environments. Figure 1 shows various types of prospective fog nodes in smart city environment.

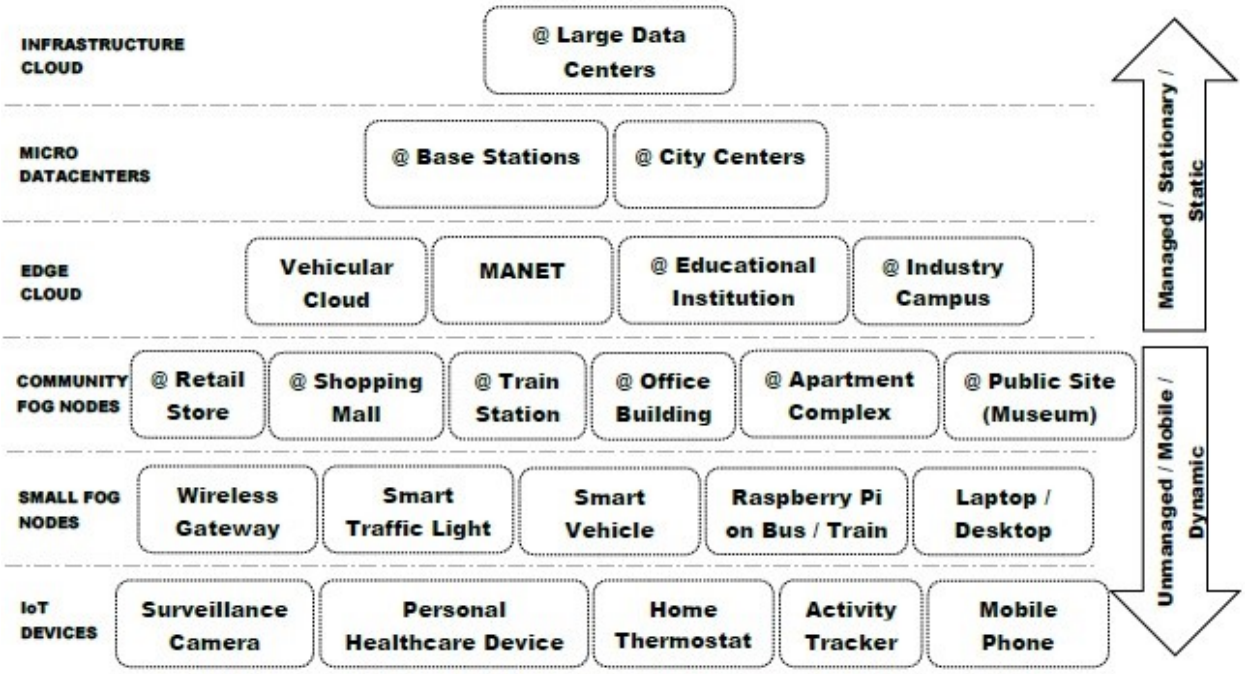

Figure 1: Various types of prospective fog nodes in smart city environment.

2.6. Case study: Fog-based applications for smart city

In a Smart City based fog environment [20], infrastructure resources can be offered by city municipal authorities in FIaaS service model to its tenants. Thus, the same set of fog nodes can be used by various tenants who offer IoT-based application services to users. In this section, we provide some examples of IoT applications which can be hosted on fog infrastructure resources leased by tenants.

Public services: In addition to services [21] which generate uniform workload such as city-wide video surveillance [22] [23], monitoring citizen activities, waste management, utilities management [24], environmental quality control e.g. water/air pollution and noise level containment, etc., city municipality offers several services such as emergency services [25] e.g. fire, accident, natural or manmade disasters, military attacks, etc., crowded event management, etc. which generate variable workload and can best leverage FIaaS infrastructure resources.

Smart grid: Deployment of Smart grid environment requires applications towards energy management [26] [24], continuous monitoring, metering, switching to alternate sources of power, pipeline management, etc. Workload generated by these applications is usually uniform and does not change over time.

Industrial IoT: IoT applications in industrial environments [27] are latency-critical closed loop operations which are best served by on premise private fog nodes deployed on campus. Workload is uniform due to standard applications and data generation at regular intervals. Any industry-specific applications which are latency-tolerant and need large amount of resources at arbitrary times may leverage FIaaS resources.

Smart home: Smart connected homes [28] comprise several smart appliances and their corresponding applications. The corresponding data and applications are placed partially on premise and private fog infrastructure, while community fog infrastructure can be leveraged for local data storage and hosting shared applications, and cloud infrastructure can be used for archival of data.

Personal healthcare: Personal healthcare [29] is being revolutionized by IoT devices and corresponding applications at various facilities such as smart hospitals and care centers for elderly and disabled. Owing to their pervasive presence and strict QoS requirements of applications, small fog devices can be best leveraged to host these applications. As the need for resources is highly dependent on the location of users, their applications, and mobility, the generated workload is highly variable and is difficult to predict.

3. Background

3.1. Interest from Academia

Research on fog computing paradigm, platform and IoT applications leveraging them has gained wide interest in recent times.

Bonomi et. al. [13] have introduced the concept of fog computing and discussed the interplay of cloud and fog computing. Shi et. al. [30] have introduced the concept of fog computing and listed its characteristics. Vaquero and Merino [15] have provided a comprehensive definition of fog computing considering various factors such as device ubiquity, network management, privacy, edge clouds, scale of connectivity, etc. Yi et. al. [31] have provided a definition of fog computing and discussed in detail various open issues with fog computing paradigm such as fog networking, quality of service, programming and interfacing, computation offloading, provisioning and resource management, etc. Krishnan, Bhagwat, and Utpat [32] have proposed a method of moving the computation from cloud to network by introducing an android like app store on networking devices e.g. IOX routers, which allows the user to choose the location for processing of a data packet, cloud or fog node, based on the packet tag.

Stojmenovic [33] discussed various scenarios and applications where fog computing can be leveraged, e.g. vehicular networks, smart grids, wireless sensor and actuator networks, etc. and have provided a brief introduction to various problems solved towards the same and open issues. Peter [34] provided a brief introduction to Cisco IoX architecture, listed characteristics of fog computing and various application domains of fog computing. Roca, Tous, and Milito [35] introduced the concept of small data, which needs to be processed in fog, an extension of cloud to edge of network and compared it with big data, which needs to be processed in cloud.

Firdhous, Ghazali, and Hassan [36] provided a comparison of cloud computing and fog computing paradigms w.r.t. various factors such as support for mobility, location awareness, real time applications, security, approximate latency, etc. Yannuzzi et. al. [37] have discussed mobility, reliability and scalability aspects for IoT environment to show the insufficiency of cloud compute and storage models for IoT deployment and provided reasons for why fog computing is a good fit for IoT considering various factors such as mobility of devices and virtual containers, addressing, handover, live migration, reliable control and actuation, real time operation, data aggregation and analytics, etc. Luan et. al. [38] have compared fog computing and cloud computing characteristics. Fog storage capability is compared with content delivery network (CDN), information centric network (ICN) and proactive caching framework. Fog compute capability is compared with cloudlets, cloud computing, transparent computing. Fog communication capability is compared with traditional Wi-Fi access points, Femto cell networks.

Bonomi et. al. [39] have proposed hierarchical distributed architecture for fog computing, considering massively distributed number of sources at the edge. Yi et. al. [14] have proposed a platform for fog computing and discussed its layers and components in brief. Datta, Bonnet, and Haerri [40] have proposed oneM2M architecture for connected vehicles with Road Side Units

(RSUs) and M2M gateways leveraging fog computing platform. OpenFog Consortium Architecture Working Group [41] proposed OpenFog architecture and fog infrastructure building blocks. Bittencourt et. al. [42] have proposed a fog computing layered architecture and provided a detailed description of various layers, components at each layer and their functionalities with a focus on VM migration. Al-Fuqaha et. al. [43] discussed some of the IoT architectures proposed, tasks performed at each of the layers and the interplay of IoT applications, Big Data, cloud and fog computing paradigms. Sarkar, Chatterjee, and Misra [44] have provided mathematical modeling of cloud and fog computing environments. By simulations, they verified the applicability of fog computing in the context of IoT.

3.1.1. PoC implementations of applications in fog

Healthcare: Gia et. al. [45] leveraged fog node in the implementation of Wireless Body Area Networks as a gateway between sensors and remote server on cloud, to offer localized services such as distributed database management, ECG feature extraction, user GUI with access management, and real-time push notifications in emergency scenarios. Zao et. al. [46] [47] developed a pervasive online Electro Encephalography (EEG) – Brain Computer Interface (BCI) system leveraging multi-tier fog and cloud computing technologies. Ahnn and Potkonjak [48] proposed a distributed and energy-saving mobile health platform, mHealthmon, where mobile users publish and access sensor data using a cloud computing based distributed P2P overlay network. Stantchev et. al. [49] presented a three-level architecture for smart healthcare infrastructure leveraging smart items, fog computing and cloud computing based on service oriented architecture (SOA). Shi et. al. [30] mentioned the applicability of fog computing in healthcare and Body Area Network (BAN) applications. Al-Fuqaha et. al. [43] implemented three IoT application use cases – nursing home patient monitoring system, monitoring and mitigation of eating disorders, in-door navigation system for the blind and visually impaired, and

two IoT service analytic use cases – efficient estimation of the number of unique IP addresses using a given service, and tracking the frequency of service usage by a given IP.

Emergency: Leveraging the concept of fog computing, Aazam and Huh [25] proposed architecture for Emergency Help Alert Mobile Cloud (E-HAMC), a smart phone based latency sensitive service, which provides automatic notifications to relevant emergency dealing departments.

Smart vehicles: Whaiduzzaman et. al. [50] discussed various applications of vehicular cloud computing such as an airport as a datacenter, parking lot data cloud, shopping mall data center, dynamic traffic light management, optimizing traffic signals, self-organized high occupancy vehicle (HOV) lanes, managing evacuation, road safety message, easing frequent congestion, managing parking facilities, vehicular clouds in developing countries perspective etc. Bonomi et. al. [13] discussed the fog applications – connected vehicles, wireless sensor and actuator networks. Hong et. al. [51] implemented two fog applications using the proposed Mobile Fog programming model – vehicle tracking using cameras, and traffic monitoring using mobility-driven distributed complex event processing (MCEP) system. Lee et. al. [52] listed several applications of VANET paradigm such as dangerous road warning, car accident warning, work zone warning, emergency vehicle warning, highway information, road congestion region information, traffic navigation map, commercial advertisement, multimedia file sharing. Kim et. al. [53] proposed a shared parking model (RFPARK) based on fog computing and Road Side Cloud (RSC) concepts by formulating the problem as a many-to-one matching game between vehicles and parking lots. They proposed parking slots association algorithm for finding ideal parking slot and verified using simulations.

Assisted living: Cao, Hou, and Chen [29] showcased a real-world pervasive health monitoring application, pervasive fall detection for stroke mitigation, to demonstrate effectiveness and efficacy of fog computing paradigm in health monitoring. Ha et. al. [54] proposed the architecture and prototype implementation of an assistive system, Gabriel, based on Google Glass devices and Cloudlets, for users in cognitive decline. Li et. al. [28] proposed EHOPES, a smart living platform based on fog computing paradigm with the components – smart energy, smart health, smart office, smart protection, smart entertainment, and smart surroundings.

Smart environment: Giang et. al. [55] designed and evaluated a distributed dataflow application framework, referred as Distributed Node Red (D-NR) by building an IoT application in Smart Environment domain. Miorandi et. al. [6] discussed IoT applications in domains – Smart home / smart buildings, Smart cities, Environment monitoring, Healthcare, Smart business / inventory and product management, Security and surveillance. Stojmenovic [33] discussed the applicability of fog computing for various applications such as micro-grid decentralized smart grid, smarter traffic lights and connected vehicles, wireless sensor and actuator networks in self-maintaining trains, decentralized smart building control, vehicular networks etc. Luan et. al. [38] discussed the fog applications – integrated localized information system in shopping center, localized travel services in parkland, on-board video streaming, gaming and social networking services on inter-state bus, vehicular fog computing networks, etc. Bonomi et. al. [39] discussed two fog applications in detail – Smart traffic light system and wind farm. OpenFog Consortium Architecture Working Group [41] discussed the fog applications – transportation, agriculture, smart cities and buildings. Yi et. al. [14] have discussed fog applications – smart home, smart grid, smart vehicle, health data management. Botta et. al. [56] discussed various IoT applications

such as healthcare, smart city, smart home and smart metering, video surveillance, automotive and smart mobility, smart energy and smart grid. Gubbi et. al. [12] discussed various smart environment application domains - Smart home / office, Smart retail, Smart city, Smart agriculture / forest, Smart water, Smart transportation. Bandyopadhyay and Sen [5] discussed IoT applications in various domains such as aerospace and aviation, automotive, telecommunications, medical and healthcare, independent living, pharmaceutical, retail, logistics and supply chain management, manufacturing, process industry, environment monitoring, transportation, agriculture and breeding, media and entertainment, insurance, recycling, etc. Atzori, Iera, and Morabito [8] discussed IoT applications: transportation and logistics domain – logistics, assisted driving, mobile ticketing, monitoring environmental parameters, augmented maps; healthcare domain – tracking, identification and authentication, data collection, sensing; smart environments domain – comfortable homes and offices, industrial plants, smart museum and gym; personal and social domain – social networking, historical queries, losses, thefts; futuristic applications domain – robot taxi, city information model, enhanced game room.

Smart home: Igarashi et. al. [57] proposed a hybrid architecture, Programmable Cloud-Enabled Home Controller (PCEHC), where home automation applications are executed both on the home controller element and cloud controller element. Willis, Dasgupta, and Banerjee [58] discussed some applications which can be deployed using proposed ParaDrop platform – motion detection service by security camera vendor, automatic real time home thermostat management, streaming media service, etc. Chang et. al. [59] implemented two applications on proposed edge cloud architecture – 3D indoor localization application and scene activity vector (SAV) based video monitoring and surveillance. Li et. al. [60] implemented a domain mediator for building

management and two applications using the mediator: temperature control and presence based light control.

Smart cities: Jin et. al. [61] presented a framework for the realization of smart cities through IoT, discussed several applications and developed a new approach for noise monitoring and mapping, which helps to understand the noise pollution and city soundscapes together with the impacts on health, well-being and quality of life. Zarko et. al. [62] discussed the use cases – smarter cities and air quality monitoring by crowd sensing, which can be implemented leveraging the proposed Cloud-based Publish/Subscribe for the Internet of Things (CUPUS) middleware platform.

Smart energy: Leveraging fog computing platform, Faruque, and Vatanparvar [24] provided hardware, software and communication architectures for energy-management-as-a-service paradigm.

Edge analytics: Satyanarayanan et. al. [63] discussed various real-world applications leveraging the concept of cloudlets (fog nodes) such as Wearable cognitive assistance, Edge analytics in the internet of things, automotive environments, Mobile access to the legacy PC world, Hostile environments, Tactile Internet, etc. Satyanarayanan et. al. [22] proposed GigaSight, a hybrid cloud architecture that uses a decentralized cloud computing infrastructure in the form of virtual machine (VM) based cloudlets, to perform sampling and denaturing of captured videos and edge analytics, such as indexing, at nearby cloudlet in real time. Khan et.al. [64] discussed various IoT applications such as prediction of natural disasters, industry applications, water scarcity monitoring, design of smart homes, medical applications, agriculture applications, intelligent transport system design, design of smart cities, smart metering and monitoring, smart security, etc. Abdelwahab et.al. [65] discussed various applications of IoT as

follows: remote tracking and monitoring – animal behaviors, environmental condition monitoring, agricultural monitoring, building surveillance and security, healthcare monitoring, smart meter readings, aviation and aerospace safety; real-time resource optimization and control – waste management, smart parking, traffic control, healthcare; smart troubleshooting – aviation and aerospace, automotive, network systems, buildings, smart grids, oil and gas pipelines. Krishnan, Bhagwat, and Utpat [32] implemented an application – time series prediction of temperature based on sensed temperature values.

Tactical environment: Preden et. al. [66] used the Data-to-Decision (D2D) approach, where the data delivered (and the level of sensing performed) by the sensor system behavior depends on the information needs as expressed by the user and discussed the applicability of fog computing paradigm for Intelligence, Surveillance, and Reconnaissance (ISR) System-of-Systems (SoS) system using D2D approach.

Augmented reality: Yi et. al. [31] have discussed the fog applications – Augmented reality and real time video analytics, content delivery and caching, mobile big data analytics. Verbelen et. al. [67] implemented an augmented reality application featuring marker-less tracking, combined with an object recognition algorithm. The components of this application were CPU intensive along with having strict real-time constraints.

Gaming: Lin and Shen [68] proposed a light weight thin-client Massively Multi-player Online Gaming (MMOG) system, referred as CloudFog, to improve Quality of Experience (QoE) for users with online gaming. Pamboris et. al. [69] discussed how the proposed edge cloud platform, NOMAD, can support a first person shooter game with stringent low latency requirements.

Preprocessing: Aazam and Huh [70] have discussed data trimming using fog nodes prior to the transfer of sensed data to cloud for further processing.

Performance: Hassan et. al. [71] leveraged fog computing to improve application performance on mobile devices. Mobile application offloading to fog nodes was tested with two applications: Picaso, an Android face recognition application and DroidSlater, an Andriod Dictionary application. Mobile storage expansion using fog nodes was evaluated using Yahoo WebScope cluster data. Valvag, Johansen, and Kvalnes [72] discussed an image sharing application with decentralized design implemented using proposed decentralized computation platform, Rusta. Lewis et. al. [73] discussed mechanisms for cloudlet provisioning using three computation intensive applications – Face recognition, Speech recognition, and Object recognition. Drolia et. al. [74] evaluated the proposed concept of Ad-hoc mobile edge-cloud by verifying the performance of two MapReduce applications: panoramic image construction, and person finder from a group of images located on multiple devices.

Personalized web access: Zhu et. al. [75] proposed a solution to improve web access performance optimization using fog computing concept by exploiting the knowledge that is only available at the edge servers to improve web page rendering performance.

Networking: Leveraging Fog computing, Truong, Lee, and Doudane [76] presented two use cases for Fog based SDN (FSDN) – a non-safety service e.g. data streaming, and a safety service e.g. lane change assistance.

4. Research Summary

4.1. Introduction

Next generation applications [77] [78] that deliver contextual experiences are becoming the norm. Applications such as smart living, smart parking, autonomous driving, structural health monitoring, smart roads, personalized medicine, and many more, are taking the standard of human life to next level. These applications depend on Internet of Things (IoT) to capture the context and build upon it to deliver the required experience to user.

Latency requirements of IoT applications vary widely depending on the type of application, e.g. control systems in industrial and healthcare domains have strict real-time response needs with millisecond to microsecond latencies whereas data analytical applications require large computation power, high bandwidth, and can tolerate higher latency. Data generated by large scale IoT environments is described by the characteristics of big data–volume, velocity, veracity, variety, variability, and value, collectively referred as 6Vs. As cloud computing paradigm falls short to deploy IoT applications with these requirements, fog computing paradigm is gaining momentum and holds promise for future of IoT, 5G and other new generation applications.

Fog computing paradigm, proposed by Cisco, extends the concept of cloud computing to include the physical nodes of varied resource configurations and capabilities distributed over vast geographical areas to support latency-sensitive and data-intensive applications. In addition, fog environments allow deployment of location-sensitive applications i.e. location-aware, and those of only local value.

4.2. Motivation

- Very large number of data sources (sensors).
- Data sources and users distributed over large geographical areas.
- Data sources generate large volume of data at regular and very short intervals.
- Data generated might have limited lifetime.
- Resource-constrained devices located at edge of network.
- Insufficient network resources to transfer raw data to cloud for centralized processing.
- Internet accessible data sources.

4.3. Significance of problem (Hypothesis)

Internet of Things (IoT) devices, distributed over vast areas, generate big data characterized by 6Vs–Volume, Velocity, Variety, Variability, Veracity, and Value [79]. IoT applications perform complex computations on such data while satisfying stringent QoS requirements [28]. Being resource-constrained and due to the need for consolidation of data from multiple data sources, IoT devices cannot by themselves execute complex services. Hence, fog computing paradigm was proposed by Cisco [13], comprising the idea of deploying fog nodes of variable resource configurations close to data sources to help support applications with low and predictable network latency as well as reduce network bandwidth requirements [41]. Considering

5G infrastructure requirements [80] to support about 1,000,000 devices per sq. km. along with services, users, their mobility, energy-constrained nature of devices, QoS requirements, etc., centralized management approach is almost impractical.

4.4. Service management for FIaaS

To facilitate the deployment of new generation IoT applications, cloud computing environments residing in large data centers are complemented with fog computing environments dispersed over large areas. Along with cloud nodes, fog nodes are leveraged to execute services. To support different types of IoT applications and their workloads, as well as user and sensor mobility, fog nodes of different resource and cost profiles are made available at different geo-locations resulting in varied user-perceived application latencies and data transfer costs. Management of application services deployed on fog nodes is more complex as compared to that in cloud environment for several reasons as discussed below.

Fog comprises heterogeneous nodes of varying compute, memory, network, storage, and energy, etc. resource characteristics to facilitate the execution of latency-critical, bandwidth-intensive, location-sensitive, and localized applications, whereas cloud nodes are considered homogeneous i.e. there is no specific preference of a given node from the set of available cloud nodes in the data center to deploy a given application.

A cloud deployment is a flat structure with connected nodes allowing services to be deployed on any of the nodes with available resources. Contrary to cloud nodes, fog nodes are dispersed over large geographical areas and are identified by their physical location to facilitate the deployment of location-sensitive, latency-critical applications and those having only local value.

In cloud, physical nodes are added or removed from the system infrequently, and the event will be part of a planned maintenance operation by system administrators. On the contrary, fog is a dynamic environment where nodes join/leave at a higher rate due to mobility or battery-outage, leaving the services running on those fog nodes, and thereby users, in a vulnerable state. Hence service management approaches in fog need to be fast and efficient.

Fog nodes are less reliable than cloud nodes due to node mobility, and/or energy limitations. Thereby the need for continuous monitoring of fog environment and redeployment of services is higher than that in cloud, which generally has high reliability. Hence, fog service deployment and migration approaches need to be more efficient.

Cloud applications are usually compute-intensive, whereas fog applications are usually latency-sensitive or data-intensive. With cloud-based applications, data required for computation is usually generated within the data center and is available in its entirety to the compute nodes at negligible latency, as the compute and storage nodes will be connected over high speed data center LAN. Whereas in fog environment, data is generated by IoT devices distributed over very large geographic areas and incur different latencies when accessed from different compute nodes. Thus, service management approaches for fog need to consider geolocation of fog nodes, IoT devices, and users in the system.

User requests are typically served entirely from a single data center, and always from the same primary data center irrespective of the user location, except during special scenarios such as high overload or business continuity during disaster scenarios. Note that these periods are brief and user requests are served from the primary data center upon restoration of normal conditions, whereas such is not the case with dynamic fog environment. Mobile users may need

to be served by different fog nodes depending on their geolocation, thus requiring (re)placement of same application service on different fog nodes.

Number of nodes in fog is of several orders of magnitude more than that in cloud. In addition, wide geographic dispersion of fog nodes make it difficult and sometimes infeasible to maintain entire system state at a centralized authority, thus service management approaches may not have complete system state available and should be developed considering local or incomplete system state, leading to distributed decision making. Service management in cloud environment differs in that all nodes are physically available in the same data center and the cloud broker has complete knowledge of services deployed in the system making centralized solution approaches feasible, whereas it would be cost-prohibitive and unnecessary to maintain such a data store in a large scale distributed fog environment. Thus, service management solutions for fog environment should be designed accordingly.

In cloud as well as fog environments, there is need for efficient placement of services to satisfy resource requirements as well as optimize various factors such as node utilization, network utilization, service execution cost, energy consumption, performance, availability, and load balancing. In this chapter, we provide a brief description of various open research problems related to infrastructure sizing and application service management in fog environments. We have discussed the problems, and possible approaches towards solving them using our proposed fog architecture as reference.

4.5. Problem description

To facilitate the deployment of new generation IoT applications, cloud computing deployments residing in large data centers are complemented with fog computing deployments distributed over large areas. Along with cloud nodes, fog nodes are leveraged to execute

services. To support different types of IoT applications and their workloads, as well as user and sensor mobility, fog nodes of different resource and cost profiles are made available at different geo-locations resulting in varied user-perceived application latencies and data transfer costs.

Fog computing environment includes large number of nodes with small resource configurations, nodes of varied configurations, usually limited by power and network capabilities along with compute, memory, and storage. Thus, individual fog nodes can't store all, can't compute all, can't send all, as well as can't find all i.e. can't have information regarding the entire system.

In such a complex environment with widely dispersed users, sensors, and nodes, varied application requirements, as well as resource availability, configuration, and cost characteristics, there is a need for efficient approach to deploy services for efficient utilization of resources as well as reduce overall cost. Objective of this research is as follows:

- Considering the huge number of users, applications, service requests, IoT devices, and fog nodes, service deployment approach complexity should be independent of these.
- Solution approach should support user, and IoT device mobility.
- Solution approach should support dynamism i.e. change in availability of fog nodes.
- Solution should be preferably distributed approach, owing to the dispersed nature of fog environment.
- Solution should support heterogeneity in fog environment.
- Solution should be efficient in terms of cost of finding node to deploy a service.

- Solution should identify cost efficient node to deploy a service i.e. with low execution + data transfer costs.
- Solution should preferably be a deterministic approach, rather than a heuristic.

4.6. Solution overview

We assume a Fog Infrastructure as a Service (FIaaS) environment managed by a Service Provider (SP) who offers infrastructure resources to tenants in a pay-as-you-go manner for deployment of services and assures Quality of Service as per contracts. Towards this, service provider deploys different types of nodes at different geolocations, which are available for tenants to deploy and execute their services. Considering an SP-maintained fog, it is safe to assume that fog nodes available in the entire system are of a small number of categories, to satisfy varied application resource requirements.

We propose a fully distributed solution to the problem of service deployment in fog, the details of which are as follows:

When a service deployment request is received by a fog broker, which may be co-located with a fog node, it has the responsibility of finding a fog node to deploy the requested service. In a centralized environment, a central authority will have information regarding the current state of entire system, i.e. resource capabilities of individual nodes and availability of sufficient free resources on them. Such a solution is infeasible for fog environment, considering the nature of fog and its expectations from users. Thus, in a fully distributed approach, each node should be able to make decisions individually to find a node as per requirements.

Considering heterogeneity of fog nodes, and need of services to be deployed on a specific type of fog node, it is helpful to identify nodes of similar type, so that they can be reached quickly without an exhaustive search from the set of all possible nodes. We identify nodes of

similar type with a layer-id. Thus, now, we have a set of fog nodes in system categorized into multiple layers.

To ensure optimal resource utilization, support overflow of requests as well as load balance during moments of high workloads, each node needs information regarding availability of resources on neighboring nodes of similar capacity. Towards this, we group nearby nodes of similar type, which can share their current status among themselves. The nodes are grouped such that the members are closer among themselves than to any other set of nodes, thus making them better candidates to share workload. Instead of having peer-to-peer exchange of possibly redundant information, we identify a local management node which can keep up-to-the-moment status of each of the group members, thus providing local state information to all its members. We refer group as a Puddle, members as PuddlePeers, and local management node as PuddleHead. – This provides answer to the questions by fog node, "What can I do? Is there someone else who has same capabilities? From that set of nodes, which node is a good choice to contact for overflow of requests? "

When a fog node/broker receives a request, it is sometimes the case that a node of its own type cannot serve the request, or it is possible that a node of a different type, e.g. that with larger resources, can serve the request in a cost-effective manner. In such situation, there is a need to identify a fog node of required category i.e. from a different layer. As obvious from the nature of fog-based IoT applications, a node reachable at lower latency is preferred to that reachable at higher latency. Considering that all nodes of a given category, a.k.a. layer, have similar resource, power and network characteristics, network latency can be approximated by between nodes belonging to same fog layer. Thus, a node which is nearer is preferable. Thus, we identify a nearest set of nodes (Puddle) from each layer and save that information with PuddleHead (to

avoid storage of redundant information in each node). Thus, any node which needs access to a set of cost-efficient nodes from a different fog layer, can get that information from its PuddleHead. To avoid redundancy of status information, each Puddle stores information regarding nearest Puddles from only its immediate neighboring layers, i.e. those with nodes of least identifiable difference in nodes' resource configurations. This information is maintained by parent-child control link. Thus, forming a tree hierarchy of Puddles, with Puddles in each tree layer belonging to same category. Thus, extended system state is available without exhaustive search. – This provides answer to the questions by fog node, "Can someone else do it better? How to find the bigger node? How far should I extend the search?"

Now remains the question, "If a request can be satisfied by different types of nodes, then how to know – who is the best one? How to prove it? How far should I look?" This requires the global system state information as we intend to find the best possible node.

As exhaustive search is not feasible due to large time taken to solve the problem resulting from huge state space, and lack of current global system state at any of the management authorities, we can perform a systematic search leveraging PuddleTree.

Puddle Tree includes links between nearest located groups of nodes from each category. Thus, we can move up/down the path from root to leaf in which the Puddle with request initiating fog node is placed. This ensures that the set of nodes which are reachable on this path are the nearest such nodes of each category, thus ensuring least network i.e. data transfer costs. As all nodes of a given category are homogeneous and have similar execution costs, this approach results in identifying a globally cost-efficient node of given type.

The proposed solution time complexity depends on number of predefined categories of fog nodes, which is a constant, and space complexity depends on number of member nodes of a Puddle, which is a very small number as compared to the total number of nodes in system. Solution does not need global state information and ensures system-wide optimality, and is independent of number of nodes, users, and services in system.

Thus, we ensure a globally efficient solution from a small state knowledge. Note that the above solution works for a scenario when the identified node has sufficient free resources to deploy the requested service. I am working on extending the solution to scenario when this is not the case.

4.6.1. Architecture

Owing to the nature of fog environments, management of nodes in fog is more complex as compared to that in cloud [81] [82] [83]. To facilitate the deployment of applications with varied resource and QoS requirements, privacy and security restrictions, on premise requirements of application data and services, the fog environment includes heterogeneous nodes of varying resource configurations, reliability, availability, mobility, ownership, network access, latency, bandwidth characteristics, and are often identified by their physical location. Cloud comprises a set of physical nodes co-located in a managed and controlled data center environment, whereas fog comprises a larger set of physical nodes, dispersed over large areas, possibly in a physically uncontrolled environment, thus affecting their reliability, and availability. Cloud nodes are usually stationary and are equipped with continuous power availability whereas fog nodes may be mobile and possibly energy-constrained, which may impact their availability, and hence the need for regular monitoring of fog environment to ensure continuous availability of resources for service execution.

Towards distributed management and efficient utilization of fog nodes in FIaaS deployment, we have proposed Hierarchical and Autonomous Fog Architecture (HAFA) [84], to organize heterogeneous fog nodes into a logically layered hierarchy. In FIaaS environment, fog service provider makes pooled infrastructure resources available to tenants based on their requirements. Thus the service provider determines the set of fog nodes and their resources based on tenants' resource needs. As nodes are being deployed by a single authority i.e. fog service provider, it may likely choose less diverse nodes which satisfy given application requirements to facilitate ease of management and maintenance. Thus for FIaaS environment, it is safe to assume that fog nodes available in the entire system are of a small number of categories, to satisfy varied application resource requirements.

Heterogeneous fog nodes are classified into a set of categories as shown in Figure 1, and each category of fog nodes forms a fog layer. Nodes belonging to a given fog layer are considered homogeneous w.r.t. computational power, type of energy resources, network connectivity characteristics, as well as deployment and execution cost. Homogeneous nodes in close vicinity form groups, called Puddles, for purpose of resource pooling and local control. Heterogeneous groups of fog nodes i.e. Puddles belonging to different layers which are in close vicinity and belonging to immediate upper and lower layers, are logically connected to provide extended control and autonomy. These logical control links are maintained by local management authority of Puddle, referred as PuddleHead. PuddleHead maintains parent/child control links with Puddles belonging to other fog layers and east/west control links with Puddles belonging to same fog layer. East/West control links are used to share workload information with neighboring PuddleHeads which helps lateral handoff of workload during overflow and failover during disaster scenarios.

The set of control links maintained independently by all the Puddles in system together form a tree-like structure as shown in Figure 2, referred as PuddleTree. Note that PuddleTree itself is not maintained in its entirety by any entity. Instead, it is dispersed in the form of logical control links, which are maintained by individual PuddleHeads. We have shown in [84] that the organization of heterogeneous fog nodes in the form of a logical connected hierarchy helps in efficient maintenance of fog environment by enabling local control while facilitating distributed approach to find nearest fog node with required resource and QoS characteristics to deploy a given application service

Shown in Figure 2 is a pictorial representation of layering, grouping, and interconnection of a set of fog nodes with varied resource configurations. The size of fog nodes represents the amount of resources available on node i.e. larger the dot, higher the resources. Nodes belonging to a given layer are represented by points of same color. Nodes belonging to a specific Puddle are shown to be enclosed in an ellipse.

4.7. Research contributions

We have performed following tasks towards the research.

- FIaaS service model
- Architecture
- Placement
- Pricing
- Simulator

These are discussed in detail in following chapters.

5. Literature Review

5.1. Introduction

In this chapter, we provide a brief description of literary works discussing resource management and service management in fog computing.

5.2. State of the art

Choi et. al. [85] proposed FogOS, a fog computing architecture for IoT services with four main components – service and device abstraction, resource management, application management, edge resource: registration, addressing, and control interface. Paper discussed in detail the challenges involved and factors to consider towards solving the listed problems, due to the nature of fog environments. No assumptions were made regarding nature of fog nodes, and their resource characteristics. There was no discussion regarding organization and management of fog nodes.

Tang et. al. [86] presented a description of various types of fog nodes to support the integration of massive number of infrastructure components and services in future smart cities. Provided a detailed explanation of type of services which can be executed at each fog layer in smart city application domain and showcased using a case study for smart pipeline monitoring system based on fiber optic sensors and sequential learning algorithms to detect events threatening pipeline safety. There is no discussion on organization of fog nodes in large scale fog environment.

Alsaffar et. al. [87] proposed a three-layer architecture of IoT service delegation and resource allocation with device, fog and cloud layers. Proposed an algorithm to allocate resources for service deployment requests to meet SLA and QoS requirements as well as optimizing big data distribution in fog and cloud computing. Any delegation of service requests involves a request to cloud to know the current state each of the other fog nodes. Thus, the proposed solution cannot accommodate latency-critical service requests.

Byers [88] referred to multi-tier hierarchical fog environment and discussed in brief the design factors to be considered towards deployment of fog networks which support applications from one or more of several mentioned IoT application domains. The fog nodes considered were primarily networking components, such as gateway/CPE for local fog, access/edge nodes for neighborhood fog, and core network/routers for regional fog, which are expected to offer compute and storage resources as well along with network resources towards fog deployment, which is quite restrictive. On the contrary, our proposed fog architecture allows inclusion of any node with sufficient compute, storage, and/or network resources to take the role of a fog node, which is logically assigned to a specific layer in fog hierarchy based on pre-defined criteria. The paper did not discuss organization of nodes to facilitate efficient management of large scale fog environment.

Considering that fog comprises a set of edge servers, Munir, Kansakar, and Khan [89] proposed a reconfigurable fog-node architecture that can adapt according to workload being run at a given time and discussed the mapping of proposed fog architecture to intelligent transportation system (ITS) application domain, and briefly discussed the applicability to several IoT application domains. Presented various features to be implemented by fog middleware as

several layers such as application layer, analytics layer, virtualization layer, reconfiguration layer, and a hardware layer. Organization and placement of fog nodes is not discussed.

Chang, Srirama, and Buyya [90] have proposed Indie-Fog infrastructure for fog deployment leveraging consumers' networking devices such as Wi-Fi access point routers, etc. Authors described four basic deployment models for Indie Fog: Clustered Indie Fog – group of stationary Indie Fog nodes in close proximity, Infrastructural Indie Fog – stationery sensor devices providing infrastructure for applications that require temporary data acquisition and processing, Vehicular Indie Fog – vehicles with infrastructure resources act as host nodes to dynamically deploy a software-defined vehicular fog, and Smart phone Indie Fog – smart phones offer infrastructure resources over dynamic, and random connection. There was no discussion regarding interaction among individual fog nodes. Pictures show that a cloud management server residing in a centralized cloud location manages each of the fog nodes in an individual manner. Hence, the proposed approach may be difficult to scale or support application deployment during time-critical scenarios.

Taneja and Davy [91] [92]considered three layer edge-fog-cloud architecture for IoT environments with fog layer being a flat layer, and assumed that, fog nodes are primarily IoT gateways, which pre-process the data from IoT devices, and forward the data for further processing to cloud.

Considering a three layer edge-fog-cloud architecture for IoT environments, Taneja and Davy proposed a strategy for allocation / placement of application modules on fog and cloud nodes. Solution includes sorting the list of available resources of all fog nodes and cloud nodes, as well as the resource requirements of all application modules, then for each application module, resource requirements are matched with those available first in fog nodes, and if not

available, then in cloud nodes, thus ensuring that fog nodes are utilized to the maximum possible extent. Proposed centralized approach has high time complexity, and includes sorting the list of fog and cloud nodes resource configurations, thus assuming that the system is static as well as that knowledge of entire environment is available. Hence the proposed approach is not scalable for large, mobile, and dynamic fog environments.

Xiao and Zhu [93] proposed the concept of vehicular fog computing, which places (mobile) fog nodes on connected vehicles such as bus, taxi, etc. to offer on-demand fog infrastructure while moving along with the traffic, along with static cellular fog nodes located at the edge of cellular networks. This is a specific case of fog infrastructure as we consider vehicular fog nodes as one type of fog nodes available in the system.

Considering a two-layer IoT architecture i.e. device and cloud, Chen and Zhang [94] proposed a centralized approach to offload tasks for local mobile execution, Device to Device (D2D) offloaded execution, and cloud offloaded execution, using a three-layer graph matching algorithm. Fog nodes were not considered.

IBM, along with other researchers, have proposed the concept of Mobile Micro Cloud (MMC) [95], which are micro data centers located at base stations and host IT infrastructure to support low latency services and user mobility. Authors have proposed solutions for the problems: Initial Service Placement and Service Migration. The granularity of approach is very low as jobs are scheduled at various MMCs and number of MMCs reachable at low latency from user location is very small. The solution proposed is a centralized one to offer services to mobile users and not explicitly for IoT environments. New IT infrastructure needs to be deployed and the resources do not scale dynamically as per user needs. Framed the problem of node/link mapping of each service graph to physical node/edge, as an optimization problem, and proposed

exact optimal and approximation algorithms to solve the problem. Numerical results are provided.

Deng et. al. [96] considered the three layered device-fog-cloud architecture and mathematically formulated the workload allocation problem with power consumption / delay tradeoff. Developed an approximate solution to decompose the primal problem into three sub-problems of corresponding subsystems, which are independently solved. Provided numerical results. It has the following limitations: it is a theoretical approach closely approximates the centralized near-optimal approach. Task deadlines are not considered. Locality of data / user is not considered. Assumes that a given task can be assigned to any of the fog nodes in system or cloud. Assumes that the task requests can be delayed and batched together to execute the optimization algorithm. Only goal is minimizing power consumption and user perceived latency.

Widjaja, Borst, and Saniee [97] considered the problem of job scheduling for jobs of multiple job types for distributed small-scale data centers with load reallocation to remote data centers. Provided a model to represent the same and formulated it as an optimization problem. Proposed online centralized and distributed optimization algorithms which operate in a measurement-based fashion to solve the dynamic load reallocation problem. Proposed decentralized iterative algorithms to solve the optimization problem: min-rule, max-rule and min-max-rule. Showed by simulation that the proposed decentralized algorithms can effectively deal with sudden load spikes and hence support elasticity. The problem considered is at the granularity of small data centers; hence the solutions cannot be applied directly to IoT/fog environments.

Zeng et. al. [98] considered a fog computing supported software defined embedded system, where task images lay in the storage server while computations can be conducted on

either embedded device or a computation server. Authors investigated three issues - How to balance the workload on a client device and computation servers i.e. task scheduling, How to place task images on storage servers i.e. resource management, and How to balance I/O interrupt requests among storage servers. These issues are jointly considered and formulated as a mixed integer nonlinear programming problem, and a computation efficient solution is proposed and validated by simulations.

6. Hierarchical and Autonomous Fog Architecture

6.1. Introduction

Internet of Things (IoT) devices, distributed over vast areas, generate big data characterized by 6Vs–Volume, Velocity, Variety, Variability, Veracity, and Value [79]. IoT applications perform complex computations on such data while satisfying stringent QoS requirements [28]. Being resource-constrained and due to the need for consolidation of data from multiple data sources, IoT devices cannot by themselves execute complex services. Hence, fog computing paradigm was proposed by Cisco [13], comprising the idea of deploying fog nodes of variable resource configurations close to data sources to help support applications with low and predictable network latency as well as reduce network bandwidth requirements [41]. Considering 5G infrastructure requirements [80] to support about 1,000,000 devices per sq. km. along with services, users, their mobility, energy-constrained nature of devices, QoS requirements, etc., centralized management approach is almost impractical.

6.2. Motivation

In spite of fog being described as cloud near the edge, management of fog infrastructure differs from that of cloud significantly. Cloud deployment includes a set of servers with nearly homogeneous configurations co-located in a managed data center environment, while fog deployment includes heterogeneous nodes dispersed over wide geographic regions, possibly at unmanaged sites. Contrary to cloud servers, fog nodes can be identified by their physical locations, which is significant in delivering location-sensitive, context-aware services and those only of local value. Cloud physical infrastructure is static, and cannot be changed dynamically to support sudden change in workload in a specific location. Fog leverages cloud resources to support the execution of large-scale data analytics, during workload overflow scenarios, as well as for long-term data access.

Not all nodes are (treated) equal–To facilitate the deployment of applications with varied resource and QoS requirements, privacy and security restrictions, on premise requirements of application data and services, etc., fog environment includes heterogeneous nodes of varying resource configurations, reliability, availability, network access latency and bandwidth characteristics, mobility, ownership, etc., which are often identified by their physical location.

Cloud or Edge does not suffice–Considering the varied resource nature of fog nodes, and the fact that each category of nodes have different network connectivity characteristics, associating each of the above categories to a different layer in fog hierarchy will help in quick identification of fog node with required characteristics such as low latency access, or highly reliable, or large resources; hence, the need for identification of infrastructure layers in addition to cloud and edge.

Incomplete knowledge of system state–In a widely distributed, large scale IoT environment supported by fog infrastructure, it is difficult for a centralized authority to acquire current state of the entire system at any given moment considering large number of devices and services, their distributed nature, volatility of nodes, mobility of users, IoT devices, and at times, fog nodes themselves, etc. and when possible, it will be cost prohibitive and may incur high overhead. Hence, for practical reasons, it is not possible for a centralized authority to make service placement decisions based on complete knowledge of current system state. A breather to this problem is, that IoT environments are usually localized, i.e. data sources, applications, and end users are usually located in close vicinity and hence, complete knowledge of entire system environment is not needed to effiicently allocate infrastructure resources to applications.

Need for organization of fog nodes–In such a fog environment, there is need for efficient management of infrastructure resources such that they can be allocated to services with minimum overhead, reducing overall system maintenance costs and maximizing utilization of fog nodes.

6.3. Contributions

Considering its unique features, we propose a fully distributed fog architecture. The contributions of this paper are as follows:

- Architecture for the organization of heterogeneous fog nodes into logically connected multi-layered hierarchical fog environment for improved load balancing, fault tolerance, and autonomy.
- Grouping of fog nodes belonging to a specific layer using Agglomerative Complete Linkage Hierarchical Clustering method for resource pooling and local control.

- Logically linking groups of fog nodes from different layers to facilitate disaster readiness, ad-hoc deployment, and distributed control over extended area. It also helps reducing effort in finding a low cost node with required resource characteristics for deployment of a service.

Table 4: Factors defining hierarchy in Fog.

Factor	**High Fog Layer**	**Low Fog Layer**
Node capacity	High	Low
Network type	Homogeneous	Heterogeneous
Network bandwidth	Low	High
Network latency	High & Variable	Low & Predictable
Power availability	Continuous (Plugged)	Intermittent / Battery-powered
Mobility	Stationary	Stationary / Mobile
Geographic dispersion	Co-located	Dispersed
Geographic scope	Covers larger areas	Localized
Elasticity	Low	High
Security / Privacy control	Low	High
Physical environment	Managed	Unmanaged public / personal

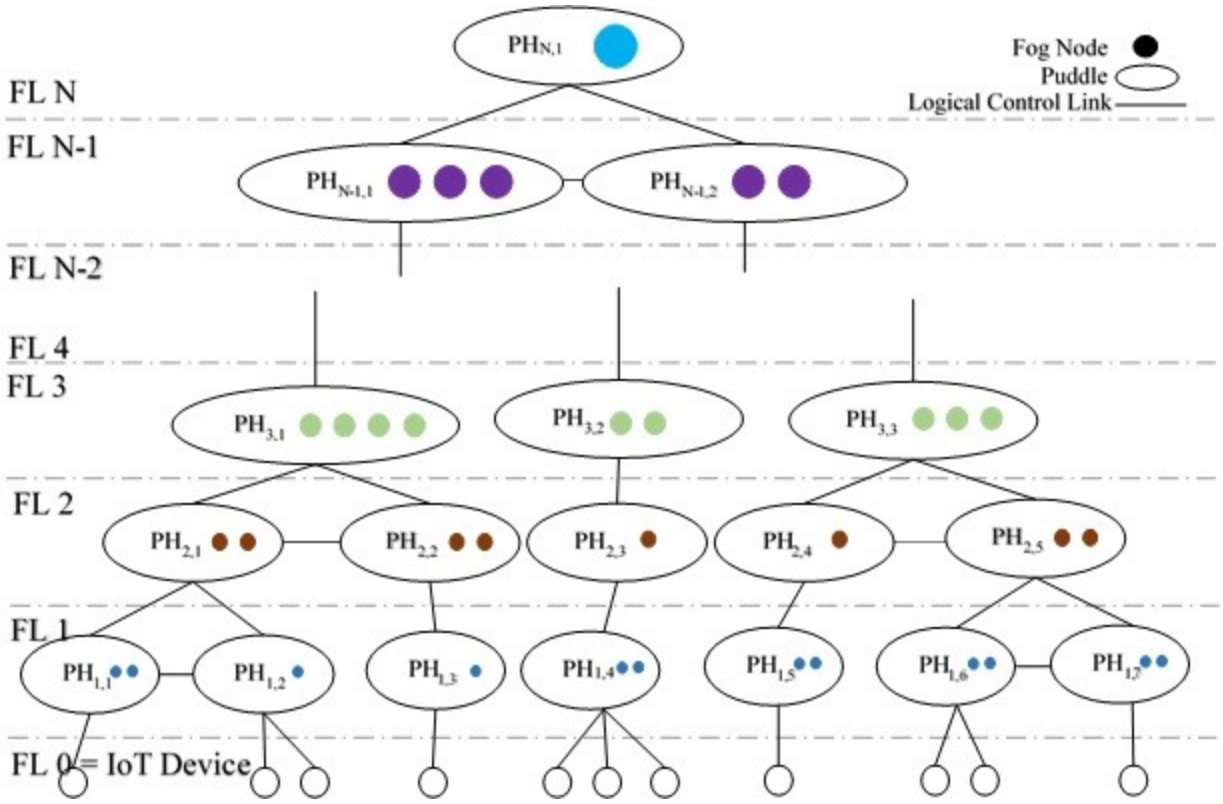

Figure 2: PuddleTree comprising fog nodes with varied resource configurations.

6.4. Hierarchical and Autonomous Fog Architecture (HAFA)

To facilitate efficient deployment of services and management of fog infrastructure, we propose Hierarchical and Autonomous Fog Architecture (HAFA) to organize fog nodes into logically connected multi-layer hierarchy based on several parameters such as location, distance from IoT devices and/or users, node resource configuration, privacy and security. Such a logical organization of nodes in fog environment by no means restricts access among the nodes. Nodes, and users remain reachable from any other node in the system over a variety of networks.

HAFA architecture is proposed in perspective of a Fog Infrastructure as a Service (FIaaS) deployment [16] in which a Fog Service Provider (FSP) offers physical or virtual IT infrastructure resources to tenants towards deployment of services. In such an environment, we can safely assume that there are only a specific number of node types and a set of nodes of

similar type are connected over networks with similar bandwidth and latency characteristics. The details of HAFA architecture are provided in this section.

6.4.1. Layering

Heterogeneity in fog nodes is attributed to several factors as listed in Table 4. A combination of these parameters will define a category. With the fact that each category of nodes has different network connectivity characteristics, associating different categories of fog nodes to a different layer in fog hierarchy helps in quick identification of fog node with required service characteristics of latency, reliability, and capacity. Thereby, resource characteristics of a given fog node can be approximately known from the layer it belongs to.

6.4.2. Grouping

For management purposes, a set of fog nodes in a given geographical area with similar resource characteristics are grouped to form a Puddle. Thus, a Puddle comprises nodes of similar characteristics, and a fog layer consists of a set of Puddles of similar characteristics. It helps reduce the complexity of system management as nodes are managed locally in groups rather than individually, thus facilitating efficient scaling of fog. It also helps in isolating a set of nodes for security, privacy, and service requirements. Fog nodes within a Puddle are located closer to each other than other nodes belonging to the same layer. All nodes in a Puddle are managed by a single authority i.e. same management domain, and are in the same security domain. Grouping of fog nodes into a Puddle facilitates resource pooling to satisfy availability and performance needs of services.

Implementation. As the primary goal of grouping a set of homogeneous fog nodes is pooling of nodes' resources considering their locality a.k.a. nearness, clustering approach can be used to group a set of nodes. As nodes belonging to a given fog layer are connected over

networks of similar characteristics, link latency between any two nodes can be approximated by the distance between them. Thus, objective of clustering is to minimize average inter-node distance between any two nodes within a group, which is synonymous to minimizing average inter-node latency. The outcome of this procedure is that neighbor fog nodes are clustered to form a Puddle. Other possible objectives for clustering are limiting size, i.e. number of nodes per Puddle–which determines the resource requirements towards management of the nodes belonging to the Puddle, limiting maximum distance between any two nodes in Puddle–which determines the maximum inter-node latency between any two nodes in the Puddle, etc.

6.4.3. Local Management

Each Puddle has a local management authority, called PuddleHead, which manages the membership of nodes by tracking their geographic positions and mobility, allows registration of new fog nodes and terminates membership of nodes that have left. Thus, it monitors local topology changes, local system performance metrics, member nodes' resource configurations as well as resources available to deploy new services, services hosted by individual member fog nodes, etc. The PuddleHead maintains fog at the level of a single Puddle such that the monitored nodes can exist as an independent fog entity and execute services to support the local IoT environment. Thus, the PuddleHead brings autonomy to the fog environment by removing dependence on a centralized management authority.

6.4.4. Inter-Layer Connections

To ensure fog autonomy at any layer within an area, we organize Puddles into a connected hierarchy. Each Puddle is logically connected only to a single Puddle in the vicinity belonging to immediately next higher layer and one or more Puddles in same and immediately next lower layer. Upon need, we can traverse up / down the hierarchy and find the Puddle that has nodes of sufficient resources to serve the given service request. As Puddles are connected

based on locality, the selected higher layer Puddle with required resources is guaranteed to be the nearest such Puddle from higher layer and hence can be considered for deployment of latency-critical applications. A Puddle at higher adjacent layer is referred to as parent Puddle and the Puddle at lower adjacent layer is referred to as child Puddle. These connections, also referred to as north-south links, are identified by logical control links between Puddles and information is maintained by corresponding PuddleHeads. Number of children Puddles to a parent Puddle, is variable and depends on relative density of upper layer Puddles as compared to those of lower layer in a given area.

6.4.5. Intra-Layer Connections

Intra-layer connections are leveraged to share information among PuddleHeads belonging to same layer regarding workload and resource availability, and can be primarily used for load sharing during workload overflow by lateral handoff of services to Puddles in neighborhood, failover during disaster scenarios, as well as for exchange of information regarding impending migration of services due to mobility of users / IoT devices, expected loss of fog nodes in current Puddle due to energy depletion, addition of new nodes in a neighboring Puddle resulting in a large amount of available resources, etc. These connections, referred as east-west links, are maintained only for siblings i.e. children Puddles of same parent Puddle. This restriction ensures locality of logically connected nodes, helps in collectively migrating multiple service components of an application which are deployed on nodes belonging to different fog layers and thereby facilitates autonomous local fog.

6.4.6. Control Path: PuddleTree

Groups of fog nodes in a given fog layer are logically connected to groups belonging to immediate upper and lower layers using control links forming a tree-like structure of Puddles, referred as PuddleTree. PuddleHeads of connected Puddles exchange system state information

over control path, thus extending the concept of fog autonomy to any possible number of layers in fog hierarchy. Thus, we have a set of homogeneous Puddles in each fog layer and heterogeneous fog nodes are organized into a connected hierarchy of fog layers. It facilitates distributed control, and hastens search for a fog node with required characteristics. Shown in Figure 2 is a pictorial representation of grouping, layering and interconnection of a set of fog nodes with varied resource configurations. The size of fog nodes represents the amount of resources available on node i.e. larger the dot, higher the resources. Nodes belonging to a given layer are represented by points of same color. Nodes belonging to a specific Puddle are shown to be enclosed in an ellipse.

6.5. Conceptual analysis: For architecture support for IoT / Fog-based applications

In the earlier sections we have proposed fog architecture and provided a description of roles of various components in fog environment. In this section, we discuss how the proposed architecture supports unique features of fog environment facilitating the deployment of applications in large scale leveraging fog infrastructure.

Dynamism: The proposed architecture allows users, IoT devices, as well as fog nodes to join and leave the system at will, resulting from loss of battery, mobility, or new nodes being deployed on-the-fly in disaster/overload scenarios. It supports mobile users, and nodes, resulting in a dynamic environment which adapts to user workloads and application needs.

Scalability: The fully distributed approach of managing IoT devices and fog users by associating them with Puddles, allow the deployment of large scale fog environments.

Elasticity: When a new fog node joins the system, it will either join an existing Puddle or form a new Puddle and system state will be updated accordingly at respective PuddleHeads.

Similarly when a fog node leaves the system, its information will be removed from its Puddle and if there are no more fog nodes in the Puddle, it will be removed as well. Thus the system can scale up or down based on number of fog nodes with less overhead and minimum impact on system uptime.

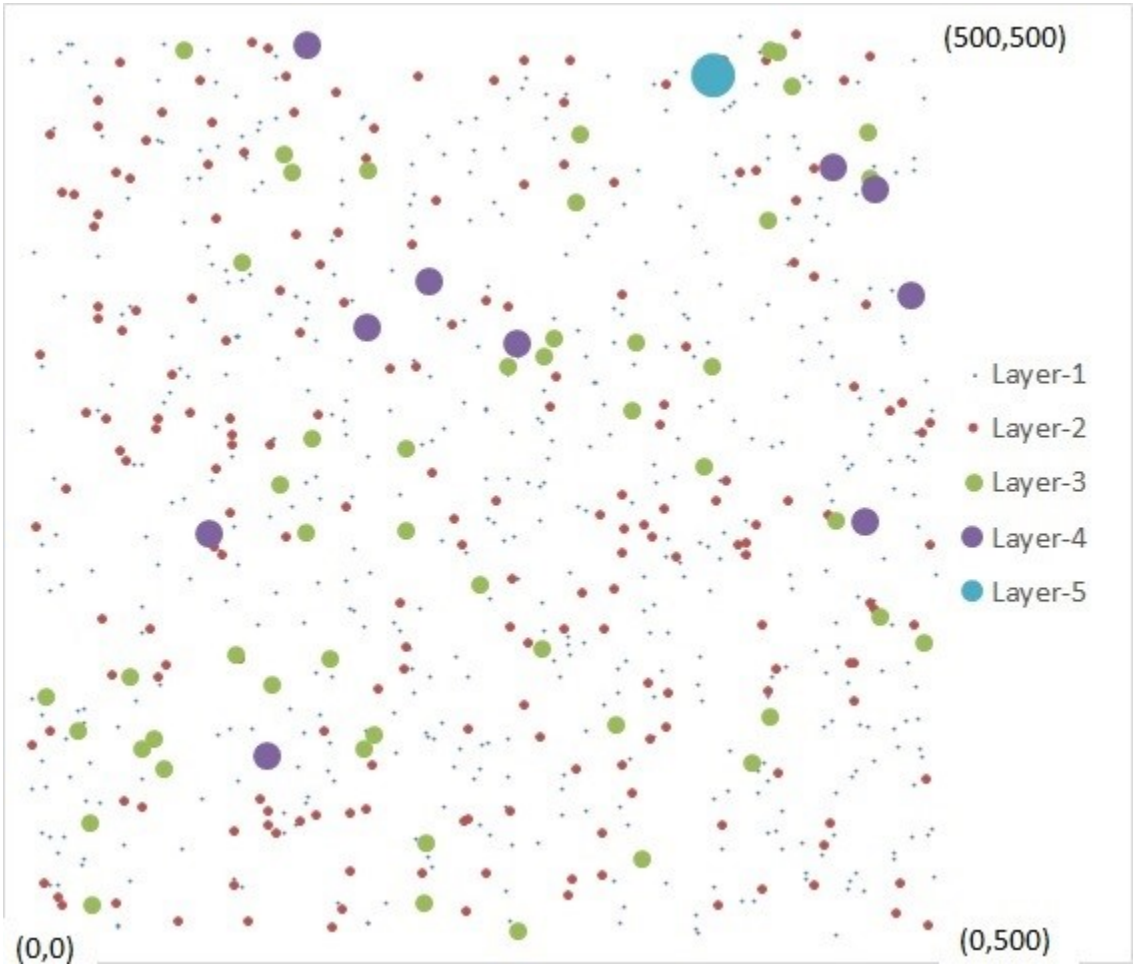

Figure 3: Randomly distributed fog nodes at various layers.

Heterogeneous nodes: The proposed architecture does not restrict fog nodes to be uniform. On the contrary, it takes advantage of heterogeneity among fog nodes by grouping them based on their location, capacity, network characteristics, etc. into layers of fog infrastructure for efficient management and application deployment.

Large number of nodes: Availability of large number of fog nodes in all sizes provide sufficient resources to execute services which process data from huge number of IoT devices. The proposed architecture eases management complexity by distributing the management responsibilities to local PuddleHeads.

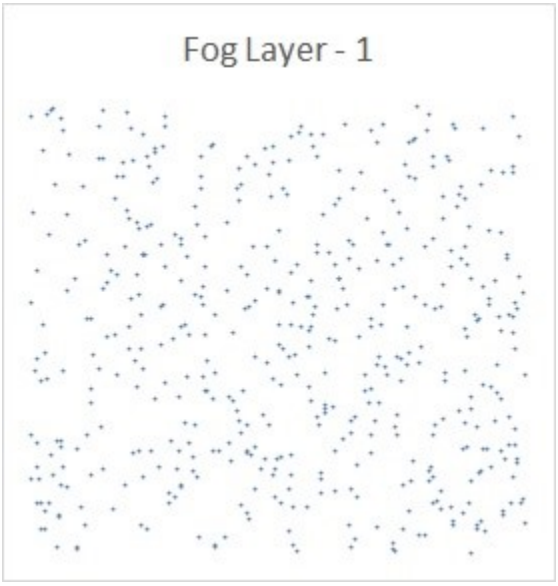

Figure 4: HAFA - Layering: Fog layer-1 nodes.

Dispersed data sources: Along with data sources i.e. IoT devices, fog nodes are also dispersed over the entire geography and are placed physically close to the IoT devices. This placement strategy allows fog nodes to process the generated data close to its source resulting in reduction of the amount of data traversing the network or sent to cloud.

Latency-critical applications: Fog nodes can be placed close enough to application users as well as IoT devices such that the QoS requirements of latency-critical applications can be satisfied facilitating real-time decision making.

Bandwidth-intensive applications: Fog nodes located close to the data sources can preprocess the generated data to prune it prior to its actual processing by removing redundant

and erroneous data elements, thus reducing the amount of data to be processed further by more than a degree and help deployment of bandwidth-intensive applications and those which generate data at a high velocity, without beefing the network infrastructure.

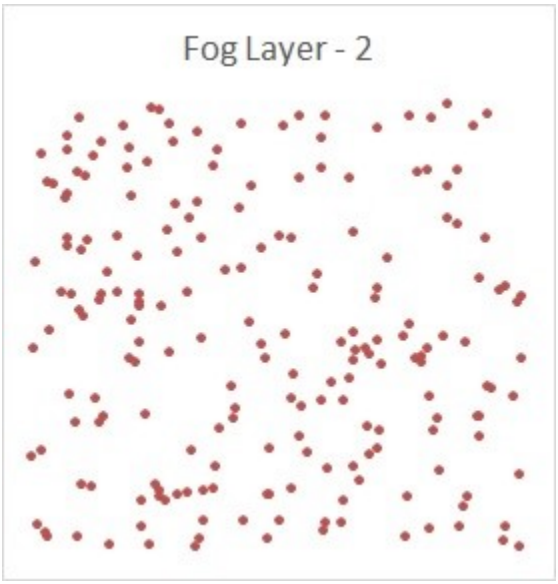

Figure 5: HAFA - Layering: Fog layer-2 nodes.

Varied QoS requirements: Fog environment includes large number of fog nodes with varied resource configurations organized in a hierarchical fashion, which can support deployment of applications with different resource and QoS requirements. These fog nodes provide infrastructure resources to cater to service requirements which vary widely for various parameters such as response time, throughput, input/output data, geographic distribution of data sources and users, etc.

Location-sensitive services: IoT devices and Fog nodes, being dispersed over the geography and being aware of their locations, can effectively execute services which depend on knowledge of location of IoT device, fog node, or the end user.

Localized services: Fog nodes, having deployed close to data sources, can leverage the data to locally execute applications and services which have high local value and very low or no global value, thus reducing the network bandwidth requirements further.

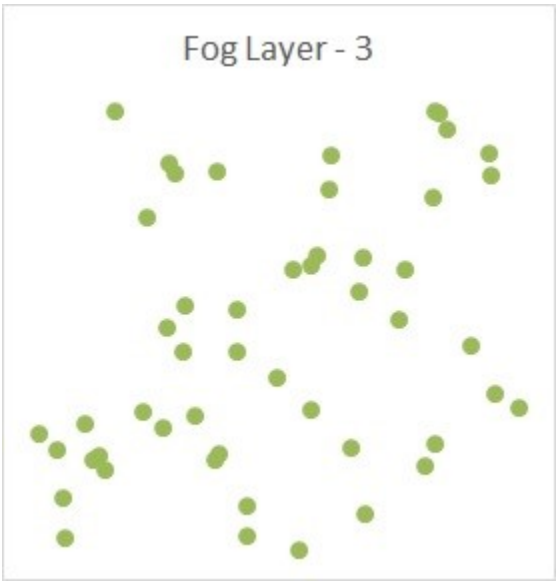

Figure 6: HAFA - Layering: Fog layer-3 nodes.

Mobility: When a user or IoT device moves around and loses connectivity with its associated fog node, the system dissociates it and re-associates with a different fog node which satisfies the applications' resource and QoS requirements. In a similar manner, when a fog node drifts away from the range of its Puddle, it will be re-associated with a different or new Puddle and system state will be updated. Thus the proposed architecture supports mobility of various system components.

Autonomy: Proposed fog architecture groups fog nodes into Puddles which can act as independent entities leveraging infrastructure resources from fog nodes comprising the Puddle to deploy applications which can consume the data produced by IoT devices. The Puddle is managed locally by its corresponding PuddleHead, which has complete freedom in decision

making towards the applications deployed on fog nodes in Puddle. Thus, introducing autonomy in system at the level of an individual Puddle, and anywhere up the hierarchy of fog.

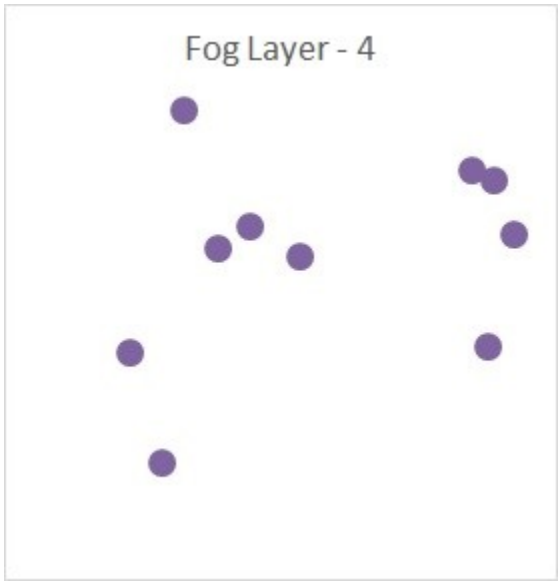

Figure 7: HAFA - Layering: Fog layer-4 nodes.

Disaster readiness: Being autonomous entity, a Puddle, or a hierarchy of Puddles can survive the loss of network connectivity to a centralized management entity available in system, if any. Upon restoration of network access, the system state can be updated with cloud environment. This feature facilitates deployment and survival of fog in disaster scenarios.

Robustness: Owing to the hierarchical Puddle-based organization of fog nodes, any problem with fog environment is contained within its corresponding Puddle or limited to a small set of Puddles, while rest of system continues to function as normal, thus making the fog a robust system.

Agility: The proposed architecture quickly updates the system state to reflect any change in network connectivity and availability of fog nodes, by initiating service migration to

appropriate nodes in the system. Any change in environment is locally managed by one or more Puddles, and there is no global action required.

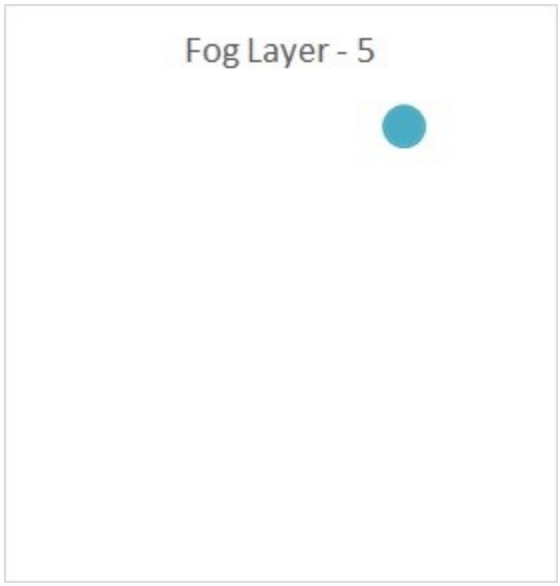

Figure 8: HAFA - Layering: Fog layer-5 nodes.

Privacy and security: Private fog nodes with restricted access can be used to deploy applications which can process private data. These fog nodes are deployed for a specific purpose and are not available for generic use. These fog nodes, connected over private network, can form an autonomous Puddle hierarchy or connect to vast fog environment using a public gateway to access extended fog infrastructure resources during special instances such as workload overflow, etc. Thus offering improved security and data privacy to users in fog environment.

Cloud integration: Integration of fog environment with cloud is required to execute latency-tolerant services which can effectively leverage low-cost cloud resources, applications which need large number of resources or large amount of input data e.g. large scale data analytics, or during instances of workload overflow when fog resources are not sufficient to

serve large number of users. Our proposed approach supports cloud-fog integration by considering cloud as another fog layer and extending connectivity to it from lower fog layers.

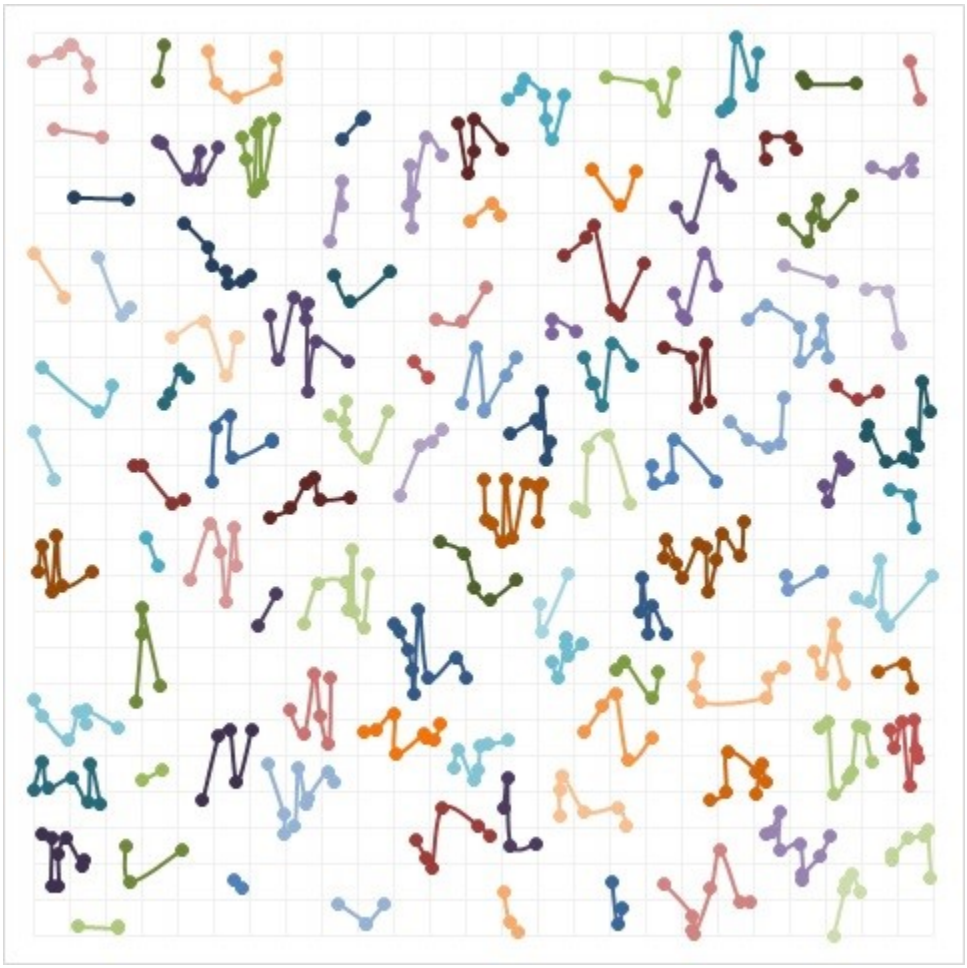

Figure 9: HAFA - Grouping: 100 Puddles formed from 500 nodes belonging to layer-2.

6.6. Experimental analysis

To demonstrate the features of HAFA, we created a hypothetical data set representing various types of fog nodes deployed in a prospective smart city environment. The dataset includes five types of fog nodes, such as those shown in Figure 1, each represented by a fog layer, and vary in resource configuration and mobility. There are a total of 781 fog nodes spread over an area of 500x500 sq. units with 500 nodes in layer-1, 200 nodes in layer-2, 50 nodes in

layer-3, 10 nodes in layer-4, and 1 node in layer-5. Location of each node was generated randomly by uniform distribution over the coordinate space of (0,0)..(499,499). The x- and y-coordinates represent latitude and longitude (from GPS) of fog nodes in a real-world deployment.

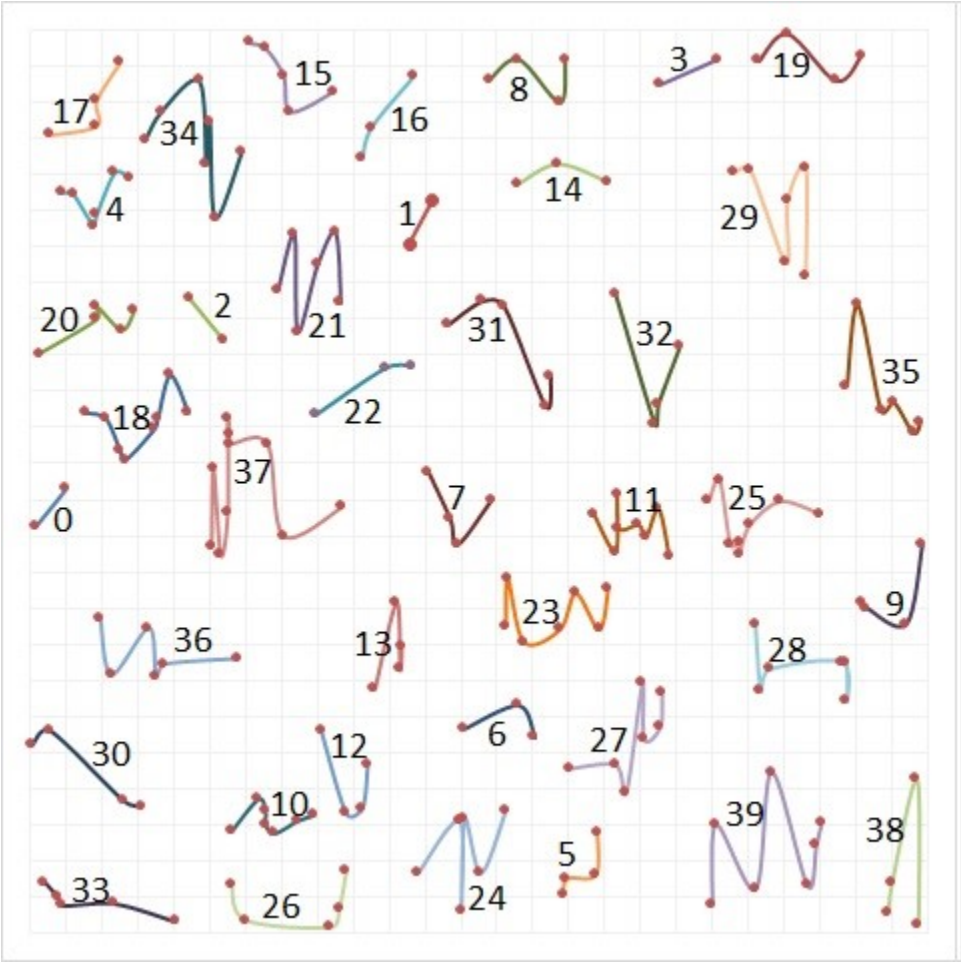

Figure 10: HAFA - Grouping: 40 Puddles formed from 200 nodes belonging to layer-2.

Uniform random distribution of fog nodes is assumed to reflect the layout of nodes in a city center which is likely to have uniform population density and hence the need for IT resources. Shown in Figure 3 are the fog nodes in the test environment, with nodes belonging to

each layer represented by a unique color. Higher layer nodes are depicted with larger size vertices to represent their larger resource capacities.

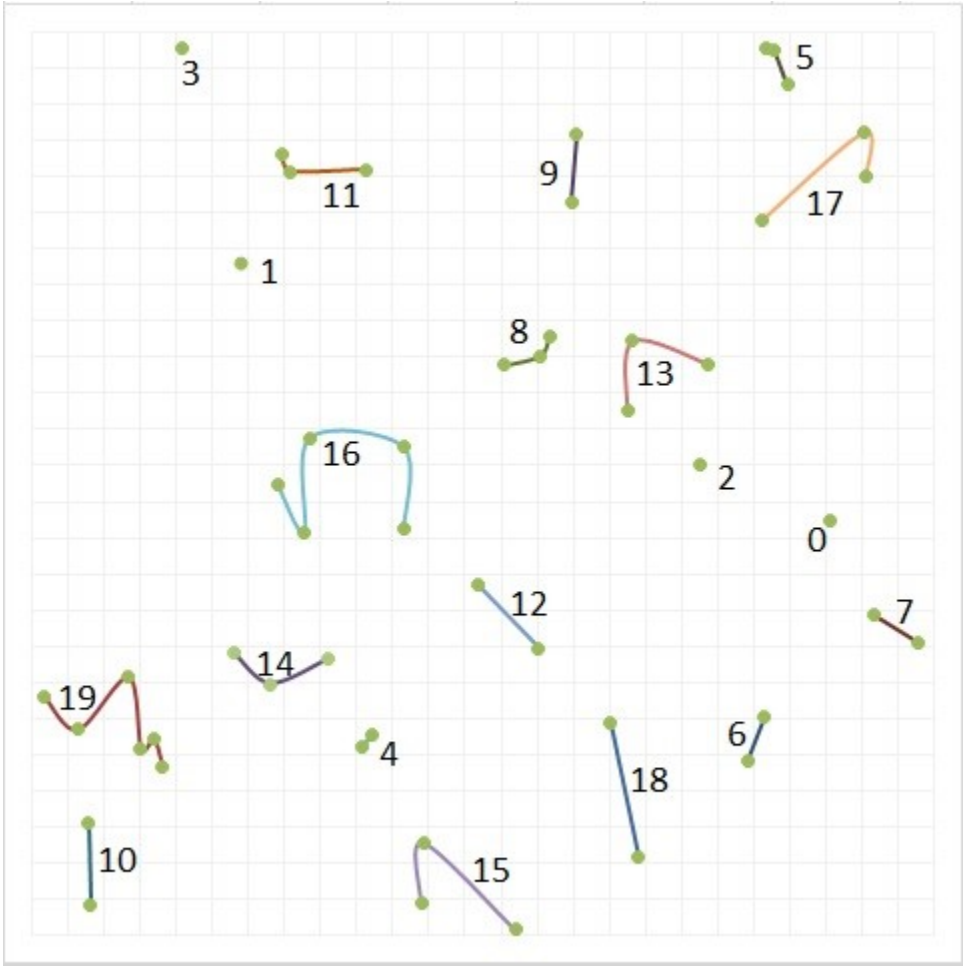

Figure 11: HAFA - Grouping: 20 Puddles formed from 50 fog nodes belonging to layer-3.

6.6.1. Layering

The given set of nodes are categorized based on their resource capacities into five layers as shown in Figure 9, Figure 10, Figure 11, Figure 12, and Figure 13.

6.6.2. Grouping

Here we demonstrate an approach to group the nodes in an efficient manner. Given the relative locations of a set of fog nodes belonging to same fog layer dispersed over the given area,

we have grouped them into Puddles using Agglomerative Complete Linkage Hierarchical Clustering approach [79]. The objective function of clustering is to form clusters of fog nodes from given set such that the average distance between nodes belonging to a cluster is minimal. This clustering approach ensures that the nodes belonging to any group are collectively closer to each other as compared to those from a different group, thus ensuring that the members of same group are better candidates for load balancing and workload overflow as compared to members of different groups.

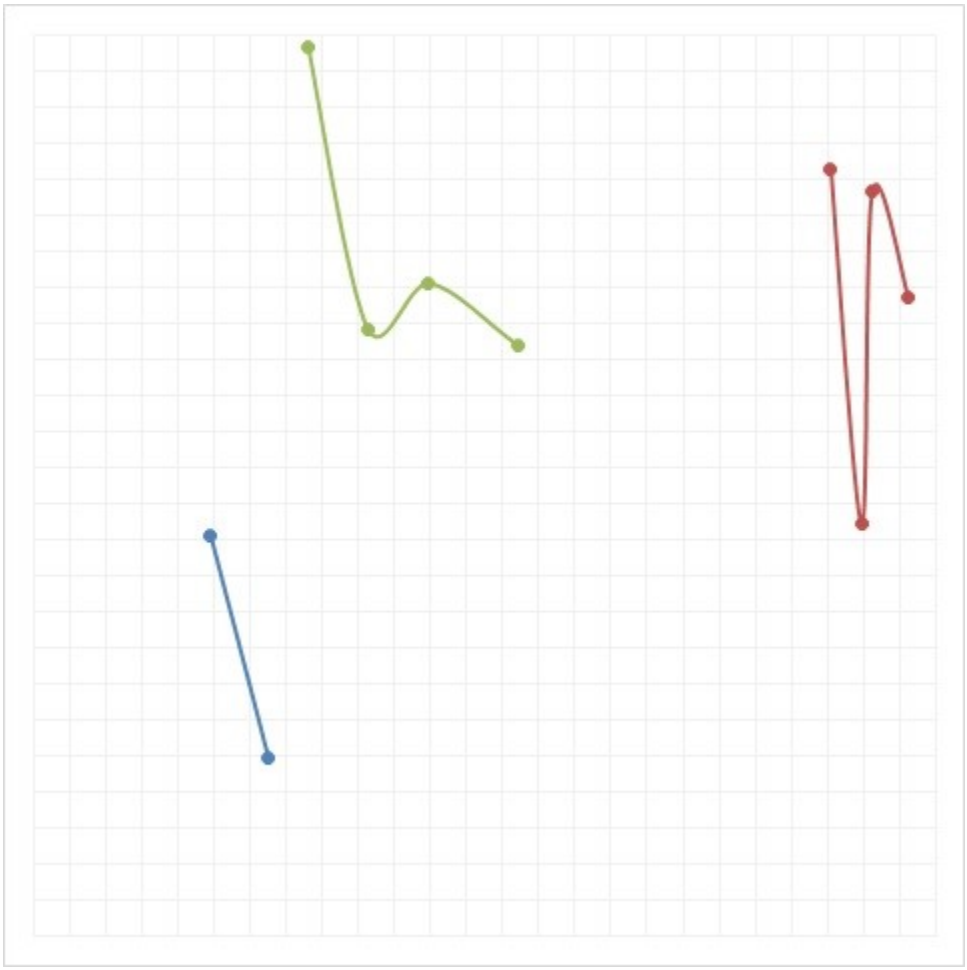

Figure 12: HAFA - Grouping: 3 Puddles formed from 10 fog nodes belonging to layer-4.

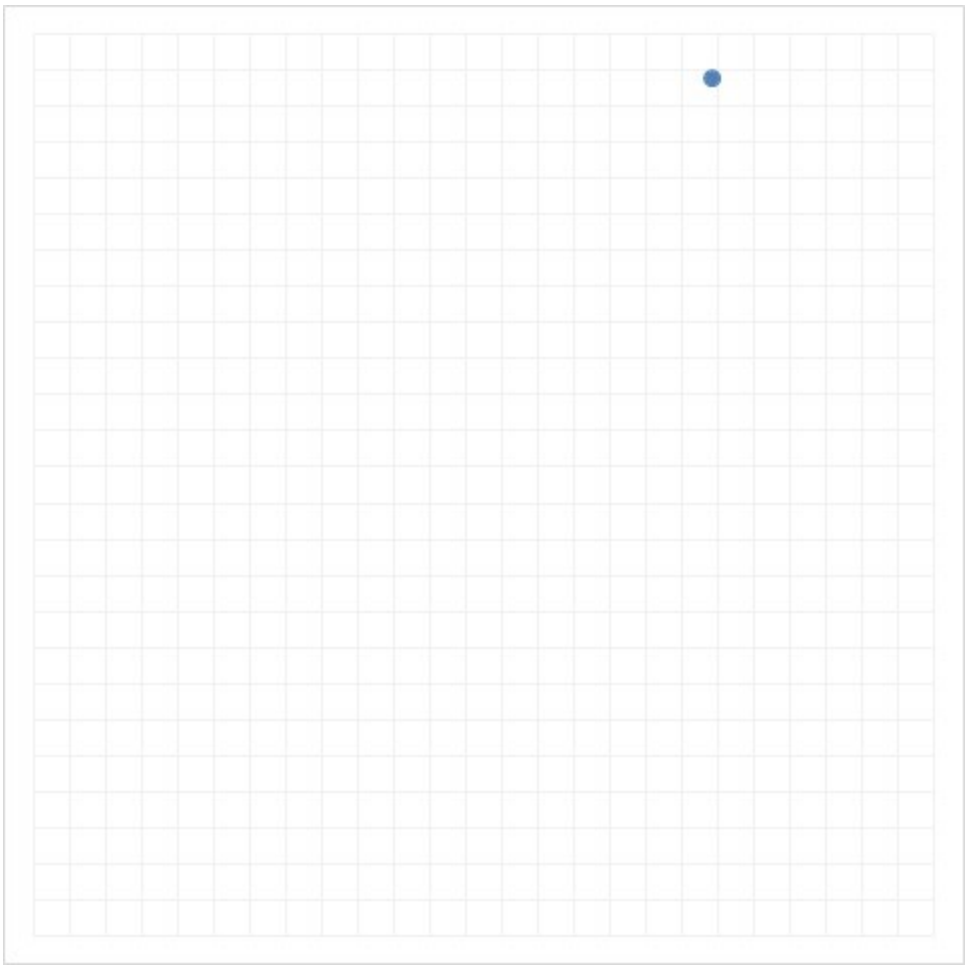

Figure 13: HAFA - Grouping: 1 Puddle formed from 1 fog nodes belonging to layer-5.

Shown in Figure 10 are 40 Puddles formed using the above mentioned clustering approach from 200 fog nodes belonging to layer-2. Member nodes of a Puddle are shown as connected by links of same color and marked by the Puddle id. Note that the links which connect the nodes represent their membership of Puddle, and not the actual network access to the nodes. Similarly, Figure 11 shows 20 Puddles formed with 50 fog nodes belonging to layer 3. 1, 3, and 100 Puddles formed with 1, 10, and 500 fog nodes belonging to layers 5, 4, and 1 respectively, are not shown in this paper for conciseness. Note that the Puddle sizes, i.e. number of nodes in a

Puddle is not same across Puddles, as the Puddle formation depends on number of nodes in vicinity to pool resources from.

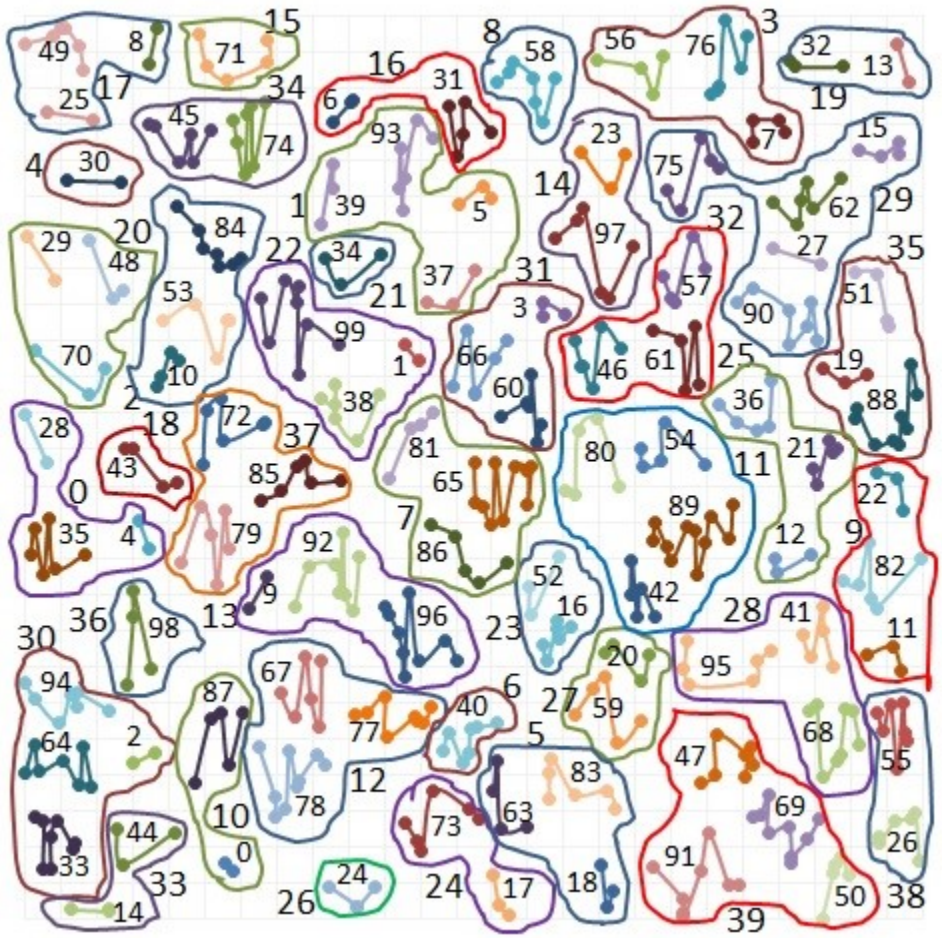

Figure 14: HAFA: Parent-Child relationships between fog layer-2 and fog layer-1.

Note that we have used inter-fog node Euclidean distance as a metric to cluster the nodes. Instead, inter-fog node access latency can be used for more accurate clustering. We have implemented other clustering methods as well such as K-Means, Divisive hierarchical clustering, Agglomerative single linkage hierarchical clustering, etc. However, we have observed that the features of selected approach are more appropriate for management of fog environment. K-

Means approach requires specification of number of clusters to be formed from given set of nodes, which is neither optimal nor feasible with dynamic fog environments.

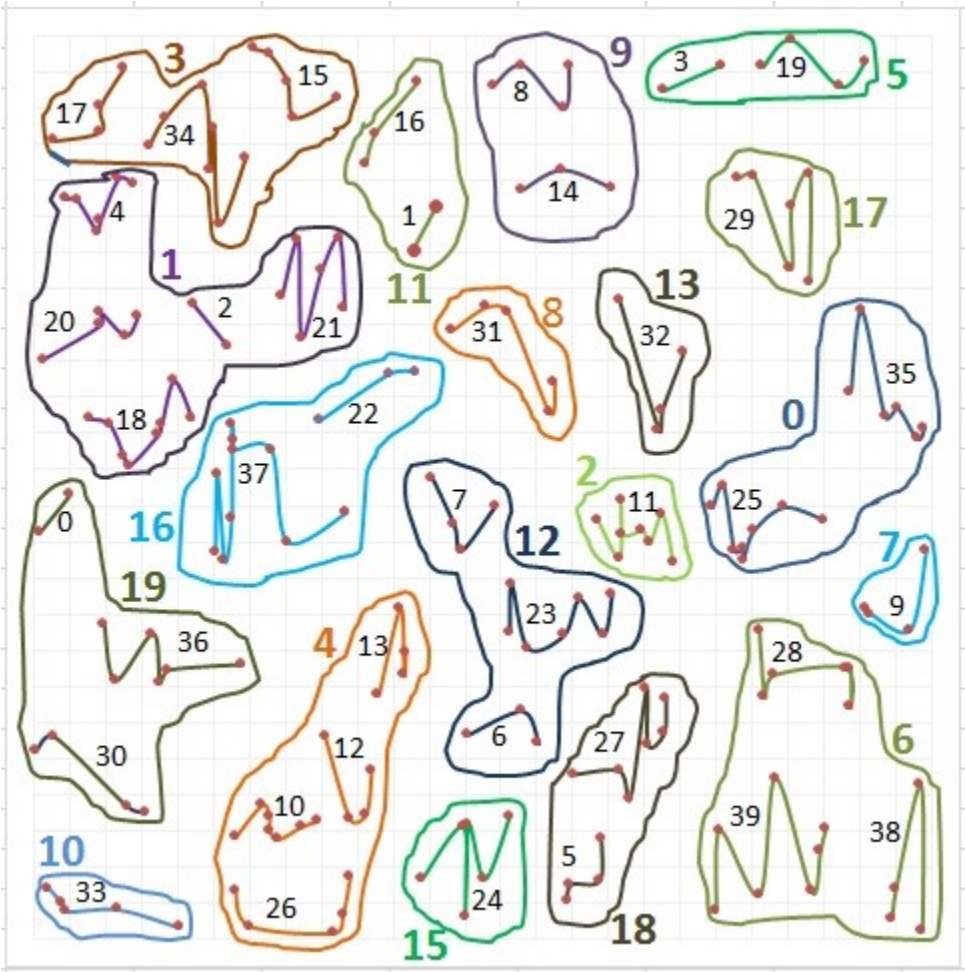

Figure 15: HAFA: Parent-Child relationships between fog layer-3 and fog layer-2.

Contrary to top-down Divisive clustering approach, Agglomerative clustering approach naturally fits the problem of grouping fog nodes from bottom-up, as fog nodes are widely distributed over large areas and number of nodes per cluster is expected to be very small compared to the overall set of available fog nodes in system. Single Linkage clustering considers only one link to identify nearest cluster to merge, whereas, Complete Linkage approach considers links between all possible sets of two nodes, one from each cluster, to identify the best

candidate cluster to merge, thus forming clusters optimal for pooling of resources. The objective of clustering procedure is based on only one feature i.e. inter-node distance. Hence, the selected approach has low computational and spatial complexity, as nodes farther apart beyond a given limit need not be considered and can be efficiently implemented using sparse adjacency matrix.

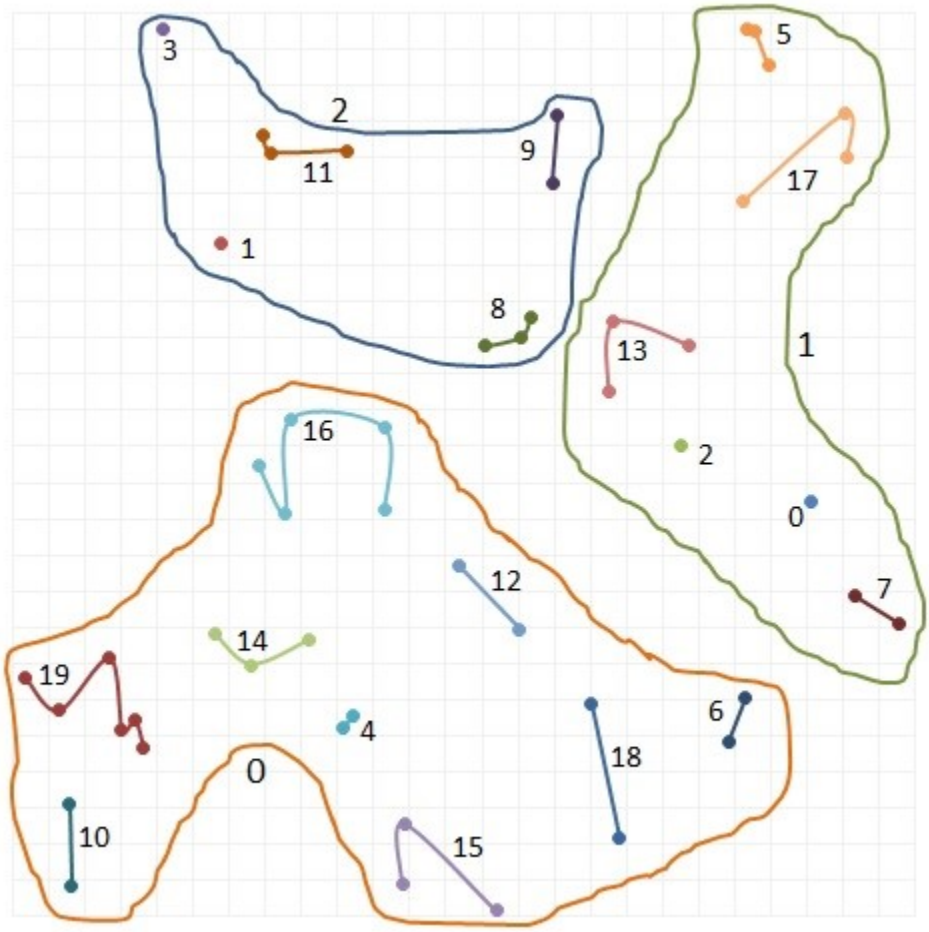

Figure 16: HAFA: Parent-Child relationships between fog layer-4 and fog layer-3.

6.6.3. Inter-Layer Connections

Here we demonstrate an approach to form the parent-child relationships among Puddles in adjacent layers. For each Puddle in lower layer, we select the parent Puddle in the immediately higher layer using the Complete Linkage method [79]. Complete linkage approach ensures that all nodes within a Puddle (not just the centroid) are collectively closer to all nodes of selected

parent Puddle as compared to any other Puddle in the immediate higher layer. Shown in Figure 15 are parent-child relationships between Puddles belonging to layer-2 and layer-3. Figure 11 was overlaid on Figure 10 to create Figure 15. The connected nodes (dots) belong to a specific Puddle in layer-2 and are marked with the corresponding Puddle id. All Puddles which are identified to be children of the same parent Puddle of layer-3 are depicted in same color and enclosed with an outline, which is labelled by the parent Puddle id.

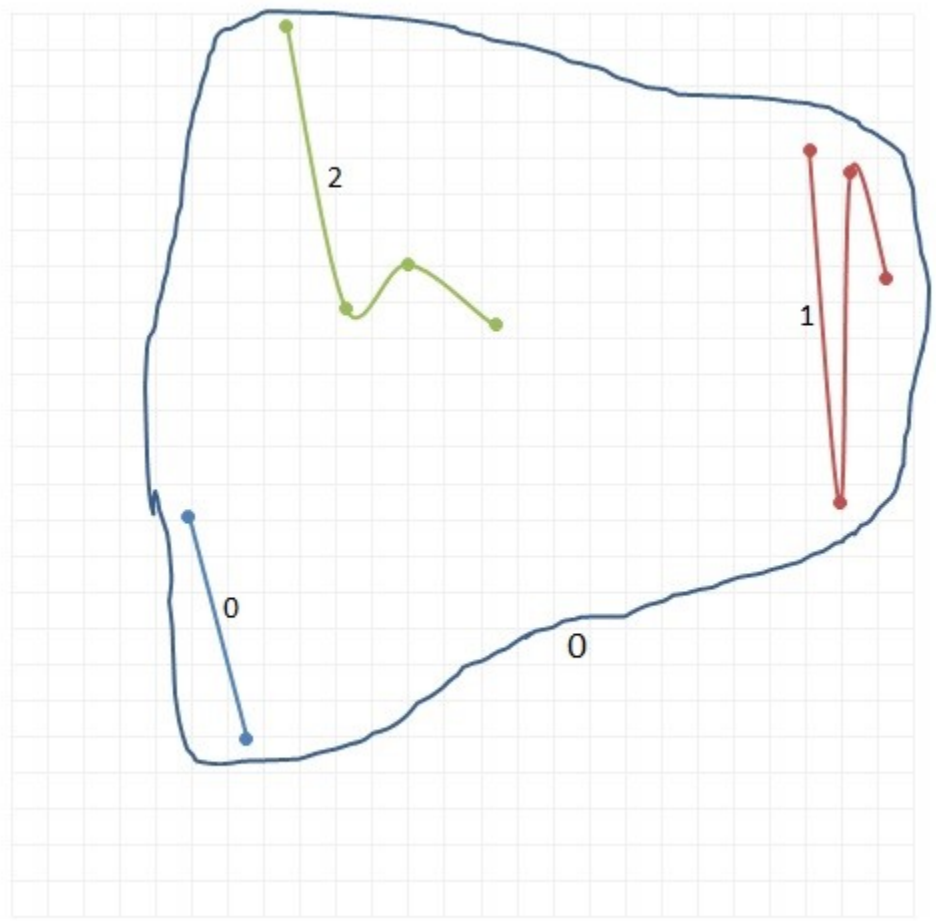

Figure 17: HAFA: Parent-Child relationships between fog layer-5 and fog layer-4.

For the test dataset, shown in Figure 18 is the PuddleTree hierarchy i.e. control links representing parent-child relationships between Puddles of immediately adjacent fog layers, i.e.

between 5-4, 4-3, 3-2, and 2-1. Note that each vertex in the tree represents a Puddle formed by a set of fog nodes belonging to the specified fog layer.

6.6.4. Search for node

Here, we demonstrate how PuddleTree control path can be leveraged to efficiently find a Puddle with fog nodes of required resource characteristics located closest to the user initiating the request. Assuming that the request is initiated by user co-located with a layer-1 fog node and the request includes resources from nearest layer-n fog node, following procedure is performed: Identify the Puddle to which the layer-1 fog node belongs to. Starting with the leaf node, which represents the Puddle, move up the control path towards root node on PuddleTree by (n-1) links. Destination vertex on control path is the Puddle comprising fog nodes of required resource characteristics, located nearest to the user.

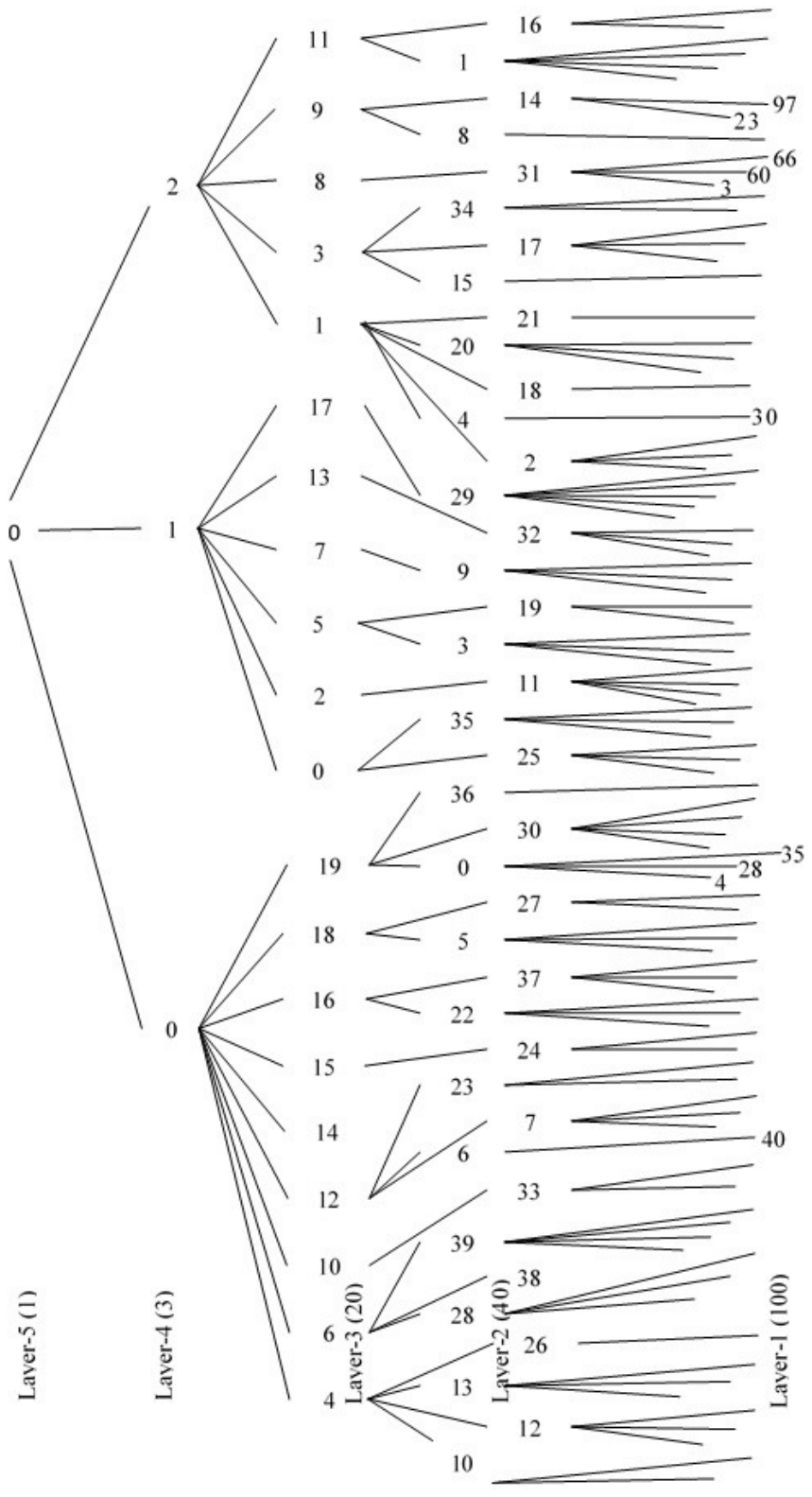

Figure 18: PuddleTree representing connected fog hierarchy.

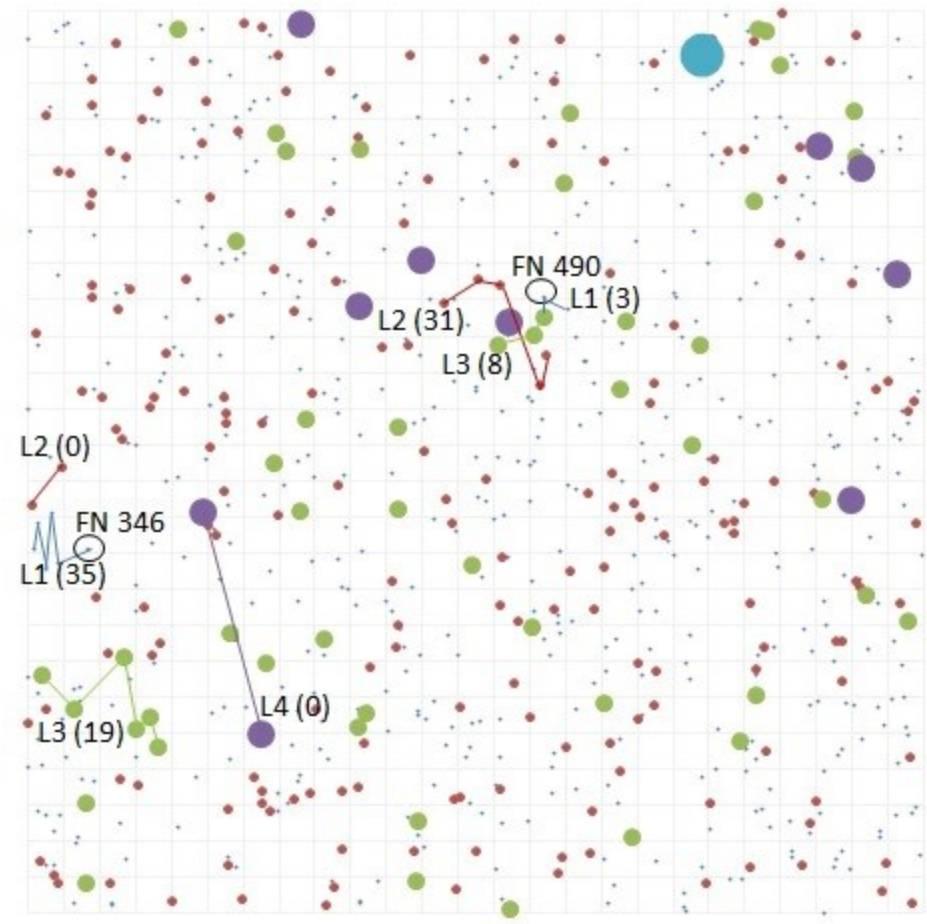

Figure 19: Search for nodes.

For instance, assume that the user initiates request for a node belonging to layer-4 from layer-1 fog node with id: 346, which is part of layer-1 Puddle with id: 35. From Figure 19, we see that layer-1 Puddle with id: 35 is connected to layer-2 Puddle with id: 0, which is in turn connected to layer-3 Puddle with id: 19, which is in turn connected to layer-4 Puddle with id: 0. Thus, by traversing n-1 = 4-1 = 3 links in PuddleTree, we can identify fog nodes which are located nearest to the user and satisfy the resource requirements. Shown in Figure 19 are the Puddles whose PuddleHeads are visited during the search for fog node with required resource

characteristics. L1, L2, L3 and L4 represent layers 1, 2, 3 and 4 and L2 (31) represents node 31 which belongs to layer 2.

Note that PuddleTree itself is not maintained in its entirety by any entity. Instead, it is dispersed in the form of logical control links, which are maintained by individual PuddleHeads. Compare this with a distributed peer-to-peer approach where there is no differentiation among fog nodes based on their resource configurations. In such scenario, resource request will be broadcasted to all neighbors, which is further forwarded until a candidate destination fog node with required resources is found. This involves high overhead for sending an indefinite number of messages as well as introduces a large delay in finding the target node while consuming network bandwidth.

6.7. Critical review

In this chapter, we have proposed Puddle-based hierarchical organization of heterogeneous fog nodes to facilitate deployment of scalable, large-scale IoT environments. Listed below are unique features of proposed fog architecture. A brief overview of feature comparison between our proposed architecture for fog environments and those from references is provided in Table 5.

- System-wide architecture – allows deployment of fog environments of any scale.
- Puddle-based grouping – supports the concept of pooling infrastructure resources to offer capacities that can be jointly offered, which is the basis of cloud computing, rather than considering the fog nodes as simple individual devices.
- Widely distributed fog – allows large scale deployments with fog nodes distributed over very large areas, potentially across the globe.

- Distributed management – complete knowledge of entire fog environment by a single managing entity is neither possible nor required, hence we proposed fully distributed approach for efficient management of fog.
- No one managing entity might even have knowledge of entire fog environment, on contrary to cloud management approach. Hence fog management is difficult.
- Heterogeneous fog nodes – allow any device with sufficient infrastructure resources to be leveraged for service execution.
- Support for mobility – allows IoT devices and application users to change their position over time while retaining access to services, while mobile fog nodes e.g. smart vehicles, etc. can be leveraged to provide services for users on-the-go.
- Support for dynamic and elastic environment – allows energy-constrained or mobile fog nodes to join and leave the system at will, to support intermittent peaks in workloads.
- Pervasive fog nodes – allow deployment of latency-critical and bandwidth-intensive applications.
- Location awareness - allows deployment of location-sensitive applications as well as ensure locality of data and applications.
- Network connectivity – allows ubiquitous access to services.
- Multi-layer hierarchical organization – allows categorization of fog nodes based on various criteria and quick identification of a node with required resource characteristics.
- Variable-layered fog hierarchy – i.e. based on infrastructure available to deploy services, number of fog layers is different as viewed from different locations, by

different tenants even in a given area, by various applications depending on their resource requirements.

- Distributed resource management – PuddleHead manages and monitors infrastructure resources of nodes in a Puddle.
- Distributed service management – PuddleHead manages deployment and execution of services on fog nodes of the Puddle for efficient utilization of resources.
- There is no specific cloud layer. Any upper layer can have ultimate control w.r.t. a tenant or a specific service.
- We consider centralized cloud located at large data center as one of the (i.e. highest) fog layers with special characteristics w.r.t. (high) reliability, (logically infinite) scalability, etc. and its management is no different from other layers of fog i.e. uniform organization and architecture and management across fog layers.
- Grouping fog nodes into Puddles allows establishment of security and privacy domains. A gateway Puddle can be used to connect to a subset of private Puddles.
- Interconnection of Puddles and establishment of hierarchy among them allows dynamic federation of fog with other fog / cloud environments, as well as support for disaster / intermittent connectivity scenarios.
- We do not accept the standard assumption that clouds are always located far from edge devices. This may not be true always, as edge devices are spread over large areas in large quantities. Hence some devices may be closer to cloud / higher fog layer nodes, as compared to edge / lower fog layer nodes.

- Reliable fog – By handling failures appropriately in individual Puddles and fog layers, required reliability standards can be provided in fog.
- Scalable architecture – as changes to fog environment due to extension to new areas by addition of new nodes, or loss of connectivity resulting from disaster occurrence in an area are local and have minimal impact on rest of the system.
- Generic architecture – can be applied to various application domains and makes no assumptions regarding nature of fog nodes, their connectivity, mobility, management domain, hosting and network usage costs, etc.
- IaaS perspective – can be applied for a fog environment which offers infrastructure as a service to its tenants, and can be leveraged to deploy a mix of applications by various tenants.

6.8. Limitations

Test environment used to validate the proposed fog architecture assumes that there are sufficient resources available on selected fog nodes to satisfy the service requests of all users and IoT devices in their range, which may not be the case at all times, owing to limited fog resources, or due to security and privacy reasons. In such scenario, advanced service deployment algorithms are required, the discussion of which is beyond the scope of current paper.

We assume that the available fog nodes can be categorized into a small number of node types based on their resource characteristics. Further research needs to be done regarding identification of an optimal number of node types into which the available fog nodes can be categorized such that the differences between nodes of various types are more evident as compared to those belonging to same type, and are significant enough to differentiate various service requests to be assigned and served by nodes from different layers.

Table 5: Feature comparison of various fog architectures.

Feature	**Proposed Fog Architecture**	**Other Fog Architectures**
Structure	Multi-layered hierarchy.	Flat [99].
Management	Distributed.	Centralized [100].
Dynamicity / Elasticity / Scalability	Grow / Shrink. Effective tracking using Puddles.	Static [101]. Difficult to track in global database [102].
Node heterogeneity	Takes advantage of heterogeneity.	Treated homogeneous [102].
Generic architecture	Yes. No assumptions regarding system configuration. Can be readily tailored for specific environments.	No. Assumptions regarding node types [94], hosting costs [103], mobility [104], physical location [105], etc.
Knowledge of complete system state	Not required.	Required [106].
Resource pooling	Supported by Puddles	Not supported. Individual fog nodes only [100].
Disaster readiness	Supported by distributed local control.	Not supported [99].
Locality support	Supported by Puddles.	Not considered [103].
Cloud integration	Supported, but not required.	Required [99]. Not discussed [105].

6.9. Summary

In this paper, we have proposed a fog architecture with multi-layered hierarchy for large scale, heterogeneous, and widely distributed fog environments based on Puddles, which are groups of fog nodes in close vicinity, fully connected over homogeneous network connections. We have presented detailed discussion on various components of architecture and reasoning towards the same. We have discussed in detail how the proposed architecture supports various features of fog, along with validation for a sample smart city fog environment.

6.10. Open questions

Several questions, as listed below, are still open and need to be resolved prior to the deployment of fog environment in real-world based on architecture proposed in this paper.

How to define the range of individual IoT device, fog node, individual Puddle, etc.? What happens when there are multiple fog nodes in communication range of an IoT device? What happens when communication range of fog nodes overlap?

How to identify the fog layer to which a given fog node can be associated with, based on its resource characteristics? How to group the available fog nodes efficiently into Puddles? We have proposed one such approach using hierarchical clustering method. How to maintain hierarchy among fog layers for an IaaS environment?

How to size a Puddle, i.e. do we restrict membership of Puddle by maximum number of nodes, by maximum distance between nodes, by network range of nodes, etc.? How to handle node joins and leaves? What are the thresholds at which a Puddle is split or multiple Puddles are joined?

7. Dynamic Service Placement in Hierarchical Fog Environments

7.1. Abstract

Fog computing paradigm emerged as a promising solution to realize deployment of large scale IoT environments and low latency real-time services, leveraging large number of resource-constrained, heterogeneous fog nodes distributed across vast geographical areas and located closer to users and data sources, as compared to core cloud which is usually located at large data centers, far from users and IoT devices. In such an IoT environment deployed using fog infrastructure, application services need to be deployed in a timely, and efficient, manner towards realization of a scalable, and resource-optimized IoT platform. In an infrastructure environment with interplay of cloud and fog nodes, there is a need for efficient placement of services to satisfy their resource requirements as well as improve various factors such as node utilization, computation cost, communication cost, energy consumption, response time, availability, and load balancing. In this chapter, we propose a fully distributed approach to select a cost-efficient fog node considering both computation and communication costs, from the set of prospective fog nodes with available infrastructure resources to host the given application service.

7.2. Introduction

To facilitate the deployment of new generation IoT applications, cloud computing environments residing in large data centers are complemented with fog computing environments dispersed over large areas. Along with cloud nodes, fog nodes are leveraged to execute services. To support different types of IoT applications and their workloads, as well as user and sensor mobility, fog nodes of different resource and cost profiles are made available at different geo-locations resulting in varied user-perceived application latencies and data transfer costs.

Cloud computing paradigm allows sharing of physical infrastructure placed in data centers by tenants to deploy their own applications in a self-service manner, lease resources on cloud nodes as required by applications' resource and QoS requirements and pay just for the leased resources rather than owning the physical infrastructure, thus utilizing the physical cloud nodes in an efficient manner.

Fog computing paradigm extends the concept of cloud computing to include the physical nodes distributed over large geographic areas to support latency-sensitive and data-intensive applications which are not currently served by centralized cloud environments. In addition, fog environments allow deployment of new location-sensitive applications i.e. location-aware, and location-based (those of local value).

Cloud and Fog are best served for different classes (types) of applications. Three primary driving factors for fog deployment are:

- Lots of data – too much to send to cloud
- Localized data value – no need to send to cloud
- Latency sensitive data – can't afford to send to cloud within constrained time

Types of services which can be deployed on fog nodes:

- Computation-intensive services which cannot be executed on IoT devices themselves due to lack of resources.
- Computation-intensive service components which cannot be executed entirely on mobile phones and need to be executed partially on cloud
- Data intensive services which need access of large amounts of transient data and hence need high bandwidth connectivity to cloud.
- Latency-sensitive services which cannot sustain the highly variable Wide Area Network (WAN) latency to be deployed on cloud. Examples of devices with applications which need real time response are Pan-Tilt-Zoom (PTZ) video cameras, motion sensitive audio devices, location tracking devices, medical sensors, fingerprint scanners, etc.
- Location-sensitive services which leverage only local data and do not have much value globally.

7.3. Problem significance

Deployment of services in fog and cloud environments differ in various factors as listed below.

Management of application services deployed on fog nodes is more complex as compared to that in cloud environment for several reasons as discussed below.

Fog comprises heterogeneous nodes of varying compute, memory, network, storage, and energy, etc. resource characteristics to facilitate the execution of latency-critical, bandwidth-intensive, location-sensitive, and localized applications, whereas cloud nodes are considered

homogeneous i.e. there is no specific preference of a given node from the set of available cloud nodes in the data center to deploy a given application.

A cloud deployment is a flat structure with connected nodes allowing services to be deployed on any of the nodes with available resources. Contrary to cloud nodes, fog nodes are dispersed over large geographical areas and are identified by their physical location to facilitate the deployment of location-sensitive, latency-critical applications and those having only local value.

In cloud, physical nodes are added or removed from the system infrequently, and the event will be part of a planned maintenance operation by system administrators. On the contrary, fog is a dynamic environment where nodes join/leave at a higher rate due to mobility or battery-outage, leaving the services running on those fog nodes, and thereby users, in a vulnerable state. Hence service management approaches in fog need to be fast and efficient.

Fog nodes are less reliable than cloud nodes due to node mobility, and/or energy limitations. Thereby the need for continuous monitoring of fog environment and redeployment of services is higher than that in cloud, which generally has high reliability. Hence, fog service deployment and migration approaches need to be more efficient.

Cloud applications are usually compute-intensive, whereas fog applications are usually latency-sensitive or data-intensive. With cloud-based applications, data required for computation is usually generated within the data center and is available in its entirety to the compute nodes at negligible latency, as the compute and storage nodes will be connected over high speed data center LAN. Whereas in fog environment, data is generated by IoT devices distributed over very large geographic areas and incur different latencies when accessed from different compute

nodes. Thus, service management approaches for fog need to consider geolocation of fog nodes, IoT devices, and users in the system.

User requests are typically served entirely from a single data center, and always from the same primary data center irrespective of the user location, except during special scenarios such as high overload or business continuity during disaster scenarios. Note that these periods are brief and user requests are served from the primary data center upon restoration of normal conditions, whereas such is not the case with dynamic fog environment. Mobile users may need to be served by different fog nodes depending on their geolocation, thus requiring (re)placement of same application service on different fog nodes.

Number of nodes in fog is of several orders of magnitude more than that in cloud. In addition, wide geographic dispersion of fog nodes make it difficult and sometimes infeasible to maintain entire system state at a centralized authority, thus service management approaches may not have complete system state available and should be developed considering local or incomplete system state, leading to distributed decision making. Service management in cloud environment differs in that all nodes are physically available in the same data center and the cloud broker has complete knowledge of services deployed in the system making centralized solution approaches feasible, whereas it would be cost-prohibitive and unnecessary to maintain such a data store in a large scale distributed fog environment. Thus, service management solutions for fog environment should be designed accordingly.

7.4. Motivation

In such a complex fog environment with widely dispersed users, sensors, and nodes, varied application requirements, as well as resource availability, resource configuration, and cost characteristics, there is need for efficient approach to deploy services for better utilization of

resources as well as reduce overall cost. Leveraging the infrastructure resources leased from service provider, a fog tenant can manage the deployment of services on leased nodes in an independent manner. To facilitate a cost-efficient execution of services, there is a need to identify cost-efficient subset of fog nodes with available resources from those leased by tenant to deploy a given set of services.

Considering the huge number of users, applications, service requests, IoT devices, and fog nodes, service deployment approach should be scalable. Additionally, solution approach should support mobility of users, and IoT devices, as well as heterogeneity in fog. Efficient placement of services on fog nodes should satisfy their QoS needs along with minimizing cost, and energy consumption. Frequency of service deployment requests in fog is higher as compared to those in a cloud environment due to various reasons such as comparatively low resource configuration of fog nodes resulting in faster expiry of services deployed, energy-constrained nature of fog nodes, mobility of IoT devices, fog nodes, and end users, unmanaged and less reliable fog nodes.

As complete system state is not available at any entity, centralized solution approach is infeasible for fog environment. Thus, solution should be preferably a distributed approach, owing to the dispersed nature of fog environment.

7.5. Contributions

In this chapter, we present the following.

- Description of the problem of service placement in Infrastructure-as-a-Service (IaaS) based large-scale fog environments.
- Description of the proposed approach to solve the problem in a distributed, scalable, and efficient manner.

- Details of the implementation of solution in a simulated environment.
- Demonstration of the efficacy of the solution using a simulation environment.

7.6. System model

Towards the scope of this paper, we assume a Fog Infrastructure as a Service (FIaaS) system [107] with several entities, the characteristics of which have been described below. System entities such as devices, compute nodes, network nodes, and users, are identified by their GPS location. No two system entities share the same GPS location.

Devices. These are the data generating entities in the system. The generated data is sent as input to corresponding application instances for processing. Devices are connected to nearest network node over wireless network. The data generation characteristics of devices such as amount of data generated and its frequency, are defined by the application profiles which leverage the produced data.

Compute nodes. These are the entities which host a given application instance and execute it to process the input received from devices and generate output data. These are referred to as fog nodes in the remainder of the document. Fog nodes have predefined CPU and network resource configurations, and cost characteristics. Fog nodes in the system belong to one of the small number of predefined categories. All fog nodes belonging to a specific category have similar CPU and network resources, as well as cost characteristics i.e. for application execution of million instructions (mi), and a kilobyte (KB) of data transferred.

Network nodes. These are the entities which interconnect the devices, users, and compute nodes. Various system entities are accessible from each other over one or more network links. Network nodes have predefined bandwidth capacities, as well as cost characteristics e.g. for a KB of data transferred from one connected neighboring entity to the other.

Users. These are the entities which receive the output data from application service residing on compute nodes.

Service request. Service requests are generated by users when they need access to a new application instance.

7.7. Problem Description

In a large-scale fog environment with widely dispersed fog nodes of various resource configurations and cost characteristics, there is a need to select a cost-efficient fog node which can host a given application service with specific resource requirements and QoS characteristics.

In this section, we provide a detailed description of the research problem, along with assumptions, constraints, and objectives.

7.7.1. Assumptions

Towards the problem of application service placement in fog computing environments, we assume the following:

- All entities of the system, i.e. data sources (devices), data consumers (users), fog compute nodes, network nodes, and any management nodes in system are physically stationary.
- All IoT devices are connected over wireless network to rest of the system via the nearest Wireless Access Point (WAP).
- No entity has knowledge of the current state of the entire system to select a cost-optimal fog node to host the given application service. Hence, centralized decision making with full system state knowledge is infeasible.
- Service requests from users are independent of one another.

7.7.2. Objectives

Fog computing environment facilitates processing of data generated by devices close to the data sources, by hosting and executing the corresponding applications on fog nodes in vicinity. Processing of input data by applications consumes infrastructure resources on fog nodes.

Cost efficiency. As fog nodes vary in their resource configurations, the cost of execution of an application service instance differs, depending on the fog node which is hosting the service. Additionally, as fog nodes are located at varying distances from data sources, i.e. devices and data consumers, i.e. users, and are accessible over one or more network hops, cost of transfer of corresponding application's input and output data will also vary. Thus, the primary objective of identifying a fog node to host a given application service is cost-efficiency in terms of both execution cost as well as data transfer cost.

Distributed solution. As discussed earlier in the paper, centralized knowledge of current system state i.e. current state of each network entity, including location, workload, and availability of free resources, is infeasible in a widely distributed large-scale fog environment. Service placement approaches depending on such complete information are not practical. Hence, our objective is to propose a fully distributed service placement approach with no knowledge of complete system environment.

7.8. Problem Definition

Total cost of hosting an application service on a given fog node is the sum of execution cost of application service on fog node as well as cost of data transfer at each network hop on the path between device generating input data for the service to the fog node and those on the path

between fog node and the user receiving processed output data from the application service. Service pricing approach considered in this paper is described in detail in [108].

Total cost = Execution cost on fog node + Σ (Data transfer cost at each network hop on inbound path) + Σ (Data transfer cost at each network hop on outbound path).

Given a set of fog nodes, devices, and users interconnected as per given network topology and accessible from each other over one or more network links, as well as service execution cost and data transfer cost on each of the network nodes, goal of service placement approach is to find a cost-efficient fog node which minimizes the total cost of hosting the service, while satisfying the resource requirements of application service such as compute capacity in MIPS and network bandwidth requirement in KB/s, as well as QoS requirements such as maximum response time (milliseconds).

7.9. Solution Description

The proposed approach for placement of application services in large-scale fog environments includes two phases, the details of which are as follows.

7.9.1. Phase-1: HAFA Architecture

Given a set of fog nodes characterized by their GPS locations and belonging to a small number of predefined categories, with all nodes belonging to a specific category having similar resource configurations and execution cost characteristics, we organize them into a logically connected hierarchy, as described below.

1. Fog nodes belonging to the same category comprise a specific fog layer. We assign ids to fog layers such that the resource configuration of nodes belonging to fog layer *i* is less than those of nodes belonging to fog layer *i+1*.

2. Fog nodes comprising a specific fog layer are grouped such that the maximum geographical distance between any two nodes belonging to a group is less than the specified limit. Each group is referred to as Puddle. Thus, fog nodes comprising a specific fog layer are grouped into a set of Puddles using Agglomerative Hierarchical Clustering Complete Link method based on inter-node distances. Further details regarding clustering approach are available in section 0.
3. Fog nodes comprising a given Puddle are managed by a local authority, referred as PuddleHead, which is one of the nodes belonging to the Puddle itself. The PuddleHead keeps track of various events such as node join/leave out of the Puddle, application services hosted by each node within the Puddle, availability of free resources on each node in the Puddle, etc.
4. Each Puddle is logically interconnected to the nearest Puddle which belongs to an adjacent layer. For example, Puddle pi belonging to fog layer i is logically connected to Puddle q_{i+1} belonging to fog layer $i+1$, which is the nearest of all the Puddles belonging to layer $i+1$. The distance between two Puddles p_i and q_{i+1} belonging to layers i and $i+1$ respectively, is defined as the maximum distance between any node a_i belonging to Puddle p_i and node b_{i+1} belonging to Puddle q_{i+1}.
5. Thus, if the number of fog nodes and/or Puddles in higher fog layers are fewer than those in lower fog layers, then the logical organization of fog nodes as described above, can be represented in the form of a tree, referred to as PuddleTree, as depicted in Figure 20. It would be so if there are fewer fog nodes

with large resources than those with smaller resources, which would be normally expected.

Logical organization of fog nodes as described above, is referred as Hierarchical and Autonomous Fog Architecture (HAFA) [84] and is discussed in detail in 6.4. Towards the scope of this paper, we have assumed that fog nodes are physically stationary, and thus their GPS locations do not change. Hence, the logical organization of fog nodes, as described above, stays valid during course of the system simulation. Therefore, the procedure of formation of links is performed only once during test system setup.

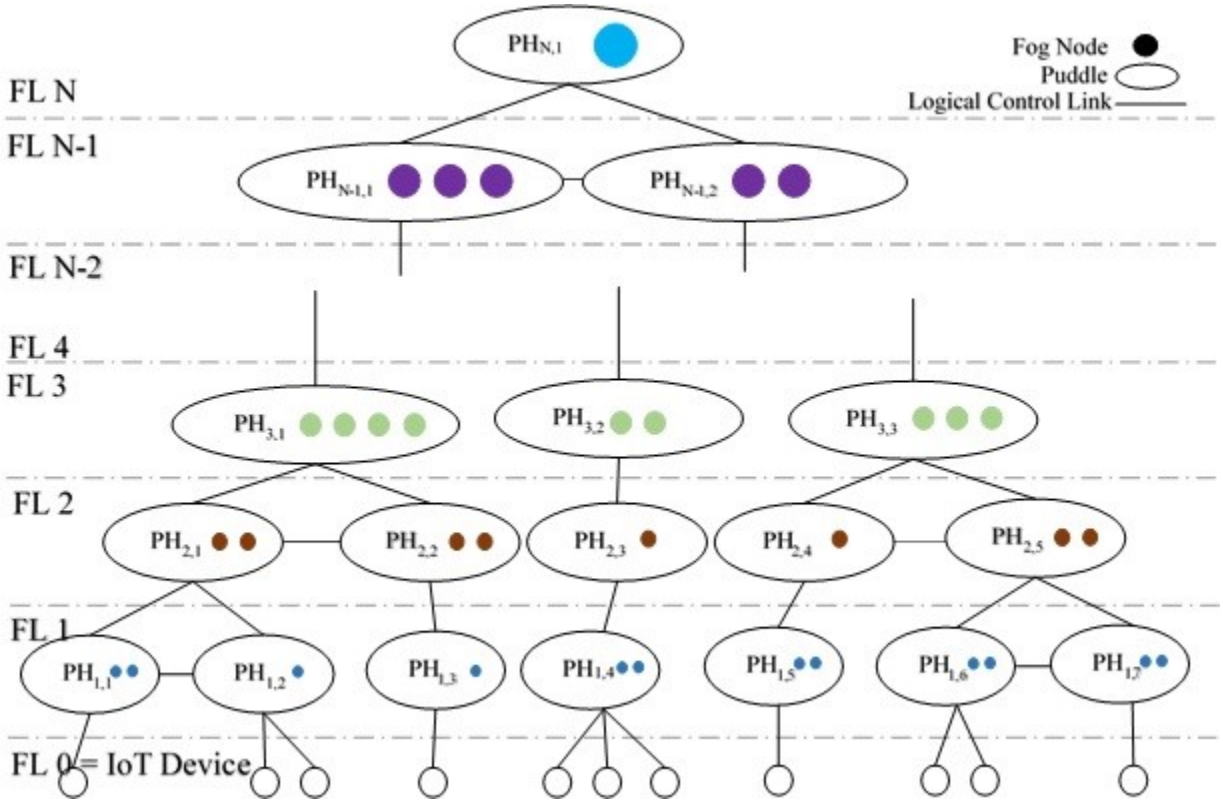

Figure 20: PuddleTree comprising fog nodes with varied resource configurations.

7.9.2. Phase-2: HAFA Orchestrator

After organizing the fog nodes in system as described above, the resulting logical structure i.e. PuddleTree and Puddle memberships information can be leveraged towards efficient search of a cost-efficient fog node to host an application service with given resource requirements and QoS characteristics.

In a fog computing environment, application service placement request is generated by user, which includes service resource and QoS requirements along with input data generating device location as well as user location information. The procedure described in Algorithm HAFA-O is followed to identify a cost-efficient fog node to host the given service. Service request generated by user is submitted to the local fog broker, which executes the service placement approach discussed in this section.

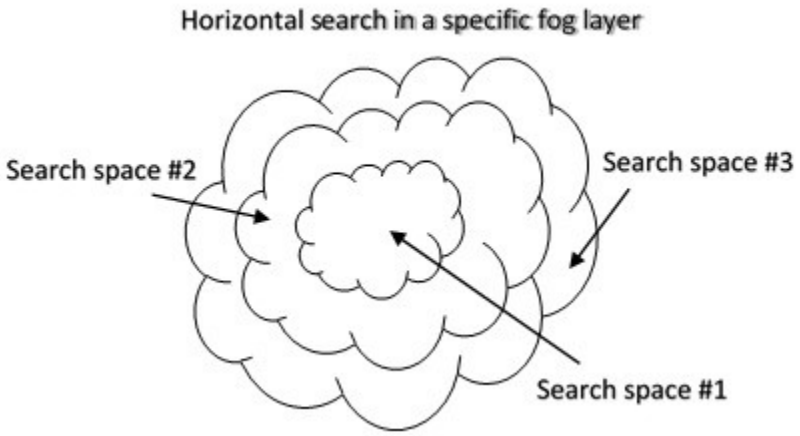

Figure 21: HAFA Orchestrator: Pictorial representation of node selection procedure per fog layer (Horizontal search).

Note that, the nearest fog node belonging to layer *i* w.r.t. a given device/user is the one located at least geographic distance to device/user from all the nodes belonging to layer *i*. Local Puddle for layer *i* is the Puddle which comprises the nearest fog node belonging to that layer *i*

and the corresponding PuddleHead is the local PuddleHead. PuddleHead Id is maintained as part of Puddle information and all member fog nodes of the Puddle are aware of it. Fog broker instance may be hosted by PuddleHead. Fog broker runs the orchestrator approach to identify the cost-efficient fog node.

Algorithm HAFA-O:

1. For each fog layer *i*, identify the local Puddle and forward the request to its PuddleHead, which returns the nearest fog node belonging to layer *i* with sufficient resources to host the given service, by following the sequence below.
    a. Is there at least one node belonging to this Puddle which can serve this request?
    b. If yes, then among the candidate nodes with sufficient free resources, identify the geographically nearest node to the device generating the service request.
    c. If no, then expand the search space to the immediate neighbors of previous search space by forwarding the request to parent PuddleHead, which will propagate the request down to its children until the request reaches PuddleHeads belonging to layer *i*. Go to step 1a unless the search has already reached the root PuddleHead.
    d. If the root PuddleHead has already been reached, then this implies that no fog node belonging to layer *i* with sufficient free resources is identified. Hence, request is not schedulable on this fog layer.
2. Among the prospective fog nodes identified, a maximum of one per fog layer, select the cost-efficient fog node. Reserve resources on the selected fog node and

deploy given application service on it. Update availability of free resources at the corresponding PuddleHead.

As described in this section, the proposed solution approach leverages the logical PuddleTree to identify a cost-efficient fog node to host a given application service, in a distributed manner. The fog node selection procedure followed by HAFA orchestrator is shown pictorially in Figure 22.

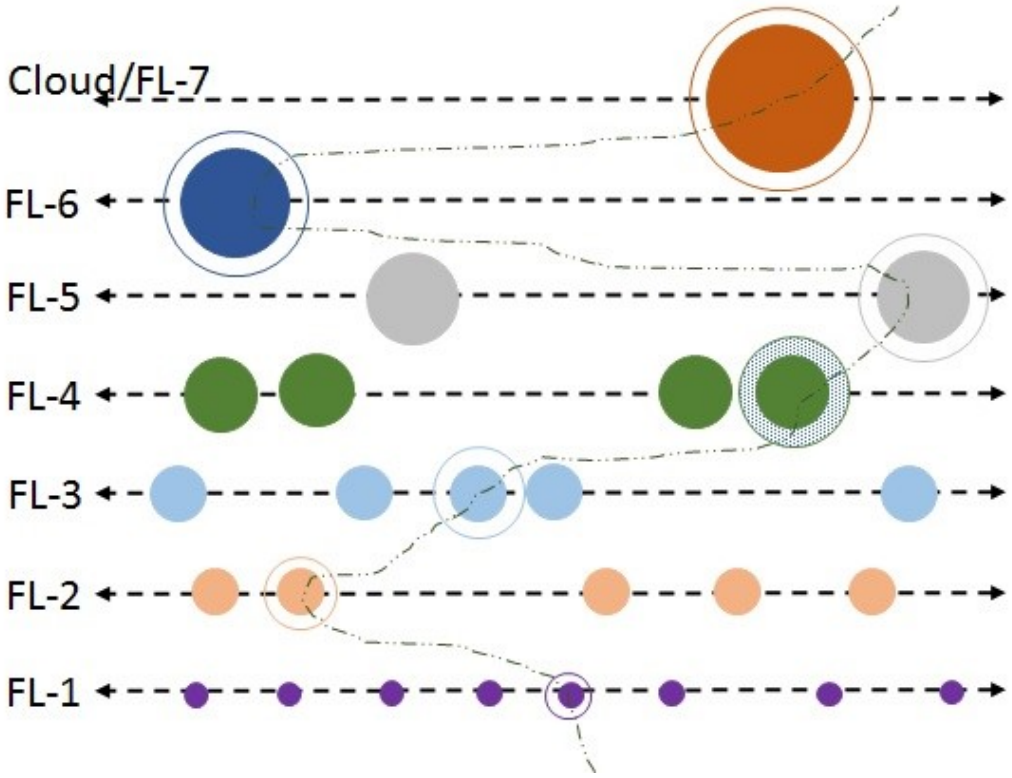

Figure 22: HAFA Orchestrator: Pictorial representation of node selection procedure (Vertical search).

7.10. Analysis

As shown in Figure 22, HAFA orchestrator selects nearest node by distance from each fog layer and has sufficient resources available to satisfy the application resource and QoS

requirements. It exchanges messages with Peer PuddleHeads and parent PuddleHeads to expand the search beyond local Puddle, which contributes to the algorithm overhead.

Here are the highlights of HAFA placement approach:

- Distributed approach.
- Leverages local state knowledge only.
- Expands search space only as far needed, until request is successful, or search space is exhausted.
- Maintains locality.
- Better network utilization.
- Cost-efficient, considering both computation and communication costs.

Limitations of HAFA approach are as follows:

- Based on the premise that the set of fog nodes can be categorized into a small number of fog node types.
- Fog nodes of each category, referred as layer, are uniformly dispersed in the fog environment.
- Fog nodes belonging to each layer have same execution cost.
- Fog nodes are physically stationary.
- Fairness and balancing of workload on fog nodes is not considered.

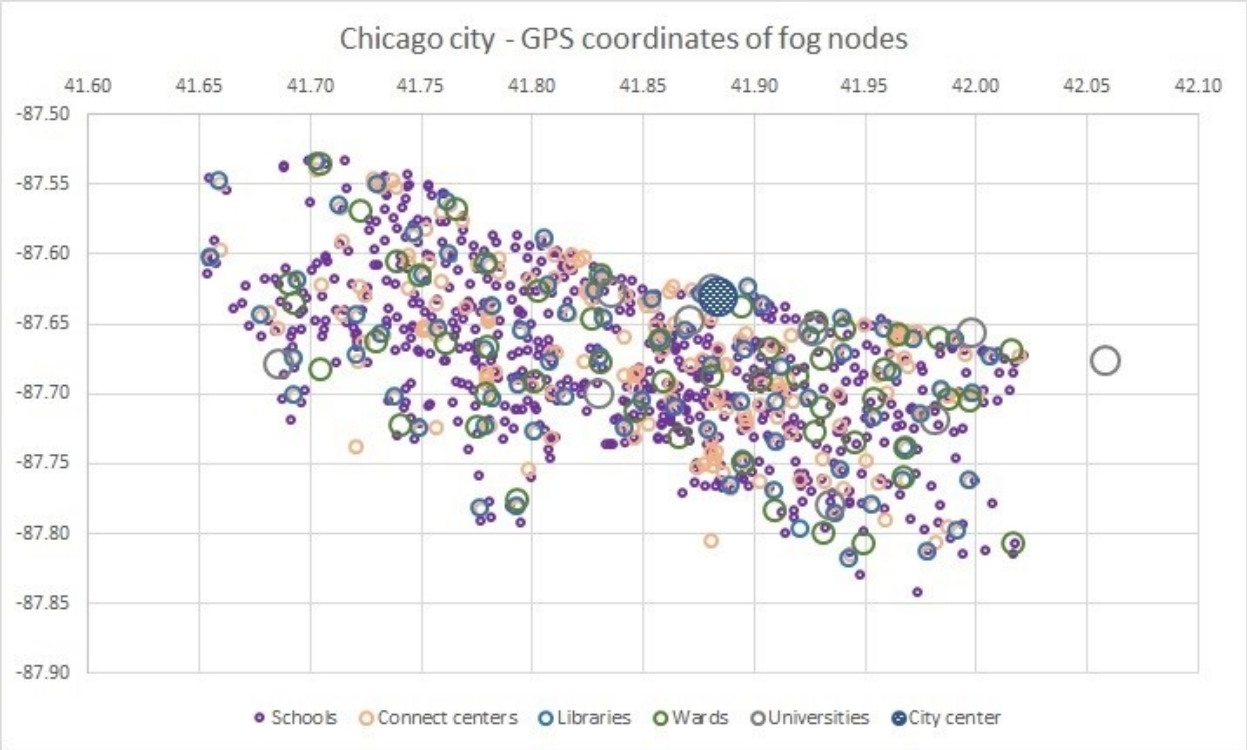

Figure 23: Test environment: Chicago city - GPS Coordinates of fog nodes.

7.11. Experimental Validation

In this section, we provide a detailed description of validation tests performed as well as the observations from analysis of test results.

7.11.1. Objective

The goal of this testing effort is to show that our solution approach for service placement is scalable and efficient, in large-scale heterogeneous fog environments distributed over large areas. The following metrics were used for evaluation.

- Compute resource utilization
- Network resource utilization
- Response time
- Percentage of successful requests

- Cost of service execution
- Network congestion

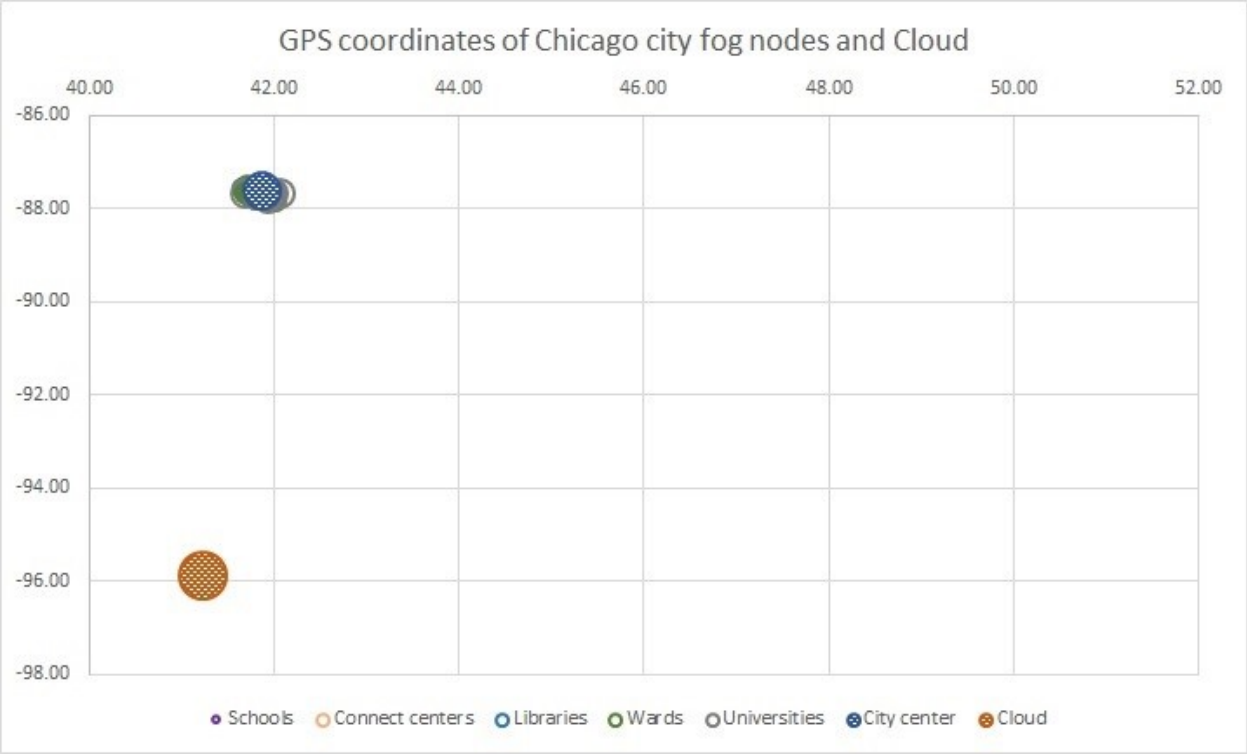

Figure 24: Test environment: Chicago city - GPS Coordinates of fog nodes and Cloud.

7.11.2. Simulator – PFogSim

Tests were executed on a simulator, called PFogSim. PFogSim is an event-driven simulator developed in Java to simulate the execution of applications in a distributed large-scale fog environment with thousands of fog nodes, data generating and consuming devices, as well as users, connected over one or more network links using intermediate network nodes.

PFogSim was extended from EdgeCloudSim [109] to reflect the unique features of fog environments such as heterogeneity, location-awareness, interconnecting network topology, mobility of devices, users, and fog nodes. It has modular design, which allows testing various approaches to service placement in fog computing environments, service pricing approaches, etc.

Several metrics are captured during test execution. The simulator facilitates analysis of test results by auto-generation of graphs using a MatLab interface, from test logs comparing selected approaches and metrics. Detailed description of various features available on PFogSim are provided in Chapter 9.

7.11.3. Implementation

The service placement approach proposed in this paper was implemented in Java and integrated with the PFogSim simulator. Solution Phase-1, which consists of creating the logical PuddleTree, is performed only once during test environment setup, whereas the Phase-2 of solution is invoked each time a new service request is initiated, and is executed by the local fog broker in a distributed manner, with no knowledge of complete system state. Figure 25, Figure 26, Figure 27, Figure 28, and Figure 29 show the number of member fog nodes associated with each Puddle from various fog layer, resulting from execution of solution Phase-1.

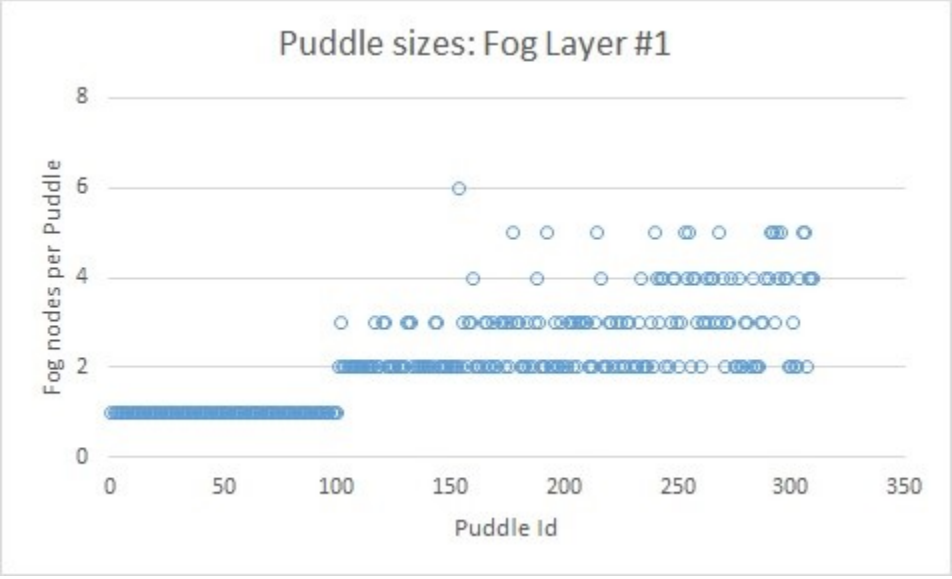

Figure 25: HAFA implementation: Puddle sizes for fog layer-1.

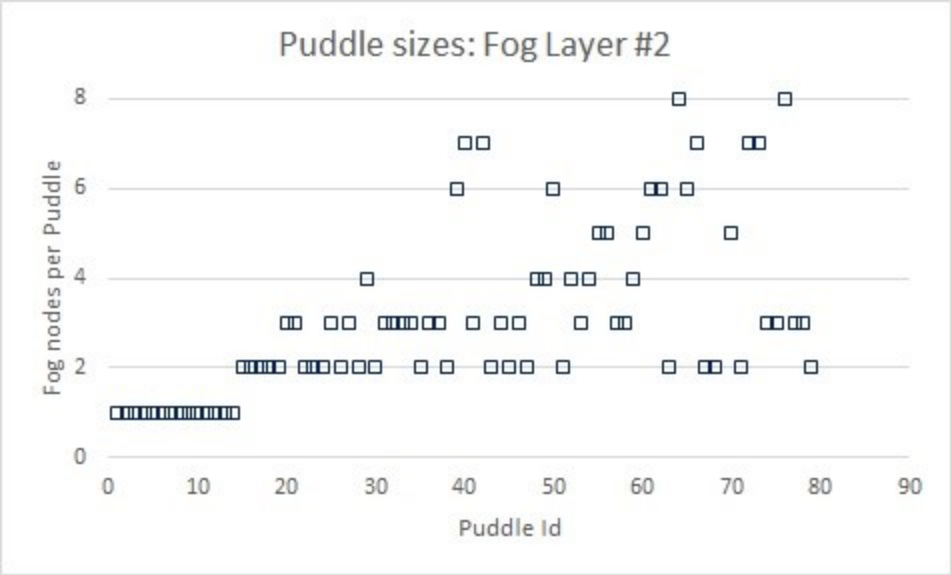

Figure 26: HAFA implementation: Puddle sizes for fog layer-2.

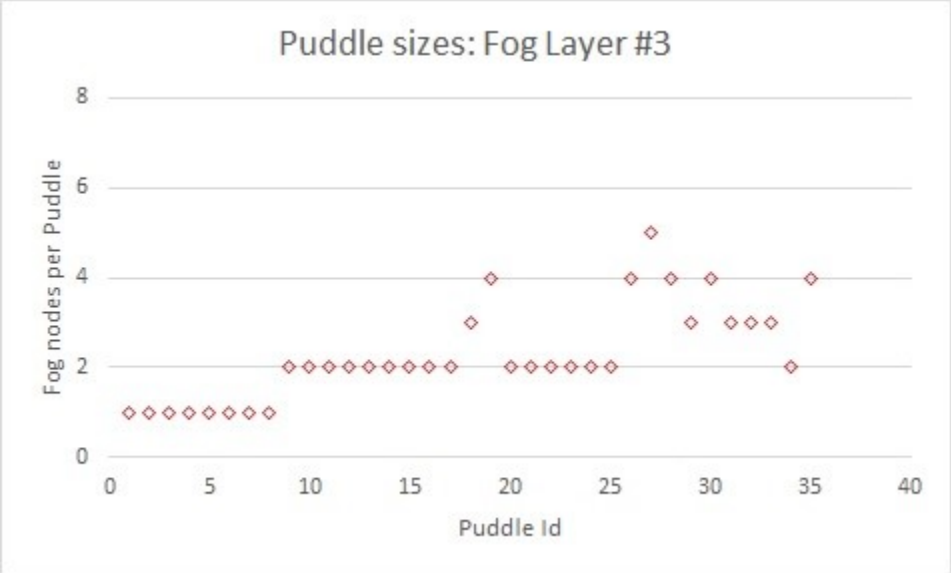

Figure 27: HAFA implementation: Puddle sizes for fog layer-3.

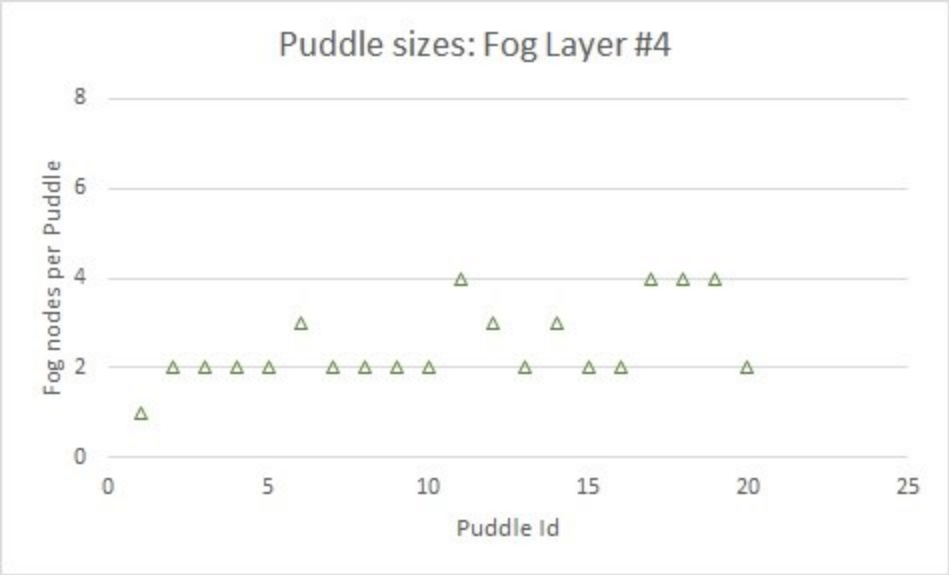

Figure 28: HAFA implementation: Puddle sizes for fog layer-4.

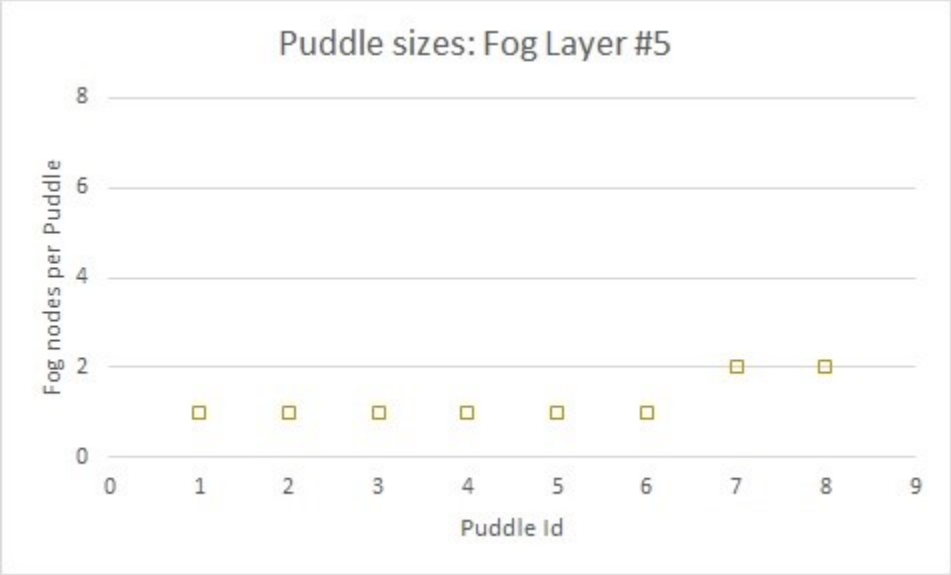

Figure 29: HAFA implementation: Puddle sizes for fog layer-5.

7.11.4. Test environment

We have considered the Smart City application domain to evaluate the proposed resource and service management approaches.

Table 6: Test environment: Fog node capacities for various layers.

Layer	Fog node	MIPS	Cores	Total MIPS	Times FL-1 node	Times Lower layer node
FL-1	School	54400	1	54400	1	N/A
FL-2	Community	217600	1	217600	4	4
FL-3	Library	326400	2	652800	12	3
FL-4	Ward	544000	7	3808000	70	6
FL-5	University	816000	71	57936000	1065	15
FL-6	City center	1305600	286	373401600	6864	6
FL-7	Cloud	13056000	28672	374341632000	6881280	1003

Table 7: Test environment: Fog node task execution costs.

Layer	Exec. Cost per Second	Exec Cost per Second per Core	Exec. Time per 2000 MI Task (Seconds)	Exec. Cost per 2000 MI Task
FL-1	0.013194444	0.013194444	0.036764706	0.000485090
FL-2	0.013194444	0.013194444	0.009191176	0.000121272
FL-3	0.013194444	0.006597222	0.006127451	0.000080848
FL-4	0.013194444	0.001884921	0.003676471	0.000048508
FL-5	0.013194444	0.000185837	0.00245098	0.000032339
FL-6	0.013194444	0.000046134	0.001531863	0.000020212
FL-7	0.006597220	0.000000230	0.000153186	0.000001010

We use Chicago city data set [110], which has the GPS coordinates of schools, universities, libraries, ward offices, connect centers, city hall, etc. to simulate a fog environment with 1073 fog nodes dispersed over 237 sq. miles area of Chicago, which includes fog nodes located one each at 679 schools, 253 connect centers, 80 libraries, 50 city wards offices, 10 universities, 1 city hall, and 1 cloud data center located at approx. 500 miles from city hall. Fog nodes placed at different locations of similar premises belong to the same fog layer and are of similar resource configuration, e.g. fog nodes placed at all city wards are of similar capacity.

Table 8: Test environment: Fog Network node link capacities.

Layer	Network Node Type	Network Node Id	Link capacity	Link capacity (Kbps)
FL-1	Edge Router	R1	1 Gbps	1048576
FL-2	Edge Router	R1	1 Gbps	1048576
FL-3	Border Router	R2	10 Gbps	10485760
FL-4	Border Router	R2	10 Gbps	10485760
FL-5	Border Router	R2	10 Gbps	10485760
FL-6	Core Router	R3	100 Gbps	104857600
FL-7	Core Router	R3	100 Gbps	104857600

Each of these locations is assumed to host a network router to facilitate device and user connectivity, as well as connect other network components. Fog node at each school, connect center, and library was connected to an individual edge router, fog node at each city ward office and each university was connected to an individual border router, which also connects to all the nearest edge routers, and fog node at city hall and cloud data center were connected to an

individual core router. Each border router was connected to two nearest border routers as well as the core router at city hall. Core routers at city hall and cloud data center were connected over direct link. Each network device is also considered as a Wireless Access Point (WAP) so that devices and users can connect over wireless network and hence support their mobility. Fog network was simulated in the above manner to closely resemble a prospective smart city fog infrastructure deployment.

Corresponding to the seven categories of fog nodes, HAFA architecture is created as a seven-layer hierarchy of fog nodes, ranging from small edge nodes belonging to layer-1 to layer-7 for centralized cloud environment with logically infinite resources. Computation capacities of fog nodes belonging to various layers are listed in Table 6.

Compute cost at lower layer fog nodes is higher as compared to that at higher layer fog nodes, due to scale of establishment and node resource configurations. Computation costs considered for various fog nodes in our test environment are listed in Table 7 for Augmented Reality (AR) application profile with an average task size of 2000 million instructions (MI).

Table 9: Test environment: Fog Network node data transfer costs.

Network Node Id	**Link capacity (Kbps)**	**Lease cost ($ / Month)**	**$ / Mb transferred (Scaled 100 times)**
R1	1048576	88.23	0.000032764
R2	10485760	151.67	0.000005632
R3	104857600	646.51	0.000002401

Communication cost increases as number of network hops increase in path from device to fog node hosting the application service. Three types of Cisco routers are assumed to comprise fog network configuration. The three types considered are Cisco ASR 901 1G Router, Cisco ASR 901 10G Router, and Cisco ASR 1013 100G Router referred as of types R1, R2, and R3 respectively in rest of the document. Network node link capacities are listed in Table 8. The cost of data transfer on each of these routers is inferred from their monthly rental rate offered by Cisco for leasing and are summarized in Table 9.

7.11.5. Test system setup

One VM is created on each fog node and is configured with all the available resources on that node. Individual application services are deployed on the VM and assigned resources as specified in service request.

7.11.6. Assumptions

Following are the assumptions made in regards with the system configuration, and entities in the system for the purpose of test execution.

- Sufficient compute resources are available in the entire system to ensure that all application requests can be served and all tasks can be executed.
- Sufficient network resources are available in the system such that all application data and service requests can be transferred to other nodes in the system if the local node does not have sufficient free resources.
- Data producers i.e. IoT devices, sensors, or other data sources, as well as data consumers i.e. IoT devices, actuators, or users, do not have sufficient resources to host and execute the application services by themselves, thus necessitating the availability of a cloud/fog node for the purpose.

- Data source i.e. device is co-located with the user i.e. data consumer.
- We assume that all applications are simple, i.e. they execute and perform one function (all the modules together). Complex applications which perform multiple activities can be split into several simple applications can be split into several simple applications. E.g. healthcare sensors generate data which can be processed and may be used as input for user interface or sent to an analytics application. Each has a different QoS requirement and hence are considered as different applications.
- All fog nodes are Wi-Fi Access points (WAPs). Hence, IoT devices can assumed to connect to the nearest WAP which is used to gain access to rest of the fog network. Router is co-located with fog node for ease of test execution.
- Devices, fog nodes can be physically stationary or mobile.

7.11.7. Test approach

All service requests were generated for individual application services. Input data is generated by devices at the rate and pace defined by their associated application service profiles, and is submitted to the assigned fog node, which is hosting the service, in the form of a task. Each task is defined with the same resource and QoS requirements as its associated application profile. Upon receipt of a task, the assigned fog node processes it, generates output, and sends the output to the corresponding user by routing it over the network.

7.11.8. Application profiles

Tests were performed for four different application profiles, Augmented Reality (AR), Cognitive Assistance (CA), Machine Learning (ML), and Remote Healthcare (RH), the details of which have been given in Table 10. The application profiles are selected to represent various

classes of applications such as computation-intensive, bandwidth-intensive, latency-tolerant, and latency-critical. Following is a description of application parameters listed in Table 10.

Table 10: Test application profiles

Parameter	**AR**	**CA**	**ML**	**RH**
Poisson_interarrival (sec)	5	5	5	5
Delay_sensitivity (sec)	5	0.370	10000	0.100
Data_upload (KB)	1500	1500	1500	4
Data_download (KB)	25	25	25	1
Task_length (MI)	2000	2000	6000	300
Required_cores	1	1	1	1

Poisson_interarrival. This parameter defines the average inter-arrival time between the submissions of application tasks by a device. It is specified in seconds. For instance, if the Poisson_interarrival time is set to 5 seconds, then over the course of a simulation test run of 30 minutes, an expected number of 30 minutes * 60 seconds/minute / 5 seconds = 360 tasks will be generated by each device and submitted to the executing fog node.

Delay_sensitivity. This parameter specifies the maximum response time acceptable for a given task, i.e. the total time taken for its execution and transfer of input and output data over the network. It is specified in seconds. A task is marked as failed if the device/user does not receive the output data from executing fog node within the time duration specified by this parameter.

Data_upload. This is the average input data size for each task submitted by device and transferred from device to executing fog node. It is specified in kilobytes (KB).

Data_download. This is the average output data size received by user for each task and transferred from executing fog node to device. It is specified in kilobytes (KB).

Task_length. This is the number of instructions to be executed per task by fog node. It is specified in million instructions (MI).

Required_cores. This is the number of cores required for execution of a given task.

7.11.9. Evaluation criteria

We have captured several metrics as listed below, from simulation tests to show the performance of proposed service placement approach in large fog environments. Note that the metrics are averaged only for the tasks successfully executed.

- **Task count**. Total number of tasks successfully executed by all fog nodes in system.
- **Task count per fog layer**. Total number of tasks successfully executed by all fog nodes belonging to a given fog layer in system.
- **Task success/failure percentage**. Percentage of total submitted tasks which were successfully executed or failed to complete execution. Execution of tasks may fail for various reasons such as lack of compute resources, lack of network resources (bandwidth), application latency constraints, device mobility, fog node mobility, queuing delay at fog node, congestion delay at network node, and unfinished tasks due to end of simulation time.
- **Average cost**. Cost incurred by each task is the sum of execution cost at fog node and data transfer cost at each network node en route from device to fog node hosting the application service.

- **Average geo-distance**. This is measured as the great circle distance between device and fog node hosting the corresponding application service. Distance between the two GPS locations is calculated using Haversine formula. Note that the stationary devices are assumed to be co-located with local fog nodes in our test environment and hence local nodes are located at a distance of zero from devices.
- **Average network distance**. This is measured as the number of network hops in the path from device to fog node hosting the corresponding application service. Note that the devices and fog nodes are not accessible from another by direct links, but are accessible only on the network and hence are connected directly to a network node. Thus a local fog node is accessible from device at one hop distance, for instance.
- **Average network delay**. This is measured as the sum of network data transfer delays incurred over each link and at each node on the network path between device and fog node hosting the corresponding application service. It includes propagation delay on each network link, as well as network processing delay and congestion delay at each network node along the path. Propagation delay is proportional to the length of the network link. It is calculated as 0.01 millisecond/kilometer [111]. For instance, if two nodes are located at 10 kilometer apart, then propagation delay on that link is calculated as 0.01 ms/km * 10 km = 0.1 millisecond. Network processing delay of 20 milliseconds is assumed towards route processing at each network hop along the path. Congestion delay depends on the number of tasks leveraging a network link at a given instant and their

input/output data size (KB). For instance, say each device has only one active task at any instant with average data transfer requirement of 1500 KB. If 100 such devices are exchanging data over a specific network node with capacity 10 Gbps, then congestion delay is calculated as 1500 KB * 8 bits/byte / 10 Gbps * 100 devices = 114 milliseconds.

- **Average processing time**. This is measured as the sum of queueing delay and task execution (compute) time taken at fog node hosting the specified application service. Processing time depends on the size of task (MI), compute capacity of fog node (MIPS), and the number of CPU cores required by the task for its execution. For instance, if the executing fog node capacity is 54400 MIPS and application task size is 2000 MI, then expected processing time taken by the task is calculated as 2000 MI / 54400 MIPS = 36 milliseconds.
- **Average service time**. This is measured as the sum of processing time at fog node and network delay along the network path between mobile device and fog node hosting the corresponding application service. This metrics signifies the total delay experienced by user after submission of task to the system.
- **Host utilization**. This is measured as the percentage ratio of the number of MIPS utilized by devices at a given fog node (host) to the total MIPS available at that node at a given instant.
- **Average host utilization per fog layer**. This is measured as the average of host utilization percentages for all fog nodes belonging to a given fog layer. This is a valid measure as we assume that all nodes belonging to a given fog layer have same MIPS capacities.

- **Average host utilization**. This is measured as the average of average host utilization percentage observed per fog layer. We assume that nodes belonging to different fog layers have different MIPS capacities.
- **Network utilization**. This is measured as the percentage ratio of the network link capacity (Kbps) utilized by devices to the total capacity available at that network link at a given instant.
- **Average network utilization per fog layer**. This is measured as the average of network utilization percentages for all network nodes belonging to a given fog layer. This is a valid measure as we assume that all nodes belonging to a given fog layer have same network capacities.
- **Average network utilization**. This is measured as the average of average network utilization percentage observed per fog layer. We assume that nodes belonging to different fog layers have different network capacities.
- **Average hosts searched**. This is measured as the number of fog nodes considered by a device towards placement of its corresponding service instance. For static placement approaches, i.e. the ones which host all services at the same fog node always, irrespective of the device characteristics such as its location, and its application resource characteristics, the number of hosts considered is always one. For centralized approaches, i.e. the ones which require the knowledge of current state of all prospective nodes towards selection of fog node for service placement, the number of hosts searched is always the total number of prospective nodes in the set. HAFA orchestrator, being a dynamic and distributed approach, the

number of hosts searched varies depending on the location of device initiating the request and the current workload in system.

- **Average messages exchanged**. This is measured as the number of messages exchanged in system towards identification of a fog node for hosting an application service instance for a given device. The request and response messages are counted separately. Hence, for static and centralized approaches, number of messages exchanged is twice the number of prospective fog nodes considered. Whereas for HAFA orchestrator, messages are exchanged to forward the request to sibling PuddleHeads and to parent PuddleHeads, as placement decision is made by PuddleHeads instead of individual nodes themselves.

7.11.10. Comparative approaches

We have implemented several service placement approaches to quantitatively compare the performance of each of them against the HAFA orchestrator. A brief description of the approaches is provided below.

Edge orchestrator. Dynamic approach. This approach deploys the application services only on edge nodes i.e. fog nodes belonging to layer-1, which are of smallest resource configuration in the system. Among all nodes belonging to fog layer-1, the one reachable at least geographical distance from device and/or user is selected to host the corresponding application service. In our test environment, the layer-1 fog nodes being more in number, have higher chance of being in vicinity of a requesting device and hence are accessible at lower latency, on average as compared to those from higher fog layers. Service requests whose resource and QoS requirements cannot be satisfied using only layer-1 fog nodes are rejected. Hosting costs may be

higher. This approach is not network topology aware and avoids the necessity of keeping track of changes in network topology.

Local orchestrator. Static approach. This approach deploys requested application services only on the fog node accessible at least distance from the device and executes corresponding tasks on the same node. The local node may belong to any of the fog layers. If the local fog node does not have sufficient resources, service request is rejected. Mobility of device, user, and/or fog node may result in failed tasks, depending on application QoS specifications.

City center orchestrator. Static approach. This approach attempts to deploy all application services only on the fog node located at City center irrespective of accessibility from requesting devices and executes tasks on the same node. If the fog node does not have sufficient resources or if service QoS requirements cannot be satisfied, service placement request is rejected. Mobility of device, user, and/or fog node may result in failed tasks.

Cloud orchestrator. Static approach. This approach deploys all application services and executes corresponding tasks on cloud nodes. All application services will be deployed in the cloud, resulting in cheaper hosting cost, but with higher data transfer costs as well as without any assurance towards satisfaction of application QoS for latency-critical applications. Latency-critical or QoS-sensitive application requests may fail.

Centralized orchestrator. Dynamic approach. This approach identifies the cost-efficient fog node to host the application service and execute corresponding tasks, leveraging the knowledge of complete system state at any given instant during the simulation. The centralized orchestrator is assumed to have knowledge of the current state of the entire system i.e. all devices, nodes, and users in the system, including their locations, resource configurations,

available free resources to serve new requests. Thus, centralized orchestrator always selects the best possible node and the best possible links for deployment of a given service. As it is obvious, this approach is infeasible in real-world fog environment owing to the dynamism and geographic dispersion of individual nodes, among other reasons. Hence, this approach is used as a reference to compare the performance of other approaches.

Centralized orchestrator considers all nodes in the system as prospective nodes towards selection of cost-efficient fog node for hosting a service. Towards this goal, it calculates the total cost to be incurred by a task from given device on each of the fog nodes as the sum of computation cost on the fog node and communication cost comprising data transfer cost on each network node along the path from device to fog node. Note that the path used in this calculation is the shortest path between device and fog node identified using Dijkstra's shortest path algorithm by considering fog network topology as a weighted directed graph with network nodes being graph vertices and links and delays representing graph edges and their weights. As congestion delay at node is variable and is dependent on workload at a given instant, we have considered only propagation delay towards assignment of weights to the edges. Thus, Centralized orchestrator is near cost-optimal.

HAFA orchestrator follows similar approach as Centralized orchestrator, however, for a smaller set of prospective nodes which are selected based on logical PuddleTree organization.

Note that, in rest of the document, the terms orchestrator and placement approach mean the same and are hence used in an interchangeable manner.

The Router module identifies the network path with minimum latency between two given entities. Router takes source and destination nodes and the network topology. It reads the

network topology as a weighted directed graph with nodes being vertices and links and delays representing edges and their weights. The router then runs Dijkstra's algorithm on the graph from the given source node. Afterwards, the router finds the destination node and builds a linked list of the nodes that form the shortest path from the source to destination. Router then passes this linked list to the network model to calculate total latency.

The simulator assumes a central router. This assumption is to abstract routing as a baseline for the ideal routing for the network. We make this assumption on the basis that the router does not need to maintain the full status of the entire environment. Rather, the router only requires the static map of the network and the links that connect it. This is purely for the simulator and any interaction between a node and the router is to simulate the activity of a realistic one. This router does not assume every node has the image of the full network. It cannot be emphasized enough how this simulator is made for a decentralized system in which it is unrealistic to know the state of the network at every time. The router does not violate this since it only ever gives information the nodes would naturally have in any given circumstance.

7.11.11. Data set

Tests were performed for a given number of devices in system. The locations of devices is generated in a uniformly random fashion, and are assumed to be co-located with one of the fog/network nodes in system. This is a reasonable assumption as we assume that devices connect to first hop network node over wireless network. The additional latency introduced by the last mile connectivity is similar for all test scenarios and has no impact on selection of fog node by service placement approaches. Hence, we have ignored the same in our test environment.

Tasks were generated by each device as per specifications provided in the application profile. Task sizes, and the data input/output sizes are generated by Poisson distributions using the mean values defined in application profile.

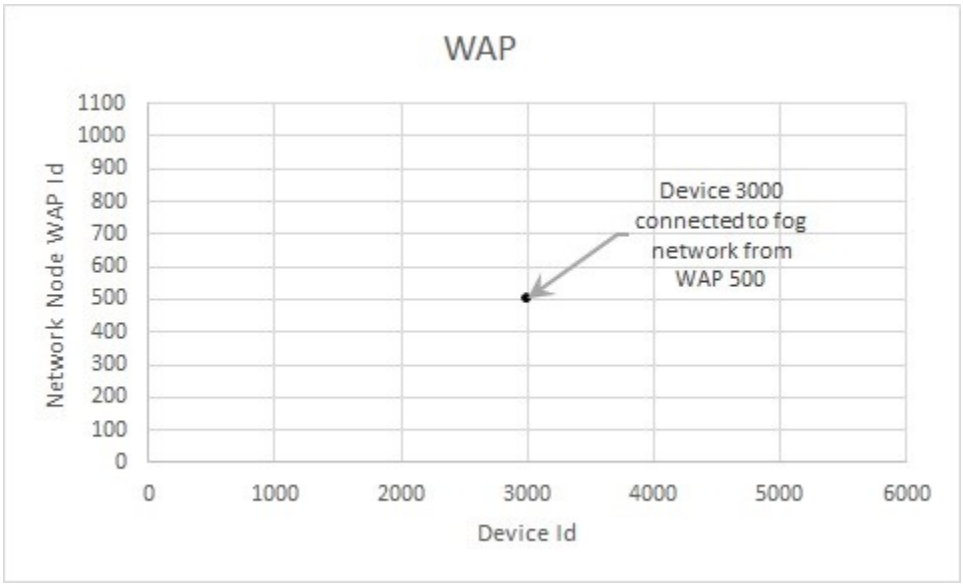

Figure 30: Test execution: Device connected Network Node WAP Id - Representation.

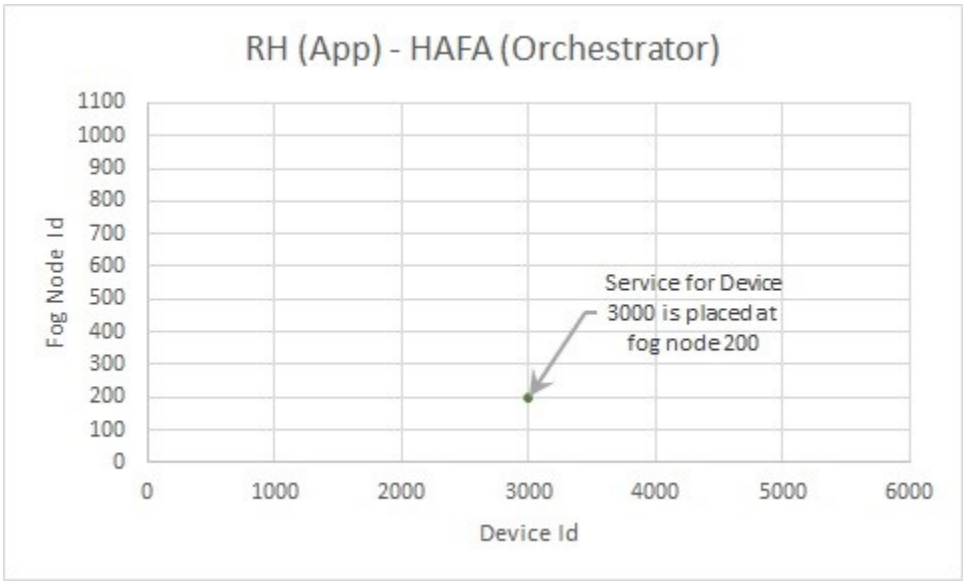

Figure 31: Test execution: Fog Node hosting service for Device - Representation.

7.11.12. Test execution

Performance tests were executed to demonstrate the behavior and effectiveness of our proposed approach in a large environment as well as provide a quantitative comparison of its performance with other centralized and distributed approaches.

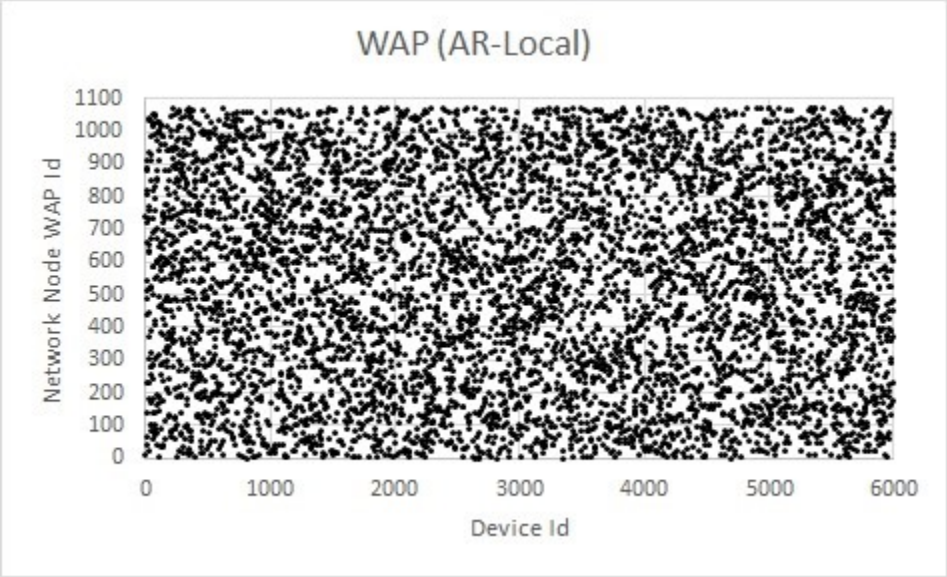

Figure 32: Test execution: Device connected Network Node WAP Id for test scenario with Augmented Reality Application and Local Orchestrator.

The test system configuration was varied to show the behavior of various service placement approaches for applications services with different profiles. Depending on the availability of resources as well as number and type of service requests, congestion may occur in the network and there is no assurance of availability of free resources to satisfy all the requests generated in the system.

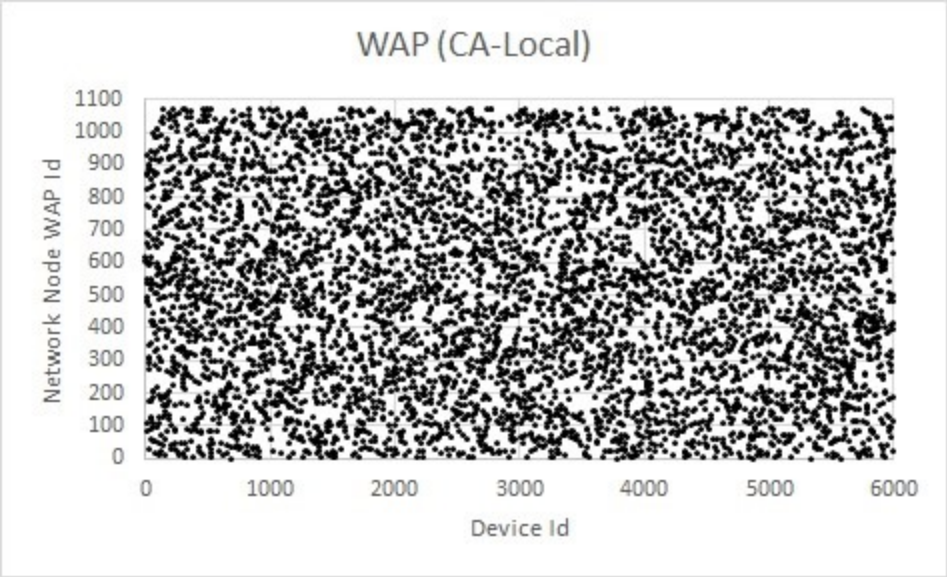

Figure 33: Test execution: Device connected Network Node WAP Id for test scenario with Cognitive Assistance Application and Local Orchestrator.

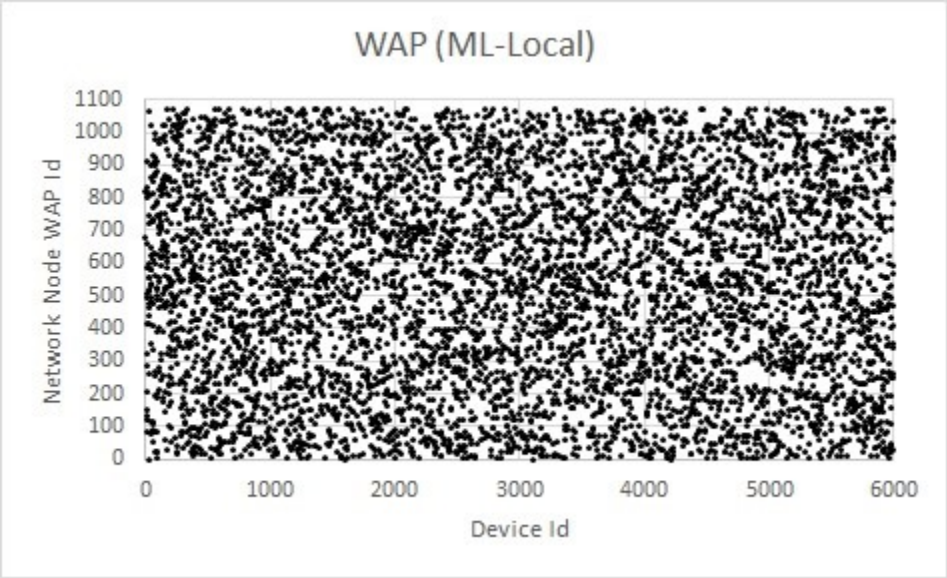

Figure 34: Test execution: Device connected Network Node WAP Id for test scenario with Machine Learning Application and Local Orchestrator.

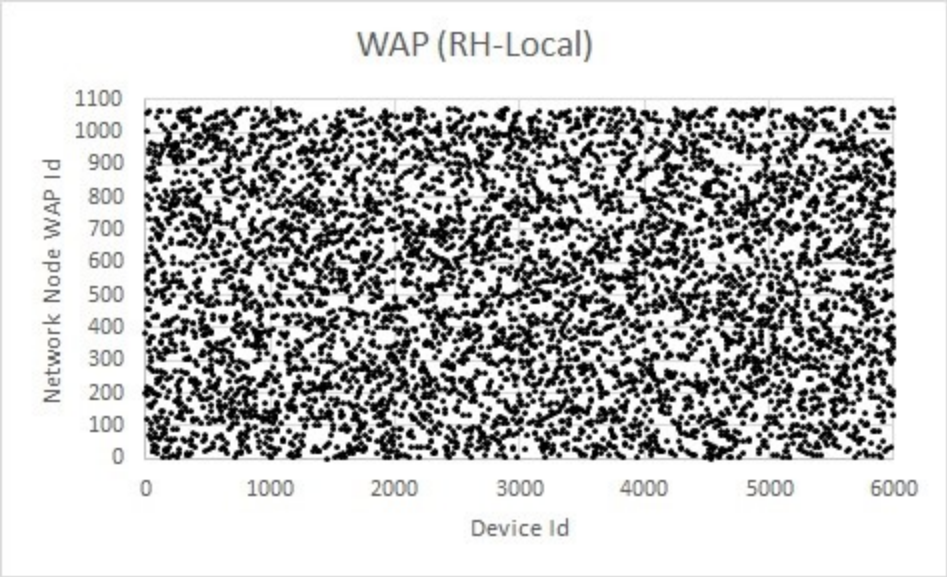

Figure 35: Test execution: Device connected Network Node WAP Id for test scenario with Remote Healthcare Application and Local Orchestrator.

Tests were executed by varying the number of devices from 1000 to 6000 in increments of 1000, for five service placement approaches described earlier, along with the HAFA orchestrator. As each device is associated with a different user, service requests are submitted by fog broker, one for each device, to deploy application services. Upon identification of hosts by the service placement approach, services are deployed on hosts.

The simulator environment facilitates generation of tasks by devices, execution by fog nodes, and results are sent to users over the network. Test runs are performed for a duration of 30 minutes simulation time. Each data point shown in the graphs in this chapter represents the observation from a separate test run for a simulation time of 30 minutes.

Tests were performed for four applications with varied profiles, namely Augmented Reality, Cognitive Assistance, Machine Learning, and Remote Healthcare. For comparison purposes, the task generation rate for all application profiles is set to the same value.

Shown in Figure 32, Figure 33, Figure 34, and Figure 35 are the service request origin locations for four of the tests executed, to demonstrate that the requests are uniformly generated across the simulated geography. A total of 144 tests were executed, the results of which are discussed in this dissertation. We assume that the requests were generated in a similarly uniform manner for rest of the 140 tests.

The fog/network node Ids assigned are as follows. Note that the network node WAP Id is the same as the fog node Id at a given location.

- Layer-1 – 395-1073 (School)
- Layer-2 – 142-394 (Connect center)
- Layer-3 – 62-141 (Library)
- Layer-4 – 12-61 (Ward office)
- Layer-5 – 2-11 (University)
- Layer-6 – 1 (City center)
- Layer-7 – 0 (Cloud)

7.11.13. Application: Augmented Reality

Scenario. Imagine a smart museum which aims to enhance the experience of its visitors using Augmented Reality technology. The visitors are provided with AR/VR set or a Google Glass which senses interest of the wearer when he stares at an artifact for a brief period, and instantly provides relevant details on the device display. This provides improved visitor experience, who may derive good value for the money and time spent at the museum. This may even reduce the need for personal guides to the visitors, whose job is to provide detailed information relevant to artifacts which is not displayed alongside, likely due to lack of space.

Application Profile. Tasks were generated at a Poisson mean inter-arrival rate of one task in 5 seconds, i.e. if the user is looking at an artifact in museum for 5 seconds, a task is generated by Google Glass device with a picture of artifact. The application service will process the request and returns details about the artifact which are displayed by Google Glass for enhanced user experience.

This application is assumed to be latency-tolerant and has medium computation and communication requirements. Response time limit for each task is set to 5 seconds, beyond which the user's interest is expected to have moved on to the next artifact down the aisle. Each task requires execution of 2000 million instructions, on average, and uses one CPU core. Input to the task is a high-definition image of the artifact and has an average size of 1500 kilobytes. Output for the task is the additional information regarding artifact provided in the form of text displayed on the screen and has an average size of 25 kilobytes.

Results analysis. Tests were performed with six service placement approaches using Augmented Reality application workload for various device counts. Observations from the tests are as follows.

Edge orchestrator.

Edge nodes are those belonging to fog layer-1. These are nodes with small resource configurations and are deployed in large quantity so that they are available in vicinity to device/user. As shown in Figure 37, Edge orchestrator resulted in high task failure rate for AR application as the collective resources of all the layer-1 fog nodes are sufficient only to support requests from approximately 10% of 6000 i.e. 600 AR users. This is reflected in Figure 38 which

shows the number of successful tasks executed by Edge placement approach for the test scenario with 6000 devices.

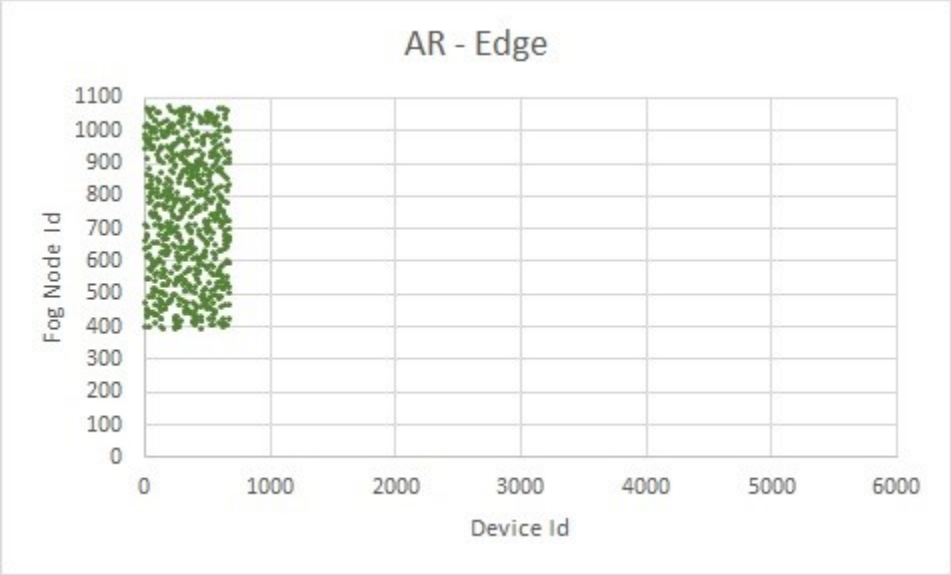

Figure 36: Augmented Reality Application – Fog nodes selected by Edge Orchestrator for service placement.

Figure 40 shows that this approach resulted in higher average cost per task as compared to all other approaches due to high compute cost on layer-1 fog nodes. The selected edge nodes are observed to be located in vicinity and hence the communication costs were lesser as compared to the computation costs.

Figure 41 depicts the average great circle distance between device/user and executing fog node measured using Haversine formula. As the number of layer-1 fog nodes are high and geographically well distributed, the average distance is observed to be lower than City center and Cloud placement approaches. At lower device counts, sufficient free resources are available in the system to facilitate selection of a remote layer-1 fog node, whereas at higher device counts,

requests are served primarily by local node, which is accessible at zero distance. It resulted in slightly lower average distance at higher device counts.

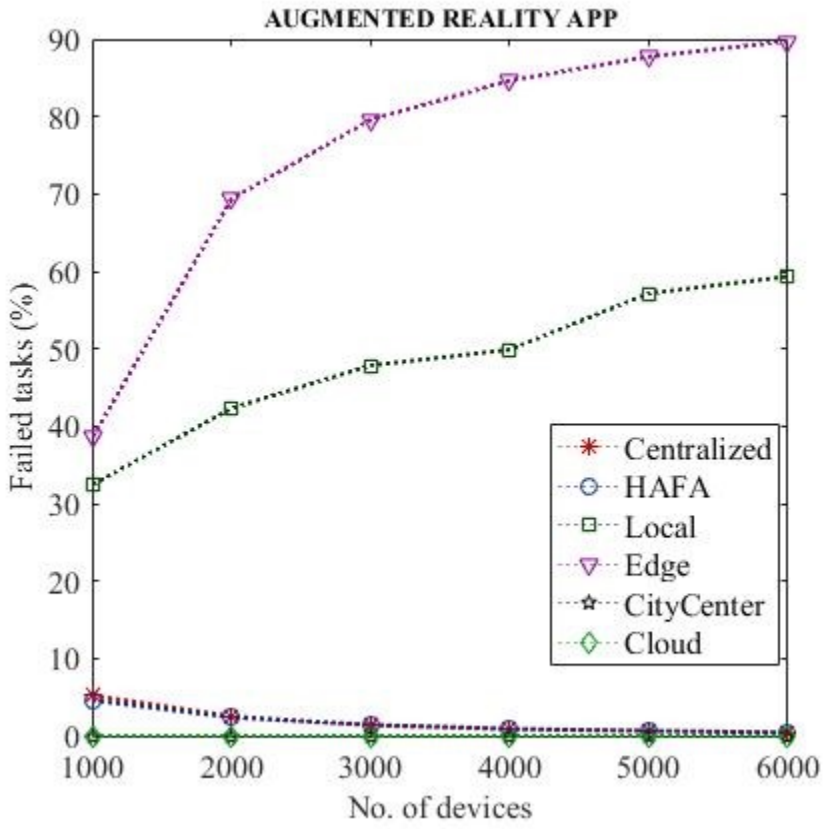

Figure 37: Augmented Reality Application – Failure rate of tasks vs. number of active devices.

Figure 43 depicts the average network distance between device and fog node. Note that the local node is accessible at one hop while the least number of hops to access any remote layer-1 fog node is three hops. At lower device counts, sufficient free resources are available to facilitate selection of a remote layer-1 fog node, whereas at higher device counts, requests are served primarily by local node, which is accessible at one hop. It resulted in lower average number of hops at higher device counts. There is minimal change observed with various device

counts, due to high percentage (40% - 90%) of task failures. In our test environment, the layer-1 nodes are accessible at fewer network hops as compared to City center and Cloud nodes.

As seen in Figure 45, average network delay is small as remote fog nodes selected are located nearby. Localizing the workload resulted in lesser network traffic which had the side effect of lower congestion delay. This combined with low propagation delay due to the selected nodes being in close vicinity, overall network delay is observed to be small. Additionally, no impact is observed at higher device counts, due to higher task failure rate resulting from lack of sufficient compute resources in layer-1 fog nodes. For instance, assuming that each device generates one task request, total number of successful tasks for 1000 and 6000 active devices is the same at 600 tasks. Edge nodes have small resource configurations. Hence, processing time is high, as compared to other approaches.

As all layer-1 fog nodes are assumed to be of similar resource configurations, average execution time of tasks is same. Due to higher task failure at larger device counts from lack of sufficient compute resources, there is no change in the number of tasks executed at higher device counts which resulted in no change with queuing delay. Thus, average processing time per task is observed to be similar for all device counts as shown in Figure 46.

As shown in Figure 48, average service time perceived per task is dominated primarily by high processing times. It remains invariant to number of devices as it is the sum of processing times and network delays, which as discussed above do not vary with device counts.

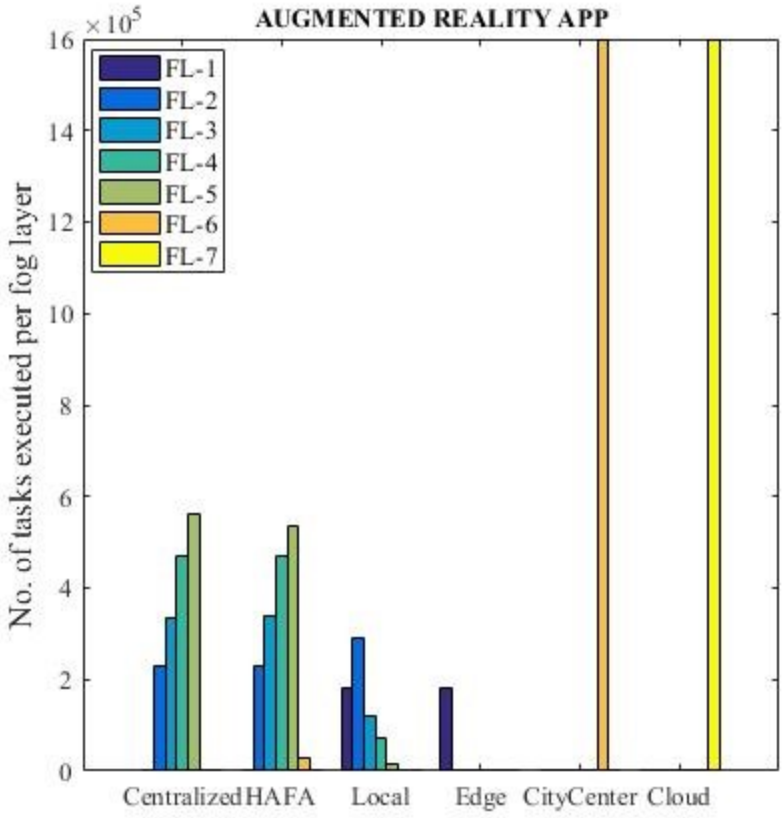

Figure 38: Augmented Reality Application – Number of tasks successfully executed per fog layer with 6000 active devices.

Figure 51 shows that only layer-1 fog nodes are utilized for task execution. Figure 49 shows that there is no impact on utilization of compute resources in entire fog environment as device counts increase, the reason being increase in task failure rate. The average host utilization percentage is lower as compared to other placement approaches as fog nodes belonging to layers 2-7 are not leveraged by the edge placement approach.

Figure 53 shows low utilization of network resources as edge placement approach selected fog nodes in vicinity which are accessible over low capacity network nodes. Additionally, high percentage of failed tasks resulted in least utilization as compared to other

placement approaches as well as no change towards utilization of fog networks at higher device counts, as can be observed from Figure 52.

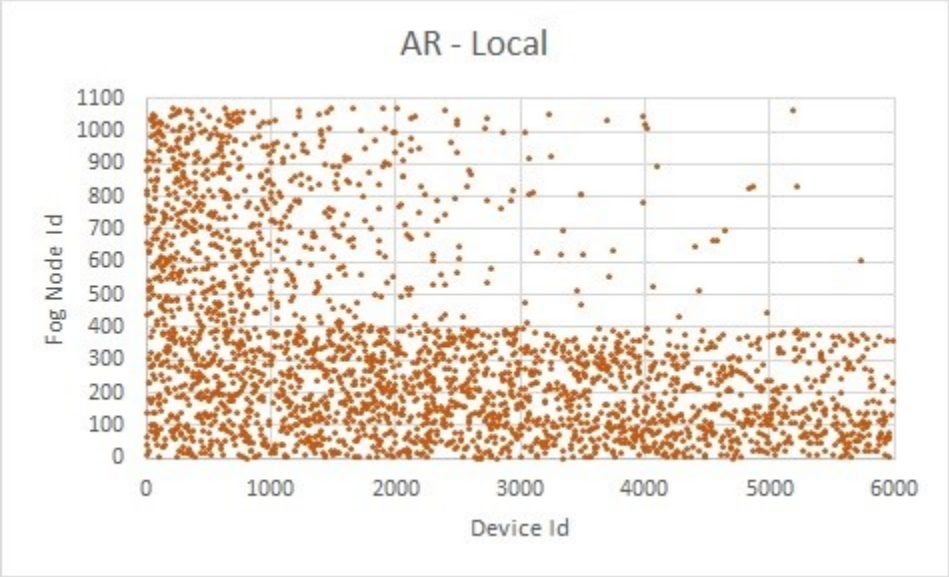

Figure 39: Augmented Reality Application – Fog nodes selected by Local Orchestrator for service placement.

As edge approach considers all available layer-1 nodes for placement, the number of prospective nodes is 679 for all device counts as shown in Figure 54 and the average number of messages exchanged, comprising both request and response messages, towards making the placement decision is 1358 as shown in Figure 55. Note the representation of Y-axis in log scale for Figure 54 and Figure 55.

Local orchestrator.

Local node is the fog node accessible over one hop from device/user. The nodes may belong to any fog layer and thus have different resource configurations and execution cost characteristics. This is a static approach as tasks generated by devices are submitted only to the

local node. If the node's resources are exhausted or insufficient to execute the task, it is marked as failed.

As shown in Figure 37, Local orchestrator resulted in lower task failure percentage for AR application as compared to Edge placement approach as local nodes may belong to any fog layer and not just the resource-constrained layer-1. Higher layer fog nodes in our test environment have larger resource configurations and hence higher percentage of tasks were executed successfully. Task failure rate is higher than other placement approaches as the lower layer fog nodes may not have sufficient resources to execute all the tasks submitted to it. This also results in increased task failure percentage at higher device counts as shown in Figure 37. Figure 38 shows that the local placement approach executes tasks on fog nodes belonging to all layers for the test scenario with 6000 devices.

Figure 40 shows that the Local placement approach resulted in lower average cost per task as compared to Edge placement approach as execution cost is highest for layer-1 fog nodes. As all nodes are local to devices, the communication cost is nonexistent. The cost is higher than that with rest of the placement approaches as the tasks are executed at layer-1 node, if that is accessible at one hop to the device, despite having a lower cost option available elsewhere in the system. It is also observed that the average cost per task has reduced at higher device counts. This can be explained by the reasoning that, as device counts increase, the task failure rate increases for those submitted to layer-1 fog nodes and most of the tasks which were successfully executed were those by higher layer fog nodes. In our test environment, execution cost on higher layer fog nodes is lesser as compared to that on lower layer fog nodes, hence total cost reduced at higher device counts as network cost is non-existent with local placement approach.

Figure 41 depicts the average great circle distance between device/user and executing fog node. As the stationary devices are assumed to be co-located with the local fog node in our test environment and are connected over wireless network, the average distance is always zero for all device counts, with Local orchestrator. This metric will have different values for the test scenario with mobile devices moving at specific velocity.

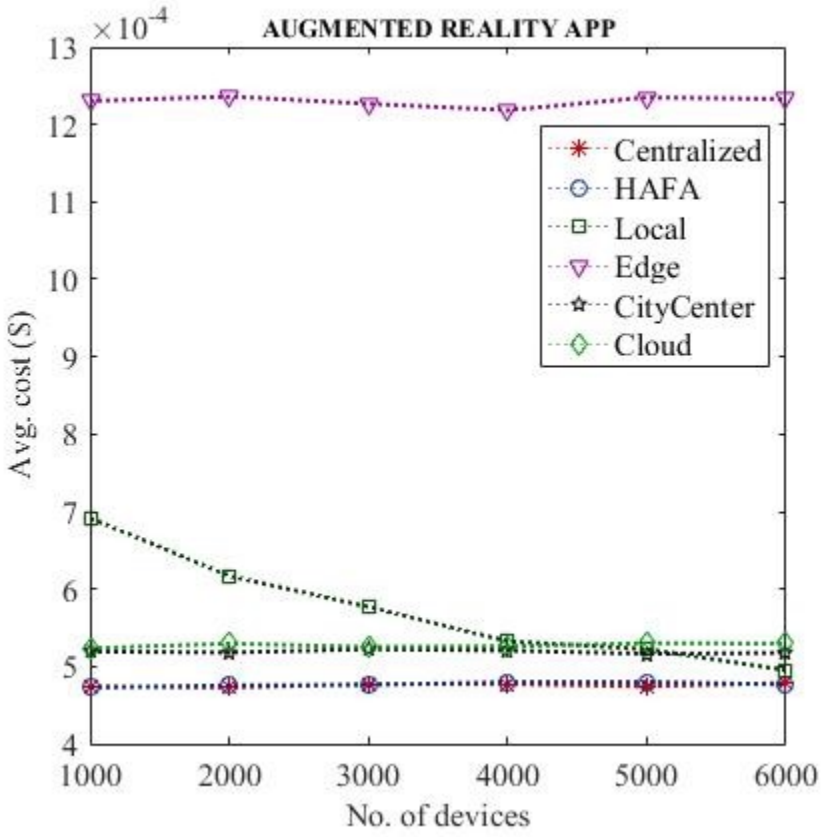

Figure 40: Augmented Reality Application – Average execution cost vs. number of active devices.

Figure 43 depicts the average network distance between device/user and executing fog node. Note that the local node is accessible at one hop from device, hence the average number of network hops is always one for all device counts, with Local orchestrator.

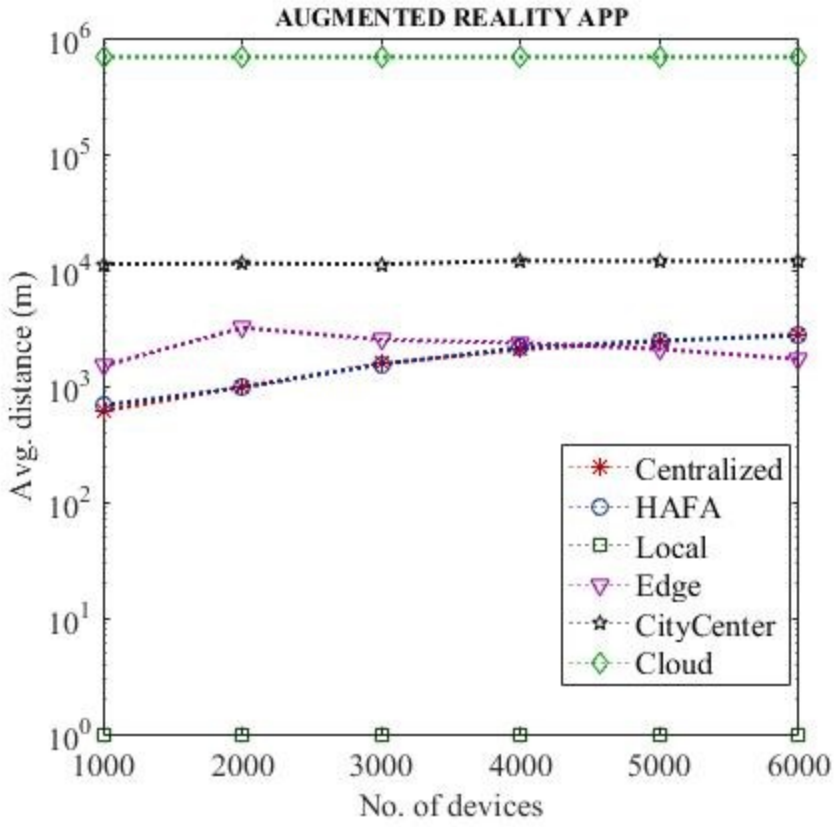

Figure 41: Augmented Reality Application – Average distance between device (user) and executing (fog) node vs. number of active devices.

As seen in Figure 45, average network delay is the least as it comprises only the congestion delay at first hop network node. Congestion delay is seen to be increasing as the data traffic at first hop network node is only from directly accessible devices. However, the rate of increase is small due to high task failure rate at higher device counts from lack of sufficient compute resources at local node. Network propagation delay is always zero due to co-location of device and local fog node.

Figure 46 shows that the average processing time is dominated by the queuing delay at fog nodes due to higher percentage of successful tasks as compared to those with Edge

placement approach, and hence is higher. Additionally, as majority of fog nodes in system belong to layer-1 or layer-2, average processing time is highest as compared to other placement approaches. As the local fog node may belong to any fog layer, tasks executed at fog nodes belonging to different layers result in different processing times. At higher device counts, statistically, more devices were located close to fog nodes belonging to higher layers, which results in lower processing times as well as most of the successful tasks were those executed at higher layer fog nodes which resulted in lower average processing time.

As shown in Figure 48, average service time perceived per task is dominated primarily by high processing times. It remains invariant to number of devices as both the components, processing time and network delay, have remained the same.

Figure 51 shows that fog nodes belonging to all layers were utilized for task execution. Figure 49 shows that the average host utilization percentage has reduced as the requests are served more by higher layer fog nodes which have larger resource configurations, resulting in lower host utilization percentage.

Figure 53 shows low utilization of network resources with Local placement approach as only one network node is utilized by tasks for data transfer, most of which were low capacity network nodes. Additionally, high percentage of failed tasks resulted in marginal increase in utilization of fog networks at higher device counts, as can be observed from Figure 52.

As local approach considers only one i.e. local fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 54 and the average number of messages exchanged, comprising both request and response messages, towards making the

placement decision is 2 as shown in Figure 55. Note the representation of Y-axis in log scale for Figure 54 and Figure 55.

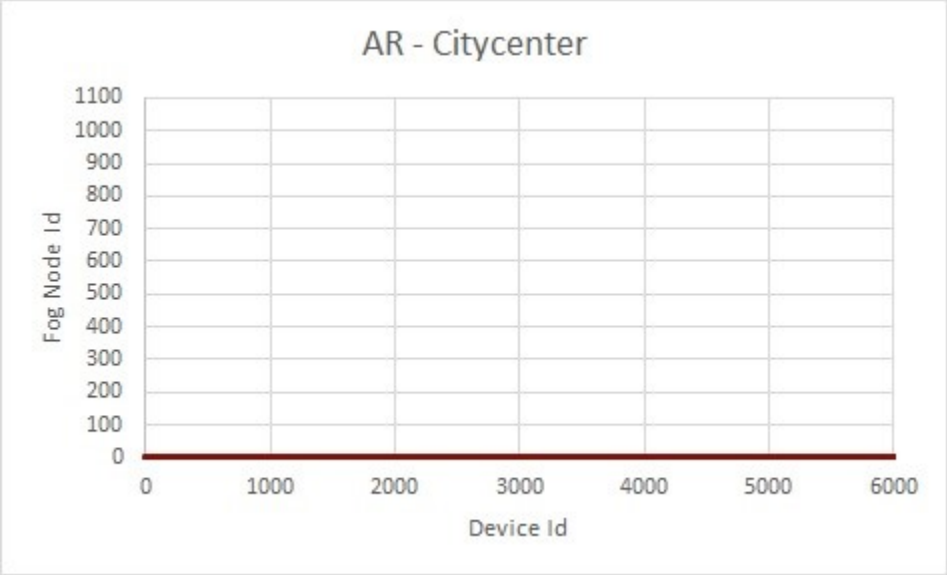

Figure 42: Augmented Reality Application – Fog nodes selected by City center Orchestrator for service placement.

City center orchestrator.

City center is the lone fog node belonging to fog layer-6 and has largest resource configuration available in the entire city. Depending on the location and network connectivity, it may be accessible over one or more hops from any device/user. Execution cost on this fog node is second lowest in our test environment. This is a static approach as tasks generated by devices are submitted only to the City center fog node. If its resources are exhausted or insufficient to execute the task, it is marked as failed.

As shown in Figure 37, City center orchestrator resulted in zero task failure percentage as City center fog node has sufficient resources to support the AR application workload from given device counts.

Figure 38 shows that the City center placement approach executed tasks on fog node belonging to layer-6 for the test scenario with 6000 devices.

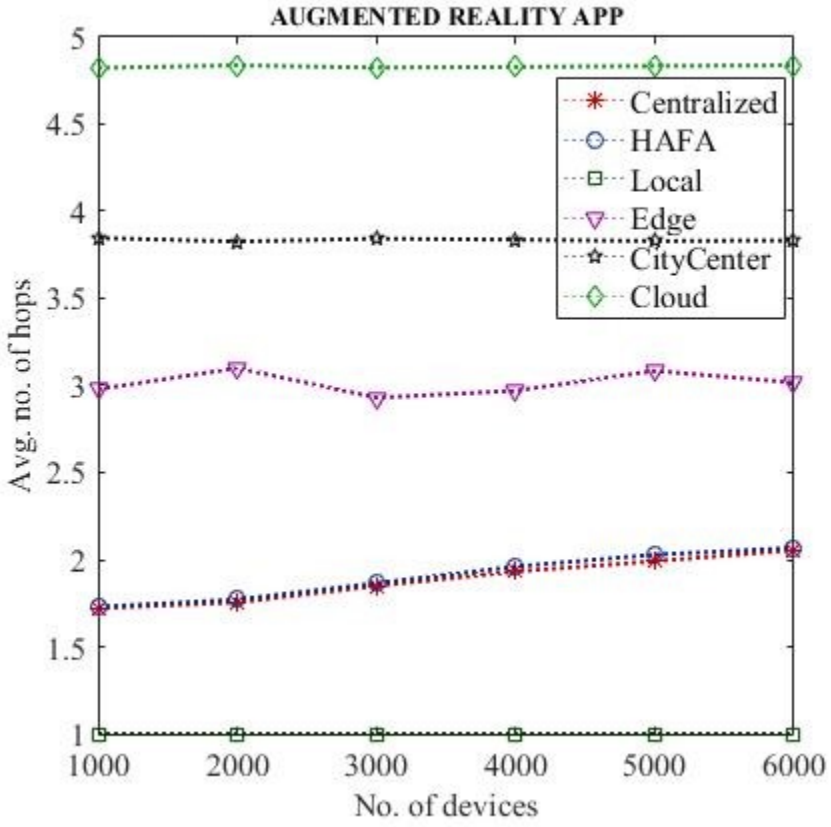

Figure 43: Augmented Reality Application – Average hops between device (user) and executing (fog) node vs. number of active devices.

Figure 40 shows that the City center orchestrator resulted in lower average cost per task as compared to Edge and Local placement approaches. This can be attributed to the lower execution cost on City center fog node as compared to that on any other fog node in the . The slight increase in average cost at higher device counts can be explained by the reasoning that, as device counts increase, higher number of tasks are generated at layer-1 and layer-2 fog nodes

which have high communication costs due to multiple network hops in path from device to City center.

Figure 41 shows that there is no change in average great circle distance between device/user and City center fog node, at higher device counts, as the devices are geographically well distributed. It is observed to be approximately 10 kilometers.

Figure 43 shows that there is no change in average network distance between device and fog node, at higher device counts. It is observed to be approximately 3.8. Note that the average number of hops is not a whole number as, in our test environment, devices may be located at different places and access the City center fog node over varied number of network hops depending on their location.

As seen in Figure 45, average network delay has increased linearly at higher device counts though the number of network hops is nearly constant. The reason for this behavior is the steep increase in the amount of network traffic as all devices submit tasks to the lone city center fog node, which increases the congestion delay at each network hop.

Figure 46 shows that the average processing time is minimal due to the high MIPS capacity of City center fog node.

As shown in Figure 48, average service time perceived per task is dominated primarily by high network delay due to low processing time.

Figure 51 shows that only fog node belonging to layer-6 is utilized for task execution. Figure 49 shows that the average host utilization percentage has steadily increased as all the tasks are served by the lone City center fog node.

Figure 53 shows high utilization of network resources with City center placement approach resulting from high data traffic. Due to centralization of task execution, as the number of devices increase, the data traffic in the system also increases resulting in an increase in average network utilization, as can be observed from Figure 52.

As City center placement approach considers only one i.e. City center fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 54 and the average number of messages exchanged is 2 as shown in Figure 55.

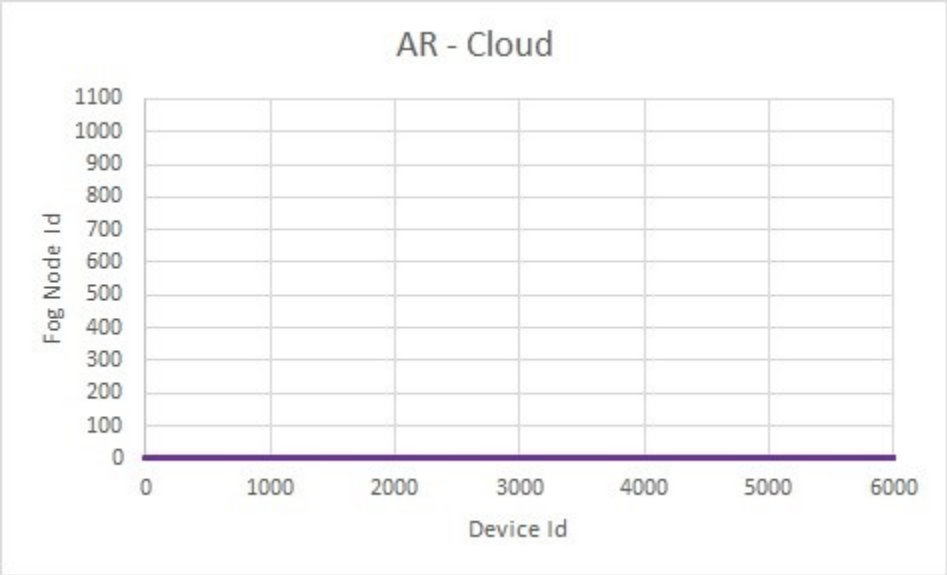

Figure 44: Augmented Reality Application – Fog nodes selected by Cloud Orchestrator for service placement.

Cloud orchestrator.

Cloud, in our test environment, comprises nodes (CPU cores) with similar MIPS capacity as that in City center fog node, but are of higher core count. Hence Cloud is referred as fog node belonging to fog layer-7 and has largest resource configuration available in the system. Depending on the location and network connectivity, it may be accessible over one or more

network hops from any device. Execution cost on this node is the least of all nodes. This is a static approach as tasks generated by devices are submitted only to the Cloud node. Cloud is assumed to have logically unlimited resources that it can satisfy any amount of workload. However, for evaluation purposes, we have configured Cloud node with a very high number of CPU cores and MIPS capacity.

As shown in Figure 37, Cloud orchestrator resulted in zero task failure percentage as Cloud node has sufficient resources to support the AR workload from given device counts.

Figure 38 shows that Cloud placement approach executed tasks on fog node belonging to layer-7 for the test scenario with 6000 devices.

Figure 40 shows that Cloud placement approach resulted in lower average cost per task as compared to Edge and Local approaches. This can be attributed to the lower execution cost on Cloud node. The slight increase in average cost at higher device counts can be explained by the reasoning that, as device counts increase, higher number of tasks are generated at layer-1 and layer-2 fog nodes which have high communication costs due to multiple network hops in path from device to Cloud. Average cost on Cloud node is slightly higher due to the extra network hop as compared to the City center node.

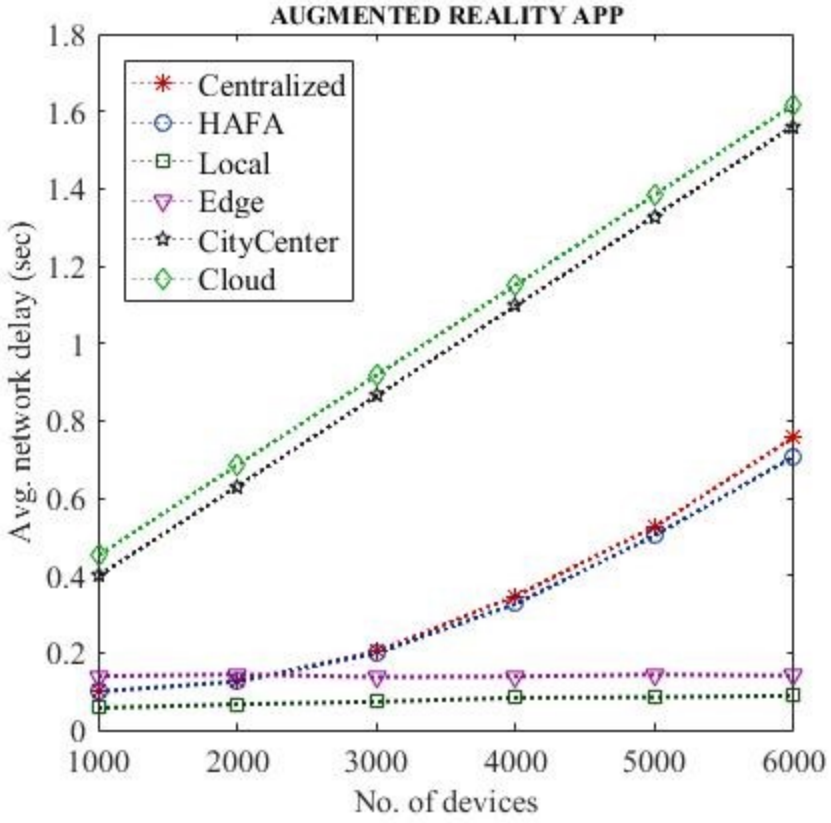

Figure 45: Augmented Reality Application – Average network delay per task vs. number of active devices.

Figure 41 shows that there is no change in average great circle distance between device/user and Cloud at higher device counts. It is observed to be approximately 700 kilometers.

Figure 43 shows that there is no change in average network distance between device and Cloud at higher device counts. It is the highest of all approaches and is observed to be approximately 4.8 as Cloud is the farthest and is accessible at one additional hop as compared to City center, for the tested network topology.

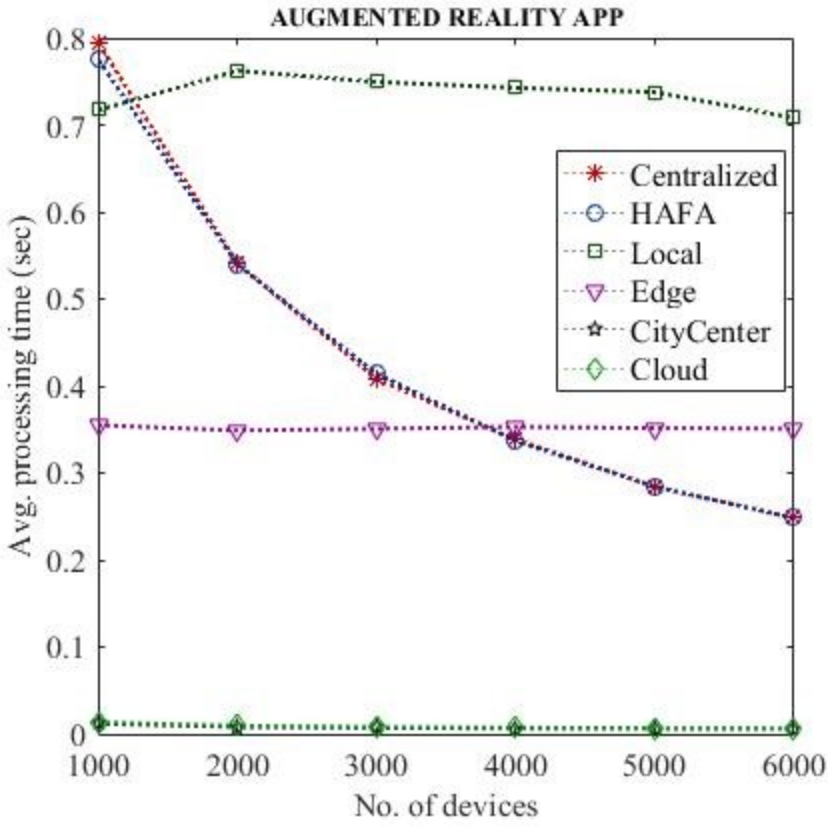

Figure 46: Augmented Reality Application – Average processing time per task vs. number of active devices.

As seen in Figure 45, average network delay has increased linearly at higher device counts due to the increase in network congestion delay. This pattern is similar to that of City center placement approach. However, the higher network delay as compared to that with City center placement approach resulted from two factors, the additional network hop on all device paths to Cloud, as well as the increase in propagation delay, as Cloud is the farthest of all nodes and is approximately 700 km away from the city.

Figure 46 shows that the average processing time is minimal due to the high MIPS capacity of city center fog node.

As shown in Figure 48, average service time perceived per AR task is highest for Cloud placement approach and is dominated primarily by high network delay due to low processing time.

Figure 51 shows that only node belonging to layer-7 is utilized for task execution. Figure 49 shows that the average host utilization percentage has steadily increased as all the tasks are served by Cloud node. However the rate of increase is lesser as compared to that with City center placement approach, owing to the larger resource configuration of Cloud node, which is set to approximately 1000X in our test environment.

Figure 53 shows highest utilization of network resources with Cloud orchestrator resulting from high data traffic across all network nodes. Additionally, the rate of increase is higher as compared to that with City center placement approach, owing to the additional network hop in our test environment as can be observed from Figure 52.

As this approach considers only Cloud node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 54 and the average number of messages exchanged is 2 as shown in Figure 55.

Centralized orchestrator.

Centralized placement approach considers fog nodes belonging to all layers towards service placement.

As shown in Figure 37, centralized orchestrator resulted in a small percentage of task failures for AR application in spite of resources available in system to satisfy the service request. It is observed from test logs that this approach hosted services on lower layer fog nodes to reduce average cost. But, as lower layer fog nodes i.e. those belonging to layer-1 and layer-2 have only

one CPU core, the high queuing delays resulted in a small percentage of failed tasks. However, as device counts increase, this approach selected higher layer nodes at comparatively higher cost as lower layer resources were exhausted, which were successfully completed owing to larger resource configurations of higher layer nodes resulting in lower queuing delays. Thus, we can observe from Figure 37 that the percentage of failed tasks has reduced at higher device counts. Figure 38 shows that the successful tasks were executed on all fog layers except layer-1 for the test scenario with 6000 devices.

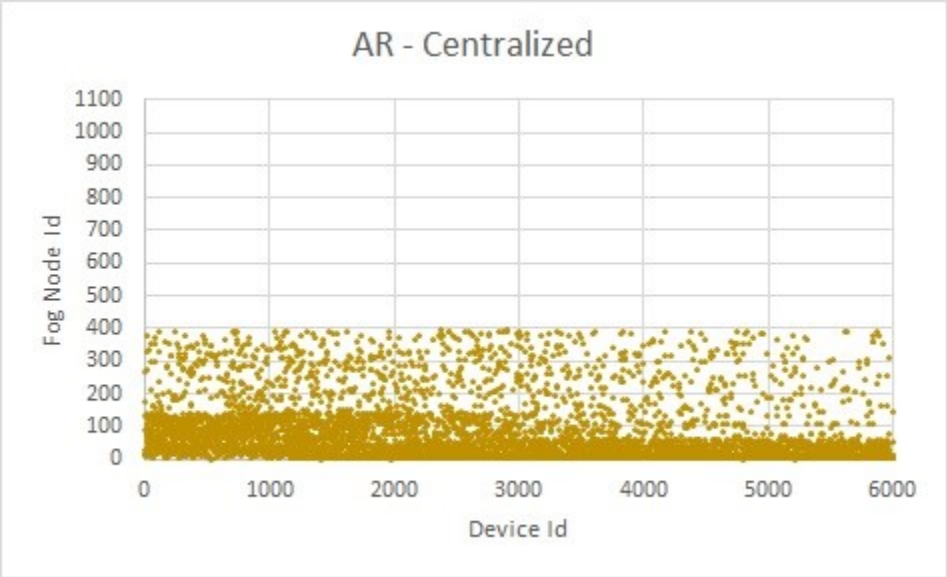

Figure 47: Augmented Reality Application – Fog nodes selected by Centralized Orchestrator for service placement.

Figure 40 shows that Centralized placement approach resulted in least average cost per task as compared to all other approaches, as it selects the node with least total cost and has sufficient resources available to host a given service from all the nodes available in system using global system state knowledge. Hence, it has the lowest average cost for all device counts.

Figure 41 shows that Centralized placement approach selected fog nodes in vicinity when free resources are available at low device counts, and selected nodes at farther distances when nearby resources are exhausted for scenarios with higher device counts. This can be explained as follows. As device counts increase, fog nodes offering overall lower cost do not have free resources and the requests are served by those at larger network costs and are likely at farther geo-distances. Hence, average distance increased as the device count is increased.

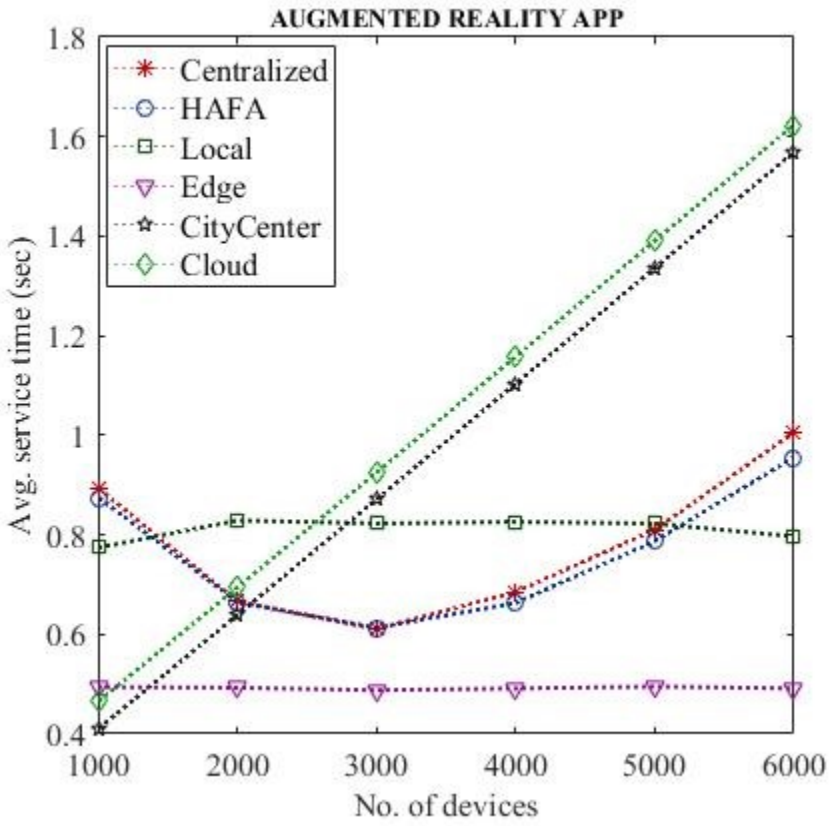

Figure 48: Augmented Reality Application – Average service time per task vs. number of active devices.

The average number of network hops is observed to be slowly increasing at higher device counts, as can be seen from Figure 43. The Centralized placement approach selected fog nodes

accessible at fewer network hops as compared to other approaches in an attempt to reduce the communication cost component.

Centralized orchestrator attempts to reduce communication cost by the reducing number of network hops which may result in lower network delay. As device counts increase, the average network delay increased due to higher network congestion, as can be seen from Figure 45. However, the network delay is smaller than that with Cloud and City center approaches. It is observed to be lower for Edge and Local placement approaches. But these values are invalid and can be ignored considering the high task failure percentage with these two approaches.

Centralized approach selected nodes accessible at fewer hops which were primarily belonging to lower layers for scenarios with low device counts, resulting in higher average processing time. At higher device counts, when resources on lower layer fog nodes were exhausted, higher layer nodes were selected which were accessible at higher number of hops and higher communication costs, resulting in lower average processing time, as can be observed in Figure 46.

As shown in Figure 48, average service time perceived per task is dominated primarily by high average processing time at lower device counts, but is dominated by increasing network delay at higher device counts. Average service time using Edge placement approach is lower than Centralized placement approach, but they are invalid and can be ignored considering its high task failure rate.

Figure 51 shows the percentage of resources utilized at each fog layer for task execution by Centralized orchestrator. It is observed that primarily fog nodes from all layers except layer-1 were utilized. These have smaller resource configurations and hence the average utilization

percentage has increased at a higher rate as shown in Figure 49 in comparison with City center and Cloud placement approaches despite leveraging similar quantities of resources for task execution.

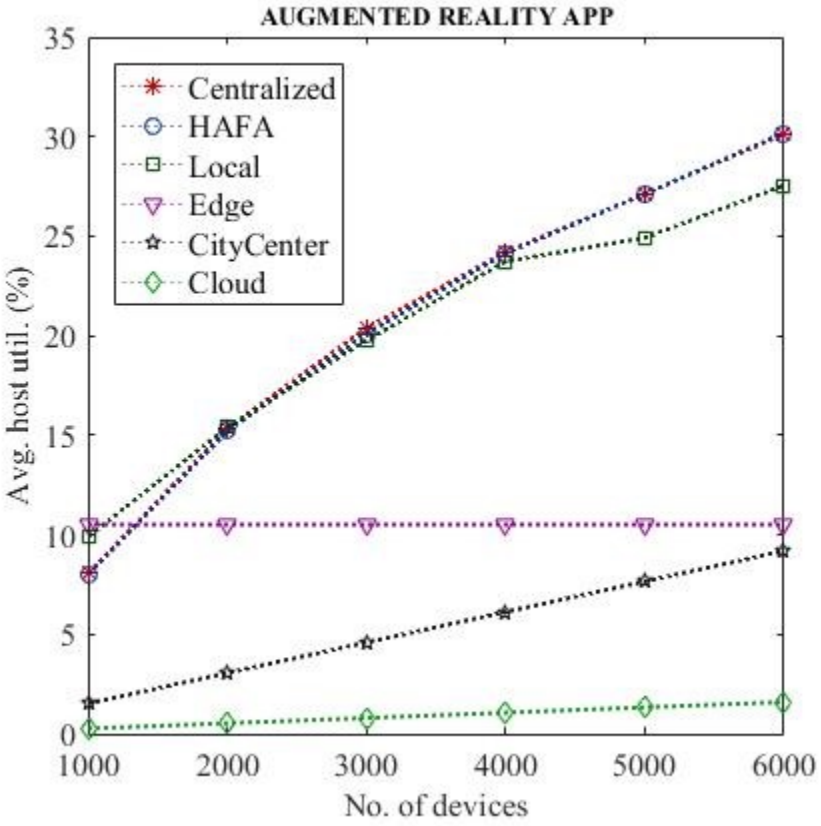

Figure 49: Augmented Reality Application – Average host utilization vs. number of active devices.

Figure 53 shows the percentage of network resources utilized at each fog layer by Centralized orchestrator. As the nodes selected for placement were accessible at lower number of network hops i.e. nodes selected were in vicinity, the average network utilization is observed to be consistently lesser than that with City center and Cloud placement approaches. The rate of

increase in average network utilization at higher device counts was also lower as can be seen from Figure 52.

As Centralized approach considers all available nodes in system for placement, the number of prospective nodes is 1074 for all device counts as shown in Figure 54 and the average number of messages exchanged towards making the placement decision is 2148 as shown in Figure 55.

HAFA orchestrator.

HAFA placement approach leverages PuddleTree and Puddle membership information to perform a search for cost-efficient fog node in an organized manner. This approach considers fog nodes belonging to each layer separately to identify the physically nearest one from each layer, following which the fog node with least cost is selected among the ones identified to be nearest from each fog layer.

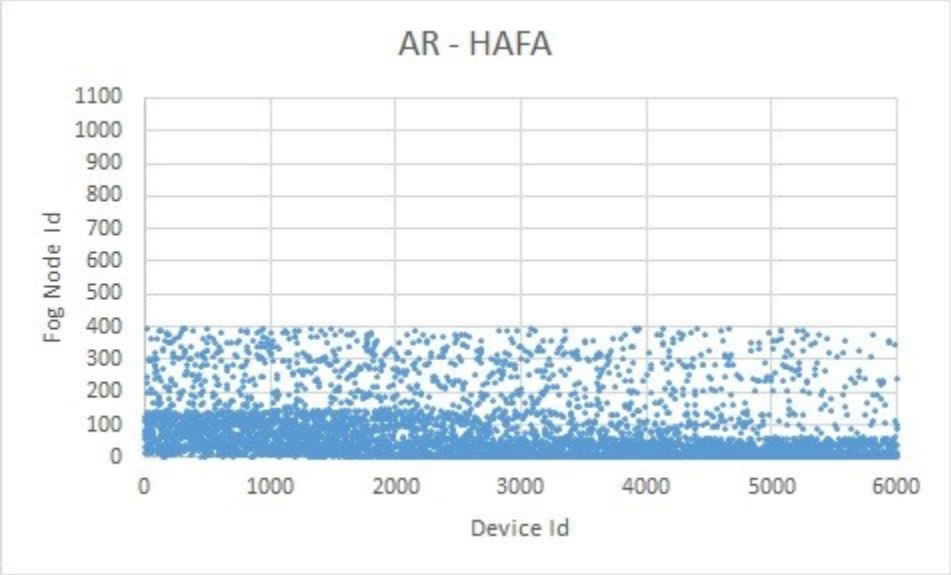

Figure 50: Augmented Reality Application – Fog nodes selected by HAFA Orchestrator for service placement.

As shown in Figure 37, HAFA placement approach resulted in a small percentage of task failures, similar to centralized approach for the same reasons as described in the corresponding subsection.

Figure 38 shows that the successful tasks were executed on all fog layers except layer-1 for the test scenario with 6000 devices.

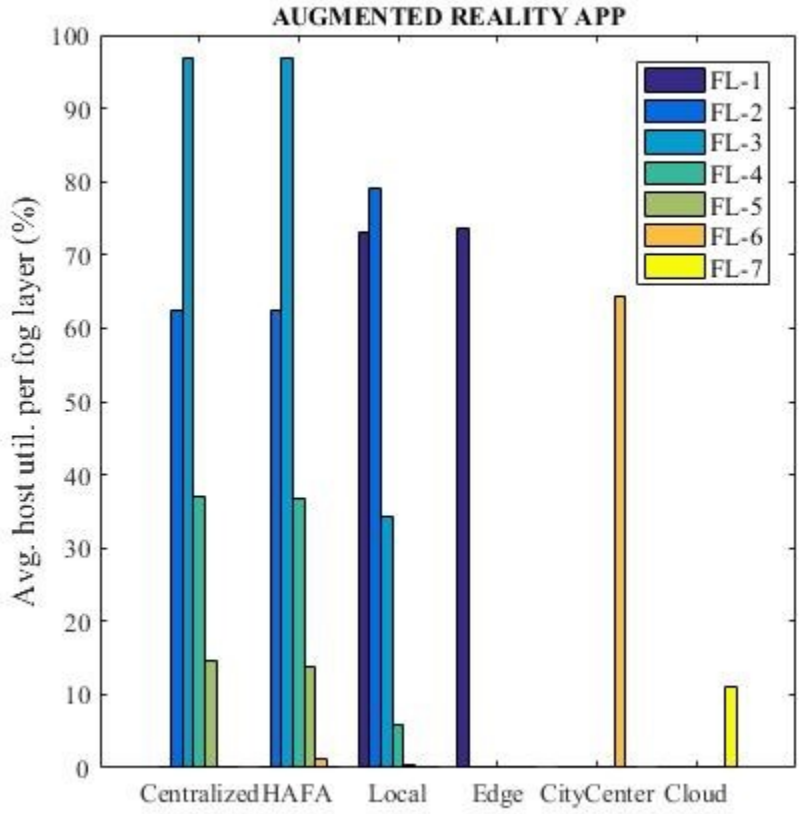

Figure 51: Augmented Reality Application – Average host utilization per fog layer with 6000 active devices.

Figure 40 shows that HAFA placement approach resulted in least average cost per task as compared to all other approaches, similar to Centralized placement approach, as it selects nodes in vicinity and extends the search only so far that it finds a node which can satisfy the given

service resource requirements. The identified fog node from each layer is likely to be accessible over a small number of hops, thus reducing communication costs. As nodes belonging to different layers have different execution costs in our test environment, the fog node with overall least cost comprising computation and communication costs is selected among the ones identified one per fog layer.

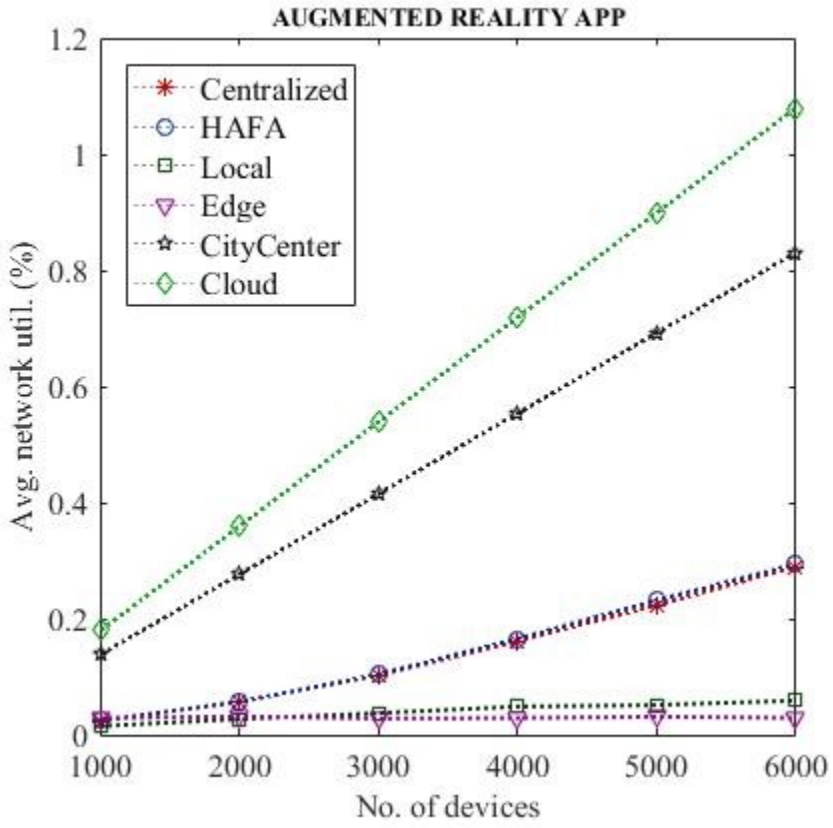

Figure 52: Augmented Reality Application – Average network utilization vs. number of active devices.

HAFA placement approach considers all nodes in the system towards placement. However, contrary to Centralized approach, the selection is made using only local system state knowledge and attempts to approximate the behavior of Centralized placement approach.

Although it has limited system state information, average cost with HAFA orchestrator is observed to be nearly the same as centralized approach. It shows the superiority of the HAFA orchestrator as its decision cost is lower than that for Centralized orchestrator.

As the number of devices scale and local resources are exhausted, HAFA orchestrator expands the search to neighbors, thus the average great circle distance between device and fog node increases with device count. However, it is observed to be similar to that with Centralized orchestrator as shown in Figure 41.

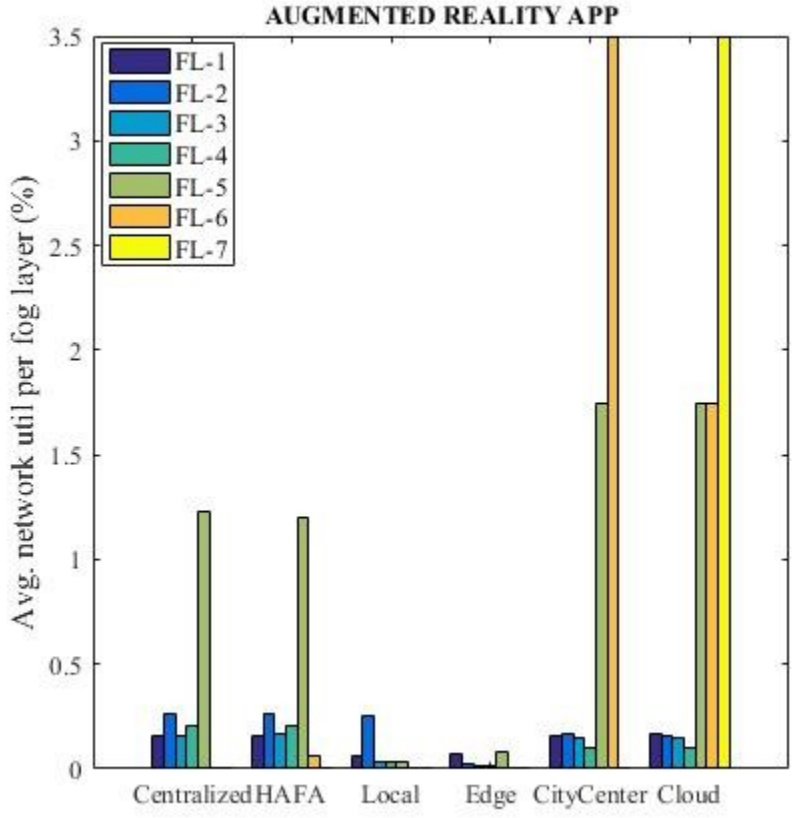

Figure 53: Augmented Reality Application – Average network utilization per fog layer with 6000 active devices.

The average number of network hops is observed to be slowly increasing at higher device counts as can be seen from Figure 43. The behavior of HAFA placement approach is approximately similar to that of Centralized placement approach for all device counts even without needing the complete system state information as required by Centralized orchestrator.

As can be observed from Figure 45, Figure 46, and Figure 48, average network delay, average processing time, and average service time with HAFA placement approach are observed to be similar to that from Centralized placement approach, as both the approaches attempt to reduce the sum of computation cost and communication cost for the tasks.

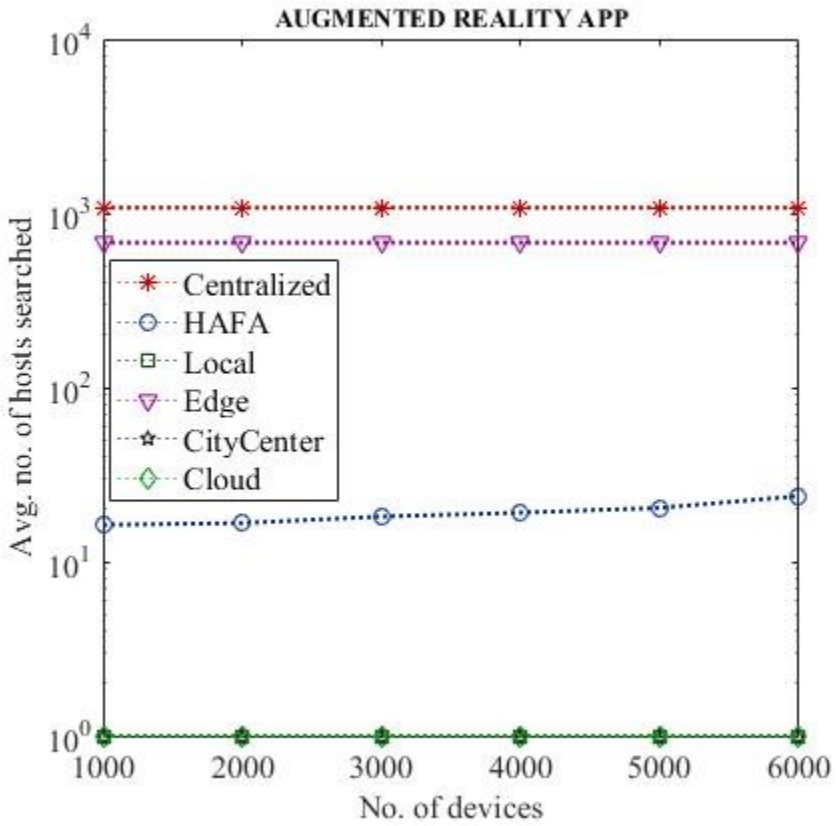

Figure 54: Augmented Reality Application – Average number of hosts searched vs. number of active devices.

Figure 51 and Figure 53 shows that HAFA placement approach utilized fog nodes from all layers except layer-1 towards task execution, in a behavior similar to that of centralized placement approach. Thus the average host compute and network utilization percentages are similar to those with Centralized approach for all device counts as shown in Figure 49 and Figure 52 respectively.

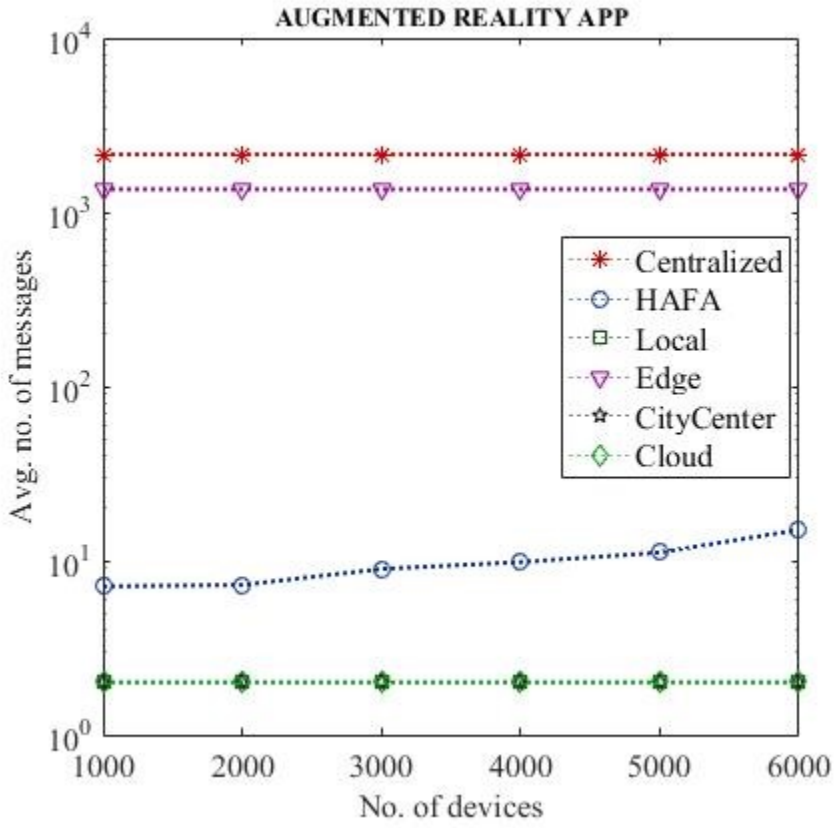

Figure 55: Augmented Reality Application – Average number of messages exchanged vs. number of active devices.

HAFA placement approach considers member nodes belonging to only local Puddle initially and expands the search space only to immediate neighbors leveraging PuddleTree parent/child logical links until the search for fog node with required free resources is successful

or the search space is exhausted. Thus, the number of prospective nodes considered towards placement of each service in fog environment may be different and are significantly lesser as compared to the number of prospective nodes considered by Centralized placement approach. The average number of prospective nodes with HAFA approach for AR application profile is observed to be 16 nodes for test scenario with 1000 devices for all device counts which steadily increased with higher device counts to 23 nodes for test scenario with 6000 devices.

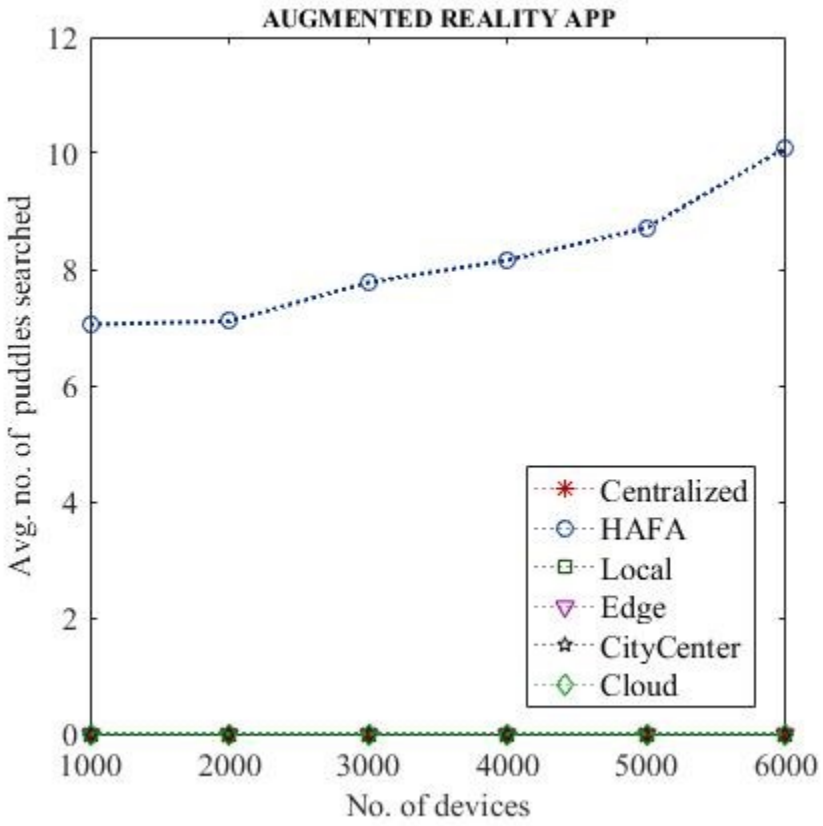

Figure 56: Augmented Reality Application – Average number of Puddles searched vs. number of active devices.

Similarly, the number of messages exchanged have ranged from 7 for 1000 devices to 15 for 6000 devices, and the number of Puddles that comprised the search space ranged from 7 for 1000 devices to 10 for 6000 devices. These patterns can be seen in Figure 54 and Figure 55.

7.11.14. Application: Cognitive Assistance

Scenario. Imagine a visually challenged person having a normal day. The Google glass device that he is wearing takes and processes pictures as he moves through the crosswalk to assist him in safe crossing. It provides a virtual view of what's going on in the street at the moment by continuously processing the current contextual information and providing audio feedback to the person in an instantaneous manner [112].

Application Profile. Tasks were generated at a Poisson mean inter-arrival rate of one task in 5 seconds to ensure that the user is provided with contextual information at a leisurely pace, but not overloaded with details.

This application is assumed to be latency-sensitive and has medium computation and communication requirements. Response time limit for each task is set to 370 milliseconds, beyond which the user's gaze may have changed and the information provided may not be relevant. It may even prove to be hazardous, considering the dependence of visually challenged person on the application for performing critical tasks.

Each task requires execution of 2000 million instructions, on average, and uses one CPU core. Input to the task is a high-definition image of the street view and has an average size of 1500 kilobytes. Output for the task is the contextual information provided in the form of audio feedback and has an average size of 25 kilobytes.

Results analysis. Tests were performed with six service placement approaches using Cognitive Assistance application workload for various device counts. Observations from the tests are as follows. Refer test results analysis section for Augmented Reality application profile for more details regarding various placement approaches.

Edge orchestrator.

As shown in Figure 58, Edge orchestrator resulted in high task failure rate for CA application as the collective resources of all the layer-1 fog nodes are sufficient only to support the requests from approximately 600 users. This is reflected in Figure 59 which shows the least number of successful tasks executed by Edge placement approach for the test scenario with 6000 devices.

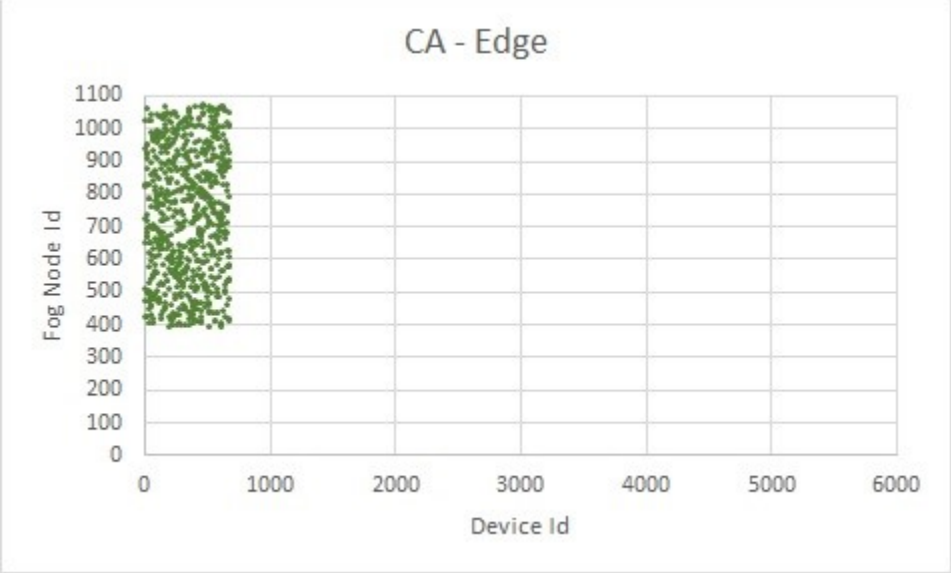

Figure 57: Cognitive Assistance Application – Fog nodes selected by Edge Orchestrator for service placement.

Figure 60 shows that Edge placement approach resulted in higher average cost per task as compared to all other approaches, the reason being high compute cost on layer-1 fog nodes.

Figure 62 depicts the average great circle distance between device/user and executing fog node. As the number of layer-1 fog nodes are high in number, the average distance is observed to be lower than City center and Cloud placement approaches. At lower device counts, sufficient free resources are available in system to facilitate selection of a remote layer-1 fog node, whereas at higher device counts, requests are served primarily by local node which is accessible at zero distance. This resulted in lower average distance at higher device counts.

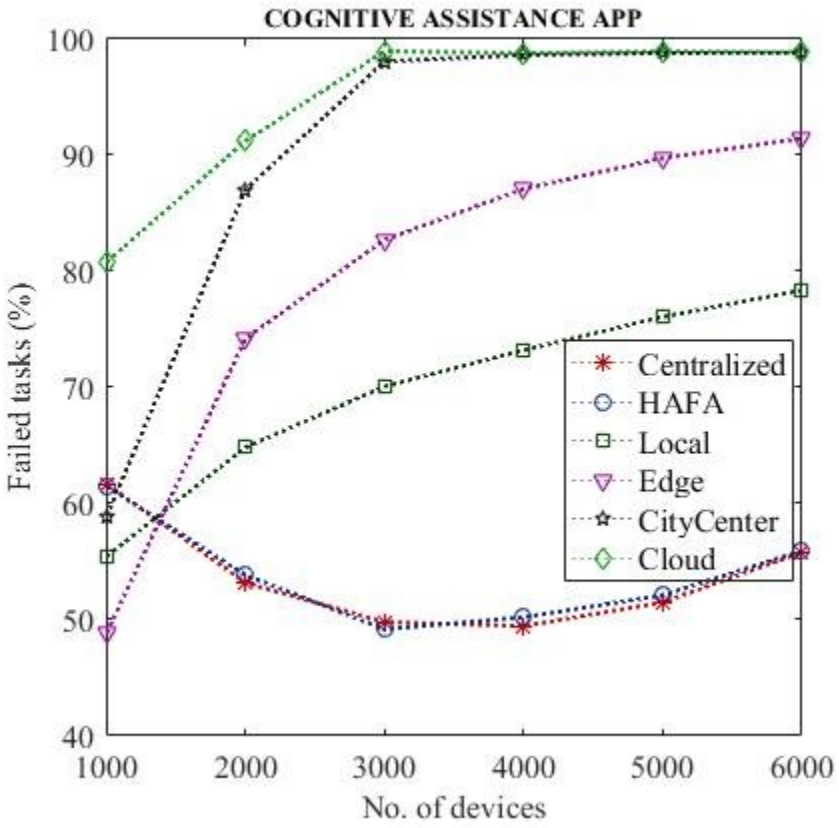

Figure 58: Cognitive Assistance Application – Failure rate of tasks vs. number of active devices.

Figure 64 depicts the average network distance between device and fog node in terms of number of network hops. No variation is observed for test scenarios with various device counts, due to high percentage of task failures.

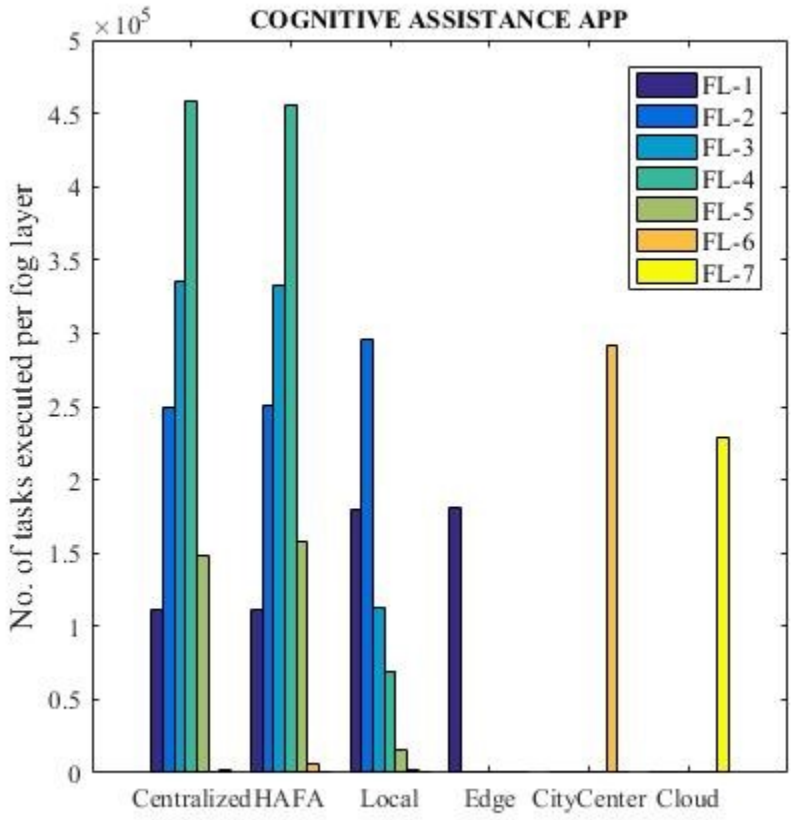

Figure 59: Cognitive Assistance Application – Number of tasks successfully executed per fog layer with 6000 active devices.

As seen in Figure 65, average network delay is small as remote fog nodes selected are located nearby to the device to satisfy the application latency constraint. Localizing the workload resulted in lesser network traffic which had the side effect of lower congestion delay. This combined with lower propagation delay due to the selected nodes being in close vicinity, overall network delay is observed to be very small. Additionally, no impact is observed at higher device

counts due to higher task failure rate resulting from lack of sufficient compute resources in layer-1 fog nodes.

Edge nodes have small resource configurations. Hence, processing time is high as compared to other approaches. As all layer-1 fog nodes are assumed to be of similar resource configurations, average execution time of tasks is same. Due to higher task failure at larger device counts from lack of sufficient compute resources, there is no change in the number of tasks executed at higher device counts which resulted in no change with queuing delay. Thus, average processing time per task is observed to be similar for all device counts as shown in Figure 66.

As shown in Figure 67, average service time perceived per task is dominated primarily by high network delay.

Figure 71 shows that only layer-1 fog nodes are utilized for task execution. Figure 69 shows that there is no impact on utilization of compute resources in entire fog environment as device counts increase, the reason being increase in task failure rate. The average host utilization percentage is lower as compared to Centralized and HAFA placement approaches as fog nodes belonging to layers 2-7 are not leveraged by Edge placement approach.

Figure 74 shows low utilization of network resources as Edge placement approach selected fog nodes in vicinity which are accessible over low capacity network nodes. Additionally, high percentage of failed tasks resulted in least utilization as compared to other placement approaches as well as no change towards utilization of fog networks at higher device counts, as can be observed from Figure 72.

As Edge placement approach considers all available layer-1 nodes for placement, the number of prospective nodes is 679 for all device counts as shown in Figure 75 and the average number of messages exchanged is 1358 as shown in Figure 76.

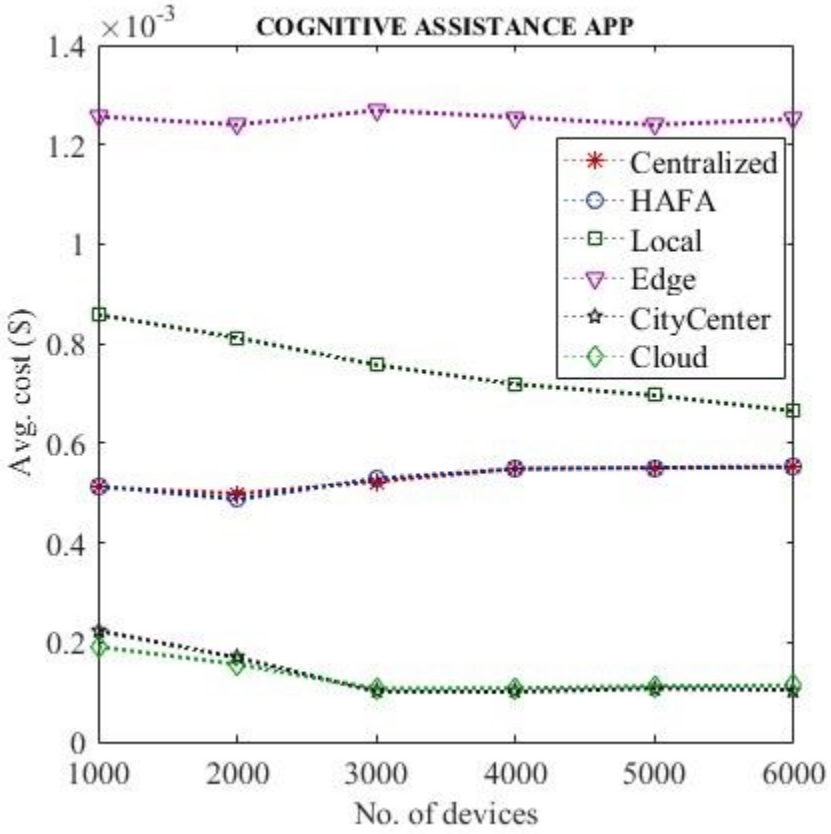

Figure 60: Cognitive Assistance Application – Average execution cost vs. number of active devices.

Local orchestrator.

As shown in Figure 58, Local orchestrator resulted in lower task failure percentage as compared to Edge placement approach as local nodes may belong to any fog layer and not just the resource-constrained layer-1. Higher layer fog nodes have larger resource configurations and hence higher percentage of tasks were executed successfully. Additionally, execution of tasks on

nodes belonging to lower fog layers have resulted in high processing time due to queuing delay, thus failed to satisfy the application latency requirements. This also resulted in increased task failure percentage at higher device counts as shown in Figure 58.

Figure 59 shows that the Local placement approach executes tasks on fog nodes belonging to all layers for the test scenario with 6000 devices.

Figure 60 shows that the Local placement approach resulted in lower average cost per task as compared to Edge placement approach as execution cost is highest for layer-1 fog nodes. The pattern and reasoning for this behavior is the same as that described for AR application profile.

Figure 62 shows that the average great circle distance between device and fog node is always zero for all device counts as device and local fog node are co-located in our test environment.

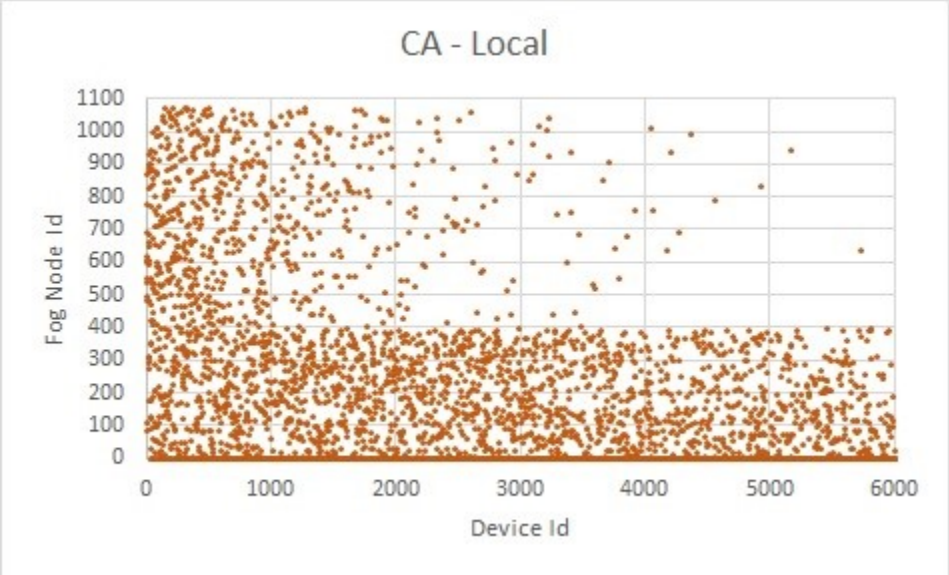

Figure 61: Cognitive Assistance Application – Fog nodes selected by Local Orchestrator for service placement.

Figure 64 shows that the average network distance between device and fog node is always one for all device counts as the local node is always accessible at one hop from device.

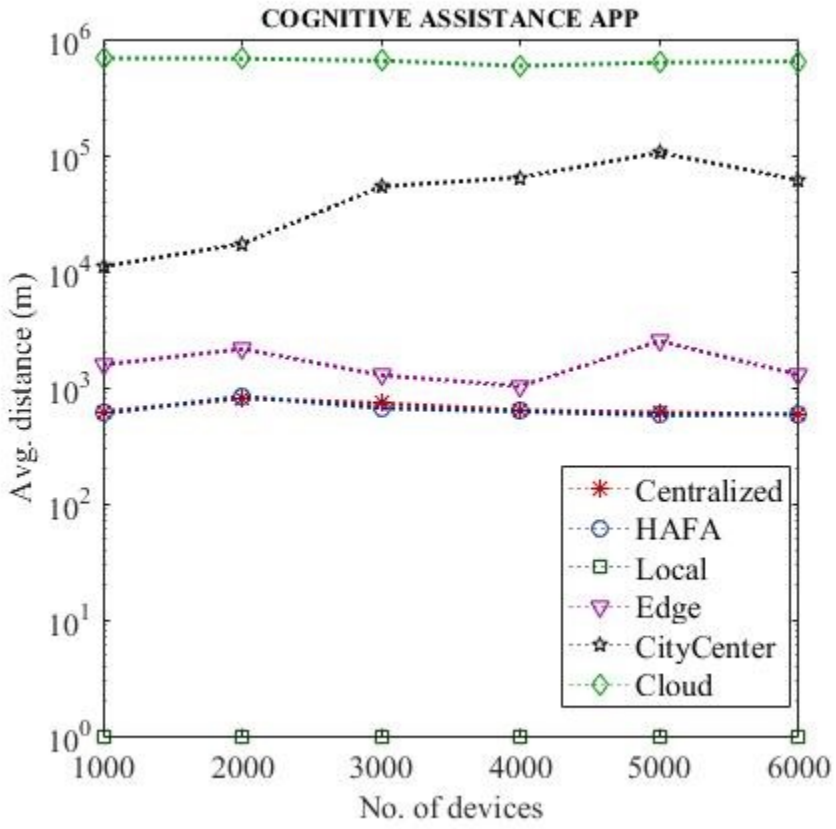

Figure 62: Cognitive Assistance Application – Average distance between device (user) and executing (fog) node vs. number of active devices.

As seen in Figure 65, average network delay is the least of all approaches as it comprises only the congestion delay at first hop network node. Congestion delay increased at higher device counts due to increase in local data traffic. However, the rate of increase is small due to high task failure rate at higher device counts.

Figure 66 shows that the average processing time is slightly higher than that from Edge placement approach at higher device counts because of the additional queuing delay introduced by higher percentage of successful requests.

As shown in Figure 67, average service time perceived per task is seen steadily increasing. However, it is less than the application latency constraint of 370 milliseconds. Processing and network latency have contributed to service time in approximately similar manner.

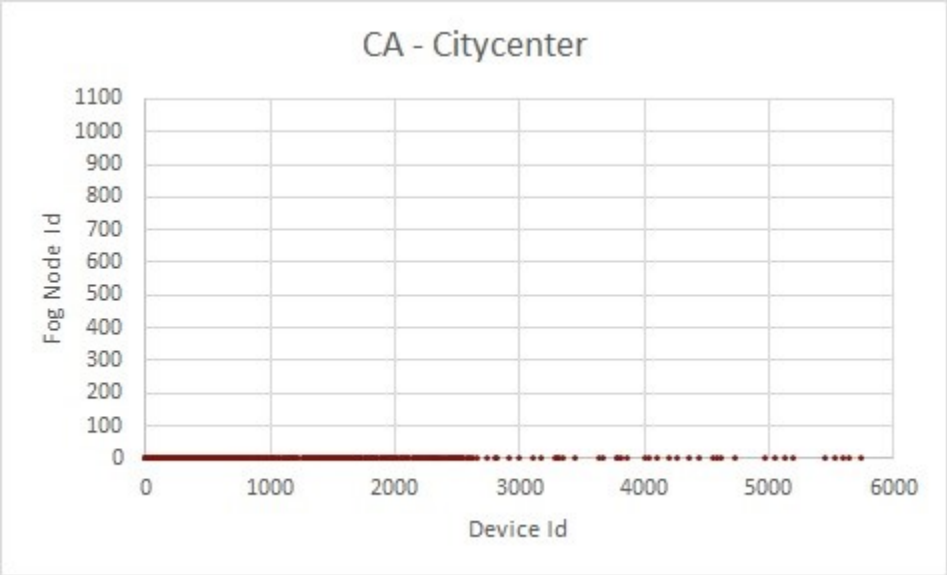

Figure 63: Cognitive Assistance Application – Fog nodes selected by City center Orchestrator for service placement.

Figure 71 shows that fog nodes belonging to all layers are utilized for task execution. Figure 69 shows that the increase in average host utilization percentage at higher device counts has reduced significantly due to high task failure rate.

Figure 74 shows low utilization of network resources with Local placement approach as only one network node is utilized by tasks for data transfer. Additionally, high percentage of

failed tasks resulted in marginal increase in utilization of fog networks at higher device counts, as can be observed from Figure 72.

As Local placement approach considers only one i.e. local fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 75 and the average number of messages exchanged is 2 as shown in Figure 76.

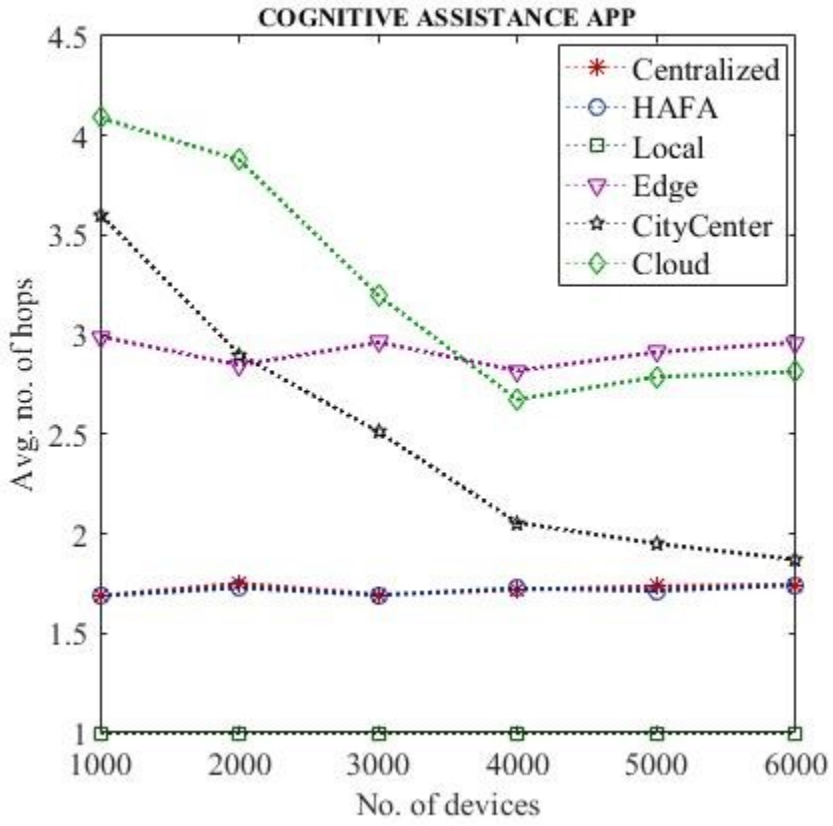

Figure 64: Cognitive Assistance Application – Average hops between device (user) and executing (fog) node vs. number of active devices.

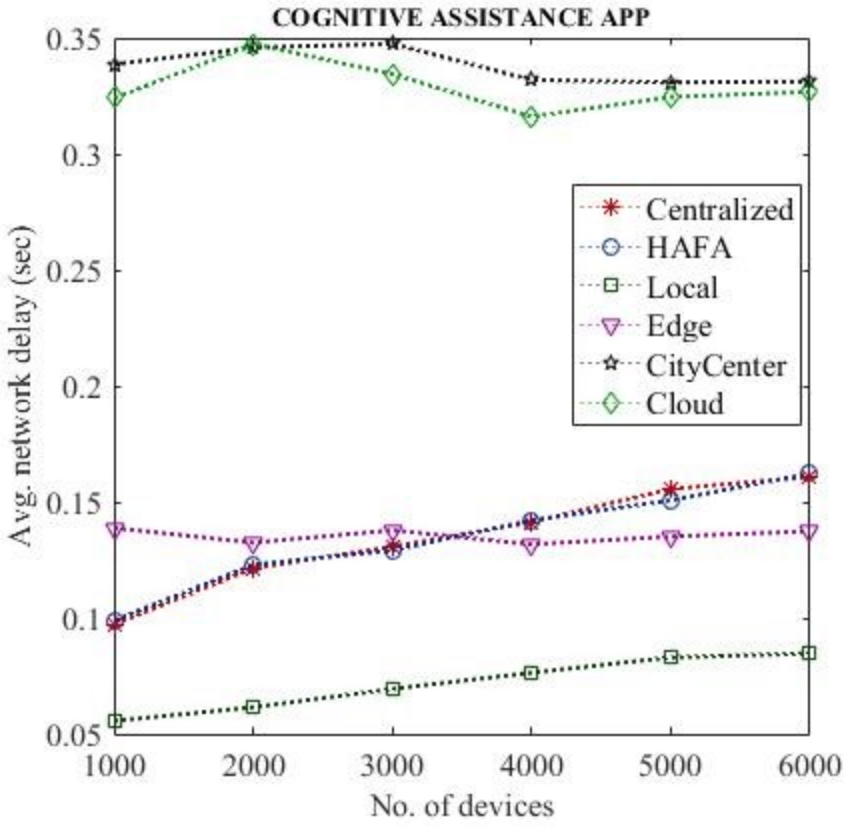

Figure 65: Cognitive Assistance Application – Average network delay per task vs. number of active devices.

City center orchestrator.

As shown in Figure 58, City center orchestrator resulted in high task failure percentage as average network path latency from device/user to City center fog node is high and did not satisfy the CA application latency requirements for a large number of devices.

Figure 59 shows that City center placement approach executed tasks on fog node belonging to layer-6 for the test scenario with 6000 devices.

Figure 60 shows that City center placement approach resulted in least average cost per task as compared to other approaches. However, these values are invalid and can be ignored considering the high task failure percentage with this approach.

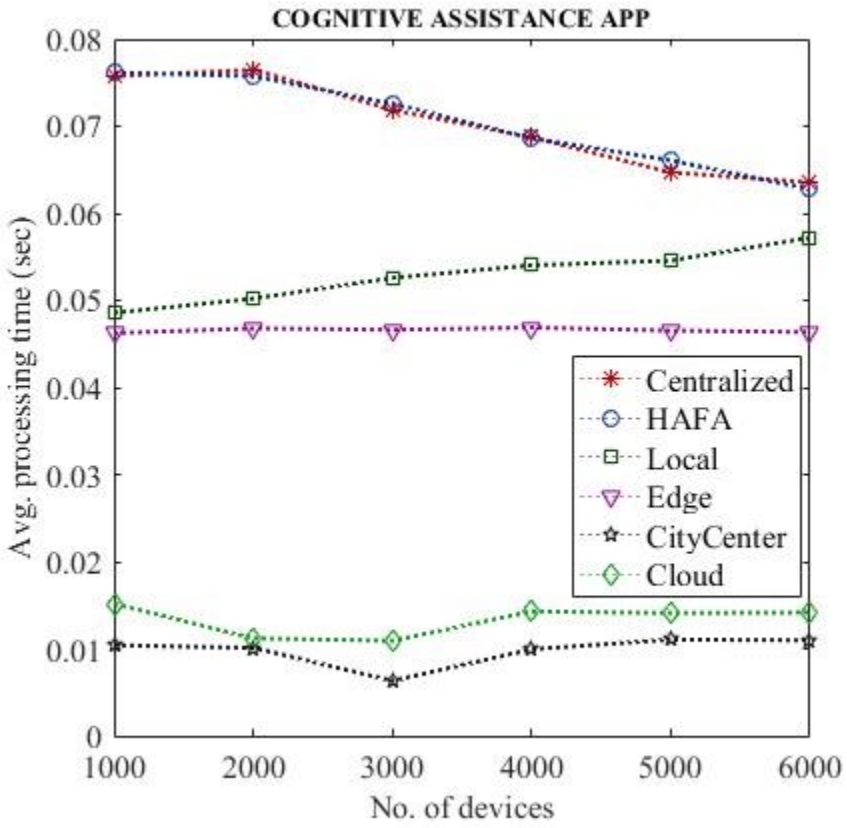

Figure 66: Cognitive Assistance Application – Average processing time per task vs. number of active devices.

Figure 62 shows that at lower device counts, some of the tasks were successfully executed. But at higher device counts, application latency requirements could not be satisfied due to increase in queuing delay, which resulted in high percentage of task failures. However, as the Cloud node is connected over single link to City center fog node, as well as City center node having large compute capacity resulted in low network congestion delay as well as low queuing

delay and execution time at node. Additionally, propagation delay of 6 millisecond over City center to Cloud network link has minor impact on overall task service time. All these factors resulted in successful execution of tasks submitted from device located near Cloud node. This can be observed in Figure 62 as average distance between device/user and City center fog node is about 100 kilometers at higher device counts, whereas the average distance from any fog node location to City center is about 12 kilometers.

Figure 64 shows that the average network distance between device/user and executing fog node has reduced at higher device counts. This is explained by the fact that at higher device counts, congestion delay increases due to increased traffic at network nodes resulting in failure of tasks due to high network delay. Thus, the successful tasks are those submitted by device accessing city center fog node over network paths with less number of hops.

As seen in Figure 65, average network delay is high as statistically more devices are closer to layer-1 and layer-2 fog nodes and access City center fog node over 3 or more network hops. There is no further increase in network delay at higher device counts due to high task failure percentage.

Figure 66 shows that the average processing time is minimal due to high MIPS capacity of City center fog node and nearly non-existent queuing delay due to high task failure rate.

As shown in Figure 67, average service time perceived per task is dominated primarily by high network delay due to very low processing time.

Figure 71 shows that only fog node belonging to layer-6 is utilized for task execution. Figure 69 shows that the average host utilization percentage has remained the same at higher device counts due to high task failure rate.

Figure 74 shows high utilization of network resources with this approach resulting from high data traffic due to centralization of task execution at city center fog node. However, as the number of devices increase, there is no growth observed with network utilization due to large number of task failures, as can be seen in Figure 72.

As this approach considers only one i.e. City center fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 75 and the average number of messages exchanged is 2 as shown in Figure 76.

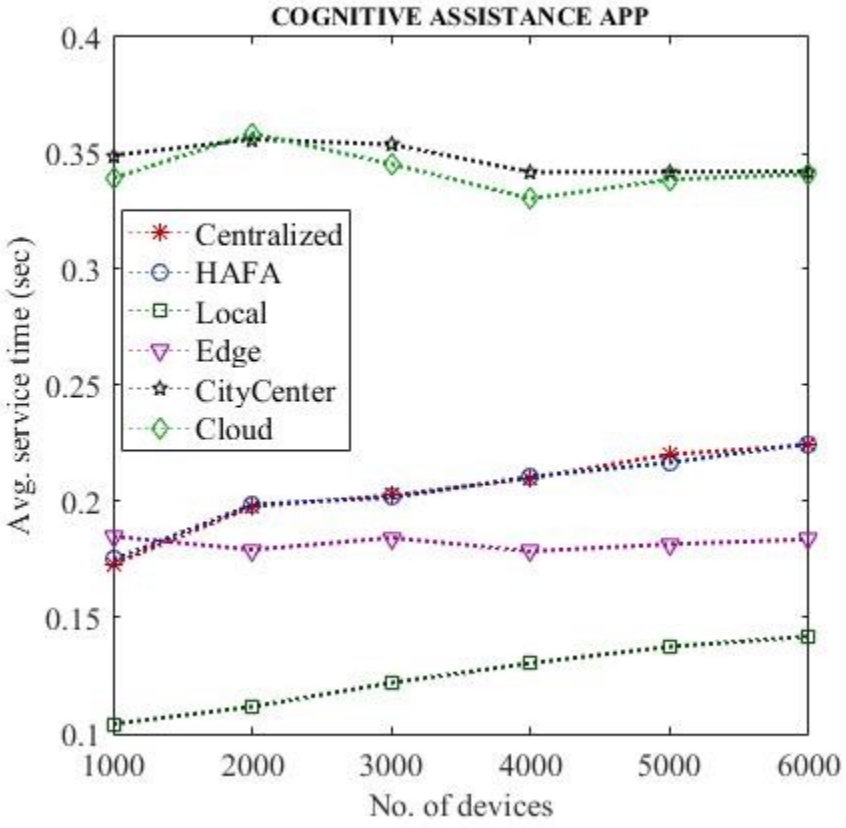

Figure 67: Cognitive Assistance Application – Average service time per task vs. number of active devices.

Cloud orchestrator.

As shown in Figure 58, Cloud orchestrator resulted in highest task failure percentage as average network path latency from device to Cloud node is high and did not satisfy the application latency requirements for a large number of devices.

Figure 59 shows that the Cloud placement approach executed tasks on fog node belonging to layer-7 for the test scenario with 6000 devices.

Figure 60 shows that the Cloud orchestrator resulted in least average cost per task as compared to other approaches. However, these values are invalid and can be ignored considering the high task failure percentage with this approach.

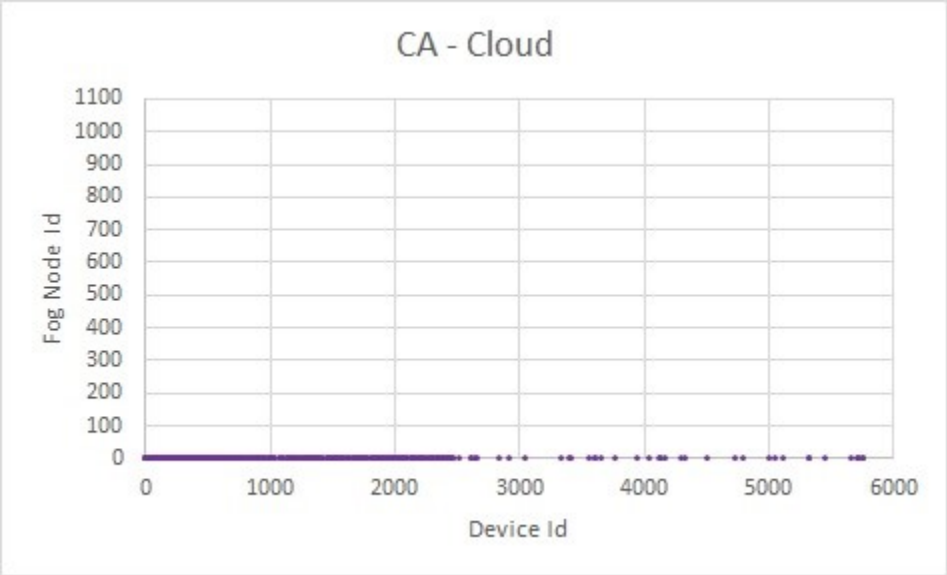

Figure 68: Cognitive Assistance Application – Fog nodes selected by Cloud Orchestrator for service placement.

Figure 62 shows that the average great circle distance between device/user and Cloud node at all device counts is observed to be approximately 700 kilometers.

Figure 64 shows that the average network distance between device and fog node has reduced at higher device counts. This is explained by the fact that at higher device counts, congestion delay increases due to increased traffic at network nodes resulting in failure of tasks due to high network delay. Thus, the successful tasks are those submitted by device accessing city center fog node over network paths with fewer number of hops.

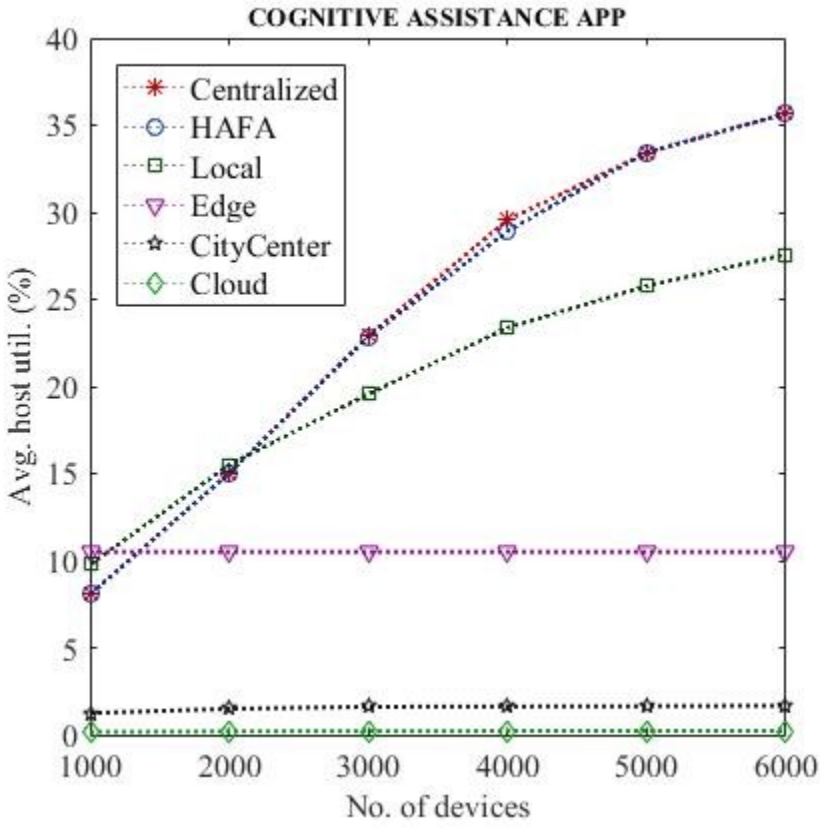

Figure 69: Cognitive Assistance Application – Average host utilization vs. number of active devices.

As seen in Figure 65, average network delay is high as statistically more devices are closer to layer-1 and layer-2 fog nodes and access Cloud node fog node over 4 or more network

hops. There is no further increase in network delay at higher device counts due to high task failure percentage.

Figure 66 shows that the average processing time is minimal due to the high MIPS capacity and large number of CPU cores at Cloud node and nearly non-existent queuing delay due to high task failure rate.

As shown in Figure 67, average service time perceived per task is dominated primarily by high network delay due to very low processing time.

Figure 71 shows that lone node belonging to layer-7 is utilized for task execution. Figure 69 shows that the average host utilization percentage has remained the same at higher device counts due to high task failure rate.

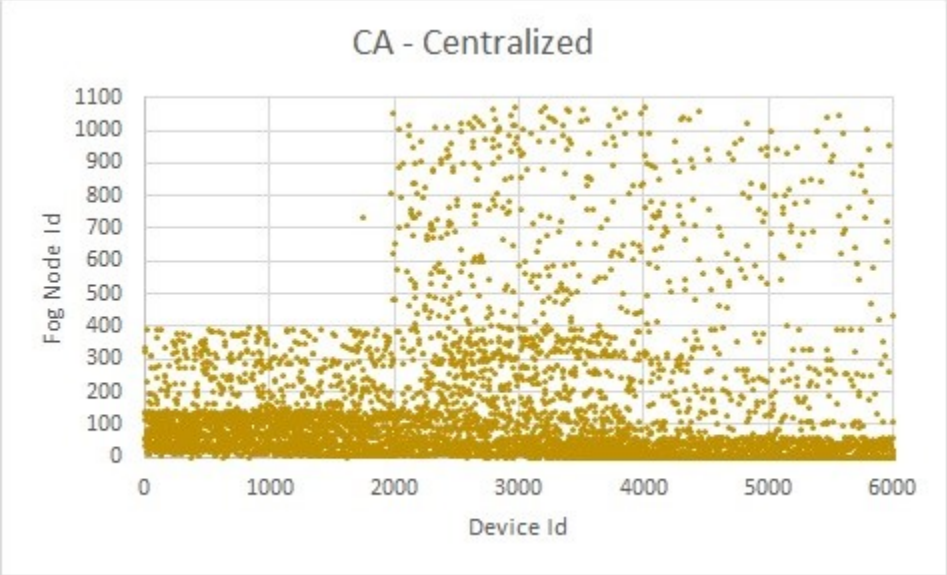

Figure 70: Cognitive Assistance Application – Fog nodes selected by Centralized Orchestrator for service placement.

Figure 74 shows highest utilization of network resources with this approach resulting from high data traffic due to centralization of task execution at Cloud node. However, as the number of devices increase, there is no growth observed with network utilization due to large number of task failures.

As this approach considers only Cloud node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 75 and the average number of messages exchanged is 2 as shown in Figure 76.

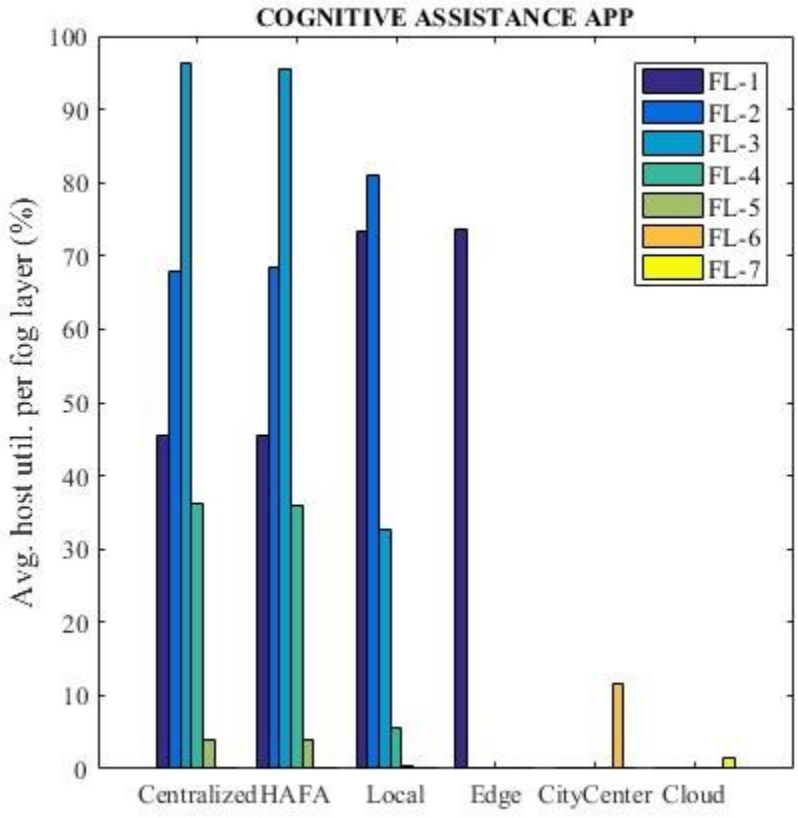

Figure 71: Cognitive Assistance Application – Average host utilization per fog layer with 6000 active devices.

Centralized orchestrator.

As shown in Figure 58, Centralizes placement approach resulted in the least percentage of task failures for CA application as compared to other approaches. It is observed that the tasks failed for two reasons, queuing delay at lower layer fog nodes for test scenarios with low device counts, and failure due to high network latency from increased network congestion delay for test scenarios with higher device counts.

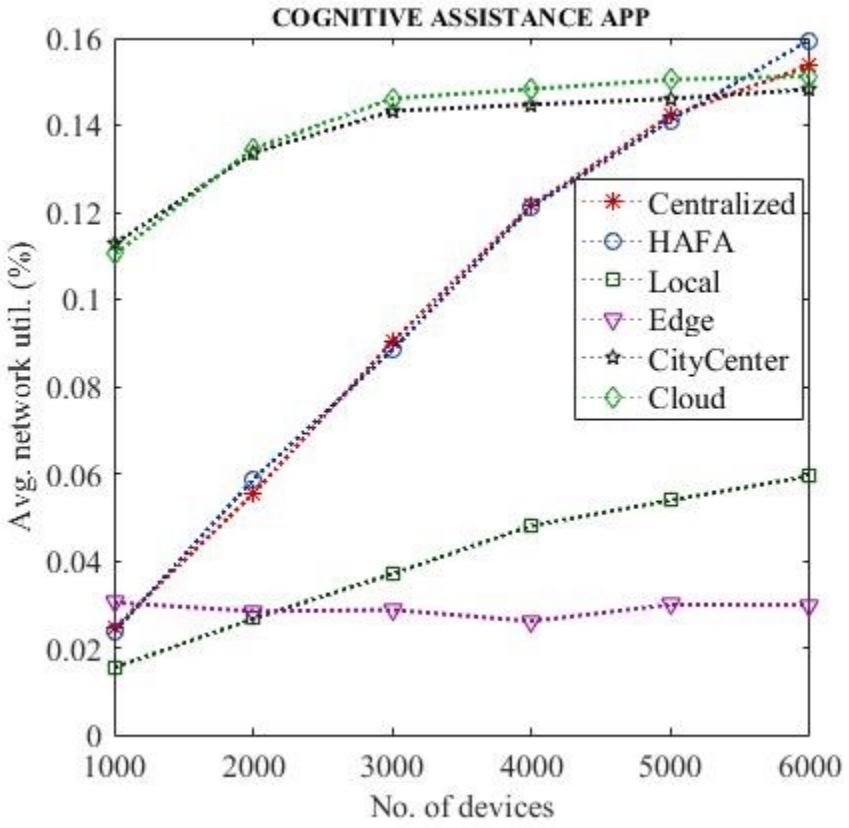

Figure 72: Cognitive Assistance Application – Average network utilization vs. number of active devices.

Figure 59 shows that the successful tasks were executed on nodes belonging to all fog layers for the test scenario with 6000 devices.

Figure 60 shows that the Centralized placement approach resulted in least average cost per task as compared to all other approaches as, in the entire system, it selects the fog node with the least total cost and has sufficient resources available to host a given service.

Figure 62 shows that the Centralized placement approach selected fog nodes in vicinity. At higher device counts, when resources are exhausted in nearby nodes, this approach selected nodes at farther distances; however, the corresponding tasks have failed due to high network latency. Thus, it is observed that there is no change in average great circle distance as the device count is increased. For the same reason, the average number of network hops is observed to remain the same at higher device counts, as can be seen from Figure 64.

Centralized placement approach attempts to reduce communication cost by reducing number of network hops, which indirectly reduces network delay as well. As device counts increase, the average network delay increased due to higher network congestion, as can be seen from Figure 65. However, the network delay is smaller than that with Edge, Cloud, and City center approaches. It is observed to be further lower for Local placement approach. However, the values are invalid and can be ignored considering the high task failure percentage with this approach.

Centralized approach selected nodes accessible at lower number of hops which were primarily belonging to lower layers, for scenarios with low device counts, resulting in higher average processing time. At higher device counts, when resources on lower layer fog nodes were exhausted, higher layer nodes were selected which were accessible at higher number of hops and higher costs. Additionally, at higher device counts, high queuing delay results in failure of tasks submitted to lower layer nodes. Thus, the average processing time is seen as decreasing at higher device counts as can be observed in Figure 66.

As shown in Figure 67, average service time perceived per task is dominated primarily by average network delay for all device counts, and is observed to be less than the application latency requirement of 370 milliseconds. Average service time using Edge and Local placement approaches is seen to be lower than Centralized approach, but they are invalid and can be ignored considering their high task failure rates.

Figure 71 shows the percentage of resources utilized at each fog layer for task execution. It is observed that fog nodes from layers 2-6 were utilized. Average utilization percentage has increased as shown in Figure 69 at a rate higher than that with Edge and Local placement approaches. This can be explained by high task failure rates with Edge and Local placement approaches at higher device counts.

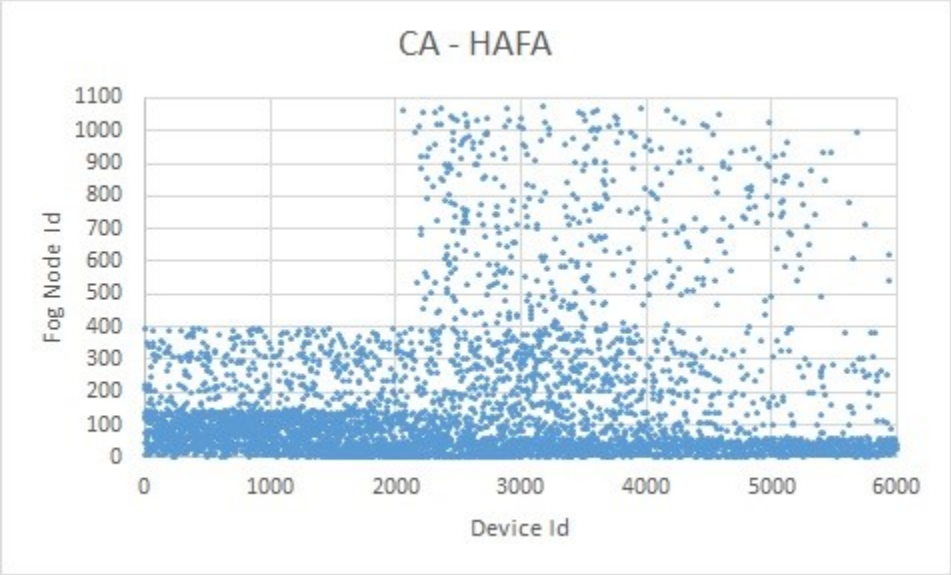

Figure 73: Cognitive Assistance Application – Fog nodes selected by HAFA Orchestrator for service placement.

Figure 74 shows the percentage of network resources utilized at each fog layer. As the nodes selected for placement were accessible at lower number of network hops i.e. nodes

selected were in vicinity, the average network utilization is observed to be consistently lesser than that with City center and Cloud placement approaches. However, the average network utilization is observed to be increasing at higher device counts as can be seen from Figure 72.

As Centralized placement approach considers all available nodes in system for placement, the number of prospective nodes is 1074 for all device counts as shown in Figure 75 and the average number of messages exchanged towards making the placement decision is 2148 as shown in Figure 76.

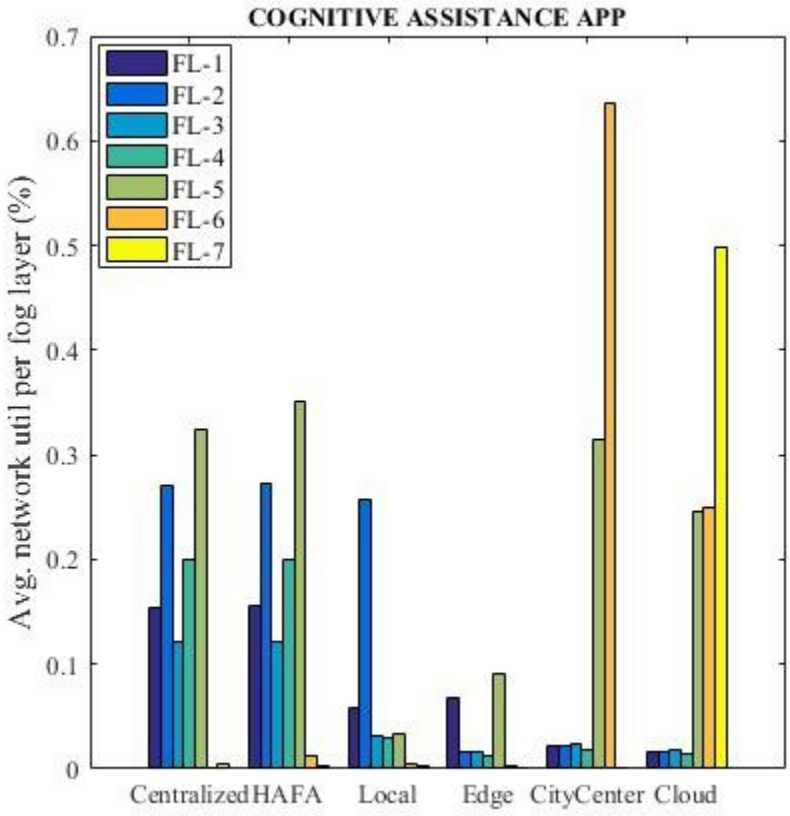

Figure 74: Cognitive Assistance Application – Average network utilization per fog layer with 6000 active devices.

HAFA orchestrator.

As shown in Figure 58, HAFA orchestrator resulted in task failures in a manner similar to Centralized placement approach for the same reasons as described in earlier subsection. Figure 59 shows that the successful tasks were executed on all layers for the test scenario with 6000 devices.

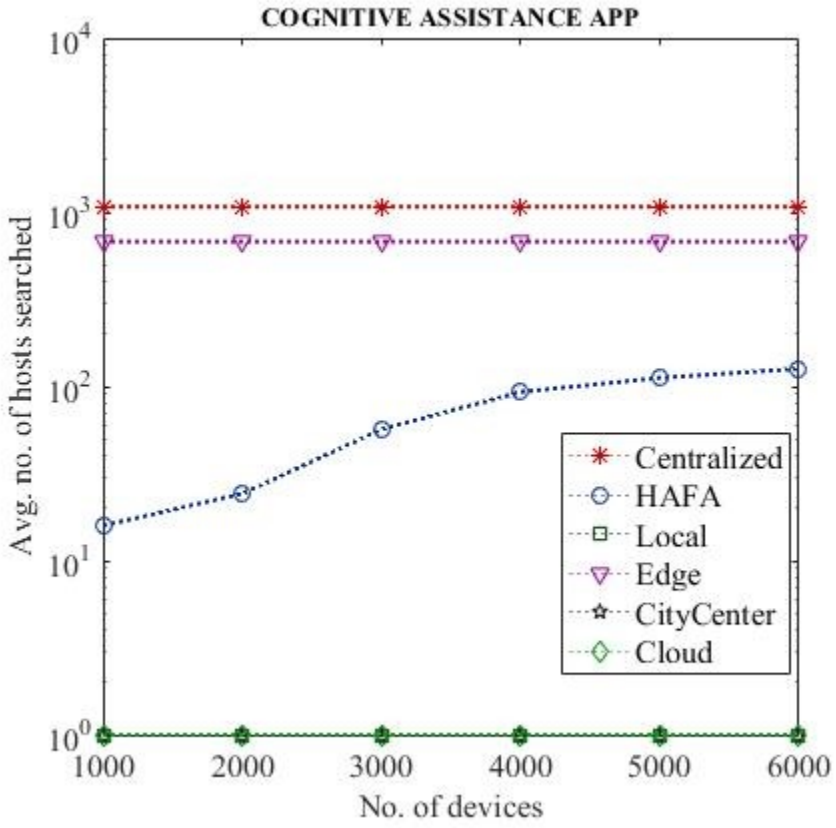

Figure 75: Cognitive Assistance Application – Average number of hosts searched vs. number of active devices.

Figure 60 shows that HAFA placement approach resulted in least average cost per task as compared to all other approaches, similar to Centralized placement approach, as HAFA orchestrator selects nodes in vicinity and extends the search only so far that it finds a node which

can satisfy the given service resource requirements. Thus, the identified fog node from each layer is likely to be accessible over a small number of hops, reducing communication costs. As nodes belonging to different layers have different execution costs in our test environment, the fog node with overall least cost comprising computation and communication costs is selected among the ones identified one per fog layer. It is observed that the average cost per task is similar to that with Centralized placement approach for all device counts.

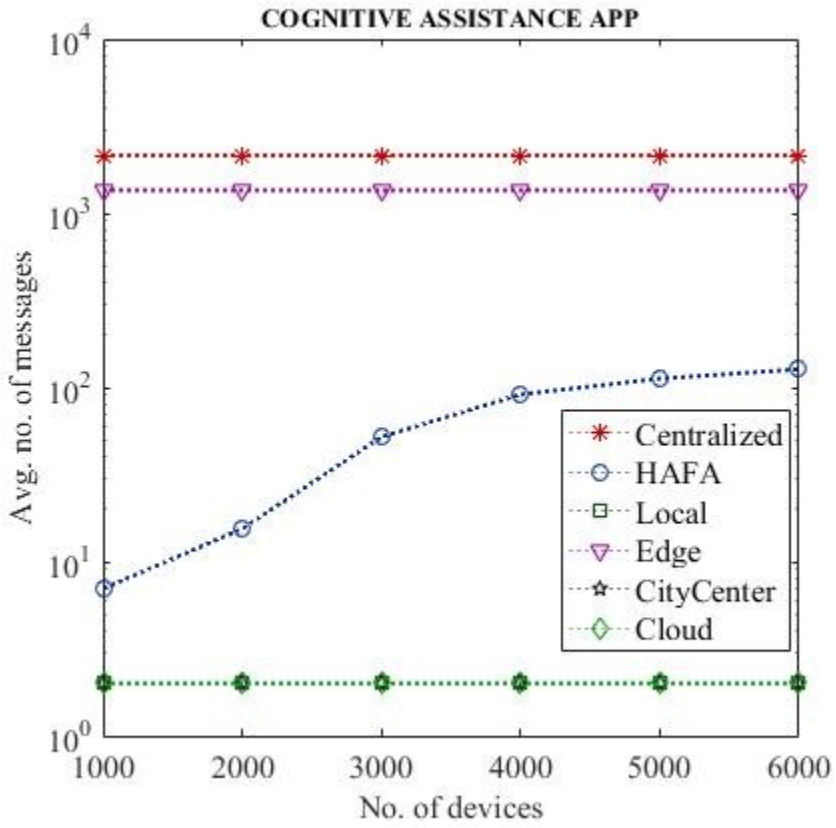

Figure 76: Cognitive Assistance Application – Average number of messages exchanged vs. number of active devices.

As the number of devices scale and local resources are exhausted, HAFA orchestrator expands the search to neighbors. However, due to latency constraints for CA application, tasks

submitted to farther fog nodes accessible at higher number of hops experienced failures due to increased network congestion delay. Thus, the average great circle distance between device and fog node has remained the same at higher device counts. This behavior is observed to be similar to that with Centralized placement approach as shown in Figure 62. Similar pattern is observed for average number of network hops for the same reason described above, as can be seen from Figure 64. Average great circle distance and average network distance, i.e. number of network hops were the least as compared to other placement approaches, except the Local placement approach.

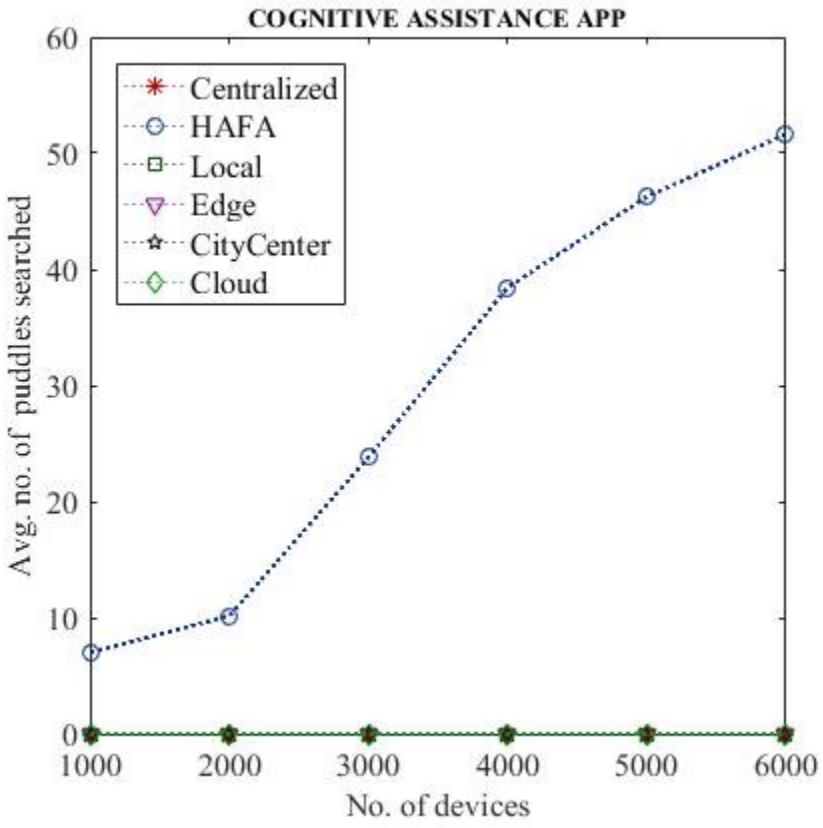

Figure 77: Cognitive Assistance Application – Average number of Puddles searched vs. number of active devices.

As can be observed from Figure 65, Figure 66, and Figure 67, average network delay, average processing time, and average service time with HAFA placement approach are observed to be similar to that from Centralized placement approach, as both the approaches attempt to reduce the average total cost for the tasks, which is the sum of compute cost and communication cost.

Figure 71 and Figure 74 shows that HAFA placement approach utilized fog nodes from all layers towards task execution, in a behavior similar to that of Centralized placement approach. Thus, the average host compute and network utilization percentages are similar to centralized approach for all device counts as shown in Figure 69 and Figure 72 respectively.

HAFA orchestrator considers member nodes belonging to only local Puddle initially and expands the search space only to immediate neighbors leveraging PuddleTree parent/child logical links until the search for fog node with required free resources is successful or the search space is exhausted. Thus, the number of prospective nodes considered towards placement of each service in fog environment may be different and are significantly lesser as compared to the number of prospective nodes considered by Centralized placement approach.

The average number of prospective nodes with HAFA approach for CA application profile is observed to be 16 nodes for test scenario with 1000 devices for all device counts which steadily increased with higher device counts to 126 nodes for test scenario with 6000 devices. Similarly, the number of messages exchanged have ranged from 7 for 1000 devices to 127 for 6000 devices, and the number of Puddles that comprised the search space ranged from 7 for 1000 devices to 51 for 6000 devices. These patterns can be seen in Figure 75 and Figure 76. The reason for steep increase in these metrics is as follows. As the number of devices increase, resources on local fog nodes are exhausted and search space is expanded to immediate

neighbors. However, the neighbors may not be able to satisfy the low latency requirements for CA applications due to longer network path lengths, in spite of having free compute resources. Thus, the search space is further expanded to neighboring nodes. However, the chances of successful identification of fog node reduces even further for the same reasons. Thus, eventually all node belonging to the layer are considered as prospective nodes resulting in higher average number of prospective nodes, average number of Puddles searched, and average number of messages exchanged during the search.

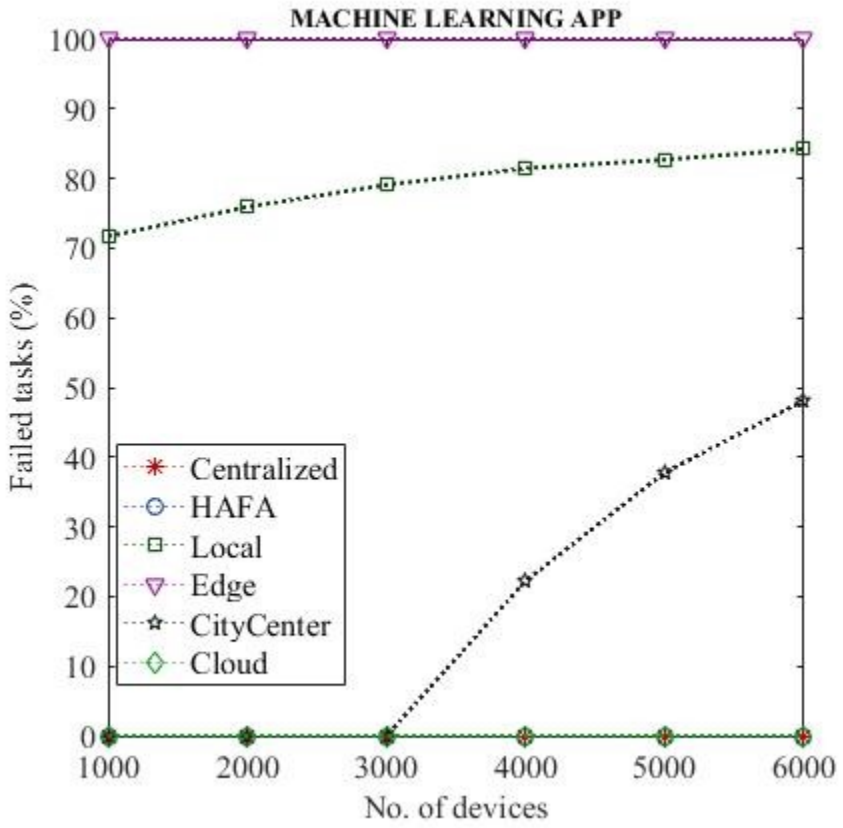

Figure 78: Machine Learning Application – Failure rate of tasks vs. number of active devices.

7.11.15. Application: Machine Learning

Scenario. Imagine a smart classroom with technology to capture the behavior and attentiveness of students while the class is in session. This scenario is realized using surveillance cameras which continuously record the actions of each student. Local analytics are performed on streamed video continuously to identify any misbehavior in class, any imminent threats, or simply capture student involvement in class or attention towards the teacher and topic being discussed.

Application Profile. Tasks were generated at a Poisson mean inter-arrival rate of one task in 5 seconds to ensure that the students are monitored throughout the classroom session.

This application is assumed to be latency-insensitive and has high computation and medium communication requirements. Response time limit for each task is set to 10000 seconds, as the system is not expected to be responsive.

Each task requires execution of 6000 million instructions, on average, and uses one CPU core. Input to the task is a high definition image or set of video frames representing classroom video recording and has an average size of 1500 kilobytes. Output for the task is simply an acknowledgement, or notification of any abnormality observed, or feedback regarding students' behavior in classroom and has an average size of 25 kilobytes.

Results analysis. Tests were performed with six service placement approaches using Machine Learning application workload for various device counts. Observations from the tests are as follows.

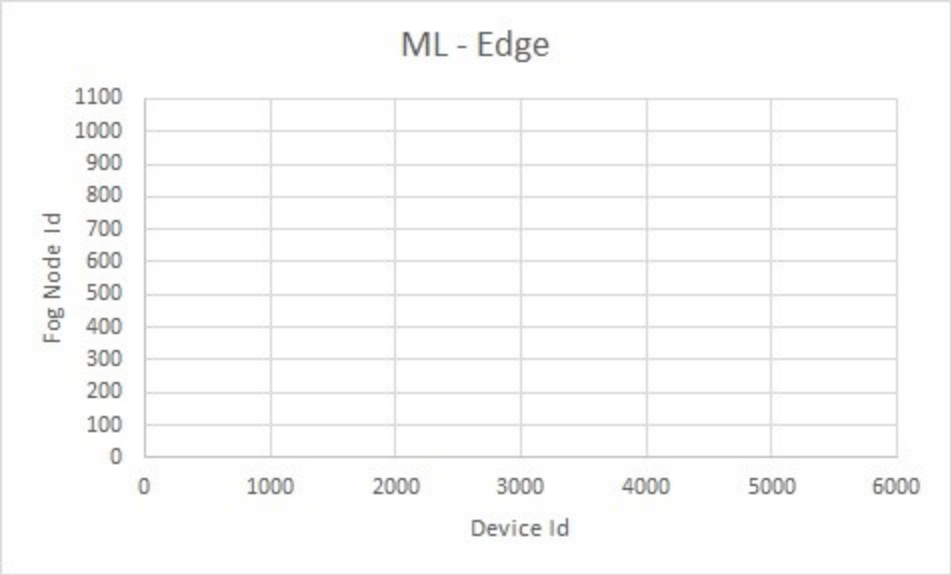

Figure 79: Machine Learning Application – Fog nodes selected by Edge Orchestrator for service placement.

Edge orchestrator.

As shown in Figure 78, Edge orchestrator resulted in 100% task failure rate for ML application as the resource configurations of layer-1 fog nodes are not sufficient to host these applications. This is reflected in Figure 81 which shows that there were no successful tasks executed by Edge placement approach for the test scenario with 6000 devices. Accordingly, average cost per task is observed to be zero as shown in Figure 82, average great circle distance between device/user and fog node is observed to be zero in Figure 84, average network distance between device/user and fog node is observed to be zero network hops in Figure 85, average network delay is observed to be zero in Figure 86, average processing time per task delay is observed to be zero in Figure 88, average service time perceived per task is observed to be zero in Figure 89, average host utilization percentage is observed to be zero in Figure 90 and Figure 91, and average network utilization percentage is observed to be zero in Figure 92 and Figure 95.

As Edge placement approach considers all available layer-1 nodes for placement, the number of prospective nodes is 679 for all device counts as shown in Figure 96 and the average number of messages exchanged, comprising both request and response messages, towards making the placement decision is 1358 as shown in Figure 97. Note the representation of Y-axis in log scale for Figure 96 and Figure 97.

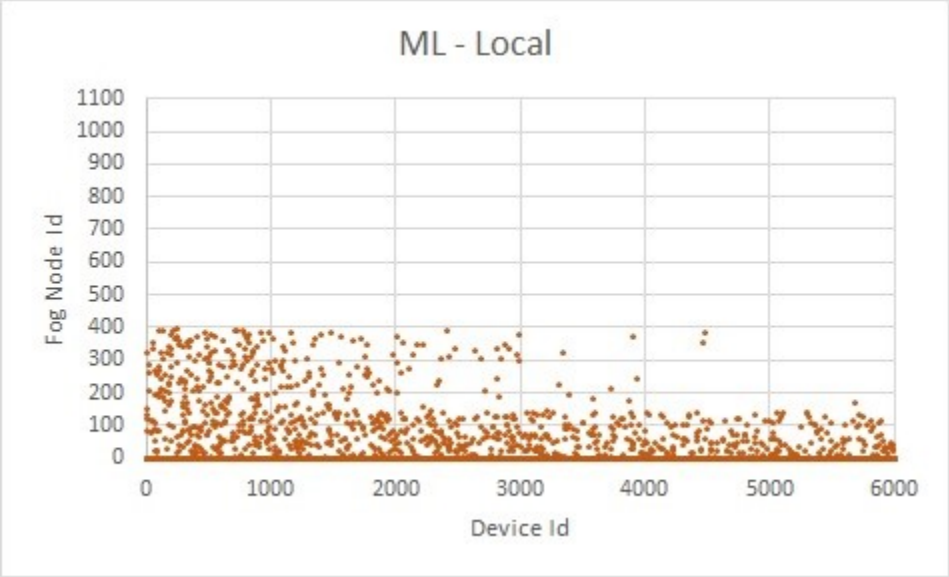

Figure 80: Machine Learning Application – Fog nodes selected by Local Orchestrator for service placement.

Local orchestrator.

As shown in Figure 78, Local orchestrator resulted in higher task failure percentage as compared to all placement approaches other than Edge orchestrator, as local nodes may belong to any fog layer and not just the resource-constrained layer-1. Task failure rate is higher than other placement approaches as the lower layer fog nodes may not have sufficient resources to execute all the tasks submitted to it. This also results in increased task failure percentage at higher device counts as shown in Figure 78.

Figure 81 shows that the Local placement approach executed tasks on fog nodes belonging to all layers except layer-1 for the test scenario with 6000 devices.

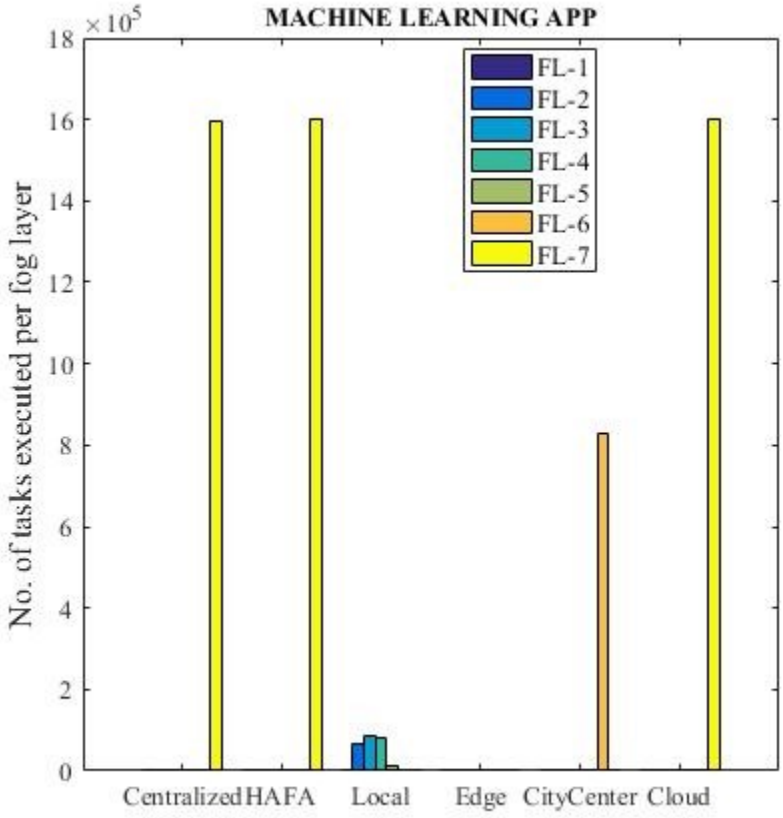

Figure 81: Machine Learning Application – Number of tasks successfully executed per fog layer with 6000 active devices.

Figure 82 shows that the Local placement approach resulted in least average cost per task as compared to all approaches except Edge placement approach. As all nodes are local to devices, the communication cost is nonexistent. It is also observed that the average cost per task has reduced at higher device counts. This can be explained by the reasoning that, as device counts increase, the task failure rate increases for those submitted to fog nodes belonging to lower layers due to exhaustion of resources and most of the tasks which were successfully

executed were those by higher layer fog nodes. In our test environment, execution cost on higher layer fog nodes is lesser as compared to that on lower layer fog nodes, hence total cost is reduced at higher device counts as communication cost is non-existent with Local placement approach.

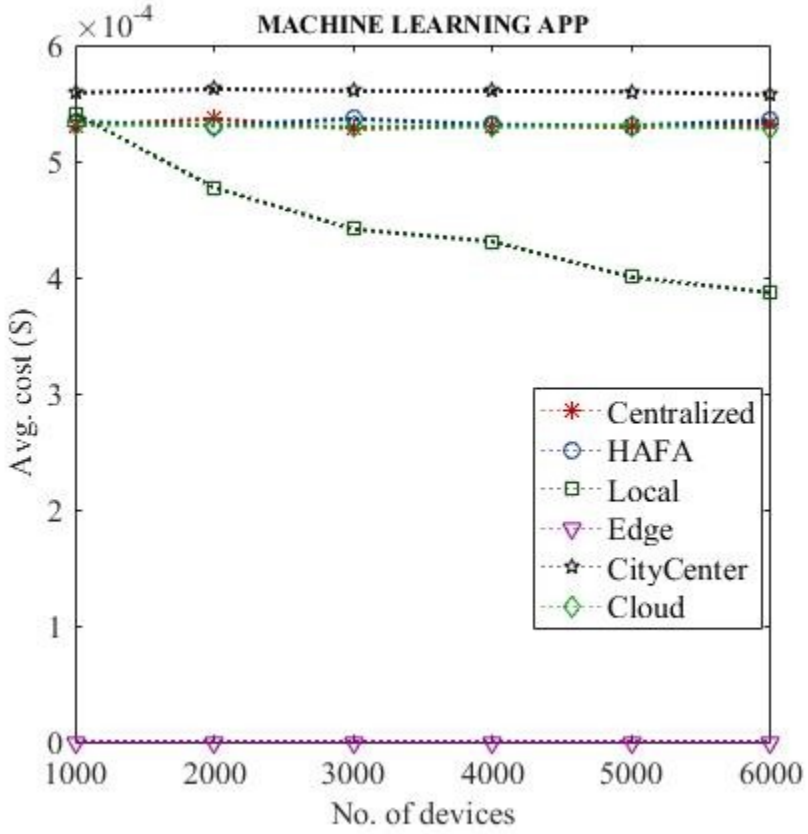

Figure 82: Machine Learning Application – Average execution cost vs. number of active devices.

Figure 84 shows that the average great circle distance between device/user and executing fog node is always zero for all device counts.

Figure 85 shows that the average network distance between device/user and executing fog node is always one for all device counts as the fog node is always accessible at a single hop from device.

As seen in Figure 86, average network delay is the least as it comprises only the congestion delay at first hop network node. Network propagation delay is always zero due to co-location of device and local fog node.

Figure 88 shows that the average processing time has decreased at higher device counts when higher percentage of successful tasks are executed by fog nodes belonging to higher layers for which execution costs are lower.

As shown in Figure 89, average service time perceived per task is dominated primarily by high processing times and followed the same pattern for all device counts, as network delay is minimal with this approach.

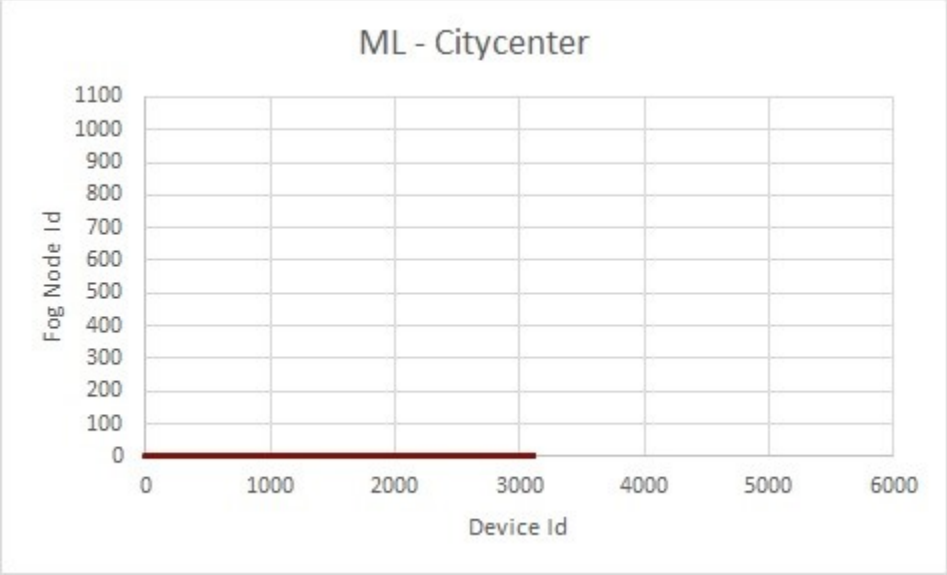

Figure 83: Machine Learning Application – Fog nodes selected by City center Orchestrator for service placement.

Figure 91 shows that fog nodes belonging to layers 2-5 were utilized for task execution. Figure 90 shows that the average host utilization percentage has increased steadily as

comparatively more requests are served by lower layer fog nodes which have smaller resource configurations, resulting in higher average host utilization percentage at higher device counts.

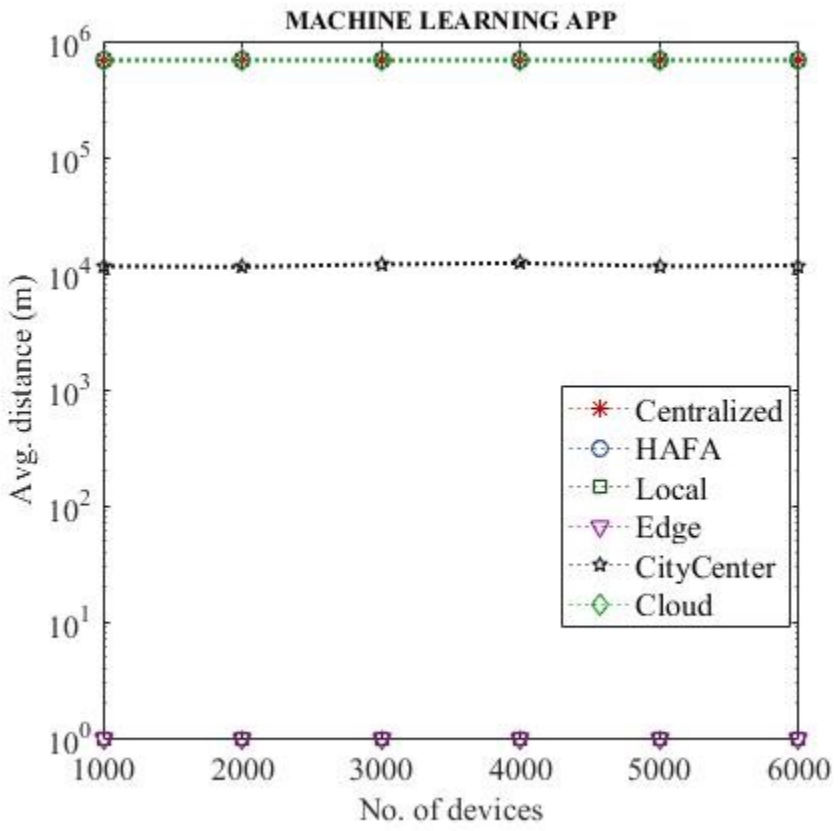

Figure 84: Machine Learning Application – Average distance between device (user) and executing (fog) node vs. number of active devices.

Figure 95 shows low utilization of network resources with Local placement approach as only one network node is utilized by tasks for data transfer, most of which were low capacity network nodes. Additionally, high percentage of failed tasks resulted in marginal increase in utilization of fog networks at higher device counts, as can be observed from Figure 92.

As Local orchestrator considers only one i.e. local fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 96 and the average number of messages exchanged is 2 as shown in Figure 97.

City center orchestrator.

As shown in Figure 78, task failure percentage with City center orchestrator was zero for device counts up to 3000 beyond which it increased steadily as resources at City center fog node were exhausted and insufficient to support the workload from higher device counts.

Figure 81 shows that the City center placement approach executed tasks only on fog node belonging to layer-6 for the test scenario with 6000 devices.

Figure 82 shows that the City center placement approach resulted in higher average cost per task as compared to Cloud placement approach as the execution cost on City center fog node is slightly higher than that on Cloud node. There is no change observed at higher device counts as 100% of the tasks submitted by devices beyond 3000 count are failed to execute, thus there is no change in average cost per task which is averaged over only the successfully completed tasks.

Figure 84 shows that there is no change in average great circle distance between device and City center fog node, at higher device counts. It is observed to be approximately 10 kilometers.

Figure 85 shows that there is no change in average network distance between device and fog node, at higher device counts. It is observed to be approximately 3.8.

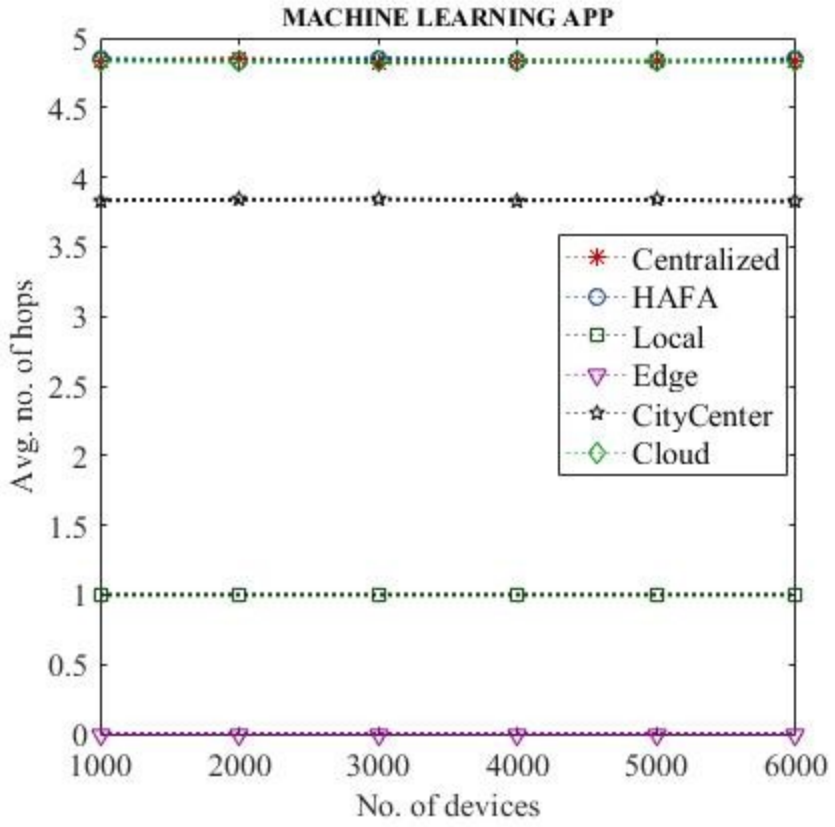

Figure 85: Machine Learning Application – Average hops between device (user) and executing (fog) node vs. number of active devices.

As seen in Figure 86, average network delay has increased linearly up to 3000 device count due to high congestion delay, past which there is no change, as all the additional requests received at higher device counts are marked as failed due to exhaustion of compute resources at city center fog node.

Figure 88 shows that the average processing time is minimal due to the high MIPS capacity of city center fog node.

As shown in Figure 89, average service time perceived per task is dominated primarily by high network delay due to low processing time, and followed similar pattern for all device counts.

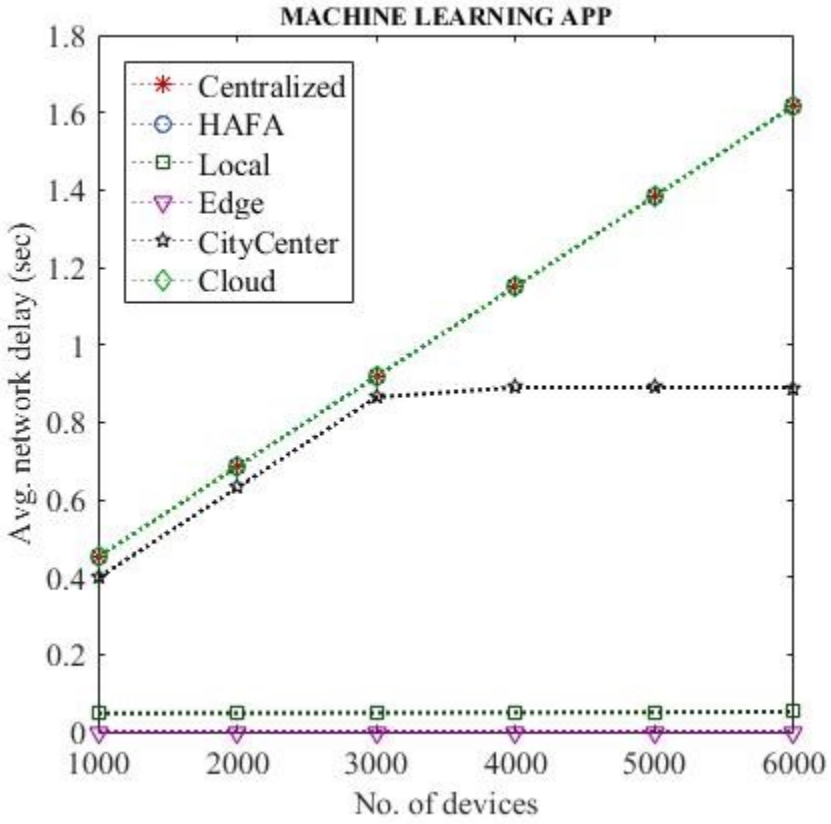

Figure 86: Machine Learning Application – Average network delay per task vs. number of active devices.

Figure 91 shows that only fog node belonging to layer-6 is utilized for task execution and that 100% of host compute resources are utilized. Figure 90 shows that the average host utilization percentage has steadily increased for device counts up to 3000 and has not changed at higher device counts.

Figure 95 shows utilization of network resources is lesser than that with Cloud approach due to failure of tasks at higher device counts, which can also be observed from Figure 92.

As this approach considers only one i.e. City center fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 96 and the average number of messages exchanged is 2 as shown in Figure 97.

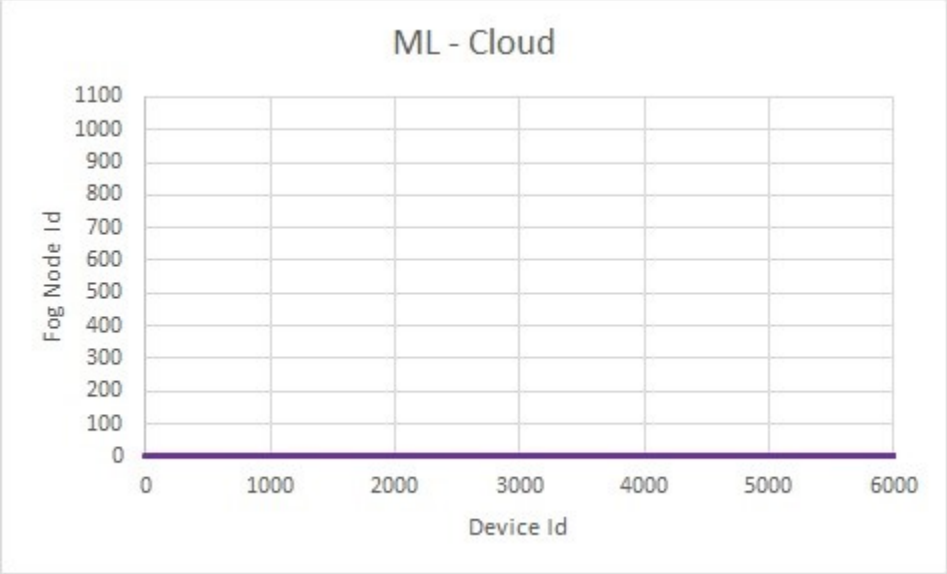

Figure 87: Machine Learning Application – Fog nodes selected by Cloud Orchestrator for service placement.

Cloud orchestrator.

As shown in Figure 78, Cloud orchestrator resulted in zero task failure percentage as Cloud node has sufficient resources to support the workload for given device counts.

Figure 81 shows that the Cloud placement approach executed tasks on node belonging to layer-7 for the test scenario with 6000 devices.

Figure 82 shows that the Cloud placement approach resulted in lower average cost per task as compared to City center placement approach. This can be attributed to the lower execution cost on Cloud node. The cost is observed to be higher than Local placement approach. However, this can be ignored due to high failure rate with Local placement approach.

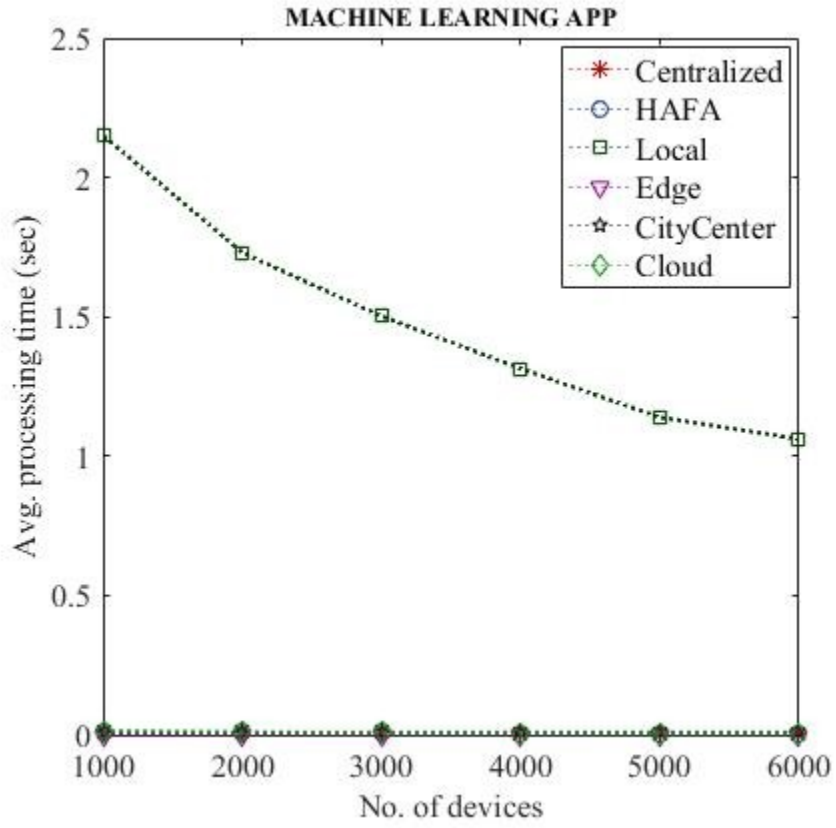

Figure 88: Machine Learning Application – Average processing time per task vs. number of active devices.

Figure 84 shows that there is no change in average great circle distance between device/user and Cloud node at higher device counts. It is observed to be approximately 700 kilometers.

Figure 85 shows that there is no change in average network distance between device/user and Cloud node at higher device counts. It is the highest of all approaches and is observed to be approximately 4.8 as Cloud node is the farthest and is accessible at one additional hop as compared to City center fog node, for the tested network topology.

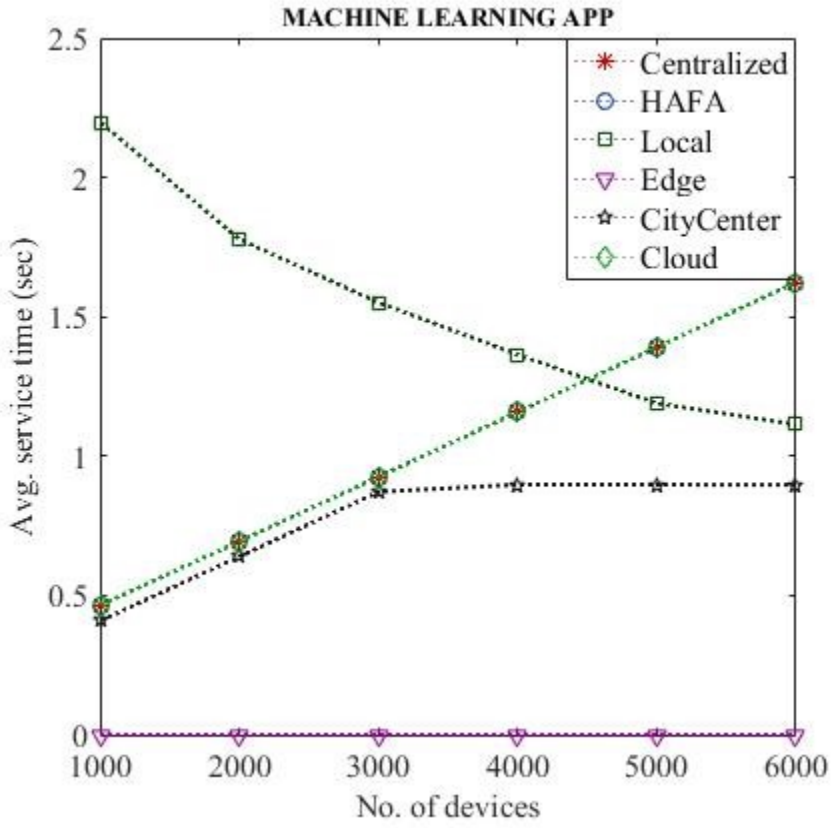

Figure 89: Machine Learning Application – Average service time per task vs. number of active devices.

As seen in Figure 86, average network delay has increased linearly at higher device counts due to the increase in network congestion delay. This pattern is similar to that of City center placement approach for device counts up to 3000 devices. The higher network delay resulted from the additional network hop on all device paths to Cloud node, along with the

increase in propagation delay as Cloud node is the farthest of all nodes and is approximately 700 km away from the city.

Figure 88 shows that the average processing time is minimal due to the high MIPS capacity of Cloud node.

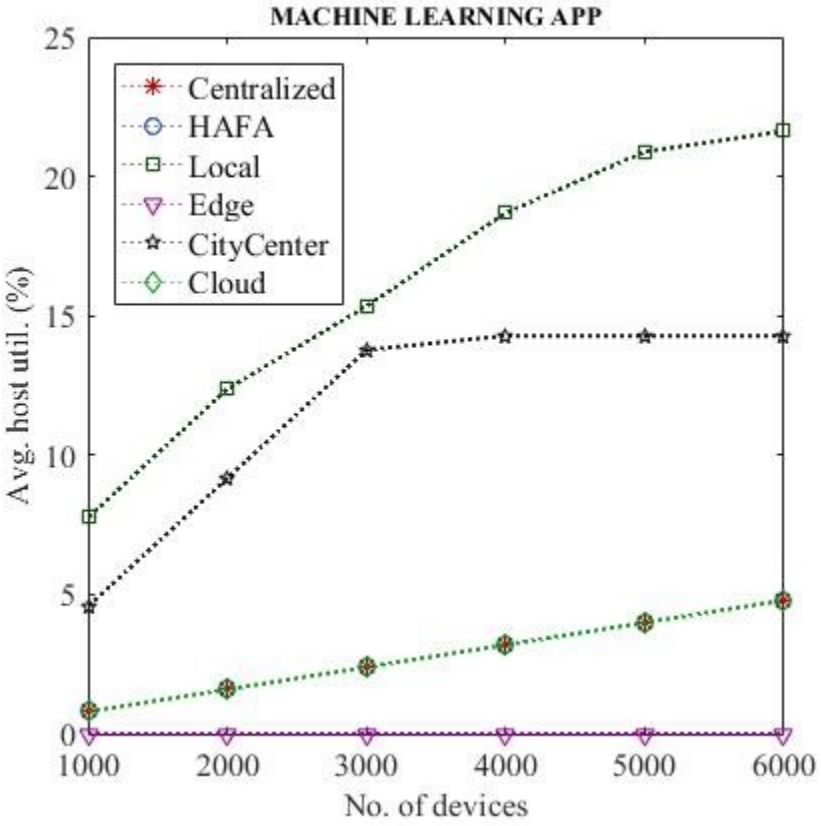

Figure 90: Machine Learning Application – Average host utilization vs. number of active devices.

As shown in Figure 89, average service time perceived per task is highest for this approach and is dominated primarily by high network delay due to low processing time.

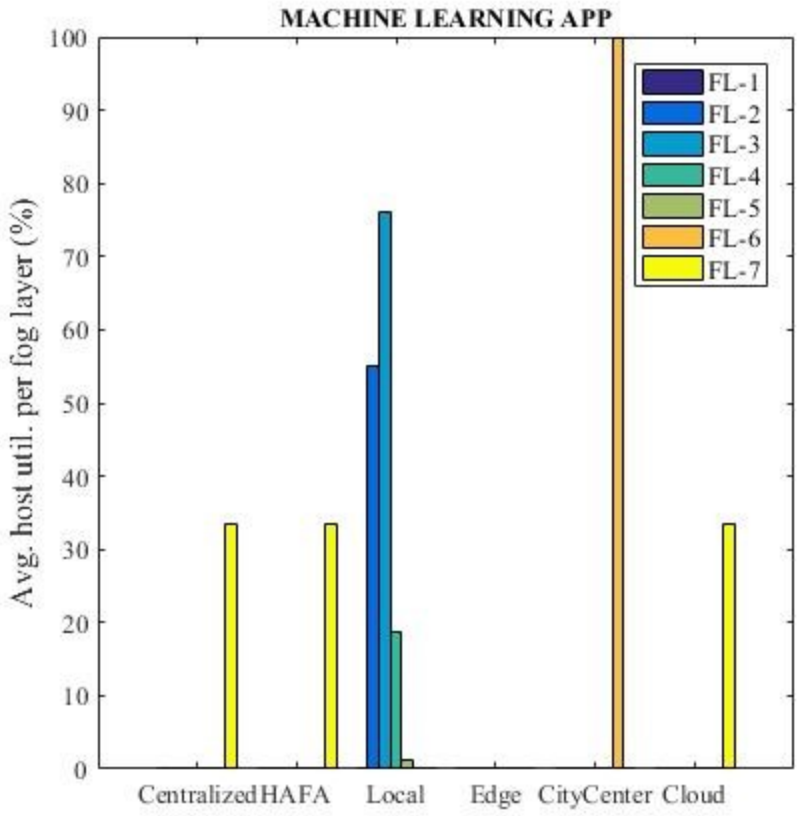

Figure 91: Machine Learning Application – Average host utilization per fog layer with 6000 active devices.

Figure 91 shows that only node belonging to layer-7 is utilized for task execution. Figure 90 shows that the average host utilization percentage has steadily increased as all the tasks are served by Cloud node. However the rate of increase is lesser as compared to that with City center placement approach, owing to the larger resource configuration of Cloud node, which is set to approximately 1000X in our test environment.

Figure 95 shows highest utilization of network resources with this approach resulting from high data traffic across all network nodes. Additionally, the rate of increase is higher as

compared to that with City center placement approach, owing to the additional network hop in our test environment as can be observed from Figure 92.

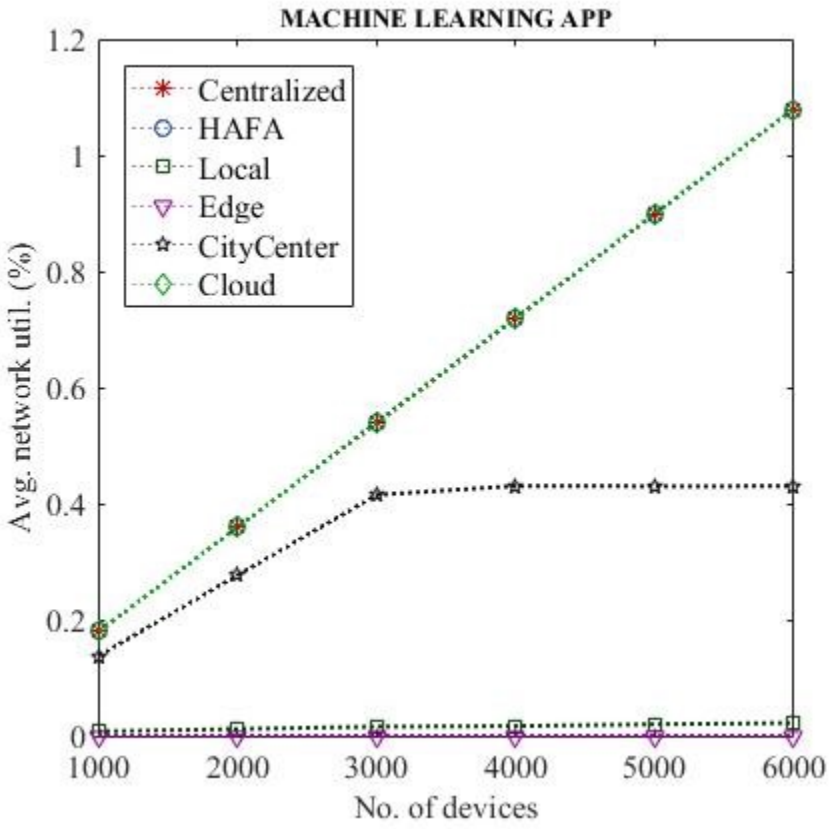

Figure 92: Machine Learning Application – Average network utilization vs. number of active devices.

As this approach considers only Cloud node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 96 and the average number of messages exchanged is 2 as shown in Figure 97.

Figure 81 shows that the Centralized and HAFA placement approaches executed tasks submitted by all devices only on node belonging to layer-7, i.e. Cloud node for the test scenario with 6000 devices.

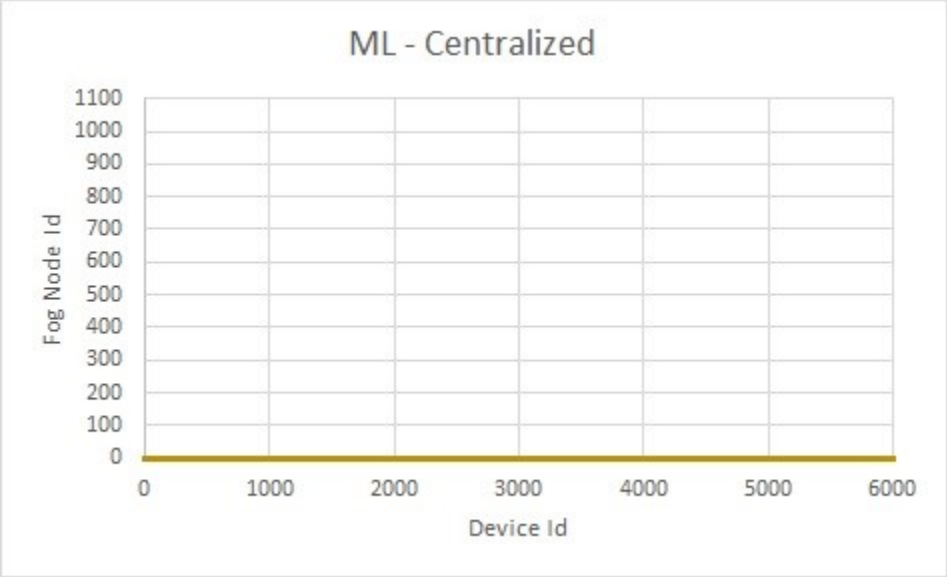

Figure 93: Machine Learning Application – Fog nodes selected by Centralized Orchestrator for service placement.

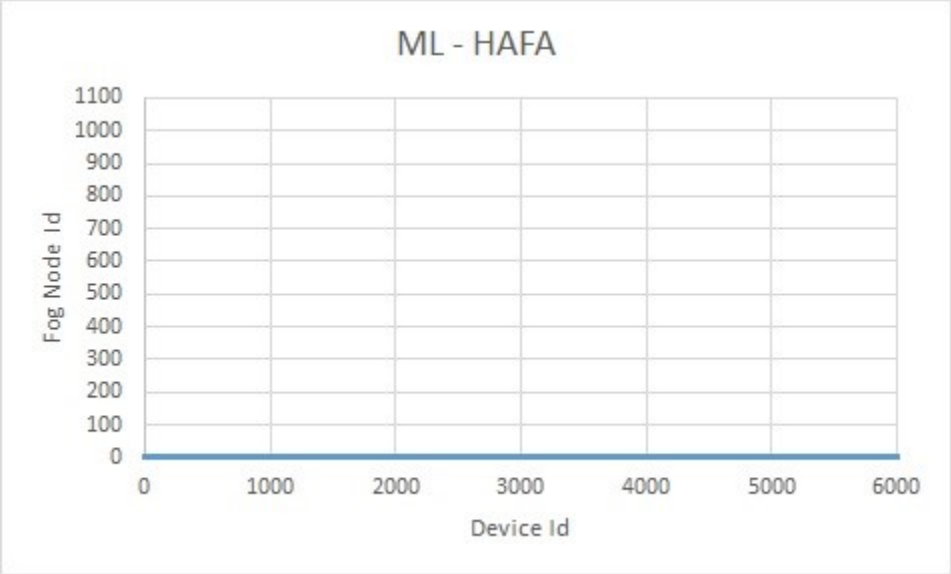

Figure 94: Machine Learning Application – Fog nodes selected by HAFA Orchestrator for service placement.

Centralized orchestrator and HAFA orchestrator.

As shown in Figure 78, the Centralized and HAFA orchestrators resulted in zero task failure percentage as Cloud node has sufficient resources to support the ML application workload for given device counts.

Figure 82 shows that the Centralized and HAFA orchestrators resulted in least average cost per task as compared to other approaches, as these approaches attempt to select the fog node with least total cost for computation and communication.

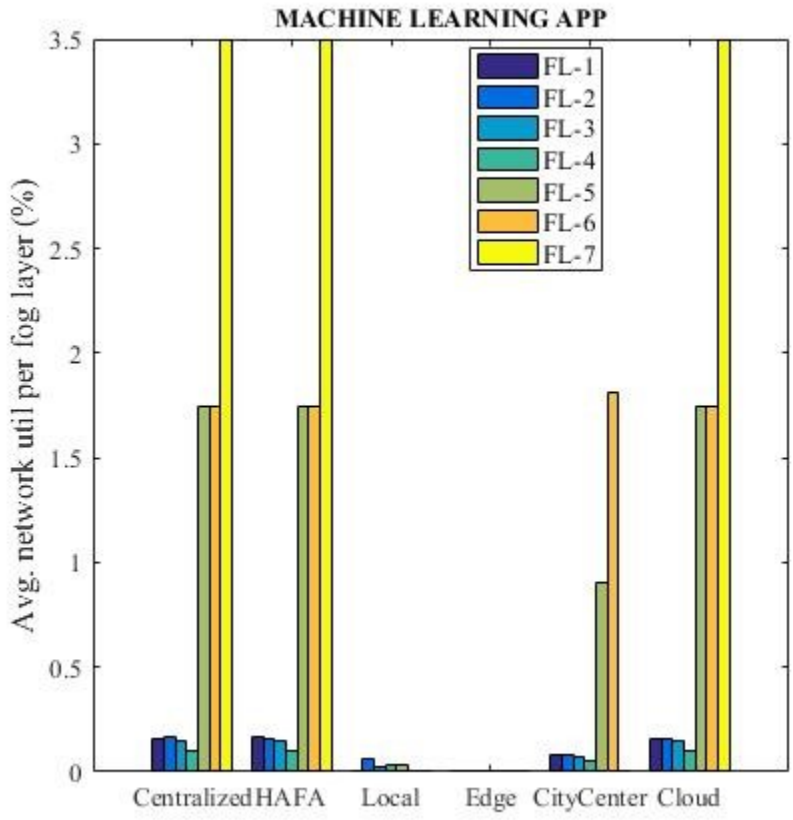

Figure 95: Machine Learning Application – Average network utilization per fog layer with 6000 active devices.

Figure 84 shows that there is no change in average great circle distance between device/user and executing fog node has remained the same at higher device counts. It is observed to be approximately 700 kilometers, as all tasks are executed only on Cloud node. For the same reason, no change is observed in average network distance between device and node at higher device counts.

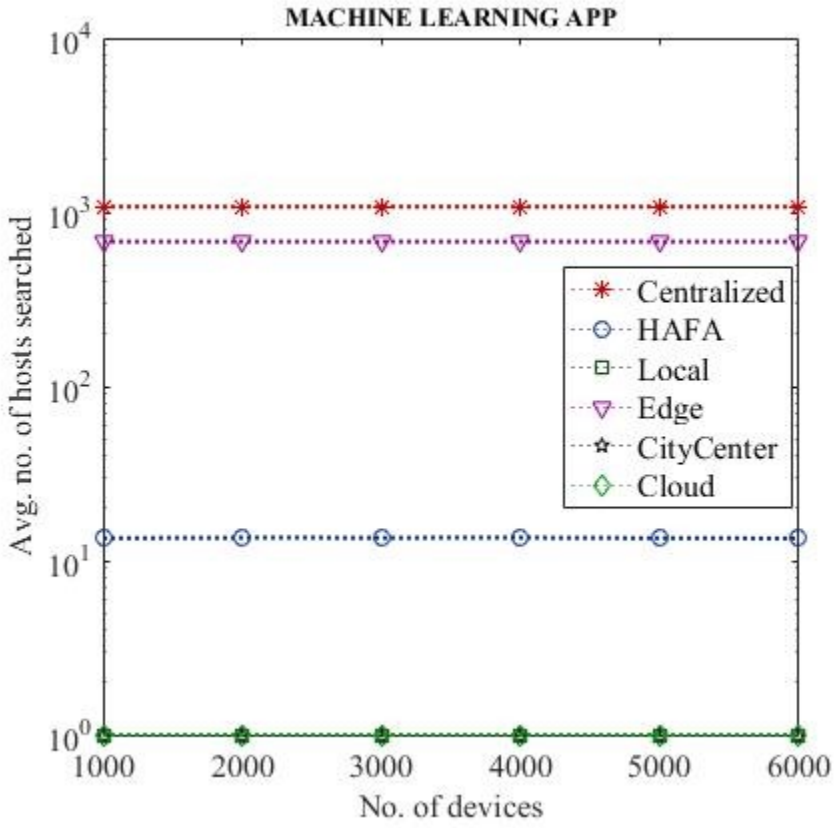

Figure 96: Machine Learning Application – Average number of hosts searched vs. number of active devices.

As can be observed from Figure 86, Figure 88, and Figure 89, average network delay, average processing time, and average service time with these approaches are observed to be

similar to that from Cloud placement approach, as Centralized and HAFA approaches selected Cloud node to execute tasks from all devices.

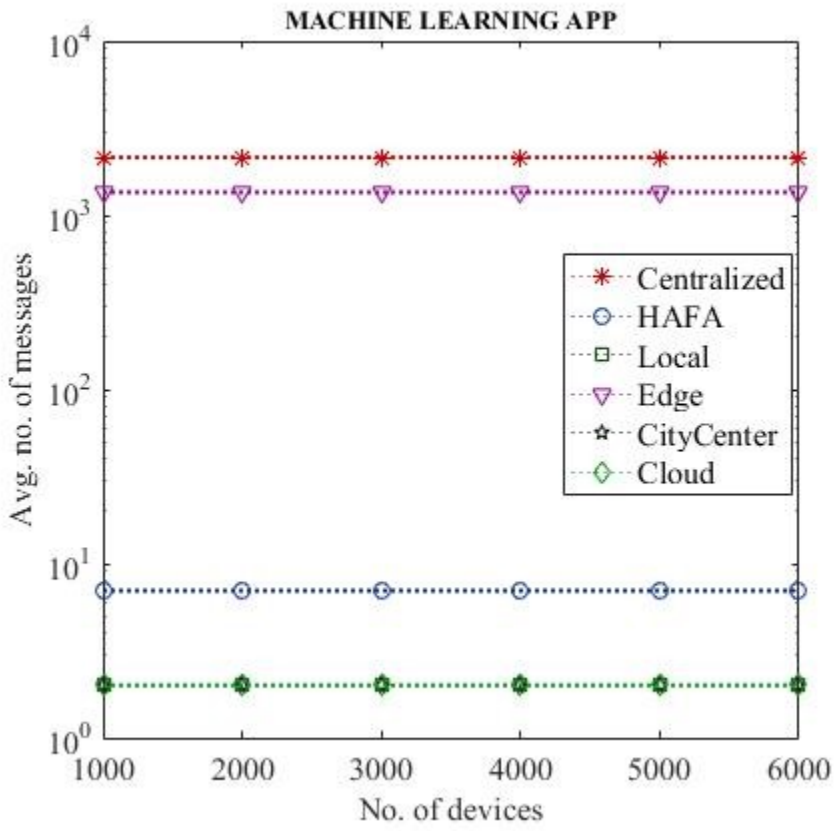

Figure 97: Machine Learning Application – Average number of messages exchanged vs. number of active devices.

Figure 91 and Figure 95 showed that these approaches utilized only Cloud node towards task execution. Thus, the average host compute and network utilization percentages are similar to those from Cloud placement approach for all device counts as shown in Figure 90 and Figure 92 respectively.

As Centralized placement approach considers all available nodes in system for placement, the number of prospective nodes is 1074 for all device counts as shown in Figure 96

and the average number of messages exchanged towards making the placement decision is 2148 as shown in Figure 97.

With HAFA placement approach, as the number of devices scale and local resources are exhausted, HAFA orchestrator expands the search to neighbors. HAFA approach considers member nodes belonging to only local Puddle initially and expands the search space only to immediate neighbors leveraging PuddleTree parent/child logical links until the search for fog node with required free resources is successful or the search space is exhausted.

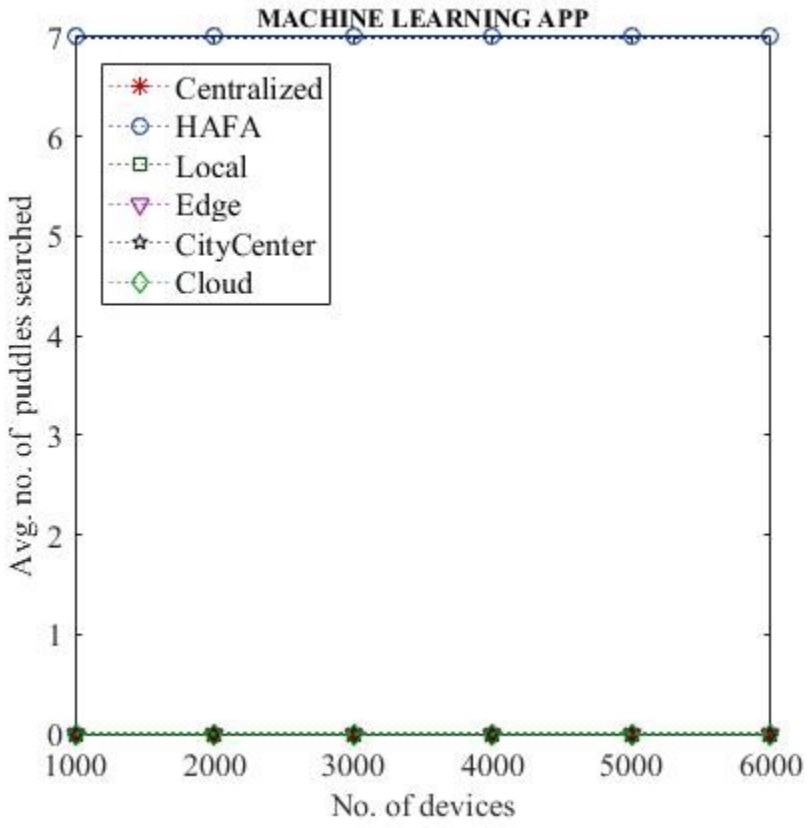

Figure 98: Machine Learning Application – Average number of Puddles searched vs. number of active devices.

For the Machine Learning application profile, all the services are hosted on Cloud node, thus the resources on nodes comprising local Puddle at each fog layer were always 100% free, and the first prospective node and Puddle considered for each layer was identified for that particular layer. Then, among these set of identified nodes, one per fog layer, the node with least total cost considering both computation and communication costs is selected to host the given service i.e. to execute all tasks submitted by corresponding device. As ML application is computation intensive and Cloud node is the one with least execution cost in our test environment such that the total cost is the least of all fog nodes, Cloud node was selected for each service request. Thus, the search for a node per layer was always successful in local Puddle resulting in low average number of messages exchanged, and prospective nodes considered.

As can be seen in Figure 96 and Figure 97, with HAFA approach for ML application profile for all device counts, the average number of prospective nodes considered were 13, average number of messages exchanged were 7, and average number of Puddles that comprised the search space were 7.

7.11.16. Application: Remote Healthcare

Scenario. Imagine the convenience of remote healthcare facility at the comforts of one's own home or place of choice. Patients are free to move around and continue their usual life, as they are not restricted to be in premises of a healthcare institute or a monitoring facility. Vitals are captured continuously or at regular intervals and processed in a near-instantaneous manner. Any aberrations in the observations are acted upon quickly, relevant authorities are notified, and appropriate actions are taken immediately, which may prove to be lifesaving at times such as sudden fall by elders, heart stroke, changes in blood glucose, or blood pressure, etc.

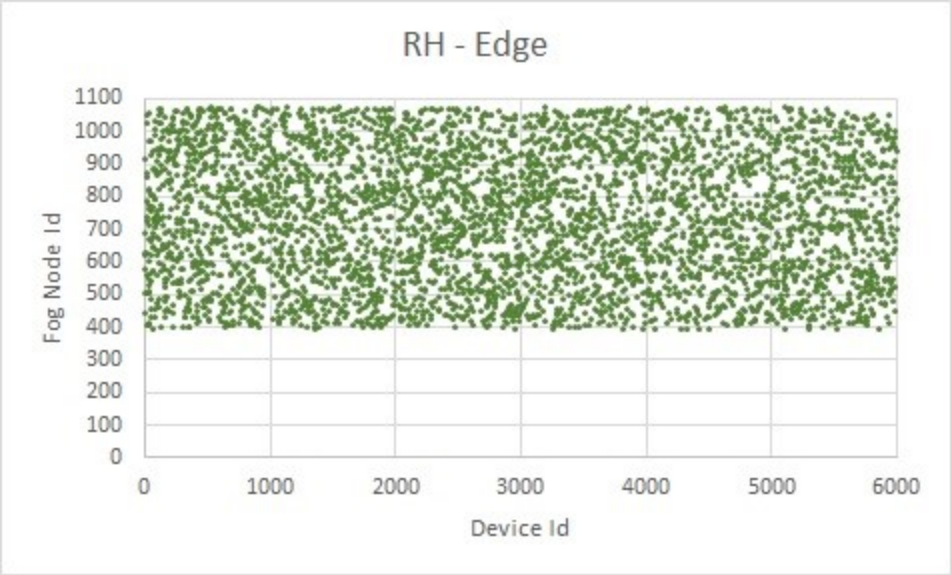

Figure 99: Remote Healthcare Application – Fog nodes selected by Edge Orchestrator for service placement.

Application Profile. Tasks were generated at a Poisson mean inter-arrival rate of one task in 5 seconds to ensure that the patient is monitored continuously and at ad hoc instants when a mishap occurs.

This application is assumed to be latency-critical and has low computation and communication requirements. Response time limit for each task is set to 100 milliseconds, as larger the delay, higher is the impact of mishap on patient, which may even prove to be life-threatening.

Each task requires execution of 300 million instructions, on average, and uses one CPU core. Input to the task is a set of readings representing body vitals and has an average size of 4 kilobytes. Output for the task is the notification of any abnormality observed from readings and has an average size of 1 kilobyte.

Results analysis. Tests were performed with six service placement approaches using Remote Healthcare application workload for various device counts. Observations from the tests are as follows.

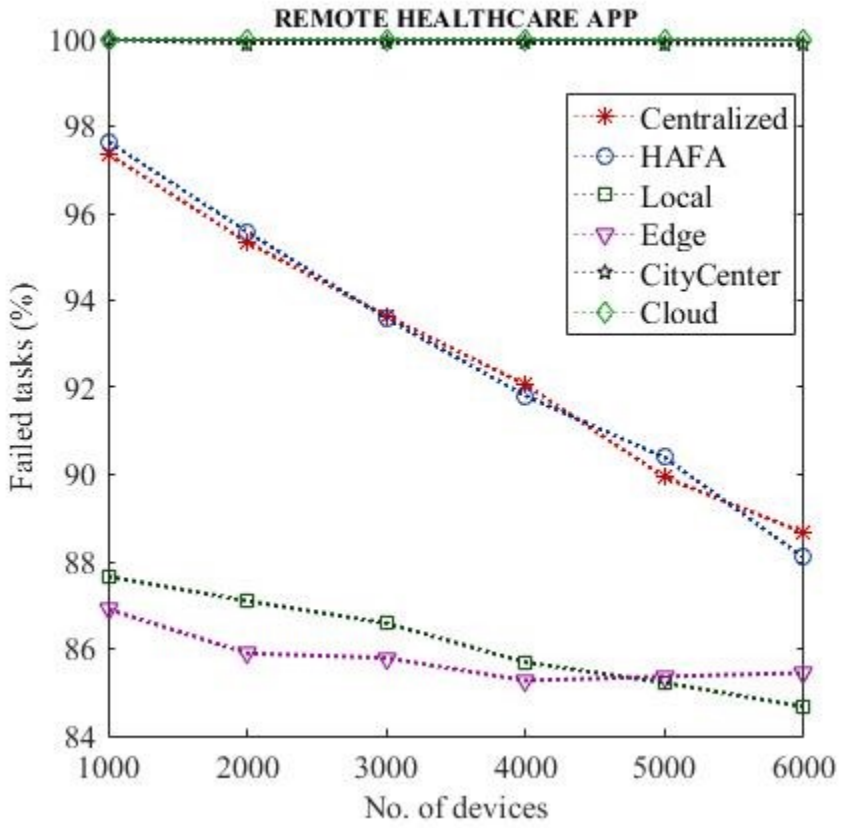

Figure 100: Remote Healthcare Application – Failure rate of tasks vs. number of active devices.

Edge orchestrator.

Remote Healthcare (RH) application has strict latency requirements. Thus, task failure rate grows slowly as queuing delay increases at higher device counts.

As shown in Figure 100, Edge placement approach resulted in high task failure rate for RH application primarily due to high queuing delay at edge nodes at lower device counts and due to exhaustion and hence non-availability of compute resources at higher device counts as can be seen in Figure 101 for the test scenario with 6000 devices.

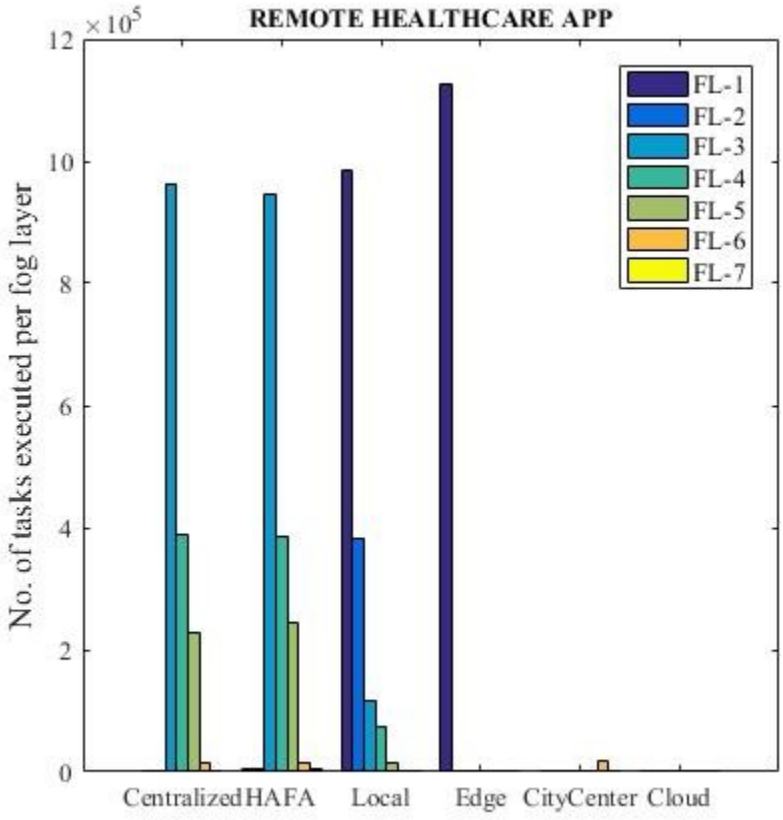

Figure 101: Remote Healthcare Application – Number of tasks successfully executed per fog layer with 6000 active devices.

Figure 102 shows that Edge orchestrator resulted in higher average cost per task as compared to all other approaches, the reason being high compute cost on layer-1 fog nodes. It is observed that the average cost per task has reduced at higher device counts as the tasks are

executed mostly on local edge nodes due to exhaustion of resources at other edge nodes in network, thus eliminating communication cost component from total cost for those tasks.

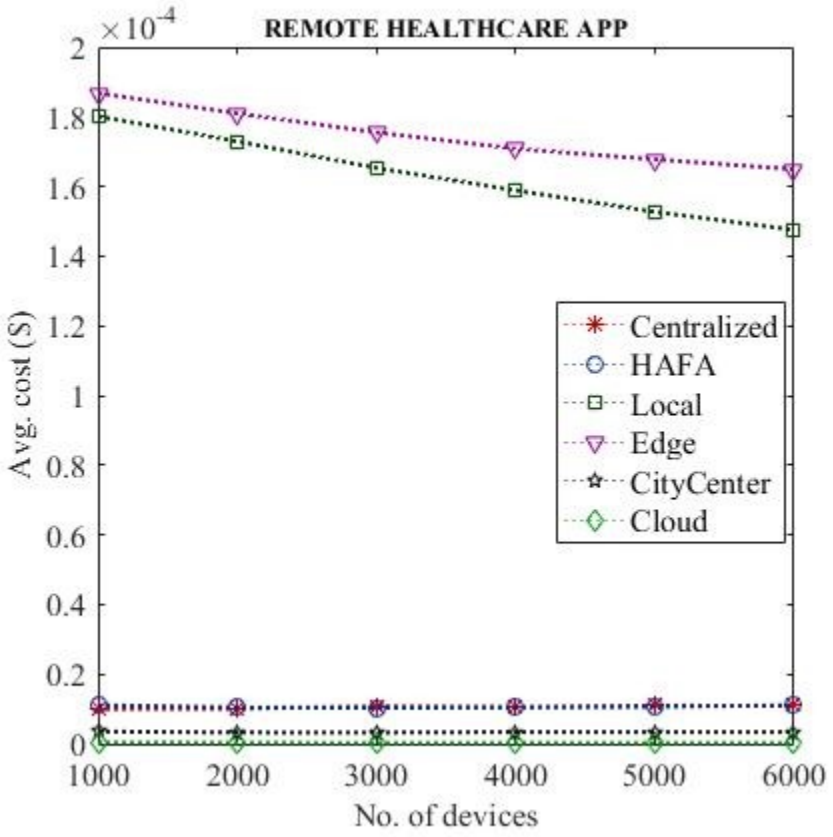

Figure 102: Remote Healthcare Application – Average execution cost vs. number of active devices.

Figure 103 depicts the average great circle distance between device/user and executing fog node. It is observed to be nearly constant at about 60 meters. Note that this value is not zero which shows that some tasks are executed by remote edge nodes as well and not just by local nodes. Note that the average distance with Edge placement approach is lower than that with Centralized and HAFA approaches. For the same reason, average network distance is observed to remain the same at higher device counts as can be seen from Figure 104.

As seen in Figure 106, average network delay is small, approximately 45 milliseconds, as remote fog nodes selected are located nearby to the device to satisfy the application latency constraint.

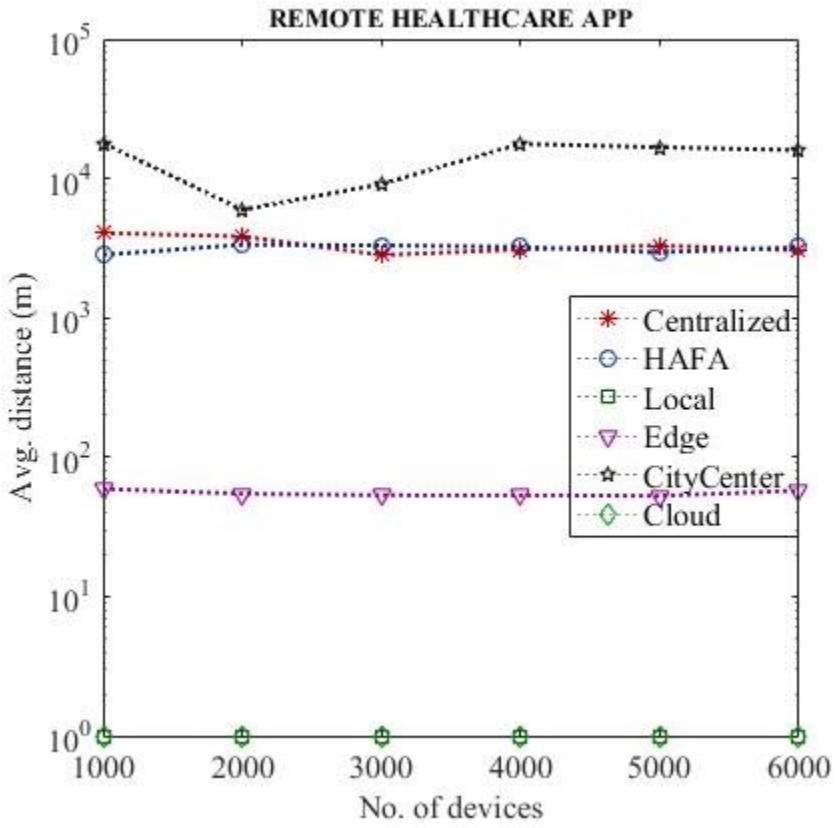

Figure 103: Remote Healthcare Application – Average distance between device (user) and executing (fog) node vs. number of active devices.

Edge nodes have small resource configurations. Hence, processing time is higher as compared to other approaches. As all layer-1 fog nodes are assumed to be of similar resource configurations, average execution time of tasks is same. At higher device counts, node queuing delay increased due to increase in workload.

As shown in Figure 110, average service time perceived per task is approximately 65 milliseconds and satisfies the RH application response time requirement of 100 milliseconds.

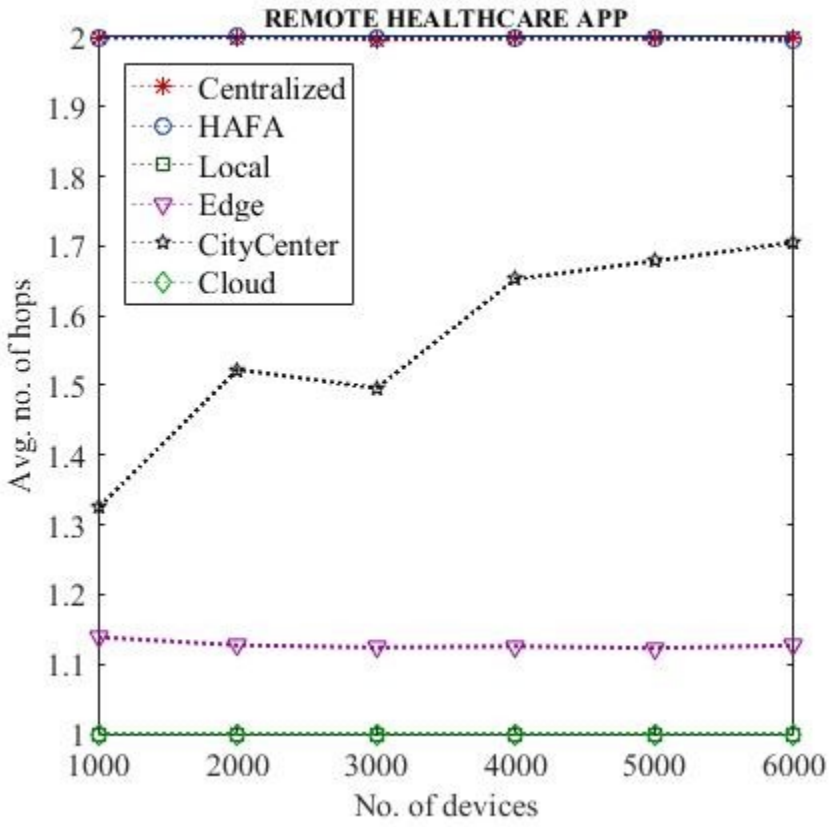

Figure 104: Remote Healthcare Application – Average hops between device (user) and executing (fog) node vs. number of active devices.

Figure 114 shows that only layer-1 fog nodes are utilized for task execution. The host utilization was about 70% for layer-1 nodes. Figure 111 shows that there is steady increase in utilization of compute resources as device counts increase. However the rate of increase is low due to the latency-critical nature of RH application.

Figure 116 shows low utilization of network resources as Edge placement approach selected local node for most service requests, and a small number of requests are submitted to

remote nodes. This also explains the slow increase in average network utilization at higher device counts as can be observed from Figure 115.

As Edge orchestrator considers all available layer-1 nodes for placement, the number of prospective nodes is 679 for all device counts as shown in Figure 117 and the average number of messages exchanged is 1358 as shown in Figure 118.

Local orchestrator.

As shown in Figure 100, Local orchestrator resulted in similar task failure percentage as that with Edge placement approach. This can be explained by the fact that, with Edge placement approach most of successful tasks were executed locally at layer-1 nodes.

Figure 101 shows that the Local placement approach executes tasks on fog nodes belonging to all layers for the test scenario with 6000 devices.

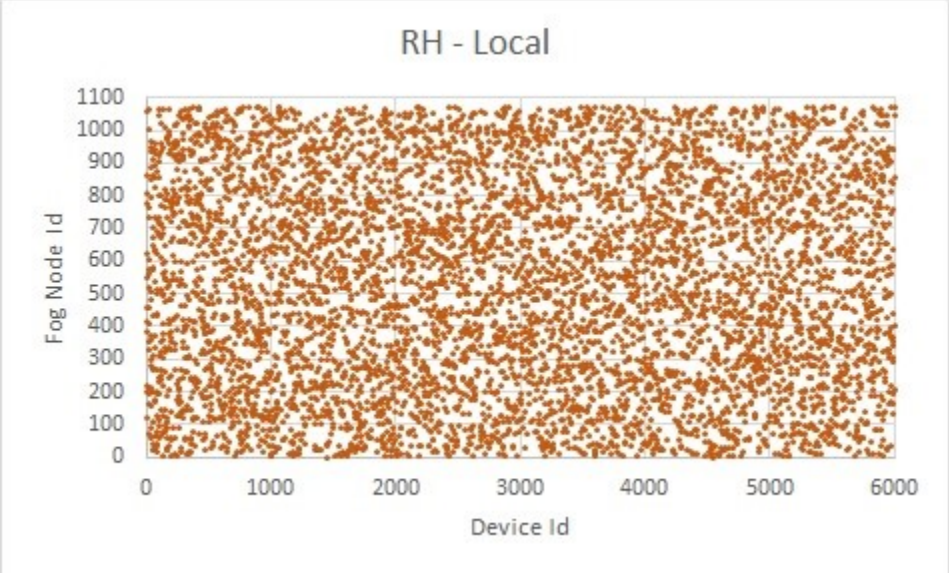

Figure 105: Remote Healthcare Application – Fog nodes selected by Local Orchestrator for service placement.

Figure 102 shows that the Local placement approach resulted in slightly lower average cost per task as compared to Edge placement approach as execution cost is highest for layer-1 fog nodes.

Figure 103 shows that the average great circle distance between device/user and executing fog node is always zero for all device counts as device and local fog node are co-located in our test environment.

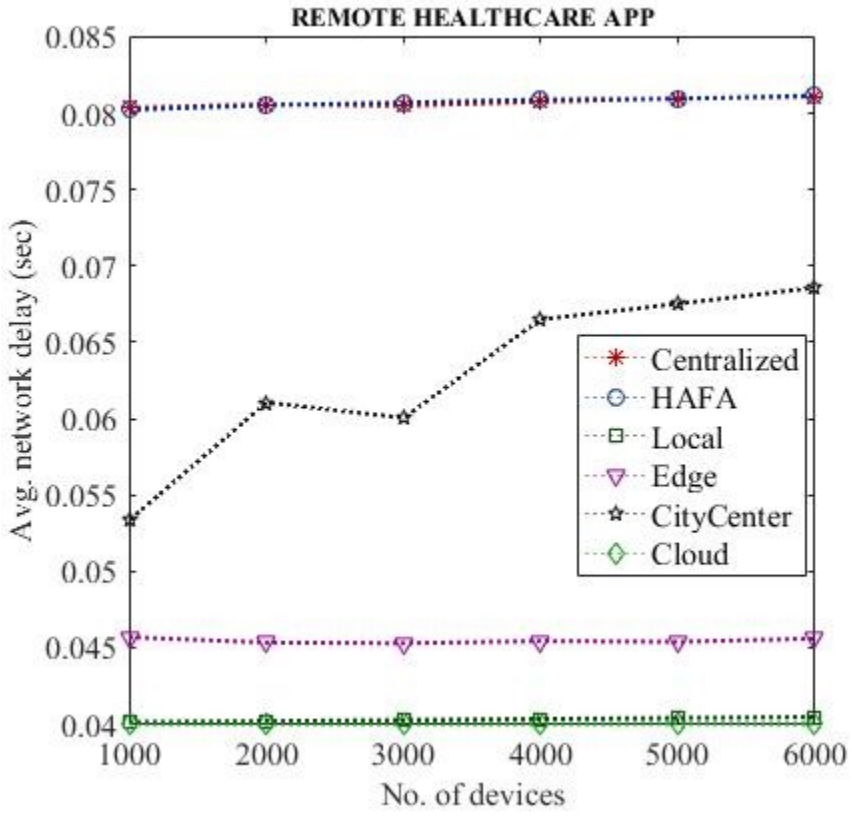

Figure 106: Remote Healthcare Application – Average network delay per task vs. number of active devices.

Figure 104 shows that the average network distance between device/user and executing fog node is always one for all device counts as the local node is always accessible at one hop from device.

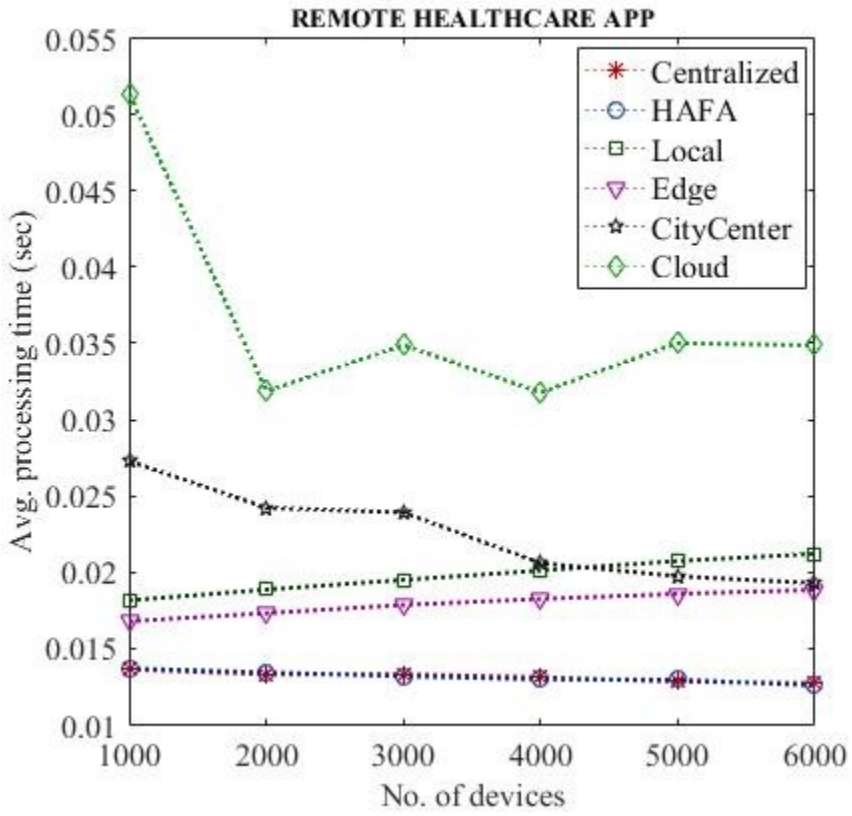

Figure 107: Remote Healthcare Application – Average processing time per task vs. number of active devices.

As seen in Figure 106, average network delay is the least of all approaches as it comprises only the congestion delay at first hop network node. Network propagation delay is always zero. Congestion delay increased slightly at higher device counts due to increase in local data traffic.

Figure 107 shows that the average processing time is approximately similar to that from Edge placement approach

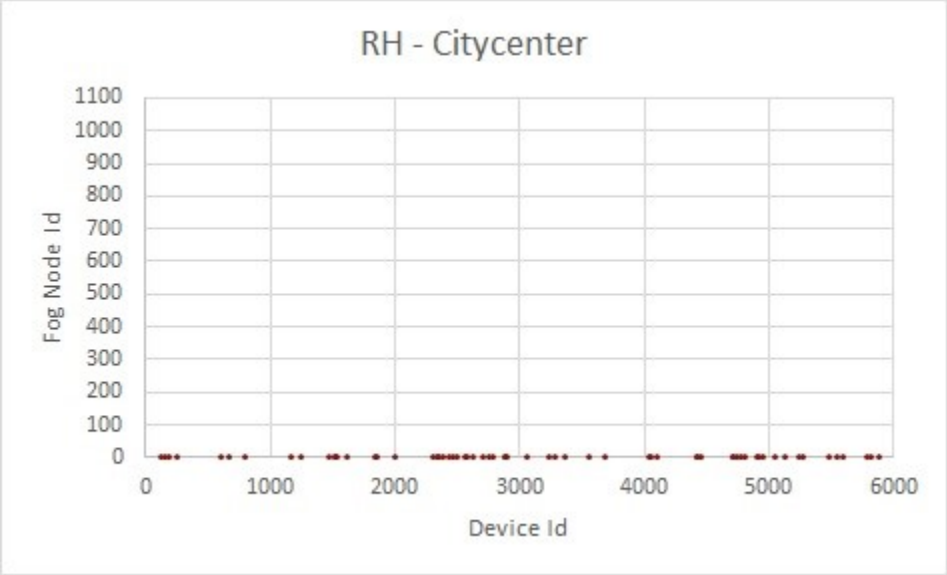

Figure 108: Remote Healthcare Application – Fog nodes selected by City center Orchestrator for service placement.

As shown in Figure 110, average service time perceived per task is seen as slowly increasing. However, it is less than the application latency constraint of 100 milliseconds. Processing and network latency have contributed to service time in approximately similar manner.

Figure 114 shows that fog nodes belonging to all layers are utilized for task execution. Figure 111 shows an increase in average host utilization percentage at higher device counts. This shows that the compute resources at nodes are not exhausted and the system can support higher device configurations.

Figure 116 shows low utilization of network resources with Local placement approach as only one network node is utilized by tasks for data transfer. Additionally, high percentage of

failed tasks resulted in marginal increase in utilization of fog networks at higher device counts, as can be observed from Figure 115.

As Local placement approach considers only one i.e. local fog node for placement, the number of prospective nodes is 1 for all device counts as shown in Figure 117 and the average number of messages exchanged is 2 as shown in Figure 118.

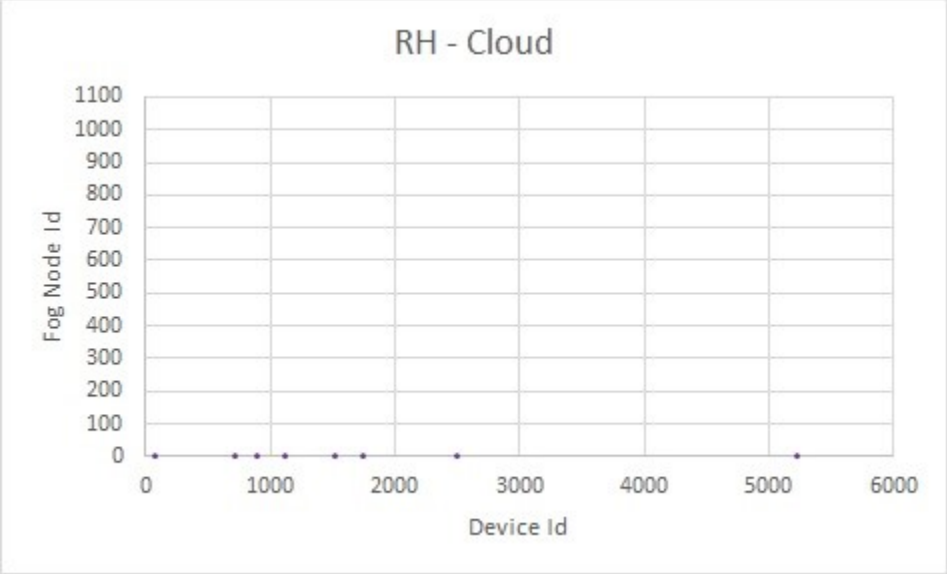

Figure 109: Remote Healthcare Application – Fog nodes selected by Cloud Orchestrator for service placement.

City center orchestrator and Cloud orchestrator.

As shown in Figure 100, the City center and Cloud placement approaches resulted in almost 100% high task failure percentage as average network path latency from device to city center fog node is high and did not satisfy the RH application latency requirements for a large number of devices. Figure 101 shows that a small number of tasks were executed on fog node belonging to layer-6 for the test scenario with 6000 devices.

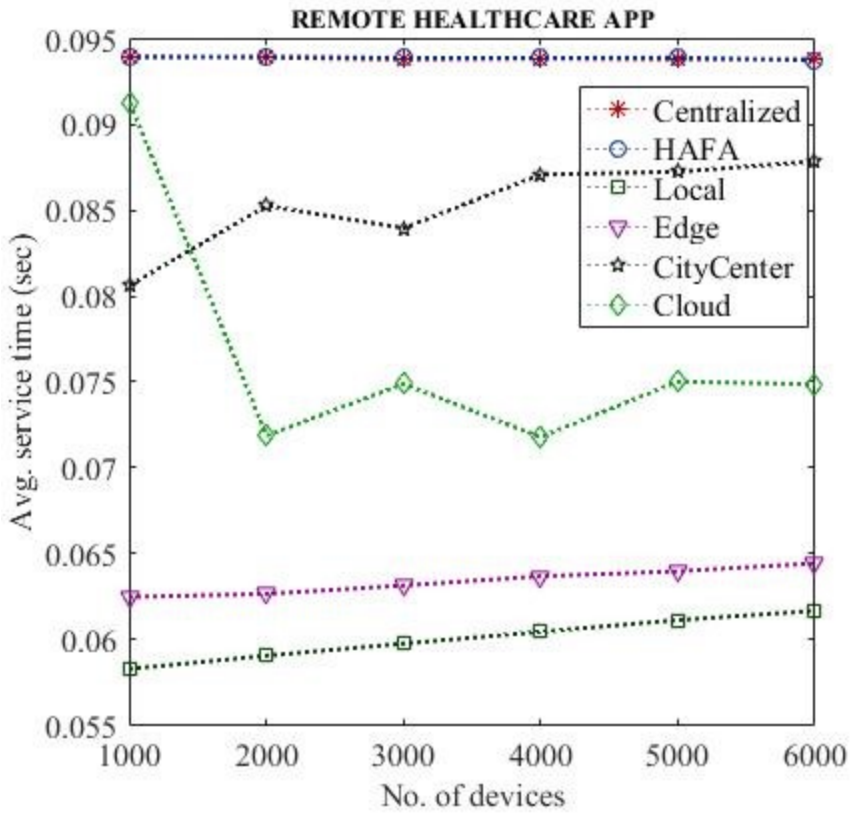

Figure 110: Remote Healthcare Application – Average service time per task vs. number of active devices.

Figure 102 shows that these approaches resulted in least average cost per task as compared to other approaches. However, these values are invalid and can be ignored considering the high task failure percentage. Accordingly, the following metrics are also invalid and need to be ignored - average great circle distance between device and fog node as shown in Figure 103, average network distance in terms of number of network hops as shown in Figure 104, average network delay as shown in Figure 106, average processing time as shown in Figure 107, average service time as shown in Figure 110, average host utilization percentage as shown in Figure 111 and Figure 114, average network utilization as shown in Figure 115 and Figure 116, average

number of prospective hosts considered as shown in Figure 117 and the average number of messages exchanged as shown in Figure 118.

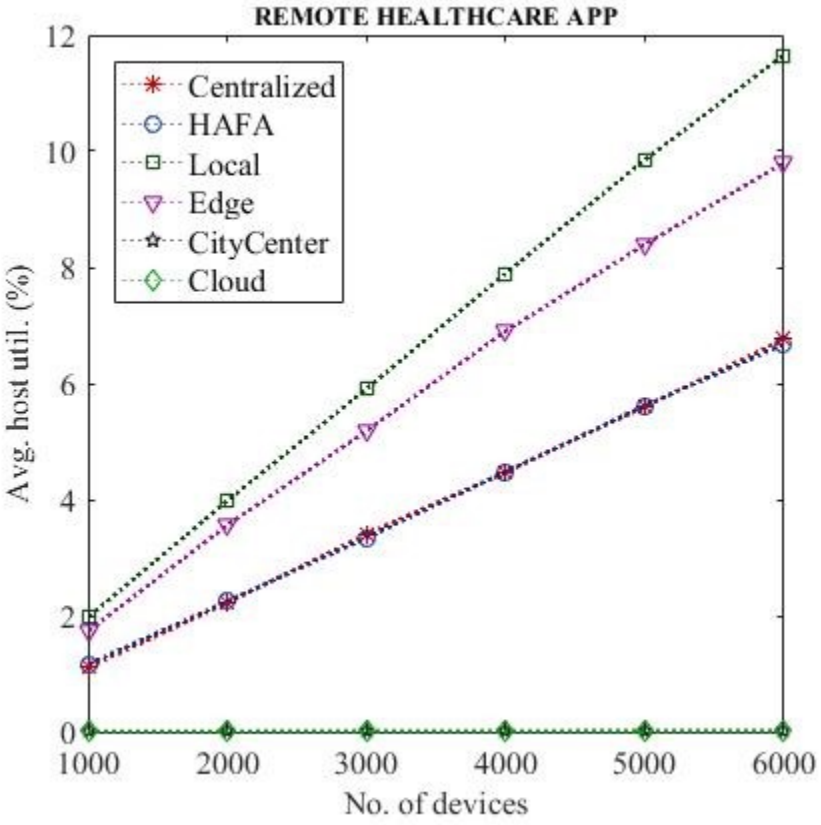

Figure 111: Remote Healthcare Application – Average host utilization vs. number of active devices.

Centralized orchestrator and HAFA orchestrator.

The test results analysis showed that the Centralized and HAFA placement approaches have demonstrated similar behavior during execution of Remote Healthcare tasks.

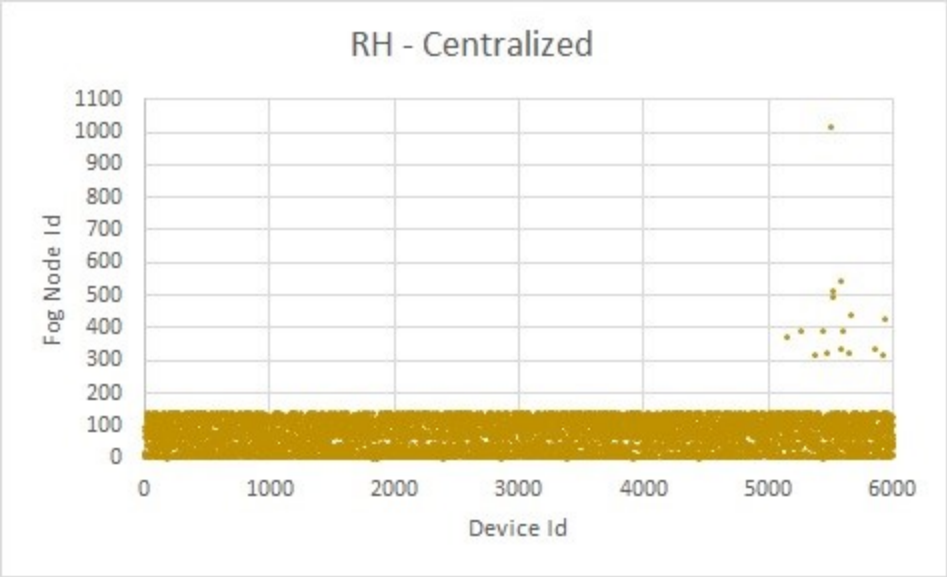

Figure 112: Remote Healthcare Application – Fog nodes selected by Centralized Orchestrator for service placement.

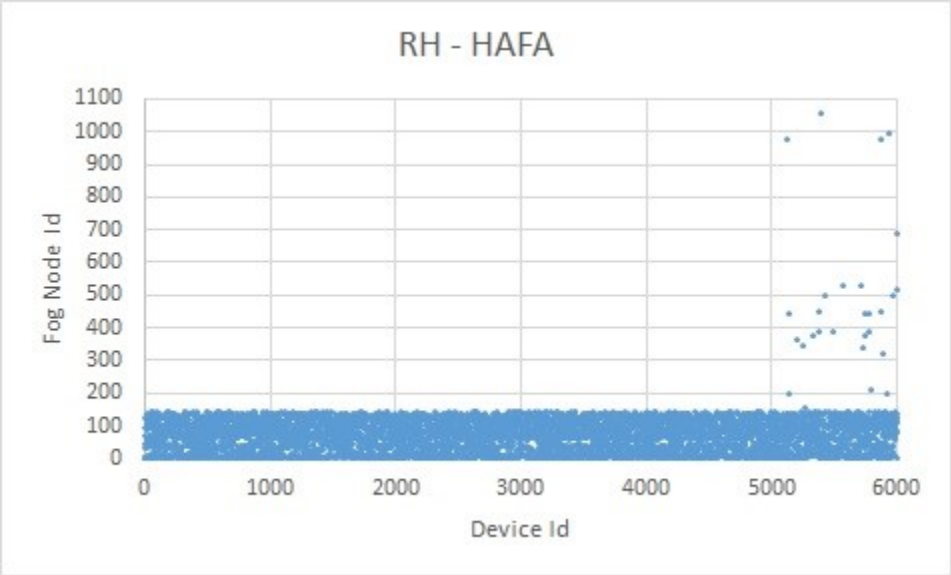

Figure 113: Remote Healthcare Application – Fog nodes selected by HAFA Orchestrator for service placement.

As shown in Figure 100, the Centralized and HAFA placement approaches resulted in higher percentage of task failures for RH application as compared to Edge and Local placement

approaches. It is observed that the tasks failed primarily due to high network delay at higher layer fog nodes for test scenarios with low device counts. At higher device counts, as the resources on immediate higher layer nodes are exhausted and the next layer nodes are not accessible within acceptable response time constraint, lower layer nodes with higher execution cost are preferred, which were mostly successful for the tested scenarios with device counts up to 6000. This resulted in slight reduction of failed task percentage. We cannot claim that the reduction trend will continue for further higher device counts.

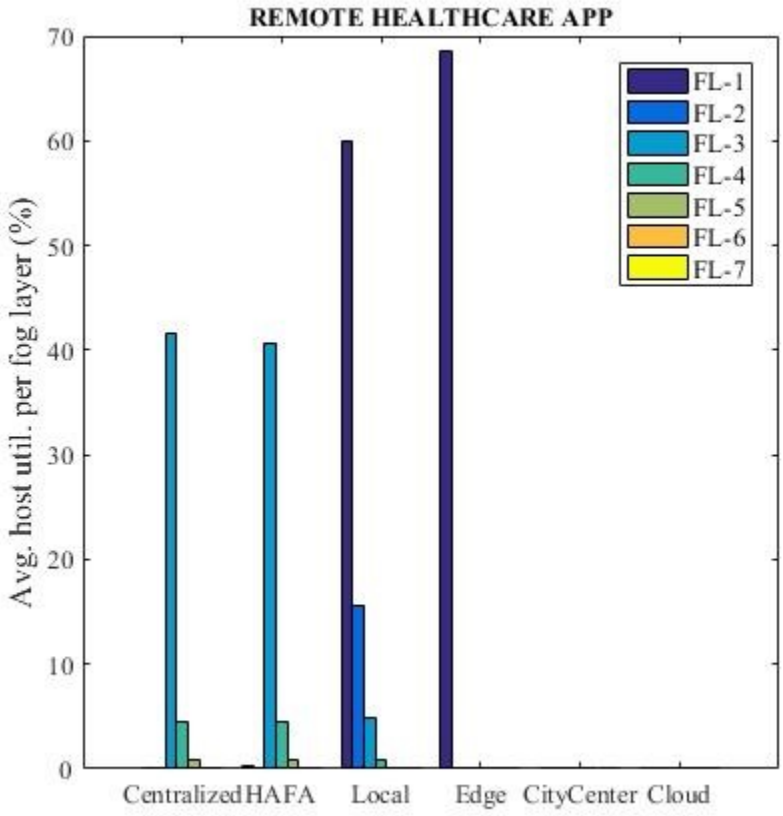

Figure 114: Remote Healthcare Application – Average host utilization per fog layer with 6000 active devices.

Figure 101 shows that the successful tasks were executed on nodes belonging to all fog layers for the test scenario with 6000 devices.

Figure 102 shows that the Centralized and HAFA placement approaches resulted in least average cost per task as compared to Edge and Local placement approaches, as these approaches select the fog node with lower total cost for computation and communication, among the prospective nodes.

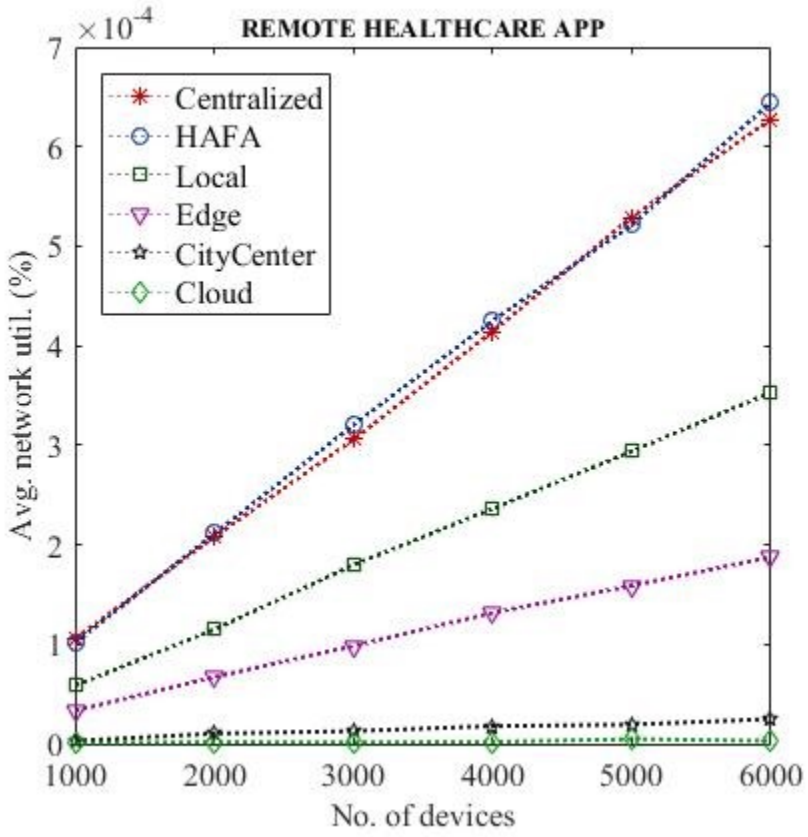

Figure 115: Remote Healthcare Application – Average network utilization vs. number of active devices.

Figure 103 and Figure 104 show that there is no change in average great circle distance between device/user and executing fog node at higher device counts. As the compute and

network resource requirements of RH application are small, the tasks were always submitted to the next higher level fog node accessible at one additional network hop from device connected network node. At higher device counts, as fog nodes accessible at network distance more than two do not satisfy the latency requirement, the tasks are submitted to local node at higher cost. Thus the average number of network hops per task is observed to be approximately 2.

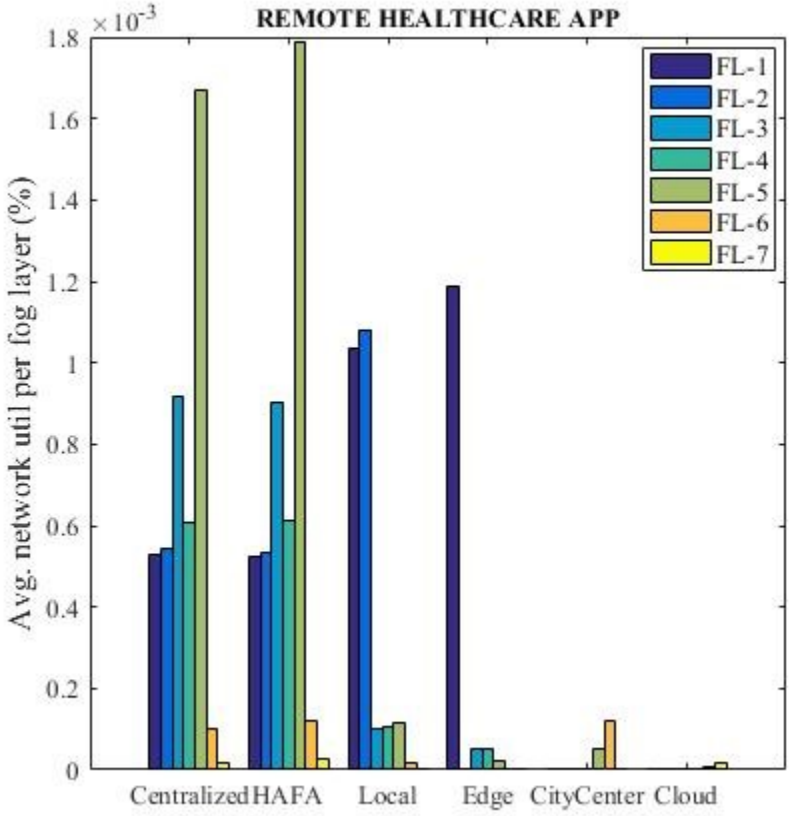

Figure 116: Remote Healthcare Application – Average network utilization per fog layer with 6000 active devices.

Centralized placement approach attempts to reduce communication cost by reducing number of network hops, which indirectly reduces network delay as well. As the average number of hops didn't change at higher device counts, there is no change observed with average network

delay as can be seen from Figure 106. It is observed to be lower for Cloud and City center placement approaches. However these values are invalid and can be ignored considering the high task failure percentage with these approaches.

It is observed that a large percentage of the successful tasks were executed at next higher layer fog node, rather than at local node. Thus, the average processing time is approximately the same at all device count as can be observed in Figure 107.

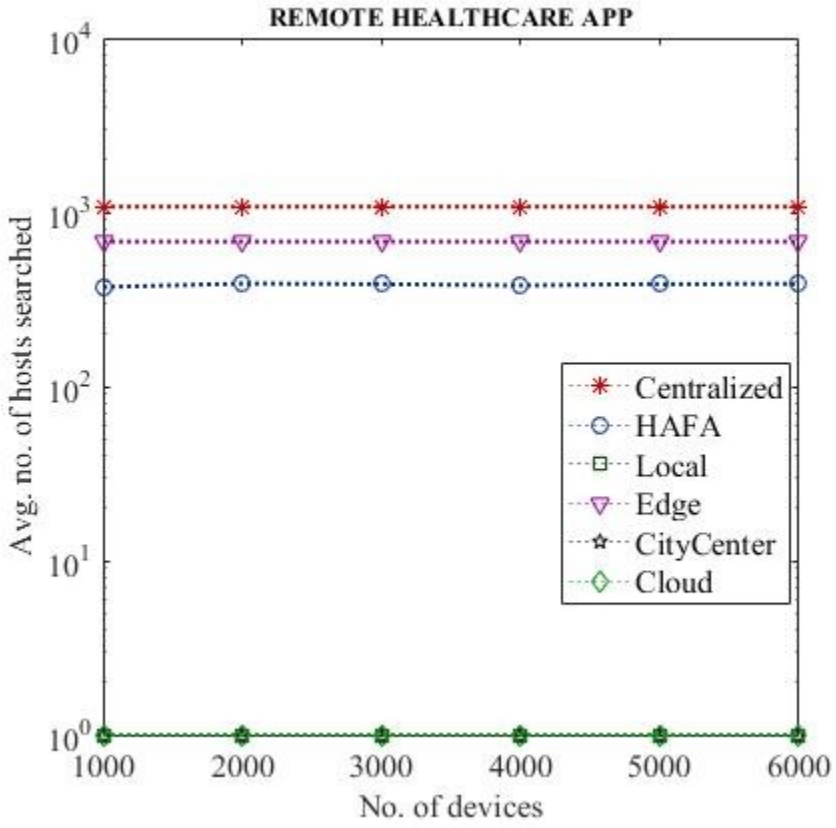

Figure 117: Remote Healthcare Application – Average number of hosts searched vs. number of active devices.

As shown in Figure 110, average service time perceived per task is dominated primarily by average network delay for all device counts, and is observed to be less than the application latency requirement of 100 milliseconds.

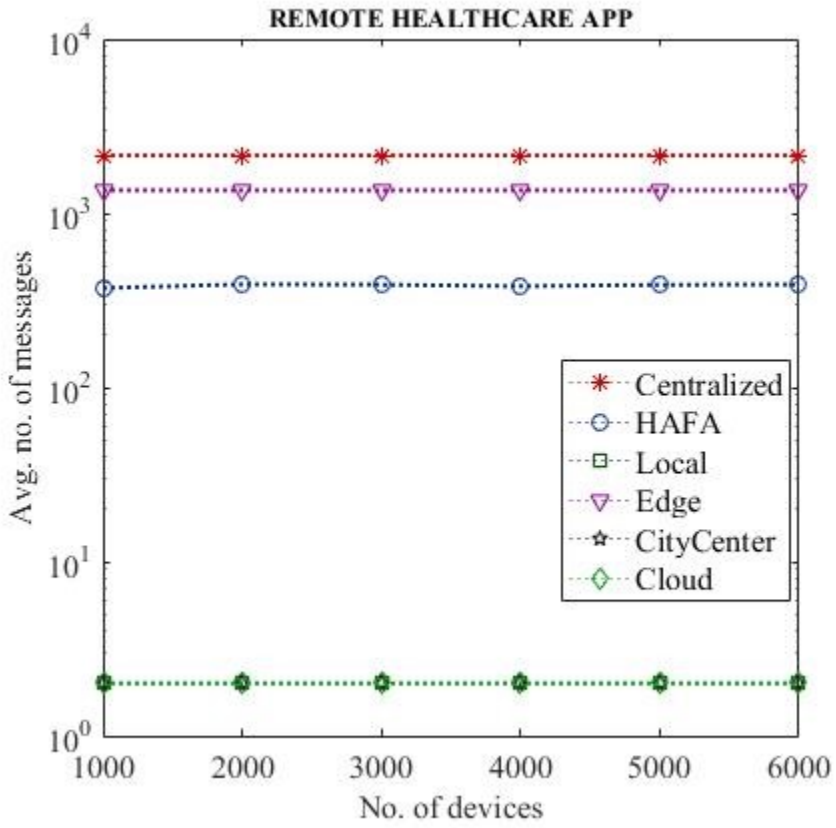

Figure 118: Remote Healthcare Application – Average number of messages exchanged vs. number of active devices.

Figure 114 shows the percentage of resources utilized at each fog layer for task execution. It is observed that fog nodes from all layers were utilized. Average utilization percentage has increased linearly as shown in Figure 111 at a rate lower than that with edge and local placement approaches. The rate of increase is smaller due to the reason that Centralized and HAFA placement approaches leverage higher layer nodes which have larger resource capacities

in our test environment. Similar pattern is observed for average network utilization in Figure 115 and Figure 116, which can be explained with same reasoning.

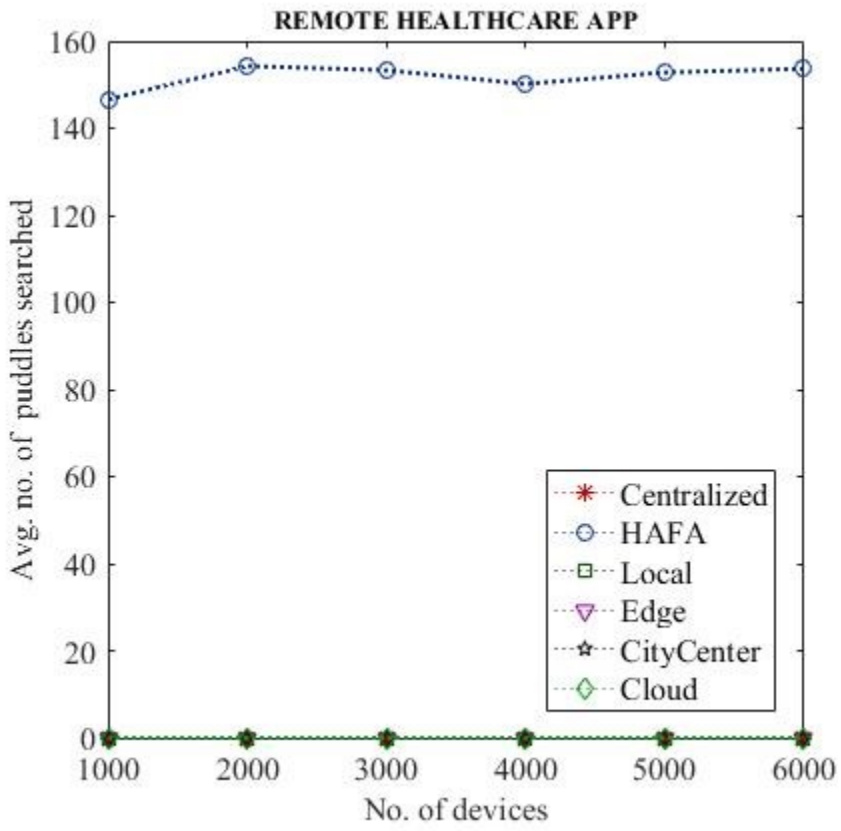

Figure 119: Remote Healthcare Application – Average number of Puddles searched vs. number of active devices.

As Centralized placement approach considers all available nodes in system for placement, the number of prospective nodes is 1074 for all device counts as shown in Figure 117 and the average number of messages exchanged towards making the placement decision is 2148 as shown in Figure 118.

The average number of prospective nodes with HAFA placement approach for RH application profile is observed to be 371 nodes for test scenario with 1000 devices which

increased with higher device counts to 387 nodes for test scenario with 6000 devices. Similarly, the number of messages exchanged have ranged from 372 for 1000 devices to 389 for 6000 devices, and the number of Puddles that comprised the search space ranged from 146 for 1000 devices to 153 for 6000 devices. These patterns can be seen in Figure 117 and Figure 118.

7.12. HAFA: Results analysis

Figure 120, Figure 121, and Figure 122 show the pictorial representation of number of hosts considered, number of messages exchanged, and number Puddles searched to identify a cost-efficient fog node for each service request received from device/user for Augmented Reality application.

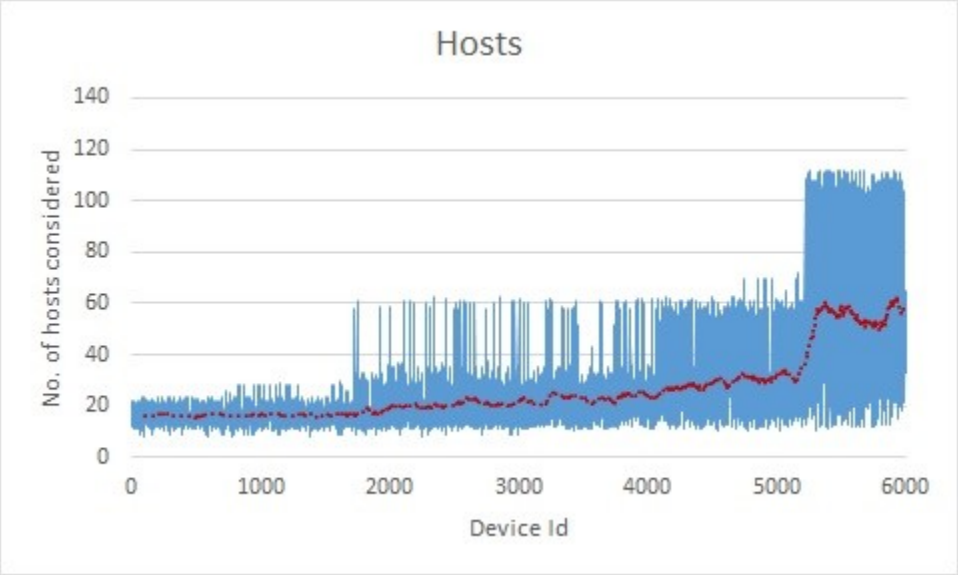

Figure 120: HAFA Analysis: Number of hosts considered per service request.

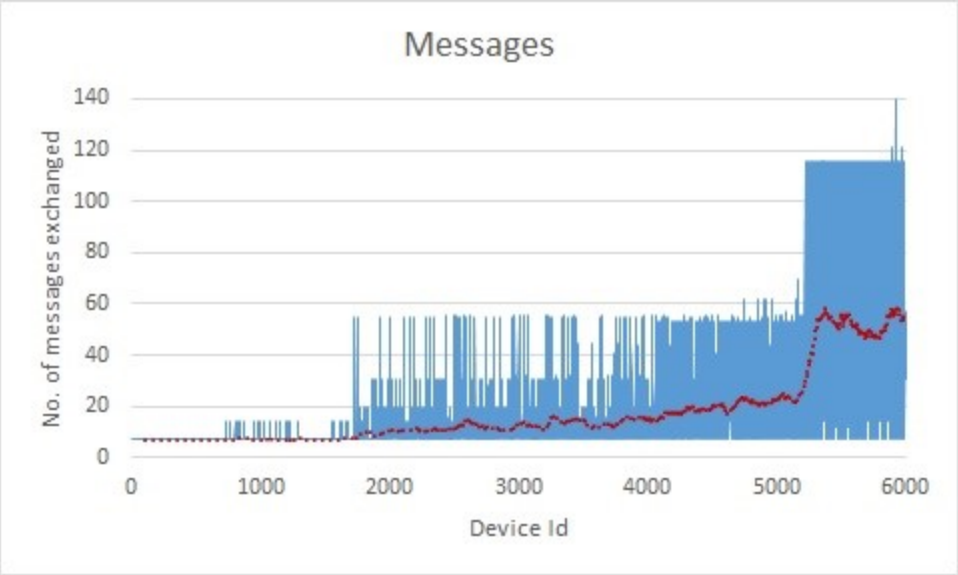

Figure 121: HAFA Analysis: Number of messages exchanged per service request.

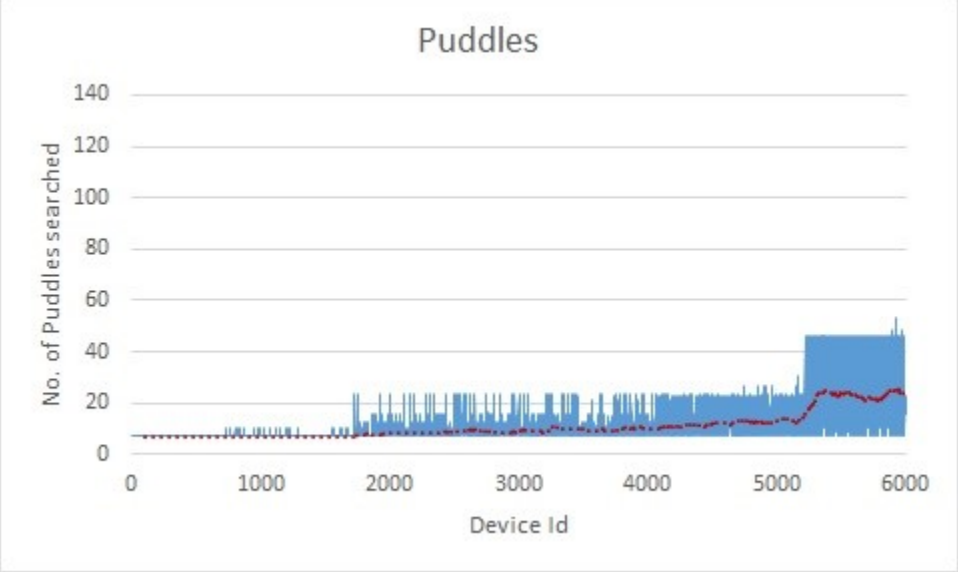

Figure 122: HAFA Analysis: Number of Puddles searched per service request.

7.13. Limitations

Listed below are some of the limitations of the HAFA service placement approach.

- Hosting an application service on all fog nodes belonging to a specific fog layer, costs the same, irrespective of the location of fog node.

- Fog nodes belonging to each fog layer are uniformly dispersed across the geography.
- Devices and fog nodes are physically stationary, i.e. there is no change in their location during the course of service request and its execution.
- In addition to GPS coordinates, elevation of devices has not been considered in determination of location.
- Fairness and load balancing of workload on various fog nodes of the system are not considered.

7.14. Conclusion

In this paper, we proposed and implemented a fully distributed approach to identify a cost-efficient fog node to host an application service with given resource requirements and QoS characteristics. We have implemented it in a simulated test environment, and demonstrated its efficiency for various application profiles. We observed that the approach closely approximates the centralized near-optimal approach.

7.15. Future work

The proposed solution approach is limited to physically stationary fog nodes and devices which generate data and access the application service. In future, we intend to develop solutions for mobile devices and fog nodes.

Energy consumption is a determining factor in deployment of large infrastructure environments. Hence, we plan to verify the applicability of proposed approach in terms of energy efficiency as well.

8. Network-Aware Service Pricing Approach for Fog Infrastructure as a Service

8.1. Abstract

Fog Infrastructure as a Service (FIaaS) service model facilitates leasing of shared infrastructure resources such as compute, network, and storage resources on fog nodes deployed by Fog Service Providers. Leased resources are leveraged by Fog Tenants to deploy and manage application environments individually. Fog Tenants are charged by Fog Service Providers based on the amount of resources consumed and as per agreed SLAs. Fog computing paradigm differs from Cloud computing paradigm in several factors such as large-scale deployment, heterogeneity, very large number of nodes spread over vast areas, location-awareness, support for mobility and real-time interaction, etc. To cater the unique needs of fog environment, we have developed a novel network-aware approach for service pricing in FIaaS environments. We have shown by experiments using a simulated test environment that our proposed approach fairly charges the tenants based only on their applications' infrastructure resource consumption.

8.2. Introduction

Proliferation of low-cost sensing devices, along with the need of humanity to store and process ever-growing amounts of data, is pushing the cloud IT infrastructure to its limits. Researchers have shown that future cloud infrastructure deployments cannot keep pace with the users' needs.

Resource demands from new generation applications such as Internet of Things (IoT), 5G, etc. are difficult to serve using resources from large cloud data centers alone considering the latency-criticality nature of IoT applications such as healthcare, emergency, etc. as well as 6Vs characteristics of Big Data generated by such applications namely, Volume, Velocity, Veracity, Variety, Variability, and Value. To cater the needs of such new generation applications, Cisco proposed Fog Computing paradigm [13] as an extension of cloud towards edge of the network. Fog computing is gaining prominence as it is embraced by industry and academia alike to facilitate realization of various new generation application environments such as 5G, IoT, etc.

Several researchers [84] have discussed deployment of large scale fog environments. Contrary to cloud, which comprises thousands of servers with high resource capacities co-located in large data centers, fog [16] [113] includes heterogeneous nodes with varying resource capacities, deployed individually, and distributed over large geographical areas to serve the local workload, and is thereby, location-aware.

Fog, being an extension to cloud, provisions resources and applications in the form of services to its tenants in an on-demand and pay-as-you-go manner as per agreed SLAs. Fog computing paradigm supports three service models–Fog Infrastructure as a Service (FIaaS), Fog Platform as a Service (FPaaS), and Fog Application as a Service (FAaaS). FIaaS service model [84] offers shared fog infrastructure resources such as compute, storage, and network to Fog Tenants (FT). Fog Service Providers (FSP) deploy and manage the physical infrastructure and offer the resources in the form of virtual machines, or containers. Fog tenants leverage the leased fog infrastructure resources to deploy and manage their application environments and are charged based on the amount of resources consumed as well as service standards.

8.3. Motivation

Cloud IaaS service model is offered by several commercial organizations [114] such as Google, Amazon, Microsoft, Rackspace, etc. Cloud tenants are offered infrastructure resources and are charged based on leased or consumed resources depending on resource type and agreed SLA. Google Cloud pricing policy [109] categorizes any point on earth as belonging to one of the seven geolocations and charges the tenant depending on the geolocation from which the request for data has originated from. Tenants are required to pay only for the egress network bandwidth consumed while ingress network bandwidth consumption is free.

Fog environments are deployed primarily for IoT applications of three types–latency-critical, bandwidth-intensive, and localized applications i.e. those having only local value. Cloud network service pricing model as discussed earlier cannot be applied as-is to the fog environments for several reasons as listed below.

Data generated by IoT environments is Big Data by nature and is characterized by 6Vs. Thus, IoT applications hosted on fog nodes receive large volume of data for processing and generate output which is only a fraction of original data quantity. As the raw data in original form loses its value in short duration and is rarely accessed in original form, if ever, FSP cannot earn enough from the data storage pricing as only the processed results will be stored, and raw data will be discarded immediately. Emerging online analytics approaches towards processing of streaming application data remove the need for storage of raw data on fog nodes, even for a brief period. Cloud pricing policy which charges the tenants per GB delivered i.e. for egress bandwidth consumption only and not the ingress bandwidth, may result in losses for FSP who deploys the network infrastructure to support both ingress and egress network traffic.

Location-aware characteristic of fog environment facilitates the deployment of latency-critical and localized application services on fog node physically closer to user. To ensure this, fog orchestrator which manages the application service instances need to be aware of geolocations and topology of network devices in fog environment. In addition, fog supports multi-hop connectivity for devices not directly reachable over network e.g. smart vehicles on road. Cisco recommends network devices to be fog nodes themselves. Recent research directions towards Fog-SDN etc. emphasize that compute nodes and network nodes may not be physically separate in fog environments. Thus, deployment of fog by FSP includes network devices, along with compute, and storage nodes and their individual resource utilization need to be considered for efficient placement of applications. Thus, fog network service pricing approach necessitates a more granular approach towards considering the location of various entities in system owing to the distributed deployment of individual fog nodes over vast geographical areas.

Under ideal conditions, centralized fog orchestrator with complete knowledge of entire fog environment should be able to differentiate between a fog node which is closer to user and one that is farther away, towards cost-efficient hosting of localized applications. For example, if a shopper at a mall needs information about one of the local stores, it is more cost-efficient for FSP to store the corresponding application and data at local fog node in the mall instead of a fog node which is 100 miles away, as average network utilization is reduced. This is unlike cloud environments where consumer network access is provided by telecom and network service providers such as AT&T, etc. and hence cloud considers only host-based bandwidth consumption towards service pricing, being completely oblivious to the physical topology of network outside the cloud data center.

To ensure fairness to fog tenants, service pricing in fog environment should consider their individual network resource utilization, for instance, localized application which is hosted on fog node only one hop away from user i.e. accessing only one network device should be charged lesser as compared to the access to an application which is hosted on a fog node three hops away i.e. resources on three network devices are utilized. As the physical network infrastructure is deployed and maintained by FSP, it is important to meter the resource consumption by individual tenants to ensure fairness in pricing to fog tenants as well as ensure that fog service provider is paid for offering the resources.

8.4. Contributions

In this paper, we have proposed a novel network-aware fog service pricing approach which considers features unique to fog to ensure fairness in pricing for both fog service provider and fog tenant as well as among fog tenants themselves.

8.5. Network-aware service pricing for FIaaS

In earlier section, we have discussed the insufficiency of host-based service pricing approach for FIaaS environments spread over vast geographical areas. To satisfy the specific needs of FIaaS environment, we propose network-aware service pricing approach to assign costs for fog infrastructure resources leased by fog tenants, the details of which are provided here.

Service pricing in fog comprises three components: compute costs resulting from leasing compute resources to execute a given application service, data storage costs resulting from storage of application instance and its corresponding data for a specific duration, as well as network data transfer costs depending on amount of data transferred.

Owing to the heterogeneity characteristic of fog environment, it may comprise compute, storage, and network nodes of varying capacities. Fog compute service pricing depends on

characteristics of compute node on which the application is hosted such as type, deployment cost, maintenance cost, lifetime, average utilization, etc. Similarly, fog data storage service pricing depends on several factors such as cost of storage node, its utilization, resource type based on reliability and speed characteristics, etc. In addition, data storage costs on fog nodes depend on type of data itself i.e. if it is compressible and if deduplication can be enabled for the data element towards efficient utilization of storage node resources, as well as any additional storage services requested such as holding multiple copies on same / different nodes for reliability / performance purposes. The factors contributing to fog IaaS compute and storage service pricing as mentioned above are similar to those considered for cloud IaaS as well and hence the corresponding pricing strategies can be applied. Compute and storage resource pricing in fog is impacted by availability and reliability of fog node which may vary for individual fog nodes depending on resource nature of the fog node itself as well as number of available fog nodes in a given area at a given time instant as well as their reliability and availability characteristics. This is different from a cloud environment where co-located nature of cloud nodes can be taken advantage of, to offer required reliability and availability levels as per predefined SLAs irrespective of individual cloud node resource characteristics.

Note that compute and storage resource pricing does not depend on geographic location of the physical fog node. Thus, the orchestrator has no preference between two fog nodes having similar resource type and utilization characteristics but are located at varying network distance to user. As obvious, with widely dispersed fog environments having large number of fog nodes, the one nearer to data producer and/or consumer is preferred. To be specific, the fog node with lower network data transfer cost is preferred.

Contrary to storage and compute service pricing, network cost incurred by transfer of data corresponding to a given application service cannot be obtained solely from the fog node which hosts the service. Rather, it depends on the individual fog network nodes through which the application data elements have passed and their pricing. As obvious, this information cannot be made available by hosting node itself. Rather the orchestrator should have knowledge of network elements in fog environment as well as the network route followed by data elements along with the cost of data transfer at each of the corresponding network elements en-route. Fog, being a heterogeneous environment and hierarchical in nature, may comprise different types of network elements with their resources offered to tenants at different costs.

To facilitate deployment of localized applications, i.e. those of only local value, on fog nodes in close vicinity to users, resources on lower layer network components can be offered at lower prices as compared to those at higher layers in fog hierarchy [84]. Thus, fog orchestrator search for a cost-efficient node to host and execute a given application service results in identification of a fog node with smaller network distance i.e. a local fog node is selected where available. This approach results in lower network data transfer costs for bandwidth-intensive application as fog node accessible at lower network distances and lesser number of hops is preferred for its hosting and execution. For latency-critical applications, orchestrator tries to find a fog node which satisfies the strict latency requirements of application. Our proposed pricing approach helps in finding a cost-efficient node among those which satisfy the latency needs of application.

Network-aware service pricing approach for FIaaS service model is summarized as follows:

Application service cost = Application hosting (storage) cost + Application execution (compute) cost + Application data transfer (network) cost

Application compute / storage service costs = host FN costs * constant depending on reliability and availability nature of local fog environment as well as that of fog node itself.

Application network data transfer service cost = Ingress network bandwidth consumption cost at host fog node + egress network bandwidth consumption cost at host fog node + data transfer costs on each fog network node en route data source to destination node (depends on resource nature of fog network node) + data transfer costs on each link connecting fog network nodes en route data source to destination node (depends on network link characteristics and link distance). Resources on lower layer fog network nodes are priced at lower rates as compared to those belonging to higher fog layers.

8.6. Qualitative analysis

8.6.1. Benefits of proposed approach

The proposed network-aware service pricing approach is applicable for all three types of applications recommended to leverage fog infrastructure namely, latency-critical, bandwidth-intensive, and localized applications, and results in fair pricing acceptable to both service providers and tenants as well as help find a cost-efficient node. It considers unique nature of fog environment such as heterogeneity, hierarchy, reliability, availability, dispersement over vast areas, large scale deployment, etc., and facilitates the following:

- Fair pricing to individual fog tenants based only on their resource consumption.
- Improved earnings to fog service provider as both ingress and egress network bandwidth consumption is charged.

- Transforms network infrastructure into revenue generating components of system from cost components; thus supporting the business model of FIaaS.
- Helps localize the placement of applications and data as fog nodes reachable at higher network distance i.e. at higher number of hops increase the overall cost of fog network service.
- Ensures transparency in system and holds the principle of cloud that tenants are charged only for the services consumed i.e. they are charged only for data transferred and only on the network components which were utilized towards the transfer of data belonging to their application services.
- Helps fog service provider identify a better cost-efficient node towards hosting and execution of a service in the entire system i.e. on a fog node located nearby and the one accessible over smaller network distance as well as lesser number of hops.
- Helps transfer to tenant, the savings from hosting and executing the application on a cost-efficient node i.e. application services are offered to tenant in a cheaper manner as compared to execution on fog nodes using host-based pricing model which considers only node type but not network path and network nodes' costs.

8.7. Experimental analysis

To demonstrate the applicability of our proposed network-aware service pricing approach, we have shown its quantitative benefits as compared to Google Cloud IaaS network pricing policy [115], referred as host-based approach in this paper and discussed in earlier section.

8.7.1. Test environment

We have executed tests in a simulated distributed environment, referred as PFogSim. PFogSim is an event-driven simulator developed in Java to simulate the execution of applications in a distributed large-scale fog environment with thousands of fog nodes, data generating and consuming devices, as well as users. It is extended from EdgeCloudSim [109] to reflect the unique features of fog environments such as heterogeneity, location-awareness, mobility of devices, users, as well as fog nodes themselves. PFogSim simulator can be used to test various orchestrators for application service placement in fog environments, fog service pricing approaches, etc. The simulator facilitates analysis of test results by auto-generation of graphs from test logs comparing selected approaches and metrics.

We have used Chicago city data set [110] to simulate a fog environment with 1073 fog nodes dispersed over 237 sq. miles area of Chicago, which includes fog nodes located one each at 679 schools, 253 connect centers, 80 libraries, 50 city wards offices, 10 universities, 1 city hall, and 1 cloud data center located at approx. 500 miles from city hall. Fog nodes placed at different locations of similar premises are of similar resource configuration, e.g. fog nodes placed at all city wards are of similar capacity. Each of these locations hosts a network router to facilitate device and user connectivity, as well as connect other network components. Fog node at each school, connect center, and library was connected to an individual edge router, fog node at each city ward office and each university is connected to an individual border router, which also connects to all the nearest edge routers, and fog node at city hall and cloud data center are connected to an individual core router. Each border router is connected to two nearest border routers as well as the core router at city hall. Core routers at city hall and cloud data center are connected over direct link. Each network device is also considered a Wireless Access Point (WAP) so that devices and users can connect over wireless network and hence support their

mobility. Fog network is simulated in the above manner to closely resemble a prospective smart city fog infrastructure deployment.

8.7.2. Test approach

We have discussed in earlier sections that compute and storage service pricing approach in FIaaS environments is similar to that with Cloud IaaS. As the location of devices, fog nodes, users, along with their network connectivity a.k.a. network topology is the major defining factor in deployment of applications in fog environment, we have focused on the tests which showcase the importance of network-awareness and its necessity in cost-efficient deployment and utilization of fog environment. We have executed tests to show that the network-aware service pricing approach ensures correct pricing of services based on amount of resources consumed and the corresponding pricing model is appropriate for both fog service providers and tenants.

To show the impact of network-aware service pricing approach, we have compared the overall service cost using it with that from host-based approach described in earlier section. Based on Cisco monthly lease pricing for 1G, 10G, and 100G routers [116], data transfer costs for various fog entities are assigned as listed in Table 11. We assume that data transfer cost on fog node is same as that on an edge router. This supports the scenario when a fog node dynamically takes the role of router to offer network connectivity to a different fog node by multi-hop access. We assume that data transfer cost for both input and output data is the same on any fog entity. A small number of co-located devices are connected to the same WAP and application services are deployed and executed on a selected fog node. Three application types were defined with data transfer profiles as shown in Table 12. We assume that all entities in system are physically stationary and that user is co-located with corresponding device, e.g. a surveillance camera and alert display located inside a building. Thus, device generated data and

processed results from application follow the same network path. During test execution, we have captured several metrics such as number of network hops, physical distance between the devices/users and fog nodes, overall cost of application execution, etc.

Table 11: Data transfer cost per Mb on various entities in simulated fog environment.

Fog Entity	**Data Transfer Cost ($/Mb)**
Fog Node	0.00032763749
Edge Router	0.00032763749
Border Router	0.00131054996
Core Router	0.00524219984

Table 12: Application data transfer characteristics (per task).

Application Type	**Data Upload (KB)**	**Data Download (KB)**	**Application Nature**
Augmented Reality	1500	25	Upload-Intensive
Infotainment	25	1500	Download-Intensive
Healthcare	200	25	Latency-critical

8.7.3. Test execution

We have executed tests for four scenarios as listed below, varying the network distance between devices and fog node.

- Scenario-1 (S1): Device ←→ Edge Router ←→ Fog Node (School)
- Scenario-2 (S2): Device ←→ Edge Router ←→ Border Router ←→ Fog Node (University)

- Scenario-3 (S3): Device ←→ Edge Router ←→ Border Router ←→ Core Router ←→ Fog Node (City Hall)
- Scenario-4 (S4): Device ←→ Edge Router ←→ Border Router ←→ Core Router ←→ Core Router ←→ Fog Node (Cloud)

Each test is executed for a simulation time of 30 minutes. A total of 1243 tasks were executed during this period. All tasks generated during a test run were of same application type and thus have similar resource profiles.

8.7.4. Results analysis

For three given application types, Figure 123 provides comparison of the amount in dollars that fog tenant is charged based on egress-only network bandwidth consumption on fog node, and egress + ingress network bandwidth consumption. Though Infotainment and Augmented Reality applications have similar profile for total data transfer, egress-only pricing policy is unable to charge the tenant in an appropriate manner whereas egress + ingress pricing policy has charged tenant in similar manner for both applications. This shows that cloud IaaS pricing policy of charging tenants only for egress network bandwidth is insufficient for fog-based applications which are bandwidth-intensive.

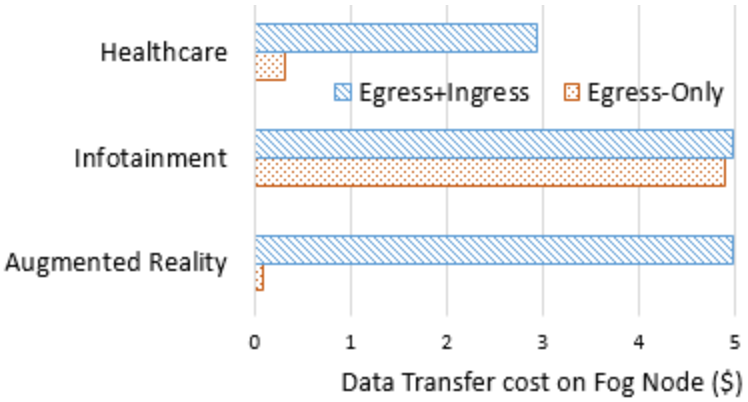

Figure 123: Comparison of data transfer costs incurred on fog node with egress+ingress and egress-only approaches.

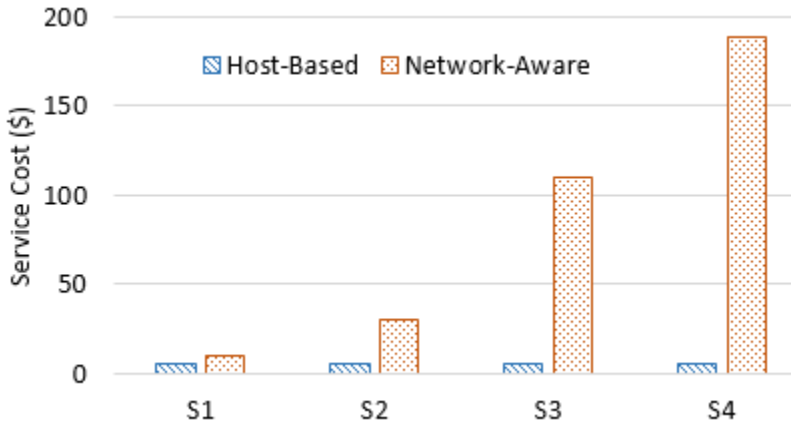

Figure 124: Comparison of service cost using Host-based and Network-aware pricing approaches.

Tests were conducted using Augmented Reality application profile for four scenarios with devices connected to fog nodes at varying network distances to observe the impact of network-aware pricing approach. Figure 124 shows the amount in dollars charged to tenant using host-based pricing approach and network-aware pricing approach. From the graph, we observed that network-aware pricing approach charges the tenant correctly based on the utilization of network resources in fog environment, whereas host-based approach charges the tenant the same

amount irrespective of the number of network components and their resources being utilized in the system, which may result in losses to FSP who is responsible to host the fog devices.

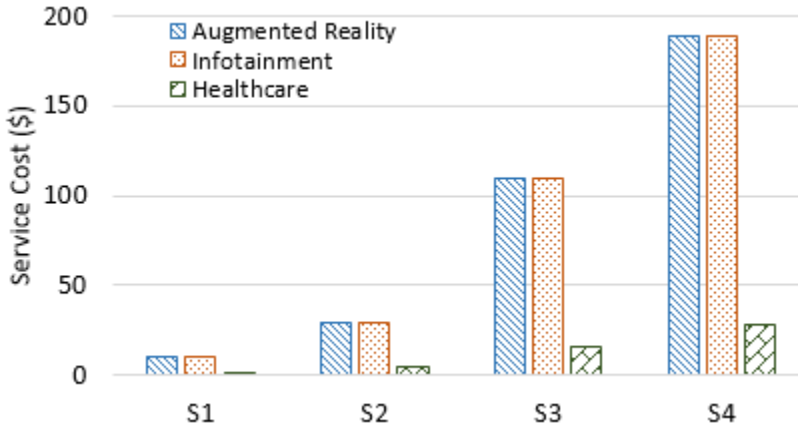

Figure 125: Comparison of service cost using Network-aware approach for various applications and network topology scenarios.

For each of the four scenarios, we have repeated the tests for tasks belonging to three application types. Figure 125 shows the amount in dollars charged to tenant towards execution of three types of applications for four fog deployment scenarios mentioned earlier. From the graph, we can say that network-aware pricing approach fairly charges tenants based on the application type and its fog infrastructure resource utilization.

Figure 126 shows the impact of number of hops in path between device and fog node for both host-based and network-aware service pricing approaches. As seen in the figure, host-based approach does not reflect the additional costs incurred by increased number of hops and thus cannot differentiate between two fog nodes of same resource capacities located at different network distances from device, resulting in orchestrator not being able to find a cost-efficient fog node to deploy and execute a given task. Network-aware approach resolves this shortcoming, as can be seen in Figure 125. Network-aware service pricing approach considers network distance

in terms of number of hops between source and destination. Thus, network aware approach helps fog orchestrator prefer local fog node to host a given application, as compared to a node farther away from user.

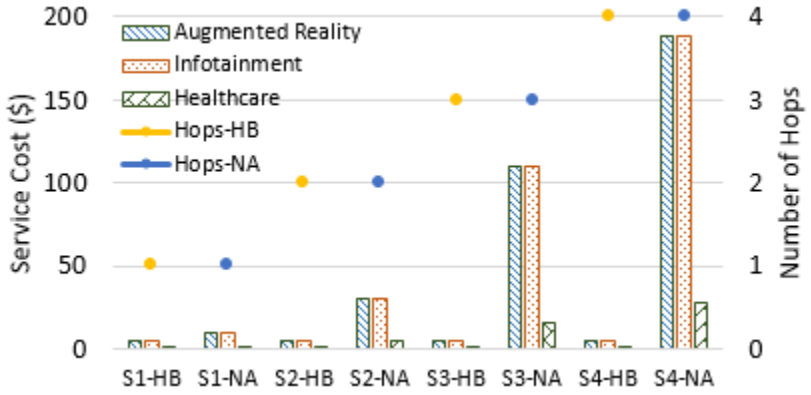

Figure 126: Comparison of service cost using Host-based and Network-aware approaches with increasing number of hops in network topology between device and fog node.

8.8. Conclusions

In this paper, we have proposed a novel approach towards pricing of fog network services which is a component of FIaaS fog service model. We have discussed the need for a pricing approach for fog different from that of cloud. We have shown the feasibility of orchestrator being knowledgeable of network topology in fog environment and proposed network-aware service pricing approach which fairly charges tenants based only on the resources consumed by their application environments as well as ensure that fog service providers are paid by fog tenants according to the infrastructure resources consumed, including compute, storage, and network. We have shown by experiments that our proposed approach fairly charges the fog tenants according to their network resource consumption while ensuring that fog service providers are not compelled to offer resources for free. We have discussed fog network pricing in

this paper. We are working on development of pricing approaches for other fog resources such as compute, storage, etc.

9. PFogSim: A Simulator for Performance Evaluation of Mobile and Hierarchical Fog Computing Environments

9.1. Abstract

This chapter provides a detailed description of a new simulation environment, PFogSim, developed to facilitate testing fog computing configurations of various sizes with fog nodes distributed over large geographies and are interconnected over one or more network links. PFogSim allows for a multi-layered fog design instead of limiting the system to a static single or two-layered design. Furthermore, PFogSim allows fog nodes and network links to be of different capacities enabling realistic, real-world networks to be tested. PFogSim fills the void in which simulations for large, practical fog networks are lacking.

9.2. Introduction

The data-intensive nature of new generation applications such as those from the IoT domain is moving computation away from centralized cloud data centers and towards the end devices. On the other hand, lack of enough resources on end devices along with their higher costs and lower reliability tends to move computations towards nodes having more resources, preferably the Cloud. Hence, to distribute the load appropriately there is a need for multi-layered fog computing deployments.

9.3. Motivation

Initiatives for pilot implementation of large-scale smart city environments are already in place at several large cities such as Barcelona, Melbourne, London, Beijing, and New York. Researchers have worked on proof-of-concept implementations of fog infrastructure to support such smart city environments [61] as well as applications leveraging the fog [68]. We have discussed in [107]the significance of management of fog infrastructure resources and application services and identified the need to solve various research problems to facilitate the realization of large-scale fog environments. There is a need for a simulation environment which satisfies the unique characteristics of fog environments as specified in Fog Computing Conceptual Model by NIST [16]. These unique characteristics are contextual location awareness and low latency, geographical distribution, heterogeneity, interoperability and federation, real-time interactions, scalability and agility of federated fog-node clusters, predominance of wireless access, and support for mobility.

9.3.1. A need for fog simulator features and functionality.

Several simulators are available to validate the performance of fog environments such as CloudSim [117] [118], DEVS [119], FogTorch [101], FogTorchII [120], iFogSim [121], EdgeCloudSim [109], etc. However, these simulators lack support for one or more of the unique characteristics of fog environments as defined by NIST [16]. Listed below are some of the features required for fog environment simulation.

1. Hierarchical structure with multiple types of fog nodes.
2. Mobility of devices which are data generators and consumers and/or users.
3. Mobility of fog nodes.
4. Large number of devices, users, and fog nodes.
5. Network interconnection and dynamic routing.

6. Location support, i.e. specification of unique locations for system entities.
7. Geographical distribution of system entities.
8. Support for cloud integration.
9. Support for performance evaluation with metrics specific to fog environment.
10. Support for evaluation of various service placement and migration approaches.
11. Support for evaluation of various pricing models.

9.4. Contributions

The main contributions of this research are as follows:

- Proposed and implemented PFogSim, a Java-based event-driven simulator, uniquely tailored for large scale, multi-layered hierarchical fog environments supporting continuous mobility of devices, users, and/or fog nodes. The simulator software is available for public use on GitHub [122] and is about 16,000 lines of code.
- Provided a detailed description of simulator components, classes implemented, their purpose, and interaction among them.
- Discussed the ease of implementation and testing of new service management approaches using PFogSim.
- Provided the use of a simulator to evaluate and compare the performance of various application service orchestrators such as Cloud-Only Orchestrator which assigns tasks only to the cloud services, Edge-Only Orchestrator which assigns tasks only to the edge nodes, and Centralized Orchestrator in which a central manager sends tasks to different fog nodes while reducing the total cost.

9.5. PFogSim Overview

PFogSim is an event-driven simulator developed in Java to simulate the execution of applications in a distributed large-scale fog environment with thousands of fog nodes, data generating and consuming devices, as well as users. It has been extended from EdgeCloudSim [109] to reflect the unique features of fog environments such as heterogeneity, location-awareness, the mobility of devices, users, as well as fog nodes themselves. The PFogSim simulator can be used to test various orchestrators which employ different service placement methods, fog service pricing approaches, and management of fog infrastructure resources. It facilitates analysis of test results by auto-generation of graphs from test logs to compare selected approaches using various metrics. It supports user-defined application types with custom orchestrators. The user can load their own application profile and use a custom orchestrator to process them.

PFogSim simulator was developed initially to validate the Hierarchical and Autonomous Fog Architecture (HAFA) proposed in [84], which is based on the concept of Puddle, a logical grouping of fog nodes having similar resource configuration and are in vicinity. The first letter 'P' of the term Puddle is used to uniquely name the developed simulator as PFogSim to signify the purpose.

9.6. Features implemented

Following is brief description of various features implemented in PFogSim.

- **Large scale:** PFogSim facilitates simulation of large smart environments with thousands of data generating devices such as sensors, data consuming devices such as actuators, as well as users and fog nodes.

- **Heterogeneity support:** PFogSim supports heterogeneity in system by categorizing fog nodes with their type. This facilitates testing service management approaches which recognize and consider heterogeneity of resource configurations towards management of application services and task scheduling.
- **N-Layered architecture:** Fog nodes with infrastructure resources which serve as compute, storage, and networking devices in the system can be identified and associated with a specific fog layer. PFogSim supports simulation of fog environments with variable, multi-layered hierarchy; thus facilitating simulation and testing of N-Layered fog environments. This feature allows testing various architectures proposed specifically for multi-layered fog environments, and not just edge computing, mobile computing, and cloud computing environments.
- **Network topology:** PFogSim builds network topology among simulated entities using predefined XML files, which can be pre-created or generated as part of the test environment configuration.
- **Dynamic networking:** PFogSim allows the fog nodes to don the role of compute devices and/or network devices dynamically. Thus, a given fog node can serve any or both roles simultaneously. Fog nodes can be dynamically configured as network devices to allow multi-hop networking between devices.
- **Dynamic routing:** Network routes are calculated on-the-fly between a given source and destination. This facilitates simulation of large scale environments with no restrictions regarding network topology. Thus, network congestion delay, multiple hops in network path, long links, and their impact on overall response time are accurately captured in metrics.

- **Mobility support:** PFogSim supports continuous mobility of devices and fog nodes in a specific direction at a given pace. The location of a system entity can be retrieved at any given time instant.
- **Locality support:** Each entity in the system such as device, user, cloud node, fog node, network node, etc. is identified by its location which can be specified in two forms – Global coordinates in the form of (latitude, longitude) representing the geographic position, or Cartesian coordinates in the form of (X, Y) depicting relative positions of various entities in the system. Representing the position of entity in the form of GPS coordinates facilitates simulation testing using real world data sets.
- **Separation of send/receive path:** Data packets between two given system entities are not required to follow the same route owing to the dynamic routing feature. Thus, current network state information can be leveraged to make more knowledgeable decisions. This facilitates mobility of devices and allows dynamic changes in test system configuration.
- **System configuration:** Initialization module of PFogSim allows inputs in the form of CSV files. Along with additional network topology information, it converts them to XML files in required format, which are leveraged by PFogSim to generate test system configuration.
- **Modularity:** Modular nature of the simulator facilitates implementation of various infrastructure resource and service management approaches as well as logical fog architectures with minimal effort.

- **Dynamic service management:** Application services can be dynamically placed on or removed from a given fog node over the course of simulation. This is an improvement from other fog simulators where services are statically placed during system setup phase as per specifications from XML documents.
- **Dynamic service migration:** Application services can be migrated to different fog node in case of mobility of device or fog node or both.
- **Task execution:** Application tasks can be generated and executed as per given resource and QoS specifications.
- **Orchestrators and task execution policies:** PFogSim supports testing various application service orchestrators and task execution policies in a single test run, facilitating comparative performance analysis. The current implementation includes several orchestrators such as cloud-only, edge-only, local-only, centralized approach, etc.
- **Metrics:** Several metrics are captured to reflect the utilization of fog resources at each layer, along with congestion in network, and its impact on task execution. All metrics are logged in specific predefined format for easier analysis of test results.
- **Graphical results analysis:** MatLab scripts are provided to facilitate pictorial comparison of metrics using various orchestrators and for varying number of devices in the simulated environment.

9.7. PFogSim Class Components

In this section, we provide a detailed description of various components of PFogSim simulator including their significance, implementation, interaction, limitations, open issues, and

any relevant additional information. Figure 127 shows the interconnection of various modules of the simulator.

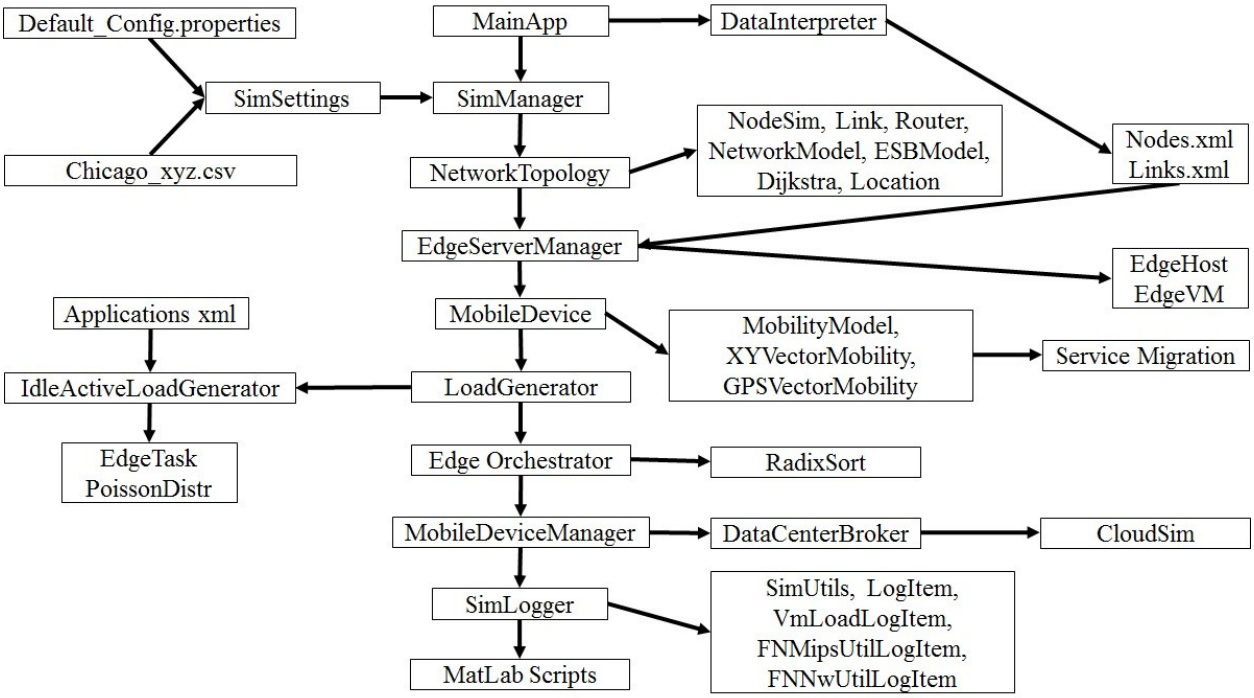

Figure 127: PFogSim – Pictorial representation of interdependency of simulator modules

9.7.1. MainApp

MainApp module includes the main method to initiate the simulation. It includes functionality to configure the test environment by specifying the number of devices, orchestrator approach selected for the test run, as well as file paths for various test system configuration files and the folder to store the test results. The Simulation environment is created as per specifications from XML files directly provided as input or those generated by DataInterpreter module. There is no predefined limit on the number of devices, and nodes of test environment, as well as number of layers of fog structure.

9.7.2. Default_Config.properties

Default_Config.properties file includes a list of system configuration parameters and their values, which define the test environment. The set of parameters include the simulation run time, warmup period, rate at which metrics are captured, list of supported orchestrators, minimum and maximum number of devices, etc. among others.

9.7.3. SimSettings

The SimSettings class contains the configuration information about the simulation such as the application configuration, mobile device number, and the total simulation time. This class provides the simulation parameters defined in Default_Config.properties file to various modules. It also assigns default values for several system-wide parameters such as supported application types,

9.7.4. SimManager

The SimManager is the overarching module which controls the execution of several individual modules to perform specific operations. It initializes the state of several modules discussed later in this section. It is extended from the SimEntity class of CloudSim [118] [117] and processes various events at scheduled epochs. For example, it schedules recurring tasks to verify and print simulation progress every 1% of elapsed simulation time and to capture node and network utilization metrics, among others, at specified intervals. The total simulation time is set in Default_Config.properties file.

9.7.5. DataInterpreter

The DataInterpreter module creates a hierarchical fog environment with a predefined number of layers. Test system configuration including the number of layers is provided as input to the simulation in the form of CSV and XML files. Locations of fog nodes belonging to each fog layer are provided in separate CSV files. These can be in the form of Cartesian (X, Y)

coordinates within the simulation space, or GPS locations in the form of (latitude, longitude) spanning the globe.

Resource configuration as well as the execution and data transfer costs for fog nodes belonging to each fog layer are provided in this module. Leveraging node resource configurations per fog layer and locations of individual nodes, it creates XML files, 'node_test.xml' and 'links_test.xml' which have specifications for individual nodes and network links between the nodes based on predefined network topology.

Network link latency is defined based on distance between nodes as distance in kilometers * 0.01 millisecond, where distance is calculated as Euclidean distance or as great circle distance using Haversine formula depending on whether the node locations are specified in the form of Cartesian (X, Y) or GPS coordinates.

The implementation of this module can be modified to test different fog network topologies by changing the inter-node connectivity as well as individual node locations provided in csv files. The node resource configurations, link characteristics, and respective costs can be modified in the xml generation component. Additionally, new resource parameters can be added or removed from the current set in this module.

DataInterpreter is an optional module and is required only to generate the nodes and links xml files. If there is no change in test network topology and resource configurations, then the initialize method of this module can be removed from simulation execution.

9.7.6. EdgeServerManager

The EdgeServerManager (ESM) module uses the XML files with fog nodes and links configuration to create respective entities in the simulated environment. Based on the specifications from corresponding XML file, each fog node is created as a Data center object, which has a single host with given node resource configurations. Each host has only one VM created which consumes all the resources available on the host fog node.

For each host, a NodeSim object is created, which is used to configure NetworkTopology using configuration from links XML file. If the nodes are logically organized for instance, HAFA architecture, the corresponding implementation needs to be incorporated in this module.

Based on the resource configuration and other specifications, each fog node can be associated with any one of the N fog layers, where N is the maximum number of layers in fog hierarchy. The fog nodes belonging to layers 1 through N are assumed to be physically stationary. Sensors (data producers), actuators (data consumers), users, and mobile fog nodes are associated with fog layer 0.

9.7.7. NetSim

The NetSim module incorporates the entire support for simulating a full network and routing environment, including the NodeSim, Link, NetworkTopology, Router, and ESBModel classes. The NodeSim and Link classes represent the physical location and network attributes of the machines on the network separate from their resources. These classes are used by the NetworkTopology to form a map of the entire network, representing only the location and static connections. The NetworkTopology is considered to be static for all intents and purposes. That is, it does not contain dynamic information about the network, but rather, represents the physical characteristics of the network that are not likely to change within the scope of the simulation.

The Router class uses the map represented by the NetworkTopology to find the least latency path from one node to another. While the router does contain the network map, it does not require the up to date dynamic state of the network and thus does not truly qualify as a centralized system.

9.7.8. NodeSim

The network is represented by nodes and the links between them. Each node in the network representation corresponds to a fog node in the system, and contains the node's location and the static links associated with that node. The Node Sim module provides the functionality for fog node specifications including its physical location, resource configuration, fog layer association, mobility status, mobility vector i.e. pace and direction, Wi-Fi Access Point availability, WAP Id, etc.

9.7.9. Link

The links are created between two node objects as specified in the corresponding XML file. A link contains the locations of the link's endpoints (called the left and right endpoints), both of which must be valid nodes, and the left and right latencies for the link. The left and right latencies refer to the static delay associated with travelling the link from the right node to the left node (left latency), and from the left node to the right node (right latency). These left/right values are assigned based on the distance from the endpoints of the link.

Static network latency is associated for each link based on the length of the length i.e. distance between the two end points of link. Default value assigned to link latency is 0.01 millisecond * link length in kilometer [111]. Direct links are assumed to be fiber optic cables in which the speed of data transfer is 2 X 10^8 m/sec, i.e. a delay of 0.005 millisecond per km. A multiplier of 2 is used to cater to TCP retransmissions over long links. Thus the individual link

latency is calculated as follows: Link propagation delay = Link length in kilometer * 0.005 * 2 millisecond.

All the links are assumed to be bidirectional. The link should have exactly two end points and each node has at least one link. There is no limit on the number of links that a fog node can have.

9.7.10. NetworkTopology

The Network Topology module defines the static network configuration such as physical characteristics of the network that are not likely to change within the scope of the simulation. The links and nodes together form a network topology. The network topology structure holds all of the nodes in the network and ensures that there are no isolated nodes, i.e. the nodes with no links and no dangling links i.e. the links with no node at one or both of its endpoints.

9.7.11. Router

The Router module identifies the network path with minimum latency between two given entities. Router takes source and destination nodes and the network topology. It reads the network topology as a weighted directed graph with nodes being vertices and links and delays representing edges and their weights. The router then runs Dijkstra's algorithm on the graph from the given source node. Afterwards, the router finds the destination node and builds a linked list of the nodes that form the shortest path from the source to destination. Router then passes this linked list to the network model to calculate total latency.

The simulator assumes a central router. This assumption is to abstract routing as a baseline for the ideal routing for the network. We make this assumption on the basis that the router does not need to maintain the full status of the entire environment. Rather, the router only requires the static map of the network and the links that connect it. This is purely for the

simulator and any interaction between a node and the router is to simulate the activity of a realistic one. This router does not assume every node has the image of the full network. It cannot be emphasized enough how this simulator is made for a decentralized system in which it is unrealistic to know the state of the network at every time. The router does not violate this since it only ever gives information the nodes would naturally have in any given circumstance.

9.7.12. NetworkModel

The NetworkModel is an abstract class in EdgeCloudSim responsible for calculating path latencies in the network. The network model calculates the local delay due to congestion at each network node and adds all the congestion and static delays to find the total delay from source to destination.

9.7.13. ESBModel

The default network model packaged with PFogSim is the ESBModel. The ESBModel class is used for calculating the network delay from one location to another, which includes congestion delay and propagation delay.

Congestion delay is calculated based on Equal Share Bandwidth (ESB) model, which divides the total available bandwidth at the given network component equally among the connected entities i.e. links. Congestion delay is defined as Average task size * Number of devices accessing the link at the moment / Link bandwidth.

Network path delay is the sum of congestion delays at each network component along with path and the static link latencies i.e. propagation delay assigned to each of the component links.

9.7.14. MobileDevice

The MobileDevice module represents device/user and includes functionality to submit tasks to the system. Devices can be configured individually as stationary or mobile. In current implementation, there is no separation between devices (data producers) and users (data consumers). Each mobile device is assigned a unique id and location. The devices connect to the fog environment over wireless link and is associated with the physically nearest Wi-Fi Access Point.

9.7.15. Mobility

The Mobility module includes functionality to support continuous independent mobility of devices, users, and fog nodes in the system. Entities move in a specific direction at a specific speed defined randomly at the start of the simulation. If they touch the borders of given area, then they turn around and take up a random direction and continue moving. So, there is no start and end position defined, but a continuous / continual movement observed at the granularity of 1 second simulation time.

The MobilityModel is an abstract class which is used for calculating the location of each mobile devices with respect to the time.

The mobility feature can be enabled collectively or individually on any of the entities of the system i.e. devices and fog nodes. During system setup, each device is co-located with any one of the fog nodes. During initialization, a randomly defined mobility vector is assigned individually for each entity which defines the direction of movement and the pace at which the entity moves in the system. At the borders of the given area, the direction is reassigned in a random manner and the movement continues at the predefined pace. The position of each entity is observed at the granularity of 1 second of simulation time.

XYVectorMobility class is used to represent the relative location of entities in the Cartesian coordinate system. Distance between two such locations is calculated using root-sum-squared formula. GPSVectorMobility class is used to represent the global location of entities in the GPS coordinate system. Distance between two GPS locations is calculated as great circle distance using Haversine formula.

9.7.16. Orchestrators

During the system initialization phase, each mobile device is assigned a fog node which hosts the corresponding application service instance and assumes responsibility to execute all tasks submitted by the device. Selection of a fog node for assignment to each device depends on the orchestration approach, and is performed by orchestrator.

EdgeOrchestrator is the abstract class which provides generic functionality for all custom orchestrators extended from it. The DistRadix class implements Radix sort for finding the nearest location from a given point among a set of locations. DistRadix is used by some of the implemented custom orchestrators.

PFogSim provides several custom orchestrators as listed below and has a modular structure to facilitate inclusion of new orchestrators with minor effort.

- Edge orchestrator
- Local orchestrator
- City center orchestrator
- Cloud orchestrator
- Centralized orchestrator
- Selected nodes orchestrator
- Selected levels orchestrator

The implementation details of orchestrators listed above are provided in section 7.11.10.

9.7.17. Service Placement

Service placement module assigns a fog node to each device such that all tasks generated by the device are submitted to the specific fog node. Assignment of fog node to any device is done according to the rules defined in custom orchestrator. This procedure also involves the reservation of compute resources in assigned fog node for the workload expected to be submitted by device as well as the reservation of network resources on fog node and at each of the network nodes en route from device to executing fog node. Resource reservation procedure is successful only if the fog node has sufficient free resources to support the device workload specifications including its QoS characteristics such as response time. Additionally, checks are performed to identify availability of free bandwidth resources on all network nodes in path from device to fog nodes. This ensures successful transfer of input and output data between device and executing fog node.

9.7.18. Applications

Application resource requirements can be specified in XML format. Multiple application profiles can be included in XML document with properties defining various factors such as task generation rate, task resource requirements for execution, acceptable response time, etc.

9.7.19. Task generation

Tasks are generated during system initialization phase as per the specifications provided in applications XML document. All the tasks created can be of same application type or they can be a mix of application types with proportion as specified in applications XML document. Tasks are submitted by each device at the time of execution to the executing fog node as assigned by the orchestrator. LoadGeneratorModel is the abstract class which provides the generic functionality towards task generation by various devices. IdleActiveLoadGenerator class is the

corresponding implementation included in PFogSim. This class generates tasks for each device according to the Poisson arrival rate defined in applications XML file. All the generated tasks are scheduled for submission to system at predefined simulation times for processing by fog nodes.

9.7.20. MobileDeviceManager

This module submits the tasks to executing fog node as assigned by orchestrator. The submitted tasks are scheduled for execution by CloudSim at specified instant, which upon completion returns the status to MobileDeviceManager module. It tracks the progress of task execution in the system, updates the task state upon its successful completion or failure due to various reasons. It also captures individual task performance metrics and logs them.

9.7.21. CloudSim

Minor updates were made to CloudSim such as changing variable types for Ram and Pes from integer to double to allow testing of system configurations with nodes having large resources. The corresponding updates files are included in PFogSim package.

9.7.22. SimLogger

The SimLogger module is responsible for maintaining execution status and other state information for each task submitted into the system. At the end of simulation, SimLogger consolidates the metrics captured for all tasks and summarizes the results, which are stored in various log files in a predefined format.

9.7.23. Metrics

Several metrics are implemented as discussed in 7.11.9 to facilitate comparison of various environment configurations and test scenarios. Metrics are recorded in separate log files for each application profile and in a consolidated manner.

9.7.24. Scripted execution

Bash script is provided for automated execution of multiple scenarios varying the number and mobility nature of devices for different orchestrator approaches towards application service management. Observations from multiple test runs are recorded in a predefined format in log files to facilitate automated log analysis.

9.7.25. Results analysis

MatLab scripts are provided to analyze the log files generated from simulation test runs and generate graphs in an automated manner for various metrics as listed earlier comparing the system performance for various test configurations and orchestrators.

9.8. General system workflow

This section discusses how to use the PFogSim simulator. The sequence of steps in using the simulator are as follows:

9.8.1. Pre-simulation

1. Import test configuration:
    a. DataInterpreter module reads Chicago city information [110] and generates two XML files with edge/fog/cloud nodes and network links configuration respectively. The Chicago city data set has been discussed in Quantitative evaluation section.
    b. SimSettings module imports simulator settings from several configuration files.
2. Create a Virtual Network System:
    a. SimManager is initialized.
    b. LoadGeneratorModel module creates tasks.
    c. EdgeServerManager module initializes datacenters, hosts, and VMs.

d. EdgeServerManager module configures Network Topology of fog environment.
e. Orchestrator is initialized.
f. MobilityModel module is started.
g. MobileDeviceManager module instantiates devices and sets their initial locations.
h. Orchestrator assigns a host to each mobile device.

3. SimManager instance starts the simulation.

9.8.2. On-simulation

1. MobileDeviceManager schedules tasks generated by devices for processing.
2. Upload task to host:
    a. MobileDeviceManager calculates upload time. If the time is smaller than the requirement, MobileDeviceManager will upload the task to the host. Otherwise, the task will be rejected.
    b. SimLogger records task status.
3. CloudSim processes the tasks on hosts and returns results.
4. Download task to the device:
    a. The MobileDeviceManager calculates download delay. If the delay is smaller than the requirement, it will download the task to the mobile device. Otherwise, the task will be abandoned.
    b. SimLogger records task status.
5. MobileDeviceManager captures metrics per task.

9.8.3. Post-simulation

1. SimLogger consolidates metrics by averaging over all successful tasks.

2. SimLogger posts the captured metrics to files and console.
3. SimManager instance stops the simulation.

9.9. Sample test run

Listed below are some of the steps to be performed towards executing a simulation test run with PFogSim:

1. Specify the following in DataInterpreter.java
    a. Number of fog layers.
    b. Names of separate csv files, one for each fog layer, with GPS locations for nodes belonging to the layer.
    c. Node resource configurations, Network link configurations, costs, mobility characteristics of fog nodes, etc. in initialize method.
2. Set the following test run parameters in Default_Config.properties file
    a. Simulation time (minutes).
    b. Minimum, maximum, and increment device counts. For example, setting these parameters to 500, 1000, 100 will result in repeating the test for same orchestrator and system configuration with 500, 600, 700, 800, 900, and 1000 devices, i.e. six test runs are performed.
    c. Simulation scenarios.
3. Select the orchestrator in MainApp::main() method by setting the variable 'iterationNumber' to the corresponding orchestrator's index value from list of simulation scenarios, specified in properties file.
4. Select one or more application profiles to test by setting their corresponding 'usage_percentage' parameter in applications.xml to a number in the range 0-100%.

If a single application profile is selected, set the parameter to 100%. If an application mix is selected, ensure that the corresponding settings for all applications sum to 100%.

5. Create the folder to store test results in PFogSim\sim_results\ folder with name ite{ $iterationNumber}.
6. Run the MainApp class.
7. After successful test run, test logs will be available in corresponding iteration folder in sim_results folder. These log files can be used to generate graphs for various metrics using MatLab scripts provided with the simulator package.

9.10. State of the art

Several simulators are developed and used by researchers to validate Internet of Things, Cloud Computing, Edge Computing, Mobile Cloud Computing, and/or Fog Computing environments. In this section, we provide a brief description of some of them, along with a comparison of the salient features offered by the simulator and how do they compare with the corresponding functionality provided by PFogSim simulator as well as their applicability to fog computing environments.

CloudSim [118] [117] was developed to simulate cloud computing environments hosted on servers located in data centers. It facilitated validation of various service management and scheduling approaches, energy management methods, pricing models, and many more. However, CloudSim support for fog computing environments was limited as it does not allow simulation of device mobility and geographic distribution of various entities.

EdgeCloudSim [109] was extended from CloudSim to simulate mobile cloud computing environments. It simulates task execution system model in which the orchestrator makes a

decision to submit the task to an edge device or to cloud for execution. All nodes are configured with all application VMs statically as defined in input XML file. As the orchestrator makes decision for every task submitted to system, this model is not applicable for simulation of IoT/Cloud environments which follow service oriented application architecture. PFogSim allows validation of dynamic service placement and migration approaches. In PFogSim, all fog nodes may not have all applications running, as opposed to EdgeCloudSim where all edge servers have the same set of four VMs running. Instead, using service deployment approach, it identifies a good fog node to deploy the application and using service selection approach, pick one fog node from such set of fog nodes having same application sunning, to serve the user request.

Additionally, EdgeCloudSim assumes wireless network only and that all edge nodes are accessible at a single hop in wireless environment. However, this model is not applicable for large scale, geographically distributed fog computing environments with fog nodes interconnected and accessible over a variety of networks. EdgeCloudSim supports nomadic mobility model, i.e. a device can connect to any of the WAP points randomly at a given instant. This approach is acceptable for a campus scenario which has campus-wide wireless network. However it is not applicable for large scale IoT environments with devices such as autonomous vehicles moving at a higher pace and over larger areas. PFogSim supports continuous mobility of devices, users, and/or fog nodes at a individual predefined pace and in a given direction. Compared to the single log output of CloudSim, EdgeCloudSim supports the graphical representation of results.

IFogSim is also extended from CloudSim. It allows simulation of IoT devices and fog nodes with predefined resource characteristics, and multi-layered architecture. It facilitates simulation of application deployment on a set of fog nodes and measures network delay,

throughput, and total cost. It allows simulation of applications with multiple service components deployed on different fog nodes. However, iFogSim does not support features such as mobility of devices and/or fog nodes as well as complex network topologies.

PFogSim is extended from EdgeCloudSim to support additional functionality which is unique for fog computing environments as defined by [16]. PFogSim, being modular in nature, can be leveraged to simulate various network topologies and architectures which represent the complex and realistic computing and network deployments. Custom orchestrators can be developed in PFogSim to simulate Cloud computing, Mobile Cloud computing, and Edge computing environments. The simulation test results can be graphically represented using MatLab scripts provided as part of simulator.

9.10.1. Gaps with other fog simulators

Table 13: Simulators for Fog environments – A comparison.

Feature	**PFogSim**	**EdgeCloudSim**	**CloudSim**	**iFogSim**
Simulator	Yes.	Yes.	Yes.	Yes.
Designed for Fog Environments	Yes.	No. Edge Computing Environments.	No. Cloud Computing Environments.	Yes.
Support for large deployments	Yes.	Limited to small campus deployments.	Yes.	No. Manual assignment of application modules to

Feature	PFogSim	EdgeCloudSim	CloudSim	iFogSim
				nodes limits scalability.
N-Layered architecture support	Yes.	No.	No.	Yes.
Location specification	Yes.	Yes.	No.	No.
Mobility support of end devices and users	Yes. Location updated continually at granularity of 1 second.	Limited. Nomadic mobility only.	No.	No.
Mobility of fog nodes	Yes.	No.	No.	No.
Dynamic networking support	Yes.	No.	No.	No. Pre-specified path only.
Separation of send/receive path	Yes.	No.	No.	No.
Modular	Yes.	Yes.	Yes.	Yes.

Feature	PFogSim	EdgeCloudSim	CloudSim	iFogSim
Metrics per fog layer	Yes.	No.	No.	No.
Graphical representation of results	Yes.	Yes.	No.	No.

9.11. Bugs / known issues

Here are some of the known bugs with simulator, as well as open issues to be resolved.

- No two fog nodes in the system can have the same position coordinates (XY/GPS). This needs to be ensured in input data set. Failure of which will lead to inconsistencies in test results. There is a troubleshooting module provided as part of code to deal with this issue, but it needs to be manually enabled when a problem is encountered. Refer to the class 'BlackHoleException' for further details.
- GPS distance between entities with latitude/longitude values of 180/0 may result in incorrect values.
- Location class cannot be used in a HashSet or a HashMap, as the 'Contains' method does not work. Instead, use TreeMap or a TreeSet.

9.12. Future work

Current implementation of PFogSim has several limitations. We are working on extending the simulator functionality to support additional features, some of which are listed below.

- Mobility of fog nodes
- Separation of device i.e. data producer and user i.e. data consumer
- Deployment of composite application services
- Support for three-dimensional position coordinates to support fog deployments in 3D structures such as multi-storied buildings, etc.
- Average energy consumption per task execution and for transfer of data.

9.13. Limitations

Following are some of the limitations of current implementation of PFogSim simulator. Further development effort is required to incorporate the corresponding functionality.

- In current implementation, we assume a single VM is configured per host fog node using all of its entire resources. The single VM can support multiple application service instances with different application types.
- The current implementation of the simulator limits fog nodes and network nodes to being physically immobile i.e. stationary.
- The simulator supports independent services. Composite services with individual component services having interdependencies are not implemented.
- Each data center has only one host and each host has only one VM defined. The reason for this decision is that fog nodes are geographically distributed and are identified by their location. However, in CloudSim, only the Datacenter object has a Location attribute, not the Host. Thus, to comply with CloudSim implementation, fog nodes are simulated using Datacenter class, instead of Host class.
- Each fog node has only one VM. All application services are deployed on it.

- Geographic locations of all fog nodes (and/or devices) must be unique. Duplicates may result in inconsistencies with results.
- Mobile fog nodes are associated only with Fog layer-0.
- The simulation test run time is proportional to the number of tasks successfully executed. This dependency limits the maximum number of device configuration that can be tested during a simulation run. Hence, PFogSim simulator software architecture needs to be reviewed and reworked to improve the test run performance as well as utilization of CPU and memory resources.

9.14. Conclusion

In this chapter, we have proposed and implemented PFogSim, a Java-based event-driven simulator, uniquely tailored for large scale layered hierarchical fog environments. It supports continuous mobility of devices, users, and fog nodes. We have provided the simulator workflow, a detailed description of the unique features of PFogSim, and their applicability in the evaluation of multi-layered fog environments. By restricting the number of fog layers and resource configurations of node types, PFogSim can be used as-is with no additional programming effort to also evaluate the performance of cloud computing, edge computing, and mobile computing deployments.

10. Conclusions

This research provided a brief overview of the concept of Fog Infrastructure as a Service and its components. It proposed a fog architecture, HAFA, with connected multi-layered logical hierarchy for large scale, heterogeneous, and widely distributed fog environments based on Puddles, which are groups of fog nodes in close vicinity, connected over homogeneous network connections. HAFA groups fog nodes in vicinity towards resource pooling and local control, and logically links groups of fog nodes from different layers to facilitate disaster readiness, ad hoc deployment, and distributed control over extended area. It also helps reducing effort in finding an efficient node with required resource characteristics for deployment of a service.

This research assumed that fog nodes comprising a FIaaS deployment can be categorized into a small number of node types based on their resource characteristics. Further research needs to be done regarding identification of an optimal number of node types into which the available fog nodes can be categorized such that the differences between nodes of various types are more evident as compared to those belonging to same type, and are significant enough to differentiate various service requests to be assigned and served by nodes from different layers.

This research proposed network-aware service pricing approach for large fog environments.

Leveraging HAFA and pricing approach, this research proposed cost-efficient approach towards placement of application services in heterogeneous fog environments.

It has been evaluated using PFogSim, a simulator we have developed to represent large scale and multi-layered fog environments. Performance of proposed approach is observed for various application profiles with different resource characteristics and service requirements.

Here are the highlights of HAFA placement approach:

- Distributed approach.
- Leverages local state knowledge only.
- Expands search space only as far needed, until request is successful, or search space is exhausted.
- Maintains locality.
- Better network utilization.
- Cost-efficient, considering both computation and communication costs.

Limitations of HAFA approach are as follows:

- Based on the premise that the set of fog nodes can be categorized into a small number of fog node types.
- Fog nodes of each category, referred as layer, are uniformly dispersed in the fog environment.
- Fog nodes belonging to each layer have same execution cost.

- Fog nodes are physically stationary.
- Fairness and balancing of workload on fog nodes is not considered.

In addition to the research problems solved during this research, this research has identified and discussed in brief a set of research problems to be solved towards management of infrastructure resources and applications in FIaaS environments. These are provided in Appendix-A and Appendix-B.

References

[1] W. Shi and M. Liu, "Tactics of handling data in Internet of things," in *2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Beijing, China, 2011.

[2] S. Dey, A. Mukherjee, H. S. Paul and A. Pal, "Challenges of Using Edge Devices in IoT Computation Grids," in *International Conference on Parallel and Distributed Systems (ICPADS)*, Seoul, South Korea, 2013.

[3] N. A. Ali and M. Abu-Elkheir, "Data management for the Internet of Things: Green directions," in *2012 IEEE Globecom Workshops*, Anaheim, CA, USA, 2012.

[4] M. Ma , P. Wang and C.-H. Chu, "Data Management for Internet of Things: Challenges, Approaches and Opportunities," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Beijing, China, 2013.

[5] D. Bandyopadhyay and J. Sen , "Internet of Things: Applications and Challenges in Technology and Standardization," *Wireless Personal Communications,* vol. 58, no. 1, pp. 49-69, 2011.

[6] D. Miorandi, S. Sicari, F. De Pellegrini and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks,* vol. 10, no. 7, pp. 1497-1516, 2012.

[7] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal,* vol. 1, no. 1, pp. 3-9, 2014.

[8] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Computer Networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[9] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang and K. Long, "Cognitive Internet of Things: A New Paradigm Beyond Connection," *IEEE Internet of Things Journal,* vol. 1, no. 2, pp. 129-143, 2014.

[10] A. M. Ortiz , D. Hussein , S. Park , S. N. Han and N. Crespi, "The Cluster Between Internet of Things and Social Networks: Review and Research Challenges," *IEEE Internet of Things Journal,* vol. 1, no. 3, pp. 206-215, 2014.

[11] H. Yue , L. Guo , R. Li , H. Asaeda and Y. Fang, "DataClouds: Enabling Community-Based Data-Centric Services Over the Internet of Things," *IEEE Internet of Things Journal,* vol. 1, no. 5, pp. 472-482, 2014.

[12] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, no. 7, pp. 1645-1660, 2013.

[13] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC '12 Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Helsinki, Finland, 2012.

[14] S. Yi , Z. Hao , Z. Qin and Q. Li, "Fog Computing: Platform and Applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Washington, DC, USA, 2015.

[15] L. M. Vaquero and L. Rodero-Merino, "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing," *ACM SIGCOMM Computer Communication Review,* vol. 44, no. 5, pp. 27-32, 2014.

[16] M. Iorga , N. Goren , L. Feldman , R. Barton , M. Martin and C. Mahmoudi , "SP 500-325 Fog Computing Conceptual Model," 19 March 2018. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/500-325/final. [Accessed 17 June 2019].

[17] NIST, "NIST Releases Report on Fog Computing for Internet of Things Devices," 29 October 2018. [Online]. Available: https://csrc.nist.gov/News/2018/Fog-Computing-for-Internet-of-Things-Devices. [Accessed 17 June 2019].

[18] NIST, "SP 800-191 (DRAFT) The NIST Definition of Fog Computing," August 2017. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-191/archive/2017-08-21. [Accessed 17 June 2019].

[19] OpenFog Consortium Architecture Working Group, "OpenFog Consortium Reference Architecture: Executive Summary," 2017.

[20] N. Mohamed , J. Al-Jaroodi , I. Jawhar , S. Lazarova-Molnar and S. Mahmoud, "SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services," *IEEE Access; Topic: The New Era of Smart Cities: Sensors, Communication Technologies and Applications,* vol. 5, pp. 17576-17588, 2017.

[21] T. Clohessy , T. Acton and L. Morgan, "Smart City as a Service (SCaaS): A Future Roadmap for E-Government Smart City Cloud Computing Initiatives," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, London, UK, 2014.

[22] M. Satyanarayanan , P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha , W. Hu and B. Amos, "Edge Analytics in the Internet of Things," *IEEE Pervasive Computing,* vol. 14, no. 2, pp. 24-31, 2015.

[23] N. Chen , Y. Chen , Y. You , H. Ling , P. Liang and R. Zimmermann, "Dynamic Urban Surveillance Video Stream Processing Using Fog Computing," in *IEEE Second International Conference on Multimedia Big Data (BigMM)*, Taipei, Taiwan, 2016 .

[24] M. A. Al Faruque and K. Vatanparvar, "Energy Management-as-a-Service Over Fog Computing Platform," *IEEE Internet of Things Journal,* vol. 3, no. 2, pp. 161-169, 2016.

[25] M. Aazam and E.-N. Huh, "E-HAMC: Leveraging Fog computing for emergency alert service," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, St. Louis, MO, USA, 2015.

[26] K. Vatanparvar and M. A. Al Faruque, "Energy management as a service over fog computing platform," in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems (ICCPS '15)*, Seattle, Washington, 2015.

[27] H. P. Breivold and K. Sandström, "Internet of Things for Industrial Automation -- Challenges and Technical Solutions," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Sydney, NSW, Australia, 2015.

[28] J. Li , J. Jin , D. Yuan , M. Palaniswami and K. Moessner, "EHOPES: Data-centered Fog platform for smart living," in *2015 International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW, Australia, 2015.

[29] Y. Cao, P. Hou, D. Brown, J. Wang and S. Chen, "Distributed Analytics and Edge Intelligence: Pervasive Health Monitoring at the Era of Fog Computing," in *Proceedings of the 2015 Workshop on Mobile Big Data (Mobidata '15)*, Hangzhou, China, 2015.

[30] Y. Shi, G. Ding, H. Wang, H. E. Roman and S. Lu, "The fog computing service for healthcare," in *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*, Beijing, China, 2015.

[31] S. Yi, C. Li and Q. Li, "A Survey of Fog Computing: Concepts, Applications and Issues," in *Mobidata '15 Proceedings of the 2015 Workshop on Mobile Big Data*, Hangzhou, China, 2015 .

[32] Y. N. Krishnan, C. N. Bhagwat and A. P. Utpat, "Fog computing - Network based cloud computing," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, 2015.

[33] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *2014 Australasian Telecommunication Networks and Applications Conference (ATNAC)*, Southbank, VIC, Australia, 2014.

[34] N. Peter, "Fog computing and its real time applications," *International Journal of Emerging Technology and Advanced Engineering,* vol. 5, no. 6, 2015.

[35] O. Ferrer-Roca, R. Tous and R. Milito, "Big and Small Data: The Fog," in *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, Beijing, China, 2014.

[36] M. Firdhous, O. Ghazali and S. Hassan, "Fog Computing: Will it be the Future of Cloud Computing?," in *The Third International Conference on Informatics & Applications (ICIA2014)*, Kuala Terengganu, Malaysia, 2014.

[37] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Athens, 2014.

[38] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei and L. Sun, "Fog computing focusing on mobile users at the edge," *arXiv:1502.01815,* vol. https://arxiv.org/abs/1502.01815, p. 1, 2015.

[39] F. Bonomi, R. Milito, P. Natarajan and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence,* vol. 546, pp. 169-186, 2014.

[40] S. K. Datta, C. Bonnet and J. Haerri, "Fog Computing architecture to enable consumer centric Internet of Things services," in *2015 International Symposium on Consumer Electronics (ISCE)*, Madrid, Spain, 2015.

[41] OpenFog Consortium Architecture Working Group, "OpenFog Architecture Overview: White paper," 2016.

[42] L. F. Bittencourt, M. M. Lopes, I. Petri and O. F. Rana, "Towards Virtual Machine Migration in Fog Computing," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Krakow, Poland, 2015.

[43] A. Al-Fuqaha, M. Guizani , M. Mohammadi , M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials,* vol. 17, no. 4, pp. 2347-2376, 2015.

[44] S. Sarkar , S. Chatterjee and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," *IEEE Transactions on Cloud Computing,* vol. 6, no. 1, pp. 46-59, 2018.

[45] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, K. Mankodiya, P. Liljeberg and H. Tenhunen, "Fog Computing in Body Sensor Networks: AnEnergy Efficient Approach," in *IEEE International Body Sensor Networks Conference (BSN)*, Cambridge, MA, 2015.

[46] J. K. Zao , T.-T. Gan , C.-K. You, C.-E. Chung , Y.-T. Wang , M. S. J. Rodríguez , T. Mullen , C. Yu , C. Kothe , C.-T. Hsiao , S.-L. Chu , C.-K. Shieh and T.-P. Jung , "Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology," *Frontiers in Human Neuroscience,* vol. 8, p. 370, 2014.

[47] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. T. Wang, T. Mullen and T. P. Jung, "Augmented Brain Computer Interaction Based on Fog Computing and Linked Data," in *2014 International Conference on Intelligent Environments*, Shanghai, 2014.

[48] J. H. Ahnn and M. Potkonjak, "Toward energy-efficient and distributed mobile health monitoring using parallel offloading," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka, Japan, 2013.

[49] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert and G. Tamm , "Smart Items, Fog and Cloud Computing as Enablers of Servitization in Healthcare," *Sensors & Transducers Journal,* vol. 185, no. 2, pp. 121-128, 2015.

[50] M. Whaiduzzaman, M. Sookhak, A. Gani and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer Applications,* vol. 40, no. 1, pp. 325-344, 2014.

[51] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing (MCC '13)*, Hong Kong, China, 2013.

[52] E. Lee , E.-K. Lee, M. Gerla and S. Y. Oh, "Vehicular cloud networking: architecture and design principles," *IEEE Communications Magazine,* vol. 52, no. 2, pp. 148-155, 2014.

[53] O. T. T. Kim , N. D. Tri , V. D. Nguyen , N. H. Tran and C. S. Hong, "A shared parking model in vehicular network using fog and cloud environment," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Busan, South Korea, 2015.

[54] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services (MobiSys '14)*, Bretton Woods, New Hampshire, USA, 2014.

[55] N. K. Giang , M. Blackstock , R. Lea and V. C. Leung, "Developing IoT applications in the Fog: A Distributed Dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, Seoul, South Korea, 2015.

[56] A. Botta , W. d. Donato , V. Persico and A. Pescapé, "On the Integration of Cloud Computing and Internet of Things," in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, Spain, 2014.

[57] Y. Igarashi, K. Joshi, M. A. Hiltunen and R. Schlichting, "Vision: Towards an Extensible App Ecosystem for Home Automation through Cloud-Offload," in *Mobile Cloud Computing and Services (MCS) 2014 At ACM MobiSys 2014*, Bretton Woods, NH, USA, 2014.

[58] D. Willis, A. Dasgupta and S. Banerjee , "ParaDrop: a multi-tenant platform to dynamically install third party services on wireless gateways," in *MobiArch '14 Proceedings of the 9th ACM workshop on Mobility in the evolving internet architecture*, Maui, Hawaii, USA, 2014 .

[59] H. Chang , A. Hari , S. Mukherjee and T. V. Lakshman, "Bringing the cloud to the edge," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2014.

[60] F. Li , M. Voegler , M. Claessens and S. Dustdar, "Efficient and Scalable IoT Service Delivery on Cloud," in *2013 IEEE Sixth International Conference on Cloud Computing*, Santa Clara, CA, USA, 2013.

[61] J. Jin , J. Gubbi , S. Marusic and M. Palaniswami, "An Information Framework for Creating a Smart City Through Internet of Things," *IEEE Internet of Things Journal,* vol. 1, no. 2, pp. 112-121, 2014.

[62] I. P. Žarko, K. Pripužić , M. Serrano and M. Hauswirth, "IoT data management methods and optimisation algorithms for mobile publish/subscribe services in cloud environments," in *2014 European Conference on Networks and Communications (EuCNC)*, Bologna, 2014.

[63] M. Satyanarayanan , R. Schuster , M. Ebling , G. Fettweis , H. Flinck , K. Joshi and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *IEEE Communications Magazine,* vol. 53, no. 3, pp. 63-70, 2015.

[64] R. Khan, S. U. Khan, R. Zaheer and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," in *2012 10th International Conference on Frontiers of Information Technology*, Islamabad, India, 2012.

[65] S. Abdelwahab, B. Hamdaoui, M. Guizani and A. Rayes, "Enabling Smart Cloud Services Through Remote Sensing: An Internet of Everything Enabler," *IEEE Internet of Things Journal,* vol. 1, no. 3, pp. 276-288, June 2014.

[66] J. Preden , J. Kaugerand , E. Suurjaak , S. Astapov , L. Motus and R. Pahtma, "Data to decision: pushing situational information needs to the edge of the network," in *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, Orlando, FL, USA, 2015.

[67] T. Verbelen, P. Simoens, F. D. Turck and B. Dhoedt, "Cloudlets: bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services (MCS '12)*, Low Wood Bay, Lake District, UK, 2012.

[68] Y. Lin and H. Shen, "Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Experience," in *2015 IEEE 35th International Conference on Distributed Computing Systems*, Columbus, OH, USA, 2015.

[69] A. Pamboris, M. Báguena, A. L. Wolf, P. Manzoni and P. Pietzuch, "Demo:: NOMAD: An Edge Cloud Platform for Hyper-Responsive Mobile Apps," in *MobiSys '15 Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, Florence, Italy, 2015.

[70] M. Aazam and E.-N. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud (FICLOUD '14)*, Washington, DC, USA, 2014.

[71] M. A. Hassan , M. Xiao , Q. Wei and S. Chen, "Help your mobile applications with fog computing," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*, Seattle, WA, USA, 2015.

[72] S. V. Valvåg, D. Johansen and Å. Kvalnes, "Position paper: elastic processing and storage at the edge of the cloud," in *Proceedings of the 2013 international workshop on Hot topics in cloud services (HotTopiCS '13)*, Prague, Czech Republic, 2013.

[73] G. Lewis , S. Echeverría , S. Simanta , B. Bradshaw and J. Root, "Tactical Cloudlets: Moving Cloud Computing to the Edge," in *2014 IEEE Military Communications Conference*, Baltimore, MD, USA, 2014.

[74] U. Drolia , R. Martins , J. Tan , A. Chheda , M. Sanghavi , R. Gandhi and P. Narasimhan, "The Case for Mobile Edge-Clouds," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, Vietri sul Mere, Italy, 2013.

[75] J. Zhu, D. S. Chan , M. S. Prabhu , P. Natarajan , H. Hu and F. Bonomi, "Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, Redwood City, 2013.

[76] N. B. Truong, G. M. Lee and Y. Ghamri-Doudane, "Software defined networking-based vehicular Adhoc Network with Fog Computing," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, 2015.

[77] T. Fisher, "What is a Next-Generation Intelligent Application?," 29 June 2017. [Online]. Available: https://mapr.com/blog/what-is-next-gen-app/. [Accessed 2 May 2018].

[78] A. Dawar, "A Modern Way to Think About Your Next-Generation Applications," 11 April 2018. [Online]. Available: https://www.datanami.com/ 2018/04/11/a-modern-way-to-think-about-your-next-generation-applications/ . [Accessed 2 May 2018].

[79] P.-N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, (First Edition), Boston, MA, USA.: Addison-Wesley Longman Publishing Co., Inc., 2005.

[80] ITU, "Draft new Report ITU-R M.[IMT-2020.TECH PERF REQ] - Minimum requirements related to technical performance for IMT-2020 radio interface(s)," International telecommunications Union (ITU), 23 February 2017. [Online]. Available: https://www.itu.int/md/R15-SG05-C-0040/en. [Accessed 17 June 2019].

[81] S. S. Manvi and G. K. Shyam, "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey," *Journal of Network and Computer Applications,* vol. 41, no. 1, pp. 424-440, 2014.

[82] S. M. Parikh , N. M. Patel and H. B. Prajapati, "Resource Management in Cloud Computing: Classification and Taxonomy," *CoRR,* vol. arXiv preprint arXiv:1703.00374, 2017.

[83] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management,* vol. 23, no. 3, p. 567–619, 2015.

[84] S. Shaik and S. Baskiyar, "Hierarchical and Autonomous Fog Architecture," in *ICPP 2018: 47th International Conference on Parallel Processing Companion Proceedings*, Eugene, OR, USA, 2018.

[85] N. Choi , D. Kim , S.-J. Lee and Y. Yi, "A Fog Operating System for User-Oriented IoT Services: Challenges and Research Directions," *IEEE Communications Magazine,* vol. 55, no. 8, pp. 44-51, 2017.

[86] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He and Q. Yang, "A Hierarchical Distributed Fog Computing Architecture for Big Data Analysis in Smart Cities," in *Proceedings of the ASE BigData & SocialInformatics 2015 (ASE BD&SI '15)*, Kaohsiung, Taiwan, 2015.

[87] A. A. Alsaffar , H. P. Pham, C.-S. Hong, E.-N. Huh and M. Aazam, "An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing," *Mobile Information Systems,* vol. 2016, 2016.

[88] . C. C. Byers, "Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks," *IEEE Communications Magazine,* vol. 55, no. 8, pp. 14-20, 2017.

[89] A. Munir , P. Kansakar and S. U. Khan, "IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things," *IEEE Consumer Electronics Magazine,* vol. 6, no. 3, pp. 74-82, 2017.

[90] 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, 2017.

[91] M. Taneja and A. Davy, "Resource Aware Placement of Data Analytics Platform in Fog Computing," *Procedia Computer Science,* vol. 97, pp. 153-156, 2016.

[92] M. Taneja and A. Davy, "Poster Abstract: Resource Aware Placement of Data Stream Analytics Operators on Fog Infrastructure for Internet of Things Applications," in *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, Washington, DC, USA, 2016.

[93] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, USA, 2017.

[94] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *2017 IEEE International Conference on Communications (ICC)*, Paris, France, 2017.

[95] S. Wang, "Dynamic service placement in mobile micro-clouds," Doctoral Dissertation, Imperial College, London, 2015.

[96] R. Deng, R. Lu, C. Lai and T. H. Luan, "Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing," in *2015 IEEE International Conference on Communications (ICC)*, London, UK, 2015.

[97] I. Widjaja , S. Borst and I. Saniee, "Building an Elastic Cloud out of Small Datacenters," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, Delft, Netherlands, 2013.

[98] D. Zeng, L. Gu, S. Guo, Z. Cheng and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," *IEEE Transactions on Computers,* vol. 65, no. 12, pp. 3702-3712, 2016.

[99] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, Portugal, 2017.

[100] N. Daneshfar, N. Pappas, V. Polishchuk and V. Angelakis, "Service Allocation in a Mobile Fog Infrastructure under Availability and QoS Constraints," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018.

[101] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," *IEEE Internet of Things Journal,* vol. 4, no. 5, pp. 1185-1192, 2017.

[102] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, Japan, 2016.

[103] I. Farris, L. Militano, M. Nitti, L. Atzori and A. Iera, "Federated edge-assisted mobile clouds for service provisioning in heterogeneous IoT environments," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy, 2015.

[104] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan and K. K. Leung, "Dynamic Service Migration in Mobile Edge-Clouds," in *IFIP Networking Conference*, Toulouse, France, 2015.

[105] L. Gu, D. Zeng, S. Guo, A. Barnawi and Y. Xiang, "Cost-Efficient Resource Management in Fog Computing Supported Medical cyber-physical system (CPS)," *IEEE Transactions on Emerging Topics in Computing,* vol. 5, no. 1, pp. 108-119, 2017.

[106] P. Guo , B. Lin , X. Li , R. He and S. Li, "Optimal Deployment and Dimensioning of Fog Computing Supported Vehicular Network," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, 2016.

[107] S. Shaik and S. Baskiyar, "Resource and Service Management for Fog Infrastructure as a Service," in *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, USA, 2018.

[108] S. Shaik and S. Baskiyar, "Network-Aware Service Pricing Approach for Fog Infrastructure as a Service," in *2018 IEEE International Conference on Smart Cloud (SmartCloud)*, New York, NY, USA, 2018.

[109] C. Sonmez, A. Ozgovde and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of Edge Computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, Valencia, Spain, 2017.

[110] City of Chicago, "Chicago Data Portal," 2018. [Online]. [Accessed 1 June 2018].

[111] A. Yousefpour , G. Ishigaki , R. Gour and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," *IEEE Internet of Things Journal,* vol. 5, no. 2, pp. 998-1010, 2018.

[112] Microsoft, "Seeing AI 2016 Prototype - A Microsoft research project," Microsoft, 30 March 2016. [Online]. Available: https://www.youtube.com/watch?v=R2mC-NUAmMk. [Accessed 17 June 2019].

[113] OpenFog Consortium Architecture Working Group, "OpenFog Consortium Reference Architecture," 2017.

[114] A. Stringfellow, "Top IaaS Providers: 42 Leading Infrastructure-as-a-Service Providers to Streamline Your Operations," 7 Oct 2017. [Online]. Available: https://stackify.com/top-iaas-providers/. [Accessed 1 May 2017].

[115] Google, "Pricing | Network Service Tiers | Google Cloud," Google, 4 April 2019. [Online]. Available: https://cloud.google.com/network-tiers/pricing. [Accessed 17 June 2019].

[116] CDW LLC, "Cisco ASR 1001-X - router - rack-mountable - with Cisco ASR 1000 Series Emb," Cisco, 2018. [Online]. Available: https://www.cdw.com/product/cisco-asr-1001-x-router-rack-mountable-with-cisco-asr-1000-series-emb/4696619?pfm=srh. [Accessed 17 June 2019].

[117] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience,* vol. 41, no. 1, p. 23–50, 2011.

[118] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *2009 International Conference on High Performance Computing & Simulation*, Leipzig, Germany, 2009.

[119] M. Etemad , M. Aazam and M. St-Hilaire, "Using DEVS for modeling and simulating a Fog Computing environment," in *International Conference on Computing, Networking and Communications (ICNC)*, Santa Clara, CA, USA, 2017.

[120] A. Brogi , S. Forti and A. Ibrahim, "How to Best Deploy Your Fog Applications, Probably," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, Madrid, Spain, 2017.

[121] 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," *Journal of Software: Practice and Experience, Special Issue: Cloud and Fog Computing,* vol. 47, no. 9, pp. 1275-1296, 2017.

[122] S. Shaik, "PFogSim Simulator: GitHub page," 17 June 2019. [Online]. Available: https://github.com/szs0117/PFogSim. [Accessed 17 June 2019].

Appendix A – Research Problems with Infrastructure Resource Management for Fog Infrastructure as a Service

1.1. Introduction

In this chapter, we provide a brief description of various open research problems [107] related to infrastructure sizing and management of resources in fog environments. We have discussed the problems, and possible approaches towards solving them using our proposed fog architecture as reference.

1.2. Deployment of fog infrastructure

Prior to deployment, fog infrastructure needs to be sized to support a given application environment, i.e. number of fog nodes, and their infrastructure resource configurations such as compute, memory, storage, and network capacities need to be identified. Fog computing paradigm recommends the placement of fog nodes close to data sources and users to facilitate the deployment of latency-critical applications while reducing network traffic. Thus, deployment of fog infrastructure needs to identify placement location of fog nodes along with their resource configurations to support the applications. Note that the fog nodes will be of different resource configurations i.e. heterogeneous depending on QoS needs of applications as well as location of data sources and users. Towards the deployment of fog, three scenarios need to be considered as discussed below.

1.2.1. Greenfield Deployment

This problem considers the scenario where fog service provider is deploying new infrastructure, i.e. fog, and cloud nodes. The fog infrastructure is designed to be cost-optimal to host the given set of applications, arising from the freedom to choose from unlimited possibilities.

1.2.2. Brownfield Deployment

This problem considers the scenario where fog service provider has prior infrastructure, i.e. cloud nodes and other devices, which are being repurposed as fog nodes. This problem differs from Greenfield Deployment, in that the resulting fog environment will be a best effort configuration leveraging the available nodes.

1.2.3. Incremental Deployment

This problem considers the scenario where a given fog environment needs to be scaled to support additional tenants, applications, and/or users. As there are active tenants and users accessing the current fog infrastructure, addition of new nodes to the system should be least disruptive to current users.

Towards deployment of fog environment, a cost-optimal subset of available nodes, and additional nodes, if any, need to be identified along with their resource configuration, placement location and network connectivity. These fog nodes are then logically organized by associating them with a fixed set of predefined fog layers and Puddles, and identifying their parent/child as well as neighbor relationships. As the new fog nodes are not yet operational and there are no active users associated with them, this problem can be solved efficiently using large-scale cloud resources in an offline manner.

1.3. Tenant Mapping

From the set of heterogeneous fog nodes dispersed over vast geographical area managed by one (or more) service provider(s) in a federated fog environment, a tenant needs to be assigned infrastructure resources on a subset of nodes to deploy applications based on specific resource requirements and expected workload. Instead of an optimal mapping of resource requirements, finding a set of mapped nodes will be a best effort search from the set of available fog nodes limited by the cost constraints specified by tenant.

To solve this problem, we need to select an optimal substructure of available fog hierarchy to satisfy the resource requirements of tenant's applications. The selected substructure represents the set of fog nodes, the resources on which can be leased by the tenant to deploy and execute own services i.e. use the resources in a Fog Infrastructure as a Service manner. Owing to multi-tenancy feature of fog, tenants will not have knowledge of other tenants and their respective services and execution environments which may be sharing the same physical fog infrastructure. As the tenant's applications are not yet deployed and there are no active users associated with them, this problem can be solved efficiently in an offline manner by fog service provider.

1.4. Local Restructuring

Fog environments are dynamic, allowing fog nodes to join or leave at will, as well as move to a different physical location. This requires local restructuring of fog hierarchy to facilitate optimal reallocation of available resources to tenants, improve performance of application services by migrating them to more appropriate nodes, and to maintain the system in an efficient manner. Local restructuring updates logical fog hierarchy by (re)placing the fog node in question at an appropriate position in the neighborhood hierarchy i.e. association with fog

layer, Puddle, parent and children and neighbors. Three scenarios need to be considered towards this problem as follows:

1.4.1. Node join

New fog nodes may join the system during times of high workload or while scaling up the environment. It may also happen when a fog node is up after regaining energy, or when it is accessible over the network again. When a new node joins the system, it needs to be made available to current and prospective tenants which can leverage the node's resources to execute services. As a new fog node is not associated with the system yet, this problem can be solved efficiently in an offline manner by fog service provider.

1.4.2. Node loss

Fog nodes may leave the system arbitrarily due to loss of energy or network connectivity, which may result in a suboptimal fog hierarchy. As the lost fog node was already associated with the system and probably had active application services and connected users, this problem needs to be solved quickly to reduce the impact of disruption from node loss on availability of services.

1.4.3. Node move

Fog computing paradigm supports mobility of fog nodes with active tenants and services. When such a node physically moves out of scope of its current Puddle, its new position may result in suboptimal performance for application services running. An out-of-range move of a fog node can be usually predicted from its pace and direction of mobility. Hence, the procedure to identify destination Puddle and update fog hierarchical relationships can be performed efficiently offline by fog service provider.

1.5. System-wide Restructuring

Owing to dynamicity of fog environment i.e. multiple instances of fog node join, loss, and move, logical management structure of fog environment may result in a state very different

from the one it started with, over the course of time. The resulting system state may not be optimal to execute the current set of services and workload due to local restructuring instead of restructuring the entire system. Hence, the logical fog node hierarchy need to be reorganized at regular intervals to facilitate efficient management and optimal utilization of system resources. This procedure may result in release of fog nodes which are not currently being utilized in the system resulting reduced overall maintenance costs. Note that it is a background task; tenants and users of system need not be aware of this operation and hence should not be affected.

This operation is performed infrequently and at times when the system can be disrupted to re-optimize the fog environment. Hence, it can be performed efficiently by fog service provider considering the knowledge of complete historical and current system state. Periodicity of this operation depends on the dynamicity of fog environment as well as the tolerance of resulting impact on system performance by tenants and users.

Appendix B – Research Problems with Application Service Management for Fog Infrastructure as a Service

2.1. Introduction

In this chapter, we provide a brief description of various open research problems [107] related to management of application services in fog environments. We have discussed the problems, and possible approaches towards solving them using our proposed fog architecture as reference.

2.2. Service Replication/Consolidation (Elasticity)

Multiple instances of latency-critical services are distributed over large geographical areas to support requests from users in the vicinity. These individual instances are identified by the physical location of fog node hosting the service instance. In cloud environment, individuality of multiple co-located instances of a given service is insignificant and a given user request is redirected by load balancer to any of the instances optimizing the response time.

In a dynamic fog environment supporting mobility, users, IoT devices, and/or fog nodes could be mobile and change their geolocations over time. Thus, IoT devices and users intermittently move in or out of the realm of a given fog node, changing the workload and services executed on the fog node. The fog environment should be elastic and scale the system up or down depending on the changes in workload. As workload increases for a service on a given fog node and node's resources are insufficient to support it, then an additional instance of the service can be started on a different fog node and users need to be redirected to new instance. On the other hand, if users or IoT devices are no longer accessing a service instance, then it can be terminated. If the workload for multiple service instances is lower than a given minimum, then users and IoT devices accessing the services can be migrated to a smaller number of instances and the instances with no active workload can be terminated. As the users are actively accessing the services, any changes to the service instances should be oblivious to users in terms of their availability and performance.

2.3. Service Selection

For performance and availability purposes, multiple instances of a service may be active on a set of fog nodes. Upon user request for a service, there is a need to find a cost-optimal node from the set of nodes to serve the user's request such that the service QoS requirements are satisfied maximizing the number of fog nodes remaining active by optimizing remaining energy on data sources and fog nodes. Several factors need to be considered towards solving this problem such as relative location of fog node w.r.t. users and data sources, current utilization and availability of free resources on fog nodes, remaining energy on fog nodes, resource, energy, and QoS requirements of services, cost of service execution, nature of service e.g. computation-intensive or data-intensive, etc.

2.4. Periodic Re-optimization of services

In a dynamic fog environment, fog nodes join or leave the system as well as change their geolocation. To ensure that the service deployment and execution is cost-optimal to tenants and users, logical fog hierarchy is adapted continually according to changes in physical fog environment by local and system-wide restructuring as discussed in earlier section. Further to these actions, the services deployed in the system may need to be reshuffled to ensure cost-optimality of their execution. This operation is performed infrequently when the system can be disrupted to re-optimize the fog environment. Care should be taken to minimize the disruption of services.