Forensic Science International: Digital Investigation 44 (2023) 301506

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS 2023 EU - Selected papers of the Tenth Annual DFRWS Europe Conference

Evidence in the fog – Triage in fog computing systems

Jens-Petter Sandvik [a, b, *], Katrin Franke [a], Habtamu Abie [c], André Årnes [a, d]

[a] Norwegian University of Science and Technology (NTNU), Norway
[b] National Criminal Investigation Service (Kripos), Norway
[c] Norwegian Computing Centre, Norway
[d] White Label Consultancy, Norway

ARTICLE INFO

Article history:

Keywords:
Digital forensics
IoT forensics
YAFS
Fog computing
Graph measures
Evidence volatility
Triage

ABSTRACT

Fog computing promises improved service scalability and lower latency for IoT systems. The concept closes the gap between full computing capabilities at the network's edge and cloud systems' centrally located processing infrastructure. The drawback of the former is the high power requirements at the edge nodes, and the latter is the high latency for the data being transmitted from the edge to the cloud and back. One of the challenges for a digital forensic investigator facing a fog is the number of possible data locations, as the node functioning as the server processing data can be selected among several nodes in the network. An investigator typically has limited resources for an investigation; the more possible evidence locations, the more resources are required to collect and examine the data locations. A *triage* is thus needed to prioritize collecting and examining the evidence. This work analyzes measures that can identify which fog nodes are more likely to contain data, and it uses simulations to test the measures' precision and sensitivity. It aims for digital forensic investigators to maximize the utility of the available investigation resources, such that all relevant evidence is found on time.

1. Introduction

With increasingly more complex network architectures and lower latency requirements, the need for placing services closer to the edge of the network becomes apparent. Fog computing was coined in 2012, characterized by low latency, location awareness, wide geographical distribution, mobility, a large number of nodes, wireless access, real-time applications, and heterogeneity (Bonomi et al., 2012). These advantages stem from the fact that the processing and data storage happen closer to the edges of the network, but the decentralization of processing can increase the system's complexity. This increase in the number of nodes and complexity means that data being produced and processed in a fog network can spread across more nodes, and the data processing might change between nodes during regular operations.

A fog system moves the processing closer to where data is produced and consumed, leading to lower latency for time-critical applications. In addition, the amount of data that needs to be moved between the producer of the data and the central cloud can be minimized, which means that more data can be processed in applications with a limited transmission capacity to centrally located cloud servers (Li et al., 2019).

Fog computing is a paradigm supported by new communication technologies, such as 5G mobile network and Software Defined Network (SDN). 5G networks promise a low latency for data communication and have the flexibility needed for implementing new and innovative solutions (Singh et al., 2017). Several studies have shown how fog computing can be implemented in a 5G network to improve latency for real-time applications (Singh et al., 2016; Kitanov and Janevski, 2016).

SDN is also the focus of several studies on fog computing (Phan et al., 2021; Lin et al., 2020). Using SDN in fog systems enables the network's topology to change and adapt to load, congestion, and node movements. The network can distribute the workload between fog nodes, and nodes can move relative to each other. This dynamic nature of the system causes uncertainty about how data has moved and been processed in the system earlier.

A fog network is typically modelled with three layers: a cloud layer, a fog layer, and an edge layer. The cloud infrastructure can be a complex network with geographically distributed nodes. The edge nodes can also consist of entire networks and communicate

* Corresponding author. Norwegian University of Science and Technology (NTNU), Norway.
E-mail address: jens.p.sandvik@ntnu.no (J.-P. Sandvik).

https://doi.org/10.1016/j.fsidi.2023.301506
2666-2817/

with the fog layer through gateways (Sabireen and Neelanarayanan, 2021). The focus of this paper is on the fog layer.

The fog layer is often organized as a hierarchical structure, where leaf nodes are connected to the edge devices or gateways, and the parent nodes will distribute the workload among the child nodes. An example of this can be found in Hong et al. (2013), where parent nodes help the load-balancing among child nodes.

An investigator or a team of investigators often has limited resources available for the investigations. The investigation resource must be carefully utilized and prioritized to identify and collect the devices containing the most valuable evidence. Evidence in a fog system is often assumed to be in the fog nodes processing data from data-generating devices, in the cloud infrastructure, or in companion applications functioning as a user interface to the system. Companion apps and cloud accounts are typically easier to find than the node processing data. The challenge for an investigator is thus to identify the fog nodes that might have processed the data before it disappears. The research questions this study will answer are:

1. In which way does fog computing affect the identification, collection, (and examination) of evidence that is disseminated in the network?
2. How can available information and metrics be used to identify and prioritize data collection from complex and dynamic networks?
3. How sensitive are the identified measures for network attributes and dynamics?

The rest of the paper is structured as follows: Section 2 discusses related work, Section 3 develops the model for analyzing the probability of finding evidence, followed by Section 4, which describes the implementation of a volatility class for the YAFS simulator. Section 5 describes the experimental setup and results. The paper is concluded with a discussion of the results in Section 6 and conclusions in Section 7.

2. Related work

2.1. Fog computing

Bonomi et al. (2012) were the first to present *fog computing*, where the system moved data processing from centrally placed nodes towards the system's edges. They defined *fog computing* as *"[…] a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud Computing Data Centers, typically, but not exclusively located at the edge of network"* (Bonomi et al., 2012).

The defining characteristics of a fog system envisioned by the authors were that they would be low-latency systems where nodes know their location. Furthermore, fog systems would have a wide geographical distribution, high mobility among the nodes, and considerable variability in the hardware and software in the system. In addition, the authors anticipated that systems would be optimized for streaming and real-time applications, and subscriber models would open up for new providers.

The fog system described by Hong et al. (2013) contains a logical tree structure, with the cloud as the root, the sensors and actuators connected to gateways in the leaf nodes, and the fog nodes forming the rest of the tree between these. In this system, nodes can enter or leave the system, and they can also move between the branches that are governed by different fog nodes. Processes in the fog nodes can be split and merged depending on the load in the system, where the parent node supervises the child nodes and can offload work and assign work to siblings or itself.

Dastjerdi et al. define a reference architecture in five layers, where the *applications* is at the top layer, based on the *software-defined resource management* (Dastjerdi et al., 2016). This management layer depends on *cloud services and resources*, which is dependent on the *network*. At the bottom of the architecture is the *sensors, edge devices, gateways, and apps*, which produce the data and handle the routing of the data into the network. The authors use the architecture by Hong et al. (2013) for their experiments.

The YAFS simulator's authors researched graph centrality indices for data placement (Lera et al., 2018). The authors differentiate between the service location and the data location, as the data cost more to move than the processing location. The authors tested various centrality indices in the network and analyzed how this affected the data placement. This experiment was repeated for three different network topologies, and they concluded that the eigenvector centrality produced better data placement locations.

The papers discussing the hierarchical setup of the nodes do not specify how the hierarchy is supposed to be formed in a fog layer. Alammari et al. (2021) assume an already existing hierarchy, and Forti et al. (2021) use as an example a network that is distributed over several locations and has a naturally occurring hierarchy.

Table 1 shows a summary of the fog system topologies, service placement, and dynamic elements described in a selection of papers on fog computing.

Hegarty et al. (2014) discussed the challenges and opportunities for forensic investigations in fog systems, where the authors show how various types of information can be collected together with the acquisition process. The authors describe a procedure for evidence acquisition in fog systems, where the investigator first *identify devices* that contain evidence and the *scope* of the investigation, followed by the examination of the *composition* of the fog system, and the decision on which *methodology* to use for data collection together with the *implementation* of the collection plan.

2.2. Volatility

The volatility of traces has always been a concern for forensic examiners and is no different for digital forensics. The relative volatility of data is described in the *order of volatility* in many textbooks on digital forensics (Casey, 2009, 2009rnes et al., 2017). However, an absolute measure of volatility has been used to quantify the data lifetime for secure deallocation (Chow et al., 2005) and to measure the lifetime of MAC address data from passing devices in routers Minnaard (2014).

Chow et al. (2005) defined a *data life cycle*, where data has an *ideal lifetime*, which is the time the data is in use, a *natural lifetime* when data exist and can be retrieved. The *secure deallocation lifetime* can shorten the natural lifetime by zeroing memory contents after deallocation.

Sandvik et al. described a volatility model for IoT systems, where the model consists of 6 elements (Sandvik et al., 2021): The *storage abstraction layers* describe the abstraction layers that encode and store the data, the *events* that affects the system, the *application activity functions* which is the functionality of the running program, the *memory device reliability* that describes the physical reliability, the *storage management functions* describe the encoding of data in each storage abstraction layer, and the *environment* contains configurations and other attributes statistically affecting the volatility.

The *volatility* is defined as the expected time before the data is inaccessible. The data can become inaccessible by data being overwritten or erased, or the available collection methods cannot access the data. The volatility can be viewed as a stochastic process, where there is a probability distribution for the time until data becomes inaccessible to the investigators.

Table 1
Topologies, Service placement, and dynamic elements described in the literature.

Description	Paper
Topology	
Hierarchical tree structure	Hong et al. (2013), Dastjerdi et al. (2016), Alammari et al. (2021), Forti et al. (2021)
Fog colonies	Skarlat et al. (2017), Guerrero et al. (2018), Lera et al. (2019a)
Random, Barabasi-Albert, Lobster graphs, Grid layout	Lera et al. (2018)
Gateways	Huang et al. (2017)
Service Placement	
Distributed services	Bonomi et al. (2012)
Hierarchical deployment, offload to siblings	Hong et al. (2013), Dastjerdi et al. (2016)
Preference to neighborhood	Skarlat et al. (2017), Lera et al. (2019a)
Close to edge	Forti et al. (2021)
Gateway	Huang et al. (2017)
Dynamic Element	
Node mobility	Bonomi et al. (2012), Hong et al. (2013), Huang et al. (2017), Forti et al. (2021)
Node failure	Lera et al. (2019a), Alammari et al. (2021)
Routing changes, including SDN	Dastjerdi et al. (2016), Skarlat et al. (2017)

The period some data exists in one location consists of two parts: the time from writing the data until the location of the data is unlinked or freed by the application using the data, and the time from the data is unlinked and until it is overwritten. The application decides the first part, and the last part hinges on how fast the location is overwritten, which is dependent on the type of memory, the writing rate, and the memory size.

The probability distribution of volatility in devices running Contiki OS was discussed in a paper by Sandvik et al. (2022), where findings showed that each device had an almost uniform distribution of the lifetime of data. In contrast, the combined lifetimes of the data over all tested combinations of writing rates showed a negative exponential distribution.

3. Analytical model

A model of the existence of data in a fog system needs to be specified. This model shows how a subset of the devices can be selected for acquisition/examination to yield the optimum combined evidence value. It will also be used as a foundation for the estimated probability of the existence of evidence.

3.1. Probability for the existence of data

The probability of data existing in a device after a time in a fog system is given by the probability that the data has existed in the device and remains after the elapsed time. The probability can be expressed as:

$$P(k_i \in k_{\text{data}}, t_{\text{acq}} < t_{\text{vol}}) = P(k_i \in k_{\text{data}}) P_{k_i}(t_{\text{acq}} < t_{\text{vol}}) \tag{1}$$

where $P(k_i \in k_{\text{data}})$ is the probability of node k_i to be among nodes in the set of nodes that contain the data, k_{data}, and $P_{k_i}(t_{\text{acq}} < t_{\text{vol}})$ is the probability for the node to contain data at the time of acquisition, $t_{\text{acquisition}}$. Thus, the model can be split into two parts: to find the probability for a particular node process data and the volatility of the data in the nodes. The probability of data existing after time, t_{vol}, is further discussed in section 3.2, and the probability that the data exists in node k_i is discussed in 3.3.

Not all data in a system is essential for a particular investigation; only the data that can be considered *evidence* in a case is sought after. Evidence can be evidence that, in the end, shows up in court, but it can also cover data that can illuminate questions that arise during an investigation. Mainly, these questions are formed as hypotheses, where data will support or refute the hypotheses. Thus, the investigation seeks to find the data that can be used as evidence to refute or strengthen the set of investigation hypotheses. This subset of the data in the system is the evidence.

3.2. Data volatility

In this work, we follow the definition from Sandvik et al. (2021), where *data volatility* is defined as the probability distribution of the data lifetime and the centroid of this distribution given by the expected lifetime of the data. This definition is similar to the concept of system reliability and the Mean Time to Failure (MTTF).

Similar to Chow et al. (2005) and Sandvik et al. (2021), the volatility is considered the sum of two variables: The first variable is the time from data is created in a storage location until the storage location is unlinked, which happens when the application using the data no longer needs it. The second variable is given as the time from the storage location is unlinked until it is overwritten by another value, which means the operating system reclaims the location.

In this work, both variables are given by a negative exponential distribution, with set parameters for the distributions:

$$P(t_x = t_{\text{unlink,erase}}) = \lambda e^{-\lambda t_x} \tag{2}$$

$$= \frac{\exp\left(\frac{-t_x}{t_{\text{unlink,erase}}}\right)}{t_{\text{unlink,erase}}} \tag{3}$$

$$P(t_x > t_{\text{unlink,erase}}) = \int_0^{t_x} \lambda e^{-\lambda t}\, dt \tag{4}$$

In reliability computing, the λ parameter in the negative exponential distribution is referred to as the static failure rate. In this paper, λ is statically set for each type of node and message, where the types of nodes can be a source, a target, or a proxy that relays traffic.

3.3. Data distribution

The nodes generating and consuming data are more likely to contain accessible data for longer than the nodes that relay the data in the network. Apart from sensor and actuator nodes, the nodes

acting as processing nodes most likely have more prolonged volatility of the generated data than the nodes relaying the data between the source and target of the communication. Therefore, identifying the nodes acting as server nodes in the fog system is crucial in an investigation.

The service placement is selected when a node sending data needs to find a node to receive the data. This receiving node is selected from a set of nodes that meets the requirements to process the data, which can be called S. The number of nodes that can act as a server is given by the size of the set, $|S|$, and the set can be an ordered or partially ordered set by the selection preference.

The distribution of data in a fog system depends on the distribution of sources and targets of the data, the processing nodes, and the nodes forwarding traffic in the system. Sensor nodes are usually found at the edges of a tree network or spread across a mesh network, and these are the source of data from their environments. Some sensor nodes are small, resource-constrained devices that have to run for a long time on batteries, while others are more powerful devices connected to external power.

Target nodes can be fog service nodes that process the data or actuation devices that act on the physical world based on the input data. One target is the processing nodes, which in addition to just processing the data, also can send new data to a new target based on the input. Another target is the actuating nodes that will physically affect their environments based on the data they receive. Just like the sensor nodes, the target nodes can be both resource-constrained and more powerful nodes, but note that physical movements will require more power than just sensing the environment.

The data needs to be transmitted from the source nodes to the target nodes, and the proxy nodes will forward data from one node to the next on the path between the source and target. These proxy nodes can contain the forwarded data if it is not encrypted during transmission.

A workload can be distributed to neighbouring nodes if the original processing node is overloaded and the new node can still process and transmit the data within the required timeframe. In many systems, the node closest to the source is used for processing the data from a source node, but if the node is overloaded, other nodes may take over.

Dynamics, such as load balancing of services between nodes, the movement of sources, and the changing topology from SDNs, make it hard to determine where data might have been stored. While the selection of service nodes is usually deterministic regarding the state, the knowledge about the state at a specific time is hard to recover. The uncertainty of the exact state of the system makes the system model a stochastic rather than a deterministic system.

The most straightforward service placement method is where only one node is used as a fog node, typically a gateway node sitting between the edge nodes and the central cloud node. In this case, the placement only depends on the source nodes' physical location. In contrast, the location of the evidence is only dependent on which gateway the edge nodes are connected. Therefore, the probable evidence location is among the gateways that administer the networks the source node can have been connected to during the period in question. As there are a limited number of gateway nodes, the number of possible evidence locations is also limited in this case. However, it can be challenging to identify all gateways if the device has moved between several physical locations and networks.

Another model is a complete mesh network, where all nodes can communicate with everyone else in their vicinity. Rafi et al. (2019) is an example of a fog network structure where all nodes are in a mesh network, and sensor nodes are evenly distributed among fog computing nodes.

Most fog systems in the literature are between these extremes and consist of a layer between the edge nodes and the cloud. The architecture and service placement algorithms in these fog systems vary between implementations, and most describe a hierarchical system where the cloud layer has a complete overview of the fog nodes. The constraints to the service placement given by a hierarchy or fog colonies affect the set of nodes considered to meet the selection criteria. However, the hierarchy and fog colonies are defined given some existing connection measures, either physical locations, organizational locations, or centrality-based measures. This paper assumes that the constraints do not affect the order of the node's probability of containing data to a considerable degree, but this assumption must be tested.

The objective functions used to optimize the service placement and routing often considers network latency, energy usage, closeness to the source nodes, and the workload in nodes. For this work, we use the messages' latency between the system endpoints and the memory constraints as the weighting and constraints of the service placement, but other attributes can also be used.

If several possible fog nodes can process the data from a source node, the number of evidence locations can increase considerably. In the extreme version, a fog network can be a complete mesh network, where any node with enough resources for the processing task can act as a fog node. In this system, the probability of a node being selected as a service location is the probability that any node with enough resources to process the data is equally likely to process the data from a sensor node. Equation (5) defines the set of nodes that meets the requirements for a processing task, and Equation (6) shows a uniform distribution over the nodes that meet the minimum required resources:

$$S = \left\{ \bigcup_i (x_i) : x_i \in X \wedge \mathrm{Res}(x_i, a) \geq \mathrm{Res}_{\min}(a) \right\} \tag{5}$$

$$P_{\mathrm{uniform}}(x) = \begin{cases} \frac{1}{|S|} & , x \in S \\ 0 & , x \notin S \end{cases}, \tag{6}$$

where X is the set of all nodes, x is one node, $|\cdot|$ is the cardinality of the set, Res is a function returning the node's resources, and $\mathrm{Res}_{\min}$ is the minimum resources required for a node to be a fog service node for a particular application, a.

However, Eq. (6) does not consider the network's attributes that affect the placement of the services, such as the network and server loads, routing, or the physical location of the data sources and cloud servers. To generalize the approach, Equation (7) shows the general equation for the probability of a node processing data:

$$P_{\mathrm{general}}(x) = \begin{cases} \frac{w(g(x))}{\sum_j w(g(x_j))} & , x \in S \\ 0 & , x \notin S \end{cases}, \tag{7}$$

where $x \in S$, $g()$ is a graph measure that corresponds with the likelihood of a node to be selected for processing data, and $w()$ is a function for weighting the graph measure. The question is thus to find a measure to use and a weighting that corresponds to the probability for nodes to be data processing nodes.

Note that in Eq. (7), if the weighting is uniform among the nodes, then the equation reduces to $\forall x$: $w(g(x)) = 1$ and $\sum_x w(g(x)) = |S|$, which is equal to Eq. (6).

The weighting can be estimated by analyzing the relative difference between the value of the graph measure and the

parameters that decide the placement of a particular service. For a randomly assigned service, the probability for a more central node to be the one that is processing the data is higher because there are more paths leading through the node between the data-producing node and any nodes using the data.

3.4. Distance based weighting

The simplest model for finding the weight a node has among all nodes is to find the shortest distance between the data source and the target nodes. The weight of a node can be expressed in the following equation:

$$w_n = \frac{1}{d(s,n) + \sum_{t \in T} d(n,t)}, \tag{8}$$

where $d(n, m)$ is the shortest distance between two nodes, s is the source node, and T is the set of target nodes that are consuming the information from the processing of data from s.

This distance measure can be the number of edges or the time it would take to send the data through the shortest path. The simplicity of the method makes it a good option for analyzing networks.

3.4.1. Path weighting

The order of the selectable nodes depends on the application's requirements. One of the main arguments for using fog computing systems is the decreased latency for data processing. The time for sending a message is likely to give an ordering similar to the exact ordering used by the service placement algorithms.

In this work, we use the inverse of the time a small probe packet would use between a source sensor, being sent and processed by a server, and retransmitted to a target actuator is used as the weight of a path. The inversion step in Eq. (10) ensures that the shorter latency will be associated with a higher weight.

$$t_{\text{lat}} = t_{\text{proc}} + \sum_{i \in \{e_{sn}\}} b_i s_{sn} + \sum_{v \in V_{\text{path}}} t_{\text{routing},v} + \sum_{j \in T} \sum_{i \in \{e_{nj}\}} b_i s_{nj} \tag{9}$$

$$w_{\text{lat}} = \frac{1}{t_{\text{lat}}}, \tag{10}$$

where $\{e_{sn}\}$ and $\{e_{nj}\}$ are the set of edges that the data will pass between the source and fog node, and the fog node and target nodes, respectively, where T is the set of target nodes. b_i is the bandwidth of the i-th edge, s_{sn} and s_{nj} are the message size/amount of data sent between the source and fog node, and the fog node and the targets, respectively. V_{path} is the set of nodes in each path, $t_{\text{routing},v}$ is the time a node will use for routing data, and t_{proc} is the time the fog node use for processing the data and send out new data for the sink node.

Note that t_{lat} used in Eq. (9) is used for the distance metric in Eq. (8).

3.4.2. Path-based internode weighting

If the fog layer is distributed among nodes in a network, there has to be some central orchestration service that knows which nodes can be used for which tasks, and it has to have some idea about which nodes that can reach the latency requirements of the task, given a source node.

As the latency is minimized, the probability of a node laying on the lowest latency path is high. In addition, more than one path can meet the minimum requirements for latency, and the actual path can be on any of the paths within the latency requirements. An investigator does not necessarily know the latency requirements or the load in the system, so these are unknowns that make determining the probability more challenging.

The probability of a node containing data is thus given by the number of non-cyclic, latency-weighted paths through the node. A high-latency path results in a lower probability of the node being selected than a low-latency path. The sum of weighted paths that pass through a node in S divided by all possible weighted paths between the source and sink gives the probability that a node has processed the data.

$$W_n = \sum_{p \in P_n} k(t_{\text{lat},p}), \tag{11}$$

where P_n is the set of paths passing fog node n between the source and sink, $t_{\text{lat},p}$ is the total latency, or cost, of the path p, and $k(t)$ is a scaling function for how suppressed higher latency paths should be, in this paper, $k(t) = t^{-1}$, but this might be needed to be updated based on the observed behaviour of the network.

The weighted ratio of paths containing a node, n, is given by Eq. (12).

$$R_n = \frac{W_n}{\sum_{x \in N} W_x} \tag{12}$$

The model for the architecture is given in Eq. (13), where the probability of a node containing data is given by the shortest path between the source and sink node that contains at least one fog node from the set, S.

$$P(S = n) = \frac{R_n}{\sum_{i \in N} R_i}, \tag{13}$$

where s is a service location, n is a fog node, and R_n is the total weight of the latency of the paths passing that node, given in Eq. (12).

An example of a small network is shown in Fig. 1, where nodes A and E are the source and target, and edges are labelled with the cost of each path connecting the nodes. The five non-cyclic paths between A and E are listed in Table 2 with the total weight of each path.

The weighted ratio of the paths going through node B is given by applying Eq (12): $W_B \approx 0.64$. The same calculation can be done for the paths through the other nodes. The weighted ratio for paths through node C is thus: $W_C \approx 0.24$.

The probability for a particular node containing the service is then given by the weighted ratio of all paths passing a node divided by all weighted ratios: $P(S = B) \approx 0.204$. Table 3 shows the results for the rest of the nodes, and it shows that the source and target

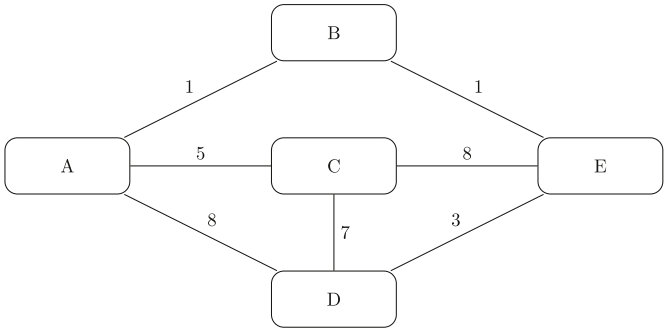

Fig. 1. A small network example showing the cost of the paths between node A and E.

Table 2
The paths Between A and E, together with the total cost.

#	Path	Cost
1	ABE	2
2	ADE	11
3	ACE	13
4	ACDE	15
5	ADCE	23

Table 3
The results for the weighted ratio and the probability of a node in the example network in Fig. 1 containing the service. Weighting function = t_{lat}^{-1}.

Node	W_n	P(S = n)	P(S = n) \{A, E}
A	1	0.318	–
B	0.642	0.204	0.562
C	0.240	0.076	0.210
D	0.260	0.083	0.228
E	1	0.318	–

nodes, A and E, are very likely to contain the service, as they are a part of all the paths. Node B is on a path with a low cost and has a 20% probability of containing the service, while nodes C and D are on high-cost paths and, therefore, unlikely to contain the service.

If there are some minimum requirements for the service nodes, not all nodes in the paths are eligible for being selected as service nodes. These nodes will be removed in calculating the probabilities for containing data as described in Eq. (13). For example, if the end nodes, A and E, are excluded from the set, then the corresponding probabilities for nodes B, C, and D to be the node processing data are 0.562, 0.210, and 0.228, respectively.

The probabilities changes to the ones shown in Table 4 if the weighting of the path cost is squared. The graph has only one path through the low-cost edges but several possible paths through the high-cost edges. The difference in the number of paths might skew the result, as several high-cost paths might not necessarily mean that they are preferred to the low-cost path. The table shows that the weighting is vital for the absolute probability of the node containing data.

3.5. Computational optimization

The path-based internode weighting of nodes is computationally expensive, and the implementation ran for more than 170 h on a network with 1000 nodes and approximately 2000 edges. Even though there are many simple paths through a particular node, the more costly the path is, the less that path affects the weighting of the node. Thus, the path enumeration can be optimized by only enumerating the paths with a high weight and dismissing those of low weight.

A path of length n always has less latency of the same path with an added node to the end. The weight of a path will therefore decrease as the length increases. The latency is related to the path length, as a longer path will have more propagation delays than a single path. For short network packets, the propagation delay in each node is relatively huge compared to the time to send the probe.

The number of edges in the path can therefore be used as a proxy for the weight of the paths, and we can use existing cutoff functionality in standard path-enumerating functions. For our work, we set the cutoff at the distance of the shortest path plus five edges or 110% of the shortest path length between the source node and the destination, depending on which is the highest. This cutoff gives most of the relevant paths while keeping the computational cost reasonably low, but a long path with a very high weight will be missed with this optimization.

Table 4
The results for the weighted ratio and the probability of a node in the example network in Fig. 1 containing a service. Weighting function = t_{lat}^{-2}.

Node	W_n	P(S = n)	P(S = n) \{A, E}
A	1	0.331	–
B	0.924	0.301	0.903
C	0.045	0.015	0.044
D	0.054	0.018	0.053
E	1	0.331	–

3.6. Fog network dynamics

A network is seldom static but continuously changes by the movement of nodes, varying connection quality between nodes, and nodes being shut down or inserted into the network. Other dynamic elements are the changing load on the nodes, both instantaneous and periodic, and changes to the effective network topology by SDN.

A varying connection quality leads to the amount of data sent between nodes varying over time. The connection quality will affect the time for transmitting a message, which will affect the weighting of the path and the graph measures that use the path weight for the measure.

Nodes failing, being repaired, and reconnecting, or nodes shut down and turned on again can be modelled by nodes suddenly disappearing from the network and later reappearing in the same place. This behaviour can also affect the measures, especially if the node is essential in the network as interpreted by the metric.

A changing load in the network will affect the ability to transmit traffic, both for the network load and the server load. A server load and the corresponding traffic will have to be routed to another server node, while a network load will have to route the traffic to the same node but via a different path.

In this work's experiments, only the dynamics of the network connections are tested by varying the bandwidth between nodes and the propagation delay and disconnecting and connecting new edges in the network. The bandwidth variation emulates the variability of the wireless communication strength between nodes, where a lower bandwidth corresponds to a longer distance between nodes, an object blocking the radio waves between the nodes or RF interference. The variation in propagation delay emulates the variance in the workload in a node.

4. Implementation in YAFS

The YAFS simulator is a framework built on the SimPy Python simulation library and the NetworkX graph library. It can simulate various fog computing systems (Lera et al., 2019b). The original simulator can be downloaded from Github.[1] The fork that contains the volatility extension and code used in the experiments can be downloaded from the author's Git repository,[2] where the volatility branch contains the added functionality.

4.1. Implementation of data location distribution

In this paper, three different fog systems are implemented, each described in the literature: A complete mesh network, a network overlay with a tree structure, and a fog colony-based overlay. The

[1] https://github.com/acsicuib/YAFS.
[2] https://github.com/jenspets/YAFS/tree/volatility.

trivial systems that include processing in a single gateway node are not covered, as the probability of data being in the gateway node is given solely by the volatility of the data.

Each node in the simulator received a randomly assigned resource amount using a negative exponential distribution. The sensor nodes were selected among the ones with the least resources. The fog server nodes were selected among the nodes with resources over a given threshold. The actuator nodes were selected among the remaining nodes such that none of the nodes was assigned different tasks simultaneously. The attributes of the connections between the nodes were set using randomly assigned bandwidth and propagation delay attributes. The weight was determined by the time it would take to send a small packet over the connection.

The processing node will process the data, which may be stored there for longer. This work assumes that the volatility of data in relaying nodes is so high that it is highly likely to be overwritten when an investigation is performed. The second assumption is that the data-producing node is known but highly volatile, so there is a high probability for the data to be overwritten. The most likely node to contain data is thus the node processing the data.

The three implementations selected for this research are: (i) fog nodes placed randomly in a mesh network, (ii) the generation of a hierarchical structure in an existing network, and (iii) a fog colony system.

There are several standards today for routing in a mesh network, such as IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) (Winter et al., 2012) and Thread (Unwala et al., 2018). Some authors have used architectures with a more generic routing algorithm in their research. Such algorithms often use the routing finding algorithms that are available in simulation algorithms, such as Dijkstra's shortest path found in NetworkX (Hagberg et al., 2008). For this research, we selected one open mesh network as a baseline for mesh networks, using Dijkstra's shortest path algorithm to find the optimal network path. The path weight is the combined propagation delay given by the size of the message sent, the bandwidth of each of the edges, and the propagation delay. These values were selected from a gamma distribution, with parameters shown in Table 5.

The hierarchical structure is often described as a tree structure with or without connections between siblings in the hierarchy. The routing hierarchy is made by first generating a network that acts as the physical network. From this network, a subgraph is generated by creating a minimal spanning tree, where the weighting of the edges is created by the latency a small probe would use between nodes. The most central node is selected as the cloud node, and the hierarchy is created from this cloud node.

The fog colony architecture is modelled after the fog colony algorithm by Guerrero et al. (2018), where the most central nodes are selected as fog colony controller nodes. These controller nodes are responsible for distributing the workload within the colony and submitting the task to other nodes in case none have available resources for processing the task. When the controller nodes have been selected, the rest of the nodes are assigned to colonies by their closest fog colony controller node. All connections between colonies from processing nodes are severed, so only the controller nodes communicate outside the colony.

Table 5
Parameters used for network generation and fog setup.

Parameter	Value
Ratio of sources	0.1
MEM distribution	N.E.D
MEM lambda	10^{-4}
Min. MEM for server	10^3 (MB)
Propagation delay distribution	Gammavariate
Propagation delay alpha	1
Propagation delay beta	0.2
Bandwidth distribution	Gammavariate
Bandwidth alpha	1.5
Bandwidth beta	1

4.2. Implementation of volatility in YAFS

The volatility extension is a new class added to the simulator and consists of an abstract Volatility class. This class contains four functions; two that set the volatility distribution (`set_unlinkdistr` and `set_erasedistr`) and two that sample from the distribution (`get_unlinktime` and `get_erasetime`). The two functions for setting and sampling the distributions represent the time from creation to deletion and the time from deletion to erasure/overwriting. These processes are represented with two different distributions, where the total volatility of the data is the sum of these two times. The distribution for each node is up to the subclasses to implement.

The sampling functions are called from the Sim class, where the network process runs, and each fog application has its Volatility class. This way, different applications can behave differently. The function `__volatility_function` in the Sim class is called whenever a message reaches a node. This function will sample one volatility time from the unlink distribution and one from the erase distribution. Two periods define volatility: the time between the message being received and deleted and the time between the message being deleted and erased.

If the node is a message source, the unlink sample time is subtracted from the message creation time, as the source message has to be created before the message is sent. The simulator then assumes that the message is deleted when it is sent.

The creation, unlinking, and erasure timestamps are then logged to a file, together with the time deltas between these timestamps and the node, application and message created in the node. From this file, the volatility statistics are calculated. Again, the Stats class is responsible for this.

5. Experiments

As noted in Section 3.3, the nodes using and processing the data from the data generator typically have lower volatility. Therefore, in the experiments, we are testing the measure developed in Section 3.4.2 to see how well this can identify the nodes that contain the service.

The absolute probability of a node being the server node is highly dependent on the path length, so the *order* of the probabilities better describes the precision of our method and can be compared with the order of other centrality measures. The placement of the actual server node in the top end of the ordered set means that the measure is better at identifying the server nodes than a result in the lower end of the ordered set.

The following section describes the experimental setup and the results from the experiments.

5.1. Experimental setup

All experiments were done using a version of the YAFS simulator with a volatility extension, as described in Section 4. The following sections describe the setup for each of the experiments for the various aspects of prioritization of nodes, and Table 6 is an overview of the experiments.

The measures were ranked by their value for each experiment, and the rank of the service node was recorded. The measure with a

Table 6
Overview of the experiments.

Ex.	Description
1	Measure compared with centrality measures for small and big network
2	Measure sensitivity for network types
3	Measure sensitivity of fog architectures
4	Measure sensitivity for dynamic networks
5	Measure sensitivity for server probability

Table 7
Configuration of networks used for comparing measures.

Id	Type	Size	Prob serv
1	Barabasi-Albert	50	0.5
2	Barabasi-Albert	250	0.5
3	Barabasi-Albert	500	0.5
4	Barabasi-Albert	1000	0.5

consistent rank close to the top will yield a higher probability for nodes to be selected as server nodes.

5.2. Comparison between measures

The following centrality measures were compared against the path-based internode weighting: *Degree centrality*, *Betweenness centrality*, *Closeness centrality*, *Eigenvector centrality*, and *Harmonic centrality*. The experiments were performed with a Barabasi-Albert graph with several network sizes, varying between 50 and 1000 nodes, and an attachment parameter of 2. For all these experiments, the probability of selecting a node as a fog service node was set to 0.5. Each experiment was done with ten runs for each set of parameters, and Table 7 shows the configuration of the simulated networks.

Two variants of the internode weighting were done: one where the end nodes were ranked among the rest and one where the end nodes were exempt from the ranking. The latter would give a more precise rank assuming that the end nodes are not server nodes.

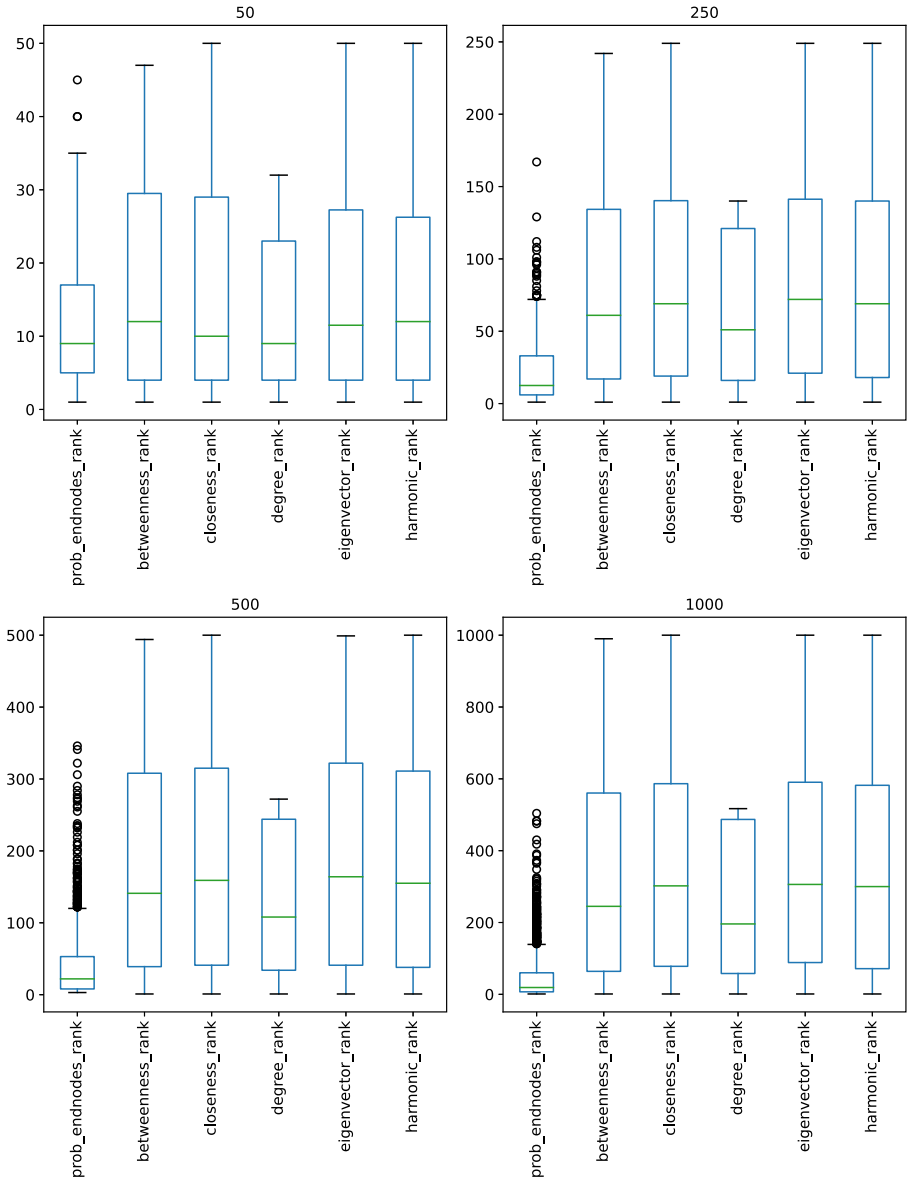

Fig. 2. Comparison of the centrality rank of the fog server node for the set of measures for different network sizes.

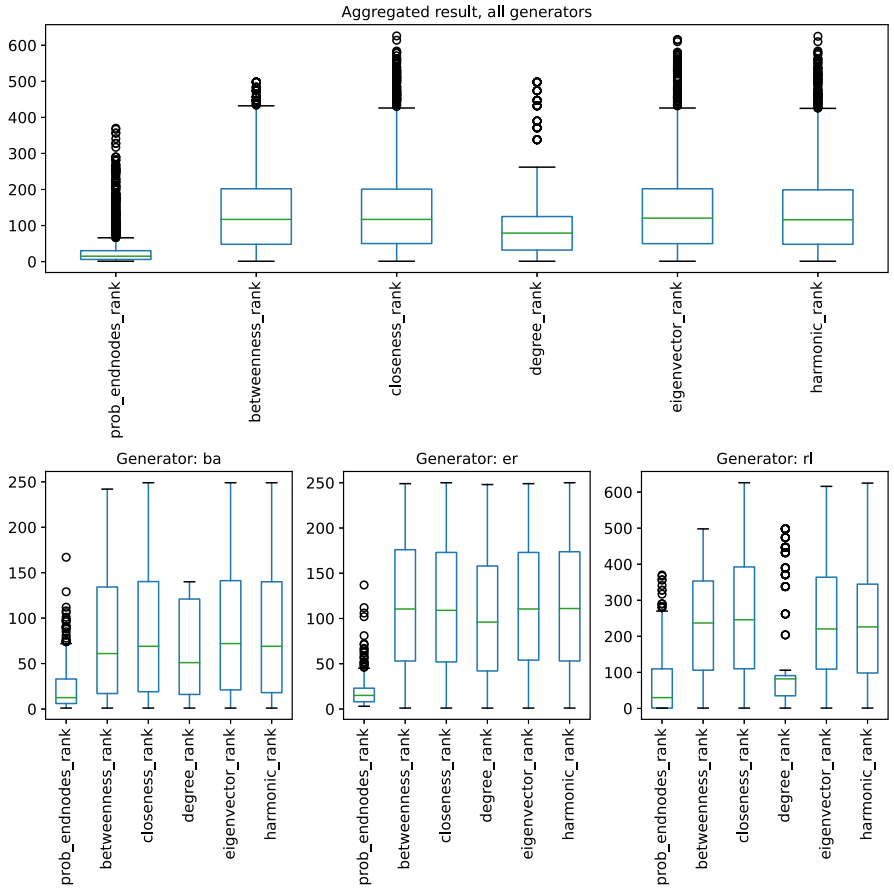

Fig. 3. Comparison of the aggregated rank of all graph types of size 250 together with individual graphs for each network type.

Fig. 2 shows the set of boxplots for the sizes of graphs, and we can see that for smaller size networks, all centrality measures give similar results, but for larger networks, the internode weighting performs better.

5.3. Sensitivity for network types

The type of network can impact the measurements, which are tested by running the experiment with the same parameters but varying the network generator used for creating the network. The *Barabasi-Albert graph generator* is a preferential attachment type of graph where popular nodes are more likely to receive new connections. A BA network is a type of network often found in nature. The *Erdos-Renyi generator* will create what is often referred to as a *random network*, with no preference for new connections. The last generator used in this experiment is the *Random Lobster generator*, which connects a backbone and adds connections to this backbone with a set probability.

The probability for an edge to form was set to 0.05 for the Erdos-Renyi generator. For the Random Lobster generator attributes, the probability of adding an edge to the backbone was set to 0.2, and adding an edge one level beyond the backbone was set to 0.1.

Fig. 3 shows a boxplot for the aggregated results from simulations of networks created with a size of 250, together with the results for the individual graphs. The internode weight is performing well for all.

5.4. Sensitivity for fog architecture

The fog architecture types tested in this experiment are the complete mesh, a tree structure, and a fog colony structure. In all three cases, the number of nodes was set to 250, each run ten times, generating new networks for each run. For all experiments, the server selection probability was set to 0.5, and 10% of the nodes were selected as message sources. The rankings were aggregated for all ten runs.

The full mesh is the original network generated by a Barabasi-Albert generator. The tree architecture was created from a BA-generated network by making a minimum spanning tree and connecting siblings with a probability of 0.5. The fog colony architecture was created from a BA-generated network by making ten colonies, disconnecting nodes in one colony from nodes outside the colony, and retaining the connections of each colony's controller node.

Fig. 4 shows the result of this experiment.

5.5. Sensitivity for dynamic networks

Dynamics is expected to be a challenge for any measure used post facto, after the event, and after the system has changed. The experiments were done by adding two sources of dynamics: changing the weight of the edges by changing the propagation delay and bandwidth and by removing and adding edges with new attributes.

The parameters for the dynamic attributes are: probability for changing an attribute is 0.1, and it is the same parameters for the gamma variate distribution for propagation delay and bandwidth as the original graph. Fig. 5 shows the results from this experiment, where the results show a degradation when the network changes during the simulation.

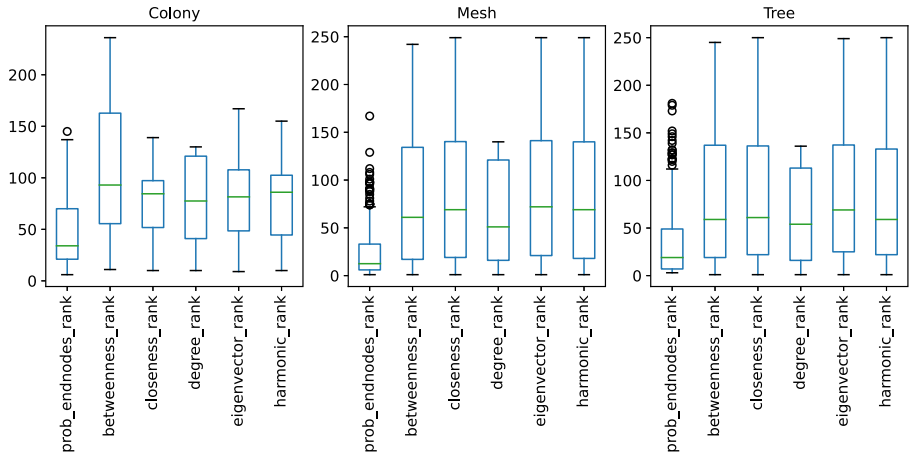

Fig. 4. Results from three different fog architectures: Mesh, Tree, and Colony.

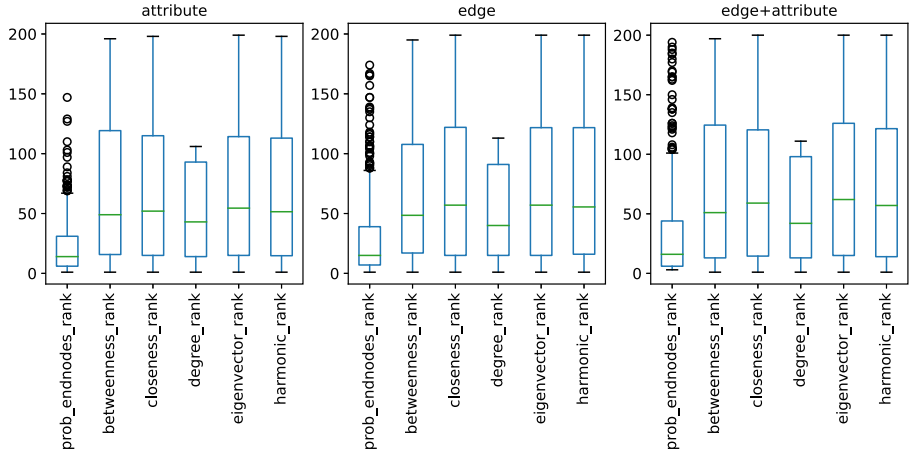

Fig. 5. The effect of dynamics on the rank of server node for the measures.

5.6. Sensitivity for number of servers

The robustness of the number of servers in the network indicates how the measure would perform if there are fewer available server nodes or if a load on the existing nodes makes the system select server nodes further away. Fig. 6 shows the result forms a gradual increase in the number of nodes for the internode weighting.

6. Discussion

The experiments in this work have been done using simulations of fog systems. The robustness of the results among the variations implemented in the simulator indicates that the method is among the best measures, even if the examined network differs from the ones simulated here.

The basis of the model is the probability of a node containing data after a time. The nodes have in this work been ordered by their likelihood to process evidence. This order is the relative probability, and while this is what the investigator would need to prioritize the evidence collection, the absolute probability for the nodes to contain evidence is also of interest if the triage has to consider more than one type of evidence. To validate the correctness of the absolute probabilities given by the model is thus a task for future work.

This method assumes that the investigator can assess the connections between nodes, the latency and bandwidth of the connections between them, the weight of the service placement probability, and the data volatility in the nodes. The cost function for the network connection can be estimated by the type of networks they use and the distance between the nodes, as the connection speed depends on network quality, which degrades the distance between the nodes. The weight of the service placement probability will not affect the order of the nodes, sorted by their probability for containing the service unless the weighting is not merely a scaling function. The volatility can be estimated using the assumed memory size, encryption, and the assumed load of the node. However, the attributes in the model need to be estimated, and the model's sensitivity to these attributes should be further investigated in later work.

As fog computing is still a young research area, no existing systems yet utilize the concept's full potential. Therefore, the architecture of future systems can still be different from what is envisioned today, and the market might take another route than this paper assumes. Still, node weightings like those described in this paper can be used for other types of network investigations, such as finding probable attack vectors by locating the more central nodes among all possible attacks.

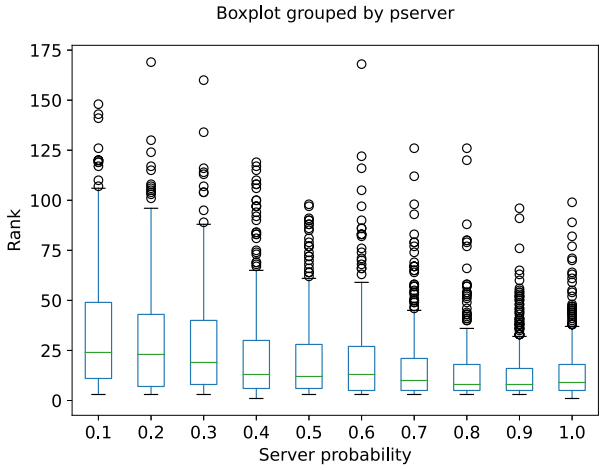

Fig. 6. The sensitivity for the number of server nodes. The horizontal axis shows the probability for a node to be a server node.

7. Conclusion and future work

In this work, we have studied how the fog computing concept affects the digital forensic process to discover relevant forensic evidence in a fog system. New methods for triage are necessary to find relevant evidence with limited resources among hundreds or thousands of evidence locations.

We presented a measure of the probability of a service being placed in the node by calculating the weight of the paths passing through the node. The measure ranked the nodes more likely to contain the service higher than other centrality measures. Further, we showed that the measure is robust when varying several attributes of the network, such as size, dynamics, network architecture, fog architecture, and the number of eligible server nodes.

However, the method is computationally expensive, even for moderately sized networks. Optimizing the algorithm by restricting the path length traversed based on the shortest path length between nodes did not impact the method's performance. Future work must address how the network connections and attributes can quickly be estimated so that the investigator does not need to use valuable time and resources to determine the nodes' ranking.

Acknowledgement

The research leading to these results has received funding from the Research Council of Norway program IKTPLUSS, under the R&D project “Ars Forensica — Computational Forensics for Large-scale Fraud Detection, Crime Investigation & Prevention”, grant agreement 248094/O70.

References

Alammari, A., Moiz, S.A., Negi, A., 2021. Enhanced layered fog architecture for IoT sensing and actuation as a service. Sci. Rep. 11 (1), 21693. https://doi.org/10.1038/s41598-021-00926-y. URL.

Årnes, A., Flaglien, A., Sunde, I.M., Dilijonaite, A., Hamm, J., Sandvik, J.-P., Bjelland, P., Franke, K., Axelsson, S., 2017. Digital Forensics. John Wiley & Sons, Ltd.

Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. In: MCC'12 - Proceedings of the 1st ACM Mobile Cloud Computing Workshop, pp. 13–15. URL. https://dl.acm.org/doi/10.1145/2342509.2342513.

Casey, E., 2009. Handbook of Digital Forensics and Investigation. Elsevier Science. URL. https://books.google.no/books?id=xNjsDprqtUYC.

Chow, J., Pfaff, B., Garfinkel, T., Rosenblum, M., 2005. Shredding your garbage: reducing data lifetime through secure deallocation. In: 14th USENIX Security Symposium, pp. 331–346.

Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R., Jan, 2016. Chapter 4 - fog computing: principles, architectures, and applications. In: Buyya, R., Vahid Dastjerdi, A. (Eds.), Internet of Things. Morgan Kaufmann, pp. 61–75. URL. https://www.sciencedirect.com/science/article/pii/B9780128053959000046.

Forti, S., Lera, I., Guerrero, C., Brogi, A., 2021. Osmotic management of distributed complex systems: a declarative decentralised approach. n/a (n/a). J. Softw.: Evolut. Process, e2405. URL. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2405.

Guerrero, C., Lera, I., Juiz, C., 2018. On the influence of fog colonies partitioning in fog application Makespan. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 377–384.

Hagberg, A.A., Schult, D.A., Swart, P.J., 2008. Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (Eds.), Proceedings of the 7th Python in Science Conference. Pasadena, CA USA, pp. 11–15. URL. https://conference.scipy.org/proceedings/scipy2008/.

Hegarty, R.C., Lamb, D.J., Attwood, A., Attwood, A., Attwood, A., 2014. Digital evidence challenges in the internet of things. In: Dowland, P.S., Furnell, S.M., Ghita, B.V. (Eds.), Proceedings of the Tenth International Network Conference. INC 2014), Plymouth, pp. 163–172.

Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., Koldehofe, B., 2013. Mobile fog: a programming model for large–scale applications on the internet of things. In: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing - MCC '13, pp. 15–20. URL. http://dl.acm.org/citation.cfm?doid=2491266.2491270.

Huang, C., Lu, R., Choo, K.K.R., 2017. Vehicular fog computing: architecture, use case, and security and forensic challenges. IEEE Commun. Mag. 55 (11), 105–111.

Kitanov, S., Janevski, T., Nov, 2016. State of the art: fog computing for 5G networks. In: 2016 24th Telecommunications Forum (TELFOR), pp. 1–4.

Apr Lera, I., Guerrero, C., Juiz, C., 2018. Comparing centrality indices for network usage optimization of data placement policies in fog devices. In: 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, pp. 115–122. URL. https://ieeexplore.ieee.org/document/8364053/.

Apr Lera, I., Guerrero, C., Juiz, C., 2019a. Availability-aware service placement policy in fog computing based on graph partitions. IEEE Internet Things J. 6 (2), 3641–3651. URL. https://ieeexplore.ieee.org/document/8588297/.

Lera, I., Guerrero, C., Juiz, C., 2019b. YAFS: a simulator for IoT scenarios in fog computing. IEEE Access 7, 91745–91758.

Li, L., Guan, Q., Jin, L., Guo, M., 2019. Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system. IEEE Access 7, 9912–9925.

May Lin, C., Han, G., Qi, X., Guizani, M., Shu, L., 2020. A distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks. IEEE Trans. Veh. Technol. 69 (5), 5481–5493.

Minnaard, W., 2014. Out of sight, but not out of mind: traces of nearby devices' wireless transmissions in volatile memory. In: Proceedings of the Digital Forensic Research Conference, DFRWS 2014 EU 11, pp. S104–S111. https://doi.org/10.1016/j.diin.2014.03.013. URL.

Apr Phan, L.-A., Nguyen, D.-T., Lee, M., Park, D.-H., Kim, T., 2021. Dynamic fog-to-fog offloading in SDN-based fog computing systems. Future Generat. Comput. Syst. 117, 486–497. URL. https://www.sciencedirect.com/science/article/pii/S0167739X20330831.

Rafi, A., Adeel-ur-Rehman, Ali, G., Akram, J., Jan, 2019. Efficient energy utilization in fog computing based wireless sensor networks. In: 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), pp. 1–5.

Sabireen, H., Neelanarayanan, V., Jun, 2021. A review on fog computing: architecture, fog with IoT, algorithms and research challenges. ICT Express 7 (2), 162–176. URL. https://www.sciencedirect.com/science/article/pii/S2405959521000606.

Apr Sandvik, J.-P., Franke, K., Abie, H., Årnes, A., 2022. Quantifying data volatility for IoT forensics with examples from Contiki OS. Forensic Sci. Int.: Digit. Invest. 40, 301343. URL. https://www.sciencedirect.com/science/article/pii/S2666281722000129.

Sandvik, J.-P., Franke, K., Årnes, A., 2021. Towards a generic approach of quantifying evidence volatility in resource constrained devices. Ch. 2. In: Montasari, R., Jahankhani, H., Hill, R., Parkinson, S. (Eds.), Digital Forensic Investigation of Internet of Things (IoT) Devices, Advanced Sciences and Technologies for Security Applications. Advanced Sciences and Technologies for Security Applications. Springer International Publishing, Cham, pp. 21–45. https://doi.org/10.1007/978-3-030-60425-7_2. URL.

Singh, S., Chiu, Y.-C., Tsai, Y.-H., Yang, J.-S., 2016. Mobile edge fog computing in 5G era: architecture and implementation. In: 2016 International Computer Symposium (ICS), pp. 731–735.

Singh, S., Saxena, N., Roy, A., Kim, H., 2017. A survey on 5G network technologies from social perspective. IETE Tech. Rev. 34 (1), 30–39. https://doi.org/10.1080/02564602.2016.1141077. URL.

Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., Leitner, P., 2017. Optimized IoT service placement in the fog. Serv. Orient. Comput. Appl. 11 (4), 427–443. https://doi.org/10.1007/s11761-017-0219-8. URL.

Unwala, I., Taqvi, Z., Lu, J., 2018. Thread: an IoT Protocol. In: 2018 IEEE Green Technologies Conference (GreenTech), pp. 161–167.

Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R., 2012. RFC 6550: RPL: IPv6 routing Protocol for low-power and Lossy networks. Tech. rep., IETF.