Journal of Parallel and Distributed Computing 137 (2020) 1–16

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

COMITMENT: A Fog Computing Trust Management Approach

Mohammed Al-khafajiy [a], Thar Baker [a,*], Muhammad Asim [b], Zehua Guo [c], Rajiv Ranjan [d], Antonella Longo [e], Deepak Puthal [d], Mark Taylor [a]

[a] Department of Computer Science, Liverpool John Moores University, UK
[b] Department of Computer Science, National University of Computer and Emerging Sciences, Pakistan
[c] School of Automation, Beijing Institute of Technology, China
[d] School of Computing, Newcastle University, UK
[e] Department of Innovation Engineering, University of Salento, Italy

ARTICLE INFO

Article history:
Received 19 April 2019
Received in revised form 14 August 2019
Accepted 24 October 2019
Available online 8 November 2019

Keywords:
Fog computing
Trust
Quality of protection

ABSTRACT

As an extension of cloud computing, fog computing is considered to be relatively more secure than cloud computing due to data being transiently maintained and analyzed on local fog nodes closer to data sources. However, there exist several security and privacy concerns when fog nodes collaborate and share data to execute certain tasks. For example, offloading data to a malicious fog node can result into an unauthorized collection or manipulation of users' private data. Cryptographic-based techniques can prevent external attacks, but are not useful when fog nodes are already authenticated and part of a networks using legitimate identities. We therefore resort to trust to identify and isolate malicious fog nodes and mitigate security, respectively. In this paper, we present a fog COMputIng Trust manageMENT (COMITMENT) approach that uses quality of service and quality of protection history measures from previous direct and indirect fog node interactions for assessing and managing the trust level of the nodes within the fog computing environment. Using COMITMENT approach, we were able to reduce/identify the malicious attacks/interactions among fog nodes by approximately 66%, while reducing the service response time by approximately 15 s.

© 2019 Elsevier Inc. All rights reserved.

1. Introduction

Fog computing puts a substantial amount of cloud computing facilities at the edge of a network as opposed to establishing dedicated channels to a more centralized remote cloud infrastructure. This approach reduces service latency, improves the Quality of Service (QoS), and provides a superior experience to end-users [2,10]. As an emerging architecture, fog supports a wide variety of applications including Internet of Things (IoT), fifth-generation (5G) wireless networks, augmented reality and artificial intelligence (AI) [18]. Moreover, fog computing is generally considered to be more secure than cloud computing due to the following reasons: Firstly, the collected data is transiently maintained and analyzed on local fog nodes closest to data sources, which decreases the dependency on the Internet connections. Secondly, local data storage, exchange and analysis potentially make it more difficult for hackers to gain access to user's data, since there can be separate and different security barriers at different fog nodes. This limits the amount of user data that could be accessed in any given data breach compared to a more centralized cloud computing environment. However, the same level of security risks could apply to the data exchange between the user devices and the fog computing node or the data exchange between different fog nodes. Thus, there exist several challenges for preserving security and privacy in fog computing [45,76].

In fog computing, fog-based services are generally owned by different parties due to various reasons: (1) the deployment choice that may include the selection of Internet service providers or wireless carriers, (2) businesses extending their existing cloud-based services to the edge for performance improvement, (3) offering spare resources on the local private cloud as fog services to local businesses on lease [76]. This flexibility of offering different fog-based services by different providers complicates the trust situation between fog nodes. Moreover, the devices used by the fog users are often considered resourceful in-terms of their capabilities, but they are still incapable of executing certain complex tasks such as those required in applications like Image processing, virtual reality, augmented reality and smart transportation [1]. Thus, such tasks are offloaded and user's control over data is handed over to fog layer where fog nodes may independently or work in collaboration on the tasks to achieve the overall objective. Since, the outsourced data can be transferred to a rogue fog node, an adversary can tamper or steal user confidential data and can easily launch more attacks. A rogue node would be a malicious fog device that appears to

* Corresponding author.
E-mail address: t.baker@ljmu.ac.uk (T. Baker).

https://doi.org/10.1016/j.jpdc.2019.10.006
0743-7315/© 2019 Elsevier Inc. All rights reserved.

be legitimate and coaxes end users to use them, but, in reality, these nodes are malicious in nature. Various cryptographic-based approaches exist that can effectively prevent external attack, but are not useful in case of internal attacks where rogue fog nodes are already part of the application using legitimate identities. We, therefore, resort to trust to "single out" malicious fog nodes and mitigate security risk, respectively. Fog nodes are expected to be collaboratively monitored by their neighboring nodes for any sign of deviation from acceptable behaviors and predict their reliability for handling future jobs based on past reputation.

Contributions

The major contributions of this paper are threefold:

1. Fog COMITMENT: *COMputIng Trust manageMENT* approach to impart useful prognostic information on fogs trustworthiness. Thus, providing a secure and trusted fog computing environment to share node's resources and exchange data securely and efficiently. Further details can be found in Section 3.
2. A load balancing algorithm to monitor fog's resources (i.e., CPU consumption), active fog processes (e.g., stakeholder services processes), and the incoming services requests volume onto fog. Thereof, it is able to monitor fog's performance and to promote load balancing via offloading to address the latency concern on fog nodes, thus, triggering the offloading function upon fog congestion. Further details can be found in Section 4.
3. Trust and Recommendation model and the algorithm that helps fogs making the right decision for selecting the appropriate fogs to collaborate with during the offloading process. Thereof, this process includes assessing the trustworthiness level of the nominated fogs to ensure that the QoP and QoS provided by hosted fogs are meet. Further details can be found in Section 5.

Preliminaries

- Fog Quality of Service (QoS): we refer to fog QoS as the ability of fog to achieve maximum bandwidth (associate with the time to upload and download a packet $\tau_{||}$) and deal with the service's requests with minimal latency and low error rate. The problem preliminaries associate with QoS are the fog's workload (f_x), service workload on fog ($s_w^{f_i}$) and the total time required to process a service (τ_s).
- Fog Quality of Protection (QoP): we refer to fog QoP as the degree of which the fog protects the received data during processing as well as transferring or sharing the data with other fogs. The QoP properties (e.g., service integrity and confidentiality) are defined according to the type of processes and services provided by the fog. RoP problem preliminaries are associated with proposed trustworthiness model and based on the direct trust ($\tau_{a,b}^{d}$) and the recommendation/indirect trust ($\tau_{a,b}^{r}$).
- Fog Secure Service Level Agreement (SSLA): this refers to the commitment between two fogs in delivering a service according to a certain level of quality, availability and protection. Thus, SSLA includes the problem preliminaries associated with both QoS and QoP.
- Level of Trust (LoT): is a score that refers to the trustworthiness among fogs. LoT is computed based on the previous collaborations experiences, and is periodically updated after each collaboration. The problem preliminaries associate with LoT are the experience satisfaction score $ES_{a,b}$, the α and β which logs the satisfied and unsatisfied experience, respectively. LoT indicates the level of trust or distrust between the fogs, therefore, LoT score used based on a fuzzy logic where the score 1 is an indicator of absolute trust and the score 0 is an indicator of absolute distrust.

Paper structure

The rest of this paper is organized as follows. Section 2 provides background and motivation. Section 3 presents the proposed fog COMITMENT approach. Section 4 provides details on workload balancing via offloading. Section 5 discusses the trust and recommendation model. Section 6 reports the experiments results that back our fog-based trust model. Finally, Section 7 concludes the paper and identifies some future work points.

2. Background and motivation

In this section we highlight the potential security threats and attacks on fog computing and we define the key security requirements in a fog-2-fog collaboration model. However, we first discuss the fog computing architecture adopted for this paper.

2.1. Fog architecture

The fog computing architecture is similar to other large-scale distributed systems (e.g., cloud computing), the architectures proposed for IoT systems with a fog layer are either application specific, or application agnostic. However, there does not appear currently to be a commonly used standard architecture for fog computing [44]. In this paper, we adopt a general fog computing architecture, which is proposed in [3,7,20,24,38,77], given it the mostly renowned fog architecture. Understanding the fog architecture helps obtain a better insight into the functionalities and benefits of adding a fog layer. The main strata of the adopted architecture is composed of *Things*, *Fogs*, and *Cloud* stratum as per Fig. 1.

Things Layer: also called the *perception* layer, is the starting point of the IoT structure where data is generated. This layer contains the networked devices (e.g., heart-rate and blood-oxygen sensors), which operate to feed the system with data. Each Thing device in this layer is facilitated with a communication protocol (such as IEEE 802.15.4, WiFi, Bluetooth, MQTT, etc.) which permits the node to transmit the generated data to the fog layer over the IoT network.

Fog Layer: The fog layer contains a number of decentralized nodes in each given location. This layer handles the primary refining, computation, and processing of data generated from the Things layer. Fog nodes aim to improve the efficiency of IoT services, thus, fog has the potential to reduce the amount of data transmitted to the cloud layer and minimizing the request–response time for IoT services. Hence, fog enhances the *QoS* by reducing latency and improves network bandwidth.

Cloud Layer: Cloud or data-centers layer is the top layer of the IoT architecture enabling omnipresent, convenient, and network access to shared resources (e.g., storage, and services) over the IoT network. Thus, Cloud performs the "heavy services" of data analysis and processing [72] that fog cannot perform, such as big data processing.

The standardized approach in which IoT systems (with a fog layer) operates is as follows: the IoT t_n generates and gathers data periodically from the surroundings. The gathered packets will flow to either the *fog* layer or directly to *cloud* layer. When t_n sends these packets of data it initiates a request for service, i.e., the IoT services request is set-of-data packets sent from the *things* layer for processing. In fog layer, the f_i can serve the t_n service request instantly, or offload it to other fog node (e.g., F_x)

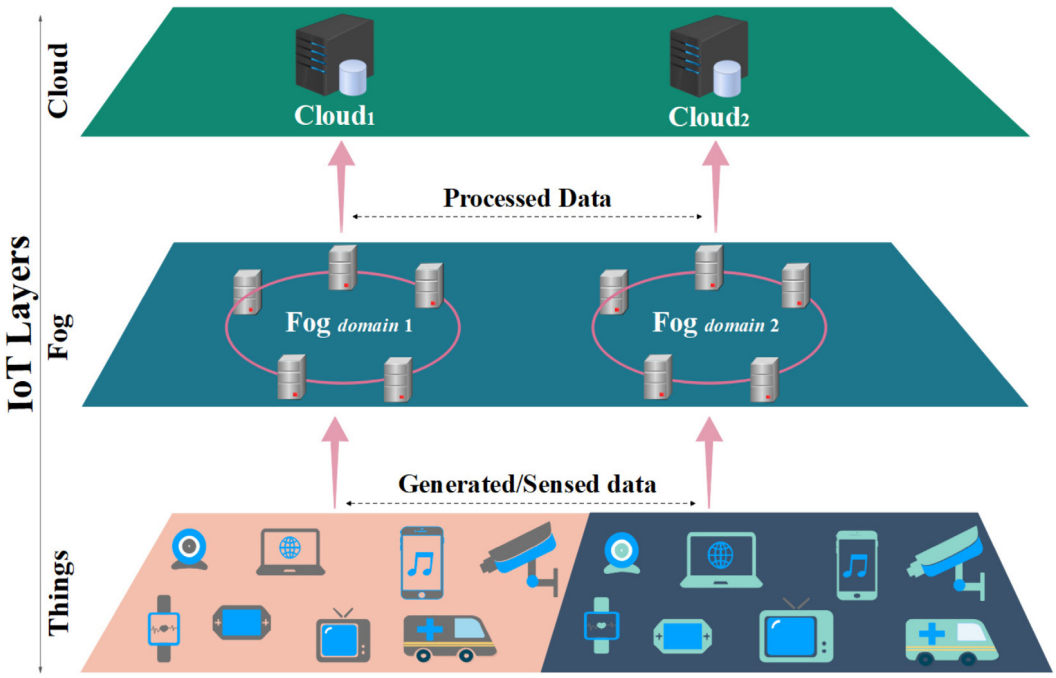


Fig. 1. LoT layers.

in the same domain to serve t_n because f_i is congested and may cause a service delay for t_n. To this end, f_i (or F_x) responds back to t_n and reports to cloud C_i for data archiving. Similarly, when packets are sent to C_i, it will be processed at this level and the response goes back to t_n. The fog layer is located between the things layer and the cloud layer, thus, it can handle a majority of IoT services in order to reduce the overall service delay. Therefore, in this research we only focus on processing all services dispatched from the things layer to the fog layer with intensive study on the offloading and cooperation between nodes to obtain the minimal service delay.

2.2. Threats and attacks on fog

A malicious fog node can disrupt network operations through various attacks, in this paper we consider the following attacks [45,54,61] that directly effect the reliability for fog-2-fog collaborations.

1. Forgery:- malicious fog nodes may forge their identities and fabricate fake data to mislead other fog nodes and IoT services. This type of node burden the network resources by excessively consuming network bandwidth, storage and computational power by running a fake services and fabricating large amounts of faked data.
2. Tampering:- malicious tampering fog nodes degrade fog efficiency by delaying, modifying or dropping the transmitted data. Detecting such malicious fog nodes is difficult as transmission failure or delay may be caused by other factors, such as unstable channel conditions or weak network signal, and not due to tampering fog.
3. Spam and Jamming:- this attack burden the network with unwanted content and data by generating big amount of bogus data to jam the network channels and fog's resources. Such attacks are generated and spread by malicious fogs to consume network and fog's resources so that fog become unavailable for other services and processes.
4. Impersonation: A malicious fog pretends to be a legitimate fog node to provide fog's services, but then it provide fake or phishing services to users and breach user's privacy.
5. Denial of Service (DoS):- malicious attacks to disrupts fog's services and make them unavailable to the intended users, by flooding the target fog nodes with superfluous service's requests. This attack consumes network resources to prevent the requests from legitimate users from being fulfilled. Fog are highly vulnerable to DoS attacks compared to the cloud due to fog's limited resources.

2.3. Fog security requirements

In order to enable a secure fog-2-fog collaboration model that provide a secure environment for outsourcing fog's resource and data sharing, the following security requirements should be fulfilled. Thus, these requirements defined as Requirements of Protection (RoP) which is a set of security requirements that includes all the security factors required to deliver the desired services securely and efficiently. Thus, RoP defines and measures the QoP among fogs, the more RoP are met, the better is the QoP.

1. Location and Identity:- fog responses to any collaboration's requests from other fogs should be based on an authentication process, such as fog's identity and location. The fog should be trusted by identifying the identity of fog nodes within the fog domain and identifies whether the provided fog location is real or fake before it accesses the desired services.
2. Service Integrity:- since the transmitted service's packets among fog nodes can be changed during the transmission time by malicious fogs, the packets must be checked so that it completely matches to what it sent initially (such as packet authentication from source). It is worth noting that the fog might be legitimate for collaboration, however the service's packets contain fabricated data, and thus, the bigger the distance between collaborating fogs, the higher is the risk of packet's attacks. Hence, the packets that are generated in a closer-distance and short-time are more reliable than packets arrived from long-distance and generated long-time ago.

3. Confidentiality:- The confidentiality in the fog-2-fog collaboration refers to data confidentiality. Since data packets are shared among fogs, the data may contain sensitive information, such as personal details (e.g., bank details), therefore, such confidentiality can be achievable by adopting public or symmetric key encryption to assure the security of the communications. Thus, the encryption of data prior to sharing is required to keep data secret and unreadable for distrusted or malicious fogs, and only trusted fogs can have the correct decryption key for the shared data.
4. Service Availability:- Fog services availability means that the services must be available when required. Unexpected situations such as service crashes would significantly affect service availability. Moreover, the fog should be able to tolerate DoS attacks that aim to crash the fog services. It is worth noting that the service distribution among fogs helps in enhancing services availability.
5. Trusted Fog:- the fogs trust each other based on past experiences obtained upon fog's collaborations. The ability of selecting the trusted fogs in a domain will help in providing the desired fog's services with high quality, hence, both QoE and QoP will be fulfilled. Moreover, the trust between fogs is:

    - Dynamic: the trust between fogs is dynamic and not static, so that fog_a trusts fog_b at a specific timestamp (e.g., t_1), however fog_a distrust fog_b at t_2 due to two reasons; (i) fog networks topology is continuously changing by adding or removing nodes from the fog domain. (ii) fogs within the domain may alter their behavior due to malicious attacks (e.g., DoS). Therefore, periodic trust assessment is essential.
    - Subjective: fog nodes may have different security measures to different types of processing so it meets the QoP. For example, fog_a can trust fog_b to carry out processes for traffic data, however, fog_b is not trusted enough to process healthcare related data.
    - Asymmetric and not transitive: each fog node has its own RoP that defines its QoP. Moreover, the RoP properties that one fog adopts can vary from one fog to another, hence, if fog_a finds fog_b is trustworthy, it is not necessarily that fog_b finds fog_a is trustworthy. Similarly, the trust is not transitive, for example, if fog_a trust fog_b and fog_b trust fog_c, it is not necessarily true that fog_a trusts fog_c

2.4. Research motivation

Fog computing is still an open research area and in its infancy stage, therefore, the motivation of providing a trusted fog environment for IoT based services comes from the open challenges and issues associated with fog computing. Many researchers are focusing on bringing the computing resources to network edges [5,51]. This will facilitate processing of the data at the edge for time-sensitive applications and services to allow quick responses. Fog nodes are deployed at the edge of the network, and they do not have enough resources and computational power like cloud [6,67]. As a result, fog nodes can easily get overloaded with incoming services requests. Also, another noted issue with the cyber-threats is of hostile/open deployment [49–51]. Hence, there are misbehaving fogs that for self-interest may perform *discriminatory* attacks to ruin the reputation of an IoT service [15]. Thus, avoiding a fraudulent or malicious fog nodes for load-balancing and collaboration is still an open challenge. These challenges rise the motivation to develop a fog COMputIng Trust manageMENT (COMITMENT) model that serves as a starting point for the development of such efficient and securely trusted fog computing environment.

2.5. Related work

In recent years, trust-based security solutions have been the focus of both industry and academia. Trust can help in detecting and isolating those malicious entities which are part of a network using legal identities. Moreover, the trust plays an important role in nurturing the relation between different fog nodes in terms of maintaining user privacy and information security [66]. Ideally, fog clients are expected to connect to any arbitrary fog node to avail its services such as computation, storage and processing, with a belief that the provided information is not to be misused. The integration of trust management in fog computing will assist fog nodes to select the most secure and trustworthy fog nodes in the vicinity according to their needs and requirements. For achieving this, all the participating fog nodes should have certain threshold of trust on each other. However, the development of a trust management mechanism for fog nodes is tricky due to its decentralized architecture. The main issue with the decentralized architecture is that it makes collection and management of evidence and behavior difficult which is required for the evaluation of trustworthiness of distributed fog nodes [46]. Table 1 shows a comparative analysis for COMITMENT with other researches, including the main objectives/scope (e.g., QoS and security enhancement) along with some features that can be provided, such as, fog's resource management and availability. It is clear that all most none of the reviewed research looked at the fog's resource management along with security aspect and availability of fog nodes.

There are many trust-based models which have been reviewed thoroughly in the literature [28,32,69]. Reputation is considered as an important parameter for the evaluation of trustworthiness. That is why, there are many mechanisms which employ this procedure for evaluating the trustworthiness in mobile ad hoc network (MANET) [19] along with vehicular ad-hoc network (VANET) [39], delay-sensitive networks [14] and mobile crowd sensing [53]. Kai Hwang with his team represented the idea for trust in clouds, in which he suggested to combine security-based data centers, data access and virtual clusters driven by reputation systems [35]. The work of [32] represents a trust mechanism using point based technique for protecting against unauthorized entry. For securing data transmission between two devices, trust was used in the gateway devices. However, it does not guarantee the credibility of sensor data and cloud providers. To overcome this short coming, the authors [36] proposed an Efficient Distributed Trust Model (EDTM) for WSNs. They randomly calculated direct trust values and recommendation trust values by evaluating the number of packets received by the sensor node. This approach is helpful in identifying different types of attacks. However, it is susceptible to processing and communication overheads. The work of [69] integrates the cloud and edge computing trust evaluation mechanisms which resulted in the considerable reduced resource usage for the evaluation of trust and increased IoT-cloud services efficiency. In this approach, they employed mean trust value, calculated on the basis of observed values obtained from the interacting devices. This may lead to communication overhead in the network.

The realization of offloading among fog nodes achieve resource efficiency and avoid bottlenecks, and overload [25]. There exist several mechanisms in the literature that focuses on the issue of offloading requests in a fog computing environment. However, they do not consider trust as a primary metric when it comes to offloading requests from one fog node to another [33]. The authors in [79] proposed a fog computing module that brings the fog computing power and resources closer to the mobile users through an offloading policy. The policy takes into account execution, energy and other expenses. Fricker et al. [27] proposed

Table 1
Comparative analysis for COMITMENT with other researches.

Research	Scope and research objectives							
	QoS	Latency	Security	Availability	Scalability	SSLA	Energy	Resource management
Deng et al. [20]	✓	✓	–	–	✓	–	✓	✓
Al-khafajiy et al. [5]	✓	✓	–	✓	–	–	–	✓
He et al. [31]	–	–	✓	–	–	–	–	–
Yannuzzi et al. [75]	✓	–	–	✓	✓	–	–	–
Chen and Hao [16]	✓	–	–	–	–	–	–	–
Giang et al. [29]	✓	–	✓	–	✓	–	✓	✓
Pahl et al. [48]	–	–	✓	–	–	–	–	–
Sarkar et al. [58]	✓	✓	–	–	–	–	✓	–
Skarlat et al. [60]	✓	–	–	✓	✓	–	–	✓
Gupta et al. [30]	✓	✓	–	–	✓	–	✓	–
Shen et al. [59]	–	–	✓	–	✓	–	–	–
Wen et al. [70]	✓	–	–	–	–	–	–	✓
Liu et al. [40]	✓	–	–	✓	✓	–	–	✓
Bhardwaj et al. [11]	–	–	✓	–	–	–	–	–
Wang et al. [65]	✓	✓	–	✓	✓	–	–	✓
Hu et al. [34]	–	–	✓	–	–	–	–	–
Vallati et al. [62]	✓	–	–	–	–	–	✓	–
Azimi et al. [9]	✓	–	–	✓	✓	–	–	✓
Markakis et al. [42]	✓	–	✓	–	–	–	–	✓
Chen and Xu [17]	✓	–	–	–	✓	–	✓	–
Ni et al. [47]	–	–	✓	–	–	–	–	–
COMITMENT (proposed)	✓	✓	✓	✓	–	✓	✓	✓

an analytic model to analyze a simple offloading strategy under heavy load for data centers in fog computing. The model considered forwarding request with a certain probability to neighboring data centers when the originally intended data center is overloaded. Moreover, requests can be blocked/rejected based on whether it can offload the arriving requests to other data centers. Zhang et al. [78] proposed an analytical framework to support fair offloading among multiple fog nodes while maintaining low delay. It selects fog nodes to offload tasks based on a fairness metric and rules that minimize the task delay. Massri et al. [43] presented a collaborative fog-to-fog communication algorithm that allows fog nodes to communicate and coordinate with each other to process IoT job requests.

Fog-based trust management is on its inception, because there has been very few reported work on the topic of trust mechanism in fog computing. In [8], the authors carried out a survey for finding the current security issues and challenges in Internet of Things and propose a fog-based security mechanism to improve the distribution of certification revocation information between IoT devices. The authors in [68] came up with the concept of fog-based hierarchical trust-based mechanism for SDN., which has two distinctive features that are based on trust in network structure, and the trust between cloud service providers (CSPs) and sensor service providers (SSPs). They focused on the packet loss rate, route failure rate and forwarding delay only. Elmisery et al. [23] proposed a fog-based middleware where trust between a fog node and the cloud is calculated in a decentralized fashion using entropy definition. The authors in [61] proposed a fuzzy trust-based model that takes into account experience and plausibility for securing vehicular networks. To ensure the correctness of information collected from authorized vehicles, a series of security checks are performed. Moreover, a fog-based facility is used to evaluate the level of accuracy of event's location.

In summary, several approaches exist in the literature that pay attention to both the issues of offloading and establishing trust between fog nodes. However, none of them consider trust as a primary metric for offloading or outsourcing requests in a fog computing environment.

3. Proposed fog COMputIng Trust manageMENT approach

Before we dive into COMITMENT details, it is worth mentioning the network environment we adopt for fog computing. In this paper we consider a distributed fog topology where nodes are physically distributed over different locations and connected to each other via communication protocol, thus every node has a unique identity address (e.g., IP). Moreover, the fog nodes are reachable to each other without a central controller (i.e., mesh networking) to help resource sharing and job offloading. In addition, there is no centralized trust authority among fogs to point out the trusted nodes within the network, thus, each node compute a trust evaluation periodically to its neighboring nodes and stores the generated list of trusted nodes locally.

COMITMENT is a software installed on each fog node within the fog layer. The COMITMENT is responsible for providing a secure and trusted environment for fogs to share their resources and exchange data packets and jobs, COMITMENT architecture shown in Fig. 2. Thus, COMITMENT provides a concise decision for the fog to *When it should offload jobs? and where to?*. The decision not only includes the best node that can handle the overload but also the most efficient fog that provides best QoS (e.g., low latency) and best QoP (e.g., meeting the SSLA). The offloading model we propose is to balance the workload and service's traffic within the fog layer by distributing service requests from the congested fog to another fog (e.g., job offloading). COMITMENT will be responsible for determining the overload on a congested fog as well as the trusted fog nodes that can handle the overload. In order to enable the COMITMENT to select the trusted node, it has to assess the QoP and QoS provided by the hosted fog through checking the trust level. The trust level is evaluated based on both direct interaction experiences of past interaction experiences and/or a recommendations from neighboring fog nodes in case of no previous experiences between two nodes. Obviously, the trust level will be computed based on the previous collaborations satisfactions, always the self experiences obtained from direct interactions will have a higher weight than recommendations from neighboring fog nodes because the trustworthiness among fogs is subjective and asymmetric as per fog security requirements in Section 2.3. Mostly used notations in this paper are given in Table 2. The main procedures and processes run by COMITMENT are categorized as follows:

1. Fog performance: COMITMENT periodically monitors fog's resources (e.g., CPU consumption), active processes (e.g., stakeholder's services processes), and the incoming

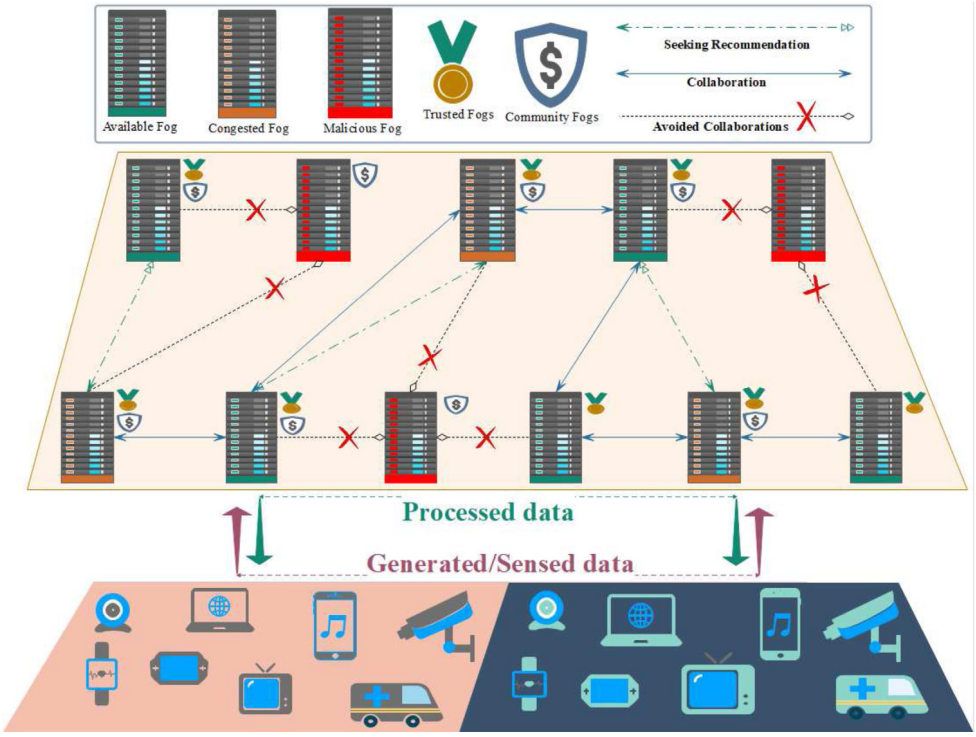


Fig. 2. Architecture of the proposed COMITMENT approach, including the different types of fog's statuses and interactions.

services requests traffic on the fog node in order to monitor fog performance. COMITMENT will trigger service's requests offloading function upon fog overload detection. Procedures to determine the overloaded service's requests, are discussed further in Section 4.

2. Fog interactions: upon overload detection, COMITMENT has the responsibility to handle the process of finding the best neighboring nodes that can handle the overload. This process includes assessing the trust level of the nominated fogs for handling the overload. This ensures that the QoP and QoS provided by the hosted fog meets the SSLA and user expectations about the desired service, for example, service run with no delay and assured data protection. This is discussed further in Section 5.

4. Workload balancing via offloading

Considering a scenario where a fog node accepts a data processing request from a thing; it will process the request and respond back. However, when the fog node is busy processing other requests, it may only be able to process part of the payload and offload the remaining parts to other fog nodes. The decision of a fog node to support the processing of a received data processing request or offloading the request to another fog is based on computing the response time of that fog [21,26,63,73]. The response time of each fog will be computed periodically based on the fog's current load (i.e., queue size) and service's request travel time (minimal latency always preferable). The procedure of offloading a received request by a fog is as follows: once a service request(s) is received by the fog node, it checks the request payload size (i.e., heavy or light) and calculates the potential response time based on the current requests that are waiting, and also under-processing, in its queue.

The workload of a fog (f_w) can refer to the overall usage of a fog's CPU, which is consumed during the processing of a particular service. Thus, there are constraints on a node's capability. This leads to a limitation on the ability of processing different types of services (i.e., heavy or light). Therefore, the workload assigned to a fog node f_w should not exceed the total capacity of the fog node f_i^c.

$$f_w \leq f_i^c, \forall f \in F \tag{1}$$

Table 2
Notations used in the paper.

Symbol	Description
t, n, T	Thing, index of t, set of things
f, i, F	Fog, index of f, set of fogs
λ	Service arrival rate to fog layer
μ	Fog node service rate
S, s	Set of services, one service
s_w	Service workload
$s_w^{f_i}$	Service workload for fog node (f_i)
s_d	Service deadline
τ_s	Total time required to process a service
t_s	Service's tasks
rs	Fog node resources
τ_{que}	Is the queuing time
τ_{pro}	Service processing time
ρ	System usage
τ_{que}^{si}	Queuing time for s at the resources of fog f_i
f_c	Fog capacity
f_w	Fog workload
f_i^c	Processing capacity of the fog node f_i
f_{rs}^s	Total fog resources (rs) allocated to processes service (s)
D_p	Propagation delay
$\tau_{\downarrow\uparrow}$	Time to upload and download a packet
α_{f_a,f_b}	Logs the satisfied experience from fog_a to fog_b
β_{f_a,f_b}	Logs the unsatisfied experience from fog_a to fog_b
$ES_{a,b}$	Experience satisfaction from fog_a to fog_b
n_{int}	Number of direct interactions between the two fogs
r_{f_a,f_b}	Recommendation of fog_a toward fog_b
$LoT(f_a, f_b)$	Level of trust score of fog_a toward fog_b
$C_{ts}^{f_i}$	Total CPU (in hertz), consumed by a t_s on fog node f_i
$\tau_{a,b}^d$	Direct trust of f_a toward f_b
$\tau_{a,b}^r$	Indirect trust of f_a toward f_b (recommendation)

It is worth noting that the total CPU used by the running services should not exceed the allocated fog resources for a service as the allocated resources are considered to be the total f_w can be handled by the fog. The total resources allocated to process a service are based on the type of service's packets (heavy-packets and low-packets) and the current load of the fog. Eq. (2) computes the total resources (rs) allocated to process all tasks (ts) for a service (s).

$$f_{rs}^{s} = s_w = \sum_{t=1}^{n} C_{ts}^{f_i}, \lceil s \rceil \leq f_c, \forall s \in S, \forall t \in T_s \tag{2}$$

The total fog's workload capacity (f_c) depends on the actual hardware specification of the allocated device. The assignment variable s_w (i.e., total service workload) is set so that total service processing workload does not exceed f_c, as per Eq. (2), where $C_{ts}^{f_i}$ denotes the total resource (CPU in consumption in hertz, having $hertz = cycles/second$) consumed by a service's tasks on fog node f_i.

In our research, we have followed real world scenarios where the data generated from the bottom layer can vary in size. Hence, we have separated between services workloads according to service's packets type, having a heavy-packets (e.g., packets generated from CCTV cameras) and low-packets (e.g., packets generated from ambient sensors [4,41]) service's requests. Hence, when a service only processes small data from sensors, this will consume low computational power, thus, the workload on fog is low. While, in services that performs heavy real-time video processing, the workload will be high on the fog node. Therefore, services workload (s_w) on fogs can vary for each service, depends on service's type [3,5]. The f_w for all services is the sum of each service workload multiples by λ as per Eq. (3). Thus, f_w should be less than the f_c assignment variable (i.e., $f_w <= f_c$).

$$f_w = \sum_{x=1}^{n} s_w^{f_i}.\lambda_s, \forall s \in S \tag{3}$$

4.1. Problem formulation and constraints

It is crucial to guarantee the minimal service delay to end-users during service processing at the fog layer. The total latency for a service's request sent from t_n to f_i is computed by adding the time of uploading a service's packets ($\tau_{\uparrow}$) to the waiting time for the service in fog queue (τ_{que}) until it gets processed. The delay for processing the service (τ_{pro}) and the time to response back ($\tau_{\downarrow}$) to t_n is also added with the total latency for the service as per Eq. (4). For simplification, we assume that($\tau_{\uparrow} = \tau_{\downarrow}$), having ($[\tau_{\uparrow} = \tau_{\downarrow}] = 2\tau_{\downarrow\uparrow}$) because logically the returned packet contents normally is similar or smaller than the sent packet.

$$\begin{aligned} \tau_s &= \tau_{\uparrow} + \tau_{que}^{s} + \tau_{pro} + \tau_{\downarrow}, \forall s \in S \\ \tau_s &= \tau_{que}^{s} + \tau_{pro} + 2\tau_{\downarrow\uparrow}, \forall s \in S \end{aligned} \tag{4}$$

We address the problem of having an optimal workload on fog nodes alongside with achieving minimal delay for IoT services. Thus, achieving reasonable load includes executing/processing the desired services within the threshold limit of fog capability. In addition, low latency for IoT services includes delivering the services within the required period, i.e., before service deadline (s_d) with the desired QoS. Therefore, the research problem can be defined as in Eq. (5).

$$P: \quad max[\tau_s] \leqslant s_d, \forall s \in S \tag{5}$$

$$\text{s.t.} \quad f_c^{min} \leqslant f_w \leqslant f_c^{max} \tag{6}$$

$$\sum \lambda_s \leqslant \sum \mu_f \tag{7}$$

$$P_s^d(n, p) \geqslant serviceLevel \tag{8}$$

$$\lambda_s \xrightarrow{min[D_p]} f_i \tag{9}$$

$$\tau_s \leqslant s_d, \forall s \in S \tag{10}$$

The constraints on this research are to reduce service latency. Therefore, our constraints are written with focus on achieving minimal service delay. In constraint (6), we indicate that (f_w) is strictly bound by an upper limit (f_c^{max}) and lower limit (f_c^{min}) which is related to fog capabilities based on CPU frequency (unit $hertz$). Constraint (7) imposes that the total traffic arrival rate (λ_s) to a fog domain should not exceed the service rate (μ_f) of this fog domain. Constraint (8) imposes the probability of directly processed services should be greater than or equal to the desired service level. Constraint (9) imposes the first destination for the IoT services traffic generated at the IoT Things layer will be to a fog node with minimal cost of propagation delay within the fog domain (ideally, lowest propagation delay is for the nearest fog node). Finally, constraint (10) is strictly bound by the service time τ_s that should be within the limit of service deadline s_d.

Algorithm 1: MAINTAIN FOG LOAD

Input: Fog (F_i); FogCapacity (F_c); QueueSize (Q_s)
Parameters : Offload (O_s); OverLoad (O_l); Services (S); ServiceType (S_t)
Initialisation: $F_i = \phi$; $F_c = \phi$; $Q_s = \phi$; $S = \phi$
Result: Determine Fog overload, if any.

Procedure 1. *Overload Threshold* **by**
  $F_c = F_i^c$ ▷ F_c `initiate fog`
  $Q_s =\longleftarrow getQueueSize(F_i)$
  $S = list\{Q_s\}$ ▷ `get list services`
  $S = sort(S,$ by $S_t)$
  for *each* $s \in S$ **do**
    $\tau_c^{si} = \tau_{que}^{si} + \frac{1}{\mu}$
    if $(\tau_c^{si} \geq S_d)$ || $\lambda \geq \mu)$ **then**
      $setFlag(O_s) = 1$
      break;
    else
      $setFlag(O_s) = 0$
    end
  end
  $F_{que} = timeCostFun(s, \tau_c^s)$
  $F_i \longleftarrow F_{que}$
  return (F_i, O_s)
End
Procedure 2. *Determine the Overload* **by**
  get (F_i, O_s)
  $F_c = getCapaxity(F_i)$
  $\mu = \frac{F_c}{F_{que}^i}$
  if $(O_s == 1$ || $\lambda \geq \mu)$ **then**
    for *each* $s \in F_{que}$ **do**
      $S = getSer\,vices(out : s \leftarrow \tau_c^s \geq S_d)$
    end
    $F_{que} = F_{que} - S$
    $O_l = S$
  else
    $get(F_i, O_s)$
    continue
  end
  return O_l
End

4.2. Offloading model

The offloading model proposes to balance the load within the fog domain by distributing service traffic from the congested fog nodes to other fogs within the domain. In order to balance services traffic in fogs domain, we assume that fogs at any giving location are reachable to each other within the same fog domain (i.e., mesh network), which models the fog network as a mesh network where each node can communicate with other nodes directly to allow load sharing, and this assumption in line with the work in [57,64]. In this research, we consider a real-world scenario of services flows where services arrival rates can significantly vary from one fog node to another [64] depending on fog location, since we have the constraint ($\lambda_s \xrightarrow{min[D_p]} f_i$) that is services are directed to nearest fog from thing for processing.

The decision factors where a node is congested and offloading is required relies significantly on fog workload (f_w), which is associated with the service traffic arrival rate (λ_s) and total processing rate (i.e., service rate μ) which is down to fog CPU frequency (i.e., node capability). In addition, service processing time τ_s, which ideally should not exceed service deadline (s_d). Therefore, to make the decision of offloading by a fog is when $\tau_s > s_d$, as per Probability (11), having (O_s) for offloading service decision:

$$O_s = \begin{cases} 1, & \text{if } \tau_s > s_d \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

Thus:

$$\tau_s > s_d, \forall s \in S$$

$$\tau^s_{que} + \tau_{pro} + \tau_{\downarrow} > s_d$$

In Probability (11), O_s value is set to either 0 or 1, where 0 refers to no offload is required and 1 refers that the newly arrived service will suffer from latency and will not be able to meet the service deadline s_d. Hence, service offloading is required. Therefore, Probability (11) is the decision maker for COMITMENT model to either allow the fog to process the upcoming service's request or offload the requests to other fog nodes.

Algorithm 1 has been developed to detect the fog nodes that suffer from congestion, and to determine the overload. The goal of this algorithm is to answer the question of *When to offload?* and *What to offload?*. The first part of the algorithm (Procedure 1) determines if the fog node is congested or not. This starts by getting fog queue size and queued services sorted by their types (i.e., heavy-services and light-services) as per lines 1–5. Later, lines 6–8 examine if one or more services in the queue will miss its deadline S_d, or if the service arrival rate λ is bigger than the outcome of the fog node μ. If any of the conditions is applied, a flag will set to indicate that the node is congested as per line 9. The second part of the algorithm (Procedure 2.) determines the overload by computing the number of services causing the congestion as per lines 24–26. The overload O_l will hold the list of services that require offloading to other nodes as per lines 27–28.

In order to balance the services on fog nodes and achieve optimal workload and minimal service delay, we adopt the offloading to the best available node that can deliver the desired services within the scheduled time (i.e., $\tau < d_s$). Therefore, to obtain the best node, which able to handle the overload, we compute the service time τ_s for the services required offloading among all available nodes using Eq. (12), thus, having some constraints on the node that participate in the process to handle the overload such as load limit.

$$min[\tau_s] = \sum_{i=1}^{n} [\tau^i_{que} + \tau^i_{pro} + \tau_{\downarrow}] \tag{12}$$

s.t.

$$f_c^{min} \leqslant f_w \leqslant f_c^{max}$$

$$\sum \lambda_s \leqslant \sum \mu_f$$

$$\tau_s \leqslant s_d, \forall s \in S$$

The best available nodes are those that are able to provide a service with minimal delay. Algorithm 2 finds the best node to handle the overload on the congested node, and than offload the overload from the congested node. In addition, the goal of the algorithm is to answer the question of *Where to offload?*. The first part of the algorithm (Procedure 1.), shows the process of finding the best available node(s) for handling the overload pointed in Algorithm 1. Lines 2–3 of the algorithm initiate the list of active fogs in the domain alongside with the node's capacity and current load (i.e., queue size). The list of available nodes will be refined by removing the nodes that are already busy with other services (i.e., $\lambda_i = \mu_i$) as per lines 6–8. The remaining part of Procedure 1, lines 9–18 will compute the time required for a service to run on each of the available nodes. If the time is within the limit allowed for the service (i.e, before S_d), the system will keep the node in the list and log the expected service time against the node as per lines 9–12. If the τ_s on F_n is less than S_d, then F_n will be removed from list as per lines 13–15. The second part of the algorithm (Procedure 2.), receives the list of best available nodes. If the list is not empty, that means there is at least one fog able to take the overload. However, if there is more than one node in the list, the system will direct the overload to a node that can provide minimal τ_s and has the lowest propagation delay D_p as per lines 21–23.

Algorithm 2: SERVICE OFFLOADING

Input: FogNode (F_n); FogLoad (F_l); OverLoad (O_l).
Parameters : FogCapacity (F_c); Propagation (D_p).
Initialisation: $F_n = \phi$; $F_c = \phi$; $F_l = \phi$; $O_l = \phi$.
Result: Share the Overload with best available node

1 **Procedure 1.** *Determine best available node* **by**
2 $F_L = list\{\phi\}$ ▷ F_L `initiate fog list`
3 $F_L = list[F_n] \longleftarrow getFogNodes(out : (F_n, F_c))$
4 $F_L = sort(F_L,$ by F_c DESC$)$
5 **for** *each* $F_n \in F_L$ **do**
6 **if** $Fn \longleftarrow (F_l \geq Fc_{max})$ **then**
7 $F_L = pop(Fn)$ ▷ `remove busy node`
8 **else**
9 $\tau_s = \sum_{i=1}^{n} [\tau^i_{que} + \tau^i_{pro} + \tau_{\downarrow}]$
10 **if** $(\tau_s < s_d)$ **then**
11 $list.add(F_n, \tau_s)$
12 **continue**
13 **else**
14 $F_L = pop(Fn)$
15 **end**
16 **end**
17 **end**
18 return F_L
19 **End**
20 **Procedure 2.** *Handover the Overload* **by**
21 **if** $F_L \neq \phi$ **then**
22 $F_n = min[F_L(\tau_s, D_p)])$
23 $F^n_l = F_l + O_l$
24 **else**
25 **goto:**1
26 **end**
27 **End**

5. Trust and recommendation model

This section will propose a model that helps fog nodes to make a right decision for selecting the appropriate fog node to collaborate with. Generally, in any network architecture there will be two types of fog nodes, *Trusted* fog nodes and *Malicious* fog nodes. *Malicious* fog node is defined as fog that seeking to breaches any of the security principles and is therefore under an attack. Such nodes exhibit behavior of packet drop, bandwidth consumption so that no other legitimate node can use it, stale packets injected into the network to congest the network and confusion other fogs, and malicious fog can purposely delay services and dispose user's data [56]. While the *Trusted* fog node is defined as nodes which are working with full capacity to satisfy users and services requirements, thus providing high QoS and QoP. However, these nodes are vulnerable to be attacked by a malicious nodes. In this following subsections will propose a trust and recommendation model to help *trusted* fog nodes to identify *malicious* fog nodes and avoid dealing with it.

5.1. Trust – Direct experiences

In the fog-2-fog collaboration model, the direct communication between the fog nodes is evaluated based on the quality-of-service (QoS) and quality-of-protection (QoP) for the services provided by both collaborated fogs, thus, each fog node score the collaboration experiences against the partner fog in terms of the QoS and QoP. The collaboration experiences score logged locally by each fog after every interaction to be used in the future to predict the collaboration success in future interactions. We refer to this as a direct experience as both node can evaluate each other based on their own experiences and not based on recommendation from other fog nodes, thus, this evaluation helps fog to determine the LoT against its partner fog. Moreover, the history of past interactions between nodes is essential to assess node's trustworthiness. Obviously, from the past direct interactions, the nodes that have a positive history should have a positive impact on the LoT score. While the nodes that have a negative history should have a negative impact on the LoT score. Therefore, in our model, it is essential for each fog node in the fog layer to log the score of its Experience Satisfaction (ES) of the direct interactions with other fog nodes. The ES score can be either 1 or 0, where 1 is indication of trust/satisfied and 0 is indication of distrust/unsatisfied, thus, this score will be given upon meeting the QoS (e.g., low latency) and QoP (e.g., data protection). In our model, we adopt a Bayesian network to evaluate the direct satisfaction experiences based on direct interactions between fogs nodes. Bayesian has been adopted because it has proven results with peer-2-peer network modeling in terms of trust/reputation and in line with [15,37]. The satisfaction experience parameter of f_a toward f_b is represented by ES score $ES_{a,b}$. Thus, the value of ES is a binary value, either is set to 1 for satisfied experience or to 0 for unsatisfied experience. The ES is distributed between satisfied and unsatisfied experiences (i.e., distributing of 1s and 0s) according to Bernoulli trial distribution, thus, we refer to the probability of satisfied experience by a positive experience parameter $p_{a,b}$ according to Beta distribution, thus, the posterior $Pr(p_{a,b}|S_{a,b})$. The direct trust $\tau^d_{a,b}$ of f_a toward f_b is computed as per Eq. (13).

$$\tau^d_{a,b} = \frac{\alpha_{f_a,f_b}}{\alpha_{f_a,f_b} + \beta_{f_a,f_b}} \in [0-1] \tag{13}$$

where the α_{f_a,f_b} and β_{f_a,f_b} refer to the parameters of Beta distribution, thus, α_{f_a,f_b} log the satisfied experience, while β_{f_a,f_b} log the unsatisfied experience. Both α_{f_a,f_b} and β_{f_a,f_b} are computed and updated after every direct interaction between f_a and f_b with a consideration for the trust decay as per Eqs. (14) and (15).

$$\alpha'_{f_a,f_b} = e^{d\Delta t}.\alpha_{f_a,f_b} + ES_{a,b} \tag{14}$$

$$\beta'_{f_a,f_b} = e^{d\Delta t}.\beta_{f_a,f_b} + 1 - ES_{a,b} \tag{15}$$

where α'_{f_a,f_b} and β'_{f_a,f_b} refer to the new score, while α_{f_a,f_b} and β_{f_a,f_b} refer to old score. The $e^{d\Delta t}$ refers to the exponential decay, thus, d is the decay factor and the Δt is the trust update interval. It is worth noting that d is a small value to represent the trust decay over time.

In order to make the trusted network reliable and scalable, the fog should not burden its resources with redundant trust scores and only logs the most recent ES score along with the number of interactions between the fog nodes. Therefore, the ES score is an accumulative score and it is periodically updated and logged in a ES_{score} as a mapping function as per Eq. (16). Where $f_a \longrightarrow f_b$ map the interaction from fog_a to fog_b and n_{int} refers to the number of direct interactions between the two fog nodes.

$$ES_{score}(a,b) =< f_a \longrightarrow f_b, n(int), \alpha_{f_a,f_b}, \beta_{f_a,f_b}, LoT > \tag{16}$$

It is worth noting that in previous researches the initial value of α and β is set to *null* or 1 since there is no previous knowledge and no prior interactions between the two fog nodes. In our model, we adopt a recommendation based approach to obtain the initial value of α and β through seeking a recommendation from a neighboring node(s) that has the same set of requirements (i.e., RoP) for the QoP, this is discussed in Section 5.2. However, if no initial value can be obtained from either the direct experience or the recommendations, then the initial value of α and β is set to 1 since no prior knowledge is available and in line with [15].

5.2. Recommendations — Indirect experiences

In this paper, we refer to the recommendations as an indirect trust experience as fog node cannot evaluate its partner trustworthiness directly based on its own experiences as there is no prior knowledge (i.e., no direct interactions in the past) but based on a recommendations from neighboring fog nodes. In the recommendations model we adopt the design concept of distributed Collaborating Filtering (CF) [15,74] to obtain trustworthiness score from neighboring fog nodes that sharing similar interests [15]. Therefore, CF classifies the received recommendations based on recommender party into two types:-

- Recommendations from *trusted fog nodes*: this includes recommendations provided from a trusted fog node based on our trust model in Section 5.1. The recommender of this type of recommendations is evaluated in terms of LoT from the interactions with the desired fog node, thus, it has a satisfactory experience score obtained from positive/sauces past interactions. With this type of recommender its sufficient to check the LoT without checking the SSLA and its RoP as it should be already met, prior to previous interactions. This recommenders are likely have a general (i.e., non subjective) trust score toward the desired fog node.
- Recommendations from *community fog nodes*: this recommendations are provided from fog nodes that have the same service's interests from the desired fog node. It is not necessarily for the recommender of this type of recommendations to have a LoT or previous interactions. However, the recommender should share the same service's interests with regard the SSLA toward the required service and provided by the desired fog node, i.e., fogs that have the same sentiment toward the desired fog node. This recommenders are likely

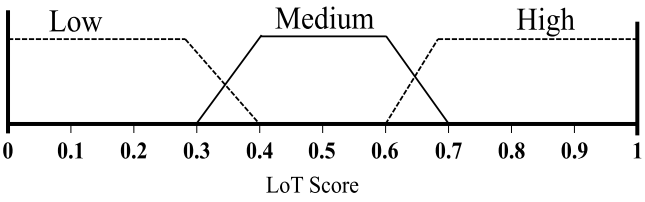


Fig. 3. Level of Trust (LOT) according to fuzzy logic.

have a similar subjective trust score toward the desired fog node.

It is worth noting that in order to consider the recommendations provided from the two types of recommenders, *trusted fog nodes* and *community fog nodes*, we first evaluate the relationship between the trustor fog and the recommender fog to avoid intruder neighboring fog nodes. Evaluating the relationship will be based on the type of the recommender, if a trusted fog has a satisfactory LoT score, then we can consider its recommendation, otherwise, ignoring the recommendation. Whereas, if the recommendation is from a community fog node, we first check if the recommender fog meets the SSLA requirements (shared by trusty fog node) before we can consider its recommendation, thus, it will only be considered if it has a similar SSLA standards (e.g., same QoS and QoP experiences). Moreover, the trustor fog will weigh the recommendations provided by the recommenders toward the trustee to get the overall trustworthiness as per Eq. (17).

$$r(a,b) = \sum_{rp \in R} [w_{rp} \times r_{f_a,f_b}], R \in [m,c] \tag{17}$$

Where w_m and w_c are the weights of recommendations obtained from trusted fog nodes and community fog nodes, respectively. Thus, $w_m + w_c = 1$ and $0 \leq w_m, w_c \leq 1$. The r_{f_a,f_b} denotes to the recommendation of fog_a toward fog_b. Each fog node can send a recommendations request to its neighboring fog nodes and upon receiving the response (recommendation score), the fog weights the recommendations from all recommenders and calculates the over all indirect trust using Eq. (18).

$$\tau^r_{a,b} = \frac{r_{f_a,f_b}}{\sum_{i=0}^{nr} r_{f_a,f_b}(a,b)} \tag{18}$$

It is worth noting that the outcome trust score $\tau^r_{a,b}$ from the obtained recommendations from recommenders is a value between 0 to 1, therefore, we apply the fuzzy logic function to the determine the level of trust as per Fig. 3, where 1 is indicator of absolute trust and 0 is indicator of utter distrust.

Algorithm 3 elaborates the process of seeking a recommendations from a neighboring fog nodes. Considering a scenario where fog f_a wish to interact with fog f_b and it has no previous interactions history, f_a go through the Procedures 1 and 2. Procedures 1: f_a will try to seek recommendations from neighboring fog nodes to get the trustworthiness of f_b, so that, f_a asks fog nodes f_c, f_d, f_e, for example, for recommendations on the trustworthiness of f_b. The recommendation requests send only to a trusted fog nodes, i.e., trusted by f_a as per lines 3–6. The recommendation messages request will be sent to the trusted fog nodes in the format of $m_r = \{f_b, SSLA\}$ as per lines 7–10, where the first part, in this case (f_b), is the desired fog node for checking its trustworthiness. While the other part is the Secure Service Level Agreement (SSLA), which is set of requirements to be used in the evaluation of the trust score of f_b. It is worth noting that the SSLA parameters are set according to f_a QoP based on the RoP parameters presented in Section 2.3. The recommenders, i.e., $f_c, f_d, f_e..f_n$ fog nodes, will evaluate the trustworthiness toward the desired fog

Algorithm 3: Proposed Recommendation Model

Input: $FogNode_a$ (f_a); $FogNode_b$ (f_b); *SSLA*
Parameters : *trustScore* ($\tau^r_{a,b}$); *FogList* (F_L); *recommendation* (r)
Initialisation: $\tau^r_{a,b} = \phi$; $F_L = list\{\phi\}$
Result: LoT from neighboring fogs ($\tau^r_{a,b}$) for f_a toward f_b

1 **Procedure 1:** *get trusted fog for recommendation* **by**
2 $F_L = list[F_n] \longleftarrow getNeighbourFogs(out : (F_n, LoT))$;
3 $F_L = sort(F_L,$ by LoT DESC) ;
4 **for** *each* $F_n \in F_L$ **do**
5 **if** $F_n \longrightarrow$ *untrusted by* F_a **then**
6 $F_L = pop(F_n)$; ▷ `remove untrusted node`
7 **else**
8 $F_n = m_r\{f_b, SSLA\}$;
9 $F_L = update(F_n, r, out : F_L)$; ▷ `update list adding r`
10 **end**
11 **end**
12 return F_L ;
13 **End**
14 **Procedure 2:** *Compute trustworthiness* **by**
15 $F_L = list[F_n, r]$; ▷ `the new fog list with r`
16 $F_L = sort(F_L,$ by LoT DESC) ;
17 **for** *each* $F_n \in F_L$ **do**
18 $r(f_n, f_b) = \sum_{rp \in R}[w_{rp} \times r_{f_n,f_b}], R \in [m,c]$
19 **end**
20 $\tau^r_{a,b} = \frac{r(a,b)}{\sum_{i=0}^{nr} r(a,b)}$; ▷ `compute the overall trustworthiness`
21 return $\tau^r_{a,b}$;
22 **End**

(i.e. f_b) based on the (SSLA) requirements from past interactions experiences, using the proposed trust model in Section 5.1. Then, the trust score returned to the trustee fog node as per line 12. Procedures 2: f_a estimates the trustworthiness of f_b according to the gained recommendations, thus, the fog f_a will decide whether f_b is trusty and can deliver the desired service. Hence, the trustworthiness estimation will be computed using Eq. (18) after filtering the recommendation by the weight recommender according to Eqs. (17), as per lines 14–22.

5.3. Reputation assessment

The reputation assessment process will provide the output of the final LoT score which will be used to identify the trustworthiness of a particular fog. In this process, both trust (i.e., direct experiences) and recommendations (i.e., indirect experiences) will be involved to get the LoT score. However, the trust score and recommendations score will be considered in different weight, score from direct experiences will always have a higher weight due to the level of satisfactory/unsatisfactory experience gained from previous collaborations. Hence, the score of recommendations will be only considered with higher weight when there is no enough direct interactions between two nodes. The LoT function $LoT(f_a, f_b)$ in Eq. (19) computes the Lot score which will use by fog to make a decision whether to collaborate or not.

$$LoT(f_a, f_b) = \begin{bmatrix} \gamma \\ \delta \end{bmatrix} [\tau^r_{a,b}, \tau^d_{a,b}] = \gamma.\tau^r_{a,b} + \delta.\tau^d_{a,b} \tag{19}$$

where δ and γ represent the corresponding weights of the direct (τ^d) and indirect (τ^r) trust score respectively. The score of LoT will be an indication for the level of trust or distrust

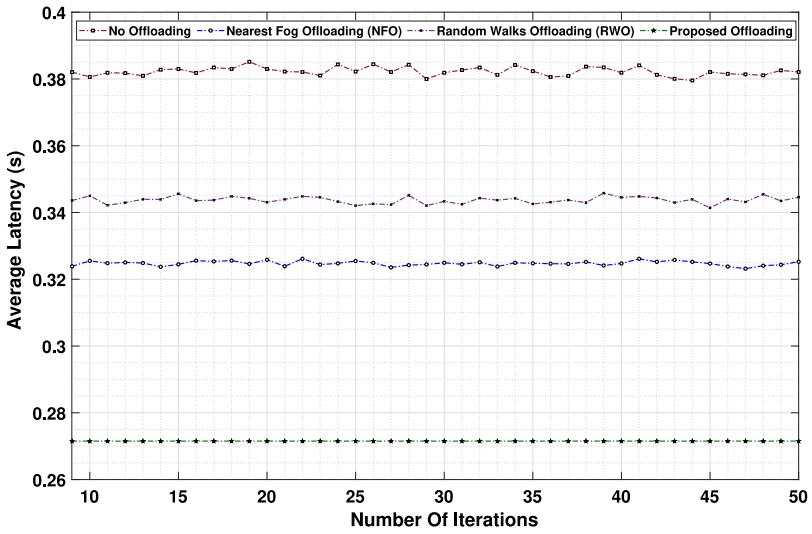


Fig. 4. Average latency against two benchmark algorithms (RWO and NFO) and based on mixed type of packets

Table 3
Simulation settings.

Parameter	Value
Operating system	Win 8.1
Simulation environment	Matlab 2018b
Number of fog nodes	15
Fog CPU	[0.2–1.5] GHz
Network topology	mesh
Number of service's requests	10^5
Package size	[0.1–80] KB
Bandwidth	up-to 54Mbps

between two fog nodes. For example, the LoT score provided by the function $LoT(f_a, f_b) \in [0-1]$ refers to the level of f_a trust or distrust toward f_b according to the previous direct/indirect experiences with f_b. The LoT score will be used based on a fuzzy logic as per Fig. 3. The fuzzy logic function classify the LoT score into three main parts, Low, Medium and High to represent the trustworthiness between the two nodes, where 1 is indicator of absolute trust and 0 is indicator of utter distrust.

6. Experimental evaluation

In this section, we evaluate the proposed COMITMENT model for a secure $\mathcal{F}og$-2-$\mathcal{F}og$ collaboration, which aims at providing secure offloading for fog service's requests. The proposed COMITMENT model has been simulated using MATLAB (2018b) on a Lenovo ideaPad with Intel Core *i*5 processor and 8 GB of RAM. Simulation settings are presented in the following subsection (Section 6.1), followed by a discussion on the simulation results.

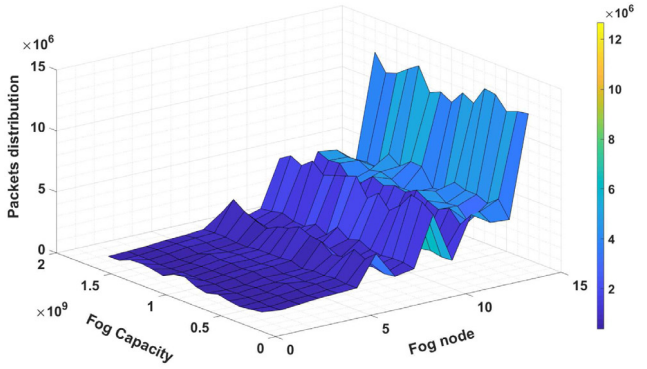
6.1. Simulation settings

The system characteristics adopted during the simulations are presented in Table 3. We specify the simulation settings in terms of network topology, propagation and transmission delay, link bandwidth and fogs capabilities.

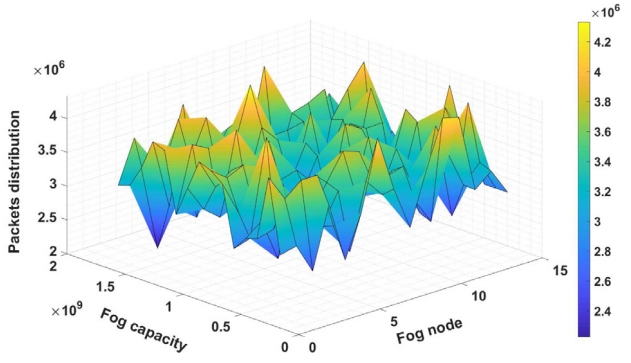
- Network Topology modeled as an indirect graph, represents fogs as a mesh network. The simulation has 15 (i.e., $f_n = 15$) fog nodes connected together through internal communication link. The links between nodes are weighted based on the propagation delay (D_p) among nodes, for instance, if D_p between fog_1 and fog_2 is four second, then it represented like ($f_1 \underset{4}{\longleftrightarrow} f_2$). It is worth nothing that the services arrived to the fog layer are assigned to a fog based on the smallest distance (i.e., smallest D_p).
- Network Bandwidth: the link bandwidth depends on the type of service's requiest, hence, *heavy-requiest* will require more bandwidth from *light-requiest*. For light-requiest (e.g., data packets from sensors) the communication bandwidth used with a transmission rate of 250 Kbps [55]. While, the heavy-packets (e.g., data packets from camera) the communication bandwidth used with a transmission rate of 54 Mbps [13]. The transmission rate between the fog nodes is expected to be higher $\simeq$ 100 Mbps [77].
- Transmission delay (D_t) for a packet is obtained from packet size l_p alongside with the associate upload bandwidth b ↑ Therefore, we impose the average packet size that will vary according to the type of packet (i.e., heavy and light packets). The average packet size for light-packets is 0.1 KB, while the average packet size for heavy-packets is 80KB [77].
- Propagation delay (D_p) for a packet is based on the round trip time (i.e., $\tau_{\downarrow\uparrow}$) same as in [52,77] by $\tau_{\downarrow\uparrow} = 0.03 \times l_d + 5$, where l_d is the distance with unit km, and the $\tau_{\downarrow\uparrow}$ time unit is ms.
- Fog nodes capabilities: fogs are simulated with different capabilities, hence, the service rate (μ) will vary from one node to another. The capabilities of fogs will significantly effect the processing ability (i.e., performance) of the fog. The capability of fog is determined by the CPU frequency, therefore, nodes vary in CPU frequency having them in the range of 0.2 GHz to 1.5GHz [22].
- Fogs interactions: as we adopted Bayesian network to evaluate the satisfaction experience among collaborated fog nodes, each fog develops a naive Bayesian network model for all other fog nodes that it has interacted with. This achieves by locally storing the binary values of ES score, which is either *satisfying* and *unsatisfying* interaction, denoted by 1 and 0, respectively. Then, computing the LoT score based on all the past interactions/collaborations between nodes and which will be used to identify the trustworthiness of partner fog node.

6.2. Results and discussion

This section shows the numerical results of the experimentations on the proposed model to validate the accuracy of our secure offloading model based on the COMITMENT.



(a) Average packets distribution according to POA



(b) Average packets distribution according to RWO and NFO

Fig. 5. Packets distribution.

We first evaluate the performance of the Proposed Offloading Algorithm (POA) against two benchmark algorithms: (i) Random Walks Offloading (RWO) [27,79]. (ii) Nearest Fog Offloading (NFO) [12,71]. Fig. 4 demonstrates the performance based on the average response time to all received service's requests considering different packets type (i.e., heavy-packets and light-packets), however, the random number of heavy or light packets is fixed through out the experiment to ensure consistency in terms of load utilization against the offloading algorithms. During the simulation of this experiment, we set the fogs different capabilities, hence, nodes vary in their service rate μ. Thus, the capability of fog is based on CPU frequency with a minimum of 300×10^6 Hertz, incremented by 100 Hertz until it gets to maximum CPU capability of 17×10^8 Hertz. In addition, service arrival rate $\lambda = 2 \times 10^2$ packets per second as in [64], and λ is fixed during the experiment to ensure all offloading algorithms have the same traffic arrival rate. Fig. 4 shows the outcome of this experiment, thus the vertical line represents the average latency per algorithm to process service's requests, and the horizontal line is the number of iterations carried out to insure that the obtained results are consistent and not due to chance. It is clear that POA has the lowest processing latency among other algorithms through all iterations. The highest processing time goes for No Offloading Consideration (NOC) as it does not consider the offloading when a node becomes congested. Hence, its end-up having small node capacity with large queue size (i.e., $\mu_i < \lambda_i$), and large node capacity with low queue size. The performance of RWO and NFO are better than NOC but still hither than POA.

The following experiment was conducted based on packets distribution over the 3 offloading algorithms (i.e., POA, RWO and NFO) on fog node. The experiment settings are similar to our previous experiment, except having fixed packet type (i.e., all heavy or light packets) to ensure consistency. Fig. 5 shows packet's distribution, Fig. 5(a) shows packet's distribution according to POA. While, Fig. 5(b) shows packet's distribution according to RWO and NFO. It is clear that POA have more sustainable packets distribution compared to RWO and NFO. Thus POA distributes the packets with respect to fogs capabilities. While, the other methods were relatively blind as they have not considered the current load (f_w) of fog.

Fig. 6 shows the results of malicious event (i.e., malicious collaboration requests) detection according to the LoT score. In this figure, the number of service's request set to 1K and we had two iterations with this experiment; the first iteration is used to make enough collaborations between the fog nodes, so that they have a precise LoT score against each other. The second iteration is to observe the interactions and flag any malicious events. The collaboration requests in Fig. 6 are grouped according to request's type; *secure*, *malicious* and *anonymous* requests. The collaboration requests are grouped based on the LoT score produced by the LoT

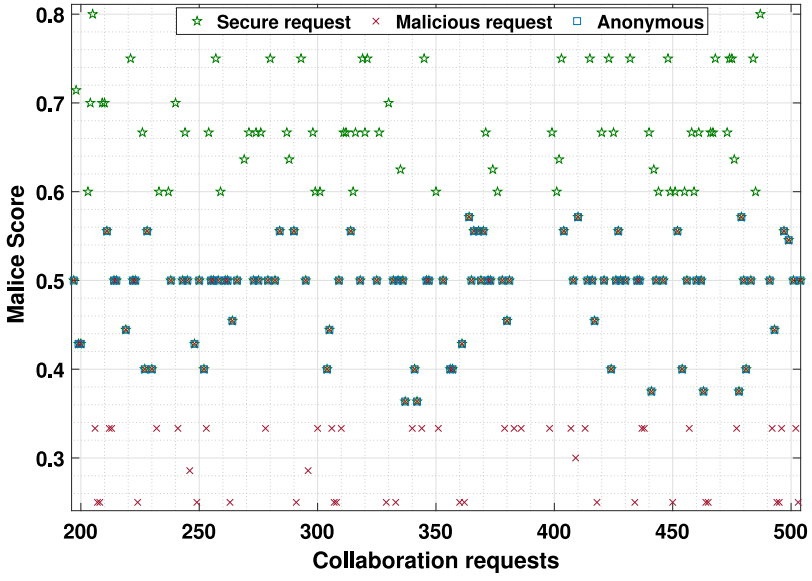


Fig. 6. Collaboration requests according to their type; *secure*, *malicious* and *anonymous* requests based on the LoT score.

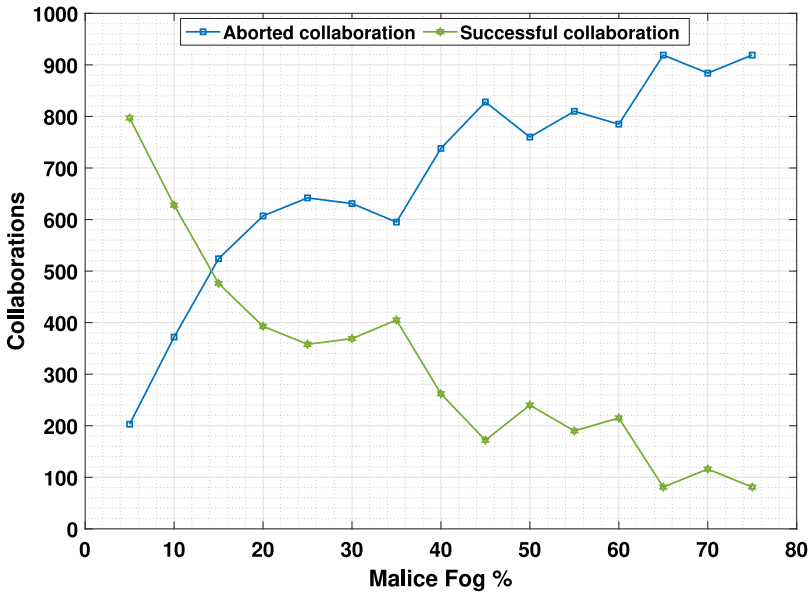


Fig. 7. Average number of *successful* and *aborted* collaborations according to the percentage of malicious fogs.

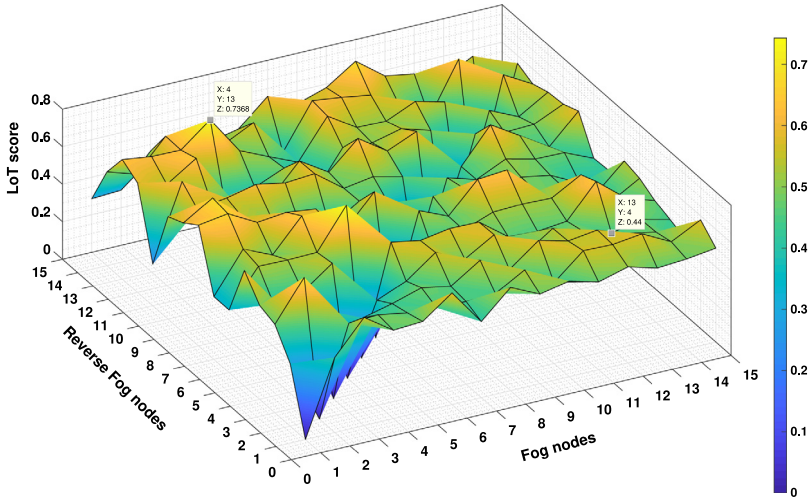


Fig. 8. LoT score for the 15 participated fogs against each other proven that LoT is asymmetric.

function and according to the fuzzy logic in Fig. 3. It is worth noting that the anonymous collaboration requests are down to the fact that either there is not enough LoT score gained from the past collaborations, or there the gained LoT score on the borderline of the trustworthiness of a particular fog.

The different types of collaboration requests (i.e., *secure*, *malicious* and *anonymous* requests) will control the decision of whether a collaboration can be accepted or rejected between two fog nodes. Fig. 7 shows the average number of *successful* and *aborted* collaborations according to the percentage of malicious fog nodes within the network. In this experiment, the initial percentage of malicious fog in the network is 5%, then it increases by 5% up until we have 75% of the fog nodes are malicious. Through out the experiment, we observe the average number of *successful* and *aborted* collaborations, it is clear that with the increase of the malicious fogs in the network; the number of *successful* collaborations will be reduced and the number of *aborted* collaborations will be increased as per Fig. 7.

The next experiment is about fog's trustworthiness policy, having the LoT score asymmetric and not transitive. Thus, each fog has its own LoT score that defines its QoP, hence, if fog_a finds fog_b is trustworthy based on fog_a LoT score that meets the its RoP toward fog_b, it is not necessarily that fog_b finds fog_a is trustworthy. Fig. 8 shows the corresponding 3-dimensional view of the LoT score for the 15 participated fogs against each other. It is clear that the fogs have different LoT score against each other, for example, the LoT score from fog_4 to fog_{13} is 0.7, while the LoT score from fog_{13} to fog_4 is 0.4 as shown in the highlighted points in Fig. 8. Similarly, the LoT is not transitive, for example, in Fig. 9, fog_1 trust fog_5 and fog_b trust fog_2, while fog_1 founds fog_2 is not trustworthiness.

7. Conclusion

This paper presented COMITMENT: a fog computing trust management approach. We, first, introduced the fog-based systems architecture and associated threats, attacks, and security requirements. Then, we discussed COMITMENT procedures and processes in terms of the performance and interactions among fog nodes. In addition, we defined the problem and formulated the proposed model of trust recommendation using the direct and indirect experiences. Finally, we performed a series of experiments

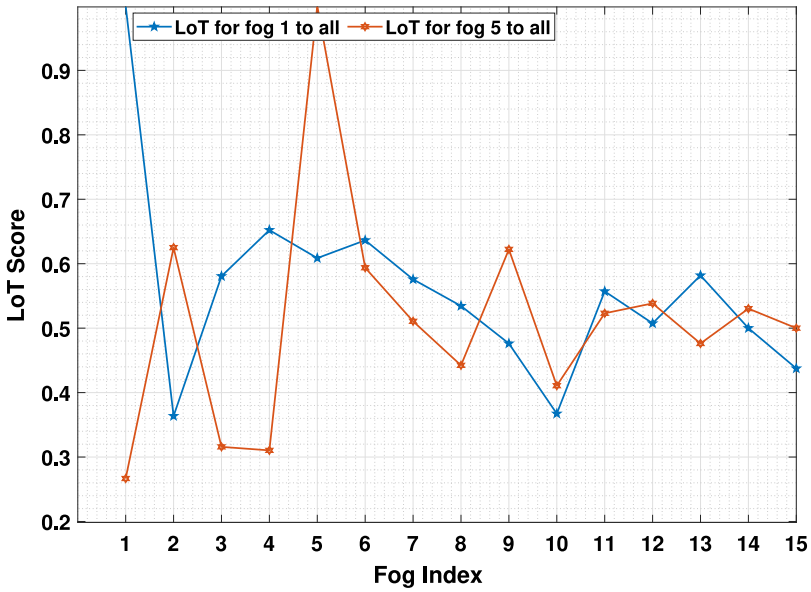


Fig. 9. Lot score for fog_1 and fog_5 proven that LoT is not transitive.

to verify the validity and performance of the proposed approach in which COMITMENT outperformed the competitive benchmark algorithms, namely Random Walks Offloading (RWO) and Nearest Fog Offloading (NFO). In our future work, we plan to extend the simulation by evaluating the energy consumption of fog nodes during the collaboration and offloading processes.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.10.006.

Acknowledgments

This research is partially funded by Engineering and Physical Sciences Research Council, United Kingdom (EPSRC – EP/R033293 /1) titled "PACE: Privacy-Aware Cloud Ecosystems". Also, the work is supported by the National Natural Science Foundation of China under Grant 61836001, the National Key Research and Development Program of China under Grant 2018YFB1003700, and the Beijing Institute of Technology Research Fund Program for Young Scholars, China.

References

[1] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities, Future Gener. Comput. Syst. 87 (2018) 278–289.

[2] N. Abbas, M. Asim, N. Tariq, T. Baker, S. Abbas, A mechanism for securing IoT-enabled applications at the fog layer, J. Sens. Actuator Netw. 8 (1) (2019) 16.

[3] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, O. Alfandi, Fog computing framework for Internet of Things applications, in: 2018 11th International Conference on Developments in eSystems Engineering, DeSE, 2018, pp. 71–77, http://dx.doi.org/10.1109/DeSE.2018.00017.

[4] M. Al-khafajiy, T. Baker, C. Chalmers, M. Asim, H. Kolivand, M. Fahim, A. Waraich, Remote health monitoring of elderly through wearable sensors, Multimedia Tools Appl. (2019).

[5] M. Al-khafajiy, T. Baker, H.A. -Libawy, Z. Maamar, M. Aloqaily, Y. Jararweh, Improving fog computing performance via fog-2-fog collaboration, Future Gener. Comput. Syst. 100 (2019) 266–280.

[6] M. Al-khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, A. Hussain, Iot-fog optimal workload via fog offloading, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, IEEE, 2018, pp. 359–364.

[7] M. Al-khafajiy, L. Webster, T. Baker, A. Waraich, Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare, in: Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, ACM, 2018, p. 9.

[8] A. Alrawais, A. Alhothaily, C. Hu, X. Cheng, Fog computing for the Internet of Things: Security and privacy issues, IEEE Internet Comput. 1 (21) (2017) 2.

[9] I. Azimi, A. Anzanpour, A.M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, N. Dutt, HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT, ACM Trans. Embed. Comput. Syst. 16 (5s) (2017) 174:1–174:20.

[10] T. Baker, M. Asim, Á. MacDermott, F. Iqbal, F. Kamoun, B. Shah, O. Alfandi, M. Hammoudeh, A secure fog-based platform for SCADA-based IoT critical infrastructure, Softw. - Pract. Exp. (2019).

[11] K. Bhardwaj, J.C. Miranda, A. Gavrilovska, Towards IoT-DDoS prevention using edge computing, in: USENIX Workshop on Hot Topics in Edge Computing, HotEdge 18, USENIX Association, Boston, MA, 2018, https://www.usenix.org/conference/hotedge18/presentation/bhardwaj.

[12] A. Bozorgchenani, D. Tarchi, C.G. E., An energy and delay-efficient partial offloading technique for fog computing architectures, in: GLOBECOM 2017-2017 IEEE Global Communications Conference, IEEE, 2017, pp. 1–6.

[13] M.J. Canet, V. Almenar, J. Marin-Roig, J. Valls, Time synchronization for the IEEE 802.11 a/g WLAN standard, in: 2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, IEEE, 2007, pp. 1–5.

[14] R. Chen, F. Bao, M. Chang, J.-H. Cho, Dynamic trust management for delay tolerant networks and its application to secure routing, IEEE Trans. Parallel Distrib. Syst. 25 (5) (2013) 1200–1210.

[15] R. Chen, J. Guo, F. Bao, Trust management for SOA-based IoT and its application to service composition, IEEE Trans. Serv. Comput. 30 (9) (2014) 3.

[16] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE J. Sel. Areas Commun. 36 (3) (2018) 587–597.

[17] L. Chen, J. Xu, Socially trusted collaborative edge computing in ultra dense networks, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, SEC '17, ACM, New York, NY, USA, 2017, pp. 9:1–9:11.

[18] M. Chiang, T. Zhang, Fog and IoT: An overview of research opportunities, IEEE Internet Things J. 3 (6) (2016) 854–864.

[19] J.-H. Cho, A. Swami, R. Chen, A survey on trust management for mobile ad hoc networks, IEEE Commun. Surv. Tutor. 13 (4) (2010) 562–583.

[20] R. Deng, R. Lu, C. Lai, T.H. Luan, H. Liang, Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption, IEEE Internet Things J. 3 (6) (2016) 1171–1181.

[21] Z. Deng, M. Wang, L. Wang, X. Huang, W. Han, J. Chu, A.Y. Zomaya, An efficient indexing approach for continuous spatial approximate keyword queries over geo-textual streaming data, ISPRS Int. J. Geo-Inf. 8 (2) (2019) 57.

[22] T.Q. Dinh, J. Tang, Q.D. La, Q.T. Q., Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Trans. Commun. 65 (8) (2017) 3571–3584.

[23] A.M. Elmisery, S. Rho, D. Botvich, A fog based middleware for automated compliance with OECD privacy principles in internet of healthcare things, IEEE Access 4 (2016) 8418–8441.

[24] Q. Fan, N. Ansari, Towards workload balancing in fog computing empowered IoT, IEEE Trans. Netw. Sci. Eng. (2018).

[25] Q. Fan, N. Ansari, Towards workload balancing in fog computing empowered IoT, IEEE Trans. Netw. Sci. Eng. (2018).

[26] J. Fan, J. Yan, Y. Ma, L. Wang, Big data integration in remote sensing across a distributed metadata-based spatial infrastructure, Remote Sens. 10 (1) (2017) 7.

[27] C. Fricker, F. Guillemin, P. Robert, G. Thompson, Analysis of an offloading scheme for data centers in the framework of fog computing, ACM Trans. Model. Perform. Eval. Comput. Syst. 1 (4) (2016) 16.

[28] L. Galluccio, S. Milardo, G. Morabito, P.S.S. wise: Design, Prototyping and experimentation of a stateful SDN solution for wireless sensor networks, in: 2015 IEEE Conference on Computer Communications, INFOCOM, IEEE, 2015, pp. 513–521.

[29] N.K. Giang, M. Blackstock, R. Lea, V.C. Leung, Developing IoT applications in the fog: A distributed dataflow approach, in: 2015 5th International Conference on the Internet of Things, IOT, IEEE, 2015, pp. 155–162.

[30] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, IFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.

[31] T. He, E.N. Ciftcioglu, S. Wang, K.S. Chan, Location privacy in mobile edge clouds: A chaff-based approach, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2625–2636.

[32] M. Henze, R. Hummen, R. Matzutt, K. Wehrle, A trust point-based security architecture for sensor data in the cloud, in: Trusted Cloud Computing, 2014, pp. 77–106.

[33] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: A survey, 2018, arXiv preprint arXiv:1810.00305.

[34] P. Hu, H. Ning, T. Qiu, H. Song, Y. Wang, X. Yao, Security and privacy preservation scheme of face identification and resolution framework using fog computing in Internet of Things, IEEE Internet Things J. 4 (5) (2017) 1143–1155.

[35] K. Hwang, S. Kulkareni, Y. Hu, Cloud security with virtualized defense and reputation-based trust mangement, in: 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, IEEE, 2009, pp. 717–722.

[36] J. Jiang, G. Han, F. Wang, L. Shu, M. Guizani, An efficient distributed trust model for wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 26 (5) (2015) 1228–1237.

[37] A. Josang, R. Ismail, The beta reputation system, in: Proceedings of the 15th Bled Electronic Commerce Conference, vol. 5, 2002, pp. 2502–2511.

[38] W.-S. Kim, S.-H. Chung, User-participatory fog computing architecture and its management schemes for improving feasibility, IEEE Access 6 (2018) 20262–20278.

[39] Q. Li, A. Malip, K.M. Martin, S.-L. Ng, J. Zhang, A reputation-based announcement scheme for VANETs, IEEE Trans. Veh. Technol. 61 (9) (2012) 4095–4108.

[40] P. Liu, L. Hartung, S. Banerjee, Lightweight multitenancy at the network's extreme edge, Computer 50 (10) (2017) 50–57.

[41] Z. Maamar, T. Baker, N. Faci, M. Al-Khafajiy, E. Ugljanin, Y. Atif, M. Sellami, Weaving cognition into the internet-of-things: Application to water leaks, Cogn. Syst. Res. 56 (2019) 233–245.

[42] E.K. Markakis, K. Karras, A. Sideris, G. Alexiou, E. Pallis, Computing, caching, and communication at the edge: The cornerstone for building a versatile 5G ecosystem, IEEE Commun. Mag. 55 (11) (2017) 152–157.

[43] W. Masri, I. Al Ridhawi, N. Mostafa, P. Pourghomi, Minimizing delay in IoT systems through collaborative fog-to-fog (F2F) communication, in: 2017 Ninth International Conference on Ubiquitous and Future Networks, ICUFN, IEEE, 2017, pp. 1005–1010.

[44] R.K. Naha, S. Garg, D. Georgakopoulos, P.P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog computing: Survey of trends, architectures, requirements, and research directions, IEEE Access 6 (2018) 47980–48009.

[45] J. Ni, K. Zhang, X. Lin, X.S. Shen, Securing fog computing for Internet of Things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2017) 601–628.

[46] J. Ni, K. Zhang, X. Lin, S. Shen, Securing fog computing for Internet of Things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2018) 601–628.

[47] J. Ni, K. Zhang, X. Lin, X.S. Shen, Securing fog computing for Internet of Things applications: Challenges and solutions, IEEE Commun. Surv. Tutor. 20 (1) (2018) 601–628.

[48] C. Pahl, N. El Ioini, S. Helmer, B. Lee, An architecture pattern for trusted orchestration in IoT edge clouds, in: 2018 Third International Conference on Fog and Mobile Edge Computing, FMEC, IEEE, 2018, pp. 63–70.

[49] D. Puthal, S.P. Mohanty, S.A. Bhavake, G. Morgan, R. Ranjan, Fog computing security challenges and future directions [energy and security], IEEE Consum. Electron. Mag. 8 (3) (2019) 92–96.

[50] D. Puthal, S. Nepal, R. Ranjan, J. Chen, Threats to networking cloud and edge datacenters in the Internet of Things, IEEE Cloud Comput. 3 (3) (2016) 64–71.

[51] D. Puthal, R. Ranjan, A. Nanda, P. Nanda, P.P. Jayaraman, A.Y. Zomaya, Secure authentication and load balancing of distributed edge datacenters, J. Parallel Distrib. Comput. 124 (2019) 60–69.

[52] A. Qureshi, Power-Demand Routing in Massive Geo-Distributed Systems (Doctoral dissertation), Massachusetts Institute of Technology.

[53] J. Ren, Y. Zhang, K. Zhang, S.X.S. Sacrm, Social aware crowdsourcing with reputation management in mobile sensing, Comput. Commun. 65 (2015) 55–65.

[54] R. Roman, J. Lopez, M. Manbo, Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges, Future Gener. Comput. Syst. 78 (2018) 680–698.

[55] Y. Sahni, J. Cao, S. Zhang, A. Yang L. Edge Mesh:, New paradigm to enable distributed intelligence in Internet of Things, IEEE Access 5 (2017) 16441–16458.

[56] R. Saini, M. Khari, Defining malicious behavior of a node and its defensive methods in ad hoc network, Int. J. Compt. Appl. 20 (4) (2011) 18–21.

[57] M.A. Salahuddin, A. Al-Fuqaha, M. Guizani, Reinforcement learning for resource provisioning in the vehicular cloud, IEEE Wirel. Commun. 23 (4) (2016) 128–135.

[58] S. Sarkar, S. Chatterjee, S. Misra, Assessment of the suitability of fog computing in the context of Internet of Things, IEEE Trans. Cloud Comput. 6 (1) (2015) 46–59.

[59] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, R. Hao, Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium, J. Netw. Comput. Appl. 82 (2017) 56–64.

[60] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, P. Leitner, Optimized IoT service placement in the fog, Serv. Orient. Comput. Appl. 11 (4) (2017) 427–443.

[61] S.A. Soleymani, A.H. Abdullah, M. Zareei, M.H. Anisi, C. Vargas-Rosales, M.K. Khan, S. Goudarzi, A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing, IEEE Access 5 (2017) 15619–15629.

[62] C. Vallati, A. Virdis, E. Mingozzi, G. Stea, Exploiting LTE D2D communications in M2M fog platforms: Deployment and practical issues, in: 2015 IEEE 2nd World Forum on Internet of Things, WF-IoT, 2015, pp. 585–590.

[63] L. Wang, Y. Ma, J. Yan, V. Chang, A.Y. Zomaya, PipsCloud: High performance cloud computing for remote sensing big data management and processing, Future Gener. Comput. Syst. 78 (2018) 353–368.

[64] X. Wang, Z. Ning, L. Wang, Offloading in internet of vehicles: A fog-enabled real-time traffic management system, IEEE Trans. Ind. Inf. 14 (10) (2018) 4568–4578.

[65] N. Wang, B. Varghese, M. Matthaiou, D.S. Nikolopoulos, ENORM: A framework for Edge NOde Resource Management, IEEE Trans. Serv. Comput. (2018) 1.

[66] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, L.T. Yang, Internet traffic classification using constrained clustering, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 2932–2943.

[67] X. Wang, L.T. Yang, X. Xie, J. Jin, M.J. Deen, A cloud-edge computing framework for cyber-physical-social services, IEEE Commun. Mag. 55 (11) (2017) 80–85.

[68] T. Wang, G. Zhang, M.D.Z.A. Bhuiyan, A. Liu, W. Jia, M. Xie, A novel trust mechanism based on fog computing in sensor–cloud system, Future Gener. Comput. Syst. (2018).

[69] T. Wang, G. Zhang, A. Liu, M.Z.A. Bhuiyan, Q. Jin, A secure iot service architecture with an efficient balance dynamics based on cloud and edge computing, IEEE Internet of Things J. (2018).

[70] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, M. Rovatsos, Fog orchestration for Internet of Things services, IEEE Internet Comput. 21 (2) (2017) 16–24.

[71] Y. Xiao, M. Krunz, Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation, in: I.I. 2017-ieee (Ed.), Conference on Computer Communications, IEEE, 2017, pp. 1–9.

[72] W. Xiong, H. Hu, N. Xiong, L.T. Yang, W.-C. Peng, X. Wang, Y. Qu, Anomaly secure detection methods by analyzing dynamic characteristics of the network traffic in cloud communications, Inform. Sci. 258 (2014) 403–415.

[73] J. Yan, Y. Ma, L. Wang, K.-K.R. Choo, W. Jie, A cloud-based remote sensing data production system, Future Gener. Comput. Syst. 86 (2018) 1154–1166.
[74] X. Yang, Y. Guo, Y. Liu, H. Steck, A survey of collaborative filtering based social recommender systems, Comput. Commun. 41 (2014) 1–10.
[75] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, 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, IEEE, 2014, pp. 325–329.
[76] S. Yi, Z. Qin, Q. Li, Security and privacy issues of fog computing: A survey, in: International Conference on Wireless Algorithms, Systems, and Applications, Springer, 2015, pp. 685–695.
[77] A. Yousefpour, G. Ishigaki, R. Gour, J.P. Jue, On reducing IoT service delay via fog offloading, IEEE Internet Things J. 5 (2) (2018) 998–1010.
[78] G. Zhang, F. Shen, Y. Yang, H. Qian, W. Yao, Fair task offloading among fog nodes in fog computing networks, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–6.
[79] Q. Zhu, B. Si, F. Yang, Y. Ma, Task offloading decision in fog computing system, China Commun. 14 (11) (2017) 59–68.

Mohammed Al-Khafajiy is a Ph.D. student in the Department of Computer Science at Liverpool John Moores University (LJMU), UK. He has also been working as a Programmer and .Net Developer at Pixus.UK since 2017. Mr Al-khafajiy has published many international peer reviewed papers in IoT, Cloud and Fog domains.

Thar Baker is a Reader (Associate Professor) in Cloud Engineering and Head of Applied Computing Research Group (ACRG) in the Faculty of Engineering and Technology at Liverpool John Moores University (LJMU, UK). He received his Ph.D. in Autonomic Cloud Applications from LJMU in 2010, and became a Senior Fellow of Higher Education Academy (SFHEA) in 2018. Dr Baker has published numerous refereed research papers in multidisciplinary research areas including: Big Data, Algorithm Design, Green and Sustainable Computing, and Energy Routing Protocols. Dr Baker has been actively involved as member of editorial board and review committee for a number peer reviewed international journals, and is on program committee for a number of international conferences. For example, he is Associate Editor of Future Generation Computer System. Dr. Baker is Expert Evaluator of EU H2020, ICTFund, and British Council.

Dr. Muhammad Asim is an Associate Professor at the Department of Computer Science, National University of Computer and Emerging Sciences, Pakistan. Having attained a Ph.D. from Liverpool John Moores University, he researches in the fields of Cloud Computing, Computer Networks, Network Security, Internet of Things and Wireless Sensor Networks.

Zehua Guo received the B.S. degree from Northwestern Polytechnical University, the M.S. degree from Xidian University, and the Ph.D. degree from Northwestern Polytechnical University. He was a Research Fellow of the Department of Electrical and Computer Engineering, New York University Tandon School of Engineering, New York City, NY, USA, and a Post-Doctoral Research Associate at the Department of Computer Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA. His research interests include software-defined networking, network function virtualization, data center networks, cloud computing, content delivery networks, network security, green networks, machine learning, and Internet exchange. He was the Session Chair of the IEEE International Conference on Communications 2018. He serves as an Associate Editor for the IEEE ACCESS and the EURASIP Journal on Wireless Communications and Networking (Springer), an Editor of the KSII Transactions on Internet and Information Systems, and the Technical Program Committee of the Computer Communications (Elsevier).

Rajiv Ranjan is a Chair Professor for the Internet of Things research in the School of Computing of Newcastle University, United Kingdom. Before moving to Newcastle University, he was Julius Fellow (2013–2015), Senior Research Scientist and Project Leader in the Digital Productivity and Services Flagship of Commonwealth Scientific and Industrial Research Organization (CSIRO — Australian Governments Premier Research Agency). Prior to that he was a Senior Research Associate (Lecturer level B) in the School of Computer Science and Engineering, University of New South Wales (UNSW). He has a Ph.D. (2009) from the department of Computer Science and Software Engineering, the University of Melbourne. He is an internationally established scientist with 260+ scientific publications. He has secured more than $12 Million AUD (£6 Million+ GBP) in the form of competitive research grants from both public and private agencies. He is an innovator with strong and sustained academic and industrial impact and a globally recognized R&D leader with the proven track record. He serves on the editorial boards of top quality international journals including IEEE Transactions on Computers (2014-2016), IEEE Transactions on Cloud Computing, ACM Transactions on the Internet of Things, The Computer (Oxford University), The Computing (Springer) and Future Generation Computer Systems.

Antonella Longo, assistant professor at the Department of Engineering for Innovation of the University of Salento, received the Ph.D. in Information Engineering in 2004. She teaches Data Management and Big data management for decision making at Management Engineering and Business school master courses. Her research interests deal with information systems and databases, service-oriented architectures design for cloud infrastructure, technology-enhanced learning and citizen science. Her current research activity focuses on big data management and exploration of cloud architecture integration with edge computing in cyber–physical social systems. On these topics, she has published more than 100 papers in peer-reviewed journals and international conference proceedings. She carries out her research activity at the SyDA — Lab (Systems, Data and Applications Lab) at University of Salento, where she coordinates the research activities about service modeling and computing and the applications in smart cities.

Deepak Puthal is a Lecturer (Assistant Professor) at the School of Computing, Newcastle University, United Kingdom. Prior to this position, he was a Lecturer at University of Technology Sydney (UTS), Australia and an associate researcher at Commonwealth Scientific and Industrial Research Organization (CSIRO Data61), Australia. He has a Ph.D. (2017) from the Faculty of Engineering and Information Technology, University of Technology Sydney. His research spans several areas in Cyber Security, Blockchain, Internet of Things and Edge/Fog Computing. He serves on the editorial boards of top quality international journals including IEEE Transactions on Big Data, IEEE Consumer Electronics Magazine, Computers & Electrical Engineering (Elsevier), International Journal of Communication Systems (John Wiley & Sons), and Internet Technology Letters (John Wiley & Sons). He has received several recognitions and best paper awards from IEEE.

Mark Taylor graduated from the University of Warwick in 1984 with a degree in Mathematics. He worked as a Computer Systems Designer for the TI Group (now part of the Smiths Group) for three years. He then worked for 5 years as a Senior Analyst Programmer at Royal Insurance UK (now Royal Sun Alliance) before joining Liverpool John Moores University in 1992 as a Senior Lecturer in Information Systems. He completed a Ph.D. at Salford University in 1999 in Methodologies and Software Maintenance. Mark has produced over 100 publications including journal papers, conference papers, papers in books, and professional publications in the computing field. During his time at LJMU he has initiated seven Knowledge Transfer Projects funded by the UK Technology Strategy Board and the UK Department of Trade and Industry, and has received research funding from the UK Department of Communities and Local Government, Merseyside Fire and Rescue Service, Liverpool Heart and Chest Hospital, and The Mersey Sports Partnership. Mark is a Chartered IT Professional, a Chartered Engineer, a Chartered Scientist, a Fellow of the British Computer Society, a Fellow of the Higher Education Academy, and an examiner for the British Computer Society.